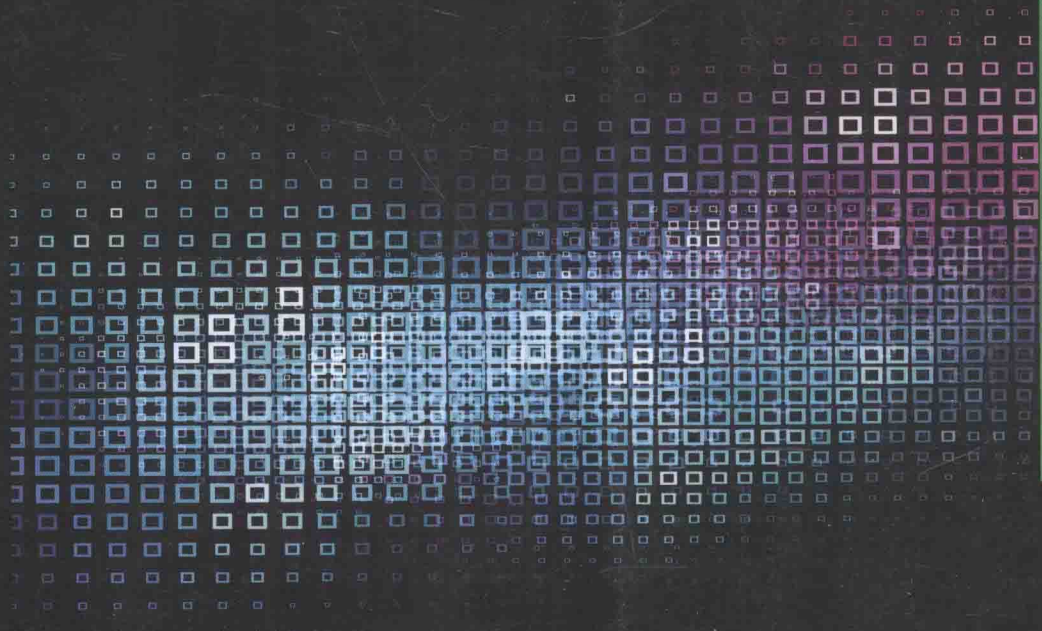
软件工程师开发大系

Java 开发实例大全（提高卷）

598个典型实例及源码分析，涵盖21个应用方向

✔工作应用速查 ✔项目开发参考 ✔学习实战练习

软件开发技术联盟◎编著



- **工作应用速查：**本书实例全面、系统，涉及程序开发的各个方面，适合各级程序开发人员速查速用。
- **项目开发参考：**程序员借助本书提供的实例源代码，可以快速搭建工程项目，提高开发效率。
- **学习实战练习：**入门者的实战训练大全，不但可以激发学习兴趣，更可提高编程实战能力和编程思维水平。

DVD

清华大学出版社

“软件工程师开发大系”简介

“软件工程师开发大系”是一套软件开发“实例类”图书。它分为7个方向，每个方向又分为“基础卷”和“提高卷”两册，每册约600个实例，两册共有约1200个实例，内容极为丰富，从基础学习到高级应用分门别类包罗万象。对每个实例按照实例说明、关键技术、设计过程（代码实现）、详尽注释、秘笈心法的顺序进行了详尽分析。大多数实例及源代码来自现实开发中，很多可以移植应用。配书光盘给出了实例完整的源代码，部分实例还给出了讲解视频。

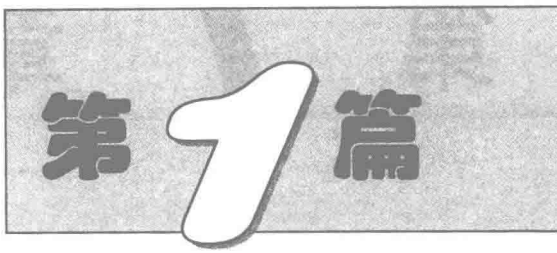
“软件工程师开发大系”丛书适合工作应用速查、项目开发参考、学习实战练习使用。“软件工程师开发大系”丛书具体品种如下：

- C#开发实例大全（基础卷）
- C#开发实例大全（提高卷）
- Java开发实例大全（基础卷）
- Java开发实例大全（提高卷）
- Java Web开发实例大全（基础卷）
- Java Web开发实例大全（提高卷）
- PHP开发实例大全（基础卷）
- PHP开发实例大全（提高卷）
- ASP.NET开发实例大全（基础卷）
- ASP.NET开发实例大全（提高卷）
- Visual C++开发实例大全（基础卷）
- Visual C++开发实例大全（提高卷）
- Visual Basic开发实例大全（基础卷）
- Visual Basic开发实例大全（提高卷）

清华大学出版社数字出版网站
WQBook 书文局泉
www.wqbook.com

ISBN 978-7-302-38439-7
9 787302 384397 >
定价：128.00元

（附光盘1张，含实例源代码、部分实例视频、实例配置说明等）



图形图像篇

第 1 章

Java 图形与文本

- 绘制图形和文本
- 笔画和图形处理
- 绘制图案
- 图形的合并运算

1.1　绘制图形和文本

实例 001	绘制直线	初级
	光盘位置：光盘\MR\001	趣味指数：★

实例说明

在几何中，直线是向两端无限延伸的，本实例所说的绘制直线，实际上是绘制直线上两点之间的线段，线段在实际生产和生活中经常使用。运行程序，将在窗体上绘制线段，效果如图 1.1 所示。

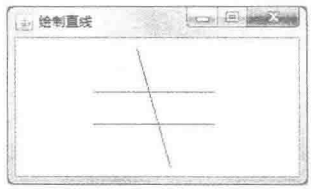


图 1.1　绘制直线

关键技术

本实例主要是通过在 JPanel 类的子类中，重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawLine()方法实现的。

（1）在 JPanel 类的子类中，重写 JComponent 类的 paint()方法，该方法的定义如下：

```
public void paint(Graphics g)
```

参数说明

g：图形上下文对象，用于绘制基本的形状和文本。

（2）使用 Graphics 类的 drawLine()方法绘制直线，该方法的定义如下：

```
public abstract void drawLine(int x1, int y1, int x2, int y2)
```

参数说明

❶ x1：第 1 个点的 x 坐标。

❷ y1：第 1 个点的 y 坐标。

❸ x2：第 2 个点的 x 坐标。

❹ y2：第 2 个点的 y 坐标。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawLineFrame 窗体类。

（3）在 DrawLineFrame 窗体类中创建内部面板类 DrawLinePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawLine()方法绘制直线。

（4）将内部面板类 DrawLinePanel 的实例添加到窗体类 DrawLineFrame 的内容面板上，用于在窗体上显示绘制的直线，代码如下：

```
class DrawLinePanel extends JPanel {                    //创建内部面板类
    public void paint(Graphics g) {                     //重写 paint()方法
        g.drawLine(70, 50, 180, 50);                    //绘制第 1 条水平线
        g.drawLine(70, 80, 180, 80);                    //绘制第 2 条水平线
```

```
            g.drawLine(110, 10, 140, 120);          //绘制斜线
        }
    }
```

秘笈心法

心法领悟 001：通过绘制直线可以实现画图板的功能。

在绘制直线时，如果两个端点间的距离很近，就相当于画了一个点，根据这个特点，可以在鼠标指针移动的路径上连续画点，完成各种图形的绘制，从而实现画图板的功能。

实例 002　绘制矩形

光盘位置：光盘\MR\002　　初级　　趣味指数：★★

实例说明

矩形在实际生产和生活中经常使用，例如书桌的桌面、房屋的门窗等，本实例将通过绘制矩形让读者初步了解 Java 绘图技术。运行程序，将在窗体上绘制矩形，效果如图 1.2 所示。

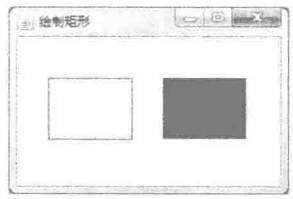


图 1.2　绘制矩形

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawRect()和 fillRect()方法来实现的。

（1）使用 Graphics 类的 drawRect()方法绘制的矩形，只有线条而没有填充色，该方法的定义如下：

```
public abstract void drawRect(int x, int y, int width, int height)
```

参数说明

❶ x：矩形左上角的 x 坐标。

❷ y：矩形左上角的 y 坐标。

❸ width：矩形的宽度。

❹ height：矩形的高度。

（2）使用 Graphics 类的 fillRect()方法可绘制带填充色的矩形，该方法的定义如下：

```
public abstract void fillRect(int x, int y, int width, int height)
```

参数说明

❶ x：填充矩形左上角的 x 坐标。

❷ y：填充矩形左上角的 y 坐标。

❸ width：填充矩形的宽度。

❹ height：填充矩形的高度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawRectangleFrame 窗体类。

（3）在 DrawRectangleFrame 窗体类中创建内部面板类 DrawRectanglePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawRect()和 fillRect()方法绘制矩形。

（4）将内部面板类 DrawRectanglePanel 的实例添加到窗体类 DrawRectangleFrame 的内容面板上，用于在窗体上显示绘制的矩形，代码如下：

```
class DrawRectanglePanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {                //重写 paint()方法
        g.drawRect(30, 40, 80, 60) ;               //绘制空心矩形
        g.fillRect(140, 40, 80, 60);               //绘制实心矩形
    }
}
```

秘笈心法

心法领悟 002：将矩形和直线组合，可以绘制军棋、象棋等的棋盘。

先绘制一个棋盘大小的矩形，然后在矩形内适当的位置绘制纵横交错的直线，可以完成军棋、象棋等棋盘的绘制。

实例 003　绘制正方形

光盘位置：光盘\MR\003　　初级　　趣味指数：★★

实例说明

本实例演示如何在 Java 中绘制正方形。运行程序，将在窗体上绘制正方形，效果如图 1.3 所示。

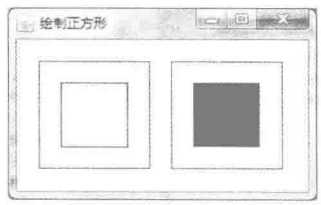


图 1.3　绘制正方形

关键技术

本实例也是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawRect()和 fillRect()方法来实现的。

使用 Graphics 类的 drawRect()和 fillRect()方法绘制矩形时，如果将这两个方法中表示宽度和高度的参数设置为相同的值，绘制出来的图形就是正方形。

例如：

```
g.drawRect( 30, 20, 120, 120 );                    //在点(30, 20)处绘制边长是 120 的正方形
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawSquareFrame 窗体类。

（3）在 DrawSquareFrame 窗体类中，创建内部面板类 DrawSquarePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawRect()和 fillRect()方法绘制正方形。

（4）将内部面板类 DrawSquarePanel 的实例添加到窗体类 DrawSquareFrame 的内容面板上，用于在窗体上显示绘制的正方形，代码如下：

```
class DrawSquarePanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {             //重写 paint()方法
        g.drawRect(20, 20, 100, 100);           //绘制空心正方形
        g.drawRect(40, 40, 60, 60);             //绘制空心正方形
        g.drawRect(140, 20, 100, 100);          //绘制空心正方形
        g.fillRect(160, 40, 60, 60) ;           //绘制实心正方形
    }
}
```

秘笈心法

心法领悟 003：使用 fillRect()方法绘制柱形图。

在实际项目中绘制柱形图表时，可以使用 fillRect()方法绘制柱形，这样就可以通过图形对数据进行分析，使数据信息看起来更加直观。

实例 004 绘制椭圆

光盘位置：光盘\MR\004

初级

趣味指数：★★

实例说明

本实例演示如何在 Java 中绘制椭圆。运行程序，将在窗体上绘制椭圆，效果如图 1.4 所示。

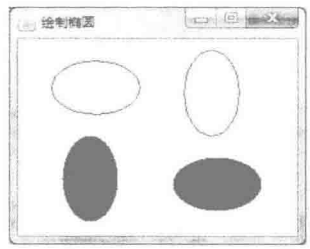


图 1.4　绘制椭圆

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawOval()和 fillOval()方法来实现的。

（1）使用 Graphics 类的 drawOval()方法绘制的椭圆，只有线条而没有填充色，该方法的定义如下：

```
public abstract void drawOval(int x, int y, int width, int height)
```

参数说明

❶ x：要绘制椭圆的左上角的 x 坐标。

❷ y：要绘制椭圆的左上角的 y 坐标。

❸ width：要绘制椭圆的宽度。

❹ height：要绘制椭圆的高度。

（2）使用 Graphics 类的 fillOval()方法可绘制带填充色的椭圆，该方法的定义如下：

```
public abstract void fillOval(int x, int y, int width, int height)
```

参数说明

❶ x：要填充椭圆的左上角的 x 坐标。

❷ y：要填充椭圆的左上角的 y 坐标。

❸ width：要填充椭圆的宽度。

❹ height：要填充椭圆的高度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawEllipseFrame 窗体类。

（3）在 DrawEllipseFrame 窗体类中，创建内部面板类 DrawEllipsePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawOval()和 fillOval()方法绘制椭圆。

（4）将内部面板类 DrawEllipsePanel 的实例添加到窗体类 DrawEllipseFrame 的内容面板上，用于在窗体上显示绘制的椭圆，代码如下：

```
class DrawEllipsePanel extends JPanel {                    //创建内部面板类
    public void paint(Graphics g) {                        //重写 paint()方法
        g.drawOval(30, 20, 80, 50);                        //绘制空心椭圆
        g.drawOval(150, 10, 50, 80);                       //绘制空心椭圆
        g.fillOval(40, 90, 50, 80);                        //绘制实心椭圆
        g.fillOval(140, 110, 80, 50);                      //绘制实心椭圆
    }
}
```

秘笈心法

心法领悟 004：通过 Graphics 类的 drawOval()和 fillOval()方法可以绘制圆形。

在绘制椭圆时，如果将 drawOval()或 fillOval()方法的后两个参数设置为相同的值，就可以绘制出圆形或填充的圆形。

实例 005　绘制圆弧

初级

趣味指数：★

实例说明

本实例演示如何在 Java 中绘制圆弧。运行程序，将在窗体上绘制圆弧，效果如图 1.5 所示。

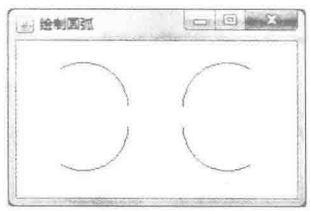


图 1.5　绘制圆弧

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawArc()方法来实现的。

drawArc()方法的定义如下：

```
public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
```

参数说明

❶ x：要绘制弧的左上角的 x 坐标。

❷ y：要绘制弧的左上角的 y 坐标。

❸ width：要绘制弧的宽度。

❹ height：要绘制弧的高度。

❺ startAngle：开始角度。

❻ arcAngle：相对于开始角度而言，弧跨越的角度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawArcFrame 窗体类。

（3）在 DrawArcFrame 窗体类中创建内部面板类 DrawArcPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawArc()方法绘制圆弧。

（4）将内部面板类 DrawArcPanel 的实例添加到窗体类 DrawArcFrame 的内容面板上，用于在窗体上显示绘制的圆弧，代码如下：

```
class DrawArcPanel extends JPanel {                    //创建内部面板类
    public void paint(Graphics g) {                    //重写 paint()方法
        g.drawArc(20, 20, 80, 80, 0, 120);             //绘制圆弧
        g.drawArc(20, 40, 80, 80, 0, -120);            //绘制圆弧
        g.drawArc(150, 20, 80, 80, 180, -120);         //绘制圆弧
        g.drawArc(150, 40, 80, 80, 180, 120);          //绘制圆弧
    }
}
```

秘笈心法

心法领悟 005：通过 drawArc()方法可以绘制扇形。

在实际开发中，如果需要绘制扇形，可以在使用 drawArc()方法绘制圆弧时，用 drawLine()方法从圆弧的两个端点向圆心画直线，这样就可以画出扇形。

实例 006 绘制指定角度的填充扇形

初级

光盘位置：光盘\MR\006

趣味指数：★★★

实例说明

本实例演示如何在 Java 中绘制指定角度的填充扇形。运行程序，将在窗体上绘制填充扇形，效果如图 1.6 所示。

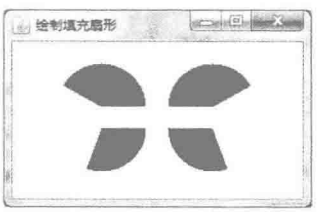


图 1.6　绘制指定角度的填充扇形

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 fillArc()方法来实现的。

fillArc()方法的定义如下：

```
public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)
```

参数说明

❶ x：要绘制填充扇形的左上角的 x 坐标。

❷ y：要绘制填充扇形的左上角的 y 坐标。
❸ width：要绘制填充扇形的宽度。
❹ height：要绘制填充扇形的高度。
❺ startAngle：开始角度。
❻ arcAngle：相对于开始角度而言，填充扇形的弧跨越的角度。

设计过程

（1）新建一个项目。
（2）在项目中创建一个继承 JFrame 类的 DrawSectorFrame 窗体类。
（3）在 DrawSectorFrame 窗体类中，创建内部面板类 DrawSectorPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 fillArc()方法绘制填充扇形。
（4）将内部面板类 DrawSectorPanel 的实例添加到窗体类 DrawSectorFrame 的内容面板上，用于在窗体上显示绘制的填充扇形，代码如下：

```
class DrawSectorPanel extends JPanel {                  //创建内部面板类
    public void paint(Graphics g) {                     //重写 paint()方法
        g.fillArc(40, 20, 80, 80, 0, 150);              //绘制填充扇形
        g.fillArc(140, 20, 80, 80, 180, -150);          //绘制填充扇形
        g.fillArc(40, 40, 80, 80, 0, -110);             //绘制填充扇形
        g.fillArc(140, 40, 80, 80, 180, 110);           //绘制填充扇形
    }
}
```

秘笈心法

心法领悟 006：使用 fillArc()方法绘制饼形图表。
在实际项目中，可以使用 fillArc()方法绘制饼形图表，这样就可以通过饼形图表对数据进行分析，从而可以更加直观地对数据信息进行分析。

实例 007　绘制多边形

初级

光盘位置：光盘\MR\007

趣味指数：★★★

实例说明

本实例演示如何在 Java 中绘制多边形。运行程序，将在窗体上绘制多边形，效果如图 1.7 所示。

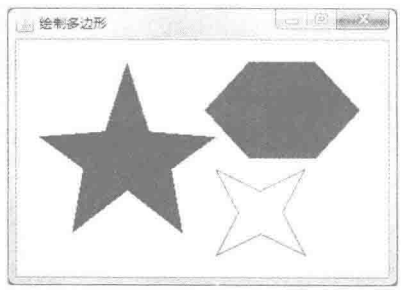


图 1.7　绘制多边形

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类

的 drawPolygon()和 fillPolygon()方法来实现的。

（1）使用 Graphics 类的 drawPolygon()方法绘制的多边形，只有线条而没有填充色，该方法的定义如下：

```
public abstract void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
```

参数说明

❶ xPoints：要绘制多边形的 x 坐标数组。

❷ yPoints：要绘制多边形的 y 坐标数组。

❸ nPoints：要绘制多边形的顶点总数。

（2）使用 Graphics 类的 fillPolygon()方法可绘制带填充色的多边形，该方法的定义如下：

```
public abstract void fillPolygon(int[] xPoints, int[] yPoints, int nPoints)
```

参数说明

❶ xPoints：要绘制填充多边形的 x 坐标数组。

❷ yPoints：要绘制填充多边形的 y 坐标数组。

❸ nPoints：要绘制填充多边形的顶点总数。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawPolygonFrame 窗体类。

（3）在 DrawPolygonFrame 窗体类中，创建内部面板类 DrawPolygonPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawPolygon()和 fillPolygon()方法绘制多边形。

（4）将内部面板类 DrawPolygonPanel 的实例添加到窗体类 DrawPolygonFrame 的内容面板上，用于在窗体上显示绘制的多边形，代码如下：

```
class DrawPolygonPanel extends JPanel {                          //创建内部面板类
    public void paint(Graphics g) {                              //重写 paint()方法
        int[] x1 = { 100,120,180,140,150,100,50,60,20,80 };      //多边形的横坐标
        int[] y1 = { 20,85,90,120,180,140,180,120,90,85 };       //多边形的纵坐标
        int n1 = 10;                                             //多边形的边数
        g.fillPolygon(x1, y1, n1);                               //绘制实心多边形
        int[] x2 = { 210, 270, 310, 270, 210, 170 };             //多边形的横坐标
        int[] y2 = { 20, 20, 65, 110, 110, 65 };                 //多边形的纵坐标
        int n2 = 6;                                              //多边形的边数
        g.fillPolygon(x2, y2, n2);                               //绘制实心多边形
        int[] x3 = { 180, 220, 260, 240, 260, 220, 180, 200 };   //多边形的横坐标
        int[] y3 = { 120, 140, 120, 160, 200, 180, 200, 160 };   //多边形的纵坐标
        int n3 = 8;                                              //多边形的边数
        g.drawPolygon(x3, y3, n3);                               //绘制空心多边形
    }
}
```

秘笈心法

心法领悟 007：快速定义多边形的顶点坐标。

由于绘制多边形需要各顶点的 x 坐标和 y 坐标数组，因此可以在草稿纸上把图形画出来，然后再根据图形定义坐标点。

实例 008 绘制二次曲线

光盘位置：光盘\MR\008

初级

趣味指数：★★

实例说明

本实例演示如何在 Java 中绘制二次曲线。运行程序，将在窗体上绘制二次曲线，效果如图 1.8 所示。

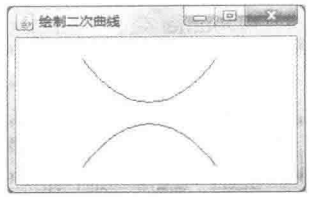


图 1.8　绘制二次曲线

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 draw()方法和使用 QuadCurve2D.Double 类创建二次曲线对象来实现的。

（1）使用 Graphics2D 类的 draw()方法，并将 QuadCurve2D.Double 类创建的二次曲线对象作为 draw()方法的参数，实现绘制二次曲线的操作，draw()方法的定义如下：

```
public abstract void draw(Shape shape)
```

参数说明

shape：要绘制的形状。

（2）使用 QuadCurve2D.Double 类创建二次曲线对象，其构造方法的定义如下：

```
public QuadCurve2D.Double(double x1, double y1, double ctrlx, double ctrly, double x2, double y2)
```

参数说明

❶ x1：起始点的 x 坐标。

❷ y1：起始点的 y 坐标。

❸ ctrlx：控制点的 x 坐标。

❹ ctrly：控制点的 y 坐标。

❺ x2：结束点的 x 坐标。

❻ y2：结束点的 y 坐标。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawQuadCurveFrame 窗体类。

（3）在 DrawQuadCurveFrame 窗体类中，创建内部面板类 DrawQuadCurvePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 QuadCurve2D.Double 类创建二次曲线对象，并使用 Graphics2D 类的 draw()方法绘制该二次曲线。

（4）将内部面板类 DrawQuadCurvePanel 的实例添加到窗体类 DrawQuadCurveFrame 的内容面板上，用于在窗体上显示绘制的二次曲线，代码如下：

```
class DrawQuadCurvePanel extends JPanel {                    //创建内部面板类
    public void paint(Graphics g) {                          //重写 paint()方法
        Graphics2D g2=(Graphics2D)g;                         //获得 Graphics2D 对象
        //创建二次曲线，其中点(120,100)是控制点，点(60,20)是起始点坐标，点(180,20)是终点坐标
        QuadCurve2D.Double quadCurve1 = new QuadCurve2D.Double(60,20,120,100,180,20);
        g2.draw(quadCurve1);                                 //绘制二次曲线
        //创建二次曲线，其中点(120,40)是控制点，点(60,120)是起始点坐标，点(180,120)是终点坐标
        QuadCurve2D.Double quadCurve2 = new QuadCurve2D.Double(60,120,120,40,180,120);
        g2.draw(quadCurve2);                                 //绘制二次曲线
    }
}
```

秘笈心法

心法领悟 008：绘制二次曲线可以更节省内存空间。

绘制二次曲线时，可以使用 QuadCurvel2D.Double 类和 QuadCurvel2D.Float 类创建二次曲线，其中，使用 QuadCurvel2D.Float 类创建更节省内存空间。

实例 009 绘制三次曲线

光盘位置：光盘\MR\009　　初级　　趣味指数：★★

实例说明

本实例演示如何在 Java 中绘制三次曲线。运行程序，将在窗体上绘制三次曲线，效果如图 1.9 所示。

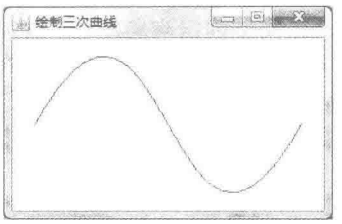


图 1.9　绘制三次曲线

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 draw()方法和使用 CubicCurvel2D.Double 类创建三次曲线对象来实现的。

使用 CubicCurvel2D.Double 类创建三次曲线对象，其构造方法的定义如下：

```
public CubicCurve2D.Double(double x1, double y1, double ctrlx1, double ctrly1, double ctrlx2, double ctrly2, double x2, double y2)
```

参数说明

❶ x1：起始点的 x 坐标。
❷ y1：起始点的 y 坐标。
❸ ctrlx1：第 1 个控制点的 x 坐标。
❹ ctrly1：第 1 个控制点的 y 坐标。
❺ ctrlx2：第 2 个控制点的 x 坐标。
❻ ctrly2：第 2 个控制点的 y 坐标。
❼ x2：结束点的 x 坐标。
❽ y2：结束点的 y 坐标。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawCubicCurveFrame 窗体类。

（3）在 DrawCubicCurveFrame 窗体类中，创建内部面板类 DrawCubicCurvePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 CubicCurvel2D.Double 类创建三次曲线对象，并使用 Graphics2D 类的 draw()方法绘制该三次曲线。

（4）将内部面板类 DrawCubicCurvePanel 的实例添加到窗体类 DrawCubicCurveFrame 的内容面板上，用于在窗体上显示绘制的三次曲线，代码如下：

```
class DrawCubicCurvePanel extends JPanel {    //创建内部面板类
    public void paint(Graphics g) {           //重写 paint()方法
        Graphics2D g2=(Graphics2D)g;          //获得 Graphics2D 对象
        //创建三次曲线，其中点(140,-140)和点(140,300)是控制点，点(20,80)是起始点坐标，点(260,80)是终点坐标
```

```
        CubicCurve2D.Double cubicCurve = new CubicCurve2D.Double(20,80,140,-140,140,300,260,80);
        g2.draw(cubicCurve);                    //绘制三次曲线
    }
}
```

秘笈心法

心法领悟 009：绘制三次曲线可以更节省内存空间。

绘制三次曲线时，可以使用 CubicCurve2D.Double 类和 CubicCurve2D.Float 类创建三次曲线，其中，使用 CubicCurve2D.Float 类创建更节省内存空间。

实例 010　绘制文本

光盘位置：光盘\MR\010

初级

趣味指数：★★★

实例说明

本实例演示如何在 Java 中绘制文本。运行程序，将在窗体上绘制文本，效果如图 1.10 所示。

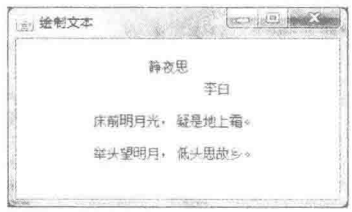


图 1.10　绘制文本

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawString()方法来实现的。

drawString()方法的定义如下：

```
public abstract void drawString(String str, int x, int y)
```

参数说明

❶ str：绘制的文本内容。

❷ x：绘制点的 x 坐标。

❸ y：绘制点的 y 坐标。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawTextStringFrame 窗体类。

（3）在 DrawTextStringFrame 窗体类中创建内部面板类 DrawTextStringPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawString()方法绘制文本。

（4）将内部面板类 DrawTextStringPanel 的实例添加到窗体类 DrawTextStringFrame 的内容面板上，用于在窗体上显示绘制的文本，代码如下：

```
class DrawTextStringPanel extends JPanel {      //创建内部面板类
    public void paint(Graphics g) {             //重写 paint()方法
        String value = "静夜思";
        int x = 120;                            //文本位置的横坐标
        int y = 30;                             //文本位置的纵坐标
```

```
            g.drawString(value, x, y);          //绘制文本
            //省略部分代码
        }
    }
```

秘笈心法

心法领悟 010：水印文字的实现。

对于一些有可能侵权的图片，或者要作为宣传的图片，可以通过使用 drawString()方法将用到的文字绘制到图片上，从而得到“水印文字”。

实例 011 设置文本的字体

光盘位置：光盘\MR\011　　初级　　趣味指数：★★

实例说明

本实例演示在 Java 中绘制文本时如何设置文本的字体，其中包括字体名称、大小和样式。运行程序，效果如图 1.11 所示。

图 1.11　设置文本字体的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 setFont()方法和使用 Font 类创建字体对象来实现的。

（1）使用 Graphics 类的 setFont()方法，并将 Font 类创建的字体对象作为 setFont()方法的参数，实现为文本设置字体的操作，setFont()方法的定义如下：

```
public abstract void setFont(Font font)
```

参数说明

font：为文本设置的字体对象。

（2）使用 Font 类创建字体对象，其构造方法的定义如下：

```
public Font(String name, int style, int size)
```

参数说明

❶ name：字体的名称。

❷ style：字体的样式。

❸ size：字体的大小。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 TextFontFrame 窗体类。

（3）在 TextFontFrame 窗体类中，创建内部面板类 ChangeTextFontPanel，并重写 JComponent 类的 paint()

方法，在该方法中使用 Font 类创建字体对象，并使用 Graphics 类的 setFont()方法设置文本的字体。

（4）将内部面板类 ChangeTextFontPanel 的实例添加到窗体类 TextFontFrame 的内容面板上，用于在窗体上显示指定字体后的文本，代码如下：

```
class ChangeTextFontPanel extends JPanel {                          //创建内部面板类
    public void paint(Graphics g) {                                 //重写 paint()方法
        String value = "明日编程词典社区";
        int x = 40;                                                 //文本位置的横坐标
        int y = 50;                                                 //文本位置的纵坐标
        Font font = new Font("华文行楷", Font.BOLD + Font.ITALIC, 26);  //创建字体对象
        g.setFont(font);                                            //设置字体
        g.drawString(value, x, y);                                  //绘制文本
        value = "http://community.mrbccd.com";
        x = 10;                                                     //文本位置的横坐标
        y = 100;                                                    //文本位置的纵坐标
        font = new Font("宋体", Font.BOLD, 20);                     //创建字体对象
        g.setFont(font);                                            //设置字体
        g.drawString(value, x, y);                                  //绘制文本
    }
}
```

秘笈心法

心法领悟 011：单一字体样式及字体样式的组合。

在绘制文本的同时，经常需要设置文本的字体样式，以达到醒目的效果。字体样式包括粗体样式 Font.BOLD、斜体样式 Font.ITALIC 和普通样式 Font.PLAIN，这些字体样式可以单独设置，也可以组合使用。在组合使用时，需要用连接符“+”进行连接，例如粗斜体样式为 Font.BOLD+Font.ITALIC。

实例 012　设置文本和图形的颜色

初级

光盘位置：光盘\MR\012

趣味指数：★★★★

实例说明

本实例演示在 Java 中绘制文本和图形时如何设置文本和图形的颜色。运行程序，效果如图 1.12 所示。

图 1.12　设置文本和图形颜色的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 setColor()方法和使用 Color 类创建颜色对象来实现的。

（1）使用 Graphics 类的 setColor()方法，并将 Color 类创建的颜色对象作为 setColor()方法的参数，实现为文本和图形设置颜色的操作，setColor()方法的定义如下：

```
public abstract void setColor(Color color)
```

参数说明

color：为文本或图形设置的颜色对象。

（2）使用 Color 类创建颜色对象，其构造方法的定义如下：

```
public Color(int r, int g, int b)
```

参数说明

❶ r：RGB 颜色的 R 值。

❷ g：RGB 颜色的 G 值。

❸ b：RGB 颜色的 B 值。

提示：Color 类提供了多个重载的构造方法，用户可以根据需要进行选择。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 TextAndShapeColorFrame 窗体类。

（3）在 TextAndShapeColorFrame 窗体类中创建内部面板类 TextAndShapeColorPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Color 类创建颜色对象，并使用 Graphics 类的 setColor()方法设置文本和图形的颜色。

（4）将内部面板类 TextAndShapeColorPanel 的实例添加到窗体类 TextAndShapeColorFrame 的内容面板上，用于在窗体上显示设置颜色后的文本和图形，代码如下：

```
class TextAndShapeColorPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {                    //重写 paint()方法
        String value = "只要努力————";
        int x = 60;                                    //文本位置的横坐标
        int y = 60;                                    //文本位置的纵坐标
        Color color = new Color(255,0,0);              //创建颜色对象
        g.setColor(color);                             //设置颜色
        g.drawString(value, x, y);                     //绘制文本
        value = "一切皆有可能";
        x = 140;                                       //文本位置的横坐标
        y = 100;                                       //文本位置的纵坐标
        color = new Color(0,0,255);                    //创建颜色对象
        g.setColor(color);                             //设置颜色
        g.drawString(value, x, y);                     //绘制文本
        color = Color.ORANGE;                          //通过 Color 类的字段获得颜色对象
        g.setColor(color);                             //设置颜色
        g.drawRoundRect(40,30,200,100,40,30);          //绘制圆角矩形
        g.drawRoundRect(45,35,190,90,36,26);           //绘制圆角矩形
    }
}
```

秘笈心法

心法领悟 012：使用 Color 类的字段获得颜色。

在绘制文本和图形时，除了使用 Color 类的构造方法创建颜色对象外，还可以使用 Color 类提供的字段获得颜色对象，如红色为 Color.RED 或 Color.red。

1.2 笔画和图形处理

实例 013 设置笔画的粗细　　初级

光盘位置：光盘\MR\013　　趣味指数：★★★

实例说明

本实例演示在 Java 中绘制图形时，如何设置笔画的粗细。运行程序，效果如图 1.13 所示。

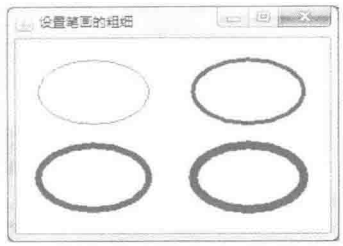


图 1.13　设置笔画粗细的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 setStroke()方法和使用 BasicStroke 类创建笔画对象来实现的。

（1）使用 Graphics2D 类的 setStroke()方法，并将 BasicStroke 类创建的笔画对象作为 setStroke()方法的参数，实现改变笔画粗细的操作，setStroke()方法的定义如下：

```
public abstract void setStroke(Stroke stroke)
```

参数说明

stroke：为图形设置的笔画对象。

（2）使用 BasicStroke 类创建笔画对象，其构造方法的定义如下：

```
public BasicStroke(float width)
```

参数说明

width：笔画的宽度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 StrokeWidthFrame 窗体类。

（3）在 StrokeWidthFrame 窗体类中，创建内部面板类 ChangeStrokeWidthPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 BasicStroke 类创建笔画对象，并使用 Graphics2D 类的 setStroke()方法设置笔画的粗细。

（4）将内部面板类 ChangeStrokeWidthPanel 的实例添加到窗体类 StrokeWidthFrame 的内容面板上，用于在窗体上显示设置笔画粗细后的图形，代码如下：

```
class ChangeStrokeWidthPanel extends JPanel {                    //创建内部面板类
    public void paint(Graphics g) {                              //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                           //获得 Graphics2D 对象
        BasicStroke stroke = new BasicStroke(1);                 //创建宽度是 1 的笔画对象
        g2.setStroke(stroke);                                    //设置笔画对象
        Ellipse2D.Float ellipse = new Ellipse2D.Float(20,20,100,60);  //创建椭圆对象
        g2.draw(ellipse);                                        //绘制椭圆
        stroke = new BasicStroke(4);                             //创建宽度是 4 的笔画对象
        g2.setStroke(stroke);                                    //设置笔画对象
        ellipse = new Ellipse2D.Float(160,20,100,60);            //创建椭圆对象
        g2.draw(ellipse);                                        //绘制椭圆
        stroke = new BasicStroke(6);                             //创建宽度是 6 的笔画对象
        g2.setStroke(stroke);                                    //设置笔画对象
        ellipse = new Ellipse2D.Float(20,100,100,60);            //创建椭圆对象
        g2.draw(ellipse);                                        //绘制椭圆
        stroke = new BasicStroke(8);                             //创建宽度是 8 的笔画对象
        g2.setStroke(stroke);                                    //设置笔画对象
        ellipse = new Ellipse2D.Float(160,100,100,60);           //创建椭圆对象
        g2.draw(ellipse);                                        //绘制椭圆
    }
}
```

秘笈心法

心法领悟 013：保留当前笔画对象。

在绘制图形时，当笔画对象改变以后，为了还能够使用原来的笔画对象，可以在改变之前使用 Graphics2D 类的 getStroke()方法获得当前的笔画对象，在需要时再使用 Graphics2D 类的 setStroke()方法进行设置。

实例 014 设置笔画样式

初级

光盘位置：光盘\MR\014

趣味指数：★★★

实例说明

本实例演示在 Java 中绘制图形时，如何设置笔画的样式。运行程序，效果如图 1.14 所示。

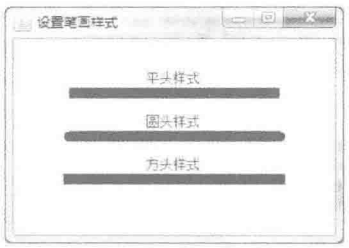


图 1.14 设置笔画样式的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 setStroke()方法和使用 BasicStroke 类创建笔画对象并指定笔画样式来实现的。

使用 BasicStroke 类重载的构造方法创建笔画对象，并指定笔画样式，该构造方法的定义如下：

```
public BasicStroke(float width, int cap, int join )
```

参数说明

❶ width：笔画的宽度。

❷ cap：笔画的样式。

❸ join：笔画的连接方式。

说明：该构造方法的第 2 个参数 cap 是用来指定笔画样式的，可以使用 BasicStroke 类的字段进行设置。笔画样式主要包括平头样式 BasicStroke.CAP_BUTT、圆头样式 BasicStroke.CAP_ROUND 和方头样式 BasicStroke.CAP_SQUARE。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 StrokeStyleFrame 窗体类。

（3）在 StrokeStyleFrame 窗体类中，创建内部面板类 ChangeStrokeStylePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 BasicStroke 类创建笔画对象并指定笔画样式，然后使用 Graphics2D 类的 setStroke()方法设置笔画对象。

（4）将内部面板类 ChangeStrokeStylePanel 的实例添加到窗体类 StrokeStyleFrame 的内容面板上，用于在窗体上显示设置笔画样式后的直线，代码如下：

```
class ChangeStrokeStylePanel extends JPanel {                                    //创建内部面板类
    public void paint(Graphics g) {                                              //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                                           //获得 Graphics2D 对象
        BasicStroke stroke = new BasicStroke(10,BasicStroke.CAP_BUTT,BasicStroke.JOIN_BEVEL);  //创建宽度是 10 的平头笔画对象
        g2.setStroke(stroke);                                                    //设置笔画对象
        Line2D.Float line = new Line2D.Float(50,50,240,50);                      //创建直线对象
        g2.drawString("平头样式", 120, 40);                                      //绘制文本
        g2.draw(line);                                                           //绘制直线
        stroke = new BasicStroke(10,BasicStroke.CAP_ROUND,BasicStroke.JOIN_BEVEL);  //创建宽度是 10 的圆头笔画对象
        g2.setStroke(stroke);                                                    //设置笔画对象
        line = new Line2D.Float(50,90,240,90);                                   //创建直线对象
        g2.drawString("圆头样式", 120, 80);                                      //绘制文本
        g2.draw(line);                                                           //绘制直线
        stroke = new BasicStroke(10,BasicStroke.CAP_SQUARE,BasicStroke.JOIN_BEVEL);  //创建宽度是 10 的方头笔画对象
        g2.setStroke(stroke);                                                    //设置笔画对象
        line = new Line2D.Float(50,130,240,130);                                 //创建直线对象
        g2.drawString("方头样式", 120, 120);                                     //绘制文本
        g2.draw(line);                                                           //绘制直线
    }
}
```

秘笈心法

心法领悟 014：何时使用笔画样式。

在绘制图形时，由于笔画样式并不能在闭合图形中体现出来，如在椭圆中就无法体现，因此，笔画样式都应用于非闭合的图形，例如直线、圆弧等。

实例 015　设置连接方式

光盘位置：光盘\MR\015　　中级　　趣味指数：★★★

实例说明

本实例演示在 Java 中绘制图形时如何设置笔画连接处的连接方式。运行程序，效果如图 1.15 所示。

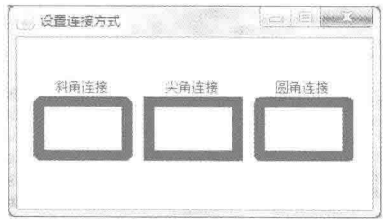


图 1.15　设置笔画连接方式的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 setStroke()方法和使用 BasicStroke 类创建笔画对象并指定笔画连接方式来实现的。

使用 BasicStroke 类重载的构造方法创建笔画对象，并指定笔画的连接方式，该构造方法的定义如下：

```
public BasicStroke(float width, int cap, int join )
```

参数说明

❶ width：笔画的宽度。

❷ cap：笔画的样式。

❸ join：笔画的连接方式。

说明：该构造方法的最后一个参数 join 是用来指定笔画连接方式的，可以使用 BasicStroke 类的字段进行设置。笔画连接方式主要包括斜角连接 BasicStroke.JOIN_BEVEL、尖角连接 BasicStroke.JOIN_MITER 和圆角连接 BasicStroke.JOIN_ROUND。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 StrokeJoinFrame 窗体类。

（3）在 StrokeJoinFrame 窗体类中，创建内部面板类 ChangeStrokeJoinPanel，并重写 JComponent 类的 paint() 方法，在该方法中使用 BasicStroke 类创建笔画对象并指定笔画的连接方式，然后使用 Graphics2D 类的 setStroke() 方法设置笔画对象。

（4）将内部面板类 ChangeStrokeJoinPanel 的实例添加到窗体类 StrokeJoinFrame 的内容面板上，用于在窗体上显示设置笔画连接方式后的图形，代码如下：

```
class ChangeStrokeJoinPanel extends JPanel {                                    //创建内部面板类
    public void paint(Graphics g) {                                             //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                                          //获得 Graphics2D 对象
        BasicStroke stroke = new BasicStroke(10,BasicStroke.CAP_BUTT,BasicStroke.JOIN_BEVEL);  //创建宽度是 10 的斜角连接笔画对象
        g2.setStroke(stroke);                                                   //设置笔画对象
        Rectangle2D.Float rect = new Rectangle2D.Float(20,60,80,50);            //创建矩形对象
        g2.drawString("斜角连接", 35, 50);                                      //绘制文本
        g2.draw(rect);                                                          //绘制矩形
        stroke = new BasicStroke(10,BasicStroke.CAP_BUTT,BasicStroke.JOIN_MITER);  //创建宽度是 10 的尖角连接笔画对象
        g2.setStroke(stroke);                                                   //设置笔画对象
        rect = new Rectangle2D.Float(120,60,80,50);                             //创建矩形对象
        g2.drawString("尖角连接", 135, 50);                                     //绘制文本
        g2.draw(rect);                                                          //绘制矩形
        stroke = new BasicStroke(10,BasicStroke.CAP_BUTT,BasicStroke.JOIN_ROUND);  //创建宽度是 10 的圆角连接笔画对象
        g2.setStroke(stroke);                                                   //设置笔画对象
        rect = new Rectangle2D.Float(220,60,80,50);                             //创建矩形对象
        g2.drawString("圆角连接", 235, 50);                                     //绘制文本
        g2.draw(rect);                                                          //绘制矩形
    }
}
```

秘笈心法

心法领悟 015：何时使用笔画连接方式。

在绘制图形时，由于笔画连接方式只能在线的连接处体现出来，因此，只能在具有连接点的图形拐角处体现出来，而不能在单一的线条上体现。

实例 016　设置虚线模式

光盘位置：光盘\MR\016　　中级　　趣味指数：★★★

实例说明

本实例演示在 Java 中绘制图形时如何将笔画设置为虚线模式。运行程序，效果如图 1.16 所示。

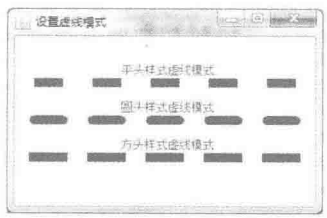


图 1.16　设置笔画为虚线模式的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 setStroke()方法和使用 BasicStroke 类创建笔画对象，并指定笔画为虚线模式来实现的。

使用 BasicStroke 类重载的构造方法创建笔画对象，并指定笔画为虚线模式，该构造方法的定义如下：

```
public BasicStroke(float width, int cap, int join, float miterlimit, float[] dash, float dash_phase)
```

参数说明

❶ width：笔画的宽度。

❷ cap：笔画的样式。

❸ join：笔画的连接方式。

❹ miterlimit：斜接处的剪裁限制，该值必须大于或等于 1.0f。

❺ dash：表示虚线模式的数组。

❻ dash_phase：开始虚线模式的偏移量。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DashedModelFrame 窗体类。

（3）在 DashedModelFrame 窗体类中，创建内部面板类 DashedModelPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 BasicStroke 类创建笔画对象并指定笔画为虚线模式，然后使用 Graphics2D 类的 setStroke()方法设置笔画对象。

（4）将内部面板类 DashedModelPanel 的实例添加到窗体类 DashedModelFrame 的内容面板上，用于在窗体上显示设置笔画为虚线模式后的图形，代码如下：

```
class DashedModelPanel extends JPanel {                                                          //创建内部面板类
    public void paint(Graphics g) {                                                              //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                                                           //获得 Graphics2D 对象
        float[] arr = {30.0f,30.0f};                                                             //创建虚线模式的数组
        BasicStroke stroke = new BasicStroke(10,BasicStroke.CAP_BUTT,BasicStroke.JOIN_BEVEL,1.0f,arr,0);  //创建平头虚线笔画对象
        g2.setStroke(stroke);                                                                    //设置笔画对象
        Line2D.Float line = new Line2D.Float(20,50,300,50);                                      //创建直线对象
        g2.drawString("平头样式虚线模式", 110, 40);                                               //绘制文本
        g2.draw(line);                                                                           //绘制直线
        stroke = new BasicStroke(10,BasicStroke.CAP_ROUND,BasicStroke.JOIN_BEVEL,1.0f,arr,0);    //创建宽度是 10 的圆头虚线笔画对象
        g2.setStroke(stroke);                                                                    //设置笔画对象
        line = new Line2D.Float(20,90,300,90);                                                   //创建直线对象
        g2.drawString("圆头样式虚线模式", 110, 80);                                               //绘制文本
        g2.draw(line);                                                                           //绘制直线
        stroke = new BasicStroke(10,BasicStroke.CAP_SQUARE,BasicStroke.JOIN_BEVEL,1.0f,arr,0);   //创建宽度是 10 的方头虚线笔画对象
        g2.setStroke(stroke);                                                                    //设置笔画对象
        line = new Line2D.Float(20,130,300,130);                                                 //创建直线对象
        g2.drawString("方头样式虚线模式", 110, 120);                                              //绘制文本
        g2.draw(line);                                                                           //绘制直线
    }
}
```

秘笈心法

心法领悟 016：设置虚线模式的关键。

在绘制图形时，如果需要设置虚线模式，关键是 BasicStroke 类的构造方法中表示虚线模式数组值的定义，不同的数组值会显示不同的虚线效果。

实例 017 缩放图形

光盘位置：光盘\MR\017　　中级　　趣味指数：★★★

实例说明

本实例演示在 Java 中绘制图形时如何对图形进行缩放，包括对图形进行放大、缩小和还原等操作。运行程序，效果如图 1.17 所示，用户可以通过单击窗体中的“放大”、“缩小”和“还原”按钮，对窗体上的图形进行相应操作。

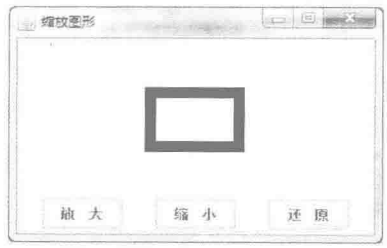


图 1.17　缩放图形

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 scale()方法来实现的。

使用 Graphics2D 类的 scale()方法可以实现图形的缩放，该方法的定义如下：

```
public abstract void scale(double sx, double sy)
```

参数说明

❶ sx：与原图形 x 坐标值相乘的量，如果 sx 大于 1.0，则在 X 坐标轴上放大原图形；如果 sx 小于 1.0，则在 X 坐标轴上缩小原图形。

❷ sy：与原图形 y 坐标值相乘的量，如果 sy 大于 1.0，则在 Y 坐标轴上放大原图形；如果 sy 小于 1.0，则在 Y 坐标轴上缩小原图形。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ZoomShapeFrame 窗体类。

（3）在 ZoomShapeFrame 窗体类中创建内部面板类 ZoomShapePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics2D 类的 scale()方法缩放图形。

（4）将内部面板类 ZoomShapePanel 的实例添加到窗体类 ZoomShapeFrame 的内容面板上，用于在窗体上显示缩放后的图形，代码如下：

```
class ZoomShapePanel extends JPanel {                                    //创建内部面板类
    public void paint(Graphics g) {                                      //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                                  //获得 Graphics2D 对象
        Rectangle2D.Float rect = new Rectangle2D.Float(120, 50, 80, 50);  //创建矩形对象
        BasicStroke stroke = new BasicStroke(10);                        //创建宽度是 10 的笔画对象
        g2.setStroke(stroke);                                            //设置笔画对象
        g2.clearRect(0, 0, 338, 220);                                    //清除原有内容
        if (flag == 0) {
            g2.draw(rect);                                               //绘制原矩形
        } else if (flag == 1) {
            g2.scale(1.3, 1.3);                                          //放大 1.3 倍
            g2.draw(rect);                                               //绘制矩形
```

```
        } else if (flag == 2) {
            g2.scale(0.5, 0.5);                    //缩小 0.5 倍
            g2.draw(rect);                         //绘制矩形
        }
    }
}
```

注意：上面代码中的 flag 是一个标记变量，当该变量的值为 0 时，图形显示为原来的大小；为 1 时，对图形进行放大；为 2 时，对图形进行缩小。

秘笈心法

心法领悟 017：复杂问题简单化。

在程序设计中，当某些操作相关联，而且实现起来比较困难时，可以通过定义标记变量将复杂问题简单化。方法是，使用 if 条件语句对标记变量的不同取值进行判断，以实现不同的操作，如本实例就使用了标记变量 flag，实现了对图形的缩放操作。

实例 018　旋转图形

光盘位置：光盘\MR\018　　中级　　趣味指数：★★★

实例说明

本实例演示在 Java 中绘制图形时，如何对图形进行旋转。运行程序，单击窗体上的“顺时针”按钮，可以将图形顺时针旋转，效果如图 1.18 所示，用户还可以通过单击“逆时针”和“还原”按钮，对窗体上的图形进行逆时针旋转和还原等操作。

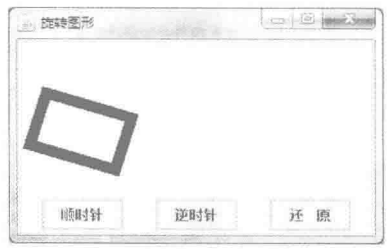


图 1.18　顺时针旋转图形的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 rotate()方法来实现的。

使用 Graphics2D 类的 rotate()方法，可以实现图形的旋转，该方法的定义如下：

public abstract void rotate(double theta, double x, double y)

参数说明

❶ theta：旋转的角度，以弧度为单位。

❷ x：旋转原点的 x 坐标。

❸ y：旋转原点的 y 坐标。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 RotateShapeFrame 窗体类。

（3）在 RotateShapeFrame 窗体类中，创建内部面板类 RotateShapePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics2D 类的 rotate()方法旋转图形。

（4）将内部面板类 RotateShapePanel 的实例添加到窗体类 RotateShapeFrame 的内容面板上，用于在窗体上显示旋转后的图形，代码如下：

```
class RotateShapePanel extends JPanel {                          //创建内部面板类
    public void paint(Graphics g) {                              //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                          //获得 Graphics2D 对象
        Rectangle2D.Float rect = new Rectangle2D.Float(40, 40, 80, 50); //创建矩形对象
        BasicStroke stroke = new BasicStroke(10);                //创建宽度是 10 的笔画对象
        g2.setStroke(stroke);                                    //设置笔画对象
        g2.clearRect(0, 0, 338, 220);                            //清除原有内容
        if (flag == 0) {
            g2.draw(rect);                                       //绘制原矩形
        } else if (flag == 1) {
            g2.rotate(rotateValue);                              //顺时针旋转
            g2.draw(rect);                                       //绘制矩形
        } else if (flag == 2) {
            g2.rotate(rotateValue);                              //逆时针旋转
            g2.draw(rect);                                       //绘制矩形
        }
    }
}
```

注意：上面代码中的 flag 是一个标记变量，当该变量的值为 0 时，图形还原到原来的位置；为 1 时，对图形进行顺时针旋转；为 2 时，对图形进行逆时针旋转。rotateValue 是与需要旋转角度对应的弧度值。

秘笈心法

心法领悟 018：角度转换为弧度。

由于使用 Graphics2D 类的 rotate()方法旋转图形时是按弧度进行旋转的，因此，当给定的旋转值是角度时，需要将其转换为弧度。在 Java 中，可以使用 Math 类的 toRadians(double angdeg)方法，将参数指定的角度 angdeg 转换为近似相等的弧度值。

实例 019　斜切图形

光盘位置：光盘\MR\019　　中级　　趣味指数：★★★

实例说明

本实例演示在 Java 中绘制图形时如何对图形进行斜切。运行程序，单击窗体上的“上斜切”按钮，可以实现对矩形进行向上斜切的操作，效果如图 1.19 所示，用户还可以通过单击窗体上的“下斜切”和“还原”按钮，对窗体上的图形进行向下斜切和还原等操作。

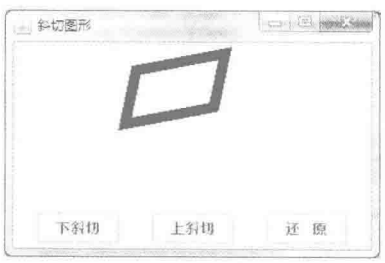


图 1.19　向上斜切图形的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 shear()方法来实现的。

使用 Graphics2D 类的 shear()方法可以实现图形的斜切，该方法的定义如下：

```
public abstract void shear(double shx, double shy)
```

参数说明

❶ shx：在正 X 轴方向移动坐标的乘数，可以作为相应 y 坐标的函数。

❷ shy：在正 Y 轴方向移动坐标的乘数，可以作为相应 x 坐标的函数。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ShearShapeFrame 窗体类。

（3）在 ShearShapeFrame 窗体类中创建内部面板类 ShearShapePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics2D 类的 shear()方法斜切图形。

（4）将内部面板类 ShearShapePanel 的实例添加到窗体类 ShearShapeFrame 的内容面板上，用于在窗体上显示斜切后的图形，代码如下：

```
class ShearShapePanel extends JPanel {                                  //创建内部面板类
    public void paint(Graphics g) {                                     //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                                 //获得 Graphics2D 对象
        Rectangle2D.Float rect = new Rectangle2D.Float(120, 50, 80, 50); //创建矩形对象
        BasicStroke stroke = new BasicStroke(10);                       //创建宽度是 10 的笔画对象
        g2.setStroke(stroke);                                           //设置笔画对象
        g2.clearRect(0, 0, 338, 230);                                   //清除原有内容
        if (flag == 0) {
            g2.draw(rect);                                              //绘制原矩形
        } else if (flag == 1) {
            g2.shear(0.2,0.2);                                          //向下斜切
            g2.draw(rect);                                              //绘制矩形
        } else if (flag == 2) {
            g2.shear(-0.2,-0.2);                                        //向上斜切
            g2.draw(rect);                                              //绘制矩形
        }
    }
}
```

注意：上面代码中的 flag 是一个标记变量，当该变量的值为 0 时，图形还原到原位置；为 1 时，对图形进行向下斜切；为 2 时，对图形进行向上斜切。

秘笈心法

心法领悟 019：任意调整斜切方向。

使用 Graphics2D 类的 shear(double shx, double shy)方法可以对图形进行斜切，用户通过调整该方法的两个参数值，就可以任意调整斜切方向。

实例 020　为图形填充渐变色

光盘位置：光盘\MR\020　　高级　　趣味指数：★★★★★

实例说明

本实例演示在 Java 中绘制图形时如何为图形填充渐变色。运行程序，效果如图 1.20 所示。

图 1.20　为图形填充渐变色的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 setPaint()方法和使用 GradientPaint 类创建封装渐变颜色的对象来实现的。

（1）使用 Graphics2D 类的 setPaint()方法，并将 GradientPaint 类创建的封装渐变颜色的对象作为 setPaint()方法的参数，实现为图形填充渐变色的操作，setPaint()方法的定义如下：

```
public abstract void setPaint(Paint paint)
```

参数说明

paint：封装了渐变颜色的 Paint 对象。

（2）使用 GradientPaint 类创建封装渐变颜色的对象，其构造方法的定义如下：

```
public GradientPaint(float x1, float y1, Color color1, float x2, float y2, Color color2, boolean cyclic)
```

参数说明

❶ x1：用户空间中第 1 个指定点的 x 坐标。
❷ y1：用户空间中第 1 个指定点的 y 坐标。
❸ color1：第 1 个指定点处的 Color 对象。
❹ x2：用户空间中第 2 个指定点的 x 坐标。
❺ y2：用户空间中第 2 个指定点的 y 坐标。
❻ color2：第 2 个指定点处的 Color 对象。
❼ cyclic：如果渐变模式在两种颜色之间重复循环，则该值设置为 true；否则设置为 false。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 FillGradientFrame 窗体类。

（3）在 FillGradientFrame 窗体类中，创建内部面板类 FillGradientPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics2D 类的 setPaint()方法设置封装了渐变色的对象，该对象是通过 GradientPaint 类创建的。

（4）将内部面板类 FillGradientPanel 的实例添加到窗体类 FillGradientFrame 的内容面板上，用于在窗体上显示填充了渐变颜色的图形，代码如下：

```
class FillGradientPanel extends JPanel {                                          //创建内部面板类
    public void paint(Graphics g) {                                               //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                                           //获得 Graphics2D 对象
        Rectangle2D.Float rect = new Rectangle2D.Float(20, 20, 280, 140);         //创建矩形对象
        GradientPaint paint = new GradientPaint(20,20,Color.BLUE,100,80,Color.RED,true);  //创建循环渐变的 GradientPaint 对象
        g2.setPaint(paint);                                                       //设置渐变
        g2.fill(rect);                                                            //绘制矩形
    }
}
```

秘笈心法

心法领悟 020：多颜色的线性渐变和径向渐变。

在实际应用中，用户还可以使用 LinearGradientPaint 类实现多颜色的线性渐变，或使用 RadialGradientPaint 类实现多颜色的径向渐变。

实例 021　平移坐标轴

光盘位置：光盘\MR\021　　初级　　趣味指数：★★

实例说明

本实例演示在 Java 中绘制图形时，如何实现坐标轴的平移。运行程序，将在窗体的左上角绘制矩形，单击窗体上的“平移坐标轴”按钮，将实现坐标轴的平移，效果如图 1.21 所示。

图 1.21　平移坐标轴

关键技术

本实例主要是通过在 JPanel 类的子类中，重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 translate()方法来实现的。

使用 Graphics2D 类的 translate()方法，可以实现坐标轴的平移，该方法的定义如下：

```
public abstract void translate(int x,int y)
```

参数说明

❶ x：平移到指定的 x 坐标处。

❷ y：平移到指定的 y 坐标处。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 TranslationAxisFrame 窗体类。

（3）在 TranslationAxisFrame 窗体类中，创建内部面板类 TranslationAxisPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics2D 类的 translate()方法，实现坐标轴的平移。

（4）将内部面板类 TranslationAxisPanel 的实例添加到窗体类 TranslationAxisFrame 的内容面板上，用于在窗体上显示平移坐标轴的效果，代码如下：

```
class TranslationAxisPanel extends JPanel {                          //创建内部面板类
    public void paint(Graphics g) {                                  //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                              //获得 Graphics2D 对象
        Rectangle2D.Float rect = new Rectangle2D.Float(10, 10, 80, 50);  //创建矩形对象
        BasicStroke stroke = new BasicStroke(10);                    //创建宽度是 10 的笔画对象
        g2.setStroke(stroke);                                        //设置笔画对象
        g2.clearRect(0, 0, 338, 230);                                //清除原有内容
        if (flag == 0) {
            g2.translate(0, 0);                                      //平移坐标轴
            g2.draw(rect);                                           //绘制矩形
        } else if (flag == 1) {
```

```
                g2.translate(120, 60);                    //平移坐标轴
                g2.draw(rect);                            //绘制矩形
            }
        }
    }
```

秘笈心法

心法领悟 021：绕指定点旋转图形。

为了实现绕指定点旋转图形，可以使用 Graphics2D 类的 translate()方法，将坐标轴移动到指定点，然后使用 Graphics2D 类的 rotate()方法，绕新的坐标原点旋转，从而实现绕指定点旋转图形的功能。

1.3 绘 制 图 案

实例 022 绘制五环图案　　高级

光盘位置：光盘\MR\022　　趣味指数：★★★★

实例说明

本实例演示奥林匹克运动会的会徽，即五环图案的绘制。运行程序，将在窗体上绘制五环图案，效果如图 1.22 所示。

图 1.22　五环图案

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 setStroke()、setColor()和 drawOval()方法来实现的。

（1）使用 Graphics2D 类的 setStroke()方法，指定笔画的粗细。

（2）使用 Graphics2D 类的 setColor()方法，指定颜色。

（3）使用 Graphics2D 类的 drawOval()方法，在指定位置绘制圆环，该方法是从 Graphics 类中继承的。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 FiveDaisyChainFrame 窗体类。

（3）在 FiveDaisyChainFrame 窗体类中，创建内部面板类 FiveDaisyChainPanel，并重写 JComponent 类的 paint()方法，在该方法中实现五环图案的绘制。

（4）将内部面板类 FiveDaisyChainPanel 的实例，添加到窗体类 FiveDaisyChainFrame 的内容面板上，用于在窗体上显示五环图案，代码如下：

```
class FiveDaisyChainPanel extends JPanel {                //创建内部面板类
```

```
    public void paint(Graphics g) {                          //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                       //获得 Graphics2D 对象
        BasicStroke stroke = new BasicStroke(3);             //创建宽度是 3 的笔画对象
        g2.setStroke(stroke);                                //设置笔画对象
        Color color = new Color(0,162,232);                  //创建颜色对象
        g2.setColor(color);                                  //设置颜色
        g2.drawOval(30, 40, 60, 60);                         //绘制第一个圆
        //省略了绘制其他圆的代码
    }
}
```

秘笈心法

心法领悟 022：获取五环图案的颜色。

在五环图案中，每种颜色都有特定的含义，为了获得五环图案的颜色，可以在 Photoshop 中按 F8 快捷键，在打开的信息面板中获得颜色的 RGB 值，然后使用 Color 类的构造方法创建颜色对象。

实例 023　绘制艺术图案

光盘位置：光盘\MR\023　　高级　　趣味指数：★★★★★

实例说明

本实例演示如何使用坐标轴平移、图形旋转和获得随机数等技术绘制艺术图案。运行程序，将在窗体上绘制艺术图案，效果如图 1.23 所示。

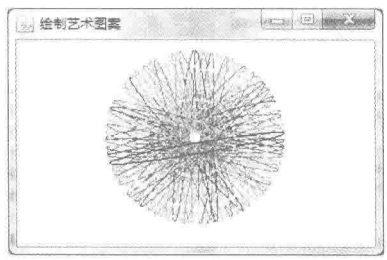


图 1.23　艺术图案

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 translate()、setColor()、rotate()和 draw()方法来实现的。

（1）使用 Graphics2D 类的 translate()方法，将坐标轴平移到指定点。

（2）使用 Graphics2D 类的 setColor()方法，设置颜色。

（3）使用 Graphics2D 类的 rotate()方法，旋转绘图上下文。

（4）使用 Graphics2D 类的 draw()方法，在指定位置绘制椭圆。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ArtDesignFrame 窗体类。

（3）在 ArtDesignFrame 窗体类中，创建内部面板类 ArtDesignPanel，并重写 JComponent 类的 paint()方法，在该方法中实现艺术图案的绘制。

（4）将内部面板类 ArtDesignPanel 的实例添加到窗体类 ArtDesignFrame 的内容面板上，用于在窗体上显示艺术图案，代码如下：

```
class ArtDesignPanel extends JPanel {                          //创建内部面板类
    public void paint(Graphics g) {                            //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                         //获得 Graphics2D 对象
        Ellipse2D.Float ellipse = new Ellipse2D.Float(-80, 5, 160, 10);  //创建椭圆对象
        Random random = new Random();                          //创建随机数对象
        g2.translate(160, 90);                                 //平移坐标轴
        int R = random.nextInt(256);                           //随机产生颜色的 R 值
        int G = random.nextInt(256);                           //随机产生颜色的 G 值
        int B = random.nextInt(256);                           //随机产生颜色的 B 值
        Color color = new Color(R,G,B);                        //创建颜色对象
        g2.setColor(color);                                    //指定颜色
        g2.draw(ellipse);                                      //绘制椭圆
        int i=0;
        while (i<100){
            R = random.nextInt(256);                           //随机产生颜色的 R 值
            G = random.nextInt(256);                           //随机产生颜色的 G 值
            B = random.nextInt(256);                           //随机产生颜色的 B 值
            color = new Color(R,G,B);                          //创建新的颜色对象
            g2.setColor(color);                                //指定颜色
            g2.rotate(10);                                     //旋转画布
            g2.draw(ellipse);                                  //绘制椭圆
            i++;
        }
    }
}
```

秘笈心法

心法领悟 023：随机获得颜色的 RGB 值。

使用 Random 类的实例生成伪随机数流，并使用该类的 nextInt(int n)方法产生一个 0（包含）~n（不包含）之间的随机整数，由于颜色的 RGB 值是 0~255 之间的整数值，所以为 nextInt(int n)方法的参数 n 传递 256，这样就可以随机产生一个 0~255 之间的整数，表示颜色 RGB 值。

实例 024 绘制花瓣

光盘位置：光盘\MR\024　　高级　　趣味指数：★★★★★

实例说明

本实例演示如何使用坐标轴平移和图形旋转等技术绘制花瓣。运行程序，将在窗体上绘制花瓣，效果如图 1.24 所示。

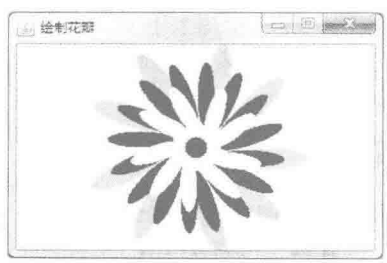


图 1.24　绘制花瓣

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 translate()、setColor()、rotate()和 fill()方法来实现的。

（1）使用 Graphics2D 类的 translate()方法，将坐标轴平移到指定点。

（2）使用 Graphics2D 类的 setColor()方法，设置颜色。

（3）使用 Graphics2D 类的 rotate()方法，旋转绘图上下文。

（4）使用 Graphics2D 类的 fill()方法，在指定位置绘制带填充色的椭圆。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawFlowerFrame 窗体类。

（3）在 DrawFlowerFrame 窗体类中创建内部面板类 DrawFlowerPanel，并重写 JComponent 类的 paint()方法，在该方法中实现花瓣的绘制。

（4）将内部面板类 DrawFlowerPanel 的实例添加到窗体类 DrawFlowerFrame 的内容面板上，用于在窗体上显示绘制的花瓣，代码如下：

```
class DrawFlowerPanel extends JPanel {                                  //创建内部面板类
    public void paint(Graphics g) {                                     //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                                  //获得 Graphics2D 对象
        g2.translate(drawFlowerPanel.getWidth() / 2, drawFlowerPanel.getHeight() / 2);  //平移坐标轴
        //绘制绿色花瓣
        Ellipse2D.Float ellipse = new Ellipse2D.Float(30, 0, 70, 20);   //创建椭圆对象
        Color color = new Color(0,255,0);                               //创建颜色对象
        g2.setColor(color);                                             //指定颜色
        g2.fill(ellipse);                                               //绘制椭圆
        int i=0;
        while (i<8){
            g2.rotate(30);                                              //旋转画布
            g2.fill(ellipse);                                           //绘制椭圆
            i++;
        }
        //绘制红色花瓣
        ellipse = new Ellipse2D.Float(20, 0, 60, 15);                   //创建椭圆对象
        color = new Color(255,0,0);                                     //创建颜色对象
        g2.setColor(color);                                             //指定颜色
        g2.fill(ellipse);                                               //绘制椭圆
        i=0;
        while (i<15){
            g2.rotate(75);                                              //旋转画布
            g2.fill(ellipse);                                           //绘制椭圆
            i++;
        }
        //绘制黄色花瓣
        ellipse = new Ellipse2D.Float(10, 0, 50, 15);                   //创建椭圆对象
        color = new Color(255,255,0);                                   //创建颜色对象
        g2.setColor(color);                                             //指定颜色
        g2.fill(ellipse);                                               //绘制椭圆
        i=0;
        while (i<8){
            g2.rotate(30);                                              //旋转画布
            g2.fill(ellipse);                                           //绘制椭圆
            i++;
        }
        //绘制红色中心点
        color = new Color(255, 0, 0);                                   //创建颜色对象
        g2.setColor(color);                                             //指定颜色
        ellipse = new Ellipse2D.Float(-10, -10, 20, 20);                //创建椭圆对象
```

```
            g2.fill(ellipse);                                    //绘制椭圆
        }
    }
```

秘笈心法

心法领悟 024：实现时钟的绘制。

利用图形旋转技术和坐标轴平移可以实现时钟的绘制，具体方法是，通过线程或定时器控件，在指定的时间间隔内绕坐标轴分别旋转表示秒针、分针和时针的图形或图像，从而达到时钟显示时间的效果。

实例 025　绘制公章

光盘位置：光盘\MR\025　　高级　　趣味指数：★★★★★

实例说明

本实例演示如何使用坐标轴平移、缩放、绘制椭圆、绘制多边形和绘制文本等技术实现公章的绘制。运行程序，将在窗体上显示绘制的公章，效果如图 1.25 所示。

图 1.25　绘制公章

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 translate()、setColor()、scale()、drawString()、fillPolygon()和 draw()等方法来实现的。

（1）使用 Graphics2D 类的 translate()方法将坐标轴平移到指定点。

（2）使用 Graphics2D 类的 setColor()方法设置颜色。

（3）使用 Graphics2D 类的 scale()方法对公章中的文本进行缩放。

（4）使用 Graphics2D 类的 drawString()方法绘制文本，该方法是从 Graphics 类继承的。

（5）使用 Graphics2D 类的 fillPolygon()方法绘制公章的五星，该方法也是从 Graphics 类继承的。

（6）使用 Graphics2D 类的 draw()方法绘制表示公章的圆。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawCachetFrame 窗体类。

（3）在 DrawCachetFrame 窗体类中创建内部面板类 DrawCachetPanel，并重写 JComponent 类的 paint()方法，在该方法中实现公章的绘制。

（4）将内部面板类 DrawCachetPanel 的实例添加到窗体类 DrawCachetFrame 的内容面板上，用于在窗体上显示绘制的公章，代码如下：

```
class DrawCachetPanel extends JPanel {                                  //创建内部面板类
    public void paint(Graphics g) {                                     //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                                 //获得 Graphics2D 对象
        g2.translate(170, 100);                                         //平移坐标轴
        BasicStroke stroke = new BasicStroke(6);                        //创建宽度是 6 的笔画对象
        g2.setStroke(stroke);                                           //设置笔画对象
        //绘制圆
        Ellipse2D.Float ellipse = new Ellipse2D.Float(-80, -80, 160, 160);  //创建圆对象
        Color color = new Color(255, 0, 0);                             //创建颜色对象
        g2.setColor(color);                                             //指定颜色
        g2.draw(ellipse);                                               //绘制圆
        //绘制五星
        int[] x1 = { 0, 8, 30, 16, 25, 0, -25, -16, -30, -8 };          //多边形的横坐标
        int[] y1 = { -35, -10, -15, 5, 28, 10, 28, 5, -15, -10 };       //多边形的纵坐标
        int n1 = 10;                                                    //多边形的边数
        g2.fillPolygon(x1, y1, n1);                                     //绘制多边形
        //绘制文本
        g2.scale(1.8, 1.8);                                             //放大
        Font font = new Font("宋体", Font.BOLD, 12);                    //创建字体
        g2.setFont(font);                                               //设置字体
        g2.drawString("专 用 章", -25, 30);                             //绘制文本
        int width = getWidth();                                         //获得面板宽度
        int height = getHeight();                                       //获得面板高度
        char[] array = "  明日科技有限公司              ".toCharArray();  //把字符串转换为字符数组
        int len = array.length * 2;                                     //定义半径
        font = new Font("宋体", Font.BOLD, 10);                         //创建新字体
        g2.setFont(font);                                               //设置字体
        double angle = 0;                                               //初始角度
        for (int i = 0; i < array.length; i++) {                        //遍历字符串中的字符
            int x = (int) (len * Math.sin(Math.toRadians(angle + 270)));    //计算每个文字的位置
            int y = (int) (len * Math.cos(Math.toRadians(angle + 270)));    //计算每个文字的位置
            g2.drawString(array[i] + "", width / 2 + x - 168, height / 2 - y - 95); //绘制每个字符，其中 168 和 95 是坐标平移值
            angle = angle + 360d / array.length;                        //改变角度
        }
    }
}
```

秘笈心法

心法领悟 025：正确设置绘图上下文的属性。

在进行图形和文本的绘制时，要求能够正确设置绘图上下文的属性，如文本的字体、颜色，图形线条的粗细、虚实及颜色等，正确设置的方法是在绘制每一种新样式的文本或图形之前，先对绘图上下文的属性进行设置，然后再绘制文本和图形，这样设置的绘图上下文属性才是有效的。

1.4　图形的合并运算

实例 026　图形的加运算　　中级

光盘位置：光盘\MR\026　　趣味指数：★★★

实例说明

本实例演示在 Java 中如何实现图形的加运算，即取两个图形的并集。运行程序，将在窗体上显示进行加运算后的图形，效果如图 1.26 所示。

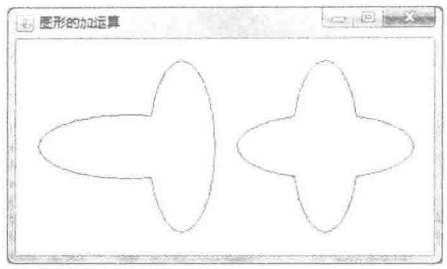


图 1.26　图形进行加运算的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 draw()方法和 Area 类来实现的，其中 Area 类用于封装图形对象，并通过 add()方法对封装的图形对象进行加运算。

（1）使用 Area 类的构造方法封装图形对象，其构造方法的定义如下：

```
public Area(Shape s)
```

参数说明

s：是 Area 类封装的图形对象。

（2）使用 Area 类的 add()方法对封装的图形对象进行加运算，该方法的定义如下：

```
public void add(Area rhs)
```

参数说明

rhs：与当前 Area 对象进行加运算的 Area 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PlusOperationFrame 窗体类。

（3）在 PlusOperationFrame 窗体类中创建内部面板类 PlusOperationPanel，并重写 JComponent 类的 paint()方法，在该方法中实现图形的加运算。

（4）将内部面板类 PlusOperationPanel 的实例添加到窗体类 PlusOperationFrame 的内容面板上，用于在窗体上显示图形进行加运算后的效果，代码如下：

```
class PlusOperationPanel extends JPanel {                          //创建内部面板类
    public void paint(Graphics g) {                                //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                             //获得 Graphics2D 对象
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(20, 70, 160, 60);    //创建椭圆对象
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(120, 20, 60, 160);   //创建椭圆对象
        Area area1 = new Area(ellipse1);                           //创建区域椭圆
        Area area2 = new Area(ellipse2);                           //创建区域椭圆
        area1.add(area2);                                          //两个区域椭圆进行加运算
        g2.draw(area1);                                            //绘制加运算后的区域椭圆
        Ellipse2D.Float ellipse3 = new Ellipse2D.Float(200, 70, 160, 60);   //创建椭圆对象
        Ellipse2D.Float ellipse4 = new Ellipse2D.Float(250, 20, 60, 160);   //创建椭圆对象
        Area area3 = new Area(ellipse3);                           //创建区域椭圆
        Area area4 = new Area(ellipse4);                           //创建区域椭圆
        area3.add(area4);                                          //两个区域椭圆进行加运算
        g2.draw(area3);                                            //绘制加运算后的区域椭圆
    }
}
```

秘笈心法

心法领悟 026：实现多个图形的加运算。

要实现多个图形的加运算，可以先对两个图形进行加运算，然后再将运算得到的图形与其他图形进行加运算，从而实现多个图形的加运算。

实例 027　图形的减运算

光盘位置：光盘\MR\027　　中级　　趣味指数：★★★

实例说明

本实例演示在 Java 中如何实现图形的减运算，即从当前图形中减去与另一个图形的交集。运行程序，将在窗体上显示进行减运算后的图形，效果如图 1.27 所示。

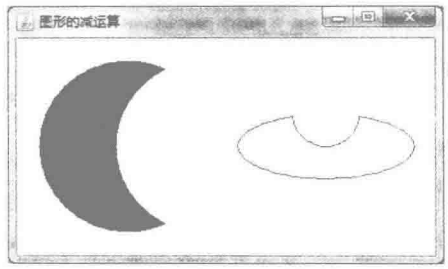


图 1.27　图形进行减运算的效果

关键技术

本实例主要是通过在 JPanel 类的子类中，重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 draw()方法和 Area 类来实现的，其中 Area 类用于封装图形对象，并通过 subtract()方法对封装的图形对象进行减运算。

使用 Area 类的 subtract()方法对封装的图形对象进行减运算，该方法的定义如下：

```
public void subtract(Area rhs)
```

参数说明

rhs：与当前 Area 对象进行减运算的 Area 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SubtractOperationFrame 窗体类。

（3）在 SubtractOperationFrame 窗体类中创建内部面板类 SubtractOperationPanel，并重写 JComponent 类的 paint()方法，在该方法中实现图形的减运算。

（4）将内部面板类 SubtractOperationPanel 的实例添加到窗体类 SubtractOperationFrame 的内容面板上，用于在窗体上显示图形进行减运算后的效果，代码如下：

```
class SubtractOperationPanel extends JPanel {                          //创建内部面板类
    public void paint(Graphics g) {                                    //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                                 //获得 Graphics2D 对象
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(20, 20, 160, 160);   //创建圆对象
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(90, 20, 160, 160);   //创建圆对象
        Area area1 = new Area(ellipse1);                               //创建区域圆
        Area area2 = new Area(ellipse2);                               //创建区域圆
        area1.subtract(area2);                                         //两个区域圆进行减运算
        g2.fill(area1);                                                //绘制减运算后的区域圆
        Ellipse2D.Float ellipse3 = new Ellipse2D.Float(200, 70, 160, 60);   //创建椭圆对象
        Ellipse2D.Float ellipse4 = new Ellipse2D.Float(250, 40, 60, 60);    //创建圆对象
```

```
            Area area3 = new Area(ellipse3);                    //创建区域椭圆
            Area area4 = new Area(ellipse4);                    //创建区域圆
            area3.subtract(area4);                              //两个区域图形进行减运算
            g2.draw(area3);                                     //绘制减运算后的区域图形
        }
    }
```

秘笈心法

心法领悟 027：进行图形的加减混合运算。

要进行图形的加减混合运算，实际上就是对图形按一定的顺序先进行加运算或减运算，然后再将运算所得的结果与其他图形进行加或减运算，这与多个图形进行加运算有些相似，只不过这里加减运算都用到了而已。

实例 028　图形的交运算

光盘位置：光盘\MR\028　　中级　　趣味指数：★★★

实例说明

本实例演示在 Java 中如何实现图形的交运算，即保留两个图形的交集。运行程序，将在窗体上显示进行交运算后的图形，效果如图 1.28 所示。

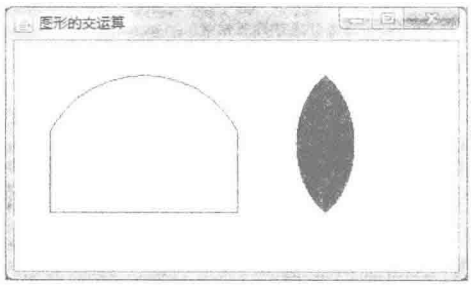


图 1.28　图形进行交运算的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 draw()方法和 Area 类来实现的，其中 Area 类用于封装图形对象，并通过 intersect()方法对封装的图形对象进行交运算。

使用 Area 类的 intersect()方法对封装的图形对象进行交运算，该方法的定义如下：

```
public void intersect(Area rhs)
```

参数说明

rhs：与当前 Area 对象进行交运算的 Area 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 IntersectOperationFrame 窗体类。

（3）在 IntersectOperationFrame 窗体类中创建内部面板类 IntersectOperationPanel，并重写 JComponent 类的 paint()方法，在该方法中实现图形的交运算。

（4）将内部面板类 IntersectOperationPanel 的实例添加到窗体类 IntersectOperationFrame 的内容面板上，用于在窗体上显示图形进行交运算后的效果，代码如下：

```
class IntersectOperationPanel extends JPanel {                          //创建内部面板类
    public void paint(Graphics g) {                                     //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                                  //获得 Graphics2D 对象
        Rectangle2D.Float rect = new Rectangle2D.Float(30, 30, 160, 120);  //创建矩形对象
        Ellipse2D.Float ellipse = new Ellipse2D.Float(20, 30, 180, 180);   //创建圆对象
        Area area1 = new Area(rect);                                    //创建区域矩形
        Area area2 = new Area(ellipse);                                 //创建区域圆
        area1.intersect(area2);                                         //两个区域图形进行交运算
        g2.draw(area1);                                                 //绘制交运算后的区域图形
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(190, 20, 100, 140);  //创建椭圆对象
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(240, 20, 100, 140);  //创建椭圆对象
        Area area3 = new Area(ellipse1);                                //创建区域椭圆
        Area area4 = new Area(ellipse2);                                //创建区域椭圆
        area3.intersect(area4);                                         //两个区域椭圆进行交运算
        g2.fill(area3);                                                 //绘制交运算后的区域椭圆
    }
}
```

秘笈心法

心法领悟 028：交运算的原理。

交运算是根据数学运算中的集合交运算原理实现的。对于集合的交运算，是取两个集合的公共元素；而对于图形的交运算，是取两个图形重叠的部分。

实例 029　图形的异或运算　　中级

光盘位置：光盘\MR\029　　趣味指数：★★★

实例说明

本实例演示在 Java 中如何实现图形的异或运算，即两个图形去除交集后剩下的部分。运行程序，将在窗体上显示进行异或运算后的图形，效果如图 1.29 所示。

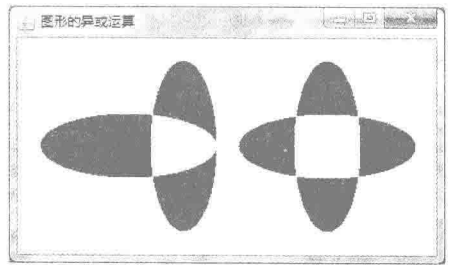


图 1.29　图形进行异或运算的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 draw()方法和 Area 类来实现的，其中 Area 类用于封装图形对象，并通过 exclusiveOr()方法对封装的图形对象进行异或运算。

使用 Area 类的 exclusiveOr()方法对封装的图形对象进行异或运算，该方法的定义如下：

public void exclusiveOr(Area rhs)

参数说明

rhs：与当前 Area 对象进行异或运算的 Area 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ExclusiveOrOperationFrame 窗体类。

（3）在 ExclusiveOrOperationFrame 窗体类中创建内部面板类 ExclusiveOrOperationPanel，并重写 JComponent 类的 paint()方法，在该方法中实现图形的异或运算。

（4）将内部面板类 ExclusiveOrOperationPanel 的实例添加到窗体类 ExclusiveOrOperationFrame 的内容面板上，用于在窗体上显示图形进行异或运算后的效果，代码如下：

```
class ExclusiveOrOperationPanel extends JPanel {                    //创建内部面板类
    public void paint(Graphics g) {                                 //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                              //获得 Graphics2D 对象
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(20, 70, 160, 60);    //创建椭圆对象
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(120, 20, 60, 160);   //创建椭圆对象
        Area area1 = new Area(ellipse1);                            //创建区域椭圆
        Area area2 = new Area(ellipse2);                            //创建区域椭圆
        area1.exclusiveOr(area2);                                   //两个区域椭圆进行异或运算
        g2.fill(area1);                                             //绘制异或运算后的区域椭圆
        Ellipse2D.Float ellipse3 = new Ellipse2D.Float(200, 70, 160, 60);   //创建椭圆对象
        Ellipse2D.Float ellipse4 = new Ellipse2D.Float(250, 20, 60, 160);   //创建椭圆对象
        Area area3 = new Area(ellipse3);                            //创建区域椭圆
        Area area4 = new Area(ellipse4);                            //创建区域椭圆
        area3.exclusiveOr(area4);                                   //两个区域椭圆进行异或运算
        g2.fill(area3);                                             //绘制异或运算后的区域椭圆
    }
}
```

秘笈心法

心法领悟 029：绘制特殊的图形。

一些特殊的图形可能不太容易绘制，这时可以考虑是否可以使用图形的加、减、交、异或等运算实现，这可能是一种捷径。

第 2 章

Java 图像处理

- 图像处理
- 颜色处理

2.1 图像处理

实例 030 绘制图像

光盘位置：光盘\MR\030

初级

趣味指数：★★★

实例说明

在进行应用程序开发时，为了使程序界面美观，可以为应用程序窗体添加背景图片，方法是通过 Java 的绘图技术在控件上绘制图像，并将带有图片的控件添加到窗体上。运行程序，可以看到在窗体显示的图片，效果如图 2.1 所示。

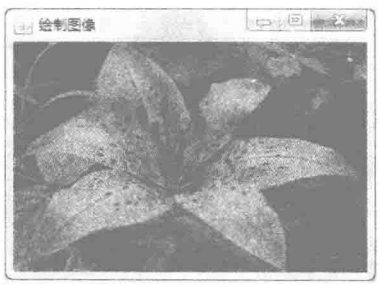


图 2.1 绘制图像的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawImage()方法来实现的。

使用 Graphics 类的 drawImage()方法可绘制图像，该方法的定义如下：

```
public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer)
```

参数说明

❶ img：需要绘制的图像对象。

❷ x：绘制顶点的 x 坐标。

❸ y：绘制顶点的 y 坐标。

❹ observer：图像观察者。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawImageFrame 窗体类。

（3）在 DrawImageFrame 窗体类中创建内部面板类 DrawImagePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawImage()方法绘制图像。

（4）将内部面板类 DrawImagePanel 的实例添加到窗体类 DrawImageFrame 的内容面板上，用于在窗体上显示绘制的图像，窗体类 DrawImageFrame 的构造方法中，创建图像对象的代码如下：

```
URL imgUrl = DrawImageFrame.class.getResource("/img/image.jpg");      //获取图片资源的路径
img = Toolkit.getDefaultToolkit().getImage(imgUrl);                   //获取图像资源
```

面板类 DrawImagePanel 的代码如下：

```
class DrawImagePanel extends JPanel {
    public void paint(Graphics g) {                                   //重写 paint()方法
```

```
            g.drawImage(img, 0, 0, this);                    //绘制图像对象
        }
    }
```

说明：本实例用到的图片文件 image.jpg 需要放到项目中 src 文件夹的子文件夹 img 中。

秘笈心法

心法领悟 030：美化应用程序窗体界面。

通过 Java 的绘制图像功能，可以美化应用程序的窗体界面，方法是将绘制有图像的面板对象添加到窗体内容面板中，这样就可以在窗体上显示图片，从而起到美化窗体的作用。

实例 031　缩放图像

光盘位置：光盘\MR\031　　初级　　趣味指数：★★★

实例说明

本实例演示如何使用 Java 绘图技术对图像进行缩放。运行程序，通过调整窗体上的滑块，可以对窗体上显示的图片进行缩放，效果如图 2.2 所示。

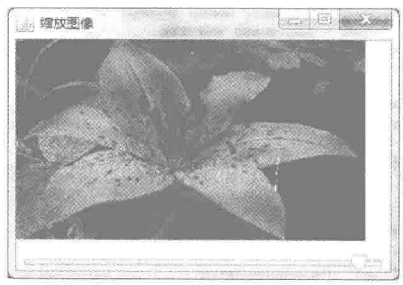


图 2.2　缩放图像

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawImage()方法来实现的。

使用 Graphics 类的 drawImage()方法可缩放图像，该方法的定义如下：

public abstract boolean drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)

参数说明

❶ img：需要绘制的图像对象。

❷ x：绘制顶点的 x 坐标。

❸ y：绘制顶点的 y 坐标。

❹ width：显示图像的矩形的宽度。

❺ height：显示图像的矩形的高度。

❻ observer：图像观察者。

说明：通过 drawImage()方法的参数 width 和 height，可以实现对图像的缩放操作。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ZoomImageFrame 窗体类。

（3）在 ZoomImageFrame 窗体类中创建内部面板类 ZoomImagePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawImage()方法绘制缩放后的图像。

（4）将内部面板类 ZoomImagePanel 的实例添加到窗体类 ZoomImageFrame 的内容面板上，并在窗体下方添加一个滑块控件，通过拖动滑块可以在窗体上显示图像的缩放效果。窗体上滑块控件的事件代码如下：

```
slider.addChangeListener(new ChangeListener() {
    public void stateChanged(final ChangeEvent e) {          //滑块位置改变时执行该方法
        imagePanel.repaint();                                //重新调用面板类的 paint()方法
    }
});
```

面板类 ZoomImagePanel 用于绘制调整滑块缩放的图像，该类的代码如下：

```
class ZoomImagePanel extends JPanel {
    public void paint(Graphics g) {
        g.clearRect(0, 0, this.getWidth(), this.getHeight());  //清除绘图上下文的内容
        imgWidth = img.getWidth(this);                         //获取图片宽度
        imgHeight = img.getHeight(this);                       //获取图片高度
        float value = slider.getValue();                       //滑块组件的取值
        newW = (int) (imgWidth * value / 100);                 //计算图片缩放后的宽度
        newH = (int) (imgHeight * value / 100);                //计算图片缩放后的高度
        g.drawImage(img, 0, 0, newW, newH, this);              //绘制指定大小的图片
    }
}
```

秘笈心法

心法领悟 031：通过滑块控件调整颜色的 RGB 值。

当需要手动对颜色值进行调整时，可以通过 3 个滑块控件分别代表颜色的 R、G 和 B 值，这样就能通过滑块控件调整颜色的 RGB 值。使用时，需要将滑块控件的最大值设置为 255，因为颜色的 RGB 值的范围是 0~255，滑块控件的最大值可通过其 maximum 属性设置。

实例 032　翻转图像

光盘位置：光盘\MR\032　　初级　　趣味指数：★★★

实例说明

本实例演示如何利用 Java 的绘图技术实现翻转图像的操作。运行程序，单击窗体上的“水平翻转”和“垂直翻转”按钮，可以实现图像的水平翻转和垂直翻转操作。单击窗体上的“垂直翻转”按钮，将对图像进行垂直翻转，效果如图 2.3 所示。

图 2.3　垂直翻转图像的效果

关键技术

本实例也是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawImage()方法来实现的。

使用 Graphics 类的 drawImage()方法可翻转图像，该方法的定义如下：

```
public abstract boolean drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver observer)
```

参数说明

❶ img：需要绘制的图像对象。
❷ dx1：目标矩形第 1 个角的 x 坐标。
❸ dy1：目标矩形第 1 个角的 y 坐标。
❹ dx2：目标矩形第 2 个角的 x 坐标。
❺ dy2：目标矩形第 2 个角的 y 坐标。
❻ sx1：源矩形第 1 个角的 x 坐标。
❼ sy1：源矩形第 1 个角的 y 坐标。
❽ sx2：源矩形第 2 个角的 x 坐标。
❾ sy2：源矩形第 2 个角的 y 坐标。
❿ observer：图像观察者。

说明：通过互换 drawImage()方法中源矩形第 1 个角和第 2 个角的 x 坐标，可以实现图像的水平翻转；通过互换 drawImage()方法中源矩形第 1 个角和第 2 个角的 y 坐标，可以实现图像的垂直翻转。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PartImageFrame 窗体类。

（3）在 PartImageFrame 窗体类中创建内部面板类 PartImagePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawImage()方法绘制图像。

（4）将内部面板类 PartImagePanel 的实例添加到窗体类 PartImageFrame 的内容面板上，在窗体上添加两个按钮，分别用于实现图像的水平翻转和垂直翻转，并在窗体上显示翻转后的图像。“水平翻转”按钮的事件代码如下：

```
btn_h.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        //下面 3 行代码用于交换 sx1 和 sx2 的值
        int x = sx1;
        sx1 = sx2;
        sx2 = x;
        imagePanel.repaint();                          //重新调用面板类的 paint()方法
    }
});
```

“垂直翻转”按钮的事件代码如下：

```
btn_v.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        //下面 3 行代码用于交换 sy1 和 sy2 的值
        int y = sy1;
        sy1 = sy2;
        sy2 = y;
        imagePanel.repaint();                          //重新调用面板类的 paint()方法
    }
});
```

面板类 PartImagePanel 用于绘制原图像和翻转后的图像，该类的代码如下：

```
class PartImagePanel extends JPanel {
    public void paint(Graphics g) {
        g.clearRect(0, 0, this.getWidth(), this.getHeight());    //清除绘图上下文的内容
```

```
            g.drawImage(img, dx1, dy1, dx2, dy2, sx1, sy1, sx2, sy2, this);    //绘制图像
        }
    }
```

秘笈心法

心法领悟 032：交换两个变量的值。

在进行程序开发时，经常需要交换两个变量 a 和 b 的值，这时可以定义一个临时变量 c，然后将变量 a 的值赋值给临时变量 c，接着将变量 b 的值赋值给变量 a，最后再将临时变量 c 的值（即 a 的值）赋值给变量 b，从而交换了变量 a 和 b 的值。

实例 033 旋转图像

光盘位置：光盘\MR\033

初级

趣味指数：★★★

实例说明

本实例演示如何利用 Java 的绘图技术实现图像的旋转操作，即让图像绕坐标原点进行旋转。运行程序，效果如图 2.4 所示。

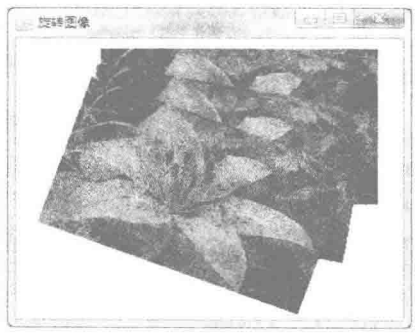


图 2.4　旋转图像的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 rotate()方法和从 Graphics 类继承的 drawImage()方法来实现的。

使用 Graphics2D 类的 rotate()方法可以实现图像的旋转，该方法的定义如下：

```
public abstract void rotate(double theta)
```

参数说明

theta：旋转的角度，以弧度为单位。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 RotateImageFrame 窗体类。

（3）在 RotateImageFrame 窗体类中创建内部面板类 RotateImagePanel，并重写 JComponent 类的 paint()方法，在该方法中使用从 Graphics 类继承的 drawImage()方法和 Graphics2D 类的 rotate()方法实现图像旋转。

（4）将内部面板类 RotateImagePanel 的实例，添加到窗体类 RotateImageFrame 的内容面板上，用于在窗体上显示旋转后的图像，代码如下：

```
class RotatePanel extends JPanel {
    public void paint(Graphics g) {
```

```
            Graphics2D g2 = (Graphics2D) g;                    //获得 Graphics2D 对象
            g2.drawImage(img, 80, 10, 260, 150, this);         //绘制指定大小的图片
            g2.rotate(Math.toRadians(10));                     //将图片旋转 10°
            g2.drawImage(img, 80, 10, 260, 150, this);         //绘制指定大小的图片
            g2.rotate(Math.toRadians(10));                     //将图片旋转 10°
            g2.drawImage(img, 80, 10, 260, 150, this);         //绘制指定大小的图片
        }
    }
```

秘笈心法

心法领悟 033：以任意角度旋转图像。

在应用程序中进行图像处理时，可以通过文本框输入任意角度值，然后在按钮事件中添加旋转图像的代码，从而实现以任意角度旋转图像的功能。

实例 034　倾斜图像

光盘位置：光盘\MR\034　　初级　　趣味指数：★★★

实例说明

本实例演示如何在应用程序中利用 Java 的绘图技术实现倾斜图像的操作。运行程序，将在窗体上显示倾斜的图像，效果如图 2.5 所示。

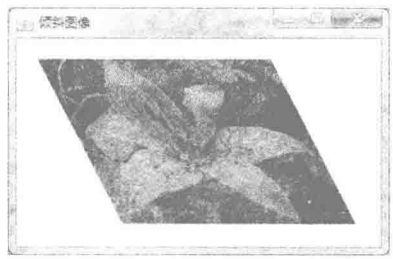


图 2.5　倾斜图像的效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 shear()方法和从 Graphics 类继承的 drawImage()方法来实现的。

使用 Graphics2D 类的 shear()方法可以实现图像的倾斜，该方法的定义如下：

```
public abstract void shear(double shx, double shy)
```

参数说明

❶ shx：在正 X 轴方向移动坐标的乘数，它可以作为其 y 坐标的函数。

❷ shy：在正 Y 轴方向移动坐标的乘数，它可以作为其 x 坐标的函数。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ShearImageFrame 窗体类。

（3）在 ShearImageFrame 窗体类中创建内部面板类 ShearImagePanel，并重写 JComponent 类的 paint()方法，在该方法中使用从 Graphics 类继承的 drawImage()方法和 Graphics2D 类的 shear()方法实现图像倾斜。

（4）将内部面板类 ShearImagePanel 的实例添加到窗体类 ShearImageFrame 的内容面板上，用于在窗体上显示倾斜后的图像，代码如下：

```
class ShearImagePanel extends JPanel {                    //绘制倾斜图像的面板类
    public void paint(Graphics g) {
        Graphics2D g2=(Graphics2D) g;                     //获得 Graphics2D 对象
        g2.shear(0.5, 0);                                 //倾斜图像
        g2.drawImage(img, 10, 20, 220, 160, this);        //绘制指定大小的图片
    }
}
```

秘笈心法

心法领悟 034：另一种实现倾斜的方法。

首先创建 AffineTransform 类的实例，并通过该实例调用 shear()方法倾斜图像，然后使用 Graphics2D 对象的 setTransform()方法为绘图上下文指定 AffineTransform 对象，从而实现倾斜图像的操作。

实例 035 裁剪图片

光盘位置：光盘\MR\035　　中级　　趣味指数：★★★★

实例说明

本实例演示如何利用 Java 的绘图技术实现裁剪图片的操作。运行程序，通过鼠标在分割面板的左侧选择图片的裁剪区域，将在分割面板的右侧显示裁剪区域中的图片，效果如图 2.6 所示。

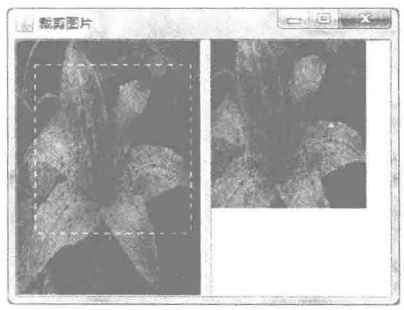


图 2.6　裁剪图片的效果

关键技术

本实例主要是通过 Robot 类的 createScreenCapture()方法，以 BasicStroke 类创建虚线对象，并结合 Graphics 类的 drawRect()方法实现图片的裁剪和绘制选区的。

（1）使用 BasicStroke 类重载的构造方法创建笔画对象，并指定笔画为虚线模式。

（2）使用 Graphics 类的 drawRect()方法绘制矩形。

（3）使用 Robot 类的 createScreenCapture()方法裁剪图片，该方法的定义如下：

```
public BufferedImage createScreenCapture(Rectangle screenRect)
```

参数说明

❶ screenRect：屏幕上被截取的矩形区域。

❷ 返回值：从屏幕上截取的缓冲图像对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 CutImageFrame 窗体类。

（3）在 CutImageFrame 窗体类中创建内部面板类 OldImagePanel，并重写 JComponent 类的 paint()方法，在该

方法中使用 Graphics 类的 drawImage()方法绘制原图片对象以及使用 drawRect()方法绘制表示选择区域的矩形。

（4）在 CutImageFrame 窗体类中创建内部面板类 CutImagePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawImage()方法绘制裁剪所得的图片。

（5）将内部面板类 OldImagePanel 和 CutImagePanel 的实例分别添加到分割面板的左右两侧，然后将分割面板添加到窗体类 CutImageFrame 的内容面板上，用于在窗体上显示原图片和裁剪所得的图片，OldImagePanel 面板类的代码如下：

```
class OldImagePanel extends JPanel {                                    //创建绘制原图像的面板类
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        g2.drawImage(img, 0, 0, this.getWidth(), this.getHeight(), this);   //绘制图像
        g2.setColor(Color.WHITE);
        if (flag) {
            float[] arr = { 5.0f };                                     //创建虚线模式的数组
            BasicStroke stroke = new BasicStroke(1, BasicStroke.CAP_BUTT,
                    BasicStroke.JOIN_BEVEL, 1.0f, arr, 0);              //创建宽度是 1 的平头虚线笔画对象
            g2.setStroke(stroke);                                       //设置笔画对象
            g2.drawRect(pressPanelX, pressPanelY, releaseX - pressX,
                    releaseY - pressY);                                 //绘制矩形选区
        }
    }
}
```

CutImagePanel 面板类的代码如下：

```
class CutImagePanel extends JPanel {                                    //创建绘制裁剪结果的面板类
    public void paint(Graphics g) {
        g.clearRect(0, 0, this.getWidth(), this.getHeight());           //清除绘图上下文的内容
        g.drawImage(buffImage, 0, 0, releaseX - pressX, releaseY - pressY, this);   //绘制图像
    }
}
```

秘笈心法

心法领悟 035：实现自动化处理。

从 JDK 1.3 开始，用户可以使用 Robot 类实现程序自动化、实现自运行演示程序和其他需要控制鼠标和键盘的应用程序，使用该类提供的方法可以实际操作鼠标和键盘，从而实现程序的自动化处理。

2.2　颜 色 处 理

实例 036　调整图片的亮度　　高级

光盘位置：光盘\MR\036　　趣味指数：★★★★

实例说明

本实例演示如何利用 Java 的绘图技术调整图片的亮度。运行程序，效果如图 2.7 所示，单击窗体上的“变亮”按钮，可以使图片变亮；单击“变暗”按钮，可以使图片变暗；单击“恢复”按钮，可以将图片恢复为原来的样式。

图 2.7　调整图片的亮度

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawImage()方法实现绘制缓冲图像，该缓冲图像是通过 RescaleOp 类的 filter()方法进行缩放处理后，使图片变亮或变暗的图像。

（1）使用 Graphics 类的 drawImage()方法绘制图像。

（2）使用 RescaleOp 类的 filter()方法对原缓冲图像对象进行重缩放，从而达到调整图片亮度的目的，filter()方法的定义如下：

```
public final BufferedImage filter(BufferedImage src, BufferedImage dst)
```

参数说明

❶ src：要过滤的源 BufferedImage 对象。

❷ dst：目标 BufferedImage 对象，或 null。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ImageBrightenFrame 窗体类。

（3）在 ImageBrightenFrame 窗体类中创建内部面板类 ImageBrightenPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawImage()方法绘制图像。

（4）将内部面板类 ImageBrightenPanel 的实例添加到窗体类 ImageBrightenFrame 的内容面板上，用于在窗体上显示变亮后的图像、原图像和变暗后的图像，代码如下：

```
class ImageBrightenPanel extends JPanel {
    public void paint(Graphics g) {                    //重写 paint()方法
        if (image != null) {
            g.drawImage(image, 0, 0, null);            //将缓冲图像对象绘制到面板上
        }
    }
}
```

（5）“变亮”按钮的事件代码，用于使图片变亮，代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        float a = 1.0f;                                //定义缩放因子
        float b = 5.0f;                                //定义偏移量
        RescaleOp op = new RescaleOp(a,b,null);        //创建具有指定缩放因子和偏移量的 RescaleOp 对象
        image = op.filter(image, null);                //对源图像中的数据进行逐像素重缩放，达到变亮的效果
        repaint();                                     //重新绘制图像
    }
});
```

（6）“变暗”按钮的事件代码，用于使图片变暗，代码如下：

```
button_3.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        float a = 1.0f;                                //定义缩放因子
        float b = -5.0f;                               //定义偏移量
        RescaleOp op = new RescaleOp(a,b,null);        //创建具有指定缩放因子和偏移量的 RescaleOp 对象
        image = op.filter(image, null);                //对源图像中的数据进行逐像素重缩放，达到变暗的效果
        repaint();                                     //重新绘制图像
    }
});
```

秘笈心法

心法领悟 036：使用 RescaleOp 类缩放图像的原理。

使用 RescaleOp 类可以缩放图像，其原理是将每个像素的样本值乘以一个缩放因子，然后加上一个偏移量，从而实现对源图像中的数据进行逐像素缩放。

实例 037　转换彩色图片为灰度图片

光盘位置：光盘\MR\037　　高级　　趣味指数：★★★★

实例说明

本实例演示如何利用 Java 的绘图技术将彩色图片转换为灰度图片。运行程序，将在窗体上显示彩色图片，单击窗体上的“转换为灰度”按钮，可以将窗体上的彩色图片转换为灰度图片，效果如图 2.8 所示。

图 2.8　彩色图片转换为灰度图片

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawImage()方法绘制缓冲图像，该缓冲图像是通过 ColorConvertOp 类的 filter()方法进行颜色转换后得到的灰度图像。

（1）使用 Graphics 类的 drawImage()方法绘制图像。

（2）使用 ColorConvertOp 类的构造方法创建 ColorConvertOp 对象，其构造方法定义如下：

```
public ColorConvertOp(ColorSpace srcCspace, ColorSpace dstCspace, RenderingHints hints)
```

参数说明

❶ srcCspace：原颜色空间对象。

❷ dstCspace：目标颜色空间对象。

❸ hints：用于控制颜色转换的 RenderingHints 对象，或 null。

（3）使用 ColorConvertOp 类的 filter()方法，将彩色图像转换为灰度图像，该方法的定义如下：

```
public final BufferedImage filter(BufferedImage src, BufferedImage dst)
```

参数说明

❶ src：要过滤的源 BufferedImage 对象。

❷ dst：目标 BufferedImage 对象，或 null。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MultiColorToGrayFrame 窗体类。

（3）在 MultiColorToGrayFrame 窗体类中创建内部面板类 ColorToGrayPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawImage()方法绘制图像。

（4）将内部面板类 ColorToGrayPanel 的实例添加到窗体类 MultiColorToGrayFrame 的内容面板上，用于在窗体上显示原图像和转换为灰度后的图像，代码如下：

```
class ColorToGrayPanel extends JPanel {
    public void paint(Graphics g) {                                    //重写 paint()方法
        if (image != null) {
            g.drawImage(image, 0, 0, null);                            //将缓冲图像对象绘制到面板上
        }
    }
}
```

（5）“转换为灰度”按钮的事件代码用于将彩色图片转换为灰度图片，代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        ColorSpace colorSpace1 = ColorSpace
                .getInstance(ColorSpace.CS_GRAY);                      //创建内置线性为灰度的颜色空间
        ColorSpace colorSpace2 = ColorSpace
        .getInstance(ColorSpace.CS_LINEAR_RGB);                        //创建内置线性为 RGB 的颜色空间
        ColorConvertOp op = new ColorConvertOp(colorSpace1,colorSpace2, null);  //创建进行颜色转换的对象
        image = op.filter(image, null);                                //对缓冲图像进行颜色转换
        repaint();                                                     //重新绘制图像
    }
});
```

秘笈心法

心法领悟 037：另一种将图像转换为灰度的方法。

使用 PixelGrabber 类从图像中检索 RGB 像素的整数数组，然后使用 ColorModel 对象获得数组中各元素的 RGB 分量值，并分别转换为灰度，转换为灰度的方法是用 RGB 的各分量值进行计算，代码为 int gray=(int)(r*0.3+g*0.59+b*0.11); r=g=b=gray;，然后使用(255 << 24) |(r << 16) |(g << 8)| b;将颜色转换为像素，并存储到数组中，最后使用 MemoryImageSource 类创建 ImageProducer 对象，并把 ImageProducer 对象传递给从 Component 类继承的 createImage()方法，从而实现将彩色图像转换为灰度图像的操作。

实例 038　像素值生成图像

高级

趣味指数：★★★★★

实例位置：光盘\MR\038

实例说明

本实例演示如何使用像素值生成图像，效果如图 2.9 所示。

图 2.9　使用像素值生成图像

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 drawImage()方法绘制生成的图像，生成图像的方法是通过 MemoryImageSource 类创建 ImageProducer 对象，并把 ImageProducer 对象传递给从 Component 类继承的 createImage()方法，从而实现了图像的创建。

（1）使用 MemoryImageSource 类创建 ImageProducer 对象，该类构造方法的定义如下：

```
public MemoryImageSource(int w, int h, int[] pix, int off, int scan)
```

参数说明

❶ w：像素矩形的宽度。

❷ h：像素矩形的高度。

❸ pix：像素数组。

❹ off：数组中存储第一个像素的偏移量。

❺ scan：数组中一行像素到下一行像素之间的距离，与像素矩形的宽度相同。

（2）使用从 Component 类继承的 createImage()方法创建图像对象，该方法定义如下：

```
public Image createImage(ImageProducer producer)
```

参数说明

❶ producer：图像生成器。

❷ 返回值：该方法执行成功返回一个 Image 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 CreateImageFrame 窗体类。

（3）在 CreateImageFrame 窗体类中创建内部面板类 CreateImagePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 drawImage()方法绘制生成的图像。

（4）将内部面板类 CreateImagePanel 的实例添加到窗体类 CreateImageFrame 的内容面板上，用于在窗体上显示生成的图像，面板类的代码如下：

```
class CreateImagePanel extends JPanel {                          //创建用像素值生成图像的面板类
    public void paint(Graphics g) {
        int w = 300;                                             //宽度
        int h = 220;                                             //高度
        int pix[] = new int[w * h];                              //存储像素值的数组
        int index = 0;                                           //存储数组的索引
        for (int y = 0; y < h; y++) {                            //在纵向进行调整，从黑色渐变到红色
            int red = (y * 255) / (h - 1);                       //计算纵向的颜色值
            for (int x = 0; x < w; x++) {                        //在横向进行调整，从黑色渐变到蓝色
                int blue = (x * 255) / (w - 1);                  //计算横向的颜色值
                //通过移位运算和逻辑或运算计算像素值，并赋值给像素数组
                pix[index++] = (255 << 24) | (red << 16) | blue;
            }
        }
            //创建使用数组为 Image 生成像素值的 ImageProducer 对象
        ImageProducer imageProducer = new MemoryImageSource(w, h, pix, 0, w);
        Image img = createImage(imageProducer);                  //创建图像对象
        g.drawImage(img, 0, 0,getWidth(),getHeight(), this);     //绘制图像
    }
}
```

秘笈心法

心法领悟 038：使用 PixelGrabber 类获得的图像像素在数组中的存储位置。

使用构造方法 PixelGrabber(Image img, int x, int y, int w, int h, int[] pix, int off, int scansize)，可以创建一个 PixelGrabber 对象，该对象用于从指定图像中抓取像素矩形部分(x, y, w, h)，并以默认的 RGB ColorModel 形式将像素存储到数组 pix 中，并且在矩形部分(x, y, w, h)内，像素(i, j)的 RGB 数据存储在数组中的 pix[(j - y) * scansize+(i - x)+off]位置处。

说明：该方法中的参数 img 是用于从中检索像素的图像；x 是从图像中进行检索的像素矩形左上角 x 坐标，其相对于默认（未缩放）图像大小；y 是从图像中进行检索的像素矩形左上角 y 坐标；w 是要检索的像素矩形的宽度；h 是要检索的像素矩形的高度；pix 用于保存从图像中检索的 RGB 像素的整数数组；off 是数组中存储第一个像素的偏移量；scansize 是数组中一行像素到下一行像素之间的距离。

第 3 章

绘图特效

- 文字特效
- 图片特效

3.1 文字特效

实例 039	立体效果的文字 光盘位置：光盘\MR\039	初级 趣味指数：★★★

实例说明

本实例演示如何利用 Java 的绘图技术实现立体效果文字的绘制。运行程序，将在窗体上显示具有立体效果的文字，效果如图 3.1 所示。

图 3.1 立体效果的文字

关键技术

使用 Graphics 类的 setFont()方法设置完字体、字型和字号后，使用 Graphics 类的 setColor()方法将绘图上下文的前景色设置为灰色，然后使用 Graphics 类的 drawString()方法绘制文本，并且每次绘制的文本都向右下方移动一小段距离，最后将绘图上下文的前景色更改为黑色，再绘制一次文本，从而实现立体字效果。

（1）使用 Graphics 类的 setFont()方法，并将 Font 类创建的字体对象作为 setFont()方法的参数，实现为文本设置字体的操作，setFont()方法的定义如下：

```
public abstract void setFont(Font font)
```

参数说明

font：为文本设置的字体对象。

（2）使用 Graphics 类的 setColor()方法，并将 Color 类创建的颜色对象作为 setColor()方法的参数，实现为文本和图形设置颜色的操作，setColor()方法的定义如下：

```
public abstract void setColor(Color color)
```

参数说明

color：为文本或图形设置的颜色对象。

（3）使用 Graphics 类的 drawString()方法绘制文本。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SolidTextFrame 窗体类。

（3）在 SolidTextFrame 窗体类中创建内部面板类 SolidTextPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 setFont()、setColor()和 drawString()方法，完成立体效果文字的绘制。

（4）将内部面板类 SolidTextPanel 的实例添加到窗体类 SolidTextFrame 的内容面板上，用于在窗体上显示绘制的立体字，代码如下：

```
class SolidTextPanel extends JPanel {                    //创建内部面板类
    public void paint(Graphics g) {                      //重写 paint()方法
```

```
        String value = "明日科技";
        int x = 16;                                    //文本位置的横坐标
        int y = 100;                                   //文本位置的纵坐标
        Font font = new Font("宋体", Font.BOLD, 72);   //创建字体对象
        g.setFont(font);                               //设置字体
        g.setColor(Color.GRAY);                        //设置颜色为灰色
        int i = 0;                                     //循环变量
        while (i<8){
            g.drawString(value, x, y);                 //绘制文本
            x+=1;                                      //调整绘制点的横坐标值
            y+=1;                                      //调整绘制点的纵坐标值
            i++;                                       //调整循环变量的值
        }
        g.setColor(Color.BLACK);                       //设置颜色为黑色
        g.drawString(value, x, y);                     //绘制文本
    }
}
```

秘笈心法

心法领悟 039：实现文本移动的动画。

在程序中使用 Graphics 类的 drawString()方法绘制文本时，可以通过变量和线程来指定和改变文本绘制点的横、纵坐标，从而可以实现文本移动的动画效果。

实例 040　阴影效果的文字

光盘\MR\040　　初级　　趣味指数：★★★

实例说明

本实例演示如何利用 Java 的绘图技术实现阴影效果文字的绘制。运行程序，将在窗体上显示具有阴影效果的文字，效果如图 3.2 所示。

图 3.2　阴影效果的文字

关键技术

使用 Graphics 类的 setFont()方法设置完字体、字型和字号后，使用 Graphics 类的 setColor()方法将绘图上下文的前景色设置为灰色，然后使用 Graphics 类的 drawString()方法绘制文本，再将绘图上下文的前景色更改为黑色，并且将绘制的文本都向左上方移动一小段距离，从而实现阴影文字的效果。

（1）使用 Graphics 类的 setFont()方法，并将 Font 类创建的字体对象作为 setFont()方法的参数，实现为文本设置字体的操作。

（2）使用 Graphics 类的 setColor()方法，并将 Color 类创建的颜色对象作为 setColor()方法的参数，实现为文本和图形设置颜色的操作。

（3）使用 Graphics 类的 drawString()方法绘制文本。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ShadowTextFrame 窗体类。

（3）在 ShadowTextFrame 窗体类中创建内部面板类 ShadowTextPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 setFont()、setColor()和 drawString()方法，完成阴影效果文字的绘制。

（4）将内部面板类 ShadowTextPanel 的实例添加到窗体类 ShadowTextFrame 的内容面板上，用于在窗体上显示阴影效果的文字，代码如下：

```
class ShadowTextPanel extends JPanel {                    //创建内部面板类
    public void paint(Graphics g) {                       //重写 paint()方法
        String value = "编程词典";
        int x = 16;                                       //文本位置的横坐标
        int y = 100;                                      //文本位置的纵坐标
        Font font = new Font("华文行楷", Font.BOLD, 72);   //创建字体对象
        g.setFont(font);                                  //设置字体
        g.setColor(Color.GRAY);                           //设置颜色为灰色
        int i = 0;                                        //循环变量
        g.drawString(value, x, y);                        //绘制文本
        x -= 3;                                           //调整绘制点的横坐标值
        y -= 3;                                           //调整绘制点的纵坐标值
        g.setColor(Color.BLACK);                          //设置颜色为黑色
        g.drawString(value, x, y);                        //绘制文本
    }
}
```

秘笈心法

心法领悟 040：解决阴影不在文字下方的方法。

在绘制阴影文字时，有时会出现阴影不在文字下方的问题，这是由于先绘制的内容是显示在下面的，而后绘制的内容是显示在上面的，所以解决的方法是在绘制阴影文字时，先将阴影绘制到绘图上下文上，然后再绘制需要显示的文本，并在横向和纵向分别偏移一小段距离，这样就能正常显示文字的阴影了。

实例 041 倾斜效果的文字

光盘位置：光盘\MR\041　　初级　　趣味指数：★★

实例说明

本实例演示如何利用 Java 的绘图技术实现倾斜效果文字的绘制。运行程序，将在窗体上显示具有倾斜效果的文字，效果如图 3.3 所示。

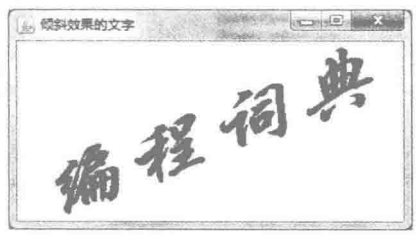


图 3.3　倾斜效果的文字

关键技术

使用 Graphics2D 类的 setShear()方法，倾斜绘图上下文对象，然后使用从 Graphics 类继承的 setFont()方法设

置字体、字型和字号，使用 drawString()方法绘制文本，从而实现倾斜文字的效果。

（1）使用从 Graphics 类继承的 setFont()方法设置绘图上下文的字体。

（2）使用 Graphics2D 类的 setShear()方法，使绘图上下文倾斜，这样绘制的文本就是倾斜的文本，该方法的定义如下：

```
public abstract void set shear(double shx, double shy)
```

参数说明

❶ shx：在正 X 轴方向移动坐标的乘数，可以作为其 y 坐标的函数。

❷ shy：在正 Y 轴方向移动坐标的乘数，可以作为其 x 坐标的函数。

（3）使用从 Graphics 类继承的 drawString()方法绘制文本。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ShearTextFrame 窗体类。

（3）在 ShearTextFrame 窗体类中创建内部面板类 ShearTextPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics2D 类的 setShear()方法，以及从 Graphics 类继承的 setFont()方法和 drawString()方法，完成倾斜文字的绘制。

（4）将内部面板类 ShearTextPanel 的实例添加到窗体类 ShearTextFrame 的内容面板上，用于在窗体上显示倾斜效果的文字，代码如下：

```
class ShearTextPanel extends JPanel {                          //创建内部面板类
    public void paint(Graphics g) {                            //重写 paint()方法
        Graphics2D g2 = (Graphics2D)g;                         //转换为 Graphics2D 类型
        String value = "编程词典";                              //绘制的文本
        int x = 10;                                            //文本位置的横坐标
        int y = 170;                                           //文本位置的纵坐标
        Font font = new Font("华文行楷", Font.BOLD, 72);        //创建字体对象
        g2.setFont(font);                                      //设置字体
        g2.shear(0.1, -0.4);                                   //倾斜画布
        g2.drawString(value, x, y);                            //绘制文本
    }
}
```

秘笈心法

心法领悟 041：快速绘制平行四边形。

在 Java 中，可以通过 Polygon 类创建多边形，从而实现平行四边形的绘制，但是使用该类有一个缺点，就是必须要正确计算出每个点的横、纵坐标，为此可以使用 Graphics2D 类的 rotate()方法旋转绘图上下文，并使用 shear()方法倾斜绘图上下文，然后再绘制矩形，这样所绘制的矩形就可以显示为平行四边形了。

实例 042 渐变效果的文字

初级

光盘位置：光盘\MR\042

趣味指数：★★★★★

实例说明

在程序当中绘制的文本信息通常都是单一的颜色，本实例使用 GradientPaint 类创建封装渐变颜色的对象，并为绘图上下文指定该对象，从而实现绘制渐变效果文字的功能。运行程序，将在窗体上显示具有渐变效果的文字，效果如图 3.4 所示。

图 3.4 渐变效果的文字

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics2D 类的 setPaint()方法，为绘图上下文指定 GradientPaint 类创建的渐变色对象，从而实现绘制渐变效果文字的功能。

（1）使用 Graphics2D 类的 setPaint()方法，并将 GradientPaint 类创建的封装渐变颜色的对象作为 setPaint()方法的参数，实现为图形填充渐变色的操作，setPaint()方法的定义如下：

```
public abstract void setPaint(Paint paint)
```

参数说明

paint：封装了渐变颜色的 Paint 对象。

（2）使用 GradientPaint 类创建封装渐变颜色的对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 GradientTextFrame 窗体类。

（3）在 GradientTextFrame 窗体类中，创建内部面板类 GradientTextPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics2D 类的 setPaint()方法，为绘图上下文指定渐变色，并使用从 Graphics 类继承的 drawString()绘制文本，完成渐变颜色效果文字的绘制。

（4）将内部面板类 GradientTextPanel 的实例添加到窗体类 GradientTextFrame 的内容面板上，用于在窗体上显示渐变效果的文字，代码如下：

```
class GradientTextPanel extends JPanel {                    //创建内部面板类
    public void paint(Graphics g) {                         //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                     //转换为 Graphics2D 类型
        String value = "Java 全能";                          //绘制的文本
        int x = 15;                                         //文本位置的横坐标
        int y = 60;                                         //文本位置的纵坐标
        Font font = new Font("楷体", Font.BOLD, 60);         //创建字体对象
        g2.setFont(font);                                   //设置字体
        //创建循环渐变的 GradientPaint 对象
        GradientPaint paint = new GradientPaint(20, 20, Color.BLUE, 100,120, Color.RED, true);
        g2.setPaint(paint);                                 //设置渐变
        g2.drawString(value, x, y);                         //绘制文本
        font = new Font("华文行楷", Font.BOLD, 60);          //创建新的字体对象
        g2.setFont(font);                                   //设置字体
        x = 80;                                             //文本位置的横坐标
        y = 130;                                            //文本位置的纵坐标
        value = "编程词典";                                  //绘制的文本
        g2.drawString(value, x, y);                         //绘制文本
    }
}
```

秘笈心法

心法领悟 042：改变渐变色中各颜色之间的距离。

在绘制具有渐变颜色的图形和文本时，如果对各种颜色的距离有所限制，可以在使用 GradientPaint 类创建渐变对象时，通过改变起始点和终止点之间的距离来实现。

实例 043 变色的文字

中级

趣味指数：★★★★★

源代码：光盘\MR\043

实例说明

本实例使用 Java 的绘图和多线程技术，实现在程序运行时动态改变文字的颜色。运行程序，在窗体上绘制的文字颜色不断改变，效果如图 3.5 所示。

图 3.5　会变色的文字

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并实现 Runnable 接口中的 run()方法实现的，其中 paint()方法中添加的代码用于绘制文本，run()方法中的代码用于随机获得颜色的 RGB 值，并创建 Color 颜色对象。

（1）使用 Random 类的实例生成伪随机数流，并使用该类的 nextInt()方法随机产生颜色的 RGB 值。

（2）使用从 Graphics 类继承的 setColor()方法设置颜色，该颜色是通过 RGB 值创建的 Color 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ChangeColorTextFrame 窗体类。

（3）在 ChangeColorTextFrame 窗体类中创建内部面板类 ChangeColorTextPanel，在面板类中重写 JComponent 类的 paint()方法，并实现 Runnable 接口的 run()方法，从而实现会变色文字的绘制。

（4）将内部面板类 ChangeColorTextPanel 的实例添加到窗体类 ChangeColorTextFrame 的内容面板上，用于在窗体上显示会变色的文字，代码如下：

```
class ChangeColorTextPanel extends JPanel implements Runnable {        //创建内部面板类
    Color color = new Color(0,0,255);
    public void paint(Graphics g) {                                     //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                                 //转换为 Graphics2D 类型
        String value = "《视频学 Java 编程》";                           //绘制的文本
        int x = 2;                                                      //文本位置的横坐标
        int y = 90;                                                     //文本位置的纵坐标
        Font font = new Font("楷体", Font.BOLD, 40);                    //创建字体对象
        g2.setFont(font);                                               //设置字体
        g2.setColor(color);                                             //设置颜色
        g2.drawString(value, x, y);                                     //绘制文本
    }
    public void run() {
        Random random = new Random();                                   //创建随机数对象
        while(true){
```

```
                int R = random.nextInt(256);                    //随机产生颜色的 R 值
                int G = random.nextInt(256);                    //随机产生颜色的 G 值
                int B = random.nextInt(256);                    //随机产生颜色的 B 值
                color = new Color(R,G,B);                       //创建颜色对象
                repaint();                                      //调用 paint()方法
                try {
                    Thread.sleep(300);                          //休眠 300 毫秒
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
```

秘笈心法

心法领悟 043：区别进程和线程。

进程和线程是容易混淆的两个概念。实际上，一个运行的程序就是一个进程，一个进程可以包含多个线程，各个线程协调工作，交替执行，从而实现整个程序的功能。

实例 044 水印文字特效

光盘位置：光盘\MR\044

中级

趣味指数：★★★★

实例说明

水印文字是通过改变绘图上下文的透明度实现的，本实例将演示水印文字的实现。运行程序，将在窗体上绘制图片，并为图片添加水印文字，效果如图 3.6 所示。

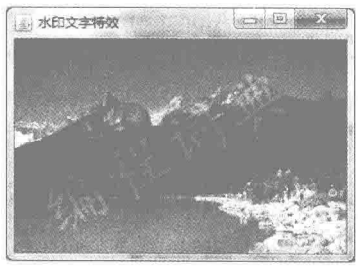


图 3.6 水印文字特效

关键技术

本实例主要是通过 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的 AlphaComposite 对象实现的。

（1）使用 AlphaComposite 类获得表示透明度的 AlphaComposite 对象，该对象使用 AlphaComposite 类的字段 SrcOver 调用 derive()方法获得，该方法的定义如下：

```
public AlphaComposite derive(float alpha)
```

参数说明

❶ alpha：闭区间 0.0f~1.0f 之间的一个浮点数字，为 0.0f 时完全透明；为 1.0f 时不透明。

❷ 返回值：表示透明度的 AlphaComposite 对象。

（2）使用 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的 AlphaComposite 对象，该方法的定义如下：

```
public abstract void setComposite(Composite comp)
```

参数说明

comp：表示透明度的 AlphaComposite 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 WatermarkTextFrame 窗体类。

（3）在 WatermarkTextFrame 窗体类中创建内部面板类 WatermarkTextPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics2D 类的 setComposite()方法设置透明度，然后绘制文本实现水印文字的绘制。

（4）将内部面板类 WatermarkTextPanel 的实例添加到窗体类 WatermarkTextFrame 的内容面板上，用于在窗体上显示绘制的水印文字，代码如下：

```
class WatermarkTextPanel extends JPanel {
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D)g;                                  //获得 Graphics2D 对象
        g2.drawImage(img, 0, 0, 300, 237, this);                        //绘制图像
        g2.rotate(Math.toRadians(-30));                                 //旋转绘图上下文对象
        Font font = new Font("楷体",Font.BOLD,60);                      //创建字体对象
        g2.setFont(font);                                               //指定字体
        g2.setColor(Color.WHITE);                                       //指定颜色
        AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.3f);     //获得表示透明度的 AlphaComposite 对象
        g2.setComposite(alpha);                                         //指定 AlphaComposite 对象
        g2.drawString("编程词典", -60, 180);                            //绘制文本，实现水印
    }
}
```

秘笈心法

心法领悟 044：获得表示透明度对象的另一种方法。

除了使用本实例提供的方法获得表示透明度的 AlphaComposite 对象以外，还可以使用 AlphaComposite 类提供的 getInstance()方法获得 AlphaComposite 对象，如语句 AlphaComposite.getInstance(AlphaComposite.SRC_OVER)就获得了一个规则是 AlphaComposite.SRC_OVER 的 AlphaComposite 对象（与本实例使用的字段 SrcOver 具有相同的规则），通过该对象调用 derive()方法，可以获得具有指定透明度的 AlphaComposite 对象。

实例 045　顺时针旋转文字

光盘位置：光盘\MR\045　　中级　　趣味指数：★★★★

实例说明

本实例演示如何在 Java 中顺时针旋转文字。运行程序，效果如图 3.7 所示。

图 3.7　顺时针旋转文字

关键技术

本实例主要是通过文本组件的 CaretListener 监听器实现，该监听器用于监听文本组件插入符的位置是否更改，如果更改了就会执行 caretUpdate()方法，同时，为了能够顺时针绘制文本，需要计算出每个文字应该在哪个角度进行绘制，因此可以用 360 除以文字个数，计算出每个文字的角度。

（1）监听器接口 CaretListener 的 caretUpdate()方法用于监听文本组件插入符的位置是否改变，该方法的定义如下：

```
void caretUpdate(CaretEvent e)
```

参数说明

e：插入符事件。

（2）计算每个文字的绘制角度，假设有 6 个字，要计算每个字的绘制角度，就可以用 360/6 计算出相邻两个字的角度差是 60° 。例如：

```
String s = "全能编程词典";                    //声明字符串
int len = s.length;                          //获得字符串的长度，即字符个数
double = 360D / len;                         //计算出每个字的角度
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ClockwiseTextFrame 窗体类。

（3）在 ClockwiseTextFrame 窗体类中创建内部面板类 ClockwiseTextPanel，并重写 JComponent 类的 paint()方法，在该方法中添加代码，实现顺时针旋转文本的功能。

（4）将内部面板类 ClockwiseTextPanel 的实例添加到窗体类 ClockwiseTextFrame 的内容面板上，用于在窗体上顺时针绘制文本，面板类 ClockwiseTextPanel 的代码如下：

```
class ClockwiseTextPanel extends JPanel {                    //创建内部面板类
    private String text;
    public ClockwiseTextPanel() {
        setOpaque(false);                                    //设置面板为透明
        setLayout(null);                                     //设置为绝对布局
    }
    public String getText() {
        return text;                                         //获得成员变量的值
    }
    public void setText(String text) {
        this.text = text;                                    //为成员变量赋值
        repaint();                                           //调整 paint()方法
    }
    public void paint(Graphics g) {                          //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                      //获得 Graphics2D 的实例
        int width = getWidth();                              //获得面板的宽度
        int height = getHeight();                            //获得面板的高度
        if (text != null) {
            char[] array = text.toCharArray();               //将文本转换为字符数组
            int len = array.length * 5;                      //定义圆的半径，同时可以调整文字的距离
            Font font = new Font("宋体", Font.BOLD, 22);     //创建字体
            g2.setFont(font);                                //设置字体
            double angle = 0;                                //定义初始角度
            for (int i = 0; i < array.length; i++) {         //遍历字符串中的字符
                if (i == 0) {
                    g2.setColor(Color.BLUE);                 //第 1 个字符用蓝色
                } else {
                    g2.setColor(Color.BLACK);                //其他字符用黑色
                }
                int x = (int) (len * Math.sin(Math.toRadians(angle + 270)));   //计算每个文字的横坐标位置
                int y = (int) (len * Math.cos(Math.toRadians(angle + 270)));   //计算每个文字的纵坐标位置
                g2.drawString(array[i] + "", width / 2 + x, height / 2 - y);   //绘制字符
                angle = angle + 360d / array.length;         //改变角度
```

```
            }
        }
    }
}
```

（5）本实例使用监听器接口 CaretListener 对文本组件的插入符进行监听，当插入符位置改变后，程序就会响应插入符事件，并执行 caretUpdate()方法，文本组件的插入符事件代码如下：

```
textField.addCaretListener(new CaretListener() {
    public void caretUpdate(CaretEvent arg0) {
        String text = textField.getText();                    //获取文本框字符串
        clockwiseTextPanel.setText(text);                     //为面板中的 text 变量赋值
    }
});
```

秘笈心法

心法领悟 045：正确计算每个字的绘制位置。

为了能够正确计算每个字的绘制位置，需要使用正弦函数和余弦函数，方法是先定义一个变量，用于表示圆的半径，然后分别使用正、余弦函数求文字绘制角度的正弦值，并乘以半径，即可计算出文字绘制点到面板中心的距离，并与面板中心的横纵坐标值相加，最终计算出文字绘制位置的横纵坐标值。

实例 046　动态绘制文本

光盘位置：光盘\MR\046　　高级　　趣味指数：★★★★★

实例说明

本实例演示如何在 Java 中利用绘图技术动态绘制文本，也就是将一个文件中的文本逐字绘制到程序界面上。运行程序，将在窗体上动态绘制文本，效果如图 3.8 所示。

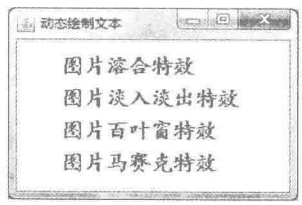


图 3.8　动态绘制文本

关键技术

本实例的实现主要是使用线程和 BufferedReader 缓冲流从指定文件中读取一个字符，然后绘制该字符，并改变下一个字符绘制点的 x、y 坐标值，最终完成动态绘制文本的功能。

（1）使用 System 类的 getProperty()方法，并为其传递实参字符串 user.dir，这样就可以获得项目的当前路径，getProperty()方法的定义如下：

```
public static String getProperty(String key)
```

参数说明

❶ key：系统属性的名称。

❷ 返回值：返回系统属性的字符串值，如果没有指定键的属性，则返回 null。

（2）使用 BufferedReader 类的 read()方法，从文本中读取一个字符，该方法的定义如下：

```
public int read() throws IOException
```

参数说明

❶ 返回值：作为一个范围从 0~65535 整数读入的字符，如果已到达流末尾，则返回-1。

❷ IOException：如果发生 I/O 错误，则抛出 IOException 异常。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DynamicDrawText 窗体类。

（3）在 DynamicDrawText 窗体类中创建内部面板类 DynamicDrawTextPanel，实现 Runnable 接口中的 run()方法，并重写 JComponent 类的 paint()方法，在该方法中完成动态绘制文本的操作。

（4）将内部面板类 DynamicDrawTextPanel 的实例添加到窗体类 DynamicDrawText 的内容面板上，用于在窗体上动态绘制文本，代码如下：

```
class DynamicDrawTextPanel extends JPanel implements Runnable {
    private BufferedReader read;                                    //声明缓冲流对象
    int x = 20;                                                     //起始点的 x 坐标
    int y = 30;                                                     //起始点的 y 坐标
    String value = "";
    public DynamicDrawTextPanel(){
        String projectPath = System.getProperty("user.dir");        //获得当前项目
        String filePath = projectPath + "/src/com/zzk/dyn.txt";     //获得项目中 loadText.java 文件的完整路径
        InputStream in = null;
        try {
            in = new FileInputStream(filePath);                     //创建输入流对象
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        read = new BufferedReader(new InputStreamReader(in));       //创建缓冲流对象
    }
    public void paint(Graphics g) {
        Font font = new Font("华文楷体", Font.BOLD, 20);            //创建字体对象
        g.setFont(font);                                            //指定字体
        g.setColor(Color.RED);                                      //指定颜色
        g.drawString(value, x, y);                                  //绘制文本
    }
    public void run() {
        int len = 0;                                                //存储读取的字符
        try {
            while ((len = read.read()) != -1) {                     //读取内容
                Thread.sleep(400);                                  //当前线程休眠 400 毫秒
                value = String.valueOf((char) len);                 //获得读取的内容
                if (value.equals("\n") || value.equals("\r")) {     //是回车或换行符
                    x = 20;                                         //下一行起始点的 x 坐标
                    y += 15;                                        //下一行文本的 y 坐标
                } else {                                            //不是回车或换行符
                    x += 20;                                        //当前行下一个字的 x 坐标
                }
                repaint();                                          //调用 paint()方法
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

秘笈心法

心法领悟 046：实现字节流到字符流的转换。

在进行程序开发时，有时需要将字节流转换为字符流，这可以通过 InputStreamReader 类来实现，该类是字节流通向字符流的桥梁，为其构造方法传递一个 InputStream 字节流对象，即可获得字节流的 InputStreamReader 对象，由于 InputStreamReader 类是 Reader 类的子类，所以也就获得了一个 Reader 对象，完成字节流到字符流的转换。

实例 047 中文验证码

光盘位置：光盘\MR\047　　中级　　趣味指数：★★★★

实例说明

为了保证软件的安全性，通常要求在登录界面中输入验证码，为此本实例演示了如何实现中文验证码。运行程序，输入正确的用户名 mrsoft、密码 mrsoft 和中文验证码即可登录，如图 3.9 所示。

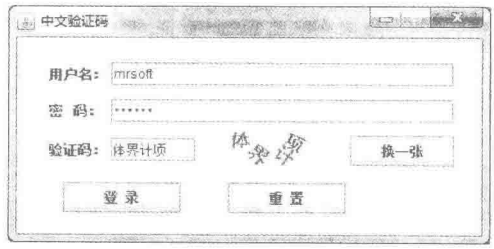


图 3.9　中文验证码

关键技术

本实例以字符串存储作为验证码的汉字，然后使用 Random 类获得随机数，作为字符串中字符的索引值，并使用 String 类的 substring()方法，从字符串中截取一个汉字作为验证码。

使用 String 类的 substring()方法，可截取字符串中的字符，该方法的定义如下：

```
public String substring(int beginIndex, int endIndex)
```

参数说明

❶ beginIndex：起始索引（包括）。

❷ endIndex：结束索引（不包括）。

❸ 返回值：起始索引和终止索引之间的子字符串。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类和一个继承 Jpanel 类的 ChineseCodePanel 面板类。

（3）在面板类 ChineseCodePanel 中重写 JComponent 类的 paint()方法，在该方法中向 BufferedImage 对象上绘制中文验证码，完成中文验证码的绘制。

（4）面板类 ChineseCodePanel 的 paint()方法代码如下：

```
public void paint(Graphics g) {
    String hanZi = "编程词典集学查用界面设计项目开发等内容于一体";      //定义验证码使用的汉字
    BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
            BufferedImage.TYPE_INT_RGB);                              //实例化 BufferedImage
    Graphics gs = image.getGraphics();                                //获取 Graphics 类的对象
    if (!num.isEmpty()) {
        num = "";                                                     //清空验证码
    }
    Font font = new Font("黑体", Font.BOLD, 20);                      //通过 Font 构造字体
    gs.setFont(font);                                                 //设置字体
    gs.fillRect(0, 0, WIDTH, HEIGHT);                                 //填充一个矩形
    //输出随机的验证文字
    for (int i = 0; i < 4; i++) {
        int index = random.nextInt(hanZi.length());                   //随机获得汉字的索引值
```

```
            String ctmp   = hanZi.substring(index,index+1);          //获得指定索引处的一个汉字
            num += ctmp;                                             //更新验证码
            Color color = new Color(20 + random.nextInt(120), 20 + random
                    .nextInt(120), 20 + random.nextInt(120));        //生成随机颜色
            gs.setColor(color);                                      //设置颜色
            Graphics2D gs2d = (Graphics2D) gs;                       //将文字旋转指定角度
            AffineTransform trans = new AffineTransform();           //实例化 AffineTransform
            trans.rotate(random.nextInt(45) * 3.14 / 180, 22 * i + 8, 7);
            float scaleSize = random.nextFloat() + 0.8f;             //缩放文字
            if (scaleSize > 1f)                                      //如果 scaleSize 大于 1，则等于 1
                scaleSize = 1f;
            trans.scale(scaleSize, scaleSize);                       //进行缩放
            gs2d.setTransform(trans);                                //设置 AffineTransform 对象
            gs.drawString(ctmp, WIDTH / 6 * i + 28, HEIGHT / 2);     //绘制出验证码
        }
        g.drawImage(image, 0, 0, null);                              //在面板中绘制出验证码
    }
```

秘笈心法

心法领悟 047：生成没有重复文字的验证码。

对于本实例生成的中文验证码，有时会出现重复的汉字，为了避免这种情况，可以使用 List 集合存储作为验证码的汉字，然后随机从 List 集合中提取汉字，并从集合中移除该汉字，再从剩余的元素中提取汉字并移除，重复上述过程即可获得没有重复汉字的中文验证码。

实例 048 图片验证码

光盘位置：光盘\MR\048　　中级　　趣味指数：★★★★

实例说明

本实例演示了如何实现图片验证码。运行程序，输入正确的用户名 mrsoft、密码 mrsoft 和验证码，即可登录，如图 3.10 所示。

图 3.10　图片验证码

关键技术

本实例通过 BufferedImage 类绘制图像，然后使用 Random 类随机获得 A~Z 的大写字母，并绘制到 BufferedImage 对象上，从而生成图片验证码。

（1）由于大写字母 A~Z 对应的 Unicode 字符集的编号为 65~90，所以要产生 65~90 之间的随机整数，可以使用下面的代码实现：

```
Random random = new Random();                                        //创建 Random 对象
int code = random.nextInt(26) + 65;                                  //生成 65~90 的随机整数
```

（2）通过强制类型转换将 65~90 的随机整数转换为大写字母，可以使用下面的代码实现：

```
Random random = new Random();                                        //创建 Random 对象
int code = random.nextInt(26) + 65;                                  //生成 65~90 的随机整数
char letter = (char) code;                                           //转换为大写字母
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类和一个继承 Jpanel 类的 ImageCodePanel 面板类。

（3）在面板类 ImageCodePanel 中重写 JComponent 类的 paint()方法，在该方法中向 BufferedImage 对象上绘制图片和验证码，完成图片验证码的绘制。

（4）面板类 ImageCodePanel 的 paint()方法代码如下：

```
public void paint(Graphics g) {
    BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
            BufferedImage.TYPE_INT_RGB);                          //实例化 BufferedImage
    Graphics gs = image.getGraphics();                            //获取 Graphics 类的对象
    if (!num.isEmpty()) {
        num = "";                                                 //清空验证码
    }
    Font font = new Font("黑体", Font.BOLD, 20);                  //通过 Font 构造字体
    gs.setFont(font);                                             //设置字体
    gs.fillRect(0, 0, WIDTH, HEIGHT);                             //填充一个矩形
    Image img = null;
    try {
        img = ImageIO.read(new File("src/img/image.jpg"));        //创建图像对象
    } catch (IOException e) {
        e.printStackTrace();
    }
    image.getGraphics().drawImage(img, 0, 0, WIDTH, HEIGHT, null);//在缓冲图像对象上绘制图像
    //输出随机的验证文字
    for (int i = 0; i < 4; i++) {
        char ctmp = (char) (random.nextInt(26) + 65);             //生成 A~Z 的字母
        num += ctmp;                                              //更新验证码
        Color color = new Color(20 + random.nextInt(120), 20 + random
                .nextInt(120), 20 + random.nextInt(120));         //生成随机颜色
        gs.setColor(color);                                       //设置颜色
        Graphics2D gs2d = (Graphics2D) gs;                        //将文字旋转指定角度
        AffineTransform trans = new AffineTransform();            //实例化 AffineTransform
        trans.rotate(random.nextInt(45) * 3.14 / 180, 22 * i + 8, 7);
        float scaleSize = random.nextFloat() + 0.8f;              //缩放文字
        if (scaleSize > 1f)
            scaleSize = 1f;                                       //如果 scaleSize 大于 1，则等于 1
        trans.scale(scaleSize, scaleSize);                        //进行缩放
        gs2d.setTransform(trans);                                 //设置 AffineTransform 对象
        gs.drawString(String.valueOf(ctmp), WIDTH / 6 * i + 28, HEIGHT / 2); //绘制出验证码
    }
    g.drawImage(image, 0, 0, null);                               //在面板中绘制出验证码
}
```

秘笈心法

心法领悟 048：避免绘图时出现屏幕闪现。

在绘图时，为了避免出现屏幕闪现，可以先将要绘制的内容绘制到 BufferedImage 对象上，然后再将 BufferedImage 对象绘制到绘图上下文上，如果还是出现屏幕闪现，可以重写 update(Graphics g)方法，并在该方法中调用 paint()方法，即可解决。

实例 049 带干扰线的验证码

高级

光盘位置：光盘\MR\049

趣味指数：★★★★

实例说明

本实例演示了如何实现带干扰线的验证码。运行程序，输入正确的用户名 mrsoft、密码 mrsoft 和验证码，

即可登录，如图 3.11 所示。

图 3.11 带干扰线的验证码

关键技术

本实例通过 BufferedImage 类绘制图像，然后绘制两条首尾相连的线段，最后使用 Random 类随机获得 A~Z 的大写字母，并绘制到 BufferedImage 对象上，从而生成带干扰线的验证码。

（1）使用 Graphics 类的 drawLine()方法绘制干扰线。

（2）使用 Graphics 类的 drawString()方法绘制随机生成的大写字母，完成带干扰线的验证码的绘制。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类和一个继承 Jpanel 类的 DisturbCodePanel 面板类。

（3）在面板类 DisturbCodePanel 中，重写 JComponent 类的 paint()方法，在该方法中向 BufferedImage 对象上绘制图片、干扰线和验证码，完成带干扰线验证码的绘制。

（4）面板类 DisturbCodePanel 的 paint()方法代码如下：

```
public void paint(Graphics g) {
    BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
            BufferedImage.TYPE_INT_RGB);                          //实例化 BufferedImage
    Graphics gs = image.getGraphics();                            //获取 Graphics 类的对象
    if (!num.isEmpty()) {
        num = "";                                                 //清空验证码
    }
    Font font = new Font("黑体", Font.BOLD, 20);                  //通过 Font 构造字体
    gs.setFont(font);                                             //设置字体
    gs.fillRect(0, 0, WIDTH, HEIGHT);                             //填充一个矩形
    Image img = null;
    try {
        img = ImageIO.read(new File("src/img/image.jpg"));        //创建图像对象
    } catch (IOException e) {
        e.printStackTrace();
    }
    image.getGraphics().drawImage(img, 0, 0, WIDTH, HEIGHT, null);          //在缓冲图像对象上绘制图像
    int startX1 = random.nextInt(20);                             //随机获取第一条干扰线起点的 x 坐标
    int startY1 = random.nextInt(20);                             //随机获取第一条干扰线起点的 y 坐标
    int startX2 = random.nextInt(30)+35;                          //随机获取第一条干扰线终点的 x 坐标，也是第二条干扰线起点的 x 坐标
    int startY2 = random.nextInt(10)+20;                          //随机获取第一条干扰线终点的 y 坐标，也是第二条干扰线起点的 y 坐标
    int startX3 = random.nextInt(30)+90;                          //随机获取第二条干扰线终点的 x 坐标
    int startY3 = random.nextInt(10)+5;                           //随机获取第二条干扰线终点的 y 坐标
    gs.setColor(Color.RED);
    gs.drawLine(startX1, startY1, startX2, startY2);               //绘制第一条干扰线
    gs.setColor(Color.BLUE);
    gs.drawLine(startX2, startY2, startX3, startY3);               //绘制第二条干扰线
    //输出随机的验证文字
    for (int i = 0; i < 4; i++) {
        char ctmp = (char) (random.nextInt(26) + 65);             //生成 A~Z 的字母
```

```
            num += ctmp;                                                    //更新验证码
            Color color = new Color(20 + random.nextInt(120), 20 + random
                    .nextInt(120), 20 + random.nextInt(120));               //生成随机颜色
            gs.setColor(color);                                             //设置颜色
            Graphics2D gs2d = (Graphics2D) gs;                              //将文字旋转指定角度
            AffineTransform trans = new AffineTransform();                  //实例化 AffineTransform
            trans.rotate(random.nextInt(45) * 3.14 / 180, 22 * i + 8, 7);
            float scaleSize = random.nextFloat() + 0.8f;                    //缩放文字
            if (scaleSize > 1f)
                scaleSize = 1f;                                             //如果 scaleSize 大于 1，则等于 1
            trans.scale(scaleSize, scaleSize);                              //进行缩放
            gs2d.setTransform(trans);                                       //设置 AffineTransform 对象
            gs.drawString(String.valueOf(ctmp), WIDTH / 6 * i + 28, HEIGHT / 2); //绘制出验证码
        }
        g.drawImage(image, 0, 0, null);                                     //在面板中绘制出验证码
    }
```

秘笈心法

心法领悟 049：使两条干扰线首尾相连。

在绘制干扰线时，为了使干扰线首尾相连，应该使第二条干扰线的起点与第一条干扰线的终点相同。

3.2 图 片 特 效

实例 050 纹理填充特效　　高级

光盘位置：光盘\MR\050　　趣味指数：★★★★

实例说明

在使用 Word 文档进行绘图时，可以对图形进行纹理填充，本实例使用 Java 的绘图技术，实现了图形的纹理填充效果。运行程序，效果如图 3.12 所示。

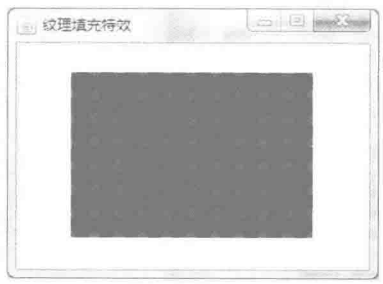


图 3.12　纹理填充特效

关键技术

本实例使用 TexturePaint 类创建纹理填充对象，然后使用 Graphics2D 类的 setPaint()方法，将该纹理填充对象设置为绘图上下文的 Paint 属性，这样再绘制填充图形时，将以 TexturePaint 类指定的纹理进行填充。

（1）使用 TexturePaint 类创建纹理填充对象，该类构造方法的定义如下：

```
public TexturePaint(BufferedImage txtr, Rectangle2D anchor)
```

参数说明

❶ txtr：绘制有纹理的 BufferedImage 对象。

❷ anchor：用于定位和复制纹理的 Rectangle2D 对象。

（2）使用 Graphics2D 类的 setPaint()方法，将纹理填充对象设置为绘图上下文的 Paint 属性。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 TextureFillFrame 窗体类。

（3）在 TextureFillFrame 窗体类中创建内部面板类 TextureFillPanel，并重写 JComponent 类的 paint()方法，在该方法中实现图形的纹理填充特效。

（4）将内部面板类 TextureFillPanel 的实例添加到窗体类 TextureFillFrame 的内容面板上，用于在窗体上显示填充有纹理的图形，代码如下：

```
class TextureFillPanel extends JPanel {
    public void paint(Graphics g) {
        BufferedImage image = new BufferedImage(200, 200,
                BufferedImage.TYPE_INT_RGB);                          //创建缓冲图像对象
        Graphics2D g2 = image.createGraphics();                       //获得缓冲图像对象的绘图上下文对象
        g2.setColor(Color.BLUE);                                      //设置颜色
        g2.fillRect(0,0,90,90);                                       //绘制填充矩形
        g2.setColor(Color.RED);                                       //设置颜色
        g2.fillOval(95,95,90,90);                                     //绘制带填充色的圆形
        Rectangle2D rect = new Rectangle2D.Float(10, 10, 20, 20);     //创建 Rectangle2D 对象
        TexturePaint textPaint = new TexturePaint(image,rect);        //创建纹理填充对象
        Graphics2D graphics2 = (Graphics2D)g;                         //转换 paint()方法的绘图上下文对象
        graphics2.setPaint(textPaint);                                //为绘图上下文对象设置纹理填充对象
        Rectangle2D.Float ellipse2 = new Rectangle2D.Float(45, 25, 200, 140); //创建矩形对象
        graphics2.fill(ellipse2);                                     //绘制填充纹理的矩形
    }
}
```

秘笈心法

心法领悟 050：改变纹理填充图案的大小。

使用 TexturePaint 类的构造方法创建填充纹理对象时，可以使用构造方法的第 2 个参数 Rectangle2D 来指定纹理填充图案的大小，也就是由 Rectangle2D 类创建的矩形的大小。

实例 051 水波效果的图片

光盘位置：光盘\MR\051　　中级　　趣味指数：★★★★

实例说明

本实例演示在 Java 中绘制图像时，如何实现水波效果的图片特效。运行程序，效果如图 3.13 所示。

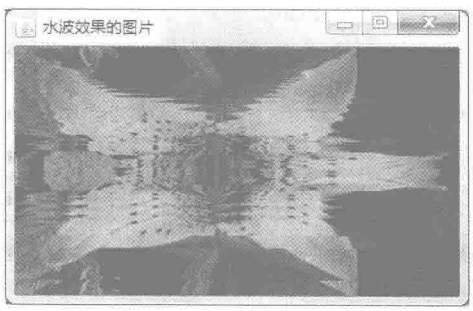


图 3.13　水波效果的图片

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 copyArea()方法复制图像区域，实现最终的水波效果。

使用 Graphics 类的 copyArea()方法可以复制图像区域，该方法的定义如下：

```
public abstract void copyArea(int x, int y, int width, int height, int dx, int dy)
```

参数说明

❶ x：源矩形的 x 坐标。

❷ y：源矩形的 y 坐标。

❸ width：源矩形的宽度。

❹ height：源矩形的高度。

❺ dx：复制像素的水平距离。

❻ dy：复制像素的垂直距离。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 WaterWavePictureFrame 窗体类。

（3）在 WaterWavePictureFrame 窗体类中创建内部面板类 WaterWavePicturePanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 copyArea()方法复制图像区域，实现水波效果。

（4）将内部面板类 WaterWavePicturePanel 的实例添加到窗体类 WaterWavePictureFrame 的内容面板上，用于在窗体上显示水波效果的图片，代码如下：

```
class WaterWavePicturePanel extends JPanel {
    private Graphics graphics;                                  //Graphics 对象
    private Graphics waveGraphics;                              //绘制水波的 Graphics 对象
    private Image oldImage;                                     //原图像对象
    private Image waveImage;                                    //声明表示水波效果的图像对象
    private int currentImage, imageWidth, imageHeight;
    private boolean isImageLoaded;                              //表示图片是否被加载的标记

    public void paint(Graphics g) {
        drawWaterWave();
        if (waveImage != null) {
            g.drawImage(waveImage, -currentImage * imageWidth, 0, this);   //绘制图像
        }
        g.clearRect(imageWidth, 0, imageWidth * 4, imageHeight * 2);       //清除显示区域右侧的内容
    }

    public void drawWaterWave() {
        currentImage = 0;
        if (!isImageLoaded) {                                   //如果未加载图片
            graphics = getGraphics();                           //获得绘图上下文对象
            MediaTracker mediatracker = new MediaTracker(this);//创建媒体跟踪对象
            URL imgUrl = WaterWavePictureFrame.class
                    .getResource("/img/image.jpg");             //获取图片资源的路径
            oldImage = Toolkit.getDefaultToolkit().getImage(imgUrl);   //获取图像资源
            mediatracker.addImage(oldImage, 0);                 //添加图片
            try {
                mediatracker.waitForAll();                      //加载图片
                isImageLoaded = !mediatracker.isErrorAny();     //是否有错误发生
            } catch (InterruptedException ex) {
            }
            if (!isImageLoaded) {                               //图片加载失败
                graphics.drawString("图片加载错误", 10, 40);     //绘制错误信息
                return;
            }
            imageWidth = oldImage.getWidth(this);               //得到图像宽度
```

```
            imageHeight = oldImage.getHeight(this);                    //得到图像高度
            createWave();                                              //创建水波效果
        }
    }

    public void createWave() {
        Image img = createImage(imageWidth, imageHeight);              //以图像宽度和高度创建图像对象
        Graphics g = null;
        if (img != null) {
            g = img.getGraphics();                                     //得到 Image 对象的 Graphics 对象
            g.drawImage(oldImage, 0, 0, this);                         //绘制原图像对象
            for (int i = 0; i < imageHeight; i++) {
                g.copyArea(0, imageHeight - 1 - i, imageWidth, 1, 0,
                        -imageHeight + 1 + (i * 2));                   //复制图像区域
            }
        }
        waveImage = createImage(13 * imageWidth, imageHeight);         //得到水波效果的图像对象
        if (waveImage != null) {
            waveGraphics = waveImage.getGraphics();                    //得到水波效果的绘图上下文对象
            waveGraphics.drawImage(img, 12 * imageWidth, 0, this);     //绘制图像
            int j = 0;
            while (j < 12) {
                simulateWaves(waveGraphics, j);                        //调用方法，模拟水波效果
                j++;
            }
        }
    }

    public void simulateWaves(Graphics g, int i) {                     //水波效果模拟
        int j = (12 - i) * imageWidth;                                 //计算复制像素的水平距离
        int waveHeight = imageHeight / 16;                             //计算水波高度
        for (int h = 0; h < imageHeight; h++) {
            int k = (int) ((waveHeight * (h + 28) * Math.sin(waveHeight
                    * (imageHeight - h) / (h + 1))) / imageHeight);    //计算复制像素的垂直距离
            if (h < -k) {
                g.copyArea(12 * imageWidth, h, imageWidth, 1, -j, 0);  //复制图像区域，形成水波
            } else {
                g.copyArea(12 * imageWidth, h + k, imageWidth, 1, -j, -k);  //复制图像区域，形成水波
            }
        }
    }
}
```

秘笈心法

心法领悟 051：防止图像加载失败。

在进行图像处理时，为了防止图像加载失败，可以使用 MediaTracker 类对图像进行跟踪，方法是使用该类的 addImage()方法加载图像，然后使用 waitForAll()方法开始加载所有图像，再使用 isErrorAny()方法判断图像加载是否出错，如果没有出错就进行图像处理，从而防止图像加载失败。

实例 052 局部图像放大

光盘位置：光盘\MR\052　　中级　　趣味指数：★★★

实例说明

本实例使用 Java 的绘图技术实现图像局部区域的放大。运行程序，拖动鼠标选择图像的局部区域，释放鼠标后，即可看到选择区域的图像被放大了，效果如图 3.14 所示。

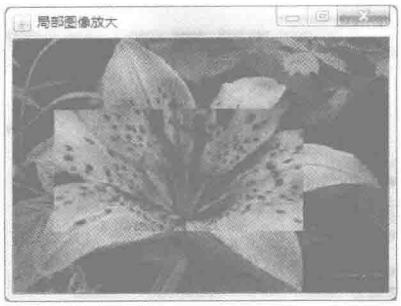


图 3.14　局部图像放大的效果

关键技术

本实例主要是通过 Robot 类的 createScreenCapture()方法，使用 BasicStroke 类创建虚线对象，并结合 Graphics 类的 drawRect()方法实现绘制选区选择图像，并使用 drawImage()方法将选择区域的图像放大，然后绘制到绘图上下文，从而实现图像局部放大的功能。

（1）使用 BasicStroke 类重载的构造方法创建笔画对象，并指定笔画为虚线模式。

```
public BasicStroke(float width, int cap, int join, float miterlimit, float[] dash, float dash_phase)
```

（2）使用 Graphics 类的 drawRect()方法绘制矩形，以获得选择区域。

（3）使用 Robot 类的 createScreenCapture()方法，用选择区域的图片创建缓冲图像对象，该方法的定义如下：

```
public BufferedImage createScreenCapture(Rectangle screenRect)
```

参数说明

❶ screenRect：屏幕上被截取的矩形区域。

❷ 返回值：从屏幕上截取的缓冲图像对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PartZoomInImageFrame 窗体类。

（3）在 PartZoomInImageFrame 窗体类中创建内部面板类 PartZoomInPanel，并重写 JComponent 类的 paint()方法，在该方法中绘制原始图像、选择区域和局部放大的图像。

（4）将内部面板类 PartZoomInPanel 的实例添加到窗体类 PartZoomInImageFrame 的内容面板上，用于在窗体上显示原始图像、选择区域和局部放大的图像，代码如下：

```
class PartZoomInPanel extends JPanel {                          //创建绘制原图像的面板类
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        g2.drawImage(img, 0, 0, this.getWidth(), this.getHeight(), this);  //绘制图像
        g2.setColor(Color.WHITE);
        if (flag) {
            float[] arr = { 5.0f };                             //创建虚线模式的数组
            BasicStroke stroke = new BasicStroke(1, BasicStroke.CAP_BUTT,
                    BasicStroke.JOIN_BEVEL, 1.0f, arr, 0);      //创建宽度是 1 的平头虚线笔画对象
            g2.setStroke(stroke);                               //设置笔画对象
            g2.drawRect(pressPanelX, pressPanelY, releaseX - pressX,
                    releaseY - pressY);                         //绘制矩形选区
        }
        if (mouseFlag) {                                        //条件为真
            int zoomX = pressPanelX - (releaseX - pressX) / 4;  //放大图像绘制点的 x 坐标
            int zoomY = pressPanelY - (releaseY - pressY) / 4;  //放大图像绘制点的 y 坐标
            if (zoomX <= 0) {
                zoomX = 0;                                      //坐标值小于等于 0，让坐标值为 0
            }
            if (zoomY <= 0) {
                zoomY = 0;                                      //坐标值小于等于 0，让坐标值为 0
```

```
                }
                g.drawImage(buffImage, zoomX, zoomY,
                        (int) ((releaseX - pressX) * 1.5f),
                        (int) ((releaseY - pressY) * 1.5f), this);          //绘制放大后的局部图像
            }
        }
    }
```

秘笈心法

心法领悟 052：获得鼠标指针在控件和屏幕上的坐标。

通过 MouseEvent 类的 getX()和 getY()方法，可以分别获得鼠标指针在控件上的 x 坐标和 y 坐标；使用 MouseEvent 类的 getXOnScreen()和 getYOnScreen()方法，可以分别获得鼠标指针在屏幕上的 x 坐标和 y 坐标。

实例 053 图片半透明特效

光盘位置：光盘\MR\053　　中级　　趣味指数：★★★

实例说明

本实例使用 Java 的绘图技术实现了图片的半透明效果。运行程序，窗体上显示不透明的图像，单击窗体上的“半透明”按钮，图片将显示为半透明效果，如图 3.15 所示。

图 3.15　图片半透明的效果

关键技术

本实例主要是通过 Graphics2D 类的 setComposite()方法为绘图上下文指定表示透明度的 AlphaComposite 对象，从而实现了图片的半透明效果。

（1）使用 AlphaComposite 类获得表示透明度的 AlphaComposite 对象，该对象使用 AlphaComposite 类的字段 SrcOver 调用 derive()方法，并指定透明度即可获得 AlphaComposite 对象。

（2）使用 Graphics2D 类的 setComposite()方法为绘图上下文指定表示透明度的 AlphaComposite 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 TranslucenceImageFrame 窗体类。

（3）在 TranslucenceImageFrame 窗体类中创建内部面板类 TranslucenceImagePanel，并重写 JComponent 类的 paint()方法，在该方法中实现图像透明度的调整。

（4）将内部面板类 TranslucenceImagePanel 的实例添加到窗体类 TranslucenceImageFrame 的内容面板上，用于在窗体上显示设置了透明度的图像，面板类 TranslucenceImagePanel 的代码如下：

```
class TranslucenceImagePanel extends JPanel {
    public void paint(Graphics g) {
```

```
            Graphics2D g2 = (Graphics2D) g;                          //获得 Graphics2D 对象
            g2.clearRect(0, 0,   getWidth(), getHeight());           //清除绘图上下文的内容
            g2.setComposite(alpha);                                  //指定 AlphaComposite 对象
            g2.drawImage(img, 0, 0, getWidth(), getHeight(), this);  //绘制图像
        }
    }
```

（5）“半透明”按钮用于实现图像的半透明效果，其事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        alpha = AlphaComposite.SrcOver.derive(0.5f);                 //获得表示半透明的 AlphaComposite 对象
        translucencePanel.repaint();                                 //调用 paint()方法
    }
});
```

（6）“不透明”按钮用于实现图像的半透明效果，其事件代码如下：

```
button_1.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        alpha = AlphaComposite.SrcOver.derive(1.0f);                 //获得表示不透明的 AlphaComposite 对象
        translucencePanel.repaint();                                 //调用 paint()方法
    }
});
```

秘笈心法

心法领悟 053：实现图片水印。

与水印文字相似，在窗体上绘制一个图像，然后调整绘图上下文的透明度，再将另一个图像绘制到该绘图上下文对象上，即可实现图片水印效果。

实例 054 图片溶合特效

光盘位置：光盘\MR\054　　中级　　趣味指数：★★★

实例说明

本实例使用 Java 的绘图技术实现了两幅图片的溶合效果。运行程序，窗体上将显示两幅图片的溶合特效，改变窗体上滑块的位置，可对两幅图片的溶合效果进行调整，效果如图 3.16 所示。

图 3.16　图片溶合特效

关键技术

本实例主要是通过 Graphics2D 类的 setComposite()方法为绘图上下文设置表示透明度的 AlphaComposite 对象，从而实现了两幅图片溶合的效果。

（1）使用 AlphaComposite 类获得表示透明度的 AlphaComposite 对象，该对象使用 AlphaComposite 类的字段 SrcOver 调用 derive()方法，并指定透明度即可获得 AlphaComposite 对象。

（2）使用 Graphics2D 类的 setComposite()方法为绘图上下文设置表示透明度的 AlphaComposite 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PictureMixFrame 窗体类。

（3）在 PictureMixFrame 窗体类中创建内部面板类 PictureMixPanel，并重写 JComponent 类的 paint()方法，在该方法中实现图像的溶合效果。

（4）将内部面板类 PictureMixPanel 的实例添加到窗体类 PictureMixFrame 的内容面板上，用于在窗体上显示溶合效果的图像，面板类 PictureMixPanel 的代码如下：

```
class PictureMixPanel extends JPanel {
    boolean flag = true;                                          //定义标记变量，用于控制 x 的值
    AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.5f); //获得表示透明度的 AlphaComposite 对象
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;                           //获得 Graphics2D 对象
        g2.drawImage(img1, 0, 0,  getWidth(), getHeight(), this); //绘制图像
        float value = slider.getValue();                          //滑块组件的取值
        float alphaValue = value / 100;                           //计算透明度的值
        alpha = AlphaComposite.SrcOver.derive(alphaValue);        //获得表示透明度的 AlphaComposite 对象
        g2.setComposite(alpha);                                   //指定 AlphaComposite 对象
        g.drawImage(img2, 0, 0, getWidth(), getHeight(), this);   //绘制调整透明度后的图片
    }
}
```

（5）通过滑块控件，可以调整图片的溶合效果，滑块控件的事件代码如下：

```
slider.addChangeListener(new ChangeListener() {
    public void stateChanged(final ChangeEvent e) {
        pictureMixPanel.repaint();                                //重新调用面板类的 paint()方法
    }
});
```

秘笈心法

心法领悟 054：滑块值与透明度值的转换。

滑块的位置值是整数（默认是 0~100 之间的整数），而表示透明度的值则是小数（在 0.0f~1.0f 之间），因此可以用滑块值除以 100，将其转换为小数，从而实现滑块值与透明度值之间的转换。

实例 055 以椭圆形显示图像

光盘位置：光盘\MR\055　　中级　　趣味指数：★★

实例说明

在窗体上显示的图像通常都以矩形显示，本实例使用 Java 的绘图技术实现以椭圆形显示图像。运行程序，效果如图 3.17 所示。

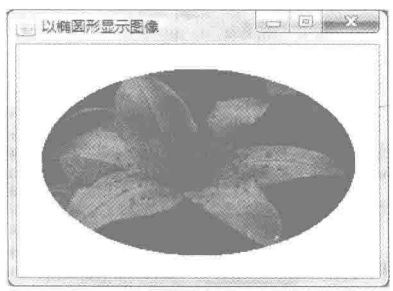


图 3.17　以椭圆形显示图像

关键技术

本实例主要是通过图形区域的减运算，将矩形区域与椭圆形区域相减，并用运算结果覆盖图像，从而实现以椭圆形显示图像的功能。

（1）使用 Area 类的构造方法封装图形对象。

（2）使用 Area 类的 subtract()方法对封装的图形对象进行减运算。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawEllipseImageFrame 窗体类。

（3）在 DrawEllipseImageFrame 窗体类中创建内部面板类 EllipseImagePanel，并重写 JComponent 类的 paint()方法，在该方法中实现以椭圆形显示图像的功能。

（4）将内部面板类 EllipseImagePanel 的实例添加到窗体类 DrawEllipseImageFrame 的内容面板上，用于在窗体上显示椭圆形图像，代码如下：

```
class EllipseImagePanel extends JPanel {
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D)g;
        g2.drawImage(img, 20, 20, 260, 160, this);                                  //绘制图像
        Rectangle2D.Float rectangle = new Rectangle2D.Float(0, 0, getWidth(),getHeight());  //创建矩形对象
        Ellipse2D.Float ellipse = new Ellipse2D.Float(20, 20, 260, 160);            //创建椭圆形对象
        Area area1 = new Area(rectangle);                                           //创建区域矩形
        Area area2 = new Area(ellipse);                                             //创建区域椭圆
        area1.subtract(area2);                                                      //两个区域形状进行减运算
        g2.setColor(getBackground());                                               //设置绘图上下文的颜色为面板的背景颜色
        g2.fill(area1);                                                             //绘制减运算后的区域形状
    }
}
```

秘笈心法

心法领悟 055：以任何形状显示图像。

本实例实现了以椭圆形显示图像，对本实例进行扩展，可以将不同图形进行加、减、交和异或等运算，得到所需要的形状，然后用该形状覆盖原图像，即可以任何形状显示图像。

实例 056　图片百叶窗特效

光盘位置：光盘\MR\056　　中级　　趣味指数：★★★

实例说明

本实例演示了如何利用 Java 的绘图技术实现图片百叶窗特效。运行程序，将在窗体上显示图片百叶窗效果，如图 3.18 所示。

图 3.18　图片百叶窗特效

关键技术

本实例主要是通过在缓冲图像对象上绘制直线，并对缓冲图像对象进行模糊处理实现的，对图像进行模糊处理需要用到 Kernel 类和 ConvolveOp 类。

（1）Kernel 类定义了一个矩阵，用于描述指定的像素及其周围像素如何影响过滤操作输出图像中像素位置的计算值，其构造方法的定义如下：

```
public Kernel(int width, int height, float[] data)
```

参数说明

❶ width：当前 kernel 的宽度。

❷ height：当前 kernel 的高度。

❸ data：以行优先顺序提供的 Kernel 数据。

（2）ConvolveOp 类实现从源到目标的卷积，是一种通过输入像素来计算输出像素的空间运算，方法是将核与输入像素邻域相乘。这种运算使得直接邻域可按核数学指定的方式影响输出像素，其构造方法定义如下：

```
public ConvolveOp(Kernel kernel)
```

参数说明

kernel：指定的 Kernel。

（3）ConvolveOp 类提供了一个 filter()方法，可以对缓冲图像进行过滤，实现对图像的特殊处理，如模糊、照亮边缘等，该方法的定义如下：

```
public final BufferedImage filter(BufferedImage src, BufferedImage dst)
```

参数说明

❶ src：要过滤的源 BufferedImage。

❷ dst：已过滤的 src 的目标 BufferedImage 或为 null。

❸ 返回值：对缓冲图像进行处理后的新 BufferedImage 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ShutterFrame 窗体类。

（3）在 ShutterFrame 窗体类中创建内部面板类 ShutterPanel，并重写 JComponent 类的 paint()方法，在该方法中进行图片百叶窗效果的绘制。

（4）将内部面板类 ShutterPanel 的实例添加到窗体类 ShutterFrame 的内容面板上，用于在窗体上显示图片百叶窗。面板类 ShutterPanel 的代码如下：

```
class ShutterPanel extends JPanel {
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        g2.drawImage(img, 0, 0, this);                          //绘制图像对象
        int y = 5;                                              //直线绘制点的 y 坐标
        int space = 10;                                         //下一条直线的偏移量
        Line2D.Float line = null;
        image = new BufferedImage(getWidth() + 10, getHeight(),
                BufferedImage.TYPE_INT_ARGB);                   //创建缓冲图像对象
        Graphics2D gs2d = (Graphics2D) image.getGraphics();     //获得缓冲图像对象的 Graphics2D 对象
        BasicStroke stroke = new BasicStroke(7);                //创建宽度是 7 的笔画对象
        gs2d.setStroke(stroke);                                 //设置笔画对象
        gs2d.setColor(Color.WHITE);                             //指定颜色
        while (y <= getHeight()) {
            line = new Line2D.Float(0, y, getWidth(), y);       //创建直线对象
            gs2d.draw(line);                                    //在缓冲图像对象上绘制直线
            y = y + space;                                      //计算下一条直线的 y 坐标
        }
        for (int i = 0; i < 3; i++) {                           //该 for 循环，实现 3 次模糊
            float[] elements = new float[9];                    //定义表示像素分量的数组
```

```
                for (int j = 0; j < 9; j++) {
                    elements[j] = 0.11f;                    //为数组赋值
                }
                convolve(elements);                         //调用方法，实现模糊功能
            }
            g2.drawImage(image, 0, 0, this);                //绘制缓冲图像对象
        }
    }
```

在面板类 ShutterPanel 中使用的 convolve()方法，用于实现模糊直线的功能，该方法是窗体类 ShutterFrame 中的成员方法，其代码如下：

```
    private void convolve(float[] elements) {
        Kernel kernel = new Kernel(3, 3, elements);         //创建 Kernel 对象
        ConvolveOp op = new ConvolveOp(kernel);             //创建 ConvolveOp 对象
        if (image == null) {
            return;
        }
        image = op.filter(image, null);                     //过滤缓冲图像对象
        repaint();                                          //调用 paint()方法
    }
```

秘笈心法

心法领悟 056：防止创建 Kernel 对象出错。

在使用 Kernel 类创建对象时，有时会出错，这是由于在使用 Kernel 类创建对象时，一定要保证数组 data 的长度，也就是 data.length 的值必须大于或等于 width*height，否则程序就会出错。

实例 057　图片马赛克特效

光盘位置：光盘\MR\057　　中级　　趣味指数：★★

实例说明

在进行马赛克处理后，可以使人物或图像变得模糊，对于不想公开的内容部分可以起到保护作用。运行程序，效果如图 3.19 所示，单击窗体上的“添加马赛克”按钮，将为图片添加马赛克，效果如图 3.20 所示。

图 3.19　原图片效果

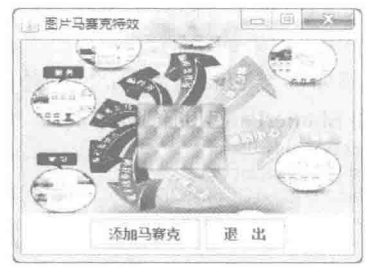


图 3.20　添加马赛克后的效果

关键技术

本实例主要是通过在缓冲图像对象上绘制渐变的矩形，并对缓冲图像对象进行模糊处理和透明度调整实现的。对图像进行模糊处理需要用到 Kernel 类和 ConvolveOp 类。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MosaicFrame 窗体类。

（3）在 MosaicFrame 窗体类中创建内部面板类 MosaicPanel，并重写 JComponent 类的 paint()方法，在该方法中实现在图像上绘制缓冲图像，完成马赛克效果的绘制。

（4）将内部面板类 MosaicPanel 的实例添加到窗体类 MosaicFrame 的内容面板上，用于在窗体上显示图像和在图像上绘制马赛克。面板类 MosaicPanel 的代码如下：

```
class MosaicPanel extends JPanel {
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        g2.drawImage(img, 0, 0, getWidth(),getHeight(), this);          //绘制图像对象
        g2.drawImage(image, 0, 0, this);                                //绘制缓冲图像对象
    }
}
```

（5）窗体类 MosaicFrame 上的“添加马赛克”按钮用于实现为图像添加马赛克的功能。“添加马赛克”按钮的事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        int x = 104;                                                    //矩形绘制点的 x 坐标
        int y = 60;                                                     //矩形绘制点的 y 坐标
        Rectangle2D.Float rect = null;
        image = new BufferedImage(getWidth() + 10, getHeight(),
                BufferedImage.TYPE_INT_ARGB);                           //创建缓冲图像对象
        Graphics2D gs2d = (Graphics2D) image.getGraphics();             //获得缓冲图像对象的 Graphics2D 对象
        AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.90f);    //获得表示透明度的 AlphaComposite 对象
        gs2d.setComposite(alpha);                                       //设置透明度
        GradientPaint paint = null;
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 3; j++) {
                paint = new GradientPaint(x, y, Color.white, x + 10,
                        y + 10, Color.gray,true);                       //创建循环渐变的 GradientPaint 对象
                gs2d.setPaint(paint);                                   //设置渐变
                rect = new Rectangle2D.Float(x, y, 20, 20);             //创建矩形对象
                gs2d.fill(rect);                                        //在缓冲图像对象上绘制矩形
                y = y + 20;                                             //计算下一个矩形的 y 坐标
            }
            y = 60;                                                     //还原 y 坐标
            x = x + 20;                                                 //计算 x 坐标
        }

        for (int i = 0; i < 3; i++) {                                   //该 for 循环实现 3 次模糊
            float[] elements = new float[9];                            //定义表示像素分量的数组
            for (int j = 0; j < 9; j++) {
                elements[j] = 0.11f;                                    //为数组赋值
            }
            convolve(elements);                                         //调用方法，实现模糊功能
        }
        mosaicPanel.repaint();                                          //调用 paint()方法
    }
});
```

（6）在“添加马赛克”按钮的事件中使用了 convolve()方法，用于实现模糊马赛克矩形的功能，该方法是窗体类 MosaicFrame 中的成员方法，其代码如下：

```
private void convolve(float[] elements) {
    Kernel kernel = new Kernel(3, 3, elements);                         //创建 Kernel 对象
    ConvolveOp op = new ConvolveOp(kernel);                             //创建 ConvolveOp 对象
    if (image == null) {
        return;
    }
    image = op.filter(image, null);                                     //过滤缓冲图像对象
}
```

秘笈心法

心法领悟 057：马赛克的实现原理。

马赛克的实现原理就是将模糊且有点透明的图形组合到一起，将已有图像的重要部分遮掩，从而无法辨清图像中的重要内容，可以起到保护隐私和机密等作用。

实例 058 模糊

光盘位置：光盘\MR\058　　中级　　趣味指数：★★

实例说明

本实例使用 Java 绘制技术实现了图片的模糊效果。运行程序，效果如图 3.21 所示，单击窗体上的“模糊”按钮，将对图片进行模糊处理，效果如图 3.22 所示。

图 3.21　图片模糊前的效果

图 3.22　图片模糊后的效果

关键技术

本实例主要是通过在缓冲图像对象上绘制图片，并对缓冲图像对象进行模糊处理实现的，对图像进行模糊处理需要用到 Kernel 类和 ConvolveOp 类。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 BlurImageFrame 窗体类。

（3）在 BlurImageFrame 窗体类中创建内部面板类 BlurImagePanel，在该类中实现创建缓冲图像对象，并将其绘制到绘图上下文对象上。

（4）将内部面板类 BlurImagePanel 的实例添加到窗体类 BlurImageFrame 的内容面板上，用于在窗体上显示原图片和进行模糊处理后的图片。面板类 BlurImagePanel 的代码如下：

```
class BlurImagePanel extends JPanel {
    public BlurImagePanel(){
        Image img = null;                                                          //声明创建图像对象
        try {
            img = ImageIO.read(new File("src/img/imag.jpg"));                      //创建图像对象
        } catch (IOException e) {
            e.printStackTrace();
        }
        image = new BufferedImage(img.getWidth(null),img.getHeight(null),BufferedImage.TYPE_INT_RGB);   //创建缓冲图像对象
        image.getGraphics().drawImage(img, 0, 0,  null);                           //在缓冲图像对象上绘制图像
    }
    public void paint(Graphics g) {
        if (image != null) {
            g.drawImage(image, 0, 0, null);                                        //绘制缓冲图像对象
        }
    }
}
```

（5）窗体类 BlurImageFrame 上的“模糊”按钮用于实现模糊图像的功能，其事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        float[] elements = new float[9];                    //定义表示像素分量的数组
        for (int i = 0; i < 9; i++) {
            elements[i] = 0.11f;                            //为数组赋值
        }
        convolve(elements);                                 //调用方法，实现模糊功能
    }
});
```

（6）在“模糊”按钮的事件中使用了 convolve()方法，用于实现模糊图像的功能，该方法是窗体类 BlurImageFrame 的成员方法，其代码如下：

```
private void convolve(float[] elements) {
    Kernel kernel = new Kernel(3, 3, elements);             //创建 Kernel 对象
    ConvolveOp op = new ConvolveOp(kernel);                 //创建 ConvolveOp 对象
    if (image == null) {
        return;
    }
    image = op.filter(image, null);                         //过滤缓冲图像对象
    repaint();                                              //调用 paint()方法
}
```

秘笈心法

心法领悟 058：模糊图像的部分区域。

为了掩盖图像的部分区域，可以对图像的该部分区域进行模糊处理，例如人物的肖像、重要或机密的物品等，都可以进行部分区域模糊。

实例 059 锐化

光盘位置：光盘\MR\059

中级

趣味指数：★★

实例说明

本实例使用 Java 绘制技术实现图片的锐化效果。运行程序，效果如图 3.23 所示，单击窗体上的“锐化”按钮，将对图片进行锐化处理，效果如图 3.24 所示。

图 3.23 图片锐化前的效果

图 3.24 图片锐化后的效果

关键技术

本实例主要是通过在缓冲图像对象上绘制图片，并对缓冲图像对象进行锐化处理实现的，对图像进行锐化处理需要用到 Kernel 类和 ConvolveOp 类。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SharpenImageFrame 窗体类。

（3）在 SharpenImageFrame 窗体类中创建内部面板类 SharpenImagePanel，在该类中实现创建缓冲图像对象，并将其绘制到绘图上下文对象上。

（4）将内部面板类 SharpenImagePanel 的实例添加到窗体类 SharpenImageFrame 的内容面板上，用于在窗体上显示原图片和进行锐化处理后的图片。面板类 SharpenImagePanel 的代码如下：

```
class SharpenImagePanel extends JPanel {
    public SharpenImagePanel(){
        Image img = null;                                                    //声明创建图像对象
        try {
            img = ImageIO.read(new File("src/img/imag.jpg"));                //创建图像对象
        } catch (IOException e) {
            e.printStackTrace();
        }
        image = new BufferedImage(img.getWidth(null),img.getHeight(null),BufferedImage.TYPE_INT_RGB);   //创建缓冲图像对象
        image.getGraphics().drawImage(img, 0, 0,  null);                     //在缓冲图像对象上绘制图像
    }
    public void paint(Graphics g) {
        if (image != null) {
            g.drawImage(image, 0, 0, null);                                  //绘制缓冲图像对象
        }
    }

}
```

（5）窗体类 SharpenImageFrame 上的“锐化”按钮用于实现锐化图像的功能，其事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        float[] elements = {0.0f,-1.0f,0.0f,-1.0f,5.0f,-1.0f,0.0f,-1.0f,0.0f};   //声明表示像素分量的数组
        convolve(elements);                                                      //调用方法实现图片锐化功能
    }
});
```

（6）在“锐化”按钮的事件中使用了 convolve()方法，用于实现锐化图像的功能，该方法是窗体类 SharpenImageFrame 的成员方法，其代码如下：

```
private void convolve(float[] elements) {
    Kernel kernel = new Kernel(3, 3, elements);                  //创建 Kernel 对象
    ConvolveOp op = new ConvolveOp(kernel);                      //创建 ConvolveOp 对象
    if (image == null) {
        return;
    }
    image = op.filter(image, null);                              //过滤缓冲图像对象
    repaint();                                                   //调用 paint()方法
}
```

秘笈心法

心法领悟 059：使图像的内容棱角分明。

为了使图像中的内容棱角分明，可以对图像进行锐化处理，进行锐化处理后，图像中的各元素区分明显，从而可以达到使图像中的内容棱角分明的效果。

实例 060　照亮边缘

光盘位置：光盘\MR\060　　中级　　趣味指数：★★

实例说明

本实例使用 Java 绘制技术实现图片的照亮边缘效果。运行程序，效果如图 3.25 所示，单击窗体上的“照亮边缘”按钮，将对图片进行照亮边缘处理，效果如图 3.26 所示。

图 3.25 图片照亮边缘前的效果

图 3.26 图片照亮边缘后的效果

关键技术

本实例主要是在缓冲图像对象上绘制图片，并对缓冲图像对象进行照亮边缘处理实现的，对图像进行照亮边缘处理需要用到 Kernel 类和 ConvolveOp 类。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 EdgeDetectImageFrame 窗体类。

（3）在 EdgeDetectImageFrame 窗体类中创建内部面板类 EdgeDetectImagePanel，在该类中实现创建缓冲图像对象，并将其绘制到绘图上下文对象上。

（4）将内部面板类 EdgeDetectImagePanel 的实例添加到窗体类 EdgeDetectImageFrame 的内容面板上，用于在窗体上显示原图片和进行照亮边缘处理后的图片。面板类 EdgeDetectImagePanel 的代码如下：

```
class EdgeDetectImagePanel extends JPanel {
    public EdgeDetectImagePanel(){
        Image img = null;                                                          //声明创建图像对象
        try {
            img = ImageIO.read(new File("src/img/image.jpg"));                     //创建图像对象
        } catch (IOException e) {
            e.printStackTrace();
        }
        image = new BufferedImage(img.getWidth(null),img.getHeight(null),BufferedImage.TYPE_INT_RGB);   //创建缓冲图像对象
        image.getGraphics().drawImage(img, 0, 0, null);                            //在缓冲图像对象上绘制图像
    }
    public void paint(Graphics g) {
        if (image != null) {
            g.drawImage(image, 0, 0, null);                                        //绘制缓冲图像对象
        }
    }
}
```

（5）窗体类 EdgeDetectImageFrame 上的“照亮边缘”按钮用于实现图片的照亮边缘处理，其事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        float[] elements = {0.0f,-1.0f,0.0f,-1.0f,4.0f,-1.0f,0.0f,-1.0f,0.0f};     //声明表示像素分量的数组
        convolve(elements);                                                        //调用方法实现图片锐化功能
    }
});
```

（6）在“照亮边缘”按钮的事件中使用 convolve()方法实现照亮图片边缘的功能，该方法是窗体类 SharpenImageFrame 的成员方法，其代码如下：

```
private void convolve(float[] elements) {
    Kernel kernel = new Kernel(3, 3, elements);                                    //创建 Kernel 对象
    ConvolveOp op = new ConvolveOp(kernel);                                        //创建 ConvolveOp 对象
    if (image == null) {
        return;
    }
```

```
        image = op.filter(image, null);                    //过滤缓冲图像对象
        repaint();                                         //调用 paint()方法
    }
```

秘笈心法

心法领悟 060：突出显示图像的边缘。

当需要将图像的边缘进行突出显示时，可以使用照亮边缘功能。对图像进行照亮边缘处理后，图像的边缘就会突出显示出来，例如需要得到图像的轮廓，就可以通过照亮图像边缘实现。

实例 061 反向

光盘位置：光盘\MR\061　　中级　　趣味指数：★★★

实例说明

本实例使用 Java 绘制技术实现图片的反向处理。运行程序，效果如图 3.27 所示，单击窗体上的“反向”按钮，将对图片进行反向处理，效果如图 3.28 所示。

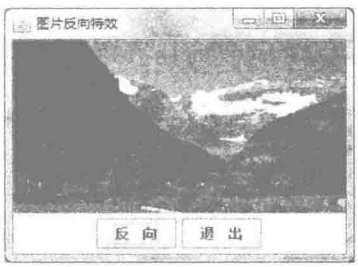


图 3.27　图片反向处理前的效果

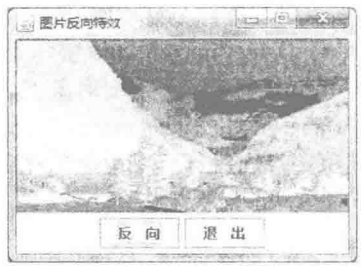


图 3.28　图片反向处理后的效果

关键技术

本实例主要是通过在缓冲图像对象上绘制图片，并对缓冲图像对象进行反向处理实现的，对图像进行反向处理需要用到 ShortLookupTable 类和 LookupOp 类。

（1）ShortLookupTable 类定义了一个查找表对象，该对象查找操作的输出被解释为一个无符号 short 量，查找表包含图像的一个或多个 band 的 short 型数据数组，并包含对数组建立索引前从输入值中减掉的偏移量，因此数组应小于为约束输入提供的本地数据的大小。如果查找表中只有一个数组，则将该数组应用于所有 band，该类的构造方法定义如下：

```
public ShortLookupTable(int offset, short[] data)
```

参数说明

❶ offset：在为数组建立索引前从输入值中减掉的值。

❷ data：short 型数据数组。

（2）LookupOp 类实现从源到目标的查找操作，其构造方法定义如下：

```
public LookupOp(LookupTable lookup, RenderingHints hints)
```

参数说明

❶ lookup：指定的 LookupTable 对象。

❷ hints：指定的 RenderingHints 对象，或者为 null。

（3）LookupOp 类提供了一个 filter()方法，可以对缓冲图像执行查找操作，该方法的定义如下：

```
public final BufferedImage filter(BufferedImage src, BufferedImage dst)
```

参数说明

❶ src：要过滤的源 BufferedImage。

❷ dst：存储过滤操作结果的 BufferedImage。

❸ 返回值：过滤后的 BufferedImage。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 NegativeImageFrame 窗体类。

（3）在 NegativeImageFrame 窗体类中创建内部面板类 NegativeImagePanel，在该类中实现创建缓冲图像对象，并将其绘制到绘图上下文对象上。

（4）将内部面板类 NegativeImagePanel 的实例添加到窗体类 NegativeImageFrame 的内容面板上，用于在窗体上显示原图片和进行反向处理后的图片，面板类 NegativeImagePanel 的代码如下：

```
class NegativeImagePanel extends JPanel {
    public NegativeImagePanel(){
        Image img = null;                                                          //声明创建图像对象
        try {
            img = ImageIO.read(new File("src/img/imag.jpg"));                      //创建图像对象
        } catch (IOException e) {
            e.printStackTrace();
        }
        image = new BufferedImage(img.getWidth(null),img.getHeight(null),BufferedImage.TYPE_INT_RGB);   //创建缓冲图像对象
        image.getGraphics().drawImage(img, 0, 0, null);                            //在缓冲图像对象上绘制图像
    }
    public void paint(Graphics g) {
        if (image != null) {
            g.drawImage(image, 0, 0, null);                                        //绘制缓冲图像对象
        }
    }
}
```

（5）窗体类 NegativeImageFrame 上的“反向”按钮用于实现图片的反向处理，其事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        short[] negative = new short[256*1];                    //创建表示颜色反向的分量数组
        for (int i = 0; i<256;i++){
            negative[i] = (short)(255-i);                       //为数组赋值
        }
        ShortLookupTable table = new ShortLookupTable(0,negative);   //创建查找表对象
        LookupOp op = new LookupOp(table,null);                 //创建实现从源到目标查找操作的 LookupOp 对象
        image = op.filter(image, null);                         //调用 LookupOp 对象的 filter()方法，实现图像反向功能
        repaint();                                              //调用 paint()方法
    }
});
```

秘笈心法

心法领悟 061：反向可以防止图像侵权。

当在网络上传输图像时，为了防止侵权，可以对图像反向处理后再进行传输，当图像到达目的地后，再对图像进行一次反向处理，即可还原图像。

实例 062 光栅图像

光盘位置：光盘\MR\062　　中级　　趣味指数：★★

实例说明

本实例演示如何利用 Java 的绘图技术实现光栅图像的绘制。运行程序，将在窗体上显示绘制的光栅图像，

效果如图 3.29 所示。

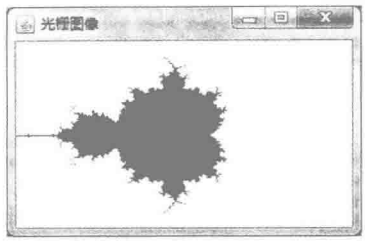


图 3.29　光栅图像

关键技术

本实例的实现主要是通过已有数字序列的数学公式，计算数字序列在指定的点(a,b)上是收敛的还是发散的，如果是收敛的，就绘制光栅上的点；如果是发散的，则不进行光栅的绘制。

公式 x * x-y * y + a 和 2 * x * y + b 分别用于计算点(a,b)在 X 轴和 Y 轴上的坐标值，如果 x 或者 y 小于等于 2，说明该点是收敛的；否则，说明该点是发散的。

注意：本实例的实现要求有一定的数学基础，这样才能真正理解该程序，否则可以跳过该实例。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 RasterImageFrame 窗体类。

（3）在 RasterImageFrame 窗体类中创建 isOrNotConvergence()方法，用于判断指定点是收敛的还是发散的，该方法的代码如下：

```
private boolean isOrNotConvergence(double a, double b) {  //判断数字序列上的点(a,b)是收敛的还是发散的
    double x = 0.0D;                                      //如果 x 大于 2，数字序列就是发散的
    double y = 0.0D;                                      //如果 y 大于 2，数字序列也是发散的
    int iterations = 0;                                   //循环变量
    while (x <= 2 && y <= 2 && iterations < MAX_ITERATIONS) {
        double xNew = x * x - y * y + a;                  //计算每个点的 x 值
        double yNew = 2 * x * y + b;                      //计算每个点的 y 值
        x = xNew;                                         //赋值给变量 x，用于判断是收敛还是发散
        y = yNew;                                         //赋值给变量 y，用于判断是收敛还是发散
        iterations++;                                     //调整迭代器变量的值
    }
    return x > 2 || y > 2;                                //返回 false 表示收敛，则进行绘制；为 true 表示发散，则透明
}
```

（4）在窗体类 RasterImageFrame 中，再创建 makeRasterImage()方法，用于获得绘制有光栅的缓冲图像，该方法的代码如下：

```
private BufferedImage makeRasterImage(int width, int height) {
    BufferedImage image = new BufferedImage(width, height,
            BufferedImage.TYPE_INT_ARGB);                 //创建缓冲图像对象
    WritableRaster raster = image.getRaster();            //获得提供像素写入功能的 WritableRaster 对象
    ColorModel model = image.getColorModel();             //获得缓冲图像的颜色模型
    Color fractalColor = Color.RED;                       //定义表示红色的颜色对象
    int argb = fractalColor.getRGB();                     //获得表示颜色的 RGB 值
    Object colorData = model.getDataElements(argb, null); //返回 ColorModel 中指定像素的数组表示形式
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            //计算点(i,j)是否导致与点(a,b)上的像素收敛
            double a = XMIN + i * (XMAX - XMIN) / width;
            double b = YMIN + j * (YMAX - YMIN) / height;
```

```
                if (!isOrNotConvergence(a, b)) {   //如果点(i,j)导致与点(a,b)上的像素收敛，escapesToInfinity()方法返回 false，则进行光栅绘制
                    raster.setDataElements(i, j, colorData);     //为类型 TransferType 基本数组中的单个像素设置数据
                }
            }
        }
        return image;                                          //返回绘制有光栅图像的缓冲图像对象
    }
```

秘笈心法

心法领悟 062：将光栅图像保存为图片。

光栅图像生成之后，如果希望将其保存为图片，可以使用 IOImage 类的 write()方法，这样当需要使用光栅图像时，可以直接引用保存的光栅图片。

实例 063 图片倒影效果

源码位置：光盘\MR\063　　高级　　趣味指数：★★★

实例说明

本实例演示如何使用 Java 绘图技术实现图片倒影效果。运行程序，效果如图 3.30 所示。

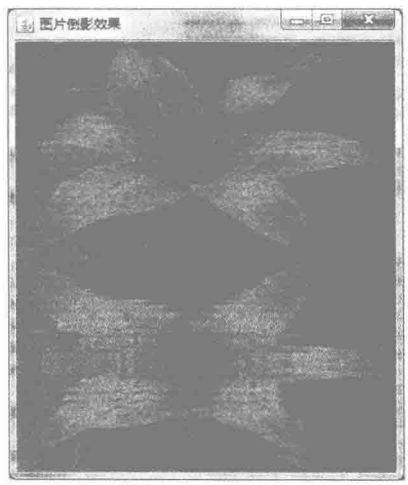


图 3.30　图片倒影效果

关键技术

本实例主要是通过在 JPanel 类的子类中重写 JComponent 类的 paint()方法，并在该方法中使用 Graphics 类的 copyArea()方法复制图像区域，实现最终的图片水波倒影效果。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PictureInvertedFrame 窗体类。

（3）在 PictureInvertedFrame 窗体类中创建内部面板类 PictureInvertedPanel，并重写 JComponent 类的 paint()方法，在该方法中使用 Graphics 类的 copyArea()方法复制图像区域，实现图片水波倒影效果。

（4）将内部面板类 PictureInvertedPanel 的实例添加到窗体类 PictureInvertedFrame 的内容面板上，用于在窗

体上显示图片的水波倒影效果，面板类 PictureInvertedPanel 的代码如下：

```
class PictureInvertedPanel extends JPanel {
    private Graphics graphics;                                          //Graphics 对象
    private Graphics waveGraphics;                                      //绘制水波的 Graphics 对象
    private Image oldImage;                                             //原图像对象
    private Image waveImage;                                            //声明表示水波效果的图像对象
    private int currentImage, imageWidth, imageHeight;
    private boolean imageLoaded;                                        //表示图片是否被加载的标记
    public void paint(Graphics g) {
        drawWaterWave();
        if (waveImage != null) {
            g.drawImage(waveImage, -currentImage * imageWidth, imageHeight, this);   //绘制图像
        }
        g.drawImage(oldImage, 0, 1, this);                              //绘制原图片
        g.clearRect(imageWidth, 0, imageWidth * 4, imageHeight*2);      //清除显示区域右侧的内容
    }
    public void drawWaterWave() {
        currentImage = 0;
        if (!imageLoaded) {                                             //如果未加载图片
            graphics = getGraphics();                                   //获得绘图上下文对象
            MediaTracker mediatracker = new MediaTracker(this);         //创建媒体跟踪对象
            URL imgUrl = PictureInvertedFrame.class
                    .getResource("/img/image.jpg");                     //获取图片资源的路径
            oldImage = Toolkit.getDefaultToolkit().getImage(imgUrl);    //获取图像资源
            mediatracker.addImage(oldImage, 0);                         //添加图片
            try {
                mediatracker.waitForAll();                              //加载图片
                imageLoaded = !mediatracker.isErrorAny();               //是否有错误发生
            } catch (InterruptedException ex) {
            }
            if (!imageLoaded) {                                         //图片加载失败
                graphics.drawString("图片加载错误", 10, 40);             //绘制错误信息
                return;
            }
            imageWidth = oldImage.getWidth(this);                       //得到图像宽度
            imageHeight = oldImage.getHeight(this);                     //得到图像高度
            createWave();                                               //创建水波效果
        }
    }
    public void createWave() {
        Image img = createImage(imageWidth, imageHeight);               //以图像宽度和高度创建图像对象
        Graphics g = null;
        if (img != null) {
            g = img.getGraphics();                                      //得到 Image 对象的 Graphics 对象
            g.drawImage(oldImage, 0, 0, this);                          //绘制原图像对象
            for (int i = 0; i < imageHeight; i++) {
                g.copyArea(0, imageHeight - 1 - i, imageWidth, 1, 0,
                        -imageHeight + 1 + (i * 2));                    //复制图像区域
            }
        }
        waveImage = createImage(13 * imageWidth, imageHeight);          //得到水波效果的图像对象
        if (waveImage != null) {
            waveGraphics = waveImage.getGraphics();                     //得到水波效果的绘图上下文对象
            waveGraphics.drawImage(img, 12 * imageWidth, 0, this);      //绘制图像
            int j = 0;
            while (j < 12) {
                simulateWaves(waveGraphics, j);                         //调用方法，模拟水波效果
                j++;
            }
        }
    }
    public void simulateWaves(Graphics g, int i) {                      //水波效果模拟
```

```
                int j = (12 - i) * imageWidth;                                    //计算复制像素的水平距离
                int waveHeight = imageHeight / 16;                                //计算水波高度
                for (int l = 0; l < imageHeight; l++) {
                    int k = (int) ((waveHeight * (l + 28) * Math.sin(waveHeight
                            * (imageHeight - l) / (l + 1))) / imageHeight);       //计算复制像素的垂直距离
                    if (l < -k){
                        g.copyArea(12 * imageWidth, l, imageWidth, 1, -j, 0);     //复制图像区域，形成水波
                    } else {
                        g.copyArea(12 * imageWidth, l + k, imageWidth, 1, -j, -k); //复制图像区域，形成水波
                    }
                }
            }
        }
```

秘笈心法

心法领悟 063：将图片的水波倒影保存为图片。

本实例实现了图片水波倒影特效，但只是在应用程序中显示了该特效，如果想把图片的水波倒影保存为图片，可以使用 Robot 类截取图片的水波倒影部分，然后使用 IOImage 类的 write()方法进行保存。

第 4 章

动画和游戏

- 文字动画
- 图片动画
- 游戏开发

4.1　文 字 动 画

实例 064　文字淡入淡出

光盘位置：光盘\MR\064　　初级　　趣味指数：★★★★

实例说明

本实例演示如何利用 Java 的绘图技术实现文字淡入淡出特效的绘制。运行程序，将在窗体上以淡入淡出的效果显示文字，如图 4.1 所示。

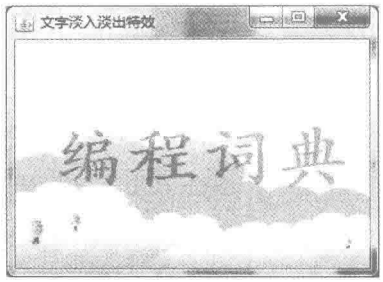


图 4.1　文字淡入淡出特效

关键技术

本实例主要是通过 Graphics2D 类的 setComposite()方法为绘图上下文指定表示透明度的 AlphaComposite 对象，以及使用从 Graphics 类继承的 drawString()方法绘制文本实现的。

（1）使用 AlphaComposite 类获得表示透明度的 AlphaComposite 对象，该对象使用 AlphaComposite 类的字段 SrcOver 调用 derive()方法获得，该方法的定义如下：

```
public AlphaComposite derive(float alpha)
```

参数说明

❶ alpha：闭区间 0.0f~1.0f 之间的一个浮点数字，为 0.0f 时完全透明，为 1.0f 时不透明。

❷ 返回值：表示透明度的 AlphaComposite 对象。

（2）使用 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的 AlphaComposite 对象，该方法的定义如下：

```
public abstract void setComposite(Composite comp)
```

参数说明

comp：表示透明度的 AlphaComposite 对象。

（3）使用 Graphics 类的 drawString()方法绘制文本，该方法的定义如下：

```
public abstract void drawString(String str, int x, int y)
```

参数说明

❶ str：绘制的文本内容。

❷ x：绘制点的 x 坐标。

❸ y：绘制点的 y 坐标。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 TextFadeFrame 窗体类。

（3）在 TextFadeFrame 窗体类中创建内部面板类 TextFadePanel，该面板类实现了 Runnable 接口，并重写 JComponent 类的 paint()方法和实现 Runnable 接口的 run()方法，在 paint()方法中使用 Graphics 类的 setFont()、setColor()、setComposite()和 drawString()方法，完成设置字体、颜色、透明度和绘制文本的操作；在 run()方法中实现改变透明度值的操作。

（4）将内部面板类 TextFadePanel 的实例添加到窗体类 TextFadeFrame 的内容面板上，用于在窗体上显示淡入淡出效果的文字。内部面板类 TextFadePanel 的代码如下：

```
class TextFadePanel extends JPanel implements Runnable {
    boolean flag = true;                                          //定义标记变量，用于控制 x 的值
    float x = 0.0f;                                               //定义表示透明度的变量 x
    AlphaComposite alpha = AlphaComposite.SrcOver.derive(x);      //获得表示透明度的 AlphaComposite 对象
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;                           //获得 Graphics2D 对象
        g2.drawImage(img, 0, 0,   getWidth(), getHeight(), this); //绘制图像
        Font font = new Font("华文楷体", Font.BOLD, 60);          //创建字体对象
        g2.setFont(font);                                         //指定字体
        g2.setColor(Color.RED);                                   //指定颜色
        g2.setComposite(alpha);                                   //指定 AlphaComposite 对象
        g2.drawString("编程词典", 30, 120);                       //绘制文本
    }
    public void run() {
        while (true) {
            if (flag) {                                           //flag 为 true 时
                x-=0.1f;                                          //x 进行减 0.1 计算
                if (x <= 0.0f) {                                  //x 小于等于 0.0f 时
                    x = 0.0f;                                     //x 等于 0.0f
                    flag = false;                                 //为 flag 赋值为 false
                }
            } else {                                              //flag 为 false 时
                x+=0.1f;                                          //x 进行加 0.1 计算
                if (x >= 1.0f) {                                  //x 大于等于 1.0f 时
                    x = 1.0f;                                     //x 等于 1.0f
                    flag = true;                                  //为 flag 赋值为 true
                }
            }
            alpha = AlphaComposite.SrcOver.derive(x);             //重新获得表示透明度的 AlphaComposite 对象
            repaint();                                            //调用 paint()方法
            try {
                Thread.sleep(150);                                //休眠 150 毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

注意：在使用本实例时，需将项目需要的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 064：解决浮点数计算的不精确性。

为 AlphaComposite 类设置透明度值时，需要使用 0.0f~1.0f 之间的浮点数，但是在进行浮点数加减运算时，结果往往是不精确的，为此当使用 if 语句判断 x 的最小值时，使用 x==0.0f 是不准确的，应该写成 x<=0.0f；同样，使用 if 语句判断 x 的最大值时，使用 x==1.0f 也是不准确的，而应该写成 x>=1.0f，从而解决浮点数计算的不精确性。

实例 065　文字缩放

光盘位置：光盘\MR\065　　初级　　趣味指数：★★★

实例说明

本实例演示如何利用 Java 的绘图技术实现文字缩放特效的绘制。运行程序，将在窗体上显示缩放的文字，效果如图 4.2 和图 4.3 所示。

图 4.2　文字缩小后的效果

图 4.3　文字放大后的效果

关键技术

本实例主要是使用从 Graphics 类继承的 setFont()方法改变字体的大小，以及使用 drawString()方法绘制文本实现的。

（1）使用 Font 类可以创建字体对象，其中包括字体名称、字型和字体大小，然后使用 Graphics 类的 setFont()方法为绘图上下文设置字体，Font 类的构造方法的定义如下：

```
public Font(String name, int style, int size)
```

参数说明

❶ name：字体的名称。

❷ style：字体的字型，也就是字体的样式。

❸ size：字体的大小。

（2）使用 Graphics 类的 drawString()方法绘制文本。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 TextZoomFrame 窗体类。

（3）在 TextZoomFrame 窗体类中创建内部面板类 TextZoomPanel，该面板类实现了 Runnable 接口，并重写 JComponent 类的 paint()方法和实现 Runnable 接口的 run()方法。在 paint()方法中使用 Graphics 类的 setFont()、setColor()和 drawString()方法，完成设置字体、字型、字体大小、颜色和绘制文本的操作；在 run()方法中实现改变字体大小的操作。

（4）将内部面板类 TextZoomPanel 的实例添加到窗体类 TextZoomFrame 的内容面板上，用于在窗体上显示缩放效果的文字。内部面板类 TextZoomPanel 的代码如下：

```
class TextZoomPanel extends JPanel implements Runnable {
    boolean flag = false;                                    //定义标记变量，用于控制 x 的值
    int x = 12;                                              //定义表示字体大小的变量 x
    Font font = new Font("华文楷体", Font.BOLD, x);          //创建字体对象
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;                      //获得 Graphics2D 对象
        g2.drawImage(img, 0, 0, getWidth(), getHeight(), this);  //绘制图像
```

```
            g2.setFont(font);                                    //指定字体
            g2.setColor(Color.WHITE);                            //指定颜色
            g2.drawString("编程词典", 30, 120);                   //绘制文本
        }
        public void run() {
            while (true) {
                if (flag) {                                      //flag 为 true 时
                    x-=1;                                        //x 进行减 1 计算
                    if (x <= 12) {                               //x 小于等于 12 时
                        x = 12;                                  //x 等于 12
                        flag = false;                            //为 flag 赋值为 false
                    }
                } else {                                         //flag 为 false 时
                    x+=1;                                        //x 进行加 1 计算
                    if (x >= 72) {                               //x 大于等于 72 时
                        x = 72;                                  //x 等于 72
                        flag = true;                             //为 flag 赋值为 true
                    }
                }
                font = new Font("华文楷体", Font.BOLD, x);        //重新创建字体对象
                repaint();                                       //调用 paint()方法
                try {
                    Thread.sleep(50);                            //休眠 50 毫秒
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
```

注意：在使用本实例时，需将项目需要的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 065：文字缩放的作用。

在现实生活中，为了突出显示某些文字（如广告语等），可以将这些需要突出显示的文字制作成文字缩放动画，以引起观赏者的注意。

实例 066　文字跑马灯

光盘位置：光盘\MR\066　　中级　　趣味指数：★★★

实例说明

本实例演示如何利用 Java 的绘图技术和多线程技术实现文字跑马灯特效。运行程序，将在窗体上以跑马灯效果显示文字信息，效果如图 4.4 所示。

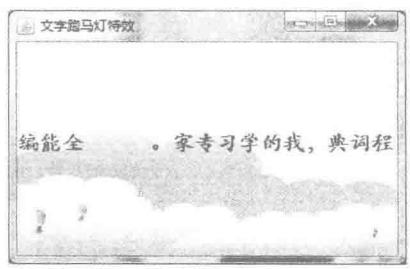


图 4.4　文字跑马灯特效

关键技术

本实例将作为跑马灯的文字转换为字符数组，然后通过另一个 int 型数组来存储字符数组中每个字符的 x 坐标值，并在线程的 run()方法中有规律地调整数组中这些 x 坐标值，从而实现文字跑马灯的特效。

（1）使用 String 类的 toCharArray()方法，可以将字符串转换为字符数组。toCharArray()方法的定义如下：

```
public char[] toCharArray()
```

参数说明

返回值：一个新分配的字符数组，其长度是此字符串的长度，其内容被初始化为包含此字符串表示的字符序列。

（2）使用 int 型数组来存储跑马灯文字数组中每个字符的 x 位置坐标，当第一次执行线程时，使 x 坐标值等比递增，以后再执行线程时，则使 x 坐标值等差递增，从而实现了文字跑马灯效果。调整 x 坐标值的代码如下：

```
if (!flag){
    x[i]=x[i]+20*i;          //x 坐标进行等比递增
} else {
    x[i]=x[i]+20;            //x 坐标进行等差递增
}
```

说明：上面代码中的 flag 是一个标记变量，为 false 时，表示第一次执行，x 坐标进行等比递增；否则，进行等差递增。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 HorseRaceLightTextFrame 窗体类。

（3）在 HorseRaceLightTextFrame 窗体类中创建内部面板类 HorseRaceLightTextPanel，该面板类实现了 Runnable 接口，并重写 JComponent 类的 paint()方法和实现 Runnable 接口的 run()方法。在 paint()方法中完成跑马灯文字的绘制；在 run()方法中实现改变跑马灯文字的 x 坐标值。

（4）将内部面板类 HorseRaceLightTextPanel 的实例添加到窗体类 HorseRaceLightTextFrame 的内容面板上，用于在窗体上显示跑马灯效果的文字。内部面板类 HorseRaceLightTextPanel 的代码如下：

```
class HorseRaceLightTextPanel extends JPanel implements Runnable {    //创建内部面板类
    String value = "全能编程词典，我的学习专家。";                      //存储绘制的内容
    char[] drawChar = value.toCharArray();                             //将绘制内容转换为字符数组
    int[] x = new int[drawChar.length];                                //存储每个字符绘制点 x 坐标的数组
    int y = 100;                                                       //存储绘制点的 y 坐标
    public void paint(Graphics g) {
        g.clearRect(0, 0, getWidth(), getHeight());                    //清除绘图上下文的内容
        g.drawImage(img, 0, 0, getWidth(), getHeight(), this);         //绘制图像
        Font font = new Font("华文楷体", Font.BOLD, 20);               //创建字体对象
        g.setFont(font);                                               //指定字体
        g.setColor(Color.RED);                                         //指定颜色
        for (int j = drawChar.length-1; j >=0; j--){
            g.drawString(drawChar[drawChar.length-1-j]+"",x[j] , y);   //绘制文本
        }
    }
    public void run() {
        try {
            boolean flag = false;                                      //为 false 时表示第一次执行，x 坐标进行等比递增，否则进行等差递增
            while (true) {                                             //读取内容
                Thread.sleep(300);                                     //当前线程休眠 300 毫秒
                for (int i = drawChar.length-1; i >=0 ; i--) {
                    if (!flag){
                        x[i]=x[i]+20*i;                                //x 坐标进行等比递增
                    } else {
                        x[i]=x[i]+20;                                  //x 坐标进行等差递增
```

```
                    }
                    if (x[i]>=360 - 20){                    //大于窗体宽度减 20 的值时
                        x[i] = 0;                           //x 坐标为 0
                    }
                }
                repaint();                                  //调用 paint()方法
                if (!flag) {
                    flag = true;                            //赋值为 true
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

注意：在使用本实例时，需将项目需要的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 066：在屏幕上显示公司的联系电话和地址。

为了使公司的联系电话和地址能够被更多的人所熟悉，可以将这些信息通过屏幕以跑马灯的形式进行循环显示，从而可以引起人们的注意，起到宣传的作用。

实例 067　字幕显示

光盘位置：光盘\MR\067　　中级　　趣味指数：★★★

实例说明

本实例演示如何利用 Java 的绘图技术实现字幕显示特效的绘制。运行程序，文字会从窗体的底部向上移动，移动一小段距离后消失，然后其他文字又从窗体底部向上移动，效果如图 4.5 和图 4.6 所示。

图 4.5　字幕显示特效 1

图 4.6　字幕显示特效 2

关键技术

本实例通过 Java 绘图技术绘制文本，通过多线程技术改变文本绘制点的 y 坐标，从而实现字幕显示的特效。

（1）使用 Graphics 类的 drawString()方法完成文本的绘制。

（2）通过实现 Runnable 接口实现多线程，并在 run()方法中改变文本绘制点的 y 坐标值，run()方法中调整 y 坐标值的代码如下：

```
if (y <= 216 - 50) {                                       //如果已经向上移动 50 像素
    y = 216;                                               //y 坐标定位到最下方
    if (value.equals("明日编程词典网址")) {
        value = "http://www.mrbccd.com";                   //改变绘制的内容
```

```
        } else {
            value = "明日编程词典网址";                    //改变绘制的内容
        }
    } else {                                              //如果还没向上移动到 50 像素
        y -= 2;                                           //y 坐标上移
    }
```

说明：上面代码中的 216 是第一个绘制点的 y 坐标值，每执行一次线程，则使 y 坐标值减 2。当上移 50 像素后，再从 y 坐标值是 216 的位置移动其他字幕。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 CaptionSpecificFrame 窗体类。

（3）在 CaptionSpecificFrame 窗体类中创建内部面板类 CaptionSpecificPanel，该面板类实现了 Runnable 接口，并重写 JComponent 类的 paint()方法和实现 Runnable 接口的 run()方法。在 paint()方法中完成字幕文字的绘制；在 run()方法中实现改变字幕文字的 y 坐标值。

（4）将内部面板类 CaptionSpecificPanel 的实例添加到窗体类 CaptionSpecificFrame 的内容面板上，用于在窗体上显示字幕效果的文字。内部面板类 CaptionSpecificPanel 的代码如下：

```
class CaptionSpecificPanel extends JPanel implements Runnable {
    int x = 50;                                           //存储绘制点的 x 坐标
    int y = 216;                                          //存储绘制点的 y 坐标
    String value = "明日编程词典网址";                     //存储绘制的内容
    public void paint(Graphics g) {
        g.clearRect(0, 0, 316, 237);                      //清除绘图上下文的内容
        g.drawImage(img, 0, 0, getWidth(), getHeight(), this);  //绘制图像
        Font font = new Font("华文楷体", Font.BOLD, 20);   //创建字体对象
        g.setFont(font);                                  //指定字体
        g.setColor(Color.RED);                            //指定颜色
        g.drawString(value, x, y);                        //绘制文本
    }
    public void run() {
        try {
            while (true) {                                //读取内容
                Thread.sleep(100);                        //当前线程休眠 1 秒
                if (y <= 216 - 50) {                      //如果已经向上移动 50 像素
                    y = 216;                              //y 坐标定位到最下方
                    if (value.equals("明日编程词典网址")) {
                        value = "http://www.mrbccd.com";  //改变绘制的内容
                    } else {
                        value = "明日编程词典网址";        //改变绘制的内容
                    }
                } else {                                  //如果还没向上移动到 50 像素
                    y -= 2;                               //y 坐标上移
                }
                repaint();                                //调用 paint()方法
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

注意：在使用本实例时，需要将项目所需的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 067：制作从左向右移动的字幕特效。

指定字幕文字的 x 坐标起始值，通过线程改变其 x 坐标值，当文字在 x 坐标轴上移动一定距离后，将 x 坐标值设置为起始值，并重新切换其他文字，从而实现从左向右移动的字幕特效。

实例 068 文字闪现

光盘位置：光盘\MR\068　　中级　　趣味指数：★★★

实例说明

本实例使用 Java 的绘图技术和多线程技术实现文字闪现的特效。运行程序，窗体上会出现闪现的文字，即窗体上的文字在很短的时间内交替消失和出现，如图 4.7 和图 4.8 所示。

图 4.7　无闪现文字的效果

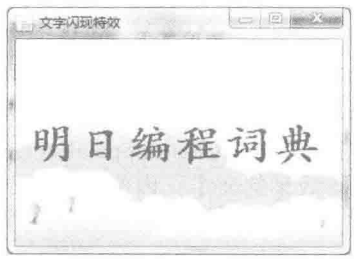


图 4.8　有闪现文字的效果

关键技术

本实例通过 Java 绘图技术绘制文本，通过多线程技术控制文字是否出现，从而实现了文字闪现的特效。

（1）使用 Graphics 类的 drawString()方法完成文本的绘制。

（2）通过实现 Runnable 接口实现多线程，并在 run()方法中通过定义标记变量来控制文字的闪现，当需要显示闪现文字时，将文字赋值给变量；当不需要显示文字时，为变量赋值为空字符串。run()方法中，通过标记变量控制闪现文字的代码如下：

```
if (flag) {                          //flag 为 true
    flag = false;                    //赋值为 false
    value="明日编程词典";             //为 value 赋值
} else {
    flag = true;                     //赋值为 true
    value="";                        //赋值为空字符串
}
```

说明：上面代码中的 flag 是一个标记变量，为 true 时，为变量 value 赋值需要显示的文字；为 false 时，为变量 value 赋值为空字符串。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 TextFlashFrame 窗体类。

（3）在 TextFlashFrame 窗体类中创建内部面板类 TextFlashPanel，该面板类实现了 Runnable 接口，并重写 JComponent 类的 paint()方法和实现 Runnable 接口的 run()方法。在 paint()方法中完成闪现文字的绘制；在 run()方法中控制是否显示闪现文字。

（4）将内部面板类 TextFlashPanel 的实例添加到窗体类 TextFlashFrame 的内容面板上，用于在窗体上显示文字闪现特效。内部面板类 TextFlashPanel 的代码如下：

```
class TextFlashPanel extends JPanel implements Runnable {
```

```
    boolean flag = true;                                    //标记变量
    String value = "";                                      //存放绘制内容的变量
    public void paint(Graphics g) {
        g.clearRect(0, 0, 310, 230);                        //清除绘图上下文的内容
        g.drawImage(img,0,0,getWidth(),getHeight(),this);   //绘制图像
        Font font = new Font("华文楷体", Font.BOLD, 42);     //创建字体对象
        g.setFont(font);                                    //指定字体
        g.setColor(Color.RED);                              //指定颜色
        g.drawString(value, 10, 110);                       //绘制文本
    }
    public void run() {
        try {
            while (true) {                                  //读取内容
                Thread.sleep(150);                          //当前线程休眠 1 秒
                if (flag) {                                 //flag 为 true
                    flag = false;                           //赋值为 false
                    value="明日编程词典";                    //为 value 赋值
                } else {
                    flag = true;                            //赋值为 true
                    value="";                               //赋值为空字符串
                }
                repaint();                                  //调用 paint()方法
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

注意：在使用本实例时，需将项目需要的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 068：实现不同文字的交替显示。

本实例当标记变量 flag 为 false 时，为变量 value 赋值一个新的字符串，就可以实现不同文字的交替显示，并可以通过调整线程的休眠时间对文字交替显示的速度进行控制。

实例 069　滚动广告字幕

光盘位置：光盘\MR\069　　中级　　趣味指数：★★★

实例说明

滚动广告字幕是指文字从窗体的一侧向另一侧移动，当所有文字都从一侧移动到窗体以外区域时，再次重复上述移动过程。运行程序，将在窗体上显示滚动广告字幕特效，效果如图 4.9 所示。

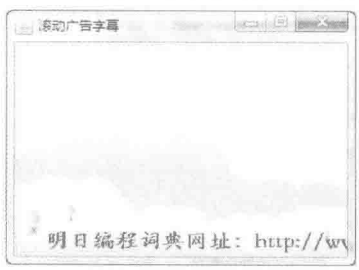


图 4.9　滚动广告字幕

关键技术

本实例通过 Java 绘图技术绘制文本，并通过多线程技术控制广告字幕的滚动显示，从而实现滚动广告字幕特效。

（1）使用 Graphics 类的 drawString()方法，完成文本的绘制。

（2）通过实现 Runnable 接口实现多线程，并在 run()方法中改变文本绘制点的 x 坐标值。run()方法中调整 x 坐标值的代码如下：

```
if (x <= -400) {                                    //该条件可以根据需要自行调整
    x = 316;                                        //x 坐标定位到最右侧
} else {
    x -= 2;                                         //x 坐标左移
}
```

说明：上面代码中的 if 语句的条件 x<=-400 是采用试探的方式确定的，主要是看 x 的值为多少时，能将所有广告字幕文字移出窗体。如果全部移出窗体，就让 x 的值为 316，即重新从窗体右侧向左侧移动广告字幕；如果 x<=-400 不成立，则使 x 的值减 2，即向左移动广告字幕。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 RollAdvertisementFrame 窗体类。

（3）在 RollAdvertisementFrame 窗体类中创建内部面板类 RollAdvertisementPanel，该面板类实现了 Runnable 接口，并重写了 JComponent 类的 paint()方法，同时也实现了 Runnable 接口的 run()方法。在 paint()方法中完成广告字幕的绘制；在 run()方法中控制广告字幕的滚动。

（4）将内部面板类 RollAdvertisementPanel 的实例添加到窗体类 RollAdvertisementFrame 的内容面板上，用于在窗体上显示滚动的广告字幕。内部面板类 RollAdvertisementPanel 的代码如下：

```
class RollAdvertisementPanel extends JPanel implements Runnable {
    int x = 316;                                                    //存储绘制点的 x 坐标
    int y = 190;                                                    //存储绘制点的 y 坐标
    String value = "明日编程词典网址：http://www.mrbccd.com";        //存储绘制的内容
    public void paint(Graphics g) {
        g.clearRect(0, 0, 316, 237);                                //清除绘图上下文的内容
        g.drawImage(img, 0, 0, getWidth(), getHeight(), this);      //绘制图像
        Font font = new Font("华文楷体", Font.BOLD, 20);            //创建字体对象
        g.setFont(font);                                            //指定字体
        g.setColor(Color.RED);                                      //指定颜色
        g.drawString(value, x, y);                                  //绘制文本
    }
    public void run() {
        try {
            while (true) {                                          //读取内容
                Thread.sleep(50);                                   //当前线程休眠 1 秒
                if (x <= -400) {                                    //该条件可以根据需要自行调整
                    x = 316;                                        //x 坐标定位到最右侧
                } else {
                    x -= 2;                                         //x 坐标左移
                }
                repaint();                                          //调用 paint()方法
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

注意：在使用本实例时，需将项目需要的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 069：制作从左向右移动的滚动广告字幕需要注意英文拼写。

在制作从左向右移动的滚动广告字幕时，通常是将广告字幕逆序书写，如将字符串“全能编程词典”逆序书写就是“典词程编能全”，由于逆序书写对于英文不是很实用，如单词 word 逆序为 drow，所以在设计从左向右移动的滚动广告字幕时，要考虑并注意这一点。

4.2　图 片 动 画

实例 070　图片淡入淡出　　中级

光盘位置：光盘\MR\070　　趣味指数：★★★★★

实例说明

本实例演示如何利用 Java 的绘图技术实现图片淡入淡出特效的绘制。运行程序，将在窗体上以淡入淡出的效果显示图片，效果如图 4.10 和图 4.11 所示。

图 4.10　只有背景图片的效果

图 4.11　图片淡入淡出的效果

关键技术

本实例主要是通过 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的 AlphaComposite 对象，以及使用从 Graphics 类继承的 drawImage()方法绘制图像实现的。

（1）使用 Composite 接口的实现类 AlphaComposite 获得表示透明度的 AlphaComposite 对象，该对象可以使用 AlphaComposite 类的字段 SrcOver 调用 derive()方法获得。

（2）使用 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的 AlphaComposite 对象。

（3）使用 Graphics 类的 drawImage()方法绘制图片，该方法的定义如下：

```
public abstract boolean drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)
```

参数说明

❶ img：要绘制的图像。

❷ x：起始点的 x 坐标。

❸ y：起始点的 y 坐标。

❹ width：绘制的宽度。

❺ height：绘制的高度。

❻ observer：图像观察者。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PictureFadeFrame 窗体类。

（3）在 PictureFadeFrame 窗体类中创建内部面板类 PictureFadePanel，该面板类实现了 Runnable 接口，并重写了 JComponent 类的 paint()方法，同时也实现了 Runnable 接口的 run()方法，并在 paint()方法中使用 Graphics 类的 drawImage()方法绘制图片，然后在 run()方法中实现改变透明度值的操作。

（4）将内部面板类 PictureFadePanel 的实例添加到窗体类 PictureFadeFrame 的内容面板上，用于在窗体上显示淡入淡出效果的图片。内部面板类 PictureFadePanel 的代码如下：

```
class PictureFadePanel extends JPanel implements Runnable {
    boolean flag = true;                                          //定义标记变量，用于控制 x 的值
    float x = 0.0f;                                               //定义表示透明度的变量 x
    AlphaComposite alpha = AlphaComposite.SrcOver.derive(x);      //获得表示透明度的 AlphaComposite 对象
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;                           //获得 Graphics2D 对象
        g2.clearRect( 0, 0, getWidth(), getHeight());             //绘制图像
        g2.drawImage(img1, 0, 0, getWidth(), getHeight(), this);  //绘制图像
        g2.setComposite(alpha);                                   //指定 AlphaComposite 对象
        g.drawImage(img2, 50, 40, getWidth() - 100, getHeight() - 80, this);  //绘制调整透明度后的图片，实现图片淡入淡出特效
    }
    public void run() {
        while (true) {
            if (flag) {                                           //flag 为 true 时
                x -= 0.1f;                                        //x 进行减 0.1 计算
                if (x <= 0.0f) {                                  //x 小于等于 0.0f 时
                    x = 0.0f;                                     //x 等于 0.0f
                    flag = false;                                 //为 flag 赋值为 false
                }
            } else {                                              //flag 为 false 时
                x += 0.1f;                                        //x 进行加 0.1 计算
                if (x >= 1.0f) {                                  //x 大于等于 1.0f 时
                    x = 1.0f;                                     //x 等于 1.0f
                    flag = true;                                  //为 flag 赋值为 true
                }
            }
            alpha = AlphaComposite.SrcOver.derive(x);             //重新获得表示透明度的 AlphaComposite 对象
            repaint();                                            //调用 paint()方法
            try {
                Thread.sleep(200);                                //休眠 200 毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

注意：在使用本实例时，需将项目需要的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 070：同时显示淡入淡出的文字和图片。

在进行图片的淡入淡出处理时，可以在设置绘图上下文的透明度后，在画布绘图上下文上绘制文字和图片，这样就可以达到同时显示淡入淡出的文字和图片的效果。

实例 071　随鼠标指针移动的图片

光盘位置：光盘\MR\071　　中级　　趣味指数：★★★★

实例说明

本实例通过鼠标的移动事件，演示了如何使显示有图片的标签随着鼠标指针移动，另外，还在显示图片的标签下方显示一个文本颜色随机变化的标签，这是通过多线程序技术实现的。运行程序，显示有图片和文本的标签会随着鼠标指针移动，效果如图 4.12 所示。

图 4.12　随鼠标指针移动的图片

关键技术

在鼠标移动的过程中，通过 MouseEvent 类的 getX()方法获得鼠标指针在背景面板中的横坐标值，通过 MouseEvent 类的 getY()方法获得鼠标指针在背景面板中的纵坐标值，然后将要移动图形的标签通过 setLocation()方法设置到对鼠标指针当前位置坐标进行计算后的位置处，使图形在背景面板上移动。可以在背景面板的鼠标移动事件中通过如下代码来实现：

```
addMouseMotionListener(new MouseMotionAdapter() {
    public void mouseMoved(final MouseEvent e) {
        int x = e.getX();                          //获得鼠标指针在窗体容器中的横坐标值
        int y = e.getY();                          //获得鼠标指针在窗体容器中的纵坐标值
        int w = lb_move.getWidth();                //获得随鼠标指针移动的图形所在标签的宽度
        int h = lb_move.getHeight();               //获得随鼠标指针移动的图形所在标签的高度
        int x1 = x - w / 2;                        //计算出随鼠标指针移动的图形所在标签的横坐标值
        int y1 = y - h / 2;                        //计算出随鼠标指针移动的图形所在标签的纵坐标值
        lb_move.setLocation(x1, y1);               //设置随鼠标指针移动的图形所在标签的显示位置
        int x2 = x - 52;                           //计算显示文字的标签的横坐标值
        int y2 = y1 + 120;                         //计算显示文字的标签的纵坐标值
        lb_tip.setLocation(x2, y2);                //设置显示文字的标签的显示位置
    }
});
```

说明：上面代码中，x 是鼠标指针在窗体上显示位置的横坐标；y 是鼠标指针在窗体上显示位置的纵坐标；x1 用于指定图片标签中心显示位置的横坐标；y1 用于指定图片标签中心显示位置的纵坐标；x2 用于指定文本标签显示位置的横坐标；y2 用于指定文本标签显示位置的纵坐标。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 FollowMousePictureFrame 窗体类。

（3）将 FollowMousePictureFrame 窗体类的内容面板设置为绝对布局，然后在内容面板上分别添加名称为 lb_move 和 lb_tip 的 JLabel 标签控件，分别用于显示图片和文本。

（4）在窗体 FollowMousePictureFrame 中创建内部线程类 FlareThread，用于改变标签中文字的颜色。FlareThread 类的代码如下：

```
class FlareThread implements Runnable {
    public void run() {
        while (true) {
            int red = (int) (Math.random() * 256);                    //随机生成 RGB 颜色中的 R，即红色
            int green = (int) (Math.random() * 256);                  //随机生成 RGB 颜色中的 G，即绿色
            int blue = (int) (Math.random() * 256);                   //随机生成 RGB 颜色中的 B，即蓝色
            lb_tip.setForeground(new Color(red, green, blue));        //设置标签上文字的前景色
            try {
                Thread.sleep(500);                                    //线程休眠 500 毫秒
            } catch (Exception e) {
            }
        }
    }
}
```

（5）为窗体类 FollowMousePictureFrame 添加鼠标移动事件，实现在鼠标指针移动时，使显示图片的标签和显示文字的标签随着鼠标指针移动，该事件的代码如下：

```
addMouseMotionListener(new MouseMotionAdapter() {
    public void mouseMoved(final MouseEvent e) {
        int x = e.getX();                                             //获得鼠标在窗体容器中的横坐标值
        int y = e.getY();                                             //获得鼠标在窗体容器中的纵坐标值
        int w = lb_move.getWidth();                                   //获得随鼠标移动的图形所在标签的宽度
        int h = lb_move.getHeight();                                  //获得随鼠标移动的图形所在标签的高度
        int x1 = x - w / 2;                                           //计算出随鼠标移动的图形所在标签的横坐标值
        int y1 = y - h / 2;                                           //计算出随鼠标移动的图形所在标签的纵坐标值
        lb_move.setLocation(x1, y1);                                  //设置随鼠标移动的图形所在标签的显示位置
        int x2 = x - 52;                                              //计算显示文字的标签的横坐标值
        int y2 = y1 + 120;                                            //计算显示文字的标签的纵坐标值
        lb_tip.setLocation(x2, y2);                                   //设置显示文字的标签的显示位置
    }
});
```

注意：在使用本实例时，需要将项目需要的图片文件放到项目的 src 文件夹的子文件夹 image 中，否则程序将无法正常运行。

秘笈心法

心法领悟 071：任意移动容器中的控件。

在 Java 应用程序中，为了使容器中的控件能任意移动，必须要将容器设置为绝对布局，即设置为 null 布局，这样就可以使用容器的 setLocation()方法任意指定控件在容器中的位置，如果是其他布局，则无法实现。

实例 072 通过键盘移动图片

光盘位置：光盘\MR\072　　中级　　趣味指数：★★★★

实例说明

本实例演示如何利用键盘移动图片。运行程序，可以通过键盘上的上、下、左、右方向键移动窗体中显示有图片的标签，效果如图 4.13 所示。

图 4.13　通过键盘移动图片

关键技术

本实例通过键盘事件类 KeyEvent 的 getKeyCode()方法获得按键的整数代码，并使用 if 语句判断是否与所按方向键的整数代码相同。

（1）KeyEvent 类的 getKeyCode()方法用于获得键盘上实际按键的整数代码，该方法的定义如下：

```
public int getKeyCode()
```

参数说明

返回值：键盘上实际按键的整数代码。

（2）键盘上非数字键上与上、下、左、右方向键相对应的整数代码分别为 VK_UP、VK_DOWN、VK_LEFT、VK_RIGHT，可以使用 KeyEvent 类进行调用，使用方法如下。

- ☑ KeyEvent.VK_UP：表示非数字键上的向上方向键。
- ☑ KeyEvent.VK_DOWN：表示非数字键上的向下方向键。
- ☑ KeyEvent.VK_LEFT：表示非数字键上的向左方向键。
- ☑ KeyEvent.VK_RIGHT：表示非数字键上的向右方向键。

（3）使用 if 语句判断是否按了非数字键上的向上方向键，可以使用如下代码实现：

```
if (e.getKeyCode() == KeyEvent.VK_UP)                    //如果按了非数字键上的向上方向键，则条件成立
```

说明：上面代码中的 e 是键盘事件中 KeyEvent 类的实例，如果按了非数字键上的向上方向键，则上述 if 语句的条件表达式就是成立的。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 KeyboardMovePictureFrame。

（3）将 KeyboardMovePictureFrame 窗体类的内容面板设置为绝对布局，然后在内容面板上添加一个名称为 lb_move 的 JLabel 标签控件，用于显示图片。

（4）为窗体类 KeyboardMovePictureFrame 的内容面板添加键盘按键事件，实现通过键盘上非数字键上的方向键移动窗体上显示图片的标签控件。窗体类内容面板的键盘按键事件的代码如下：

```
getContentPane().addKeyListener(new KeyAdapter() {
    public void keyPressed(final KeyEvent e) {
        int x = lb_move.getLocation().x;                    //获得移动标签的 x 坐标
        int y = lb_move.getLocation().y;                    //获得移动标签的 y 坐标
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            lb_move.setLocation(x - 10, y);                 //向左移动，x 坐标减小
        } else if (e.getKeyCode() == KeyEvent.VK_UP) {
            lb_move.setLocation(x, y - 10);                 //向上移动，y 坐标减小
        } else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            lb_move.setLocation(x + 10, y);                 //向右移动，x 坐标增加
        } else if (e.getKeyCode() == KeyEvent.VK_DOWN) {
            lb_move.setLocation(x, y + 10);                 //向下移动，y 坐标增加
        }
    }
});
```

注意：在使用本实例时，需要将项目需要的图片文件放到项目的 src 文件夹的子文件夹 image 中，否则程序将无法正常运行。

秘笈心法

心法领悟 072：解决按方向键无法移动图片的问题。

本实例为窗体类的内容面板添加了键盘按键事件，但是当程序运行时，按键盘上的方向键却无法移动图片，这是由于窗体类的内容面板没有获得焦点，为此可以在窗体显示时通过 requestFocus()方法使窗体类的内容面板

获得焦点，这样就可以通过键盘移动窗体上显示的图片了。

实例 073 图片动态拉伸

光盘位置：光盘\MR\073　　中级　　趣味指数：★★★

实例说明

本实例通过 Java 的绘图技术实现了使图片动态拉伸的功能。运行程序，窗体上的图片将动态放大和缩小，并不断重复该过程，效果如图 4.14 和图 4.15 所示。

图 4.14　图片缩小的效果

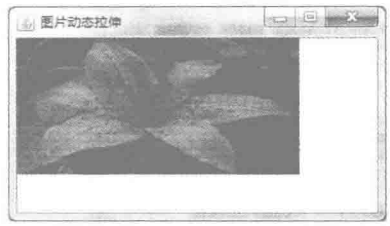


图 4.15　图片放大的效果

关键技术

本实例通过 Graphics 类的 drawImage()方法绘制图片，并使用线程动态改变所绘制图片的宽度和高度，从而实现了图片动态拉伸的动画效果。

（1）使用 Graphics 类的 drawImage()方法，可以在指定位置绘制指定大小的图片。

（2）在实现 Runnable 接口的 run()方法中，使用标记变量确定是放大还是缩小图片，并在图片放大和缩小到一定程度时，改变标记变量的值，以防止图片的大小超出窗体的范围，代码如下：

```
if (flag) {
    width+=2;                                    //调整宽度值
    height++;                                    //调整高度值
    if (width >= getWidth() || height >= getHeight()) {
        flag = false;                            //达到图像的宽度或高度时，改变标记变量的值
    }
} else {
    width -= 2;                                  //调整宽度值
    height--;                                    //调整高度值
    if (width <= 0 || height <= 0) {
        flag = true;                             //图像的宽度或高度小于等于 0 时，改变标记变量的值
    }
}
```

说明：上面代码中的 flag 就是标记变量，当其值为 true 时，放大图片；当其值为 false 时，缩小图片。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的 DynamicFlexImageFrame 窗体类。

（3）在 DynamicFlexImageFrame 窗体类中创建内部面板类 DynamicFlexPanel，该面板类实现了 Runnable 接口，并重写了 JComponent 类的 paint()方法，同时也实现了 Runnable 接口的 run()方法，并在 paint()方法中使用 Graphics 类的 drawImage()方法绘制图片，然后在 run()方法中实现改变图片大小的操作。

（4）将内部面板类 DynamicFlexPanel 的实例添加到窗体类 DynamicFlexImageFrame 的内容面板上，用于在窗体上显示动态拉伸的图片。内部面板类 DynamicFlexPanel 的代码如下：

```
class DynamicFlexPanel extends JPanel implements Runnable {
    private boolean flag = true;                          //标记变量
    int width = 0;                                        //调整图像宽度的变量
    int height = 0;                                       //调整图像高度的变量
    public void paint(Graphics g) {
        g.clearRect(0, 0, getWidth(), getHeight());       //清除绘图上下文的内容
        g.drawImage(img, 0, 0, width, height, this);      //绘制指定大小的图片
    }
    public void run() {
        while (true) {
            if (flag) {
                width+=2;                                 //调整宽度值
                height++;                                 //调整高度值
                if (width >= getWidth() || height >= getHeight()) {
                    flag = false;                         //达到图像的宽度或高度时，改变标记变量的值
                }
            } else {
                width -= 2;                               //调整宽度值
                height--;                                 //调整高度值
                if (width <= 0 || height <= 0) {
                    flag = true;                          //图像的宽度或高度小于等于 0 时，改变标记变量的值
                }
            }
            repaint();                                    //调用 paint()方法
            try {
                Thread.sleep(20);                         //线程休眠 20 毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

注意：在使用本实例时，需将项目需要的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序将无法正常运行。

秘笈心法

心法领悟 073：实现图片的放射效果。

将本实例中 paint()方法中的语句 g.clearRect(0, 0, getWidth(), getHeight(), this);删除，并在 run()方法中只使图片从大到小变化，即可实现图片的放射效果。

实例 074　桌面弹球

光盘位置：光盘\MR\074　　中级　　趣味指数：★★★★

实例说明

本实例实现了桌面弹球动画。运行程序，小球将在窗体中移动，当遇到窗体边缘时，小球会改变运动轨迹，效果如图 4.16 所示。

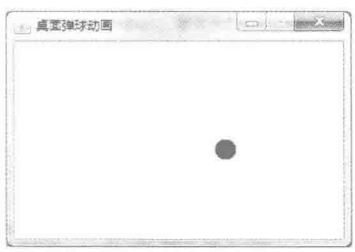


图 4.16　桌面弹球动画

关键技术

本实例通过自定义标签实现，该自定义标签继承自 JLabel 类，同时实现了 Runnable 接口，并重写了控件的 paintComponent()方法，该方法是从 Container 类继承来的，用于在自定义标签上绘制小球，并在自定义标签类的构造方法中指定标签的首选大小，使其正好与小球的大小相同，最后在 run()方法中对绘制有小球的自定义标签在父级容器中的运动轨迹进行控制，这主要是通过从 Component 类继承的方法来实现的。

（1）在自定义标签中重写控件的 paintComponent()方法，用于在标签上绘制小球，该方法的代码如下：

```
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);                    //调用父类的方法
    g.setColor(ballColor);                      //设置默认颜色
    g.fillOval(0, 0, width, height);            //在标签上绘制球体
}
```

（2）通过 getParent()方法，可以获得当前控件所在父级容器的对象，该方法是从 Component 类继承来的，其定义如下：

```
public Container getParent()
```

参数说明

返回值：当前控件的父级容器对象。

（3）通过 getLocation()方法，可以获得当前控件所在父级容器的位置对象，该方法也是从 Component 类继承来的，其定义如下：

```
public Point getLocation()
```

参数说明

返回值：是一个 Point 对象，表示当前控件左上角在父级容器中的坐标。

（4）通过 setLocation()方法，可以设置当前控件在父级容器中的位置，该方法也是从 Component 类继承来的，其定义如下：

```
public void setLocation(int x, int y)
```

参数说明

❶ x：指定当前控件左上角在父级容器中新位置的 x 坐标。

❷ y：指定当前控件左上角在父级容器中新位置的 y 坐标。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 BallFrame 窗体类和一个继承 JLabel 类并实现 Runnable 接口的自定义标签类 BallPanel。

（3）在标签类 BallPanel 中重写控件的 paintComponent()方法，用于在标签上绘制小球，在实现 Runnable 接口的 run()方法中，每隔一小段时间就对自定义标签在父级容器中的位置进行改变，从而实现桌面弹球的动画效果。

（4）自定义标签类 BallPanel 的代码如下：

```
package com.zzk;
import java.awt.*;
import javax.swing.JLabel;
/**
 * @author 张振坤
 */
public class BallPanel extends JLabel implements Runnable {
    private int r = 10;                                 //小球半径
    private int width = r * 2;                          //球宽度
    private int height = r * 2;                         //球高度
    private Color ballColor = Color.BLUE;               //默认颜色
    public BallPanel() {
        setPreferredSize(new Dimension(width, height)); //初始化大小
        new Thread(this).start();                       //启动小球跳跃线程
```

```
    }
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(ballColor);                          //设置默认颜色
        g.fillOval(0, 0, width, height);                //在标签上绘制球体
    }
    @Override
    public void run() {
        Container parent = getParent();                 //获得当前标签的父级容器对象
        Point myPoint = getLocation();                  //获取初始位置
        while (true) {                                  //循环读取父容器对象
            if (parent == null) {
                try {
                    Thread.sleep(50);                   //线程休眠
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                myPoint = getLocation();                //获取初始位置
                parent = getParent();                   //获得当前标签的父级容器对象
            } else {                                    //如果已经获取到父容器
                break;                                  //跳出循环
            }
        }
        int sx = myPoint.x;                             //x 坐标
        int sy = myPoint.y;                             //y 坐标
        int step = (int) (Math.random() * 10) % 3 + 1;  //移动步进
        int dx = (Math.random() * 100) >= 50 ? step : -step;  //水平步进值
        step = (int) (Math.random() * 10) % 3 + 1;      //移动步进
        int dy = (Math.random() * 100) >= 50 ? step : -step;  //垂直步进值
        int stime = (int) (Math.random() * 80 + 10);    //随机移动速度
        while (parent.isVisible()) {
            int parentWidth = parent.getWidth();        //容器宽度
            int parentHeight = parent.getHeight();      //容器高度
            setLocation(sx, sy);                        //指定小球的位置
            try {
                Thread.sleep(stime);                    //通过休眠改变移动速度
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            sx = sx + dx * 5;                           //水平坐标偏移 5 个像素
            sy = sy + dy * 5;                           //垂直坐标偏移 5 个像素
            if (sy > parentHeight - height || sy < 0)   //检测垂直边界
                dy = -dy;                               //防止坐标超出垂直边界
            if (sx > parentWidth - width || sx < 0)     //检测水平边界
                dx = -dx;                               //防止坐标超出水平边界
        }
    }
}
```

秘笈心法

心法领悟 074：在窗体中同时进行多球弹动。

本实例运行后，只有一个小球在窗体上弹动，如果希望有多个球在窗体上弹动，可以在窗体类 BallFrame 中创建多个自定义标签类 BallPanel 的实例，即创建多个小球，并使用 setBounds()方法指定位置，然后添加到窗体类 BallFrame 的内容面板中，再运行程序就会有多个小球在窗体上弹动。

实例 075　循环滚动图片

光盘位置：光盘\MR\075　　中级　　趣味指数：★★★

实例说明

本实例使用 Java 的绘图技术和多线程技术实现在窗体上循环滚动图片的功能。运行程序，图片将从窗体的

左侧向右侧移动，当移动到右侧边缘时返回，即向左侧移动，并重复上述过程循环滚动，效果如图 4.17 所示。

图 4.17　循环滚动图片

关键技术

在循环移动图片时，当图片右侧边缘移动到所在面板右侧边缘时，图片不再向右移动，这可以通过判断图片左侧边缘的 x 坐标值是否大于等于所在面板的宽度减去图片的宽度进行控制；当图片左侧边缘移动到所在面板的左侧边缘时，图片不再向左移动，这可以通过判断图片左侧边缘的 x 坐标值是否小于等于 0 进行控制。

（1）当图片向右移动时，为了使图片右侧边缘移动到所在面板右侧边缘时不再向右移动，可以用如下代码来实现：

```
if (x >= getWidth() - img.getWidth(this)) {          //图片的 x 坐标值大于等于面板与图片宽度的差
    x = getWidth() - img.getWidth(this);             //图片的 x 坐标值等于面板与图片宽度的差
}
```

说明：上面代码中的 x 是图片面板左上角的 x 坐标，getWidth() - img.getWidth(this)是所在面板宽度与图片面板宽度的差值。

（2）当图片向左移动时，为了使图片左侧边缘移动到所在面板左侧边缘时不再向左移动，可以用如下代码来实现：

```
if ( x <= 0 ) {                                      //图片的 x 坐标值小于等于 0，即超出了面板的左侧边缘
    x = 0;                                           //图片的 x 坐标值等于 0
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 CircularRollPictureFrame 窗体类。

（3）在 CircularRollPictureFrame 窗体类中创建内部面板类 CircularRollPicturePanel，并重写 JComponent 类的 paint()方法，实现 Runnable 接口中的 run()方法。

（4）将内部面板类 CircularRollPicturePanel 的实例添加到窗体类 CircularRollPictureFrame 的内容面板上，用于在窗体上显示循环滚动的图片。内部面板类 CircularRollPicturePanel 的代码如下：

```
private class CircularRollPicturePanel extends JPanel implements Runnable {
    int x = 0;                                                          //定义图片移动位置的 x 坐标
    int y = 30;                                                         //定义图片移动位置的 y 坐标
    URL imgUrl = CircularRollPictureFrame.class.getResource("/image/picture.png");  //获取图片资源的路径
    Image img = Toolkit.getDefaultToolkit().getImage(imgUrl);           //获取图像对象
    public void paint(Graphics g){
        g.clearRect(0, 0, getWidth(), getHeight());                     //清除面板上的内容
        g.drawImage(img, x, y, this);                                   //在面板的指定位置绘制图像
    }
    public void run() {
        boolean flag = true;                                            //声明标记变量
        while (true){
            if (flag) {                                                 //标记变量为 true
                x = x + 10;                                             //图片 x 坐标值加 10
                if (x >= getWidth() - img.getWidth(this)) {             //图片的 x 坐标值大于等于面板与图片宽度的差
                    x = getWidth() - img.getWidth(this);                //图片的 x 坐标值等于面板与图片宽度的差
```

```
                flag = false;                          //设置标记变量为 false
            }
        }else {                                        //标记变量为 false
            x = x - 10;                                //图片 x 坐标值减 10
            if ( x <= 0 ) {                            //图片的 x 坐标值小于等于 0
                x = 0;                                 //图片的 x 坐标值等于 0
                flag = true;                           //设置标记变量为 true
            }
        }
        try {
            Thread.sleep(200);                         //休眠 200 毫秒
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        repaint();                                     //调用 paint()方法
    }
  }
}
```

秘笈心法

心法领悟 075：制作图片在窗体内任意位置移动的动画效果。

本实例实现了图片在窗体上左右循环滚动的动画效果，如果需要，可以将图片制作成在窗体内任意位置移动的动画效果，方法是通过线程同时控制图片绘制点的横坐标和纵坐标。

实例 076　撞球动画

光盘位置：光盘\MR\076　　中级　　趣味指数：★★★★★

实例说明

本实例实现了一个撞球动画，并模拟了两球相撞时会产生挤压变形的变化。运行程序，两个小球在窗体上水平相向运动，相撞后又向相反的方向运动，两球没相撞与相撞时的效果分别如图 4.18 和图 4.19 所示。

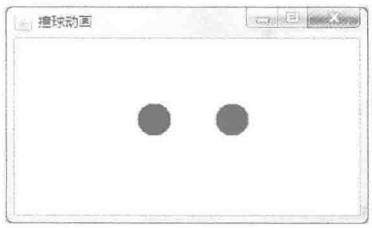


图 4.18　两球没相撞的效果

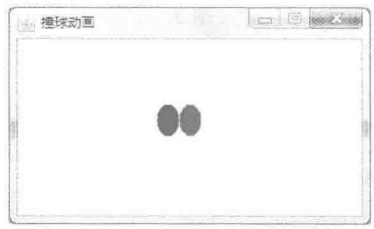


图 4.19　两球相撞时的效果

关键技术

本实例主要实现在两球相遇时使两球的宽度变小，这样就产生了挤压的效果，并通过两个标记变量分别控制两个球的运行方向。

（1）为了使两球相遇时使两球的宽度变小，需要判断两个球是否相撞，代码如下：

```
if (x1 + width >= x2 + 1) {                            //两球相撞
    x1 = x1 + 5;                                       //计算第 1 个球的 x 坐标
    width = width - 10;                                //球的宽度减 10
    x2 = x1 + width;                                   //计算第 2 个球的 x 坐标
    flag1 = false;                                     //设置第 1 个球的标记变量为 false
    flag2 = false;                                     //设置第 2 个球的标记变量为 false
    repaint();                                         //调用 paint()方法
}
```

（2）使用标记变量 flag1 控制第 1 个球的运行方向，为 true 时第 1 个球右移，为 false 时第 1 个球左移，代码如下：

```
if (flag1) {                                    //标记变量为 true，第 1 个球右移
    x1 = x1 + 10;                               //图片 x 坐标值加 10，第 1 个球右移
    width = 30;                                 //球的宽度
} else {                                        //标记变量为 false，第 1 个球左移
    x1 = x1 - 10;                               //图片 x 坐标值减 10，第 1 个球左移
    width = 30;                                 //球的宽度
    if (x1 <= 0) {                              //图片的 x 坐标值小于等于 0
        x1 = 0;                                 //图片的 x 坐标值等于 0
        flag1 = true;                           //设置标记变量为 true
    }
}
```

（3）使用标记变量 flag2 控制第 2 个球的运行方向，为 true 时第 2 个球左移，为 false 时第 2 个球右移，代码如下：

```
if (flag2) {                                    //标记变量为 true，第 2 个球左移
    x2 = x2 - 10;                               //图片 x 坐标值减 10，即第 2 个球左移
    width = 30;
} else {                                        //标记变量为 false，第 2 个球右移
    x2 = x2 + 10;                               //图片 x 坐标值加 10，即第 2 个球右移
    width = 30;                                 //球的宽度
    if (x2 >= getWidth() - width) {             //图片的 x 坐标值大于等于面板的宽度与球的宽度之差
        x2 = getWidth() - width;                //图片的 x 坐标值等于面板的宽度与球的宽度之差
        flag2 = true;                           //设置标记变量为 true
    }
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 BilliardBallFrame 窗体类。

（3）在 BilliardBallFrame 窗体类中创建内部面板类 BilliardBallPanel，重写 JComponent 类的 paint()方法，并实现 Runnable 接口中的 run()方法，完成撞球动画。

（4）将内部面板类 BilliardBallPanel 的实例添加到窗体类 BilliardBallFrame 的内容面板上，用于在窗体上显示撞球动画。内部面板类 BilliardBallPanel 的代码如下：

```
private class BilliardBallPanel extends JPanel implements Runnable {
    int x1 = 0;                                 //定义第 1 个球移动位置的 x 坐标
    int y1 = 60;                                //定义第 1 个球移动位置的 y 坐标
    int x2 = 326 - 30;                          //定义第 2 个球移动位置的初始 x 坐标为窗体宽度减球的宽度
    int y2 = 60;                                //定义第 2 个球移动位置的 y 坐标
    int width = 30;                             //定义球的宽度
    int height = 30;                            //定义球的高度
    public void paint(Graphics g) {
        g.clearRect(0, 0, getWidth(), getHeight());  //清除面板上的内容
        g.setColor(Color.BLUE);                 //设置颜色
        g.fillOval(x1, y1, width, height);      //绘制第 1 个球
        g.setColor(Color.RED);                  //设置颜色
        g.fillOval(x2, y2, width, height);      //绘制第 2 个球
    }
    public void run() {
        boolean flag1 = true;                   //声明第 1 个球的标记变量
        boolean flag2 = true;                   //声明第 2 个球的标记变量
        while (true) {
            //第 1 个球的 x 坐标值加上球的宽度大于等于第 2 个球的 x 坐标值加 1，表示两球相遇
            if (x1 + width >= x2 + 1) {         //两球相撞
                x1 = x1 + 5;                    //计算第 1 个球的 x 坐标
                width = width - 10;             //球的宽度减 10
                x2 = x1 + width;                //计算第 2 个球的 x 坐标
                flag1 = false;                  //设置第 1 个球的标记变量为 false
                flag2 = false;                  //设置第 2 个球的标记变量为 false
                repaint();                      //调用 paint()方法
            } else {                            //两球没相撞
                if (flag1) {                    //标记变量为 true，第 1 个球右移
                    x1 = x1 + 10;               //图片 x 坐标值加 10，第 1 个球右移
                    width = 30;                 //球的宽度
```

```
                } else {                                    //标记变量为 false，第 1 个球左移
                    x1 = x1 - 10;                           //图片 x 坐标值减 10，第 1 个球左移
                    width = 30;                             //球的宽度
                    if (x1 <= 0) {                          //图片的 x 坐标值小于等于 0
                        x1 = 0;                             //图片的 x 坐标值等于 0
                        flag1 = true;                       //设置标记变量为 true
                    }
                }
                if (flag2) {                                //标记变量为 true，第 2 个球左移
                    x2 = x2 - 10;                           //图片 x 坐标值减 10，即第 2 个球左移
                    width = 30;
                } else {                                    //标记变量为 false，第 2 个球右移
                    x2 = x2 + 10;                           //图片 x 坐标值加 10，即第 2 个球右移
                    width = 30;                             //球的宽度
                    if (x2 >= getWidth() - width) {         //图片的 x 坐标值大于等于面板的宽度与球的宽度之差
                        x2 = getWidth() - width ;           //图片的 x 坐标值等于面板的宽度与球的宽度之差
                        flag2 = true;                       //设置标记变量为 true
                    }
                }
            }
            try {
                Thread.sleep(200);                          //休眠 200 毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            repaint();                                      //调用 paint()方法
        }
    }
}
```

秘笈心法

心法领悟 076：实现拍球动画效果。

在拍球时，球接触地面的瞬间也有挤压的效果，实现拍球动画就是使球向下运动，并在球接触地面的瞬间使球的高度变小，产生挤压的效果，然后再使球的高度恢复正常并向上运动，运动到一定高度再向下运动，并重复这个过程。

实例 077　电影胶片特效

中级

光盘位置：光盘\MR\077

趣味指数：★★★

实例说明

本实例使用 Java 的多线程技术实现电影胶片特效。运行程序，窗体上显示 5 个带有图片的标签，并将依次从窗体的右侧向左侧运动，就像放电影时胶片的运动一样，而且每个标签上显示的图片都是在两个图片之间变换的，效果如图 4.20 所示。

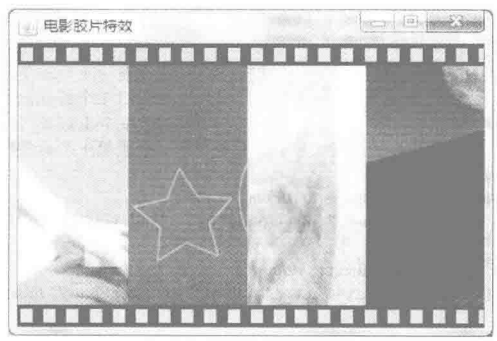


图 4.20　电影胶片特效

关键技术

根据图片的宽度和高度，创建 5 个从左向右运动的标签，并将其中 4 个标签放到绝对布局的面板容器中，另一个则在移动的过程中进行填补，使图片能够连贯地移动，并使用控件的 setBounds()方法指定每个控件的位置和大小，从而实现电影胶片特效。

（1）根据图片的宽度创建 5 个标签，然后计算出每个标签左上角的 x 坐标，代码如下：

```
int x1 = 0;                                          //第 1 个标签显示位置的变量
int x2 = 98;                                         //第 2 个标签显示位置的变量
int x3 = 196;                                        //第 3 个标签显示位置的变量
int x4 = 294;                                        //第 4 个标签显示位置的变量
int x5 = 392;                                        //第 5 个标签显示位置的变量
```

（2）为了连贯地移动图片，需要使每个控件的 x 坐标值不断变化，代码如下：

```
x1 = x1 - 7;                                         //第 1 个标签的左边界减 7，使其左移
x2 = x2 - 7;                                         //第 2 个标签的左边界减 7，使其左移
x3 = x3 - 7;                                         //第 3 个标签的左边界减 7，使其左移
x4 = x4 - 7;                                         //第 4 个标签的左边界减 7，使其左移
x5 = x5 - 7;                                         //第 5 个标签的左边界减 7，使其左移
```

（3）使用控件的 setBounds()方法设置每个标签控件的位置，该方法的定义如下：

```
public void setBounds(int x, int y, int width, int height)
```

参数说明

❶ x：控件左上角的 x 坐标。

❷ y：控件左上角的 y 坐标。

❸ width：控件的宽度。

❹ height：控件的高度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 CinefilmEffectFrame 窗体类。

（3）在 CinefilmEffectFrame 窗体类中创建实现了 Runnable 接口的内部类 CinefilmThread，用于实现电影胶片的动画效果。内部类 CinefilmThread 的代码如下：

```
private class CinefileThread implements Runnable {
    public void run() {
        while (true) {
            x1 = x1 - 7;                                         //第 1 个标签的左边界减 7，使其左移
            x2 = x2 - 7;                                         //第 2 个标签的左边界减 7，使其左移
            x3 = x3 - 7;                                         //第 3 个标签的左边界减 7，使其左移
            x4 = x4 - 7;                                         //第 4 个标签的左边界减 7，使其左移
            x5 = x5 - 7;                                         //第 5 个标签的左边界减 7，使其左移
            label_1.setBounds(x1, 0, 98, 210);                   //设置第 1 个标签的显示位置
            label_2.setBounds(x2, 0, 98, 210);                   //设置第 1 个标签的显示位置
            label_3.setBounds(x3, 0, 98, 210);                   //设置第 1 个标签的显示位置
            label_4.setBounds(x4, 0, 98, 210);                   //设置第 1 个标签的显示位置
            label_5.setBounds(x5, 0, 98, 210);                   //设置第 1 个标签的显示位置
            if (x1 == -98) {                                     //当第 1 个标签的显示位置是-98 时执行
                indexFlag = !indexFlag;                          //改变 indexFlag 的值
                x1 = 392;                                        //设置第 1 个标签的显示位置
                if (indexFlag) {
                    label_1.setIcon(SwingResourceManager.getIcon(
                            CinefilmEffectFrame.class, "/image/6.jpg"));   //indexFlag 为 true 时改变的图片
                } else {
                    label_1.setIcon(SwingResourceManager.getIcon(
                            CinefilmEffectFrame.class, "/image/1.jpg"));   //indexFlag 为 false 时改变的图片
                }
            }
            //省略了其他 4 个标签的代码
            try {
```

```
                Thread.sleep(150);                                    //线程睡眠 150 毫秒
            } catch (Exception ex) {

            }
        }
    }
}
```

秘笈心法

心法领悟 077：标签的其他应用。

本实例中的图片标签还可以用作其他用途，可以将标签内的图片内容自定义为其他图片，然后添加到程序界面中，以达到丰富界面的效果。例如用来装饰圣诞树的标签、为寒冷的冬季自定义的雪花标签和枫叶标签等。

实例 078　随机移动的图片

光盘位置：光盘\MR\078　　中级　　趣味指数：★★★

实例说明

本实例使用 Java 的绘图技术和多线程技术实现随机移动图片的功能。运行程序，图片将在窗体上随机移动，但是不会移出窗体，效果如图 4.21 所示。

图 4.21　随机移动的图片

关键技术

本实例使用 Random 类获得随机数，随机生成图片显示位置的坐标，这需要使用 Random 类的构造方法创建对象，然后使用 nextInt()方法获得随机整数。

（1）使用 Random 类的构造方法创建对象，用于创建一个新的随机数生成器。

该类的构造方法定义如下：

```
public Random()
```

（2）使用 Random 类的对象调用 nextInt()方法生成随机整数，该方法的定义如下：

```
public int nextInt(int n)
```

参数说明

❶ n：要返回的随机数的范围必须为正数。

❷ 返回值：下一个伪随机数，是在此随机数生成器序列中 0（包括）和 n（不包括）之间均匀分布的整数值。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 RandomMovePictureFrame 窗体类。

（3）在 RandomMovePictureFrame 窗体类中创建内部面板类 RandomMovePicturePanel，重写 JComponent

类的 paint()方法，并实现 Runnable 接口的 run()方法，从而实现随机移动图片的功能。

（4）将内部面板类 RandomMovePicturePanel 的实例添加到窗体类 RandomMovePictureFrame 的内容面板上，用于在窗体上显示随机移动的图片。面板类 RandomMovePicturePanel 的代码如下：

```
private class RandomMovePicturePanel extends JPanel implements Runnable {
    Random random = new Random();                          //创建 Random 对象
    int x = 0;                                             //定义图片移动位置的 x 坐标
    int y = 0;                                             //定义图片移动位置的 y 坐标
    URL imgUrl = RandomMovePictureFrame.class
            .getResource("/image/picture.png");            //获取图片资源的路径
    Image img = Toolkit.getDefaultToolkit().getImage(imgUrl);  //获取图像对象
    public void paint(Graphics g) {
        g.clearRect(0, 0, getWidth(), getHeight());        //清除面板上的内容
        g.drawImage(img, x, y, this);                      //在面板的指定位置绘制图像
    }
    public void run() {
        while (true) {
            x = random.nextInt(winWIDTH - 110);            //随机获得图片移动位置的 x 坐标
            y = random.nextInt(winHEIGHT - 140);           //随机获得图片移动位置的 y 坐标
            try {
                Thread.sleep(200);                         //休眠 200 毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            repaint();                                     //调用 paint()方法
        }
    }
}
```

秘笈心法

心法领悟 078：在窗体中左右循环移动图片。

本实例实现了图片在窗体上任意位置移动的动画效果，如果需要，可以制作图片在窗体上左右循环滚动的动画效果。另外，当图片向两侧边缘移动时，如果图片处于边缘位置就不再向外移动，而是向相反的方向移动。

实例 079　雪花飘落动画

高级

光盘位置：光盘\MR\079

趣味指数：★★★★

实例说明

本实例使用 Java 的绘图技术实现雪花飘落动画。运行程序，效果如图 4.22 所示。

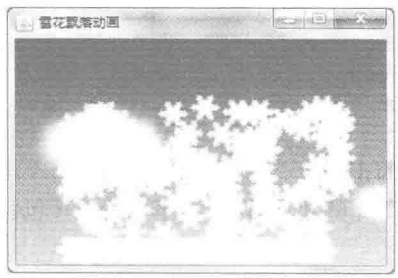


图 4.22　雪花飘落动画

关键技术

本实例中创建继承自 JLabel 类的自定义标签，同时实现了 Runnable 接口，并在 run()方法中实现在父级容

器中放置显示有雪花图片的自定义标签对象。

（1）通过 getParent()方法，可以获得当前控件所在父级容器的对象。

（2）通过 getLocation()方法，可以获得当前控件所在父级容器的位置对象。

（3）通过 setLocation()方法，可以设置当前控件在父级容器中的位置。

📖 **说明**：上述 3 种方法的详细说明，可以参考实例 074 中的关键技术部分。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类、一个继承 JLabel 类并实现 Runnable 接口的自定义标签类 SnowFlakeLabel，以及一个背景面板类 BackgroundPanel。

（3）在标签类 SnowFlakeLabel 中的实现 Runnable 接口的 run()方法中，每隔一小段时间就对自定义标签在父级容器中的位置进行改变，从而实现雪花飘落的效果。

（4）自定义标签类 SnowFlakeLabel 的代码如下：

```
package com.zzk;
import java.awt.*;
import javax.swing.*;
/**
 * @author 张振坤
 */
public class SnowFlakeLabel extends JLabel implements Runnable {
    private final static ImageIcon snow = new ImageIcon(SnowFlakeLabel.class
            .getResource("/image/snowflake.png"));
    private int width = snow.getIconWidth();                    //雪花的宽度
    private int height = snow.getIconHeight();                  //雪花的高度
    /**
     * 构造方法
     */
    public SnowFlakeLabel() {
        setSize(new Dimension(width, height));                  //雪花的初始化大小
        setIcon(snow);                                          //指定图标
        new Thread(this).start();                               //创建并启动线程
    }
    public void run() {
        Container parent = getParent();                         //获取父容器对象
        Point myPoint = getLocation();                          //获取初始位置
        while (true) {                                          //循环读取父容器对象
            if (parent == null) {
                try {
                    Thread.sleep(50);                           //线程休眠
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                myPoint = getLocation();                        //获取初始位置
                parent = getParent();                           //获取父容器对象
            } else {                                            //如果已经获取到父容器
                break;                                          //跳出循环
            }
        }
        int sx = myPoint.x;                                     //x 坐标
        int sy = myPoint.y;                                     //y 坐标
        int stime = (int) (Math.random() * 30 + 10);            //随机移动速度
        int parentHeight = parent.getHeight();                  //容器高度
        while (parent.isVisible() && sy < parentHeight - height) {
            setLocation(sx, sy);                                //指定位置
            try {
                Thread.sleep(stime);                            //线程休眠
```

```
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                sy += 2;                                    //垂直偏移 2 个像素
            }
        }
    }
```

（5）在 MainFrame 窗体类中，为背景面板类 BackgroundPanel 的实例添加鼠标移动事件，用于向背景面板中添加自定义雪花标签对象，代码如下：

```
backgroundPanel.addMouseMotionListener(new MouseAdapter() {
    public void mouseMoved(MouseEvent e) {              //鼠标移动事件
        SnowFlakeLabel snow = new SnowFlakeLabel();     //创建雪花飘落标签
        Point point = e.getPoint();                     //获得鼠标指针位置
        snow.setLocation(point);                        //指定雪花在背景面板上的位置
        backgroundPanel.add(snow);                      //将雪花添加到背景面板上
    }
});
```

秘笈心法

心法领悟 079：利用 update()方法消除动画闪烁问题。

AWT 接收到一个 Applet 的重绘请求时，会调用 update()方法。默认情况下，update()方法会清除 Applet 中的背景，然后调用 paint()方法。重载 update()方法就可以将以前在 paint()方法中的绘图代码包含在 update()方法中，从而避免了重绘时将整个区域清除。

实例 080　图片旋转动画

光盘位置：光盘\MR\080　　中级　　趣味指数：★★★★

实例说明

本实例实现图片旋转动画。运行程序，效果如图 4.23 所示。

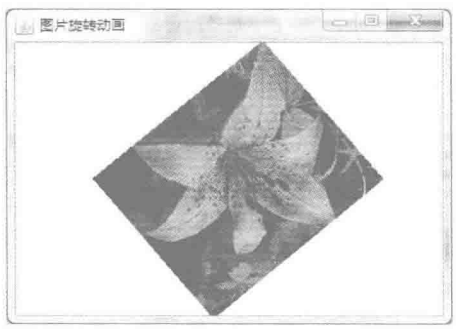


图 4.23　图片旋转动画

关键技术

本实例是使用 Graphics2D 类的 translate()和 rotate()方法实现的，其中 translate()方法用于平移坐标轴，rotate()方法用于旋转绘图上下文。

（1）Graphics2D 类的 translate()方法用于平移坐标轴，该方法的定义如下：

```
public abstract void translate(int x, int y)
```

参数说明

❶ x：指定的 x 坐标。

❷ y：指定的 y 坐标。

（2）Graphics2D 类的 rotate()方法用于旋转绘图上下文，该方法的定义如下：

参数说明

public abstract void rotate(double theta)

theta：以弧度为单位的旋转角度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 RotateImageFrame 窗体类。

（3）在 RotateImageFrame 窗体类中创建内部面板类 RotatePanel，重写 JComponent 类的 paint()方法，用于绘制旋转的图片，并使该类实现 Runnable 接口中的 run()方法，用于改变图片的旋转角度。

（4）将内部面板类 RotatePanel 的实例添加到窗体类 RotateImageFrame 的内容面板上，用于在窗体上显示旋转的图片。内部面板类 RotatePanel 的代码如下：

```
class RotatePanel extends JPanel implements Runnable {
    int angle = 0;                                          //初始旋转角度
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;                     //获得 Graphics2D 对象
        g2.translate(190, 120);                             //平移坐标轴
        g2.clearRect(-190, -120, getWidth(), getHeight());  //清除画布上的内容
        g2.rotate(Math.toRadians(angle));                   //旋转画布
        g2.drawImage(img, -95, -80, 190, 160, this);        //绘制指定大小的图片
    }
    public void run() {
        while (true) {
            angle = (angle + 10) % 360;                     //计算旋转角度
            try {
                Thread.sleep(200);                          //休眠 200 毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            repaint();                                      //调用 paint()方法
        }
    }
}
```

秘笈心法

心法领悟 080：工程学常用量 Math。

本实例中，paint 方法中旋转画布的语句 g2.rotate(Math.*toRadians*(angle));中的 Math 为工程学的常用量，表示用度数测量的角转换为近似相等的用弧度测量的角，Java 中的 Math 类也提供了一些在数学计算中经常使用的方法，通过这些方法可以完成大部分数学计算操作，使用本实例讲解的方法还可以制作出风车、风扇转动等逼真的动画效果。

实例 081　图片闪现动画

光盘位置：光盘\MR\081　　中级　　趣味指数：★★★★

实例说明

本实例使用 Java 的绘图和多线程技术，实现图片闪现动画。运行程序，窗体上会出现闪现的图片，即窗体上的图片在很短的时间内，交替消失和出现，效果如图 4.24 和图 4.25 所示。

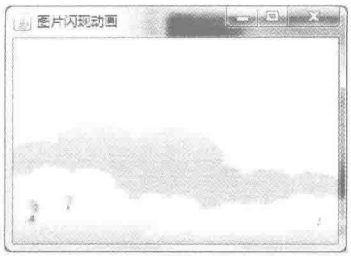


图 4.24　无闪现图片的效果

图 4.25　有闪现图片的效果

关键技术

本实例通过 Java 绘图技术绘制图片，通过多线程技术控制图片是否出现，从而实现了图片闪现的动画特效。

（1）使用 Graphics 类的 drawImage()方法完成图片的绘制。

（2）通过实现 Runnable 接口实现多线程，并在 run()方法中通过定义标记变量来控制图片的闪现，当需要显示闪现图片时，就通过指定图片的 URL 创建图像对象；当不需要显示闪现的图片时，就通过空字符串创建一个空的图像对象。run()方法中通过标记变量控制闪现图片的代码如下：

```
if (flag) {                                                    //flag 为 true
    flag = false;                                              //赋值为 false
    fadeImage = Toolkit.getDefaultToolkit().getImage(imgUrl);  //通过指定图片的 URL 获取图像对象
} else {
    flag = true;                                               //赋值为 true
    fadeImage = Toolkit.getDefaultToolkit().getImage("");      //获取空的图像对象
}
```

说明：上面代码中的 flag 是一个标记变量，为 true 时，通过指定图片的 URL 创建图像对象；为 false 时，创建一个空的图像对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PictureFadeFrame 窗体类。

（3）在 PictureFadeFrame 窗体类中创建内部面板类 PictureFadePanel，该面板类实现了 Runnable 接口，并重写 JComponent 类的 paint()方法和实现 Runnable 接口的 run()方法。在 paint()方法中完成绘制闪现图片的功能；在 run()方法中控制是否显示闪现图片。

（4）将内部面板类 PictureFadePanel 的实例添加到窗体类 PictureFadeFrame 的内容面板上，用于在窗体上显示图片闪现特效。内部面板类 PictureFadePanel 的代码如下：

```
class TextFadePanel extends JPanel implements Runnable {
    boolean flag = true;                                                    //标记变量
    String value = "";                                                      //存放绘制内容的变量
    public void paint(Graphics g) {
        g.clearRect(0, 0, getWidth(),getHeight());                          //清除绘图上下文的内容
        g.drawImage(img,0,0,getWidth(),getHeight(),this);                   //绘制图像
        g.drawImage(fadeImage,10,10,getWidth()-20,getHeight()-20,this);     //绘制闪现的图像对象
    }
    public void run() {
        try {
            while (true) {                                                  //读取内容
                Thread.sleep(200);                                          //当前线程休眠 200 毫秒
                if (flag) {                                                 //flag 为 true
                    flag = false;                                           //赋值为 false
                    fadeImage = Toolkit.getDefaultToolkit().getImage(imgUrl);   //获取图像对象
                } else {
                    flag = true;                                            //赋值为 true
```

```
                fadeImage = Toolkit.getDefaultToolkit().getImage("");          //获取图像对象
            }
            repaint();                                                         //调用 paint()方法
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

秘笈心法

心法领悟 081：实现不同图片的交替显示。

本实例中，当标记变量 flag 为 false 时，如果不是创建一个空的图像对象，而是通过指定图片的 URL 创建新的图像对象，就可以实现不同图片的交替显示，并可以通过调整线程的休眠时间对图片交替显示的速度进行控制。

实例 082　帧动画效果

光盘位置：光盘\MR\082　　高级　　趣味指数：★★★★★

实例说明

本实例利用多线程技术实现帧动画效果。运行程序，将显示卡通小人吃饼干的全过程，效果如图 4.26 所示。单击窗体上的“暂停”按钮，可以暂停动画的播放；单击“继续”按钮，可以继续播放动画；单击“停止”按钮，将停止动画的播放；单击“播放”按钮，又开始重新播放动画。

图 4.26　帧动画效果

关键技术

本实例通过将作为帧动画的图片文件按数字 1、2、3 等为主文件名顺序进行命名，然后使用多线程技术，每隔一段时间就动态生成一幅图片的完整路径，并使用 JLabel 标签的 setIcon()方法为标签指定图片，从而实现帧动画效果。

（1）本实例需要 8 幅图片，分别按顺序命名为 1.jpg、2.jpg、3.jpg、4.jpg、5.jpg、6.jpg、7.jpg 和 8.jpg，这样就可以定义一个 int 型变量 index，使其初始值为 1，然后与字符串.jpg 连接形成图片文件的完整名称 1.jpg，并在一段时间间隔后使 index 加 1，重复上述过程，即可得到其他图片文件的名称。当 index 达到 8 时，为 index 赋值为 1，这样就可以循环获得图片文件的名称了。

（2）在线程的 run()方法中生成每幅图片的完整路径，并使用 JLabel 类的 setIcon()方法为标签指定图片，主要代码如下：

```
String picture = "/image/";                                    //创建图片存放位置和文件名的变量
index++;                                                       //图片的主文件名索引加 1
if (index <= 8) {
    picture = picture + index + ".jpg";                        //通过索引获得图片的位置和文件名
```

```
} else {
    index = 1;                                                    //图片的主文件名索引赋值为 1
    picture = picture + index + ".jpg";                           //通过索引获得图片的位置和文件名
}
label.setIcon(SwingResourceManager.getIcon(FrameActionFrame.class, picture)); //改变标签上显示的图片，实现动画效果
```

说明：上面代码中，picture 变量的值是图片文件存放的位置，即项目的 src 文件夹的子文件夹 image；index 是代表图片文件主文件名的 int 值。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 FrameActionFrame 窗体类。

（3）在 FrameActionFrame 窗体类中创建内部线程类 ActionThread，用于完成图片文件的切换，改变标签上的图标实现帧动画效果。线程类 ActionThread 的代码如下：

```
private class ActionThread implements Runnable {
    private int index = 1;                                        //用于控制图形文件的主文件名
    public void run() {
        while (true) {
            if (stop) {
                thread = null;                                    //销毁线程
                break;                                            //跳出循环，结束线程的执行
            }
            if (flag) {
                String picture = "/image/";                       //创建图片存放位置和文件名的变量
                index++;
                if (index <= 8) {
                    picture = picture + index + ".jpg";           //通过索引获得图片的位置和文件名
                } else {
                    index = 1;
                    picture = picture + index + ".jpg";           //通过索引获得图片的位置和文件名
                }
                label.setIcon(SwingResourceManager.getIcon(
                        FrameActionFrame.class, picture));        //改变标签上显示的图片实现动画效果
                try {
                    Thread.sleep(200);                            //线程睡眠 200 毫秒
                } catch (Exception ex) {

                }
            }
        }
    }
}
```

（4）在窗体类 FrameActionFrame 中定义两个 boolean 型的成员变量 flag 和 stop，其中 flag 用于标识动画播放、暂停和继续，stop 是用于标识动画播放和停止的成员变量，代码如下：

```
boolean flag = true;                                              //用于标识动画播放、暂停和继续的成员变量
boolean stop = false;                                             //用于标识动画播放和停止的成员变量
```

（5）窗体类 FrameActionFrame 上的“播放”按钮用于实现播放帧动画，其事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent arg0) {
        if (thread == null) {
            thread = new Thread(new ActionThread());              //如果线程为空，则创建线程对象
        }
        if (!thread.isAlive()) {                                  //如果线程不是活动线程则执行下面的代码，启动线程的执行
            //stop 为 false，flag 为 true 表示执行帧动画
            stop = false;
            flag = true;
            thread.start();                                       //启动线程的执行
        }
    }
});
```

（6）窗体类 FrameActionFrame 上的“暂停”按钮用于实现暂停帧动画的播放，其事件代码如下：

```
button_1.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent arg0) {
        flag = false;                    //暂停动画播放
    }
});
```

（7）窗体类 FrameActionFrame 上的“继续”按钮用于实现继续播放暂停后的帧动画，其事件代码如下：

```
button_3.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent arg0) {
        flag = true;                     //继续播放动画
    }
});
```

（8）窗体类 FrameActionFrame 上的“停止”按钮用于实现停止帧动画的播放，其事件代码如下：

```
button_2.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent arg0) {
        stop = true;                     //停止播放动画
    }
});
```

秘笈心法

心法领悟 082：Java 中的动画。

在 Java 中实现动画的基本原理就是通过线程和循环等，在指定的时间间隔内不断地改变事先准备好的图片和文字，给人以动画的感觉，从而实现动画特效。

实例 083　水波动画

光盘位置：光盘\MR\083　　高级　　趣味指数：★★★★

实例说明

本实例实现图片的水波动画。运行程序，将在窗体上显示图片的水波动画，就像风吹水面产生波纹的效果，如图 4.27 所示。

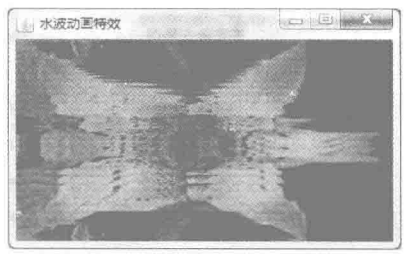


图 4.27　水波动画特效

关键技术

本实例主要是使用 Graphics 类的 copyArea()方法复制图像区域，并通过多线程技术在指定的时间间隔内，使所复制图片区域的位置发生改变，从而实现最终的水波动画效果。

使用 Graphics 类的 copyArea()方法可以复制图像区域，该方法的定义如下：

```
public abstract void copyArea(int x, int y, int width, int height, int dx, int dy)
```

参数说明

❶ x：源矩形的 x 坐标。

❷ y：源矩形的 y 坐标。

❸ width：源矩形的宽度。

❹ height：源矩形的高度。

❺ dx：复制像素的水平距离。

❻ dy：复制像素的垂直距离。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 WaterWaveActionFrame 窗体类。

（3）在 WaterWaveActionFrame 窗体类中创建内部面板类 WaterWaveActionPanel，并实现 Runnable 接口，用于实现水波动画效果。

（4）将内部面板类 WaterWaveActionPanel 的实例添加到窗体类 WaterWaveActionFrame 的内容面板上，用于在窗体上显示水波动画效果的图片。面板类 WaterWaveActionPanel 的代码如下：

```
class WaterWaveActionPanel extends JPanel implements Runnable {
    private Graphics graphics;                          //Graphics 对象
    private Graphics waveGraphics;                      //倒影的 Graphics 对象
    private Image image;                                //原 Image 对象
    private Image waveImage;                            //表示倒影的 Image 对象
    private int currentIndex;                           //当前图像索引
    private int imageWidth;                             //图像的宽度
    private int imageHeight;                            //图像的高度
    private boolean imageLoaded;                        //表示图片是否被加载的标记
```

（5）调用 paint()方法，用于在面板上绘制水波动画效果的图片，代码如下：

```
    public void paint(Graphics g) {
        if (waveImage != null) {
            g.drawImage(waveImage, -currentIndex * imageWidth, 0, this);    //绘制图像
        }
        g.clearRect(imageWidth, 0, imageWidth * 4, imageHeight);            //清除显示区域右侧的内容
    }
```

（6）使用多线程对复制的图像区域进行动态调整，以达到实现水波动画特效。代码如下：

```
    public void run() {
        currentIndex = 0;
        while (!imageLoaded) {                          //如果图片未加载
            repaint();                                  //重绘屏幕
            graphics = getGraphics();                   //获得 Graphics 对象
            MediaTracker mediatracker = new MediaTracker(this);     //创建媒体跟踪对象
            URL imgUrl = WaterWaveActionFrame.class
                    .getResource("/img/image.jpg");     //获取图片资源的路径
            image = Toolkit.getDefaultToolkit().getImage(imgUrl);   //获取图像资源
            mediatracker.addImage(image, 0);            //添加图片
            try {
                mediatracker.waitForAll();              //加载图片
                imageLoaded = !mediatracker.isErrorAny();   //是否有错误发生
            } catch (InterruptedException ex) {
            }
            if (!imageLoaded) {                         //加载图片失败
                graphics.drawString("加载图片错误", 10, 40);   //输出错误信息
                continue;
            }
            imageWidth = image.getWidth(this);          //得到图像宽度
            imageHeight = image.getHeight(this);        //得到图像高度
            createWave();                               //调用方法，实现动画效果
            break;
        }
        try {
            while (true) {
                repaint();                              //重绘屏幕
                currentIndex++;                         //调整当前图像索引
                if (currentIndex == 12) {               //如果当前图像索引为 12
                    currentIndex = 0;                   //设置当前图像索引为 0
                }
                Thread.sleep(50);                       //线程休眠
```

```
            }
        } catch (InterruptedException ex) {
        }
    }
```

（7）创建水波效果的图片，代码如下：

```
    public void createWave() {
        Image img = createImage(imageWidth, imageHeight);                    //以图像高度创建 Image 实例
        Graphics g = null;
        if (img != null) {
            g = img.getGraphics();                                           //得到 Image 对象的 Graphics 对象
            g.drawImage(image, 0, 0, this);                                  //绘制 Image
            for (int i = 0; i < imageHeight; i++) {
                g.copyArea(0, imageHeight - 1 - i, imageWidth, 1, 0,
                        -imageHeight + 1 + (i * 2));                         //复制图像区域
            }
        }
        waveImage = createImage(13 * imageWidth, imageHeight);               //得到波浪效果的 Image 实例
        if (waveImage != null) {
            waveGraphics = waveImage.getGraphics();                          //得到波浪效果的 Graphics 实例
            waveGraphics.drawImage(img, 12 * imageWidth, 0, this);           //绘制图像
            int j = 0;
            while (j < 12) {
                simulateWaves(waveGraphics, j);                              //调用方法
                j++;
            }
            g.drawImage(image, 0, 0, this);                                  //绘制图像
        }
    }
```

（8）模拟波浪效果，代码如下：

```
    public void simulateWaves(Graphics g, int i) {                           //波浪效果模拟
        double d = (6.0 * i) / 12;
        int j = (12 - i) * imageWidth;                                       //计算水平移动的距离
        int waveHeight = imageHeight / 16;                                   //用于计算水波的高度
        for (int m = 0; m < imageHeight; m++) {
            int k = (int) ((waveHeight * (m + 28) * Math.sin(waveHeight
                    * (imageHeight - m) / (m + 1) + d)) / imageHeight);      //用于控制要复制矩形区域的宽度
            if (m < -k)
                g.copyArea(12 * imageWidth, m, imageWidth, 1, -j, 0);        //复制图像区域，形成波浪
            else
                g.copyArea(12 * imageWidth, m + k, imageWidth, 1, -j, -k);   //复制图像区域，形成波浪
        }
    }
}
```

秘笈心法

心法领悟 083：关于 Java 的 repaint()方法。

本实例中使用的 repaint()方法是一个具有刷新页面效果的方法，即重绘 Component 组件。当 Component 组件中已有的图形发生变化后，并不会立刻在程序界面中显示，这时可以使用 repaint()方法重新绘制页面。

4.3　游 戏 开 发

实例 084　图片配对游戏　　高级

光盘位置：光盘\MR\084　　趣味指数：★★★★★

实例说明

本实例通过为标签控件添加图标以及鼠标事件，完成图片配对游戏。运行程序，初始显示效果如图 4.28 所

示，用鼠标拖动窗体上显示图标的标签到相应的文字标签上，如果配对正确，则下面的文字标签显示匹配成功的信息和图片，效果如图 4.29 所示。

图 4.28　图片配对之前的效果

图 4.29　图片配对成功的效果

关键技术

本实例通过玻璃面板、鼠标事件以及判断控件是否在其他控件的范围内，完成图片配对游戏的开发。

（1）玻璃面板位于 JRootPane 中所有其他组件之上，这为在所有其他组件上绘图和截取鼠标事件提供了方便，对拖动和绘图都非常有用。因此开发人员可在玻璃面板上使用 setVisible()方法控制玻璃面板在所有其他子级控件上是否显示，默认情况下，玻璃面板是不可见的，可以通过 JFrame 类的 getGlassPane()方法获得玻璃面板对象，该方法的定义如下：

```
public Component getGlassPane()
```

参数说明

返回值：此窗体的玻璃面板对象，该对象是 Component 类型的容器。

（2）通过使窗体类实现 MouseListener 和 MouseMotionListener 接口实现鼠标事件，完成鼠标按下、释放和拖动等事件的处理。

（3）本实例定义了一个 checkPosition()方法，用于检查所有拖动的控件是否匹配成功，以及是否在下面显示文字的控件内，该方法的代码如下：

```
private boolean checkPosition() {                          //检查配对是否正确
    boolean result = true;
    for (int i = 0; i < 3; i++) {
        Point location = img[i].getLocationOnScreen();     //获取每个图像标签的位置
        Point seat = targets[i].getLocationOnScreen();     //获取每个对应位置的坐标
        targets[i].setBackground(Color.GREEN);             //设置匹配后的颜色
        //如果配对错误
        if (location.x < seat.x || location.y < seat.y
                || location.x > seat.x + 80 || location.y > seat.y + 80) {
            targets[i].setBackground(Color.ORANGE);        //回复对应位置的颜色
            result = false;                                //检测结果为 false
        }
    }
    return result;                                         //返回检测结果
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PictureMatchingFrame 窗体类，该类实现了 MouseListener 和 MouseMotionListener 接口，因此可以响应鼠标事件。

（3）在 PictureMatchingFrame 窗体类的声明区声明如下成员：

```
private JLabel img[] = new JLabel[3];                    //显示图标的标签
private JLabel targets[] = new JLabel[3];                //窗体下面显示文字的标签
private Point pressPoint;                                //鼠标按下时的起始坐标
```

（4）在 PictureMatchingFrame 窗体类的构造方法中初始化界面，关键代码如下：

```
for (int i = 0; i < 3; i++) {
    img[i] = new JLabel(icon[i]);                        //创建图像标签
    img[i].setSize(50, 50);                              //设置标签大小
    img[i].setBorder(new LineBorder(Color.GRAY));        //设置线性边框
    int x = (int) (Math.random() * (getWidth() - 50));   //随机生成 x 坐标
    int y = (int) (Math.random() * (getHeight() - 150)); //随机生成 y 坐标
    img[i].setLocation(x, y);                            //设置随机坐标
    img[i].addMouseListener(this);                       //为每个图像标签添加鼠标事件监听器
    img[i].addMouseMotionListener(this);
    imagePanel.add(img[i]);                              //添加图像标签到图像面板
    targets[i] = new JLabel();                           //创建匹配位置标签
    targets[i].setOpaque(true);                          //使标签不透明，以设置背景色
    targets[i].setBackground(Color.ORANGE);              //设置标签背景色
    targets[i].setHorizontalTextPosition(SwingConstants.CENTER);  //设置文本与图像水平居中
    targets[i].setVerticalTextPosition(SwingConstants.BOTTOM);    //设置文本显示在图像下方
    targets[i].setPreferredSize(new Dimension(80, 80));  //设置标签首选大小
    targets[i].setHorizontalAlignment(SwingConstants.CENTER);     //文字居中对齐
    bottomPanel.add(targets[i]);                         //添加标签到底部面板
}
targets[0].setText("显示器");                            //设置匹配位置的文本
targets[1].setText("衣服");
targets[2].setText("自行车");
```

（5）鼠标按下事件用于获得拖放图片标签时的起始坐标，代码如下：

```
public void mousePressed(MouseEvent e) {
    pressPoint = e.getPoint();                           //保存拖放图片标签时的起始坐标
}
```

（6）鼠标释放事件用于检查图片配对是否正确，如果正确，能隐藏玻璃面板，并在下面的文字标签上显示图片和匹配成功的文字，代码如下：

```
public void mouseReleased(MouseEvent e) {
    if (checkPosition()) {                               //如果配对正确
        getGlassPane().setVisible(false);
        for (int i = 0; i < 3; i++) {                    //遍历所有匹配位置的标签
            targets[i].setText("匹配成功");              //设置正确提示
            targets[i].setIcon(img[i].getIcon());        //设置匹配的图标
        }
    }
}
```

（7）鼠标拖动事件用于设置控件新的位置，代码如下：

```
public void mouseDragged(MouseEvent e) {
    JLabel source = (JLabel) e.getSource();              //获取事件源控件
    Point imgPoint = source.getLocation();               //获取控件坐标
    Point point = e.getPoint();                          //获取鼠标坐标
    source.setLocation(imgPoint.x + point.x - pressPoint.x, imgPoint.y
            + point.y - pressPoint.y);                   //设置控件新坐标
}
```

注意：在使用本实例时，需要将项目需要的图片文件放到与窗体类 PictureMatchingFrame 相同的文件夹中，否则程序将发生异常。

秘笈心法

心法领悟 084：Java 中关于 this 关键字的解析。

this 代表当前对象名，如果方法的形参与类中的成员变量同名，可以使用 this 关键字指明引用的是成员变量，this 关键字除了可以调用成员变量之外，还可以调用构造方法。

实例 085 小猪走迷宫

光盘位置：光盘\MR\085　　高级　　趣味指数：★★★★★

实例说明

本实例实现了小猪走迷宫的游戏开发。运行程序，开始游戏，效果如图 4.30 所示，用户可以通过键盘上的方向键控制小猪的行走，如果在行走过程中撞到墙壁，会弹出消息框进行提示；如果顺利走出迷宫，则效果如图 4.31 所示。

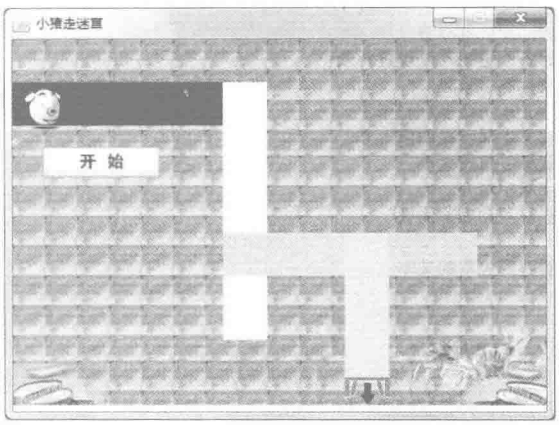


图 4.30　游戏开始时的效果

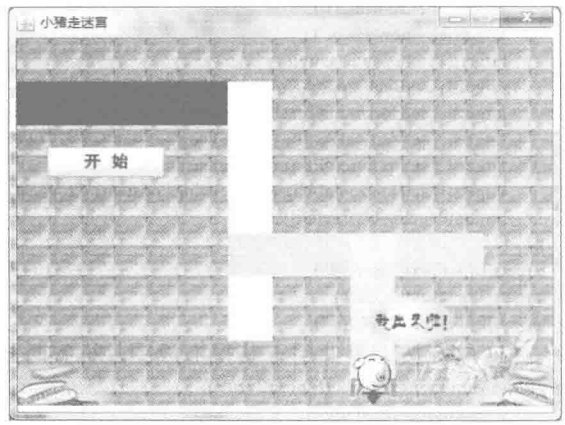


图 4.31　走出迷宫后的效果

关键技术

本实例通过键盘事件类 KeyEvent 的 getKeyCode()方法获得按键的整数代码，来控制小猪的移动，并使用 Rectangle 类的 intersects()方法判断两个 Rectangle 对象是否相交，实现对小猪是否撞墙的判断。

（1）KeyEvent 类的 getKeyCode()方法用于获得键盘上实际按键的整数代码，该方法的定义如下：

```
public int getKeyCode()
```

参数说明

返回值：键盘上实际按键的整数代码。

（2）使用 Rectangle 类的 intersects()方法判断两个 Rectangle 对象是否相交，从而实现判断小猪是否撞墙的操作，Rectangle 类的 intersects()方法定义如下：

```
public boolean intersects(Rectangle r)
```

参数说明

❶ r：指定的 Rectangle 对象。

❷ 返回值：如果指定的 Rectangle 对象与当前 Rectangle 对象相交，则返回 true；否则返回 false。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PigWalkMazeFrame 窗体类，并实现 KeyListener 和 Runnable 接口，目的就是通过键盘控制小猪的移动，并在小猪走出迷宫时，通过新的线程改变小猪走出迷宫的图片。

（3）在窗体类 PigWalkMazeFrame 中通过键盘的按键事件控制小猪的移动，键盘按键事件的代码如下：

```
public void keyPressed(KeyEvent e) {
    if ((gobuttonY == 286)) {                                    //如果小猪的纵坐标等于 286
        Thread thread = new Thread(this);
```

```
            thread.start();                                          //启动线程
        }
        if (e.getKeyCode() == KeyEvent.VK_UP) {                      //如果用户按下了向上键
            Rectangle rectAngle = new Rectangle(gobuttonX, gobuttonY, 20, 20);   //创建 Rectangle 对象
            if (rectAngle.intersects(rect1)
                    || rectAngle.intersects(rect2)
                    || rectAngle.intersects(rect3)
                    || rectAngle.intersects(rect4)) {                //判断小猪是否走出了迷宫
                gobuttonY = gobuttonY - 2;                           //设置变量坐标
                goButton.setLocation(gobuttonX, gobuttonY);          //设置按钮坐标
            } else {                                                 //如果小猪走出了迷宫
                JOptionPane.showMessageDialog(this, "撞墙了吧！ 重新开始吧！ ", "撞墙啦！ ",
                        JOptionPane.DEFAULT_OPTION);
            }
                        } else if (e.getKeyCode() == KeyEvent.VK_DOWN) {     //判断用户是否按向下键
            Rectangle rectAngle = new Rectangle(gobuttonX, gobuttonY, 20, 20);
            if (rectAngle.intersects(rect1)
                    || rectAngle.intersects(rect2)
                    || rectAngle.intersects(rect3)
                    || rectAngle.intersects(rect4)) {
                gobuttonY = gobuttonY + 2;
                goButton.setLocation(gobuttonX, gobuttonY);
            } else {
                JOptionPane.showMessageDialog(this, "撞墙了吧！ 重新开始吧！ ", "撞墙啦！ ",
                        JOptionPane.DEFAULT_OPTION);
            }
        } else if (e.getKeyCode() == KeyEvent.VK_LEFT) {             //如果用户按向左键
            Rectangle rectAngle = new Rectangle(gobuttonX, gobuttonY, 20, 20);
            if (rectAngle.intersects(rect1)
                    || rectAngle.intersects(rect2)
                    || rectAngle.intersects(rect3)
                    || rectAngle.intersects(rect4)) {
                gobuttonX = gobuttonX - 2;
                goButton.setLocation(gobuttonX, gobuttonY);
            } else {
                JOptionPane.showMessageDialog(this, "撞墙了吧！ 重新开始吧！ ", "撞墙啦！ ",
                        JOptionPane.DEFAULT_OPTION);
            }
        } else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {            //如果用户按向右键
            Rectangle rectAngle = new Rectangle(gobuttonX, gobuttonY, 20, 20);
            if (rectAngle.intersects(rect1)
                    || rectAngle.intersects(rect2)
                    || rectAngle.intersects(rect3)
                    || rectAngle.intersects(rect4)) {
                gobuttonX = gobuttonX + 2;
                goButton.setLocation(gobuttonX, gobuttonY);
            } else {
                JOptionPane.showMessageDialog(this, "撞墙了吧！ 重新开始吧！ ", "撞墙啦！ ",
                        JOptionPane.DEFAULT_OPTION);
            }
        }
    }
```

（4）当小猪走出迷宫后，可以通过“开始”按钮重新开始游戏。“开始”按钮的事件代码如下：

```
public void buttonAction(ActionEvent e) {
    goButton.setIcon(imageIcon);                                     //重新设置按钮的显示图片
    goButton.addKeyListener(this);                                   //为按钮添加键盘事件
    goButton.setBounds(0, 40, imageIcon.getIconWidth(), imageIcon
            .getIconHeight());                                       //设置小猪位置
    gobuttonX = goButton.getBounds().x;                              //获得小猪当前位置的 x 坐标
    gobuttonY = goButton.getBounds().y;                              //获得小猪当前位置的 y 坐标
    goButton.requestFocus();                                         //设置按钮获取焦点
}
```

（5）当小猪走出迷宫后，通过线程来改变小猪走出迷宫的图标。run()方法实现该功能的代码如下：

```
public void run() {
    URL out = getClass().getResource("/images/pigOut.png");          //获取图片 URL
    ImageIcon imageout = new ImageIcon(out);                         //创建图片对象
```

```
        goButton.setIcon(imageout);                                    //设置小猪按钮显示图片
        goButton.setBounds(gobuttonX,
                gobuttonY - imageout.getIconHeight() + 50, imageout
                        .getIconWidth(), imageout.getIconHeight());    //重新设置按钮位置
        goButton.removeKeyListener(this);                              //按钮移除键盘事件
    }
```

秘笈心法

心法领悟 085：键盘事件（KeyEvent）。

在 KeyEvent 类中，以 VK_开头的静态常量代表各个按键的 keycode，可以通过这些静态常量判断事件中的按键，从而可以获得所按键的字符。

实例 086 拼图游戏

光盘位置：光盘\MR\086　　高级　　趣味指数：★★★★★

实例说明

本实例实现了拼图游戏的开发。运行程序，单击“开始”按钮将打乱图片的位置，效果如图 4.32 所示，然后通过鼠标单击图片进行移动，直到将所有图片都移动到正确位置，游戏过关，过关后的效果如图 4.33 所示。

图 4.32　打乱图片位置的效果

图 4.33　图片移动到正确位置的效果

关键技术

本程序主要通过 Swing 与枚举类实现，程序将一幅完整的图片平均分成 9 部分，每一部分为一个正方形，并将最后一个图片修改为空白图片，作为游戏中的一个空位置。对于每一个图片部分，程序封装了一个按钮对象进行装载，当该按钮对象被单击后，程序将调换该按钮与装载空白图片的按钮，其关键技术是使用枚举类控制方向，以及使用 setLocation()方法设置按钮的位置。

（1）本实例通过枚举类定义了图片移动的 4 个方向，分别为上、下、左、右，其定义方式与定义一个类相似，但定义枚举类使用关键字 enmu，枚举类 Direction 的定义如下：

```
public enum Direction {
    UP,                                                            //上
    DOWN,                                                          //下
    LEFT,                                                          //左
    RIGHT                                                          //右
}
```

（2）当图片移动后，按钮的坐标发生改变，此操作通过 setLocation()方法实现。setLocation()方法是从 Component 类继承的，其定义如下：

```
public void setLocation(int x, int y)
```

参数说明

❶ x：当前控件左上角在父级坐标空间中新位置的 x 坐标。

❷ y：当前控件左上角在父级坐标空间中新位置的 y 坐标。

设计过程

（1）新建一个项目。

（2）在项目中创建一个名称为 Direction 的枚举类，用于定义图片移动的 4 个方向，其关键代码见关键技术部分。

（3）创建名称为 Cell 的类，用于封装一个单元图片对象，此类继承 JButton 对象，并对 JButton 按钮组件进行重写，其关键代码如下：

```
public class Cell extends JButton {
    private static final long serialVersionUID = 718623114650657819L;
    public static final int IMAGEWIDTH = 117;                    //图片宽度
    private int place;                                           //图片位置
    public Cell(Icon icon, int place) {
        this.setSize(IMAGEWIDTH, IMAGEWIDTH);                    //单元图片的大小
        this.setIcon(icon);                                      //单元图片的图标
        this.place = place;                                      //单元图片的位置
    }
    public void move(Direction dir) {                            //移动单元图片的方法
        Rectangle rec = this.getBounds();                        //获取图片的 Rectangle 对象
        switch (dir) {                                           //判断方向
            case UP:                                             //向上移动
                this.setLocation(rec.x, rec.y - IMAGEWIDTH);
                break;
            case DOWN:                                           //向下移动
                this.setLocation(rec.x, rec.y + IMAGEWIDTH);
                break;
            case LEFT:                                           //向左移动
                this.setLocation(rec.x - IMAGEWIDTH, rec.y);
                break;
            case RIGHT:                                          //向右移动
                this.setLocation(rec.x + IMAGEWIDTH, rec.y);
                break;
        }
    }
    public int getX() {
        return this.getBounds().x;                               //获取单元图片的 x 坐标
    }
    public int getY() {
        return this.getBounds().y;                               //获取单元图片的 y 坐标
    }
    public int getPlace() {
        return place;                                            //获取单元图片的位置
    }
}
```

（4）创建名称为 GamePanel 的类，此类继承 Jpanel 类、实现 MouseListener 接口，用于创建游戏面板对象。在 GamePanel 类中定义长度为 9 单元的图片数组对象，并通过 init()方法对所有单元图片对象进行实例化，其关键代码如下：

```
public class GamePanel extends JPanel implements MouseListener {
    private static final long serialVersionUID = -6538319477834401 22L;
    private Cell[] cells = new Cell[9];                          //创建单元图片数组
    private Cell cellBlank = null;                               //空白
    public GamePanel() {
        super();
        setLayout(null);                                         //设置空布局
```

```
        init();                                                    //初始化
    }
    //初始化游戏
    public void init() {
        int num = 0;                                               //图片序号
        Icon icon = null;                                          //图标对象
        Cell cell = null;                                          //单元图片对象
        for (int i = 0; i < 3; i++) {                              //循环行
            for (int j = 0; j < 3; j++) {                          //循环列
                num = i * 3 + j;                                   //计算图片序号
                icon = SwingResourceManager.getIcon(GamePanel.class, "/pic/"
                        + (num + 1) + ".jpg");                     //获取图片
                cell = new Cell(icon, num);                        //实例化单元图片对象
                cell.setLocation(j * Cell.IMAGEWIDTH, i * Cell.IMAGEWIDTH);  //设置单元图片的坐标
                cells[num] = cell;                                 //将单元图片存储到单元图片数组中
            }
        }
        for (int i = 0; i < cells.length; i++) {
            this.add(cells[i]);                                    //向面板中添加所有单元图片
        }
    }
    /**
     * 对图片进行随机排序
     */
    public void random() {
        Random rand = new Random();                                //实例化 Random
        int m, n, x, y;
        if (cellBlank == null) {                                   //判断空白的图片位置是否为空
            cellBlank = cells[cells.length - 1];                   //取出空白的图片
            for (int i = 0; i < cells.length; i++) {               //遍历所有单元图片
                if (i != cells.length - 1) {
                    cells[i].addMouseListener(this);               //对非空白图片注册鼠标监听
                }
            }
        }
        for (int i = 0; i < cells.length; i++) {                   //遍历所有单元图片
            m = rand.nextInt(cells.length);                        //产生随机数
            n = rand.nextInt(cells.length);                        //产生随机数
            x = cells[m].getX();                                   //获取 x 坐标
            y = cells[m].getY();                                   //获取 y 坐标
            //对单元图片调换
            cells[m].setLocation(cells[n].getX(), cells[n].getY());
            cells[n].setLocation(x, y);
        }
    }
    @Override
    public void mousePressed(MouseEvent e) {
        Cell cell = (Cell) e.getSource();                          //获取触发时间的对象
        int x = cellBlank.getX();                                  //获取 x 坐标
        int y = cellBlank.getY();                                  //获取 y 坐标
        if ((x - cell.getX()) == Cell.IMAGEWIDTH && cell.getY() == y) {
            cell.move(Direction.RIGHT);                            //向右移动
            cellBlank.move(Direction.LEFT);
        } else if ((x - cell.getX()) == -Cell.IMAGEWIDTH && cell.getY() == y) {
            cell.move(Direction.LEFT);                             //向左移动
            cellBlank.move(Direction.RIGHT);
        } else if (cell.getX() == x && (cell.getY() - y) == Cell.IMAGEWIDTH) {
            cell.move(Direction.UP);                               //向上移动
            cellBlank.move(Direction.DOWN);
        } else if (cell.getX() == x && (cell.getY() - y) == -Cell.IMAGEWIDTH) {
            cell.move(Direction.DOWN);                             //向下移动
            cellBlank.move(Direction.UP);
        }
        if (isSuccess()) {                                         //判断是否拼图成功
            int i = JOptionPane.showConfirmDialog(this, "成功，再来一局？", "拼图成功",
                    JOptionPane.YES_NO_OPTION);                    //提示成功
            if (i == JOptionPane.YES_OPTION) {
                random();                                          //开始新一局
            }
        }
    }
```

```
    /**
     * 判断是否拼图成功
     * @return 布尔值
     */
    public boolean isSuccess() {
        for (int i = 0; i < cells.length; i++) {                          //遍历所有单元图片
            int x = cells[i].getX();                                      //获取 x 坐标
            int y = cells[i].getY();                                      //获取 y 坐标
            if (i != 0) {
                if (y / Cell.IMAGEWIDTH * 3 + x / Cell.IMAGEWIDTH != cells[i]
                        .getPlace()) {                                    //判断单元图片位置是否正确
                    return false;                                         //只要有一个单元图片的位置不正确，就返回 false
                }
            }
        }
        return true;                                                      //所有单元图片的位置都正确返回 true
    }
}
```

秘笈心法

心法领悟 086：枚举类型的用途。

在 Java 中，可以使用 enum 关键字创建枚举类型。枚举类型可以对数据进行检查，例如在使用 switch 语句时，可以让该语句只接受枚举类型的值，这样就可以使用枚举类型检查是否为合法的数据，如果只是简单地传递整数就会出错。

实例 087　海滩捉螃蟹　　高级

光盘位置：光盘\MR\087　　趣味指数：★★★★★

实例说明

本实例使用线程和鼠标事件监听器开发一个捉螃蟹的程序。程序启动后，会有一个线程控制螃蟹随机出现在沙滩的某个小洞里，用鼠标单击某个螃蟹，表示它被抓，螃蟹会“流泪”，而鼠标操作也会变成拾取物品的动作。运行程序，效果如图 4.34 所示。

图 4.34　海滩捉螃蟹

关键技术

本实例使用随机数随机确定螃蟹出现的顺序，并使用线程进行循环控制，使螃蟹不停地出现。

（1）开发中唯一的一个难点就是如何控制螃蟹的显示。考虑到游戏对用户的吸引程度，螃蟹的显示应该是随机确定位置，而不能以固定的顺序出现，这就需要使用随机数。本程序采用的是 Math 类的 random()方法，关键代码如下：

```
int index = (int) (Math.random() * 6);                    //生成随机的螃蟹索引
```

（2）螃蟹的显示需要不停地循环，在循环中检测螃蟹是否显示，并为空位置添加螃蟹图片，run()方法的代码如下：

```
public void run() {
    while (true) {                                        //使用无限循环
        try {
            Thread.sleep(1000);                           //使线程休眠 1 秒
            int index = (int) (Math.random() * 6);        //生成随机的螃蟹索引
            if (carb[index].getIcon() == null) {          //如果螃蟹标签没有设置图片
                carb[index].setIcon(imgCarb);             //为该标签添加螃蟹图片
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 CaptureCarbFrame 窗体类，同时该窗体类实现了 Runnable 接口，因此可以通过该类创建线程对象，其中 run()方法的代码见关键技术部分。

（3）编写内部类 MouseCrab，该类实现了 MouseListener 接口，是窗体控件的鼠标事件监听器，用于在鼠标按键的按下和释放，以及鼠标离开控件区域动作发生时替换鼠标的指针图片，这样会使程序界面更加形象，因为替换的两个图片能够形成抓取物品的连贯动作，分别作为窗体和标签控件的事件监听器，MouseCrab 类的代码如下：

```
private final class MouseCrab implements MouseListener {
    private final Cursor cursor1;                         //鼠标图标 1
    private final Cursor cursor2;                         //鼠标图标 2
    private MouseCrab(Cursor cursor1, Cursor cursor2) {
        this.cursor1 = cursor1;
        this.cursor2 = cursor2;
    }
    @Override
    public void mouseReleased(MouseEvent e) {
        setCursor(cursor1);                               //鼠标按键释放时设置指针为 cursor1
    }
    @Override
    public void mousePressed(MouseEvent e) {
        setCursor(cursor2);                               //鼠标按键按下时设置指针为 cursor2
    }
    @Override
    public void mouseExited(MouseEvent e) {
        setCursor(cursor1);                               //鼠标离开控件区域时设置指针为 cursor1
    }
    @Override
    public void mouseEntered(MouseEvent e) {
    }
    @Override
    public void mouseClicked(MouseEvent e) {
    }
}
```

（4）本实例还创建了一个鼠标事件监听器类 Catcher，它对鼠标按键进行判断之后才设置控件图片，但是

设置的不是鼠标指针图片，而是显示螃蟹的标签控件的图片。在单击时，换成螃蟹流泪的图片，而鼠标释放和初始状态都是一个微笑的螃蟹图片。该事件监听器主要作为显示螃蟹的标签控件的事件监听器，代码如下：

```
private final class Catcher extends MouseAdapter {
    @Override
    public void mousePressed(MouseEvent e) {
        if (e.getButton() != MouseEvent.BUTTON1)
            return;
        Object source = e.getSource();                    //获取事件源，即螃蟹标签
        if (source instanceof JLabel) {                   //如果事件源是标签组件
            JLabel carb = (JLabel) source;                //强制转换为 JLabel 标签
            if (carb.getIcon() != null)
                carb.setIcon(imgCarb2);                   //为该标签添加螃蟹图片
        }
    }
    @Override
    public void mouseReleased(MouseEvent e) {
        if (e.getButton() != MouseEvent.BUTTON1)
            return;
        Object source = e.getSource();                    //获取事件源，即螃蟹标签
        if (source instanceof JLabel) {                   //如果事件源是标签组件
            JLabel carb = (JLabel) source;                //强制转换为 JLabel 标签
            carb.setIcon(null);                           //清除标签中的螃蟹图片
        }
    }
}
```

秘笈心法

心法领悟 087：强制类型转型。

关于强制转型，如本实例中的 **int** index =(**int**)(Math.random() * 6)，之所以这样做，是因为 Java 中不能将高精度的类型值赋值给低精度类型的变量，但是使用强制类型转换时会有精度的损失，所以使用时一定要注意。

实例 088　荒山打猎游戏

高级

光盘位置：光盘\MR\088

趣味指数：★★★★★

实例说明

本实例程序界面上，底部会有野猪随机出现并以不固定的速度移动，上方有小鸟以反方向飞过，当用鼠标在它们身上进行单击操作时，会打中该动物，动物消失，在界面左上角得到相应分数，但是如果动物跑出界面，游戏就会扣除一定的分数。另外，界面右上角会显示当前剩余子弹数量，如无子弹，需要等待系统装载子弹。运行程序，效果如图 4.35 所示。

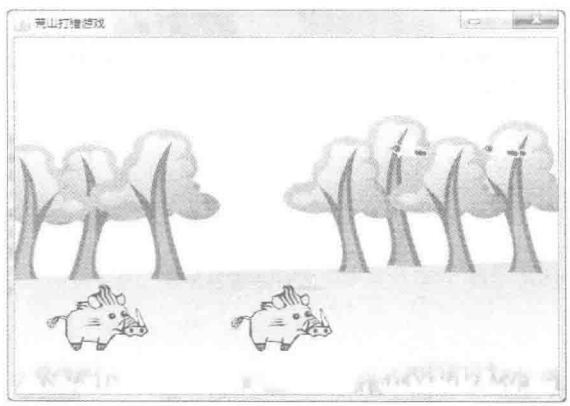


图 4.35　荒山打猎游戏

关键技术

本程序的难点在于控制动物控件的移动速度。如果每个动物移动的速度相同，就会使程序运行效果枯燥乏味，没有游戏难度也就没有进行下去的意义，所以要在线程中控制每个控件的移动速度。在线程循环中，可以通过随机数来确定新创建的动物控件移动线程的休眠时间，这样就可以为每个动物控件设置不同的移动速度，如在 MainFrame 窗体类的内部线程类 PigThread 中，调用了 Math 类的 random()方法随机确定线程休眠的时间。

窗体类 MainFrame 的内部线程类 PigThread 的代码如下：

```
class PigThread extends Thread {
    @Override
    public void run() {
        while (true) {
            PigLabel pig = new PigLabel();                    //创建代表野猪的标签控件
            pig.setSize(120, 80);                             //设置控件初始大小
            backgroundPanel.add(pig);                         //添加控件到背景面板
            try {
                sleep((long) (random() * 3000) + 500);        //线程随机休眠一段时间
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

说明：上段代码的 random()方法是 Math 类的静态方法，所以在 MainFrame 类的包引用位置使用了静态导入语句 import static java.lang.Math.random;，将该 Math 类的 random()方法作为本类的一部分，因此在上述代码中省略了 Math 类名称而直接调用了 random()方法。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JLabel 类的 BirdLabel 标签类，用于表示小鸟，并且实现 Runnable 接口。通过线程控制小鸟的移动效果，以及实现扣分功能。BirdLabel 标签类中 run()方法的代码如下：

```
public void run() {
    parent = null;
    int width = 0;
    try {
        while (width <= 0 || parent == null) {
            if (parent == null){
                parent = getParent();                         //获取父容器
            } else {
                width = parent.getWidth();                    //获取父容器的宽度
            }
            Thread.sleep(10);
        }
        for (int i = width; i > 0 && parent != null; i -= 8) {
            setLocation(i, y);                                //从右向左移动本组件位置
            Thread.sleep(sleepTime);                          //休眠片刻
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    if (parent != null) {
        MainFrame.appScore(-score * 10);                      //自然销毁将扣分
    }
    destory();                                                //移动完毕，销毁本组件
}
```

（3）在项目中创建一个继承 JLabel 类的 PigLabel 标签类，用于表示野猪，并且实现 Runnable 接口。通过线程控制野猪的移动效果，以及实现扣分功能。PigLabel 标签类中 run()方法的代码如下：

```
public void run() {
    parent = null;
    int width = 0;
    while (width <= 0 || parent == null) {                    //获取父容器宽度
        if (parent == null)
            parent = getParent();                             //获取父容器
        else
            width = parent.getWidth();                        //获取父容器的宽度
    }
    for (int i = 0; i < width && parent != null; i += 8) {
        setLocation(i, y);                                    //从左向右移动本组件
        try {
            Thread.sleep(sleepTime);                          //休眠片刻
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    if (parent != null) {
        MainFrame.appScore(-score * 10);                      //自然销毁将扣分
    }
    destory();                                                //移动完毕，销毁本组件
}}
```

（4）在项目中创建一个继承 JFrame 类的主窗体类 MainFrame，在该类中分别创建生成小鸟和小猪角色的内部线程类，其中生成小鸟角色的类代码如下：

```
class BirdThread extends Thread {
    @Override
    public void run() {
        while (true) {
            BirdLabel bird = new BirdLabel();                 //创建代表小鸟的标签控件
            bird.setSize(50, 50);                             //设置控件初始大小
            backgroundPanel.add(bird);                        //添加控件到背景面板
            try {
                sleep((long) (Math.random() * 3000) + 500);   //线程随机休眠一段时间
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

说明：由于篇幅原因，这里没有将所有代码进行一一讲解，如果需要对本实例有更进一步的了解，可以参考源程序代码。

秘笈心法

心法领悟 088：Java 中的父类和子类。

父类和子类是相互独立的，子类可以继承父类已有的属性和行为，从而可以提高代码的重用。在 Java 语言中，将两个具有父子关系类中的“父亲”称为超类或父类，将“孩子”称为子类。一个类只能有一个直接父类，但是可以有多个间接父类，这种父子关系在 Java 中被称为继承。

实例 089　打字母游戏　　高级

光盘位置：光盘\MR\089　　趣味指数：★★★★★

实例说明

本实例开发了一个练习指法的打字母游戏。程序运行后，不断从上向下“掉”苹果，苹果上标有字母，按

键盘上对应的字母键，若正确，则该苹果消失；若错误，可以在苹果没有落地之前继续按键，如果正确将得分，如果苹果已经落地，则该字母没有打中，不得分，苹果落地后还会出现新的苹果。运行程序，效果如图 4.36 所示。

图 4.36　打字母游戏

关键技术

随机获得大写字母的 ASCII 值，然后把 ASCII 值转换为大写字母字符，再将大写字母字符转换为字符串，这些操作可以通过如下代码来实现：

```
int letter=(int)(Math.random()*26)+65;          //产生 65~90 之间的随机整数，即大写字母 A~Z 的 ASCII 值
char c=(char)letter;                            //将字符的 ASCII 值转换为字符
String s=String.valueOf(c);                     //将字符转换为字符串
```

说明：这段代码中的 letter 是 int 型的整数，用于存储 65~90 之间的随机整数；c 是 char 型的字符，用于存储转换的大写字母；s 是 String 类型的变量，用于存储单个大写字母字符组成的字符串。

设计过程

（1）新建一个项目。

（2）在项目中创建一个类，名称为 RandomBuildLetter，调用该类的方法可以随机产生 65~90 之间的随机整数，用于以后转换成大写字母，该类的代码如下：

```
/**
 * @author 张振坤
 * 产生 65~90 之间随机整数的类
 */
public class RandomBuildLetter {
    public RandomBuildLetter() {
        super();
    }

    public int[] getLetter(int letterCounts) {
        int[] letter = new int[letterCounts];                   //根据参数创建整型数组
        for (int i = 0; i < letterCounts; i++) {
            int a = (int) getRandomLetter();                    //调用方法产生 65~90 之间的随机整数，即大写字母 A~Z 的 ASCII 值
            letter[i] = a;                                      //将产生的数赋值给数组
        }
        return letter;                                          //返回数组对象
    }
```

```
    public int getRandomLetter() {
        int letter = (int) (Math.random() * 26) + 65;                    //产生 65~90 之间的随机整数，即大写字母的 ASCII 值
        return letter;
    }
}
```

（3）在项目中创建一个继承 JFrame 类的主窗体类 TypeLetterFrame，并在成员声明区定义标签数组，用于显示随机产生的带字母的苹果，定义 RandomBuildLetter 类的实例用于获得随机产生的大写字母的 ASCII 值，定义一个向量对象用于存储准备击打的字母，关键代码如下：

```
private RandomBuildLetter buildLetter = new RandomBuildLetter();  //创建随机产生字母的类的对象
private JLabel[] labels = null;                                   //创建标签数组
private Vector<String> vector = new Vector<String>();             //创建存储准备击打字母的向量
```

（4）在窗体 TypeLetterFrame 中创建 addLetter()方法，用于实现在窗体上显示字母、计算出现的字母总数等操作，该方法的代码如下：

```
private void addLetter() {
    int seed = 10;                                                //设置标签之间偏移量的变量
    //调用 RandomBuildLetter 类的方法随机产生 8 个整数并赋值给数组，即 8 个 A~Z 之间字母的 ASCII 值
    int[] letters = buildLetter.getLetter(8);
    labels = new JLabel[letters.length];                          //创建显示带字母苹果的标签数组
    //实例化标签数组的每个对象
    for (int i = 0; i < letters.length; i++) {
        int value = letters[i];                                   //获得数组 letters 中的值
        char c = (char) value;                                    //将数组 letters 中的值转换为字符
        String s = String.valueOf(c);                             //将字符转换为字符串
        labels[i] = new JLabel();                                 //实例化标签对象
        labels[i].setToolTipText(s);                              //设置标签的提示文本
        labels[i].setIcon(SwingResourceManager.getIcon(
                TypeLetterFrame.class, "/icon/" + s + ".png"));   //设置标签显示的图片，即带字母的苹果
        int x = (int) (Math.random() * 60) + seed;                //随机产生标签显示位置的横坐标
        int y = (int) (Math.random() * 80);                       //随机产生标签显示位置的纵坐标
        labels[i].setBounds(x, y, 100, 30);                       //设置标签的显示位置和大小
        backgroundPanel.add(labels[i]);                           //将标签添加到背景面板中
        seed += 60;                                               //调整标签之间的偏移量
        vector.add(s);                                            //将字符串字母添加到向量对象中
        totalLetters++;                                           //计算出现字母的总个数
    }
}
```

说明：由于篇幅原因，这里没有对所有代码进行一一讲解，如果需要对本实例有更进一步的了解，掌握打字母游戏的完整开发过程，可以参考本实例的源程序代码。

秘笈心法

心法领悟 089：使用 private 关键字修饰类。

private 定义的类是私有的，可以使程序更安全。但是在程序中一般不使用 private 修饰类，private 通常用于修饰内部类，可以被其所在的外部类使用，并可以通过类的实例访问内部类的成员。

实例 090　警察抓小偷

高级

光盘位置：光盘\MR\090

趣味指数：★★★★★

实例说明

本实例开发了一个警察抓小偷游戏。程序运行后，小偷在窗体中左侧位置的一定范围内随机跑动，警察在窗体的中上部位置，当用鼠标单击击中小偷时，则表示小偷被打中，游戏过关，单击“再来一次”按钮，又可以开始新的游戏。运行程序，效果如图 4.37 所示。

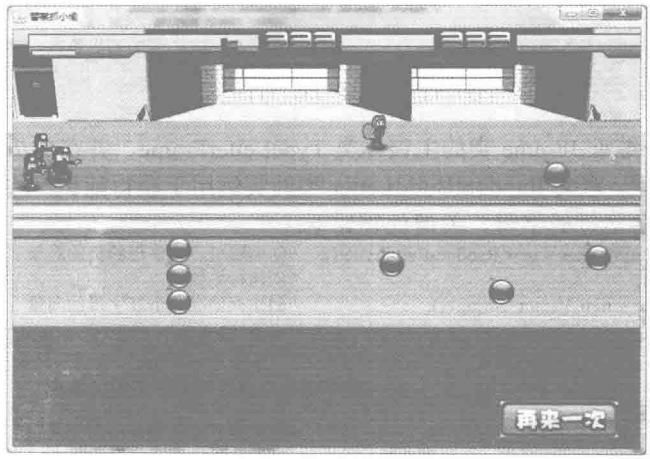


图 4.37　警察抓小偷

关键技术

在使用线程控制小偷标签的移动位置时，通过线程类的成员变量 x 来控制小偷左右运动；通过随机数控制小偷上下运动。改变小偷标签移动位置可以通过如下代码实现：

```
if (flag == false) {                                    //flag 为 false
    x += 20;                                            //x 的值增加表示向右运动
    if (x == 640) {                                     //当小偷标签左侧边界的横坐标是 640 时
        flag = true;                                    //将 flag 赋值为 true
    }
} else {                                                //flag 为 true
    x -= 20;                                            //x 的值减少表示向左运动
    if (x == 100) {                                     //当小偷标签左侧边界的横坐标是 100 时
        flag = false;                                   //将 flag 赋值为 false
    }
}
//生成 100~200 之间的随机整数，用于设置小偷标签上边界的纵坐标
int y = (int) (Math.random() * 100) + 100;
lb_thief.setLocation(x, y);                             //设置小偷标签的显示位置
```

说明：这段代码中的 x 是 int 型的整数，是线程的成员变量，用于指定小偷左右移动；y 是 int 型的整数，是线程中的局部变量，用于指定小偷上下移动。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PolicemanGraspThief 窗体类，在窗体 PolicemanGraspThief 中创建内部线程类 GraspThiefThread，用于实现动画效果，该内部线程类的代码如下：

```
private class GraspThiefThread implements Runnable {
    boolean flag = false;                               //标识小偷向左运动还是向右运动的变量
    int x = 400;                                        //小偷标签左侧边界的横坐标

    public void run() {
        while (true) {
            if (stop) {                                 //stop 为 true 时，显示提示文本为“打中了”标签
                int x = lb_thief.getX();                //获得小偷标签的横坐标
                int y = lb_thief.getY();                //获得小偷标签的纵坐标
                lb_tip.setBounds(x, y + 60, 50, 50);    //设置提示文本为“打中了”标签的显示位置和大小
                lb_tip.setVisible(true);                //显示提示文本为“打中了”的标签
                thread = null;                          //释放线程资源
                break;                                  //退出循环，结束线程的执行
            }
            if (flag == false) {                        //flag 为 false 向右运动
                x += 20;                                //x 的值增加表示向右运动
```

```
                if (x == 640) {                          //当小偷标签左侧边界的横坐标是 640 时
                    flag = true;                         //将 flag 赋值为 true
                }
            } else {                                     //flag 为 true 向左运动
                x -= 20;                                 //x 的值减少表示向左运动
                if (x == 100) {                          //当小偷标签左侧边界的横坐标是 100 时
                    flag = false;                        //将 flag 赋值为 false
                }
            }
            //生成 100～200 之间的随机整数，用于设置小偷标签上边界的纵坐标
            int y = (int) (Math.random() * 100) + 100;
            lb_thief.setLocation(x, y);                  //设置小偷标签的显示位置
            try {
                Thread.sleep(200);                       //休眠 200 毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

说明：由于篇幅原因，这里没有对所有代码进行一一讲解，如果需要对本实例有更进一步的了解，掌握警察抓小偷游戏的开发过程，可以参考本实例的源程序代码。

秘笈心法

心法领悟 090：Java 中的抽象类。

在 Java 语言中，如果不希望一个类被实例化，可以使用 abstract 关键字声明一个抽象类。抽象类中可以包含实例方法，也可以包含抽象方法，抽象类只能被其子类实例化，并且子类必须实现抽象类中的所有抽象方法。

实例 091　掷骰子

光盘位置：光盘\MR\091　　高级　　趣味指数：★★★★★

实例说明

本实例开发了一个单机掷骰子游戏，玩家可以模拟下注 10 元、20 元、50 元或 100 元，而且单击钱数按钮后，可以通过单击“加倍”按钮进行加倍下注，然后还需要单击“确认下注”按钮，才算完成了玩家的下注操作，同时在庄家处显示“跟了”的信息，这时玩家可以选择压大还是压小，然后骰子开始动画变换，动画结束后将显示消息框进行提示。运行程序，效果如图 4.38 所示。

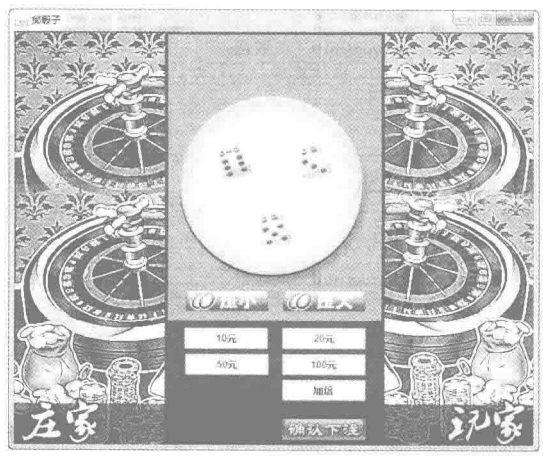


图 4.38　掷骰子游戏

关键技术

使用线程控制骰子的切换，并将 6 个骰子图片文件的主文件名用阿拉伯数字命名（例如 1.png、2.png、3.png、4.png、5.png 和 6.png），这样命名之后，可以在程序中方便地通过数字来改变图片文件，以实现切换骰子的操作，在切换骰子时使用随机数产生骰子的主文件名编号，改变骰子的图片可以通过如下代码实现：

```
v1=(int)(Math.random()*6+1);
v2=(int)(Math.random()*6+1);
v3=(int)(Math.random()*6+1);
lb_dice_1.setIcon(SwingResourceManager.getIcon(DiceGameFrame.class, "/icon/"+v1+".png"));
lb_dice_2.setIcon(SwingResourceManager.getIcon(DiceGameFrame.class, "/icon/"+v2+".png"));
lb_dice_3.setIcon(SwingResourceManager.getIcon(DiceGameFrame.class, "/icon/"+v3+".png"));
```

说明：这段代码中的 v1 是 int 型的整数成员变量，用于存储骰子图片的主文件名编号，同时也是骰子的点数；v2 是 int 型的整数成员变量，用于存储骰子图片的主文件名编号，同时也是骰子的点数；v3 是 int 型的整数成员变量，用于存储骰子图片的主文件名编号，同时也是骰子的点数。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DiceGameFrame 窗体类，在窗体 DiceGameFrame 中创建内部线程类 DiceThread，用于判断骰子的点数并确定玩家赢还是庄家赢，该内部线程类的代码如下：

```
private class DiceThread implements Runnable {
    public void run() {
        while (true) {
            stopIndex++;
            v1 = (int) (Math.random() * 6 + 1);              //随机产生第 1 个骰子的点数
            v2 = (int) (Math.random() * 6 + 1);              //随机产生第 2 个骰子的点数
            v3 = (int) (Math.random() * 6 + 1);              //随机产生第 3 个骰子的点数
            //显示骰子的图片
            lb_dice_1.setIcon(SwingResourceManager.getIcon(
                    DiceGameFrame.class, "/icon/" + v1 + ".png"));
            lb_dice_2.setIcon(SwingResourceManager.getIcon(
                    DiceGameFrame.class, "/icon/" + v2 + ".png"));
            lb_dice_3.setIcon(SwingResourceManager.getIcon(
                    DiceGameFrame.class, "/icon/" + v3 + ".png"));
            int totalValues = v1 + v2 + v3;                  //骰子的点数总和
            if (stopIndex == 50) {                           //当 stopIndex 为 50 时，显示消息框，并提示最终的点数
                if (flag == true) {
                    if (totalValues > 9) {                   //骰子的点数为大时显示的提示信息
                        JOptionPane.showMessageDialog(null, "点数是："
                                + totalValues + "，大。\n 玩家赢！！！\n 总钱数：人民币"
                                + totalMoney + "元");
                    } else {
                        JOptionPane.showMessageDialog(null, "点数是："
                                + totalValues + "，小。\n 庄家赢！！！\n 总钱数：人民币"
                                + totalMoney + "元");
                    }
                } else {
                    if (totalValues <= 9) {                  //骰子的点数为小时显示的提示信息
                        JOptionPane.showMessageDialog(null, "点数是："
                                + totalValues + "，小。\n 玩家赢！！！\n 总钱数：人民币"
                                + totalMoney + "元");
                    } else {
                        JOptionPane.showMessageDialog(null, "点数是："
                                + totalValues + "，大。\n 庄家赢！！！\n 总钱数：人民币"
                                + totalMoney + "元");
                    }
                }
                //这里省略了部分代码
                break;
            }
            try {
```

```
                Thread.sleep(20);
            } catch (Exception ex) {
                System.out.println(flag);
            }
        }
    }
}
```

秘笈心法

心法领悟 091：关于线程与多线程。

世间万物可同时完成很多工作，例如人体同时进行呼吸、血液循环、思考问题等活动；你也可以边听音乐边打游戏，这些活动同时进行的思想在 Java 中被称为并发。Java 语言提供了并发机制，程序员可以在程序中执行多个线程，每一个线程完成一个功能，并与其他线程并发执行。

实例 092　画梅花　　高级

光盘位置：光盘\MR\092　　趣味指数：★★★★★

实例说明

本实例开发了一个画梅花的小游戏，运行程序，效果如图 4.39 所示，在树枝上只有很少的几朵梅花，然后在窗体的左侧选择花的形状，在右侧的树枝上单击即可画梅花，画梅花后的效果如图 4.40 所示。右击，即可删除绘制的梅花。

图 4.39　画梅花之前的效果

图 4.40　画梅花之后的效果

关键技术

在实现画梅花程序时，主要应用了 JPanel 面板的鼠标事件。在画梅花时，捕获到鼠标的左键被按下的事件，并且获取到鼠标指针的位置，在该位置画梅花，即添加一个 JLabel 控件，并将该组件的 ico 属性设置为所选择的梅花样式；在删除梅花时，捕获到鼠标的右键按下事件，并且获取到鼠标指针所在位置的 JLabel 控件（即梅花），并将该组件从面板中移除即可。

说明：在添加梅花和删除梅花时，都需要调用 JPanel 的 repaint()方法重绘 JPanel 面板。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 DrawPlumBlossomFrame 窗体类。

（3）在 DrawPlumBlossomFrame 窗体的左侧添加一个 Jpanel 控件，将其 opaque 属性设置为 false，用于将该 JPanel 的背景设置为透明，并将窗体的布局设置为 null（绝对布局），然后添加 4 个 JLabel 控件，用于显示梅花形状。

（4）分别将 4 个 Jlabel 控件的 variabel 属性设置为 flower1、flower2、flower3 和 flower4，并将其 ico 属性设置为 images 文件夹中的 flower1.png、flower2.png、flower3.png 和 flower4.png。

（5）在 DrawPlumBlossomFrame 窗体的右侧添加一个 JPanel 控件，将其 variable 属性设置为 canvasPane；将其 opaque 属性设置为 false，并将窗体的布局设置为 null（绝对布局），以便在鼠标单击位置处画梅花，然后为该 JPanel 控件添加鼠标按下事件，并判断如果按下的是鼠标左键，则根据选择的花朵形状在窗体上画梅花；如果按下的是鼠标右键，则获取鼠标指针所在位置的梅花，并将其删除，具体代码如下：

```
canvasPane.addMouseListener(new MouseAdapter() {
    public void mousePressed(final MouseEvent e) {
        if (e.getModifiers() == InputEvent.BUTTON1_MASK) {          //按下鼠标左键
            final JLabel flower = new JLabel();                     //显示梅花的标签
            flower.setIcon(SwingResourceManager.getIcon(
                    DrawPlumBlossomFrame.class, "/images/" + flowerType
                            + ".png"));
            if ("flower1".equals(flowerType)) {                     //设置第 1 种类型梅花的大小及位置
                flower.setBounds(e.getX() - 6, e.getY() - 12, 30, 36);
            } else if ("flower2".equals(flowerType)) {              //设置第 2 种类型梅花的大小及位置
                flower.setBounds(e.getX() - 28, e.getY() - 30, 51, 43);
            } else if ("flower3".equals(flowerType)) {              //设置第 3 种类型梅花的大小及位置
                flower.setBounds(e.getX() - 5, e.getY() - 15, 30, 23);
            } else {                                                //设置第 4 种类型梅花的大小及位置
                flower.setBounds(e.getX() - 29, e.getY() - 25, 58, 51);
            }
            canvasPane.add(flower);                                 //添加梅花
            canvasPane.repaint();                                   //重绘面板
        } else if (e.getModifiers() == InputEvent.BUTTON3_MASK) {   //按下右键
            Component at = canvasPane.getComponentAt(e.getPoint()); //获取鼠标位置的组件
            if (at instanceof JLabel) {                             //判断是否为 JLabel
                canvasPane.remove(at);                              //移除组件
                canvasPane.repaint();                               //重绘面板
            }
        }
    }
});
```

说明：由于篇幅原因，这里没有对所有代码进行一一讲解，如果需要对本实例有更进一步的了解，掌握画梅花游戏的开发过程，可以参考本实例的源程序代码。

秘笈心法

心法领悟 092：final 关键字的作用。

本实例语句 **final** JLabel flower = **new** JLabel();中，final 关键字可以用于变量声明，一旦该变量被设定后，就不可以再改变该变量的值。通常 final 定义的变量为常量。

实例 093 打造自己的开心农场

光盘位置：光盘\MR\093　　高级　　趣味指数：★★★★★

实例说明

开心农场是目前比较流行的一款网络游戏。在开心农场里，用户可以种植自己的蔬菜或水果，当一个快乐

的农民。本实例开发了一个开心农场游戏，运行程序，效果如图 4.41 所示，单击“播种”按钮，可以播种种子；单击“生长”按钮，可以让作物处于生长阶段；单击“开花”按钮，可以让作物处于开花阶段；单击“结果”按钮，可以让作物结果；单击“收获”按钮，可以收获果实到仓库中，收获两个果实并且第 3 棵作物已经结果的效果如图 4.42 所示。

图 4.41　游戏刚刚运行时的效果

图 4.42　收获两个果实且第 3 棵作物已结果的效果

关键技术

本实例主要应用了 Java 中的类、对象、成员变量、成员方法和局部变量等技术。另外，在实现作物各种状态的改变时，可以应用已有类 Crop。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类，用于完成播种、生长、开花、结果和收获等操作。

（3）编写一个农场类，名称为 Farm，在该类中编写 seed()方法，用于实现播种操作。在该方法中，如果作物的状态为未播种，则进行播种，将作物显示为播种状态，并修改成员变量 state 的值为 1（表示已播种），否则，设置提示信息为不能播种。

（4）在农场类 Farm 中编写 grow()方法，用于实现生长操作。在该方法中，如果作物的状态为已播种，则让其处于生长阶段，并修改成员变量 state 的值为 2（表示已生长），否则，设置提示信息为不能生长。

（5）在农场类 Farm 中编写 fruit()方法，用于实现结果操作。在该方法中，如果作物的状态为已开花，则让其处于结果阶段，并修改成员变量 state 的值为 4（表示已结果），否则，设置提示信息为不能结果。

（6）在农场类 Farm 中编写 harvest()方法，用于实现收获操作。在该方法中，如果作物的状态为已结果，则收获果实，让作物处于未播种状态，并修改成员变量 state 的值为 0（表示未播种），否则，设置提示信息为不能收获。

（7）在农场类 Farm 中编写 getMessage()方法，用于根据作物的状态属性，确定对应的提示信息，具体代码如下：

```
public String getMessage() {
    String message = "";
    switch (state) {
        case 0:
            message = "作物还没有播种";
            break;
        case 1:
            message = "作物刚刚播种";
            break;
        case 2:
            message = "作物正在生长";
            break;
        case 3:
```

```
                message = "作物正处于开花期";
                break;
            case 4:
                message = "作物已经结果";
                break;
        }
        return message;
    }
```

说明：由于篇幅原因，这里没有对所有代码进行一一讲解，如果需要对本实例有更进一步的了解，掌握整个游戏的开发过程，可以参考本实例的源程序代码。

秘笈心法

心法领悟 093：成员变量与局部变量的区别。

在类体中定义的变量被称为成员变量，它在整个类中都是有效的；在类的方法中定义的变量，即在方法内部定义的变量称为局部变量，它只能被其所在的方法中使用，而成员变量在该类内部始终都是可用的。

第 5 章

打印报表

- 打印控制
- 打印的应用

5.1 打 印 控 制

实例 094 “打印”对话框

光盘位置：光盘\MR\094

初级

趣味指数：★★★★

实例说明

在打印时需要使用“打印”对话框对打印页进行设置，如纸张大小、横向、纵向、打印范围和打印份数等，本实例演示了如何在 Java 应用程序中打开“打印”对话框。运行程序，将打开“打印”对话框，效果如图 5.1 所示，单击“属性”按钮，将打开打印属性对话框，可以对纸张大小、横向、纵向、纸张来源和纸张类型等属性进行设置，效果如图 5.2 所示。

图 5.1 “打印”对话框

图 5.2 打印属性对话框

关键技术

本实例主要是通过 PrinterJob 类的静态方法 getPrinterJob()获得 PrinterJob 对象，然后使用 PrinterJob 对象调用 printDialog()方法打开“打印”对话框。

（1）使用 PrinterJob 类的静态方法 getPrinterJob()，可以获得 PrinterJob 对象，该方法的定义如下：

```
public static PrinterJob getPrinterJob()
```

参数说明

返回值：获得一个新的 PrinterJob 对象。

（2）使用 PrinterJob 对象调用 printDialog()方法，可以打开“打印”对话框，该方法的定义如下：

```
public abstract boolean printDialog()
```

参数说明

返回值：如果用户不取消该对话框，则返回 true；否则返回 false。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PrintDialogDemo 类，在该类的主方法中实现打开“打印”对话框、设置打印任务的名称等操作，该类的代码如下：

```
public class PrintDialogDemo {
    public static void main(String[] args) {
        PrinterJob job = PrinterJob.getPrinterJob();  //获得打印对象
```

```
            if (!job.printDialog()) {                    //打开“打印”对话框
                return;                                  //单击“打印”对话框的“取消”按钮或关闭“打印”对话框，结束程序的执行
            }
            job.setJobName("测试打印对话框");             //设置打印任务的名称
            String jobName = job.getJobName();           //获得打印任务的名称
            System.out.println("打印任务的名称是：  " + jobName);
        }
    }
```

秘笈心法

心法领悟 094：使用“打印”对话框是个很好的习惯。

虽然通过 Java 的打印技术一样可以实现打印，但是使用“打印”对话框确实是个很好的习惯，因为当单击窗体上的“打印”按钮时，若突然想终止打印，或者想更改一下打印内容等，就可以通过“打印”对话框取消打印，否则将无法撤销本次打印作业。

实例 095 实现打印

光盘位置：光盘\MR\095　　初级　　趣味指数：★★★

实例说明

本实例演示了如何在 Java 中实现打印。运行程序，效果如图 5.3 所示，单击窗体上的“打印”按钮，然后单击“打印”对话框中的“确定”按钮，将开始打印唐诗，效果如图 5.4 所示。

图 5.3　本实例的运行效果

图 5.4　正在执行打印任务

关键技术

本实例主要是使用 PrinterJob 类的 setPrintable()方法，并为其传递一个实现了 Printable 接口的实例，该实例实现了接口中的 print()方法，从而实现打印的功能。

接口 Printable 的 print()方法用于绘制打印内容，从而实现打印的功能，该方法的定义如下：

```
int print(Graphics graphics, PageFormat pageFormat, int pageIndex) throws PrinterException
```

参数说明

❶ graphics：用来绘制页面的上下文。

❷ pageFormat：要绘制的页面的大小和方向。

❸ pageIndex：要绘制的页面从 0 开始的索引。

❹ 返回值：如果成功呈现该页面，则返回 PAGE_EXISTS；如果 pageIndex 指定不存在的页面，则返回 NO_SUCH_PAGE。

❺ 异常处理：打印作业被终止时将抛出 PrinterException 异常。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PrintControlFrame 窗体类，该类中的“打印”按钮用于实现打印

功能，其事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        try {
            if (!job.printDialog())                          //打开“打印”对话框
                return;                                      //单击“打印”对话框中的“取消”按钮或关闭“打印”对话框结束程序的执行
            job.setPrintable(new Printable() {
                //实现 print()方法，绘制打印内容
                public int print(Graphics graphics,
                        PageFormat pageFormat, int pageIndex)
                        throws PrinterException {
                    if (pageIndex > 0)
                        return Printable.NO_SUCH_PAGE;       //返回该值表示没有打印页
                    Graphics2D g2 = (Graphics2D) graphics;   //获得图形上下文对象
                    g2.setColor(Color.BLUE);                 //设置当前绘图颜色
                    Font font = new Font("宋体", Font.BOLD, 24);
                    g2.setFont(font);                        //设置字体
                    g2.drawString("静夜思", 80, 40);         //绘制文本
                    font = new Font("宋体", Font.BOLD | Font.ITALIC, 18);
                    g2.setFont(font);                        //设置字体
                    g2.drawString("李白", 130, 70);          //绘制文本
                    font = new Font("宋体", Font.PLAIN, 14);
                    g2.setFont(font);                        //设置字体
                    g2.drawString("床前明月光，", 40, 100);  //绘制文本
                    g2.drawString("疑是地上霜。", 120, 100); //绘制文本
                    g2.drawString("举头望明月，", 40, 120);  //绘制文本
                    g2.drawString("低头思故乡。", 120, 120); //绘制文本
                    return Printable.PAGE_EXISTS;            //返回该值表示存在打印页
                }
            });
            job.setJobName("打印唐诗《静夜思》");            //设置打印任务的名称
            job.print();                                     //调用 print()方法开始打印
        } catch (PrinterException ee) {
            ee.printStackTrace();
        }
    }
});
```

秘笈心法

心法领悟 095：绘图技术是绘制打印内容的关键。

打印页的内容是由绘图技术实现的，包括打印文本的字体、颜色、位置以及其他需要打印的内容等，都是通过 Graphics 类或其子类 Graphics2D 实现的，因此要绘制好的打印内容，必须要掌握并能够灵活应用这两个类。

实例 096 打印图形

初级

光盘位置：光盘\MR\096

趣味指数：★★★

实例说明

本实例演示了如何在 Java 应用程序中实现图形的打印。运行程序，效果如图 5.5 所示，单击窗体上的“打印图形”按钮，将通过打印机打印出五环图形。

图 5.5 打印图形

关键技术

本实例主要是通过 Graphics 类的 drawOval()方法绘制椭圆，使用 Graphics 类的 setColor()方法指定颜色，以及使用 Graphics2D 类的 setStroke()方法设置笔画对象实现的。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PrintShapeFrame 窗体类，该类中的“打印图形”按钮用于实现图形的打印，其事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        try {
            if (!job.printDialog())                          //打开“打印”对话框
                return;                                      //单击“打印”对话框的“取消”按钮或关闭“打印”对话框结束程序的执行
            job.setPrintable(new Printable() {
                //实现 print()方法，绘制打印内容
                public int print(Graphics graphics,
                        PageFormat pageFormat, int pageIndex)
                        throws PrinterException {
                    if (pageIndex > 0)
                        return Printable.NO_SUCH_PAGE;
                    Graphics2D g2 = (Graphics2D)graphics;    //获得 Graphics2D 对象
                    BasicStroke stroke = new BasicStroke(3); //创建宽度是 3 的笔画对象
                    g2.setStroke(stroke);                    //设置笔画对象
                    Color color = new Color(0,162,232);      //创建颜色对象
                    g2.setColor(color);                      //设置颜色
                    g2.drawOval(30, 40, 60, 60);             //绘制第 1 个圆
                    color = new Color(233,123,16);           //创建新的颜色对象
                    g2.setColor(color);                      //设置颜色
                    g2.drawOval(60, 70, 60, 60);             //绘制第 2 个圆
                    color = new Color(28,20,100);            //创建新的颜色对象
                    g2.setColor(color);                      //设置颜色
                    g2.drawOval(92, 40, 60, 60);             //绘制第 3 个圆
                    color = new Color(0,255,0);              //创建新的颜色对象
                    g2.setColor(color);                      //设置颜色
                    g2.drawOval(122, 70, 60, 60);            //绘制第 4 个圆
                    color = new Color(255,0,0);              //创建新的颜色对象
                    g2.setColor(color);                      //设置颜色
                    g2.drawOval(154, 40, 60, 60);            //绘制第 5 个圆
                    return Printable.PAGE_EXISTS;
                }
            });
            job.setJobName("打印五环图形");                   //设置打印任务的名称
            job.print();                                     //调用 print()方法开始打印
        } catch (PrinterException ee) {
            ee.printStackTrace();
        }
    }
});
```

秘笈心法

心法领悟 096：添加形象的说明文字或标题。

本实例实现了五环图形的打印，为了使打印内容更加完美，可以使用 Graphics 类的 drawString()方法为五环图形添加具有指定字体和颜色属性的说明文字或标题，从而实现美化打印内容的功能。

实例 097 打印图片

光盘位置：光盘\MR\097

初级

趣味指数：★★★

实例说明

本实例利用 Java 的打印技术实现了打印图片的功能。运行程序，效果如图 5.6 所示，单击“选择图片”按钮，选择需要打印的图片（选择图片后的效果如图 5.7 所示），单击“打印”按钮即可打印该图片。

图 5.6 选择图片之前的效果

图 5.7 选择图片之后的效果

关键技术

本实例通过文件选择器选择需要打印的图片，然后通过从 Graphics 类继承的 drawImage()方法绘制需要打印的图片，最终完成图片的打印功能。

（1）通过文件选择器 JFileChooser 类的 showOpenDialog()方法，可以打开文件选择对话框，该方法的定义如下：

```
public int showOpenDialog(Component parent)
```

参数说明

❶ parent：该对话框的父组件，可以为 null。

❷ 返回值：单击对话框的“打开”按钮，返回 JFileChooser.APPROVE_OPTION，否则返回 JfileChooser.CANCEL_OPTION。

（2）打开文件选择对话框后，可以通过 getSelectedFile()方法获得所选文件的 File 对象，getSelectedFile()方法的定义如下：

```
public File getSelectedFile()
```

参数说明

返回值：返回所选择文件的 File 对象。

（3）使用 Graphics 类的 drawImage()方法绘制图像。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JPanel 类的 PrintPicturePanel 面板类，实现 Printable 接口，并在实现的 print()方法中绘制打印的内容，print()方法的代码如下：

```
public int print(Graphics graphics, PageFormat pageFormat, int pageIndex)
        throws PrinterException {
    if (pageIndex > 0)
        return Printable.NO_SUCH_PAGE;                    //没有打印页
    int x = (int) pageFormat.getImageableX();             //获得可打印区域的 x 坐标
    int y = (int) pageFormat.getImageableY();             //获得可打印区域的 y 坐标
    Graphics2D g2 = (Graphics2D) graphics;
    int ableW = (int) pageFormat.getImageableWidth();     //获得可打印区域的宽度
```

```
    int ableH = (int) pageFormat.getImageableHeight();          //获得可打印区域的高度
    int imgW = 0;                                               //定义打印图片的宽度
    int imgH = 0;                                               //定义打印图片的高度
    if (src != null) {
        imgW = src.getWidth();                                  //获得图片的宽度
        imgH = src.getHeight();                                 //获得图片的高度
        if (imgW > ableW) {                                     //图片宽度大于打印区域的宽度
            imgW = ableW;                                       //图片宽度为打印区域的宽度
        }
        if (imgH > ableH) {                                     //图片高度大于打印区域的高度
            imgH = ableH;                                       //图片高度为打印区域的高度
        }
    }
    g2.drawImage(src, x, y, imgW, imgH, this);                  //绘制打印内容
    return Printable.PAGE_EXISTS;                               //返回存在打印内容的信息
}
```

（3）面板类 PrintPicturePanel 中的“打印”按钮用于实现打印所选图片的功能，其关键代码如下：

```
try {
    job.setPrintable(PrintPicturePanel.this);                   //为打印对象指定 Printable 对象
    job.setJobName("打印图片");                                  //设置打印任务的名称
    job.print();                                                //执行打印
} catch (PrinterException e1) {
    e1.printStackTrace();
}
```

秘笈心法

心法领悟 097：如何避免无法完全打印所需要的内容。

在打印时需要防止打印内容不全，关键是在绘制打印内容时，确定可打印区域的起始位置、宽度和高度，这样就可以很好地控制打印内容的绘制位置，从而可以完全打印所需要的内容。

实例 098　打印预览

光盘位置：光盘\MR\098　　中级　　趣味指数：★★★

实例说明

为了在打印之前能够清晰地看到打印内容，可以通过打印预览来实现，本实例实现了打印预览功能。运行程序，选择需要打印的图片文件，单击“打印预览”按钮，将显示打印效果，如图 5.8 所示。

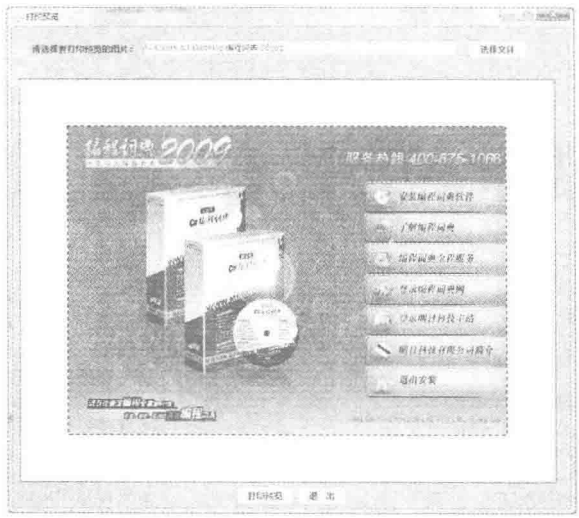


图 5.8　打印预览的效果

关键技术

本实例通过 Canvas 画布对象绘制打印图像的预览界面，并通过 PageFormat 类的方法获得可打印区域的坐标和大小，从而实现打印预览效果。

（1）Canvas 画布用于表示屏幕上一个空白矩形区域，应用程序可以在该区域内绘图或者捕获用户的输入事件，应用程序必须为 Canvas 类创建子类，以获得有用的功能，并要重写 paint()方法，以便在 Canvas 画布上执行自定义图形，paint()方法的定义如下：

```
public void paint(Graphics g)
```

参数说明

g：用于在画布上绘制图像的绘图上下文对象。

（2）使用 PageFormat 类的方法，可以获得可打印区域的坐标和大小，其方法如下：

☑ getImageableX()方法

PageFormat 类的 getImageableX()方法用于获得可打印区域左上角的 x 坐标，该方法的定义如下：

```
public double getImageableX()
```

参数说明

返回值：可打印区域左上角的 x 坐标。

☑ getImageableY()方法

PageFormat 类的 getImageableY()方法用于获得可打印区域左上角的 y 坐标，该方法的定义如下：

```
public double getImageableY()
```

参数说明

返回值：可打印区域左上角的 y 坐标。

☑ getImageableWidth()方法

PageFormat 类的 getImageableWidth()方法用于获得可打印区域的宽度，该方法的定义如下：

```
public double getImageableWidth()
```

参数说明

返回值：可打印区域的宽度。

☑ getImageableHeight()方法

PageFormat 类的 getImageableHeight()方法用于获得可打印区域的高度，该方法的定义如下：

```
public double getImageableHeight()
```

参数说明

返回值：可打印区域的高度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类。

（3）在 MainFrame 窗体类中创建继承 Canvas 类的内部画布类 PreviewCanvas，用于绘制打印预览的图像，该类的定义如下：

```
class PreviewCanvas extends Canvas {
    public void paint(Graphics g) {
        try {
            super.paint(g);
            Graphics2D g2 = (Graphics2D) g;
            g2.translate(10, 10);                                          //平移绘图上下文
            int x = (int) (pf.getImageableX() - 1);                        //获得可打印区域的 x 坐标偏左，用于绘制虚线
            int y = (int) (pf.getImageableY() - 1);                        //获得可打印区域的 y 坐标偏上，用于绘制虚线
            int width = (int) (pf.getImageableWidth() + 1);                //获得可打印区域的宽度偏右，用于绘制虚线
            int height = (int) (pf.getImageableHeight() + 1);              //获得可打印区域的高度偏下，用于绘制虚线
            int mw = (int) pf.getWidth();                                  //获得打印页的宽度
            int mh = (int) pf.getHeight();                                 //获得打印页的高度
            g2.drawRect(0, 0, mw, mh);                                     //绘制实线外框
```

```
                g2.setColor(new Color(255, 253, 234));                          //设置前景色
                g2.fillRect(1, 1, mw - 1, mh - 1);                              //绘制填充区域
                g2.setStroke(new BasicStroke(1f, BasicStroke.CAP_ROUND,
                        BasicStroke.JOIN_ROUND, 10f, new float[] { 5, 5 }, 0f));//指定虚线模式
                g2.setColor(Color.BLACK);                                       //设置前景色
                g2.drawRect(x, y, width, height);                               //绘制虚线内框
                g2.setColor(Color.WHITE);                                       //设置前景色
                g2.fillRect(x + 1, y + 1, width - 1, height - 1);               //绘制填充区域
                MainFrame.this.print(g2, pf, 0);                                //调用 print()方法
            } catch (PrinterException e) {
                e.printStackTrace();
            }
        }
    }
```

说明：由于篇幅原因，这里只给出部分代码，其中 print()是在窗体类 MainFrame 中自定义的方法，用于绘制打印预览的效果图。

秘笈心法

心法领悟 098：绘制专业的打印预览效果。

在实现打印预览时，为了能够更专业地绘制预览内容，可以通过打印页及打印内容的起始位置和大小，绘制不同的矩形（颜色、实线或虚线）、填充矩形等，这样就可以绘制出更加专业的打印预览效果。

实例 099 倒序打印

光盘位置：光盘\MR\099

高级

趣味指数：★★★★

实例说明

本实例利用 Java 的绘图技术和打印技术实现了图片的倒序打印。运行程序，选择图片文件后，单击“打印预览”按钮，进行打印预览，效果如图 5.9 所示，单击“开始打印”按钮即可实现倒序打印。

图 5.9 倒序打印的预览效果

关键技术

本实例通过 Graphics 类的 drawImage()方法实现倒序绘制图像，并通过 Canvas 画布对象绘制打印图像的预览界面。

（1）Canvas 画布用于表示屏幕上一个空白矩形区域，应用程序可以在该区域内绘图或者捕获用户的输入事件，应用程序必须为 Canvas 类创建子类，以获得有用的功能，并要重写 paint()方法，以便在 Canvas 画布上绘制自定义图形。

（2）使用 Graphics 类的 drawImage()方法缩放图像。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类。

（3）在 MainFrame 窗体类中创建实现倒序绘制图像的 printPicture()方法，该方法的代码如下：

```
public void printPicture(Graphics graphics, PageFormat pageFormat,
        int pageIndex) {
    int x = (int) pageFormat.getImageableX();                    //获取可打印区域的 x 坐标
    int y = (int) pageFormat.getImageableY();                    //获取可打印区域的 y 坐标
    Graphics2D g2 = (Graphics2D) graphics;
    if (imgFile != null && isPreview) {
        try {
            src = ImageIO.read(imgFile);                         //构造 BufferedImage 对象
        } catch (IOException e) {
            e.printStackTrace();
        }
        double imgWidth = src.getWidth(this);                    //获得图像的宽度
        double imgHeight = src.getHeight(this);                  //获得图像的高度
        double imgWidthS = imgWidth;                             //存储图像的宽度
        double imgHeightS = imgHeight;                           //存储图像的高度
        int mw = (int) pf.getWidth();                            //获得打印页的宽度
        int mh = (int) pf.getHeight();                           //获得打印页的高度
        if (imgWidth > mw) {                                     //如果宽大于可打印区域
            imgWidth = mw - x;                                   //设置新宽度值
        }
        if (imgHeight > mh) {                                    //如果高大于可打印区域
            imgHeight = mh - y;                                  //设置新的高度值
        }
        g2.drawImage(src, x, y, (int) imgWidth, (int) imgHeight, x,
                (int) imgHeightS, (int) imgWidthS, y, this);     //绘制倒序图像
    }
    isPreview = false;                                           //设置不可以打印
}
```

注意：由于篇幅原因，这里只给出了本实例中关键的代码，即实现倒序打印的 printPicture()方法的代码，其他代码可以参看本实例的源程序代码。

秘笈心法

心法领悟 099：实现各种图像效果的打印。

利用 Java 的绘图技术可以实现各种图像效果，如缩放、旋转和倾斜等，只要通过绘图技术可以实现的图像效果，都可以将其绘制到打印页中，从而实现各种图像效果的打印。

实例 100 为打印内容添加水印

光盘位置：光盘\MR\100　　初级　　趣味指数：★★★

实例说明

本实例利用 Java 的绘图技术和打印技术实现了为打印内容添加水印的功能。运行程序，选择图片文件后，单击“打印预览”按钮，在打开的对话框中进行打印设置后，将在打印页添加水印文字“明日科技”，效果如图 5.10 所示，单击“开始打印”按钮可以打印带有水印的内容。

图 5.10　为打印内容添加水印的效果

关键技术

本实例主要是通过 Graphics2D 类的 setComposite()方法为绘图上下文指定表示透明度的 AlphaComposite 对象，以及使用从 Graphics 类继承的 drawString()方法绘制文本，从而实现了为打印内容添加水印。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类。

（3）在 MainFrame 窗体类中创建为打印内容添加水印的 printPicture()方法，该方法中实现添加水印文字的代码如下：

```
/********************** 添加水印文字 ****************************/
Graphics2D g = src.createGraphics();                    //获取图片绘图上下文
Font font = new Font("黑体", Font.BOLD, wordSize);      //创建字体对象
g.setFont(font);                                        //设置绘图字体
g.setPaint(Color.RED);                                  //设置绘图颜色
Rectangle2D rec = font.getStringBounds(watermarkWord, g
```

```
            .getFontRenderContext());                          //获取文字占用的像素区域
    double pw = rec.getWidth();                                //获取水印文字占用的像素宽度
    double ph = rec.getHeight();                               //获取水印文字占用的像素高度
    g.rotate(Math.toRadians(30), wordSize + pw / 2, wordSize + ph / 2);   //转换角度
    g.setComposite(AlphaComposite.SrcOver.derive(0.4f));       //设置水印透明合成规则
    g.drawString(watermarkWord, wordSize * 2 + (int) pw / 2, wordSize
            * 2 + (int) ph / 2);                               //绘制水印文字
    /*************************************************************/
```

注意：由于篇幅原因，这里只给出了实现为打印内容添加水印的关键代码，其他代码可以参看本实例的源程序代码。

秘笈心法

心法领悟 100：在打印页中添加多处水印。

本实例只在打印页中添加了一处水印，根据需要，在绘制完一处水印后，可以在其他位置继续绘制，从而实现在打印页的多处添加水印。

实例 101　自动为打印内容添加水印

光盘位置：光盘\MR\101　　中级　　趣味指数：★★★★

实例说明

本实例利用 Java 的绘图技术和打印技术实现了自动为打印内容添加水印的功能。运行程序，单击“选择文件”按钮选择图片文件，然后在“请输入水印文字：”标签右侧的文本框中输入作为水印的文字，单击“打印预览”按钮，在打开的对话框中进行打印设置后，将在打印页中自动为打印内容添加水印文字，效果如图 5.11 所示，单击“开始打印”按钮可以打印带有指定水印的内容。

图 5.11　自动为打印内容添加水印的效果

关键技术

通过 JTextField 文本框的 getText()方法，获得用户输入的水印文字，然后将该文字设置为打印内容的水印文字，并通过变量在程序中自动对水印文字的绘制位置和旋转角度的中心点进行控制，从而实现了自动为打印内容添加水印的功能。

（1）使用 JTextField 类从 JTextComponent 类继承的 getText()方法，可以获得用户在文本框中输入的文本，该方法的定义如下：

```
public String getText()
```

参数说明

返回值：用户在文本框中输入的内容。

（2）通过变量在程序中自动对水印文字的绘制位置和旋转角度的中心点进行控制，可以通过如下代码实现：

```
g.rotate(Math.toRadians(30), wordSize + pw / 2, wordSize + ph / 2);          //转换角度
g.setComposite(AlphaComposite.SrcOver.derive(0.4f));                         //设置水印透明合成规则
g.drawString(watermarkWord, wordSize * 2, wordSize * 2 + (int) ph);          //绘制水印文字
```

说明：上面代码中，wordSize 是图片的宽度除以 10 得到的；pw 是水印文字占用的像素宽度；ph 是水印文字占用的像素高度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类。

（3）在窗体 MainFrame 中，“打印预览”按钮用于显示所选图片的预览效果以及获得水印文字，并为图片添加水印文字。“打印预览”按钮的事件代码如下：

```
previewButton.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        //打印预览
        if (imgFile != null) {
            isPreview = true;                                                //表示可以打印
            watermarkWord = watermarkText.getText();                         //获取输入的水印文字
            PrinterJob job = PrinterJob.getPrinterJob();                     //获得打印对象
            pf = job.pageDialog(pf);                                         //显示修改 PageFormat 实例的对话框
            canvas.repaint();                                                //调用 paint()方法
        } else {
            JOptionPane.showMessageDialog(null, "请先选择要打印的图片！");
        }
    }
});
```

（4）在窗体 MainFrame 中，“开始打印”按钮用于打印添加有水印文字的图片，其事件代码如下：

```
printButton.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        PrinterJob job = PrinterJob.getPrinterJob();                         //获取 PrinterJob 对象的实例
        if (!job.printDialog())                                              //打开“打印”对话框
            return;
        //设置打印内容
        job.setPrintable(new Printable() {
            @Override
            public int print(Graphics graphics, PageFormat pageFormat,
                    int pageIndex) throws PrinterException {
                isPreview = true;                                            //设置可以打印
                if (pageIndex < 1) {
                    watermarkWord = watermarkText.getText();                 //获取输入的水印文字
                    printPicture(graphics, pageFormat, pageIndex);           //绘制打印内容
                    return Printable.PAGE_EXISTS;                            //存在打印页
                } else {
                    return Printable.NO_SUCH_PAGE;                           //不存在打印页
                }
```

```
            }
        });
        job.setJobName("打印图形");                    //设置打印任务的名称
        try {
            job.print();                               //调用 print()方法，实现打印
        } catch (PrinterException e1) {
            e1.printStackTrace();
        }
    }
});
```

注意：本实例中为打印内容添加水印文字的方法 printPicture()与实例 100 相同，这里没有给出代码，如果需要可以参看源程序代码。

秘笈心法

心法领悟 101：自动为打印内容添加水印的好处。

自动为打印内容添加水印可以根据需要决定是否添加水印以及所添加水印的内容，可以说这种添加水印的方式更加灵活，如果没有必要添加水印，完全可以不加，并且可以根据不同的打印内容添加相应的水印文字。

5.2 打印的应用

实例 102　打印快递单　　中级

光盘位置：光盘\MR\102　　趣味指数：★★★★

实例说明

本实例演示了如何实现快递单的打印。运行程序，在快递单图片上需要添加文字的位置右击，即可添加一个文本框，输入文字后按 Enter 键确认，可以隐藏文本框的边框，单击该文本框，又可以显示其边框，重复上述过程可以实现快递单文字的添加，效果如图 5.12 所示。

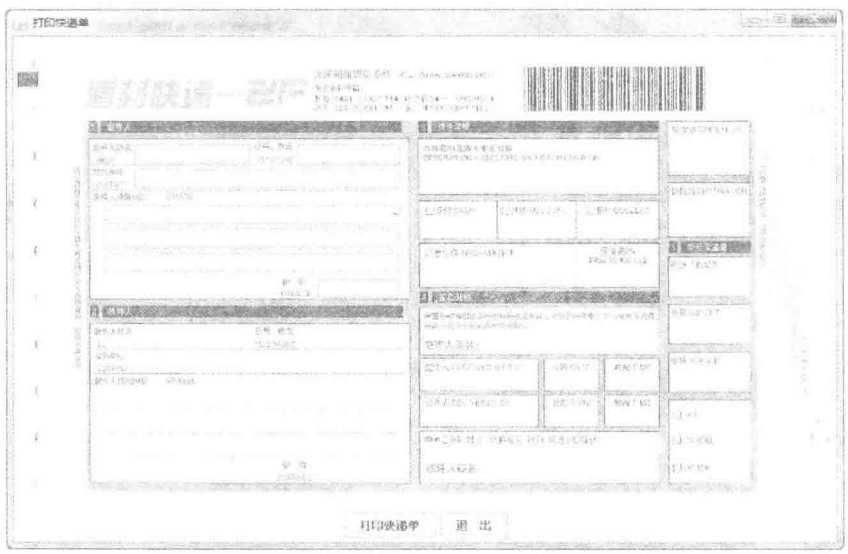


图 5.12　打印快递单界面

关键技术

本实例根据快递单图片背景面板的位置和大小创建 Rectangle 对象，然后将该对象传递给 Robot 类的 createScreenCapture()方法，捕获窗体上快递单图片的图像缓冲对象，并将其绘制到打印页面上，从而实现了快递单的打印。

（1）获得窗体上快递单图片背景面板的位置和大小，并创建 Rectangle 对象，代码如下：

```
int x = (int)(ExpressPrintFrame.this.getBounds().getX())+8;                    //背景面板在屏幕上的 x 坐标
int y = (int)(ExpressPrintFrame.this.getBounds().getY())+30;                   //背景面板在屏幕上的 y 坐标
int w = (int)backPanel.getBounds().getWidth();                                 //背景面板的宽度
int h = (int)backPanel.getBounds().getHeight();                                //背景面板的高度
Rectangle rect = new Rectangle(x, y, w, h);                                    //创建 Rectangle 对象
```

（2）使用 Robot 类的 createScreenCapture()方法，捕获窗体上快递单图片的图像缓冲对象，并将其绘制到打印页，关键代码如下：

```
buffImage = robot.createScreenCapture(rect);                                   //获得缓冲图像对象
//省略了部分代码
g2.drawImage(buffImage, printX, printY, imgWidth, imgHeight, ExpressPrintFrame.this);   //将缓冲图像绘制到打印页
```

说明：上面代码中，rect 是 Rectangle 对象，即窗体上显示快递单图片面板 Rectangle 对象，buffImage 是图像缓冲对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JTextField 类的文本框类 TargetTextField，以实现用鼠标改变文本框在父级容器中的位置和大小。

（3）在项目中创建一个继承自 JFrame 类的窗体类 ExpressPrintFrame，以实现快递单的打印功能，其中打印文字可以通过在背景面板上右击添加文本框实现。背景面板的鼠标右键事件代码如下：

```
backPanel.addMouseListener(new MouseAdapter() {
    public void mouseClicked(final MouseEvent e) {
        if (e.getButton() == MouseEvent.BUTTON3) {                             //右击
            int x = e.getX();                                                  //获得鼠标位置的 x 坐标
            int y = e.getY();                                                  //获得鼠标位置的 y 坐标
            TargetTextField tf = new TargetTextField();                        //创建自定义文本框的实例
            tf.addMouseListener(tf);                                           //添加鼠标监听器
            tf.addMouseMotionListener(tf);                                     //添加鼠标监听器
            tf.addActionListener(tf);                                          //添加动作监听器
            tf.setBounds(x, y, 147, 22);                                       //指定文本框的位置和大小
            backPanel.add(tf);                                                 //添加到背景面板上
            tf.requestFocus();                                                 //使文本框获得焦点
        }
    }
});
```

说明：由于篇幅原因，这里并没有给出所有代码，具体实现过程可以查看源程序代码。

秘笈心法

心法领悟 102：创建和设置鼠标指针的图标。

在进行 Java 应用程序开发时，可以根据需要改变鼠标指针的图标，这可以使用 Cursor 类创建鼠标指针对象，然后使用 setCursor()方法为控件设置鼠标指针对象来实现。

实例 103 打印报表

光盘位置：光盘\MR\103

高级

趣味指数：★★★★

实例说明

本实例通过 Java 的绘图技术和打印技术实现了打印报表的功能。运行程序，单击窗体上的“预览报表”按钮，在弹出的对话框中进行打印设置后，将把数据库中的信息显示在预览界面中，效果如图 5.13 所示，单击“打印报表”按钮，将把这些信息打印出来。

图 5.13 打印报表的预览效果

关键技术

本实例使用 JDBC 技术从数据库表中获得结果集，然后通过 Graphics 类的 drawString()方法绘制文本，使用 drawLine()方法绘制报表的表格，从而实现了打印报表的功能。

（1）为了获得数据库表中的数据记录，本实例定义了一个 QueryResultSet 类，在该类中定义了一个 gainRecord()方法，用于获得数据库表的结果集对象，该方法的代码如下：

```
public static ResultSet gainRecord() {
    Connection conn = null;                                          //声明连接
    Statement st = null;                                             //声明 Statement 对象
    ResultSet rs = null;                                             //声明结果集对象
    try {
        String url = "jdbc:jtds:sqlserver://localhost:1433/db_database";  //数据库 db_database 的 URL
        String username = "sa";                                      //数据库的用户名
        String password = "";                                        //数据库密码
        conn = DriverManager.getConnection(url, username, password); //建立连接
        st = conn.createStatement();                                 //创建 Statement 对象
        String sql = "select * from tb_employee";                    //定义 SQL 查询语句
        rs = st.executeQuery(sql);                                   //执行 SQL 语句获得结果集
        return rs;                                                   //返回结果集
```

```
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "数据库异常：\n" + e.getMessage());
            return null;
        }
    }
```

注意：在使用本实例时，需要将 SQL Server 数据库的 JTDS 驱动配置到项目的类库路径中，否则程序将无法连接数据库。

（2）使用 Graphics 类的 drawString()方法绘制结果集中的记录数据，代码如下：

```
g2.drawString(String.valueOf(rs.getInt(1)), x + 30, y);                    //绘制报表内容
g2.drawString(rs.getString(2), x + 60, y);                                 //绘制报表内容
//省略了部分代码
```

说明：上面代码中的 rs 是通过 QueryResultSet 类的 gainRecord()方法获得的结果集对象，用于获得数据库表中的数据记录。

（3）使用 Graphics 类的 drawLine()方法，绘制报表的表格，代码如下：

```
g2.drawLine(x + 20, y1 + 10, x + 20, y + 10);                              //绘制垂直直线
g2.drawLine(x + 50, y1 + 10, x + 50, y + 10);                              //绘制垂直直线
//省略了部分代码
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 QueryResultSet 类，用于加载数据库驱动、连接数据库以及获得数据库表中的结果集。

（3）在项目中创建一个继承自 JFrame 类的 MainFrame 窗体类，在该类中定义一个 printPicture()方法，用于绘制打印的报表内容，该方法的关键代码如下：

```
try {
    y = y + 80;                                                            //调整打印位置的 y 值
    int y1 = y;                                                            //保存调整后打印位置的 y 值
    while (rs.next()) {
        y = y + 30;                                                        //调整打印位置的 y 值
        g2.drawLine(x + 20, y - 20, x + w - 20, y - 20);                   //绘制水平直线
        g2.drawString(String.valueOf(rs.getInt(1)), x + 30, y);            //绘制报表内容
        g2.drawString(rs.getString(2), x + 60, y);                         //绘制报表内容
        g2.drawString(rs.getString(3), x + 120, y);                        //绘制报表内容
        g2.drawString(String.valueOf(rs.getInt(4)), x + 170, y);           //绘制报表内容
        g2.drawString(rs.getString(5), x + 220, y);                        //绘制报表内容
        g2.drawString(rs.getString(6), x + 420, y);                        //绘制报表内容
        g2.drawString(rs.getString(7), x + 480, y);                        //绘制报表内容
    }
    g2.drawLine(x + 20, y1 + 10, x + 20, y + 10);                          //绘制垂直直线
    g2.drawLine(x + 50, y1 + 10, x + 50, y + 10);                          //绘制垂直直线
    g2.drawLine(x + 110, y1 + 10, x + 110, y + 10);                        //绘制垂直直线
    g2.drawLine(x + 160, y1 + 10, x + 160, y + 10);                        //绘制垂直直线
    g2.drawLine(x + 210, y1 + 10, x + 210, y + 10);                        //绘制垂直直线
    g2.drawLine(x + 410, y1 + 10, x + 410, y + 10);                        //绘制垂直直线
    g2.drawLine(x + 470, y1 + 10, x + 470, y + 10);                        //绘制垂直直线
    g2.drawLine(x + w - 20, y1 + 10, x + w - 20, y + 10);                  //绘制垂直直线
    g2.drawLine(x + 20, y + 30 - 20, x + w - 20, y + 30 - 20);             //绘制水平直线
    rs.close();                                                            //关闭结果集
} catch (Exception ex) {
    ex.printStackTrace();
}
```

秘笈心法

心法领悟 103：使用 JTable 类打印报表。

在 Java 中，JTable 类提供了重载的 print()方法，通过这些方法可以实现报表的打印功能，并可以根据需要决定是否需要打印标题和脚注等信息。

实例 104 打印桌面图片

光盘位置：光盘\MR\104

中级

趣味指数：★★★★

实例说明

本实例实现了打印桌面图片的功能。运行程序，单击窗体上的“获取桌面”按钮，将隐藏打印桌面图片窗体，并抓取屏幕图片，效果如图 5.14 所示，用户还可以单击“页面设置”按钮进行页面设置，然后单击“打印”按钮就可以打印桌面图片。

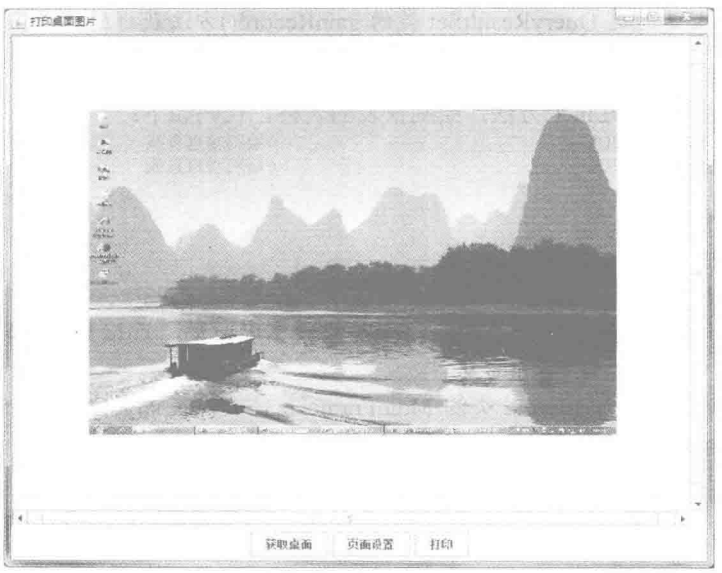


图 5.14 获取桌面图片的效果

关键技术

本实例是在单击窗体上的“获取桌面”按钮时隐藏当前窗体，然后使用 Robot 类的 createScreenCapture()方法来捕获桌面图片，再显示当前窗体，从而可以使用 Java 的打印技术实现打印桌面图片的功能。

（1）使用 JFrame 类从 Window 类继承的 setVisible()方法可以隐藏和显示窗体，该方法的定义如下：

```
public void setVisible(boolean b)
```

参数说明

b：布尔型值，如果为 true，则显示窗体；如果为 false，则隐藏窗体。

（2）使用 Robot 类的 createScreenCapture()方法，可以捕获桌面图片，该方法的定义如下：

```
public BufferedImage createScreenCapture(Rectangle screenRect)
```

参数说明

❶ screenRect：在屏幕中捕获的矩形区域的位置和大小。

❷ 返回值：存储桌面图片的缓冲图像对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JPanel 类并实现 Printable 接口的自定义面板类 PrintPanel，在实现的 print()方法中完成打印内容的绘制，该方法的代码如下：

```
public int print(Graphics g1, PageFormat pageFormat, int pageIndex)
        throws PrinterException {
    Graphics2D g = (Graphics2D) g1;
    px = (int) pageFormat.getImageableX();                    //获取可打印区域的 x 坐标
    py = (int) pageFormat.getImageableY();                    //获取可打印区域的 y 坐标
    pwidth = (int) pageFormat.getImageableWidth();            //获取可打印的宽度
    pheight = (int) pageFormat.getImageableHeight();          //获取可打印的高度
    int pageWidth = (int) pageFormat.getWidth();              //获取打印页面宽度
    int pageHeight = (int) pageFormat.getHeight();            //获取打印页面高度
    Dimension preferredSize = new Dimension(pageWidth, pageHeight);
    setPreferredSize(preferredSize);                          //设置面板大小
    getParent().doLayout();                                   //重写布局父容器
    g.setColor(Color.WHITE);                                  //设置前景色为白色
    g.fill3DRect(0, 0, pageWidth, pageHeight, true);          //绘制与打印页面相同大小的矩形
    if (pageIndex < 1) {                                      //如果当前打印页数小于 1
        g.drawImage(image, px, py, pwidth, pheight, this);    //绘制打印内容
        return Printable.PAGE_EXISTS;                         //返回可打印标识
    } else                                                    //否则
        return Printable.NO_SUCH_PAGE;                        //返回不支持打印标识
}
```

（3）在项目中创建一个继承 JFrame 类的 PrintDesktop 窗体类，其中的“获取桌面”按钮，用于捕获桌面图片，其事件代码如下：

```
getDesktopBtn.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        try {
            Robot robot = new Robot();                                  //创建 Robot 类的实例对象
            Dimension size = getToolkit().getScreenSize();              //获取屏幕大小
            Rectangle screenRect = new Rectangle(0, 0, size.width,
                    size.height);                                       //创建屏幕大小的矩形对象
            setVisible(false);                                          //隐藏窗体
            Thread.sleep(1000);                                         //休眠 1 秒钟
            BufferedImage image = robot.createScreenCapture(screenRect); //抓取屏幕图像
            setVisible(true);                                           //显示窗体
            printPanel.setImage(image);                                 //为打印面板设置图像
            printPanel.repaint();                                       //重新绘制打印面板
        } catch (AWTException e1) {
            e1.printStackTrace();
        } catch (InterruptedException e1) {
            e1.printStackTrace();
        }
    }
});
```

注意：为了使本实例能够打印出桌面图片，在获取桌面图片之前，需要将其他窗口全部最小化或关闭，否则将无法获得或者无法完全获得桌面图片。

秘笈心法

心法领悟 104：使用 Robot 类控制鼠标和键盘。

在进行程序开发时，有时需要通过程序对鼠标和键盘进行控制，这时可以使用 Robot 类提供的 mouseMove()、mousePress()、mouseRelease()和 mouseWheel()方法对鼠标进行控制，使用 keyPress()和 keyRelease()方法对键盘进行控制。

实例 105 打印柱形图表

高级

光盘位置：光盘\MR\105

趣味指数：★★★★

实例说明

本实例使用柱形图表统计出来自各个城市男女员工的平均年龄。运行程序，将以柱形图表显示各个城市男

女员工的平均年龄，效果如图 5.15 所示，单击窗体上的“打印”按钮，可以打印该柱形图表。

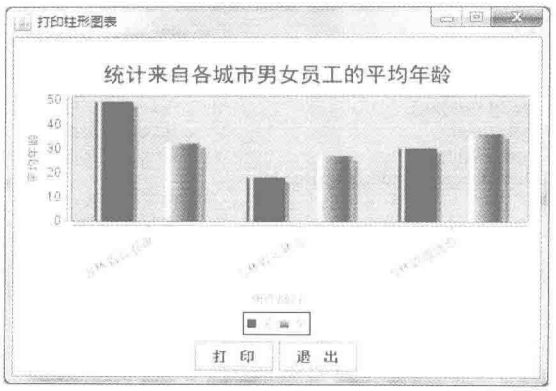


图 5.15　柱形图表显示男女员工的平均年龄

关键技术

本实例使用图表工厂类 ChartFactory，调用 createBarChart()方法创建了一个 JFreeChart 对象，该对象是一个柱形图表对象。createBarChart()方法的定义如下：

```
public static JFreeChart createBarChart(String title,
                                        String categoryAxisLabel,
                                        String valueAxisLabel,
                                        CategoryDataset dataset,
                                        PlotOrientation orientation,
                                        boolean legend,
                                        boolean tooltips,
                                        boolean urls)
```

参数说明

❶ title：图表的标题文本。

❷ categoryAxisLabel：X 轴上的标签文字。

❸ valueAxisLabel：Y 轴上的标签文字。

❹ dataset：数据集。

❺ orientation：图表的方向。

❻ legend：是否显示图例。

❼ tooltips：是否显示说明文字。

❽ urls：是否生成链接。

❾ 返回值：柱形图表的 JFreeChart 对象。

说明：在创建图表对象时，需要到其官方网站 http://www.jfree.org 下载对应的包，并将相应的包配置到类库路径中，否则将无法进行图表开发。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 QueryResultSet 类，用于加载数据库驱动、连接数据库以及获得数据库表中的结果集。

（3）在项目中创建一个继承自 JFrame 类的 PrintHistogramFrame 窗体类，在该类中定义一个 createDataset()方法，用于获得数据集对象，该方法的代码如下：

```
private static CategoryDataset createDataset() {
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();                    //创建数据集对象
```

```
        ResultSet rs = QueryResultSet.gainRecord();                          //获得查询结果集
        try {
            while (rs.next()) {
                int value = rs.getInt(1);                                    //获得年龄
                String sex = rs.getString(2);                                //获得性别
                String address = rs.getString(3);                            //获得地址
                dataset.addValue(value, sex, address);                       //在数据集中添加数据信息
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return dataset;                                                      //返回数据集对象
    }
```

（4）在 PrintHistogramFrame 窗体类中定义一个 createChart()方法，用于获得柱形图表对象，该方法中创建柱形图表的代码如下：

```
JFreeChart chart = ChartFactory.createBarChart("\n\n 统计来自各城市男女员工的平均年龄",   //图表的标题文本
        "所在城市          ",                                                //x 轴上的标签文字
        "平均年龄",                                                          //y 轴上的标签文字
        dataset,                                                             //数据集
        PlotOrientation.VERTICAL,                                            //垂直方向
        true,                                                                //显示图例
        true,                                                                //显示说明文字
        false                                                                //不生成链接
        );
```

秘笈心法

心法领悟 105：绘制立体效果的柱形图表。

在使用图表工厂类 ChartFactory 创建 JFreeChart 对象时，如果调用 createBarChart3D()方法创建图表对象，则所创建的图表对象就是一个立体效果的柱形图表对象。

实例 106 打印饼形图表

光盘位置：光盘\MR\106　　高级　　趣味指数：★★★

实例说明

本实例使用饼形图表，统计出来自各个城市男女员工的平均年龄。运行程序，将以饼形图表显示各个城市男女员工的平均年龄，效果如图 5.16 所示，单击窗体上的“打印”按钮，可以打印该饼形图表。

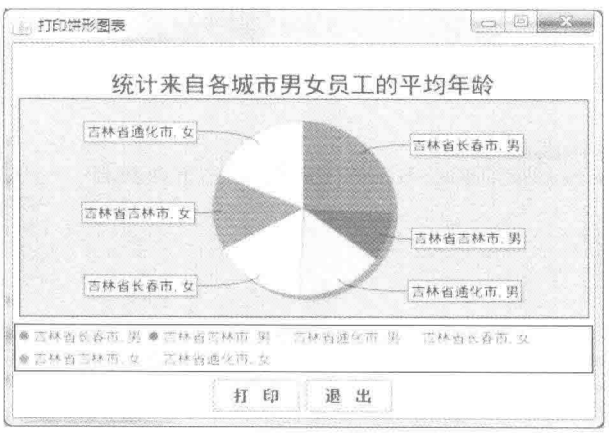


图 5.16　饼形图表显示男女员工的平均年龄

关键技术

本实例使用图表工厂类 ChartFactory，调用 createPieChart()方法创建了一个 JFreeChart 对象，该对象是一个饼形图表对象。createPieChart()方法的定义如下：

```
public static JFreeChart createPieChart(String title,
                                        PieDataset dataset,
                                        boolean legend,
                                        boolean tooltips,
                                        boolean urls)
```

参数说明

❶ title：图表的标题文本。

❷ dataset：数据集。

❸ legend：是否显示图例。

❹ tooltips：是否显示说明文字。

❺ urls：是否生成链接。

❻ 返回值：饼形图表的 JFreeChart 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 QueryResultSet 类，用于加载数据库驱动、连接数据库以及获得数据库表中的结果集。

（3）在项目中创建一个继承自 JFrame 类的 PrintPieChartFrame 窗体类，在该类中定义一个 createDataset()方法，用于获得数据集对象，该方法的代码如下：

```
private static PieDataset createDataset() {
    DefaultPieDataset dataset = new DefaultPieDataset();                    //创建数据集对象
    ResultSet rs = QueryResultSet.gainRecord();                             //获得查询结果集
    try {
        while (rs.next()) {
            int value = rs.getInt(1);                                       //获得平均年龄
            String sex = rs.getString(2);                                   //获得性别
            String address = rs.getString(3);                               //获得地址
            dataset.setValue(address + "," + sex, value);                   //设置数据集的值
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return dataset;
}
```

（4）在 PrintPieChartFrame 窗体类中定义一个 createChart()方法，用于获得饼形图表对象，该方法中创建饼形图表的代码如下：

```
private static JFreeChart createChart(PieDataset dataset) {
    JFreeChart chart = ChartFactory.createPieChart("\n\n 统计来自各城市男女员工的平均年龄",   //图表的标题文本
            dataset,                                                        //数据集
            true,                                                           //显示图例
            true,                                                           //显示说明文字
            false                                                           //不生成链接
            );
    chart.setBackgroundPaint(Color.white);                                  //设置背景颜色
    chart.getTitle().setFont(new Font("黑体", Font.PLAIN, 20));             //设置标题字体
    chart.getLegend().setItemFont(new Font("黑体", Font.PLAIN, 12));        //设置图例文本的字体
    PiePlot plot = (PiePlot) chart.getPlot();                               //获得图表的 PiePlot 对象
    plot.setLabelFont(new Font("黑体", Font.PLAIN, 12));                    //设置饼状图标签的字体
    return chart;
}
```

秘笈心法

心法领悟 106：绘制立体效果的饼形图表。

在使用图表工厂类 ChartFactory 创建 JFreeChart 对象时，如果调用 createPieChart3D()方法创建图表对象，则所创建的图表对象就是一个立体效果的饼形图表对象。

实例 107 打印折线图表

光盘位置：光盘\MR\107

中级

趣味指数：★★★

实例说明

本实例使用折线图表统计出来自各个城市男女员工的平均年龄。运行程序，将以折线图表显示各个城市男女员工的平均年龄，效果如图 5.17 所示，单击窗体上的“打印”按钮，可以打印该折线图表。

图 5.17 折线图表显示男女员工的平均年龄

关键技术

本实例使用图表工厂类 ChartFactory，调用 createLineChart()方法创建了一个 JFreeChart 对象，该对象是一个折线图表对象。createLineChart()方法的定义如下：

```
public static JFreeChart createLineChart(String title,
                                         String categoryAxisLabel,
                                         String valueAxisLabel,
                                         CategoryDataset dataset,
                                         PlotOrientation orientation,
                                         boolean legend,
                                         boolean tooltips,
                                         boolean urls)
```

参数说明

❶ title：图表的标题文本。

❷ categoryAxisLabel：X 轴上的标签文字。

❸ valueAxisLabel：Y 轴上的标签文字。

❹ dataset：数据集。

❺ orientation：图表的方向。

❻ legend：是否显示图例。

❼ tooltips：是否显示说明文字。

❽ urls：是否生成链接。

❾ 返回值：折线图表的 JFreeChart 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 QueryResultSet 类，用于加载数据库驱动、连接数据库以及获得数据库表中的结果集。

（3）在项目中创建一个继承自 JFrame 类的 PrintLineChartFrame 窗体类，在该类中定义一个 createDataset()方法，用于获得数据集对象，该方法的代码如下：

```
private static CategoryDataset createDataset() {
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();                    //创建数据集对象
    ResultSet rs = QueryResultSet.gainRecord();                                       //获得查询结果集
    try {
        while (rs.next()) {
            int value = rs.getInt(1);                                                 //获得平均年龄
            String sex = rs.getString(2);                                             //获得性别
            String address = rs.getString(3);                                         //获得地址
            dataset.addValue(value, sex, address);                                    //向数据集添加数据
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return dataset;
}
```

（4）在 PrintLineChartFrame 窗体类中定义一个 createChart()方法，用于获得折线图表对象，该方法中创建折线图表的代码如下：

```
JFreeChart chart = Chart Factory.createLineChart("\n\n 统计来自各城市男女员工的平均年龄",    //图表的标题文本
        "所在城市",                                                                   //X 轴上的标签文字
        "平均年龄",                                                                   //Y 轴上的标签文字
        dataset,                                                                      //数据集
        PlotOrientation.VERTICAL,                                                     //垂直方向
        true,                                                                         //显示图例
        true,                                                                         //显示说明文字
        false                                                                         //不生成链接
        );
```

秘笈心法

心法领悟 107：绘制立体效果的折线图表。

在使用图表工厂类 ChartFactory 创建 JFreeChart 对象时，如果调用 createLineChart3D()方法创建图表对象，则所创建的图表对象就是一个立体效果的折线图表对象。

实例 108　打印区域图表

光盘位置：光盘\MR\108　　高级　　趣味指数：★★★★★

实例说明

本实例实现区域图表对象的打印。运行程序，将在窗体上以区域图表的形式随机显示明日科技图书在 2009 年各个月份的用户满意度，效果如图 5.18 所示。

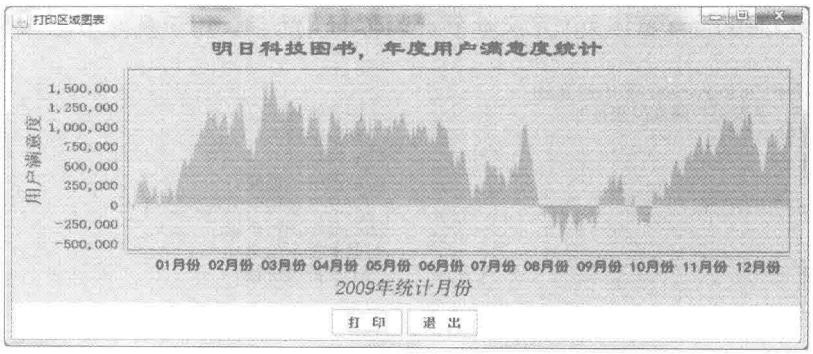


图 5.18 区域图表的显示效果

关键技术

本实例使用图表工厂类 ChartFactory，调用 createXYAreaChart()方法创建了一个 JFreeChart 对象，该对象是一个区域图表对象。createXYAreaChart()方法的定义如下：

```
public static JFreeChart createXYAreaChart(String title,
                                           String xAxisLabel,
                                           String yAxisLabel,
                                           XYDataset dataset,
                                           PlotOrientation orientation,
                                           boolean legend,
                                           boolean tooltips,
                                           boolean urls)
```

参数说明

❶ title：图表的标题文本。

❷ xAxisLabel：X 轴上的标签文字。

❸ yAxisLabel：Y 轴上的标签文字。

❹ dataset：数据集。

❺ orientation：图表的方向。

❻ legend：是否显示图例。

❼ tooltips：是否显示说明文字。

❽ urls：是否生成链接。

❾ 返回值：折线图表的 JFreeChart 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的 PrintAreaChartFrame 窗体类，在该类中定义一个 createDataset()方法，用于使用随机获得的数据创建数据集对象，该方法的代码如下：

```
private XYDataset createDataset() {
    long value = 0;
    Day day = new Day(1, 1, 2009);                          //获得 2009 年 1 月 1 日的 Day 对象
    long seed = System.currentTimeMillis();                 //获得当前时间的毫秒数
    Random ran = new Random(seed);                          //创建随机数种子为 seed 的 Random 对象
    TimeSeries soft = new TimeSeries("明日科技图书");        //创建时间序列
    for (int i = 0; i < 365; i++) {                         //添加一年 365 天的数据
        value += ran.nextInt() / 10000;                     //随机获得数据
        soft.add(day, value);                               //添加数据
        day = (Day) day.next();                             //获得下一个日期的 Day 对象
    }
    TimeSeriesCollection dataset = new TimeSeriesCollection(soft);   //创建数据集对象
    return dataset;
}
```

（3）在 PrintAreaChartFrame 窗体类中定义一个 createChart()方法，用于获得区域图表对象，该方法中创建区域图表的代码如下：

```
JFreeChart jfreechart = ChartFactory.createXYAreaChart(
        "明日科技图书，年度用户满意度统计",          //图表标题
        "2009 年统计月份",                          //X 轴标题
        "用户满意度",                               //Y 轴标题
        xydataset,                                  //制图的数据集
        PlotOrientation.VERTICAL,                   //定义区域图的方向为纵向
        false,                                      //不显示图例标识
        true,                                       //显示提示信息
        false);                                     //不支持超链接
```

秘笈心法

心法领悟 108：使用数据库表中的记录创建区域图表。

使用图表工厂类 ChartFactory，调用 createAreaChart()方法，可以创建区域图表对象，该方法的参数与 createXYAreaChart()方法基本相同，唯一不同的就是创建数据集的第 4 个参数，只需要用 CategoryDataset 创建数据集，并将数据库表中的记录信息添加到该数据集中，即可实现使用数据库表中的记录创建区域图表。

实例 109　打印带柱形图表的报表

光盘位置：光盘\MR\109　　高级　　趣味指数：★★★★

实例说明

本实例实现了打印带柱形图表的报表。运行程序，单击窗体上的“预览报表”按钮，将显示带柱形图表的报表预览图，效果如图 5.19 所示，单击“打印报表”按钮，将打印该报表。

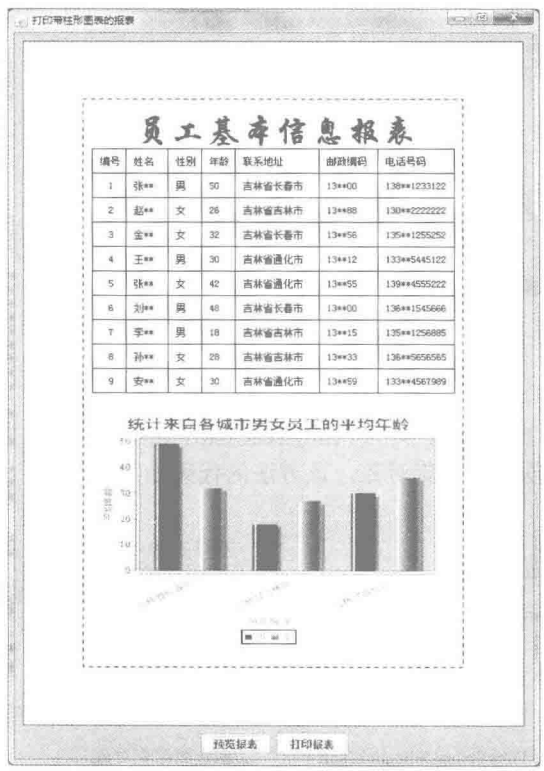


图 5.19　带柱形图表的报表预览效果

关键技术

本实例使用图表工厂类 ChartFactory，调用 createBarChart()方法创建了一个柱形图表对象，然后使用 ChartUtilities 类的 writeChartAsJPEG()方法将柱形图表对象保存为 JPG 图片，最后将柱形图表的 JPG 图片对象绘制到打印页，从而实现了打印带柱形图表的报表。

（1）使用图表工厂类 ChartFactory，调用 createBarChart()方法，可以创建 JFreeChart 对象，该对象是一个柱形图表对象，有关图表工厂类 ChartFactory 的 createBarChart()方法请查看实例 105。

（2）使用 ChartUtilities 类的 writeChartAsJPEG()方法将柱形图表对象保存为 JPG 格式的图片。writeChartAsJPEG()方法的定义如下：

```
public static void writeChartAsJPEG(OutputStream out, JFreeChart chart, int width, int height)
```

参数说明

❶ out：将图表输出为磁盘图片的输出流对象。

❷ chart：被写入磁盘的图表对象。

❸ width：生成图片的宽度。

❹ height：生成图片的高度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 QueryResultSet 类，用于加载数据库驱动、连接数据库以及获得数据库表中的数据信息。

（3）在项目中创建一个继承自 JFrame 类的 PrintBarReportFrame 窗体类，在该类的构造方法中，添加将图表对象保存为图片的代码：

```
CategoryDataset dataset = createDataset();                                  //创建数据集对象
JFreeChart chart = createChart(dataset);                                    //创建图表对象
String path = System.getProperty("user.dir")+"/src/chartImg/chart.jpg";     //图表的存储位置
try {
    ChartUtilities.writeChartAsJPEG(new FileOutputStream(path), chart, 450, 360);   //将图表存储为图片
} catch (FileNotFoundException e1) {
    e1.printStackTrace();
} catch (IOException e1) {
    e1.printStackTrace();
}
```

说明：上面代码将图表图片存储到项目的 src 文件夹的子文件夹 chartImg 中，图片文件的名称为 chart.jpg，由于上述代码放在构造方法中，所以在程序运行时就将图表保存为图片了。

（4）在 PrintBarReportFrame 窗体类中定义一个 printPicture()方法，用于完成打印内容的绘制，该方法中用于绘制图表图片的代码如下：

```
URL imgUrl = PrintBarReportFrame.class.getResource("/chartImg/chart.jpg");   //获取图片资源的路径
Image img = Toolkit.getDefaultToolkit().getImage(imgUrl);                    //获取图表图片的图像对象
g2.drawImage(img, x+10, y+20,420,300, this);                                 //在打印页绘制图表图片
```

秘笈心法

心法领悟 109：使用换位思考的方法解决问题。

由于不能在绘图上下文中直接绘制 JFreeChart 图表，而只能绘制文本、图形和图像，所以可以考虑是否可以将图表存储为图片，如果可以，就可以将图表存储的图片绘制到绘图上下文中，从而实现图表在打印页中的绘制，这就是使用了换位思考的方法。

实例 110 打印带饼形图表的报表

光盘位置：光盘\MR\110

高级

趣味指数：★★★★

实例说明

本实例实现了打印带饼形图表的报表。运行程序，单击窗体上的“预览报表”按钮，将显示带饼形图表的报表预览图，效果如图 5.20 所示，单击“打印报表”按钮，将打印该报表。

编号	姓名	性别	年龄	联系地址	邮政编码	电话号码
1	张**	男	50	吉林省长春市	13**00	138**1233122
2	赵**	女	26	吉林省吉林市	13**88	130**2222222
3	金**	女	32	吉林省长春市	13**56	135**1255252
4	王**	男	30	吉林省通化市	13**12	133**5445122
5	张**	女	42	吉林省通化市	13**55	139**4555222
6	刘**	男	48	吉林省长春市	13**00	136**1545666
7	李**	男	18	吉林省吉林市	13**15	135**1256885
8	孙**	女	28	吉林省吉林市	13**33	136**5656565
9	安**	女	30	吉林省通化市	13**59	133**4567989

图 5.20 带饼形图表的报表预览效果

关键技术

本实例使用图表工厂类 ChartFactory，调用 createPieChart()方法创建了一个饼形图表对象，然后使用 ChartUtilities 类的 writeChartAsJPEG()方法将饼形图表对象保存为 JPG 图片，最后将饼形图表的 JPG 图片对象绘制到打印页，从而实现了打印带饼形图表的报表。

（1）使用图表工厂类 ChartFactory，调用 createPieChart()方法，可以创建 JFreeChart 对象，该对象是一个饼形图表对象，有关图表工厂类 ChartFactory 的 createPieChart()方法请查看实例 106。

（2）使用 ChartUtilities 类的 writeChartAsJPEG()方法，将饼形图表对象保存为 JPG 格式的图片，ChartUtilities 类的 writeChartAsJPEG()方法请查看实例 109。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 QueryResultSet 类，用于加载数据库驱动、连接数据库以及获得数据库表中的数据信息。

（3）在项目中创建一个继承自 JFrame 类的 PrintPieReportFrame 窗体类，在该类的构造方法中，添加将图表对象保存为图片的代码：

```
PieDataset dataset = createDataset();                                    //创建数据集对象
JFreeChart chart = createChart(dataset);                                 //创建图表对象
String path = System.getProperty("user.dir") + "/src/chartImg/chart.jpg";  //图表的存储位置
try {
    ChartUtilities.writeChartAsJPEG(new FileOutputStream(path), chart, 450, 360);  //将图表存储为图片
} catch (FileNotFoundException e2) {
    e2.printStackTrace();
} catch (IOException e2) {
    e2.printStackTrace();
}
```

说明：上面代码将图表图片存储到项目的 src 文件夹的子文件夹 chartImg 中，图片文件的名称为 chart.jpg，由于上述代码放在构造方法中，所以在程序运行时就将图表保存为图片了。

（4）在 PrintPieReportFrame 窗体类中定义一个 printPicture()方法，用于完成打印内容的绘制，该方法中用于绘制图表图片的代码如下：

```
URL imgUrl = PrintPieReportFrame.class.getResource("/chartImg/chart.jpg");  //获取图片资源的路径
Image img = Toolkit.getDefaultToolkit().getImage(imgUrl);                   //获取图表图片的图像对象
g2.drawImage(img, x + 10, y + 20, 420, 300, this);                          //在打印页绘制图表图片
```

秘笈心法

心法领悟 110：将图表保存为图片的另一种方法。

除了使用本实例中提供的方法将图表保存为图片，还可以使用 ChartUtilities 类的 SaveChartAsJPEG()方法，但是使用该方法传递存储图片的对象是 File 对象。

实例 111 打印带折线图表的报表

光盘位置：光盘\MR\111　　高级　　趣味指数：★★★★

实例说明

本实例实现了打印带折线图表的报表。运行程序，单击窗体上的“预览报表”按钮，将显示带折线图表的报表预览图，效果如图 5.21 所示，单击“打印报表”按钮，将打印该报表。

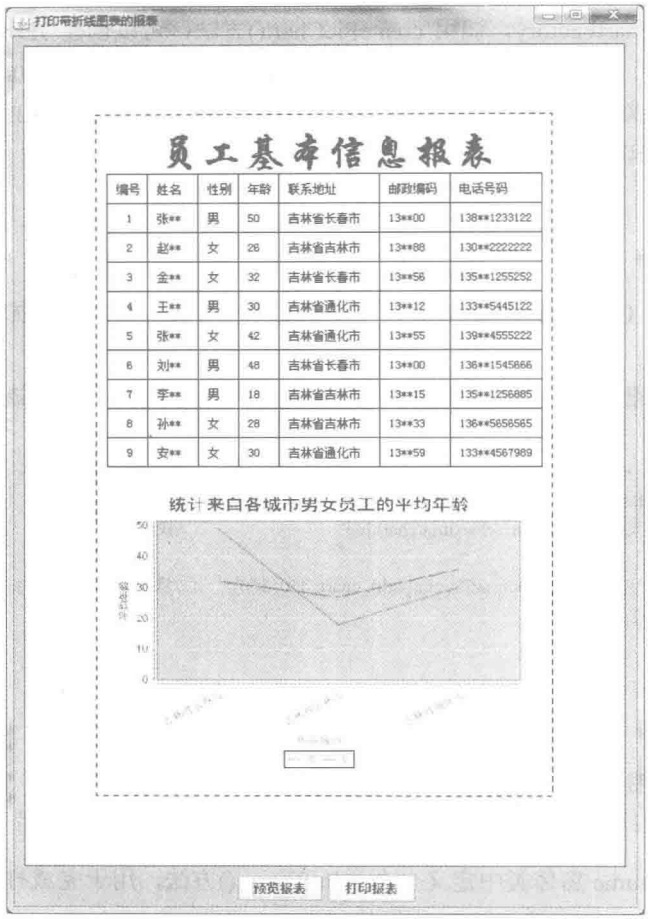


图 5.21　带折线图表的报表预览效果

关键技术

本实例使用图表工厂类 ChartFactory，调用 createLineChart()方法创建了一个折线图表对象，然后使用 ChartUtilities 类的 writeChartAsJPEG()方法将折线图表对象保存为 JPG 图片，最后将折线图表的 JPG 图片对象绘制到打印页，从而实现了打印带折线图表的报表。

（1）使用图表工厂类 ChartFactory，调用 createLineChart()方法，可以创建 JFreeChart 对象，该对象是一个折线图表对象，有关图表工厂类 ChartFactory 的 createLineChart()方法请查看实例 107。

（2）使用 ChartUtilities 类的 writeChartAsJPEG()方法，将折线图表对象保存为 JPG 格式的图片，ChartUtilities 类的 writeChartAsJPEG()方法请查看实例 109。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 QueryResultSet 类，用于加载数据库驱动、连接数据库以及获得数据库表中的数据信息。

（3）在项目中创建一个继承自 JFrame 类的 PrintLineReportFrame 窗体类，在该类的构造方法中，添加将图表对象保存为图片的代码：

```
CategoryDataset dataset = createDataset();                    //创建数据集对象
JFreeChart chart = createChart(dataset);                      //创建图表对象
```

```
String path = System.getProperty("user.dir") + "/src/chartImg/chart.jpg";          //图表的存储位置
try {
    ChartUtilities.writeChartAsJPEG(new FileOutputStream(path), chart, 450, 360);  //将图表存储为图片
} catch (FileNotFoundException e2) {
    e2.printStackTrace();
} catch (IOException e2) {
    e2.printStackTrace();
}
```

说明：上面代码将图表图片存储到项目的 src 文件夹的子文件夹 chartImg 中，图片文件的名称为 chart.jpg，由于上述代码放在构造方法中，所以在程序运行时就将图表保存为图片了。

（4）在 PrintLineReportFrame 窗体类中定义一个 printPicture()方法，用于完成打印内容的绘制，该方法中用于绘制图表图片的代码如下：

```
URL imgUrl = PrintLineReportFrame.class.getResource("/chartImg/chart.jpg");   //获取图片资源的路径
Image img = Toolkit.getDefaultToolkit().getImage(imgUrl);                     //获取图表图片的图像对象
g2.drawImage(img, x + 10, y + 20, 420, 300, this);                            //在打印页绘制图表图片
```

秘笈心法

心法领悟 111：将图表保存的图片放到项目路径中。

由于项目在计算机中的存储位置可以发生变化，因此为了保证图表保存的图片放到项目的某个路径中，可以使用 System 类的静态方法 getProperty()，并为其传递实现字符串 user.dir，就可以获得 Java 应用程序的项目路径，如果需要使用 src 文件夹，只需要连接字符串/src，即 System.getProperty("user.dir")+"/src"。

实例 112 导出报表到 Excel 表格

光盘位置：光盘\MR\112　　初级　　趣味指数：★★★

实例说明

本实例实现了将报表导出到 Excel 表格的功能。运行程序，将在窗体上显示数据库表中的数据信息，效果如图 5.22 所示，单击窗体上的“导出到 Excel”按钮，可以将报表中的数据导出到 C 盘名为 report.xls 的 Excel 文件中。

图 5.22　导出报表到 Excel 表格

关键技术

本实例通过 POI 实现了将报表导出到 Excel 表格的功能，其中，关键技术是工作簿、工作表、工作表中的行和单元格的创建，以及向单元格中添加数据值，使用 POI 实现这些功能的代码如下：

```
HSSFWorkbook workbook = new HSSFWorkbook();                    //创建工作簿对象
HSSFSheet sheet = workbook.createSheet("ReportToExcel");       //创建名称为 ReportToExcel 的工作表
HSSFRow row = sheet.createRow(0);                              //在工作表中创建行，其行索引为 0
```

```
HSSFCell cell = null;                                    //声明单元格
//导出的报表在 Excel 中的标题
for (int i = 0; i < title.length; i++) {
    cell = row.createCell((short) i);                    //创建单元格
    cell.setCellType(HSSFCell.CELL_TYPE_STRING);         //设置单元格类型
    cell.setCellValue(title[i]);                         //设置单元格的内容
}
```

说明：上面代码中的 title 是 String 类型的数组，用于存储导出报表的标题。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 QueryResultSet 类，用于加载数据库驱动、连接数据库以及获得数据库表中的数据信息。

（3）在项目中创建一个继承自 JFrame 类的 ReportToExcelFrame 窗体类，在该类中创建 writeExcel()方法，用于将报表导出到 Excel 文件中，该方法的代码如下：

```
@SuppressWarnings("deprecation")
public void writeExcel(String filename) throws IOException {
    HSSFWorkbook workbook = new HSSFWorkbook();                    //创建工作簿对象
    HSSFSheet sheet = workbook.createSheet("ReportToExcel");       //创建名称为 ReportToExcel 的工作表
    HSSFRow row = sheet.createRow(0);                              //在工作表中创建行，其行索引为 0
    HSSFCell cell = null;                                          //声明单元格
    //导出的报表在 Excel 中的标题
    for (int i = 0; i < title.length; i++) {
        cell = row.createCell((short) i);                          //创建单元格
        cell.setCellType(HSSFCell.CELL_TYPE_STRING);               //设置单元格类型
        cell.setCellValue(title[i]);                               //设置单元格的内容
    }
    //导出的报表在 Excel 中的内容
    ResultSet rs = QueryResultSet.gainRecord();                    //获得数据库表的结果集对象
    int rowIndex = 1;                                              //单元格内容的行索引值
    try {
        while (rs.next()) {                                        //遍历结果集
            row = sheet.createRow(rowIndex);
            for (int i = 0; i < title.length; i++) {
                String cellContent = "";                           //定义存放单元格内容的变量
                if (i == 0 || i == 3) {
                    cellContent = String.valueOf(rs.getInt(i + 1));//从记录集获得单元格内容
                } else {
                    cellContent = rs.getString(i + 1);             //从记录集获得单元格内容
                }
                cell = row.createCell((short) i);                  //创建单元格对象
                cell.setCellType(HSSFCell.CELL_TYPE_STRING);       //设置单元格类型
                cell.setCellValue(cellContent);                    //指定单元格的值
            }
            rowIndex++;                                            //调整单元格内容的行索引值
        }
        FileOutputStream fout = new FileOutputStream(filename);    //创建导出 Excel 的输出流
        workbook.write(fout);                                      //将工作簿写入输出流
        fout.flush();                                              //刷新输出流并强制写出所有缓冲的输出字节
        fout.close();                                              //关闭输出流对象
        JOptionPane.showMessageDialog(null, "已经成功地将报表\n 导出到“c:/report.xls”中。");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

说明：在使用本实例时，需要将 POI 的 poi-3.2-FINAL-20081019.jar 包配置到类库路径中，否则将无法开发和运行本实例。

秘笈心法

心法领悟 112：设置导出到 Excel 表格中的数据类型。

使用 HSSFCell 类的 setCellType()方法，可以设置保存到 Excel 表格中的数据类型。数据类型可以通过 HSSFCell 类的静态属性获得，如 HSSFCell.CELL_TYPE_STRING 代表字符串类型，HSSFCell.CELL_TYPE_BOOLEAN 代表布尔类型，HSSFCell.CELL_TYPE_NUMERIC 代表数值类型等，然后再使用 setCellValue()方法设置保存的值，从而保证了数据类型的对应。

实例 113　导出报表到 PDF 文档

光盘位置：光盘\MR\113　　初级　　趣味指数：★★★

实例说明

本实例实现了将报表导出到 PDF 文档的功能。运行程序，将在窗体上显示数据库表中的数据信息，效果如图 5.23 所示，单击窗体上的“导出为 PDF”按钮，可以将报表中的数据导出到 C 盘名为 report.pdf 的 PDF 文档中。

图 5.23　导出报表到 PDF 文档

关键技术

本实例通过 iText 实现了将报表导出到 PDF 文档的功能，其中，关键技术是文档、PDF 输出流、PDF 单元格、表格的列数、总的宽度以及向单元格添加信息，使用 iText 实现这些功能的主要代码如下：

```
Document document = new Document(PageSize.A4, 71.4f, 71.4f, 71.4f, 71.4f);          //定义文档以 A4 纸张的格式打印
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\report.pdf"));                                    //设置输出的位置及文件名
document.open();                                                                    //打开文档
BaseFont bfChinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                                     //定义基本字体
Font titleFont = new Font(bfChinese, 12, Font.BOLD);                                //定义加粗字体，用于设置表格标题的字体
Font contentFont = new Font(bfChinese, 12);                                         //定义普通字体
PdfPCell cell = null;                                                               //定义 PDF 单元格
PdfPTable table = new PdfPTable(new float[] { 40.9f, 40.9f, 40.9f,
        40.9f, 79.1f, 40.9f, 79.1f });                                              //建立一个 7 列的 PDF 表格
table.setTotalWidth(460);                                                           //设置表格的宽度为 460
table.setLockedWidth(true);                                                         //锁定表格的宽度
for (int i = 0; i < title.length; i++) {
    cell = new PdfPCell(new Paragraph(title[i], titleFont));                        //创建单元格对象
    cell.setHorizontalAlignment(Element.ALIGN_CENTER);                              //单元格内容水平居中
    cell.setVerticalAlignment(Element.ALIGN_MIDDLE);                                //单元格内容垂直居中
    cell.setFixedHeight(20.0f);                                                     //设置单元格的高度
    table.addCell(cell);                                                            //为表格添加单元格
}
```

说明：上面代码中的 title 是 String 类型的数组，用于存储导出报表的标题。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 QueryResultSet 类，用于加载数据库驱动、连接数据库以及获得数据库表中的数据信息。

（3）在项目中创建一个继承自 JFrame 类的 ReportToPdfFrame 窗体类，在该类中创建 toPdf()方法，用于将报表导出到 PDF 文档中，该方法的代码如下：

```
public void toPdf() {
    try {
        Document document = new Document(PageSize.A4, 71.4f, 71.4f, 71.4f, 71.4f);          //定义文档以 A4 纸张的格式打印
        PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\report.pdf"));   //设置输出的位置及文件名
        document.open();                                                                    //打开文档
        BaseFont bfChinese = BaseFont.createFont("STSong-Light",
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                                     //定义基本字体
        Font titleFont = new Font(bfChinese, 12, Font.BOLD);                                //定义加粗字体，用于设置表格标题的字体
        Font contentFont = new Font(bfChinese, 12);                                         //定义普通字体
        PdfPCell cell = null;                                                               //定义 PDF 单元格
        PdfPTable table = new PdfPTable(new float[] { 40.9f, 40.9f, 40.9f,
                40.9f, 79.1f, 40.9f, 79.1f });                                              //建立一个 7 列的 PDF 表格
        table.setTotalWidth(460);                                                           //设置表格的宽度为 460
        table.setLockedWidth(true);                                                         //锁定表格的宽度
        for (int i = 0; i < title.length; i++) {
            cell = new PdfPCell(new Paragraph(title[i], titleFont));                        //创建单元格对象
            cell.setHorizontalAlignment(Element.ALIGN_CENTER);                              //单元格内容水平居中
            cell.setVerticalAlignment(Element.ALIGN_MIDDLE);                                //单元格内容垂直居中
            cell.setFixedHeight(20.0f);                                                     //设置单元格的高度
            table.addCell(cell);                                                            //为表格添加单元格
        }
        ResultSet rs = QueryResultSet.gainRecord();                                         //获得数据库表的结果集对象
        while (rs.next()) {                                                                 //遍历结果集
            for (int i = 0; i < title.length; i++) {
                String cellContent = "";                                                    //定义存放单元格内容的变量
                if (i == 0 || i == 3) {
                    cellContent = String.valueOf(rs.getInt(i + 1));                         //从记录集获得单元格内容
                } else {
                    cellContent = rs.getString(i + 1);                                      //从记录集获得单元格内容
                }
                cell = new PdfPCell(new Paragraph(cellContent, contentFont));               //创建单元格对象
                if (i == 6) {
                    cell.setBorderWidthRight(0.01f);                                        //单元格右边框的宽度
                } else {
                    cell.setBorderWidthRight(0f);                                           //单元格右边框的宽度
                }
                cell.setBorderWidthTop(0f);                                                 //单元格上边框的宽度
                cell.setBorderWidthBottom(0.01f);                                           //单元格下边框的宽度
                cell.setFixedHeight(20.0f);                                                 //单元格的高度
                table.addCell(cell);                                                        //为表格添加单元格
            }
        }
        titleFont = new Font(bfChinese, 36, Font.BOLD);                                     //定义加粗字体，用于设置标题的字体
        Paragraph pdfTitle = new Paragraph("员工基本信息报表\n\n", titleFont);              //创建用于指定标题的 Paragraph 对象
        pdfTitle.setAlignment(Element.ALIGN_CENTER);                                        //居中 Paragraph 对象
        document.add(pdfTitle);                                                             //为 PDF 文档添加标题
        document.add(table);                                                                //为 PDF 文档添加表格
        document.close();                                                                   //关闭文档
        JOptionPane.showMessageDialog(null, "已经成功地将报表\n 导出到“c:/report.pdf”中。");
    } catch (Exception e2) {
```

```
            e2.printStackTrace();
        }
    }
```

说明：在使用本实例时，需要将 iText 的 iText-2.1.3.jar、iText-toolbox-2.1.7.jar 和 itextasian-1.5.2.jar 这 3 个 jar 包配置到类库路径中，否则将无法开发和运行本实例。

秘笈心法

心法领悟 113：确保数据成功导出到 PDF 表格中。

在导出数据到 PDF 表格时，如果没有合并的单元格，除了表格的第一行，应该保证以后导出的每行数据的列数都与第一行相同，对于合并的列也要保证合并前的列数与第一行相同，否则不会将数据导出到 PDF 表格中。

实例 114 批量打印条形码

光盘位置：光盘\MR\114　　高级　　趣味指数：★★★

实例说明

条形码广泛地应用在商品的外包装或标签上，这样可以方便在商场和超市中通过扫描仪对条形码进行扫描，计算出商品的价钱，从而提高了工作人员的工作效率，为此本实例实现了条形码的批量打印。运行程序，在窗体下方的文本框中输入条形码编号，即可在窗体上绘制出各种类型的条形码，效果如图 5.24 所示。

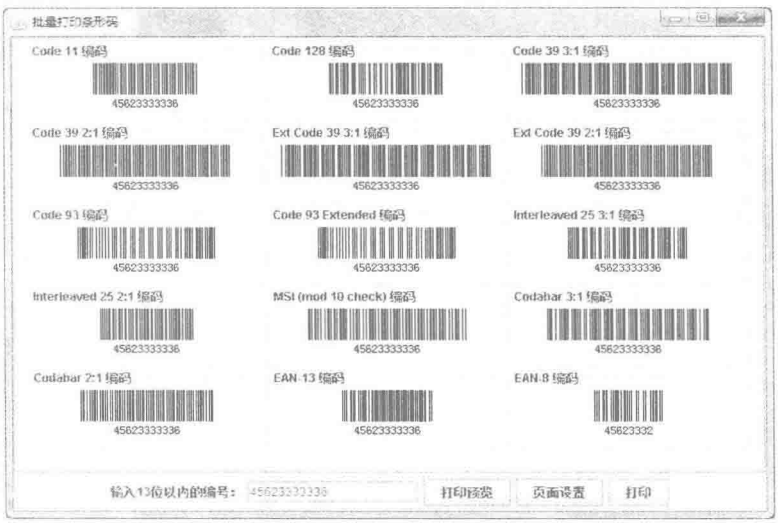


图 5.24　批量打印条形码

关键技术

本实例主要是通过 JBarcodeBean 组件实现条形码的生成，并可以根据输入的条形码编号生成各种类型的条形码，从而实现了条形码的批量打印功能。

（1）使用 JBarcodeBean 可以创建条形码对象，然后使用该对象调用 setBorder()、setShowText()和 setCodeType()方法，完成各种类型条形码的创建，例如要创建 Code11 类型编码的条形码，则可以用如下代码实现：

```
final JBarcodeBean barcodeBean = new JBarcodeBean();                    //创建条形码对象
barcodeBean.setBorder(new TitledBorder(null, "Code 11 编码",
        TitledBorder.DEFAULT_JUSTIFICATION,
        TitledBorder.DEFAULT_POSITION, null, null));                    //设置条形码的边框
barcodeBean.setShowText(true);                                          //设置显示条形码上的编码
barcodeBean.setCodeType(new jbarcodebean.Code11());                     //设置条形码的类型
```

说明：上面代码创建了一个 Code11 类型编码的条形码，其中 setBorder()方法用于指定边框；setShowText()方法用于指定是否显示条形码的编号；setCodeType()方法用于指定条形码的类型编码。

（2）遍历父容器中的所有控件，并根据输入的条形码编号在这些控件上生成各种类型编码的条形码，代码如下：

```
String text = textField.getText();                                      //获取文本框内容
Component[] codes = codePanel.getComponents();                          //获取面板所有组件
for (Component component : codes) {                                     //遍历组件数组
    if (component instanceof JBarcodeBean) {                            //如果组件是条形码组件
        JBarcodeBean bean = (JBarcodeBean) component;                   //强制转换为条形码对象
        bean.setCode(text);                                             //设置组件的编码
    }
}
codePanel.repaint();                                                    //重新绘制绘图上下文对象
```

说明：为了能够在文本框内容更新时动态生成所输入编号的条形码，需要将上述代码放到文本框的内容更新事件中，即添加 CaretListener 监听器，实现 caretUpdate()方法，并将上述代码放到该方法中。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JPanel 类的 CodePanel 面板类，用于显示生成的条形码，以及编写实现打印等功能的方法，该类的关键代码如下：

```
public class CodePanel extends JPanel {
    private PageFormat format;                                          //定义页面格式
    private PrinterJob job = PrinterJob.getPrinterJob();                //获得打印任务
    public CodePanel() {
        //省略了非关键代码
        final JBarcodeBean barcodeBean_9 = new JBarcodeBean();          //创建条形码对象
        barcodeBean_9.setBorder(new TitledBorder(null, "Code 11 编码",
                TitledBorder.DEFAULT_JUSTIFICATION,
                TitledBorder.DEFAULT_POSITION, null, null));            //设置条形码的边框
        barcodeBean_9.setShowText(true);                                //设置显示条形码上的编码
        barcodeBean_9.setCodeType(new jbarcodebean.Code11());           //设置条形码的类型
        add(barcodeBean_9);
        //省略了部分代码
    }
    /**
     * 页面设置的方法
     */
    public void pageSetup() {
        if (job != null) {
            format = job.pageDialog(format);                            //打开页面设置对话框
            repaint();                                                  //重新绘制界面
        }
    }
    /**
     * 执行打印操作的方法
     */
    public void doPrint() {
        if (job != null) {
```

```
                try {
                    if (job.printDialog()) {                          //如果用户确认打印
                        job.setJobName("批量条形码打印");              //设置打印任务名称
                        Printable page = PreviewDialog.getPage();     //获取页面对象
                        job.setPrintable(page);                       //设置打印页面
                        job.print();                                  //执行打印
                    }
                } catch (PrinterException e) {
                    e.printStackTrace();
                }
            }
        }
        public PageFormat getFormat() {
            return format;                                            //获得页面格式
        }
    }
```

（3）在项目中创建一个继承 JFrame 类的 PrintCodingFrame 窗体类，在该类中为“输入 13 位以内的编号：”标签右侧的文本框添加内容更新事件，用于监听文本框中内容的变化，并根据输入的编号产生条形码，其事件代码如下：

```
textField.addCaretListener(new CaretListener() {
    public void caretUpdate(final CaretEvent arg0) {
        String text = textField.getText();                            //获取文本框内容
        Component[] codes = codePanel.getComponents();                //获取面板所有组件
        for (Component component : codes) {                           //遍历组件数组
            if (component instanceof JBarcodeBean) {                  //如果组件是条形码组件
                JBarcodeBean bean = (JBarcodeBean) component;         //强制转换为条形码对象
                bean.setCode(text);                                   //设置组件的编码
            }
        }
        codePanel.repaint();                                          //重新绘制条形码
    }
});
```

注意：在使用本实例时，需要将项目需要的 JBarcodeBean 包配置到类库路径中，否则将无法进行该实例的开发。

秘笈心法

心法领悟 114：批量打印类型相同、编码不同的条形码。

如果需要批量打印类型相同、编码不同的条形码，只需要将窗体上条形码对象指定为相同类型编码，然后将文本框中输入的内容设置为条形码的起始编号，并通过循环为每个条形码设置值，即可以实现批量打印类型相同而编码不同的条形码。

实例 115 相册特效打印程序

高级

光盘位置：光盘\MR\115

趣味指数：★★★★

实例说明

本实例实现了相册特效的打印程序。运行程序，单击窗体上的“选择文件”按钮选择一幅图片，然后单击边框样式，将显示相册特效的图片，效果如图 5.25 所示，并且还可以通过“页面设置”按钮进行页面设置，单击“开始打印”按钮，可以打印相册特效图片。

图 5.25　相册特效打印程序的预览效果

关键技术

本实例使用透明背景的图片（该图片中心是空白的，没有任何内容，只是图片的边缘绘制有图像）作为相册特效图片的边框，然后将其绘制到其他图片之上，就仿佛为图片添加了相册边框，其中绘制相册效果边框的代码如下：

```
g2.drawImage(src, x, y, (int) imgWidth, (int) imgHeight, this);                    //绘制正常图像，如图片、照片等
/********************** 绘制边框 ************************/
if (!border.equals("")) {
    ImageIcon icon = SwingResourceManager.getIcon(MainFrame.class, "/com/zzk/" + border + ".png");    //获取 ImageIcon 对象
    Image borderImg = icon.getImage();                                             //获取用于绘制边框的 Image 对象
    g2.drawImage(borderImg, x, y, (int) imgWidth, (int) imgHeight, this);          //绘制边框
}
/*****************************************************/
```

说明：上面代码中的 x、y、imgWidth 和 imgHeight 分别代表打印页中可打印区域的 x 坐标、y 坐标、宽度和高度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JLabel 类的 ImgLabel 标签类用于显示相册效果的边框样式，该类的代码如下：

```
public class ImgLabel extends JLabel {
    public ImgLabel() {
        super();                                                                   //调用超类的构造方法
    }
```

```
    @Override
    public void paint(Graphics g) {
        try {
            int width = this.getWidth();                                    //获取图片宽度
            int height = this.getHeight();                                  //获取图片高度
            ImageIcon icon = (ImageIcon) getIcon();                         //获取 ImageIcon 对象
            if (icon != null) {                                             //图片不为空
                g.drawImage(icon.getImage(), 0, 0, width, height, this);    //绘制图片
            }
        } catch (Exception e) {
            e.printStackTrace();                                            //输出异常信息
        }
    }
}
```

（3）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类，在该类中定义一个 printPicture()方法，用于绘制原图片或照片，以及在图片或照片上绘制相册效果的边框，该方法的代码如下：

```
public void printPicture(Graphics graphics, PageFormat pageFormat, int pageIndex) {
    int x = (int) pageFormat.getImageableX();                               //获取可打印区域的 x 坐标
    int y = (int) pageFormat.getImageableY();                               //获取可打印区域的 y 坐标
    Graphics2D g2 = (Graphics2D) graphics;
    if (imgFile != null && isPreview) {
        try {
            src = ImageIO.read(imgFile);                                    //构造 Image 对象
        } catch (IOException e) {
            e.printStackTrace();
        }
        double imgWidth = src.getWidth(this);                               //获取图片的宽
        double imgHeight = src.getHeight(this);                             //获取图片的高
        double percent = imgWidth / imgHeight;                              //宽高比例
        int mw = (int) pf.getWidth() - x * 2;                               //计算可打印区域的宽
        int mh = (int) pf.getHeight() - y * 2;                              //计算可打印区域的高
        if (imgWidth > mw) {                                                //如果宽度大于可打印区域
            imgWidth = mw;                                                  //宽度等于可打印区域的宽度
            imgHeight = mw * percent;                                       //通过百分比计算高度
        }
        if (imgHeight > mh) {                                               //如果高度大于可打印区域
            imgHeight = mh;                                                 //高度等于可打印区域的高度
            imgWidth = mh * percent;                                        //通过百分比计算宽度
        }
        g2.drawImage(src, x, y, (int) imgWidth, (int) imgHeight, this);     //绘制正常图像，如图片、照片等
        /********************* 绘制边框 ************************/
        if (!border.equals("")) {
            ImageIcon icon = SwingResourceManager.getIcon(MainFrame.class,
                    "/com/zzk/" + border + ".png");                         //获取 ImageIcon 对象
            Image borderImg = icon.getImage();                              //获取用于绘制边框的 Image 对象
            g2.drawImage(borderImg, x, y, (int) imgWidth, (int) imgHeight, this);   //绘制边框
        }
        /*****************************************************/
    }
    isPreview = false;                                                      //设置不可以打印
}
```

秘笈心法

心法领悟 115：在打印的图片上绘制商标。

由于在 Java 的绘图上下文中绘制图片后，还可以在已经绘制有图片的绘图上下文中重新绘制新的内容，并且新的内容可以覆盖已经绘制的内容，但是不影响先绘制图片的其他位置，因此，在打印页中绘制完图片后，如果需要绘制商标等信息，可以在图片的适当位置绘制商标图片，完成在打印的图片上绘制商标的操作。

实例 116 镜面效果文本打印

光盘位置：光盘\MR\116

中级

趣味指数：★★★

实例说明

本实例实现了镜面效果的文本打印。运行程序，效果如图 5.26 所示，然后单击窗体上的“镜面效果/还原”按钮，将水平翻转文字，就像在镜子里的成像效果，如图 5.27 所示，此时如果再次单击“镜面效果/还原”按钮，又将恢复为图 5.26 的效果。

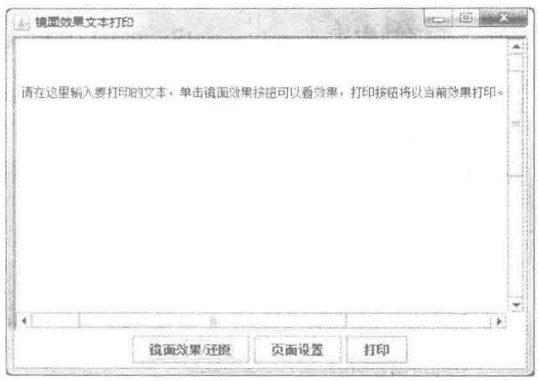


图 5.26 实现镜面效果之前的内容

图 5.27 实现镜面效果之后的内容

关键技术

本实例通过将文本域绘制到缓冲图像对象上，然后使用 Java 的绘图技术水平翻转缓冲图像对象，从而实现了文本的镜面效果打印。

（1）将文本域绘制到缓冲图像对象上，是通过从 JComponent 类继承的 print()方法实现的，JComponent 类的 print()方法定义如下：

```
public void print(Graphics g)
```

参数说明

g：绘图上下文对象。

说明：在本实例中，参数 g 是缓冲图像对象的绘图上下文对象，也就是通过 BufferedImage 类的 createGraphics()方法获得的绘图上下文对象。

（2）使用 Graphics 类的 drawImage()方法实现图像的翻转，从而实现镜面效果文本的绘制。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JPanel 类，并实现 Printable 接口的 PrintPanel 打印面板类，在该类中实现 Printable 接口中的 print()方法，在该方法中完成打印内容的绘制，其中实现镜面效果文本的代码如下：

```
if (pageIndex < 1 && textArea != null && reverse) {                    //如果当前打印页数小于 1 并且开启镜面效果
    BufferedImage image = new BufferedImage(pwidth - px, pheight - py,
            BufferedImage.TYPE_INT_RGB);                               //创建缓冲图像对象
    Graphics2D graphics = image.createGraphics();                      //获取图片对象的绘图上下文
    graphics.setColor(Color.WHITE);                                    //设置前景色为白色
    graphics.fillRect(0, 0, image.getWidth(), image.getHeight());      //使用白色填充界面
    graphics.setColor(Color.BLACK);                                    //设置前景色为黑色
    Font font = textArea.getFont();                                    //获取文本域组件的字体对象
```

```
        graphics.setFont(font);                                    //把字体对象设置给图片的绘图上下文
        textArea.print(graphics);                                  //把文本域界面绘制到缓冲图像对象上
        image.flush();                                             //刷新图片绘图缓冲区
        g.drawImage(image, px, py, pwidth, pheight, image.getWidth(), 0, 0,
                image.getHeight(), this);                          //反向绘制打印内容，实现镜面效果
        return Printable.PAGE_EXISTS;                              //返回可打印标识
    } else {                                                       //否则
        return Printable.NO_SUCH_PAGE;                             //返回不支持打印标识
    }
```

（3）在项目中创建一个继承 JFrame 类的 PrintReverseTextFrame 窗体类，该窗体类中的“镜面效果/还原”按钮事件用于实现原文本与镜面效果文本的转换，其代码如下：

```
reverseBtn.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        boolean reverse = reverseBtn.isSelected();                 //获得按钮的按下状态
        printPanel.setReverse(reverse);                            //重新绘制打印面板
    }
});
```

说明：上面按钮事件代码中的 setReverse()方法用于设置是否反转文本中的内容，如果反转，则隐藏显示文本的文本域对象，并绘制反转的文本；否则，显示文本域对象，恢复正常的文本显示。

秘笈心法

心法领悟 116：保留原有图片的镜面效果。

在绘图上下文中绘制原有图片，然后在原有图片的右侧绘制对原有图片进行水平翻转的图片，或在原有图片的下方绘制对原有图片进行垂直翻转的图片，都可以实现保留原有图片的镜面效果。

实例 117 透明的打印预览对话框

光盘位置：光盘\MR\117　　高级　　趣味指数：★★★★★

实例说明

本实例实现了一个透明的打印预览对话框。运行程序，在窗体上可以透出后面的内容，单击窗体上的“选择文件”按钮选择图片，然后单击“打印预览”按钮进行页面打印预览，效果如图 5.28 所示。

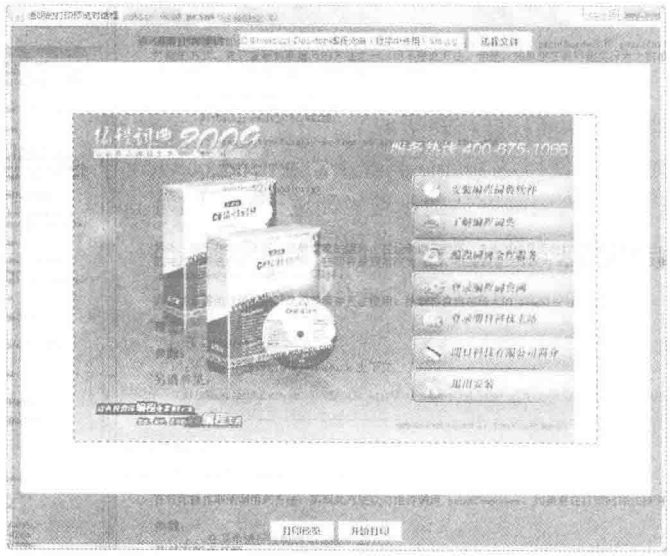


图 5.28　透明的打印预览对话框效果

关键技术

本实例通过使用 AWTUtilities 类的 setWindowOpacity()方法改变窗体的透明度，使窗体透明，从而实现了透明的打印预览对话框。

从 JDK 6.0 开始，可以使用 AWTUtilities 类的 setWindowOpacity()方法设置窗体透明度，从而实现窗体的透明效果，该方法的定义如下：

```
public static void setWindowOpacity(Window win, float opacity)
```

参数说明

❶ win：需要调整透明度的窗体控件。

❷ opacity：窗体的透明度，为 1.0f 时不透明；为 0.0f 时完全透明；在 0.0f~1.0f 之间为不完全透明状态。

说明：在使用 AWTUtilities 类的 setWindowOpacity()方法时，为其传递的小数值必须在 0.0f~1.0f 之间，并且在小数值后要加一个小写的 f 或大写的 F，表示这个小数是一个 float 型的数值。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类，并实现 Printable 接口的 MainFrame 窗体类，在该类的 main()主方法中添加实现窗体透明的效果，代码如下：

```
public static void main(String args[]) {
    MainFrame frame = new MainFrame();                  //创建窗体对象
    AWTUtilities.setWindowOpacity(frame, 0.8f);         //设置窗体 80%透明
    frame.setVisible(true);                             //显示窗体
}
```

（3）在 MainFrame 窗体类中实现 Printable 接口中的 print()方法，在该方法中完成打印内容的绘制，代码如下：

```
public int print(Graphics graphics, PageFormat pageFormat, int pageIndex) throws PrinterException {
    printPicture(graphics, pageFormat, pageIndex);      //绘制打印内容
    return Printable.PAGE_EXISTS;                       //返回 PAGE_EXISTS
}
```

说明：上面代码中的 printPicture()方法是进行打印页内容绘制的方法，由于篇幅原因，这里没有给出，具体实现可以查看源程序代码。

秘笈心法

心法领悟 117：显示和隐藏窗体的标题栏和边框。

Java 中的 JFrame 窗体类，可以使用从 Frame 类继承的 setUndecorated()方法，设置是否显示和隐藏窗体的标题栏和边框，如果其参数为 true，则隐藏窗体的标题栏和边框；如果为 false，则显示窗体的标题栏和边框，并且该方法必须要在使用 setVisible()方法显示窗体之前使用。

第 6 章

管理图像文件

- 图像的修改与保存
- 图片在数据库中的存取
- 其他应用

6.1 图像的修改与保存

实例 118	保存图片文件 光盘位置：光盘\MR\118	初级 趣味指数：★★

实例说明

本实例使用 Java 程序实现了图片文件的保存。运行程序，将在窗体上显示项目中给出的图片，效果如图 6.1 所示，单击窗体上的“保存图片”按钮，就会将该图片以名称 image.jpg 保存到 C 盘根目录中。

图 6.1 保存图片文件窗体

关键技术

本实例主要是通过 ImageIO 类的 write()方法，将图片文件的缓冲图像对象以指定的格式保存到由 File 对象指定的磁盘中。

使用 ImageIO 类的 write()方法，可以将图片写入磁盘中，该方法的定义如下：

```
public static boolean write(RenderedImage im, String formatName, File output) throws IOException
```

参数说明

❶ im：要写入磁盘的 RenderedImage 接口，BufferedImage 类实现了该接口，可以使用 BufferedImage 类创建 RenderedImage 对象。

❷ formatName：要写入磁盘的图片文件格式，如.jpg、.png 等。

❸ output：文件保存位置的 File 对象。

❹ 返回值：执行成功返回 true，否则返回 false。

❺ 异常处理：使用该方法时需要处理 IO 异常。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SaveImageFrame 窗体类。

（3）在 SaveImageFrame 窗体类中，为“保存图片”按钮的事件添加将图片保存到磁盘的代码。其事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        BufferedImage bufferImage = new BufferedImage(img
                .getWidth(SaveImageFrame.this), img
                .getHeight(SaveImageFrame.this),
                BufferedImage.TYPE_INT_RGB);                    //创建缓冲图像对象
        Graphics g = bufferImage.getGraphics();                 //获得绘图上下文对象
```

```
            g.drawImage(img, 0, 0, null);                                    //在缓冲图像上绘制图像
            try {
                ImageIO.write(bufferImage, "jpg", new File("c:/image.jpg"));  //将缓冲图像保存到磁盘
                JOptionPane.showMessageDialog(null,
                        "已将图片 image.jpg\n 成功地保存到 c:盘");             //显示提示信息
            } catch (IOException e1) {
                e1.printStackTrace();
            }
        }
    });
```

注意：上面代码中的 img 是一个所要保存图片的 Image 对象，本实例就是通过该对象创建了 BufferedImage 对象，从而实现了将图片保存到磁盘的功能。

秘笈心法

心法领悟 118：判断图片保存成功的方法。

由于 ImageIO 类的 write()方法在文件写入磁盘成功时返回值为 true，否则返回 false，因此为了判断图片是否保存成功，可以通过 if 语句判断 ImageIO 类，调用 write()方法的返回值，从而给出相应的提示。

实例 119　修改图片文件名

光盘位置：光盘\MR\119　　初级　　趣味指数：★

实例说明

本实例演示了如何利用 Java 程序实现修改图片文件名的功能。运行程序，单击窗体上的“选择图片”按钮并选择图片，将在窗体上显示所选择的图片，效果如图 6.2 所示，然后在窗体下方的文本框中输入新的文件名，单击“重命名图片”按钮，将完成原图片文件名的修改。

图 6.2　修改图片文件名窗体

关键技术

本实例使用 File 类的 renameTo()方法实现了修改图片文件名的操作，File 类的 renameTo()方法的定义如下：

public boolean renameTo(File dest)

参数说明

❶ dest：修改后新文件的路径和文件名。

❷ 返回值：修改成功返回 true，否则返回 false。

注意：在使用该方法修改文件名时，如果新文件名与原文件名相同，并且在同一路径中，修改就会失败。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 RenameImageFrame 窗体类。

（3）由于本实例只是对图片文件名进行修改，而不需要改变图片的存储位置，并且所输入的新文件名可以有扩展名，也可以没有扩展名，因此可以方便用户的操作，窗体类 RenameImageFrame 中的“重命名图片”按钮实现了这一功能。其事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        if (path != null && fileName != null) {
            String newName = tf_newFileName.getText().trim();                    //获得新输入的文件名
            if (newName.indexOf(":") >= 0 || newName.indexOf("\\") >= 0) {       //判断文件名是否包含路径
                JOptionPane.showMessageDialog(null, "请直接输入新文件名。\n 不需要指定路径。");
                return;
            } else if (newName.equals("")) {                                     //如果没有输入文件名则进行提示
                JOptionPane.showMessageDialog(null, "请输入新文件名。");
                return;
            } else {
                if (newName.indexOf(".") > 0) {
                    imgFile.renameTo(new File(path + "\\" + newName));           //新文件名带扩展名
                } else {
                    imgFile.renameTo(new File(path + "\\" + newName + fileName.substring(fileName.indexOf("."))));  //新文件名不带扩展名
                }
            }
            JOptionPane.showMessageDialog(null, "修改成功。");
        }
    }
});
```

注意：在使用本实例时，需将项目需要的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 119：灵活地修改各种类型文件的名称。

本实例只是对选择的图片文件的名称进行修改，如果希望能够更加灵活地修改各种类型文件的名称，可以将指定路径中的所有文件通过列表框进行显示，然后在列表框中选择要修改名称的文件，再通过文本框等文本控件进行修改，即可实现灵活地修改各种类型文件的名称。

实例 120 缩放图片并保存

光盘位置：光盘\MR\120　　初级　　趣味指数：★★

实例说明

本实例演示了如何利用 Java 的绘图技术实现图片的缩放，并将缩放的图片保存到磁盘中。运行程序，效果如图 6.3 所示，单击窗体上的“放大”和“缩小”按钮，可以放大和缩小窗体上的图片；单击“保存”按钮，可以将缩放后的图片保存到磁盘中。

图 6.3　缩放图片并保存

关键技术

本实例通过 int 型成员变量 newW 和 newH 保存图片缩放后的宽度和高度，然后在保存时根据这两个成员变量创建指定大小的缓冲图像对象，并在其上绘制显示有图片的图像对象，再将该缓冲图像对象写入磁盘，从而实现缩放图片并保存的功能。

（1）定义两个 int 型的成员变量，用于保存图片缩放后的宽度和高度，代码如下：

```
private int newW, newH;                                    //用于存储图片缩放后的宽度和高度
```

说明：上面代码中的 newW 用于存储图片缩放后的宽度；newH 用于存储图片缩放后的高度。

（2）在保存图片时，使用成员变量 newW 和 newH 创建指定大小的缓冲图像对象，并在其绘图上下文上绘制指定大小的图像，完成缓冲图像的创建，代码如下：

```
BufferedImage bufferImage = new BufferedImage(newW, newH,
        BufferedImage.TYPE_INT_RGB);                       //创建缓冲图像对象
Graphics g = bufferImage.getGraphics();                    //获得绘图上下文对象
g.drawImage(img, 0, 0, newW, newH, null);                  //在缓冲图像上绘制指定大小的图像
```

说明：上面代码中的 img 是一个显示有图片的 Image 对象，这样就可以将图片绘制到缓冲图像对象上。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SaveZoomImageFrame 窗体类。

（3）在 SaveZoomImageFrame 窗体类中为“放大”按钮添加事件代码，用于实现放大图片的操作。“放大”按钮的事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        value += 5;                                        //调整 value 的值，用于放大图片
        if (value >= 200.0f) {                             //如果 value 的值大于等于 200
            value = 200.0f;                                //使 value 的值等于 200
        }
        imagePanel.repaint();                              //重新调用面板类的 paint()方法
    }
});
```

（4）在 SaveZoomImageFrame 窗体类中，为“缩小”按钮添加事件代码，用于实现缩小图片的操作。“缩小”按钮的事件代码如下：

```
button_1.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        value -= 5;                                        //调整 value 的值，用于缩小图片
        if (value <= 0.0f) {                               //如果 value 的值小于等于 0
            value = 0.0f;                                  //使 value 的值等于 0
        }
        imagePanel.repaint();                              //重新调用面板类的 paint()方法
    }
});
```

（5）在 SaveZoomImageFrame 窗体类中为“保存”按钮添加事件代码，用于实现保存缩放后的图片。“保存”按钮事件的关键代码如下：

```
String fileExtName = fileName.substring(fileName .indexOf(".") + 1);          //文件扩展名，不含点（.）
String pathAndName = path + "\\" + fileName;                                   //文件的完整路径
BufferedImage bufferImage = new BufferedImage(newW, newH, BufferedImage.TYPE_INT_RGB);  //创建缓冲图像对象
Graphics g = bufferImage.getGraphics();                                        //获得绘图上下文对象
g.drawImage(img, 0, 0, newW, newH, null);                                      //在缓冲图像上绘制图像
try {
    ImageIO.write(bufferImage, fileExtName, new File( pathAndName));           //将缓冲图像保存到磁盘
} catch (IOException e1) {
    JOptionPane.showMessageDialog(null, "保存失败\n" + e1.getMessage());       //显示提示信息
}
```

注意：在使用本实例时，需要将项目中用到的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 120：直观灵活地控制图片的缩放比例。

本实例以每次递增或递减 5 个百分点对图片进行放大或缩小操作，但是无法控制缩放的百分比，为此可以通过文本框等控件对缩放的百分比进行控制，这样即可直观灵活地控制图片的缩放比例。

实例 121　为图片添加水印并保存　　初级

光盘位置：光盘\MR\121　　趣味指数：★★★

实例说明

本实例演示了如何将添加有水印的图片保存到磁盘。运行程序，单击窗体上的“添加水印”按钮，将为窗体上的图片添加水印，效果如图 6.4 所示，单击窗体上的“保存图片”按钮，可以保存添加有水印的图片。

图 6.4　为图片添加水印并保存

关键技术

本实例通过在缓冲图像对象的绘图上下文中绘制图像和文本，使缓冲图像对象含有图片和水印文字，另外，为了使缓冲图像上的水印文字与在窗体上添加的水印文字的大小、位置、颜色和角度等属性相同，要将两者的这些属性都设置为相同的值，然后保存该缓冲图像对象，实现了保存添加有水印的图片。

（1）在缓冲图像对象的绘图上下文中绘制含有水印的图片，并使缓冲图像上的水印文字与在窗体上添加的水印文字的大小、位置、颜色和角度等属性相同。

☑　在缓冲图像上绘制图片，代码如下：

```
BufferedImage bufferImage = new BufferedImage(img
        .getWidth(SaveWatermarkPictureFrame.this), img
        .getHeight(SaveWatermarkPictureFrame.this),
        BufferedImage.TYPE_INT_RGB);                          //创建缓冲图像对象
Graphics2D g2 = (Graphics2D)bufferImage.getGraphics();        //获得绘图上下文对象
g2.drawImage(img, 0, 0, img.getWidth(SaveWatermarkPictureFrame.this),
        img.getHeight(SaveWatermarkPictureFrame.this), null); //在缓冲图像上绘制图像
```

☑　在缓冲图像上为图片添加水印，并设置与在窗体上为图片添加的水印文字相同的属性，代码如下：

```
g2.rotate(Math.toRadians(-30));                               //旋转绘图上下文对象
Font font = new Font("楷体", Font.BOLD, 72);                  //创建字体对象
g2.setFont(font);                                             //指定字体
g2.setColor(Color.RED);                                       //指定颜色
AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.4f);   //获得表示透明度的 AlphaComposite 对象
g2.setComposite(alpha);                                       //指定 AlphaComposite 对象
g2.drawString("编程词典", -30, 240);                          //绘制文本，实现水印
```

（2）将绘制有水印图片的缓冲图像对象保存到磁盘的代码如下：

```
try {
    ImageIO.write(bufferImage, fileExtName, new File(pathAndName));   //将缓冲图像保存到磁盘
} catch (IOException e1) {
    JOptionPane.showMessageDialog(null,"保存失败\n"+e1.getMessage()); //显示提示信息
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SaveWatermarkPictureFrame 窗体类。

（3）在 SaveWatermarkPictureFrame 窗体类中创建一个 DrawWatermarkPanel 面板类，用于在窗体上显示水印文字。该类的代码如下：

```
class DrawWatermarkPanel extends JPanel {
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;                           //获得 Graphics2D 对象
        g2.drawImage(img, 0, 0, getWidth(), getHeight(), this);   //绘制图像
        if (watermark) {
            g2.rotate(Math.toRadians(-30));                       //旋转绘图上下文对象
            Font font = new Font("楷体", Font.BOLD, 72);          //创建字体对象
            g2.setFont(font);                                     //指定字体
            g2.setColor(Color.RED);                               //指定颜色
            AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.4f);  //获得表示透明度的 AlphaComposite 对象
            g2.setComposite(alpha);                               //指定 AlphaComposite 对象
            g2.drawString("编程词典", -30, 240);                  //绘制文本，实现水印
        }
    }
}
```

（4）在 SaveWatermarkPictureFrame 窗体类中为“保存图片”按钮添加事件代码，完成保存水印图片的功能，其关键代码已经在关键技术中给出。

注意：由于篇幅原因，这里只给出了部分功能的代码，如果需要对本实例有更深入的了解，可以查看源程序代码。

秘笈心法

心法领悟 121：将两张或多张图片拼接成一张图片。

通过创建每幅图片的 Image 对象，并通过该对象获得每幅图片的大小，然后计算出图片的总大小，接着创建一个大小适合的 BufferedImage 对象，把每个 Image 对象按原图片大小绘制到该 BufferedImage 对象的适合位

置，再将这个 BufferedImage 对象保存到磁盘，即可完成将两张或多张图片拼接成一张图片的操作。

实例 122 溶合两张图片并保存

光盘位置：光盘\MR\122

初级

趣味指数：★★★

实例说明

本实例使用 Java 的绘图技术实现了溶合两张图片，并将溶合后的图片保存到磁盘中。运行程序，首先在窗体上显示第一张图片，效果如图 6.5 所示；单击“第二张”按钮，将在窗体上显示第二张图片，效果如图 6.6 所示；单击窗体上的“溶合图片”按钮，将溶合这两张图片，效果如图 6.7 所示；图片溶合后单击“保存图片”按钮，将保存溶合的图片。

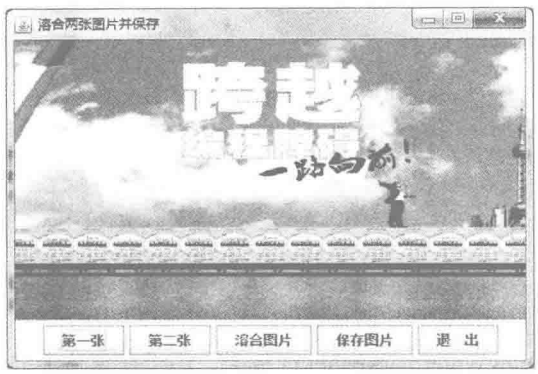


图 6.5 显示第一张图片的效果

图 6.6 显示第二张图片的效果

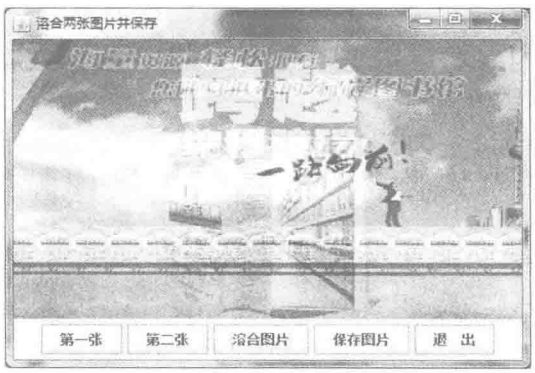


图 6.7 溶合两张图片的效果

关键技术

本实例首先在面板对象和缓冲图像对象的绘图上下文中绘制第一张图像，然后调用绘图上下文的透明度，接着再绘制第二张图像，从而实现了溶合两张图片和保存溶合后的图片的操作，下面是在面板对象的绘图上下文中溶合两张图片的代码：

```
public void paint(Graphics g) {
    Graphics2D g2 = (Graphics2D) g;                    //获得 Graphics2D 对象
    panelWidth = getWidth();                           //获得图像面板的宽度
    panelHeight = getHeight();                         //获得图像面板的高度
    g2.drawImage(img1, 0, 0, panelWidth, panelHeight, this);  //绘制图像
```

```
        if (mixFlag) {                                          //标记溶合图片
            AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.4f);   //获得表示透明度的 AlphaComposite 对象
            g2.setComposite(alpha);                             //指定 AlphaComposite 对象
        }
        if (firstOrSecondFlag) {                                //标记绘制第二张图片
            g2.drawImage(img2, 0, 0, panelWidth, panelHeight, this);   //绘制调整透明度后的图片
        }
    }
```

说明：由于在缓冲图像的绘图上下文中溶合两张图片的功能与面板类相似，这里不再给出，可以通过源程序代码进行查看。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SaveMixPictureFrame 窗体类。

（3）在 SaveMixPictureFrame 窗体类中创建一个 MixPicturePanel 面板类，用于在窗体上显示两张图片的溶合效果，面板类 MixPicturePanel 中的代码已经在关键技术中给出。

（4）在 SaveMixPictureFrame 窗体类中为“保存图片”按钮添加事件代码，用于将两张图片溶合后的效果保存为图片。“保存图片”按钮事件的关键代码如下：

```
BufferedImage bufferImage = new BufferedImage(panelWidth, panelHeight,
        BufferedImage.TYPE_INT_RGB);                            //创建缓冲图像对象
Graphics2D g2 = (Graphics2D)bufferImage.getGraphics();          //获得绘图上下文对象
g2.drawImage(img1, 0, 0, panelWidth, panelHeight, SaveMixPictureFrame.this);   //绘制图像
    AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.4f);   //获得表示透明度的 AlphaComposite 对象
    g2.setComposite(alpha);                                     //指定 AlphaComposite 对象
    g2.drawImage(img2, 0, 0, panelWidth, panelHeight, SaveMixPictureFrame.this);   //绘制调整透明度后的图片
try {
    ImageIO.write(bufferImage, fileExtName, new File(pathAndName));   //将缓冲图像保存到磁盘
} catch (IOException e1) {
    JOptionPane.showMessageDialog(null,"保存失败\n"+e1.getMessage());   //显示提示信息
}
```

注意：在使用本实例时，需将项目需要的两个图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 122：保存任意两张图片溶合的效果。

本实例实现了将项目中指定的两张图片溶合，并保存为图片，如果希望将任意两张图片溶合的效果保存为图片，可以通过文件选择对话框来选择两张图片，然后将这两张图片溶合并保存为图片。

实例 123　模糊图片并保存

初级

光盘位置：光盘\MR\123

趣味指数：★★★

实例说明

本实例利用 Java 的绘图技术实现了模糊图片并保存。运行程序，单击窗体上的“模糊图片”按钮，将模糊窗体上显示的图片，效果如图 6.8 所示；图片模糊后单击“保存图片”按钮，将把模糊后的图片保存到磁盘中。

图 6.8 模糊图片并保存

关键技术

本实例通过 ConvolveOp 类的 filter()方法实现了图片的模糊操作，该方法返回一个 BufferedImage 对象，该对象上绘制有模糊后的图片，然后使用 ImageIO 类的 write()方法，将绘制有模糊图片的 BufferedImage 对象保存到磁盘，从而实现了本实例的功能。

（1）实现对图片的模糊操作，并获得 BufferedImage 对象。

☑ 定义 convolve()方法，用于获得 BufferedImage 对象，该方法的代码见实例 122。

☑ 为 convolve()方法传递参数，实现模糊图片的功能，代码如下：

```
float[] elements = new float[9];                                    //定义表示像素分量的数组
for (int i = 0; i < 9; i++) {
    elements[i] = 0.11f;                                            //为数组赋值
}
convolve(elements);                                                 //调用方法，实现模糊功能
```

说明：上面代码中的 image 就是为 convolve()方法传递参数后获得的绘制有模糊图片的 BufferedImage 对象。

（2）使用 ImageIO 类的 write()方法，将绘制有模糊图片的 BufferedImage 对象保存到磁盘，代码如下：

```
try {
    ImageIO.write(image, fileExtName, new File(pathAndName));       //将缓冲图像保存到磁盘
} catch (IOException e1) {
    JOptionPane.showMessageDialog(null,"保存失败\n"+e1.getMessage()); //显示提示信息
}
```

说明：上面代码中的 image 就是绘制有模糊图片的 BufferedImage 对象；fileExtName 是图片保存到磁盘上的扩展名；pathAndName 是图片保存的完整路径和文件名。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SaveBlurPictureFrame 窗体类。

（3）在 SaveBlurPictureFrame 窗体类中创建一个 BlurPicturePanel 面板类，用于在窗体上显示图片的模糊效果，面板类 BlurPicturePanel 的代码如下：

```
class BlurPicturePanel extends JPanel {
    public BlurPicturePanel() {
        Image img = null;                                           //声明创建图像对象
        try {
            img = ImageIO.read(new File("src/img/imag.jpg"));       //创建图像对象
        } catch (IOException e) {
            e.printStackTrace();
        }
        image = new BufferedImage(img.getWidth(null), img.getHeight(null),
```

```
                BufferedImage.TYPE_INT_RGB);                    //创建缓冲图像对象
        image.getGraphics().drawImage(img, 0, 0, null);         //在缓冲图像对象上绘制图像
    }
    public void paint(Graphics g) {
        if (image != null) {
            g.drawImage(image, 0, 0, null);                     //绘制缓冲图像对象
        }
    }
}
```

（4）在 SaveBlurPictureFrame 窗体类中为“保存图片”按钮添加事件代码，用于将图片模糊后的效果保存为图片。“保存图片”按钮事件的关键代码如下：

```
String path = dialog.getDirectory();                            //获得文件的保存路径
String fileName = dialog.getFile();                             //获得保存的文件名
if (path == null || fileName == null) {
    return;
}
String fileExtName = fileName.substring(fileName.indexOf(".") + 1);   //文件扩展名，不含点（.）
String pathAndName = path + "\\" + fileName;                    //文件的完整路径
try {
    ImageIO.write(image, fileExtName, new File(pathAndName));   //将缓冲图像保存到磁盘
} catch (IOException e1) {
    JOptionPane.showMessageDialog(null, "保存失败\n" + e1.getMessage());   //显示提示信息
}
```

注意：在使用本实例时，需将项目需要的图片文件放到项目的 src 文件夹的子文件夹 img 中，否则程序无法正常运行。

秘笈心法

心法领悟 123：获得 ImageIO 类可读取的图片格式。

使用 ImageIO 类并不能读取磁盘上所有格式的图片，为此，可以使用 ImageIO 类的 getReaderFormatNames()方法，获得所有可以从磁盘读取的图片格式，再使用 ImageIO 类的 read()方法对图片进行读取，从而可以正确地获得 Image 对象。

实例 124　锐化图片并保存

初级

光盘位置：光盘\MR\124

趣味指数：★★★

实例说明

本实例利用 Java 的绘图技术实现锐化图片，并将锐化后的效果保存为图片的功能。运行程序，效果如图 6.9 所示；单击窗体上的“锐化图片”按钮，将锐化窗体上显示的图片，效果如图 6.10 所示；图片锐化后单击“保存图片”按钮，将把锐化后的图片保存到磁盘中。

图 6.9　锐化之前的图片效果

图 6.10　图片锐化后的效果

关键技术

本实例通过 ConvolveOp 类的 filter()方法实现了图片的锐化操作，该方法返回一个 BufferedImage 对象，该对象上绘制有锐化后的图片，然后使用 ImageIO 类的 write()方法将绘制有锐化图片的 BufferedImage 对象保存到磁盘，从而实现了锐化图片并保存到磁盘的功能。

（1）实现对图片的锐化操作，并获得 BufferedImage 对象。

☑ 定义 convolve()方法，用于获得 BufferedImage 对象，该方法的代码见实例 122。

☑ 为 convolve()方法传递参数，实现锐化图片的功能，代码如下：

```
float[] elements = {0.0f,-1.0f,0.0f,-1.0f,5.0f,-1.0f,0.0f,-1.0f,0.0f};      //声明表示像素分量的数组
convolve(elements);                                                          //调用方法实现图片锐化功能
sharpenFlag = true;                                                          //锐化图片标记
```

说明：上面代码中的 image 就是为 convolve()方法传递参数后获得的绘制有锐化效果图片的 BufferedImage 对象。

（2）使用 ImageIO 类的 write()方法，将绘制有锐化效果图片的 BufferedImage 对象保存到磁盘，完成锐化图片并保存到磁盘的操作。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SaveSharpenPictureFrame 窗体类。

（3）在 SaveSharpenPictureFrame 窗体类中创建一个 SharpenPicturePanel 面板类，用于在窗体上显示图片的锐化效果。

（4）在 SaveSharpenPictureFrame 窗体类中为“保存图片”按钮添加事件代码，用于将图片锐化后的效果保存为图片。“保存图片”按钮事件的关键代码如下：

```
String fileExtName = fileName.substring(fileName.indexOf(".")+1);          //文件扩展名，不含点（.）
String pathAndName = path + "\\" + fileName;                               //文件的完整路径
try {
    ImageIO.write(image, fileExtName, new File(pathAndName));              //将缓冲图像保存到磁盘
} catch (IOException e1) {
    JOptionPane.showMessageDialog(null,"保存失败\n"+e1.getMessage());      //显示提示信息
}
```

说明：上面代码中的 fileName 是在文件保存对话框中输入的文件名；path 是文件的保存路径。

秘笈心法

心法领悟 124：防止文件扩展名错误。

由于使用 ImageIO 类的 write()方法保存图片时需要使用文件的扩展名，为了防止本实例中获得的文件扩展名不正确，最好使用字符串类的 lastIndexOf()方法获得文件名中最后一个点的位置，然后再使用 substring()方法即可获得正确的文件扩展名。

实例 125 照亮边缘并保存

光盘位置：光盘\MR\125

初级

趣味指数：★★★★

实例说明

本实例利用 Java 的绘图技术，实现了对图片照亮边缘，并将照亮边缘后的效果保存为图片的功能。运行程

序，效果如图 6.11 所示，单击窗体上的“照亮边缘”按钮，将对窗体上的图片执行照亮边缘操作，效果如图 6.12 所示；对图片执行照亮边缘操作后，单击“保存图片”按钮，将把照亮边缘后的图片保存到磁盘中。

图 6.11　照亮边缘之前的效果

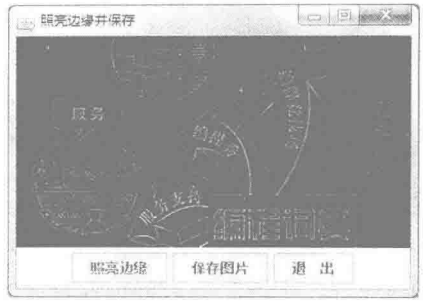


图 6.12　照亮边缘之后的效果

关键技术

本实例通过 ConvolveOp 类的 filter()方法，实现了对图片进行照亮边缘的操作，该方法返回一个 BufferedImage 对象，该对象上绘制有照亮边缘后的图片，然后使用 ImageIO 类的 write()方法，将绘制有照亮边缘效果图片的 BufferedImage 对象保存到磁盘，从而实现了照亮边缘并保存到磁盘的功能。

（1）实现对图片的照亮边缘操作，并获得 BufferedImage 对象。

☑　定义 convolve()方法，用于获得 BufferedImage 对象，该方法的代码见实例 122。

☑　为 convolve()方法传递参数，实现对图片执行照亮边缘的操作，代码如下：

```
float[] elements = {0.0f,-1.0f,0.0f,-1.0f,4.0f,-1.0f,0.0f,-1.0f,0.0f};          //声明表示像素分量的数组
convolve(elements);                                                              //调用方法实现图片照亮边缘功能
edgeDetectFlag = true;                                                           //照亮边缘图片标记
```

说明：上面代码中的 image 就是为 convolve()方法传递参数后获得的绘制有照亮边缘效果图片的 BufferedImage 对象。

（2）使用 ImageIO 类的 write()方法，将绘制有照亮边缘效果图片的 BufferedImage 对象保存到磁盘，完成对图片照亮边缘并保存到磁盘的操作。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SaveEdgeDetectPictureFrame 窗体类。

（3）在 SaveEdgeDetectPictureFrame 窗体类中创建一个 EdgeDetectPicturePanel 面板类，用于在窗体上显示图片照亮边缘的效果。

（4）在 SaveEdgeDetectPictureFrame 窗体类中为“保存图片”按钮添加事件代码，用于将图片照亮边缘后的效果保存为图片。“保存图片”按钮事件的关键代码如下：

```
button_2.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        if (!edgeDetectFlag) {
            JOptionPane.showMessageDialog(null,"还没执行照亮边缘操作。");          //显示提示信息
            return;
        }
        FileDialog dialog = new FileDialog(SaveEdgeDetectPictureFrame.this,"保存");   //创建对话框
        dialog.setMode(FileDialog.SAVE);                                          //设置对话框为保存对话框
        dialog.setVisible(true);                                                  //显示保存对话框
        String path = dialog.getDirectory();                                      //获得文件的保存路径
        String fileName = dialog.getFile();                                       //获得保存的文件名
        if (path == null || fileName == null){
            return;
```

```
            }
            String fileExtName = fileName.substring(fileName.indexOf(".")+1);          //文件扩展名，不含点
            String pathAndName = path + "\\" + fileName;                               //文件的完整路径
            try {
                ImageIO.write(image, fileExtName, new File(pathAndName));              //将缓冲图像保存到磁盘
            } catch (IOException e1) {
                JOptionPane.showMessageDialog(null,"保存失败\n"+e1.getMessage());      //显示提示信息
            }
        }
    });
```

秘笈心法

心法领悟 125：避免关闭对话框或单击“取消”按钮程序出错。

在使用本实例时，如果已经打开保存对话框，但是又不想保存图片了，此时关闭对话框或单击“取消”按钮，就会出现保存路径不正确的错误，这时可以先对保存路径和文件名进行判断，如果路径和文件名为 null，就让程序执行 return 语句，这样程序就会结束当前方法而不会出错。

实例 126　反向并保存图片　　初级

光盘位置：光盘\MR\126　　趣味指数：★★★

实例说明

本实例利用 Java 的绘图技术实现了对图片进行反向操作，并将反向后的效果保存为图片的功能。运行程序，效果如图 6.13 所示，单击窗体上的“反向图片”按钮，将对窗体上的图片执行反向操作，效果如图 6.14 所示；对图片执行反向操作后，单击“保存图片”按钮，将把反向后的图片保存到磁盘中。

图 6.13　反向之前的效果

图 6.14　反向之后的效果

关键技术

本实例使用 ShortLookupTable 类和 LookupOp 类对绘制在缓冲图像对象上的图片进行反向处理，然后使用 ImageIO 类的 write()方法，将绘制有反向效果图片的 BufferedImage 对象保存到磁盘，从而实现了将图片反向并保存到磁盘的功能。

（1）实现对图片的反向操作，并获得 BufferedImage 对象。

☑　创建表示颜色反向的分量数组，并使用 ShortLookupTable 类定义一个查找表对象，代码如下：

```
short[] negative = new short[256];                          //创建表示颜色反向的分量数组
for (int i = 0; i<256;i++){
    negative[i] = (short)(255-i);                           //为数组赋值
}
ShortLookupTable table = new ShortLookupTable(0,negative);  //创建查找表对象
```

☑　使用 LookupOp 类实现从源到目标的查找操作，并调用 filter()方法实现反向操作，代码如下：

```
LookupOp op = new LookupOp(table,null);                     //创建实现从源到目标查找操作的 LookupOp 对象
image = op.filter(image, null);                             //调用 LookupOp 对象的 filter()方法，实现图像反向功能
```

（2）使用 ImageIO 类的 write()方法，将绘制有反向效果图片的 BufferedImage 对象保存到磁盘，完成对图片反向并保存到磁盘的操作。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 SaveNegativePictureFrame。

（3）在 SaveNegativePictureFrame 窗体类中创建一个 NegativePicturePanel 面板类，用于在窗体上显示图片反向的效果。

（4）在 SaveNegativePictureFrame 窗体类中为“反向图片”按钮添加事件代码，用于实现反向图片和还原图片的功能。“反向图片”按钮事件的代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        short[] negative = new short[256];                          //创建表示颜色反向的分量数组
        for (int i = 0; i<256;i++){
            negative[i] = (short)(255-i);                           //为数组赋值
        }
        ShortLookupTable table = new ShortLookupTable(0,negative);  //创建查找表对象
        LookupOp op = new LookupOp(table,null);                     //创建实现从源到目标查找操作的 LookupOp 对象
        image = op.filter(image, null);                             //调用 LookupOp 对象的 filter()方法，实现图像反向功能
        repaint();                                                  //调用 paint()方法
        negativeFlag = !negativeFlag;                               //标记是否已反向
        if (negativeFlag){
            button.setText("还原图片");                             //设置按钮标题
        }else{
            button.setText("反向图片");                             //设置按钮标题
        }
    }
});
```

（5）在 SaveNegativePictureFrame 窗体类中为“保存图片”按钮添加事件代码，用于将图片反向后的效果保存为图片。“保存图片”按钮事件的关键代码如下：

```
try {
    ImageIO.write(image, fileExtName, new File(pathAndName));          //将缓冲图像保存到磁盘
} catch (IOException e1) {
    JOptionPane.showMessageDialog(null,"保存失败\n"+e1.getMessage());  //显示提示信息
}
```

说明：上面代码中的 image 就是绘制有反向图片的 BufferedImage 对象；fileExtName 是图片保存到磁盘上的扩展名；pathAndName 是图片保存的完整路径和文件名。

秘笈心法

心法领悟 126：两次单击同一个按钮执行两种不同的操作。

在程序中，有时希望通过两次单击同一个按钮执行两种不同的操作，这时可以定义一个布尔类型的标记变量，然后在单击按钮时对该标记变量取相反的值，并使用 if 语句对标记变量的值进行判断，以执行两种不同的操作，例如：

```
negativeFlag = !negativeFlag;                                   //标记是否已反向
```

```
if (negativeFlag){
    button.setText("还原图片");                    //设置按钮标题
}else{
    button.setText("反向图片");                    //设置按钮标题
}
```

实例 127　填充纹理并保存为图片

光盘位置：光盘\MR\127　　中级　　趣味指数：★★★

实例说明

本实例利用 Java 的绘图技术实现了填充纹理并保存为图片的功能。运行程序，效果如图 6.15 所示，单击窗体上的“保存为图片”按钮，将把填充的纹理保存为图片。

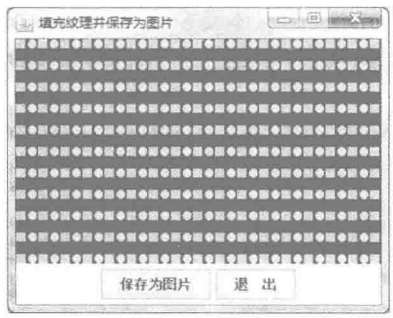


图 6.15　填充纹理的效果

关键技术

本实例通过为绘图上下文对象指定使用 TexturePaint 类创建的纹理填充对象，然后绘制填充图形实现填充纹理的功能，并使用 ImageIO 类的 write()方法将绘制有填充纹理的 BufferedImage 对象保存到磁盘，从而实现了填充纹理并保存为磁盘图片的功能。

（1）使用 Graphics2D 类的 setPaint()方法，为绘图上下文设置 TexturePaint 类创建的纹理填充对象，然后绘制填充图形，实现纹理填充功能，代码如下：

```
BufferedImage image = new BufferedImage(200, 200,
        BufferedImage.TYPE_INT_RGB);                    //创建缓冲图像对象
Graphics2D g2 = image.createGraphics();                 //获得缓冲图像对象的绘图上下文对象
g2.setColor(Color.BLUE);                                //设置颜色
g2.fillRect(0, 0, 90, 90);                              //绘制填充矩形
//省略部分代码
Rectangle2D rect = new Rectangle2D.Float(10, 10, 20, 20);   //创建 Rectangle2D 对象
TexturePaint textPaint = new TexturePaint(image, rect);     //创建纹理填充对象
Graphics2D graphics2 = (Graphics2D) g;                  //转换 paint()方法的绘图上下文对象
graphics2.setPaint(textPaint);                          //为绘图上下文对象设置纹理填充对象
newW = getWidth();
newH = getHeight();
Rectangle2D.Float ellipse2 = new Rectangle2D.Float(0, 0, newW, newH);   //创建矩形对象
graphics2.fill(ellipse2);                               //绘制填充纹理的矩形
```

（2）使用 ImageIO 类的 write()方法，将绘制有填充纹理的 BufferedImage 对象保存到磁盘，完成填充纹理并保存为磁盘图片的操作。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的 SaveTextureFillFrame 窗体类。

（3）在 SaveTextureFillFrame 窗体类中创建内部面板类 TextureFillPanel，该面板类用于在窗体上显示纹理填充效果。

（4）在 SaveTextureFillFrame 窗体类中为“保存图片”按钮添加事件代码，用于将填充纹理保存为图片。“保存图片”按钮事件的关键代码如下：

```
TexturePaint textPaint = new TexturePaint(image, rect);                    //创建纹理填充对象
BufferedImage bufferImage = new BufferedImage(newW, newH,
        BufferedImage.TYPE_INT_RGB);                                       //创建缓冲图像对象
Graphics g = bufferImage.getGraphics();                                    //获得缓冲图像的绘图上下文对象
Graphics2D graphics2 = (Graphics2D) g;                                     //转换绘图上下文对象为 Graphics2D 类型
graphics2.setPaint(textPaint);                                             //为绘图上下文对象指定纹理填充对象
Rectangle2D.Float ellipse2 = new Rectangle2D.Float(0, 0, newW, newH);       //创建矩形对象
graphics2.fill(ellipse2);                                                  //绘制填充纹理的矩形
try {
    ImageIO.write(bufferImage, fileExtName, new File( pathAndName));       //将缓冲图像保存到磁盘
} catch (IOException e1) {
    JOptionPane.showMessageDialog(null, "保存失败\n" + e1.getMessage());   //显示提示信息
}
```

说明：上面代码中的 image 就是绘制有填充纹理的 BufferedImage 对象；rect 是决定填充纹理大小的 Rectangle2D 对象。

秘笈心法

心法领悟 127：实现多种图案的纹理填充效果。

为了实现多种图案的纹理填充效果，可以在用作填充纹理的缓冲图像上绘制更多的图形，并使用不同的颜色，甚至也可以添加一些图片，这样在绘制填充图形时，就会显示更多的图案，使填充的纹理内容更加丰富。

6.2　图片在数据库中的存取

实例 128　图片存储到 Access 数据库中　　中级

光盘位置：光盘\MR\128　　趣味指数：★★

实例说明

本实例实现了在 Access 数据库中保存图片文件的功能。运行程序，在“姓名：”标签右侧的文本框中输入姓名，然后单击“选择图片”按钮选择一幅图片，效果如图 6.16 所示，单击“保存”按钮即可将姓名和图片保存到 Access 数据库中。

图 6.16　图片存储到 Access 数据库中

关键技术

本实例通过建立到 Access 数据库的连接，并通过 PreparedStatement 的 setBinaryStream()方法将图片保存到 Access 数据库表的 OLE 对象类型的字段中，从而实现了本实例的功能。

（1）建立到 Access 数据库的连接，可以通过如下代码来实现：

```
Connection conn = null;                                                    //定义数据库连接
String url = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ= src/database/db_picture.mdb ";   //数据库 db_picture.mdb 的 URL
String username = "";                                                      //数据库的用户名
String password = "";                                                      //数据库的密码
conn = DriverManager.getConnection(url, username, password);               //建立连接
```

（2）通过 PreparedStatement 的 setBinaryStream()方法将图片保存到 Access 数据库表的 OLE 对象类型的字段中，代码如下：

```
String sql = "insert into tb_picture (name,picture) values(?,?)";          //添加记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                         //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                                     //为第 1 个参数赋值
ps.setBinaryStream(2, in, (int) file.length());                            //为第 2 个参数赋值
int flag = ps.executeUpdate();                                             //执行 SQL 语句，获得更新记录数
```

说明：上面代码中的 conn 是 Access 数据库的连接对象；in 是图片文件的输入流对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DAO 类，用于加载数据库驱动和建立到 Access 数据库的连接。

（3）在项目中创建一个继承 JFrame 类的 SavePictureToAccessFrame 窗体类，并完成窗体界面的设计，在“保存”按钮的事件中完成保存图片到 Access 数据库中的操作。“保存”按钮事件的关键代码如下：

```
FileInputStream in = new FileInputStream(file);                            //图片文件的输入流对象
Connection conn = DAO.getConn();                                           //获得数据库连接对象
String sql = "insert into tb_picture (name,picture) values(?,?)";          //添加记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                         //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                                     //为第 1 个参数赋值
ps.setBinaryStream(2, in, (int) file.length());                            //为第 2 个参数赋值
int flag = ps.executeUpdate();                                             //执行 SQL 语句，获得更新记录数
if (flag > 0) {
    JOptionPane.showMessageDialog(null, "保存成功！");
} else {
    JOptionPane.showMessageDialog(null, "保存失败！");
}
```

秘笈心法

心法领悟 128：Access 数据库存储图片的字段类型。

在数据库中存储图片是非常重要的操作，为了能够在 Access 数据库中存储图片，必须要将存储图片的字段设置为“OLE 对象”类型。

实例 129 图片存储到 MySQL 数据库中

光盘位置：光盘\MR\129　　中级　　趣味指数：★★

实例说明

本实例实现了在 MySQL 数据库中保存图片文件的功能。运行程序，在“姓名：”标签右侧的文本框中输入姓名，然后单击“选择图片”按钮选择一幅图片，效果如图 6.17 所示，单击“保存”按钮即可将姓名和图片

保存到 MySQL 数据库中。

图 6.17　图片存储到 MySQL 数据库中

关键技术

本实例通过建立到 MySQL 数据库的连接，并通过 PreparedStatement 的 setBinaryStream()方法，将图片保存到 MySQL 数据库表的 BLOB 类型的字段中，从而实现了本实例的功能。

（1）建立到 MySQL 数据库的连接，可以通过如下代码来实现：

```
Connection conn = null;                                                          //定义数据库连接
String url = "jdbc:mysql://localhost:3306/db_database?useUnicode=true&characterEncoding=utf-8";  //数据库 db_database 的 URL
String username = "root";                                                        //数据库的用户名
String password = "111";                                                         //数据库的密码
conn = DriverManager.getConnection(url, username, password);                     //建立连接
```

（2）通过 PreparedStatement 的 setBinaryStream()方法，将图片保存到 MySQL 数据库表的 BLOB 类型的字段中，代码如下：

```
String sql = "insert into tb_picture (name,picture) values(?,?)";   //添加记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                  //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                              //为第 1 个参数赋值
ps.setBinaryStream(2, in, (int) file.length());                     //为第 2 个参数赋值
int flag = ps.executeUpdate();                                      //执行 SQL 语句，获得更新记录数
```

说明：上面代码中的 conn 是 MySQL 数据库的连接对象；in 是图片文件的输入流对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DAO 类，用于加载数据库驱动和建立到 MySQL 数据库的连接。

（3）在项目中创建一个继承 JFrame 类的 SavePictureToMySQLFrame 窗体类，并完成窗体界面的设计，在“保存”按钮的事件中完成保存图片到 MySQL 数据库中的操作。“保存”按钮事件的关键代码如下：

```
FileInputStream in = new FileInputStream(file);                     //图片文件的输入流对象
Connection conn = DAO.getConn();                                    //获得数据库连接对象
String sql = "insert into tb_picture (name,picture) values(?,?)";   //添加记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                  //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                              //为第 1 个参数赋值
ps.setBinaryStream(2, in, (int) file.length());                     //为第 2 个参数赋值
int flag = ps.executeUpdate();                                      //执行 SQL 语句，获得更新记录数
if (flag > 0) {
    JOptionPane.showMessageDialog(null, "保存成功！");               //提示保存成功
} else {
    JOptionPane.showMessageDialog(null, "保存失败！");               //提示保存失败
}
```

秘笈心法

心法领悟 129：MySQL 数据库存储图片的字段类型。

在数据库中存储图片是非常重要的操作，为了能够在 MySQL 数据库中存储图片，必须将存储图片的字段

设置为 BLOB 类型。

实例 130　图片存储到 SQLServer 数据库中

光盘位置：光盘\MR\130　　中级　　趣味指数：★★

实例说明

本实例实现了在 SQLServer 数据库中保存图片文件的功能。运行程序，在“姓名：”标签右侧的文本框中输入姓名，然后单击“选择图片”按钮选择一幅图片，效果如图 6.18 所示，单击“保存”按钮即可将姓名和图片保存到 SQLServer 数据库中。

图 6.18　图片存储到 SQLServer 数据库中

关键技术

本实例通过建立到 SQLServer 数据库的连接，并通过 PreparedStatement 的 setBinaryStream()方法，将图片保存到 SQLServer 数据库表的 image 类型的字段中，从而实现了本实例的功能。

（1）建立到 SQLServer 数据库的连接，可以通过如下代码来实现：

```
Connection conn = null;                                          //定义数据库连接
String url = "jdbc:jtds:sqlserver://localhost:1433/db_database"; //数据库 db_database 的 URL
String username = "sa";                                          //数据库的用户名
String password = "";                                            //数据库的密码
conn = DriverManager.getConnection(url, username, password);     //建立连接
```

（2）通过 PreparedStatement 的 setBinaryStream()方法将图片保存到 SQLServer 数据库表的 Image 类型的字段中，代码如下：

```
String sql = "insert into tb_picture (name,picture) values(?,?)";  //添加数据记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                 //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                             //为第 1 个参数赋值
ps.setBinaryStream(2, in, (int) file.length());                    //为第 2 个参数赋值
```

说明：上面代码中的 conn 是 SQLServer 数据库的连接对象；in 是图片文件的输入流对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DAO 类，用于加载数据库驱动和建立到 SQLServer 数据库的连接。

（3）在项目中创建一个继承 JFrame 类的 SavePictureToSQLServerFrame 窗体类，并完成窗体界面的设计，在“保存”按钮的事件中完成保存图片到 SQLServer 数据库中的操作。“保存”按钮事件的关键代码如下：

```
FileInputStream in = new FileInputStream(file);                    //图片文件的输入流对象
Connection conn = DAO.getConn();                                   //获得数据库连接对象
String sql = "insert into tb_picture (name,picture) values(?,?)";  //添加记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                 //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                             //为第 1 个参数赋值
```

```
ps.setBinaryStream(2, in, (int) file.length());          //为第 2 个参数赋值
int flag = ps.executeUpdate();                           //执行 SQL 语句，获得更新记录数
if (flag > 0) {
    JOptionPane.showMessageDialog(null, "保存成功！");
} else {
    JOptionPane.showMessageDialog(null, "保存失败！");
}
```

秘笈心法

心法领悟 130：SQLServer 数据库存储图片的字段类型。

在数据库中存储图片是非常重要的操作，为了能够在 SQLServer 数据库中存储图片，可以将存储图片的字段设置为 Image 类型。

实例 131　读取 Access 数据库中存储的图片

光盘位置：光盘\MR\131　　中级　　趣味指数：★★

实例说明

本实例根据编号将 Access 数据库中存储的图片通过程序读取出来并显示在窗体上。运行程序，在“编号”文本框中输入编号，单击“按编号查询”按钮，将把与该编号对应的信息查询出来，并显示在窗体上，效果如图 6.19 所示。

图 6.19　读取 Access 数据库中存储的图片

关键技术

通过 ResultSet 结果集对象的 getBytes()方法，将 Access 数据库表中的图片存储到字节数组中，然后通过该字节数组创建 Icon 图标对象，并设置为窗体上标签显示的图标，从而实现了本实例的功能。

读取 Access 数据库表中的图片，并设置为标签上显示的图标，代码如下：

```
byte[] bt = rs.getBytes(2);                  //获得表中照片的字节数组
Icon icon = new ImageIcon(bt);               //创建图标对象
Dimension size = lb_picture.getSize();       //获得显示图标的标签大小
lb_picture.setIcon(icon);                    //为标签指定图标
lb_picture.setSize(size);                    //设置标签的大小
```

说明：上面代码中的 rs 是从数据库表中查询到的满足条件的结果集对象；lb_picture 是用于显示图标的标签控件。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DAO 类，用于加载数据库驱动和建立到 Access 数据库的连接。

（3）在项目中创建一个继承 JFrame 类的 ReadPictureFromAccessFrame 窗体类，并完成窗体界面的设计，在“按编号查询”按钮的事件中实现从 Access 数据库中读取满足条件的记录。“按编号查询”按钮事件的关键代码如下：

```
ps = conn.prepareStatement(sql);                    //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setInt(1, id);                                   //为参数赋值
rs = ps.executeQuery();                             //执行 SQL 语句
if (rs.next()) {
    String name = rs.getString(1);                  //获得姓名
    tf_name.setText(name);                          //在文本框中显示姓名
    byte[] bt = rs.getBytes(2);                     //获得表中照片的字节数组
    Icon icon = new ImageIcon(bt);                  //创建图标对象
    Dimension size = lb_picture.getSize();          //获得显示图标的标签大小
    lb_picture.setIcon(icon);                       //为标签指定图标
    lb_picture.setSize(size);                       //设置标签的大小
}
```

秘笈心法

心法领悟 131：将 ImageIcon 对象转换为 Image 对象。

本实例将从数据库中读取的图片存储到 ImageIcon 对象中，如果需要将图片存储到 Image 中，可以使用 ImageIcon 类的 getImage()方法获得 Image 对象，这样即可使用 Image 对象对图片进行处理。

实例 132　读取 MySQL 数据库中存储的图片

光盘位置：光盘\MR\132　　中级　　趣味指数：★★

实例说明

本实例根据编号将 MySQL 数据库中存储的图片通过程序读取出来并显示在窗体上。运行程序，在“编号”文本框中输入编号，单击“按编号查询”按钮，将把与该编号对应的信息查询出来，并显示在窗体上，效果如图 6.20 所示。

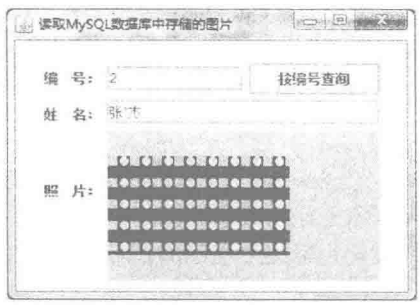


图 6.20　读取 MySQL 数据库中存储的图片

关键技术

通过 ResultSet 结果集对象的 getBlob()方法，将 MySQL 数据库表中的图片存储到 Blob 对象中，然后通过该 Blob 对象创建 Icon 图标对象，并设置为窗体上标签显示的图标，从而实现了本实例的功能。

读取 MySQL 数据库表中的图片，并设置为标签上显示的图标，代码如下：

```
Blob img = (Blob) rs.getBlob(2);                                    //获得图片的 Blob 对象
Icon icon = new ImageIcon(img.getBytes(1, (int) img.length()));     //创建图标对象
Dimension size = lb_picture.getSize();                              //获得显示图标的标签大小
```

```
lb_picture.setIcon(icon);                    //为标签指定图标
lb_picture.setSize(size);                    //设置标签的大小
```

说明：上面代码中的 rs 是从数据库表中查询到的满足条件的结果集对象；lb_picture 是用于显示图标的标签控件。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DAO 类，用于加载数据库驱动和建立到 MySQL 数据库的连接。

（3）在项目中创建一个继承 JFrame 类的 ReadPictureFromMySQLFrame 窗体类，并完成窗体界面的设计，在“按编号查询”按钮的事件中实现从 MySQL 数据库中读取满足条件的记录。“按编号查询”按钮事件的关键代码如下：

```
ps = conn.prepareStatement(sql);                                    //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setInt(1, id);                                                   //为参数赋值
rs = ps.executeQuery();                                             //执行 SQL 语句
if (rs.next()) {
    String name = rs.getString(1);                                  //获得姓名
    tf_name.setText(name);                                          //在文本框中显示姓名
    Blob img = (Blob) rs.getBlob(2);                                //获得图片的 Blob 对象
    Icon icon = new ImageIcon(img.getBytes(1, (int) img.length())); //创建图标对象
    Dimension size = lb_picture.getSize();                          //获得显示图标的标签大小
    lb_picture.setIcon(icon);                                       //为标签指定图标
    lb_picture.setSize(size);                                       //设置标签的大小
}
```

秘笈心法

心法领悟 132：Image 对象转换为 ImageIcon 对象。

如果已经创建了图片的 Image 对象，可以将该对象作为 ImageIcon 类构造方法的参数创建 ImageIcon 对象。

实例 133　读取 SQLServer 数据库中存储的图片　　中级

实例位置：光盘\MR\133　　趣味指数：★★

实例说明

本实例根据编号将 SQLServer 数据库中存储的图片通过程序读取出来并显示在窗体上。运行程序，在“编号：”文本框中输入编号，单击“按编号查询”按钮，将把与该编号对应的信息查询出来并显示在窗体上，效果如图 6.21 所示。

图 6.21　读取 SQLServer 数据库中存储的图片

关键技术

通过 ResultSet 结果集对象的 getBlob()方法，将 SQLServer 数据库表中的图片存储到 Blob 对象中，然后通过该 Blob 对象创建 Icon 图标对象，并设置为窗体上标签显示的图标，从而实现了本实例的功能。

读取 SQLServer 数据库表中的图片，并设置为标签上显示的图标，代码如下：

```
Blob img = (Blob) rs.getBlob(2);                                    //获得图片的 Blob 对象
Icon icon = new ImageIcon(img.getBytes(1, (int) img.length()));     //创建图标对象
Dimension size = lb_picture.getSize();                              //获得显示图标的标签大小
lb_picture.setIcon(icon);                                           //为标签指定图标
lb_picture.setSize(size);                                           //设置标签的大小
```

说明：上面代码中的 rs 是从数据库表中查询到的满足条件的结果集对象；lb_picture 是用于显示图标的标签控件。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DAO 类，用于加载数据库驱动和建立到 SQLServer 数据库的连接。

（3）在项目中创建一个继承 JFrame 类的 ReadPictureFromSQLServerFrame 窗体类，并完成窗体界面的设计，在“按编号查询”按钮的事件中实现从 SQLServer 数据库中读取满足条件的记录。“按编号查询”按钮事件的关键代码如下：

```
ps = conn.prepareStatement(sql);                                    //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setInt(1, id);                                                   //为参数赋值
rs = ps.executeQuery();                                             //执行 SQL 语句
if (rs.next()) {
    String name = rs.getString(1);                                  //获得姓名
    tf_name.setText(name);                                          //在文本框中显示姓名
    byte[] bt = rs.getBytes(2);                                     //获得表中照片的字节数组
    Icon icon = new ImageIcon(bt);                                  //创建图标对象
    Dimension size = lb_picture.getSize();                          //获得显示图标的标签大小
    lb_picture.setIcon(icon);                                       //为标签指定图标
    lb_picture.setSize(size);                                       //设置标签的大小
}
```

秘笈心法

心法领悟 133：SQLServer 数据库中可以存储图片的数据类型。

在 SQLServer 数据库中存储图片，可以使用两种字段数据类型，分别是 Image 和 binary 类型。

实例 134 修改 Access 数据库中存储的图片

中级

光盘位置：光盘\MR\134

趣味指数：★★★

实例说明

本实例实现对存储在 Access 数据库中的图片进行修改的功能。运行程序，在“编号：”标签右侧的文本框中输入编号，单击“按编号查询”按钮，将从数据库查询所输入编号的信息，效果如图 6.22 所示，如果需要修改图片，可以单击“选择图片”按钮选择一幅图片，然后单击“修改”按钮，修改 Access 数据库中存储的图片。

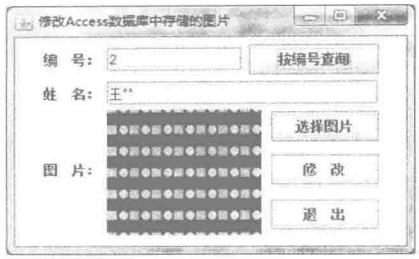


图 6.22　修改 Access 数据库中存储的图片

关键技术

本实例通过建立到 Access 数据库的连接，并通过 PreparedStatement 的 setBinaryStream()方法，实现对 Access 数据库中存储的图片进行修改的功能。

（1）建立到 Access 数据库的连接，代码见实例 128。

（2）通过 PreparedStatement 的 setBinaryStream()方法，对 Access 数据库表中存储的图片进行修改，代码如下：

```
String sql = "update tb_picture set name = ?,picture = ? where id = ?";     //定义修改数据记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                           //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                                       //为第 1 个参数赋值，修改姓名
ps.setBinaryStream(2, in, (int) file.length());                              //为第 2 个参数赋值，修改图片
ps.setInt(3, Integer.parseInt(tf_id.getText()));                             //为第 3 个参数赋值，设置要修改图片的编号
int flag = ps.executeUpdate();                                               //执行 SQL 语句，获得更新记录数
```

说明：上面代码中的 conn 是 Access 数据库的连接对象；in 是图片文件的输入流对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DAO 类，用于加载数据库驱动和建立到 Access 数据库的连接。

（3）在项目中创建一个继承 JFrame 类的 UpdateAccessPictureFrame 窗体类，并完成窗体界面的设计，在“按编号查询”按钮的事件中实现从 Access 数据库中查询满足指定编号的记录，并在窗体上的相应控件中显示。

（4）在 UpdateAccessPictureFrame 窗体类的“修改”按钮的事件中，修改 Access 数据库中满足条件的记录，实现修改图片的功能。“修改”按钮事件的关键代码如下：

```
FileInputStream in = new FileInputStream(file);                              //创建图片的输入流对象
Connection conn = DAO.getConn();                                             //获得数据库连接对象
String sql = "update tb_picture set name = ?,picture = ? where id = ?";     //定义修改数据记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                           //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                                       //为第 1 个参数赋值
ps.setBinaryStream(2, in, (int) file.length());                              //为第 2 个参数赋值
ps.setInt(3, Integer.parseInt(tf_id.getText()));                             //为第 3 个参数赋值
int flag = ps.executeUpdate();                                               //执行 SQL 语句，获得更新记录数
if (flag > 0) {
    JOptionPane.showMessageDialog(null, "修改成功！");                        //提示修改成功
} else {
    JOptionPane.showMessageDialog(null, "修改失败！");                        //提示修改失败
}
```

秘笈心法

心法领悟 134：在 Access 中存储音频文件。

在实例 128 中讲解了图片的存储，由于图片和音频都是二进制文件，所以可以用同样的方法，将音频文件存储到 Access 数据库中。

实例 135 修改 MySQL 数据库中存储的图片

光盘位置：光盘\MR\135

中级

趣味指数：★★★

实例说明

本实例实现了对存储在 MySQL 数据库中的图片进行修改的功能。运行程序，在“编号：”标签右侧的文本框中输入编号，单击“按编号查询”按钮，将从数据库查询所输入编号的信息，效果如图 6.23 所示，如果需要修改图片，可以单击“选择图片”按钮选择一幅图片，然后单击“修改”按钮，修改 MySQL 数据库中存储的图片。

图 6.23 修改 MySQL 数据库中存储的图片

关键技术

本实例通过建立到 MySQL 数据库的连接，并通过 PreparedStatement 的 setBinaryStream()方法，实现对 MySQL 数据库中存储的图片进行修改的功能。

（1）建立到 MySQL 数据库的连接，代码见实例 129。

（2）通过 PreparedStatement 的 setBinaryStream()方法，对 MySQL 数据库表中存储的图片进行修改，代码如下：

```
String sql = "update tb_picture set name = ?,picture = ? where id = ?";   //定义修改数据记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                         //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                                     //为第 1 个参数赋值，修改姓名
ps.setBinaryStream(2, in, (int) file.length());                            //为第 2 个参数赋值，修改图片
ps.setInt(3, Integer.parseInt(tf_id.getText()));                           //为第 3 个参数赋值，设置要修改图片的编号
int flag = ps.executeUpdate();                                             //执行 SQL 语句，获得更新记录数
```

说明：上面代码中的 conn 是 MySQL 数据库的连接对象；in 是图片文件的输入流对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DAO 类，用于加载数据库驱动和建立到 MySQL 数据库的连接。

（3）在项目中创建一个继承 JFrame 类的 UpdateMySQLPictureFrame 窗体类，并完成窗体界面的设计，在“按编号查询”按钮的事件中实现从 MySQL 数据库中查询满足指定编号的记录，并在窗体上的相应控件中显示。

（4）在 UpdateMySQLPictureFrame 窗体类的“修改”按钮的事件中修改 MySQL 数据库中满足条件的记录，实现修改图片的功能。“修改”按钮事件的关键代码如下：

```
FileInputStream in = new FileInputStream(file);                            //创建图片的输入流对象
Connection conn = DAO.getConn();                                           //获得数据库连接对象
String sql = "update tb_picture set name = ?,picture = ? where id = ?";    //定义修改数据记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                         //创建 PreparedStatement 对象，并传递 SQL 语句
```

```
ps.setString(1, name);                                          //为第 1 个参数赋值
ps.setBinaryStream(2, in, (int) file.length());                 //为第 2 个参数赋值
ps.setInt(3, Integer.parseInt(tf_id.getText()));                //为第 3 个参数赋值
int flag = ps.executeUpdate();                                  //执行 SQL 语句，获得更新记录数
if (flag > 0) {
    JOptionPane.showMessageDialog(null, "修改成功！");          //提示修改成功
} else {
    JOptionPane.showMessageDialog(null, "修改失败！");          //提示修改失败
}
```

秘笈心法

心法领悟 135：在 Oracle 数据库中存储图片的字段类型。

在使用 Oracle 数据库时，为了存储图片，可以将数据表中存储图片的字段设置为 BLOB 类型，这样就可以通过 Java 程序在 Oracle 数据库中存储图片了，代码如下：

```
FileInputStream in = new FileInputStream(file);
ps.setBinaryStream(1, in, (int) file.length());
```

实例 136　修改 SQLServer 数据库中存储的图片

光盘位置：光盘\MR\136　　初级　　趣味指数：★★★

实例说明

本实例实现了对存储在 SQLServer 数据库中的图片进行修改的功能。运行程序，在“编号：”标签右侧的文本框中输入编号，单击“按编号查询”按钮，将从数据库查询所输入编号的信息，效果如图 6.24 所示，如果需要修改图片，可以单击“选择图片”按钮选择一幅图片，然后单击“修改”按钮，修改 SQLServer 数据库中存储的图片。

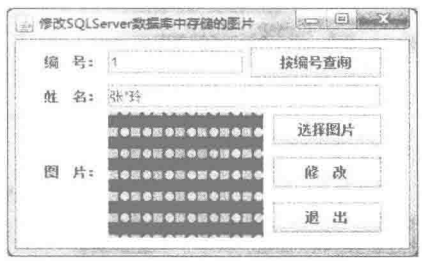


图 6.24　修改 SQLServer 数据库中存储的图片

关键技术

本实例通过建立到 SQLServer 数据库的连接，并通过 PreparedStatement 的 setBinaryStream()方法，实现对 SQLServer 数据库中存储的图片进行修改的功能。

（1）建立到 SQLServer 数据库的连接，代码见实例 130。

（2）通过 PreparedStatement 的 setBinaryStream()方法，对 SQLServer 数据库表中存储的图片进行修改，代码如下：

```
String sql = "update tb_picture set name = ?,picture = ? where id = ?";   //定义修改数据记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                         //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                                     //为第 1 个参数赋值，修改姓名
ps.setBinaryStream(2, in, (int) file.length());                            //为第 2 个参数赋值，修改图片
ps.setInt(3, Integer.parseInt(tf_id.getText()));                           //为第 3 个参数赋值，设置要修改图片的编号
int flag = ps.executeUpdate();                                             //执行 SQL 语句，获得更新记录数
```

说明：上面代码中的 conn 是 SQLServer 数据库的连接对象；in 是图片文件的输入流对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DAO 类，用于加载数据库驱动和建立到 SQLServer 数据库的连接。

（3）在项目中创建一个继承 JFrame 类的 UpdateSQLServerPictureFrame 窗体类，并完成窗体界面的设计，在“按编号查询”按钮的事件中实现从 SQLServer 数据库中查询满足指定编号的记录，并在窗体上的相应控件中显示。

（4）在 UpdateSQLServerPictureFrame 窗体类的“修改”按钮的事件中修改 SQLServer 数据库中满足条件的记录，实现修改图片的功能。“修改”按钮事件的关键代码如下：

```
FileInputStream in = new FileInputStream(file);                    //创建图片的输入流对象
Connection conn = DAO.getConn();                                   //获得数据库连接对象
String sql = "update tb_picture set name = ?,picture = ? where id = ?";  //定义修改数据记录的 SQL 语句
PreparedStatement ps = conn.prepareStatement(sql);                 //创建 PreparedStatement 对象，并传递 SQL 语句
ps.setString(1, name);                                             //为第 1 个参数赋值
ps.setBinaryStream(2, in, (int) file.length());                    //为第 2 个参数赋值
ps.setInt(3, Integer.parseInt(tf_id.getText()));                   //为第 3 个参数赋值
int flag = ps.executeUpdate();                                     //执行 SQL 语句，获得更新记录数
if (flag > 0) {
    JOptionPane.showMessageDialog(null, "修改成功！ ");              //提示修改成功
} else {
    JOptionPane.showMessageDialog(null, "修改失败！ ");              //提示修改失败
}
```

秘笈心法

心法领悟 136：读取 Oracle 数据库中存储的图片。

在使用 Oracle 数据库时，可以在数据表的 BLOB 列存储图片，这就要求在需要时能够从数据表中读取存储的图片。通过 Java 程序从 Oracle 数据表中读取图片的代码如下：

```
Blob img = (Blob) rs.getBlob(1);
if (img != null) {
    ImageIcon icon = new ImageIcon(img.getBytes(1, (int) img.length()));
}
```

6.3 其 他 应 用

实例 137　获取鼠标指针在任意位置的颜色值

光盘位置：光盘\MR\137　　初级　　趣味指数：★★★

实例说明

本实例实现了获取鼠标指针在屏幕上任意位置颜色值的功能。运行程序，将在窗体上显示鼠标指针在屏幕上当前位置的横纵坐标和颜色的 RGB 值，效果如图 6.25 所示。

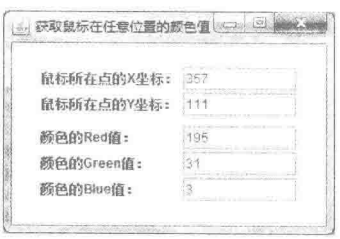


图 6.25　获取鼠标指针在任意位置的颜色值

关键技术

本实例通过 MouseInfo 类和 Robot 类实现了获得鼠标指针在屏幕上任意位置的坐标和颜色值的操作。

（1）使用 MouseInfo 类的 getPointerInfo()方法，可以获得鼠标指针当前位置的 PointerInfo 对象。getPointerInfo()方法的定义如下：

```
public static PointerInfo getPointerInfo()
```

参数说明

返回值：鼠标指针在屏幕上当前位置的 PointerInfo 对象。

（2）使用 Robot 类的 getPixelColor()方法获得屏幕上指定位置的 Color 对象，该方法的定义如下：

```
public Color getPixelColor(int x, int y)
```

参数说明

❶ x：获得颜色值位置的 x 坐标。

❷ y：获得颜色值位置的 y 坐标。

❸ 返回值：屏幕上指定位置的 Color 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 GainAnyPointColorFrame 窗体类，在窗体中添加标签和文本框，完成界面的设计。

（3）在 GainAnyPointColorFrame 窗体类中定义一个实现 Runnable 接口的 GainColor 线程类，在 run()方法中完成获得鼠标指针位置和颜色值的操作，并显示在窗体上的相应控件中。GainColor 类的 run()方法代码如下：

```
while (true) {
    PointerInfo mi = MouseInfo.getPointerInfo();        //鼠标指针当前位置的 PointerInfo 对象
    Point p = mi.getLocation();                         //获得屏幕上表示指针坐标的 Point 对象
    int x = p.x;                                        //获得 x 坐标
    int y = p.y;                                        //获得 y 坐标
    try {
        Robot robot = new Robot();                      //创建 Robot 对象
        Color color = robot.getPixelColor(x, y);        //获得屏幕指定位置的颜色对象
        int r = color.getRed();                         //获得颜色的 R 值
        int g = color.getGreen();                       //获得颜色的 G 值
        int b = color.getBlue();                        //获得颜色的 B 值
        tf_x.setText(String.valueOf(x));                //显示 x 坐标值
        tf_y.setText(String.valueOf(y));                //显示 y 坐标值
        tf_red.setText(String.valueOf(r));              //显示颜色的 R 值
        tf_green.setText(String.valueOf(g));            //显示颜色的 G 值
        tf_blue.setText(String.valueOf(b));             //显示颜色的 B 值
        Thread.sleep(10);                               //线程休眠 10 毫秒
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

（4）在 GainAnyPointColorFrame 窗体类的构造方法中实现初始化界面和创建线程对象，其关键代码如下：

```
GainColor gc = new GainColor();                         //创建 GainColor 对象
Thread thread = new Thread(gc);                         //创建线程对象
thread.start();                                         //启动线程对象
```

秘笈心法

心法领悟 137：判断是否移动了鼠标。

在某些特殊的程序（如屏幕保护程序）中，需要判断是否在屏幕上移动了鼠标，为此可以使用 MouseInfo 类的 getPointerInfo()方法获得 PointerInfo 对象，然后通过该对象，在指定的时间间隔内分别调用 getLocation()方法获得两个 Point 对象，并比较这两个对象是否相同，从而可以判断是否在屏幕上移动了鼠标。

实例 138 图片浏览器

光盘位置：光盘\MR\138

中级

趣味指数：★★★

实例说明

本实例实现了图片浏览器的功能。运行程序，单击“选择”按钮，在弹出的打开文件对话框中选择一幅图片，将在窗体上显示该图片，如图 6.26 所示，选择图片后，就可以通过窗体下方的按钮浏览该图片所在文件夹中所有相同格式的图片。

图 6.26 图片浏览器

关键技术

本实例主要是通过 File 类的 list()方法，将指定文件夹中符合格式要求的图片放到 List 集合列表中，然后通过索引值对 List 集合列表中的文件名进行浏览，从而实现图片浏览器的功能。

（1）使用 File 类的 list()方法，可以对指定目录下满足条件的记录进行过滤，返回一个符合条件的所有文件的字符串数组，该方法的定义如下：

```
public String[] list(FilenameFilter filter)
```

参数说明

❶ filter：文件名过滤器，用于过滤符合条件的文件名。

❷ 返回值：字符串数组，是符合过滤条件的所有文件名。

（2）将过滤出来的符合条件的文件名添加到 List 集合列表中，代码如下：

```
for (int i=0;i<fileNames.length;i++){                    //遍历数组中的文件名
    if (fileNames[i].equals(currentFileName)){           //判断是否为当前文件
        currentFileIndex = i;                            //记忆当前文件的索引值
    }
    fileNameList.add(fileNames[i]);                      //将文件添加到集合列表中
}
```

（3）通过索引值对 List 集合列表中的文件名进行浏览，实现浏览图片功能的代码如下：

```
imgFile = new File(filePath+"/"+fileNameList.get(currentFileIndex).toString());   //创建当前索引值对应图片的 File 对象
try {
    img = ImageIO.read(imgFile);                         //创建当前图片的图像对象
    imgPanel.repaint();                                  //调用 paint()方法，显示图片
} catch (IOException e1) {
    e1.printStackTrace();
}
```

说明：上面代码中，第一条语句中的 fileNameList.get(currentFileIndex).toString()，就是从列表中获得的当前索引的图片文件名。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 BrowerPictureFrame 窗体类。

（3）在 BrowerPictureFrame 窗体类中创建内部面板类 DrawImagePanel，用于在窗体上显示当前图片。

（4）在 BrowerPictureFrame 窗体类中为“上一张”按钮添加事件代码，实现向上浏览图片的功能。其事件代码如下：

```
button_2.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        currentFileIndex--;                                                    //调整当前图片的索引值
        if (currentFileIndex < 0) {
            currentFileIndex = fileNameList.size() - 1;                        //当前图片索引为最后一张图片的索引
        }
        imgFile = new File(filePath+"/"+fileNameList.get(currentFileIndex).toString());   //创建当前索引值对应图片的 File 对象
        try {
            img = ImageIO.read(imgFile);                                       //创建当前图片的图像对象
            imgPanel.repaint();                                                //调用 paint()方法，显示图片
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
});
```

（5）在 BrowerPictureFrame 窗体类中为“下一张”按钮添加事件代码，实现向下浏览图片的功能。其事件代码如下：

```
button_3.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        currentFileIndex++;                                                    //调整当前图片的索引值
        if (currentFileIndex > fileNameList.size() - 1) {
            currentFileIndex = 0;                                              //当前图片索引为第 1 张图片的索引
        }
        imgFile = new File(filePath+"/"+fileNameList.get(currentFileIndex).toString());   //创建当前索引值对应图片的 File 对象
        try {
            img = ImageIO.read(imgFile);                                       //创建当前图片的图像对象
            imgPanel.repaint();                                                //调用 paint()方法
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
});
```

秘笈心法

心法领悟 138：可以旋转图片的浏览器。

本实例实现的图片浏览器只能对指定文件夹中的图片进行浏览，为此可以为程序添加旋转图像的功能，当浏览图像的角度不适合时，可以将图像旋转一定的角度，以方便图片的浏览。

实例 139　转换图片格式

光盘位置：光盘\MR\139　　高级　　趣味指数：★★

实例说明

本实例实现了图片格式的转换。运行程序，单击“选择图片”按钮选择一幅图片，效果如图 6.27 所示，然后在窗体下方的下拉列表框中选择图片转换后的格式，单击“保存”按钮完成图片格式的转换。

图 6.27　转换图片格式

关键技术

本实例使用 ImageIO 类的 write()方法，将绘制有原图片的 BufferedImage 对象以指定格式存储到磁盘上，实现了转换图片格式的功能。

（1）创建原图片的 BufferedImage 对象，实现代码如下：

```
buffImage = ImageIO.read(imgFile);                    //构造 BufferedImage 对象
```

说明：上面代码中的 imgFile 是原图片的 File 对象；buffImage 是在类的成员声明区定义的 BufferedImage 对象。

（2）以指定格式将原图片的 BufferedImage 对象存储到磁盘上，实现代码如下：

```
ImageIO.write(buffImage, extName, file);              //将缓冲图像保存到磁盘
```

说明：上面代码中的 buffImage 是原图片的 BufferedImage 对象；extName 是转换后图片的扩展名；file 是转换格式后的图片存储位置的 File 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ConversionPictureFormatFrame 窗体类，在窗体类中定义一个内部面板类 DrawImagePanel，用于在窗体上显示选择的原图片，代码如下：

```
class DrawImagePanel extends JPanel {
    public void paint(Graphics g) {
        g.clearRect(0, 0, getWidth(), getHeight());
        g.drawImage(buffImage, 0, 0, this);           //绘制指定的图片
    }
}
```

（3）在项目中创建一个继承 JFrame 类的 ConversionPictureFormatFrame 窗体类，并完成窗体的设计，然后在“保存”按钮的事件中完成图片格式的转换，其事件代码如下：

```
try {
    String extName = (String)comboBox.getSelectedItem();                    //新格式的扩展名，不含点
    String pathAndName = pathAndFileName.substring(0, pathAndFileName.lastIndexOf(".") + 1)+extName;    //转换后图片的完整路径和文件名
    File file = new File(pathAndName);                                      //创建转换后图片的 File 对象
    ImageIO.write(buffImage, extName, file);                                //将缓冲图像保存到磁盘
    File oldFile = new File(pathAndFileName);                               //原图片的 File 对象
    oldFile.delete();                                                       //删除原图片文件
    JOptionPane.showMessageDialog(null, "文件格式更改成功！");               //显示提示信息
} catch (IOException e1) {
    JOptionPane.showMessageDialog(null, "保存失败\n" + e1.getMessage());    //显示提示信息
}
```

秘笈心法

心法领悟 139：判断图片格式是否可以写入磁盘。

使用 ImageIO 类并不能在磁盘上写入所有格式的图片，为此，可以使用 ImageIO 类的 getWriterFormatNames()方法获得所有可以写入磁盘的图片格式，然后再对要写入的格式进行判断。

实例 140　绘制石英钟

高级

趣味指数：★★★★★

光盘位置：光盘\MR\140

实例说明

本实例绘制了一个石英钟，在石英钟上显示有时针、分针、秒针、日期和星期。运行程序，效果如图 6.28 所示。

图 6.28　石英钟

关键技术

本实例中，Calendar 类获得日历对象，进而获得毫秒值、秒值、分值和小时值，然后根据这几个值计算出秒针、分针和时针的角度，最终计算出秒针、分针和时针指向点的坐标，从而实现了相应功能。

（1）通过 Calendar 日历对象，获得毫秒值、秒值、分值和小时值，代码如下：

```
Calendar calendar = Calendar.getInstance();                    //获取日历对象
int millisecond = calendar.get(MILLISECOND);                   //获取毫秒值
int sec = calendar.get(SECOND);                                //获取秒值
int minutes = calendar.get(MINUTE);                            //获取分值
int hours = calendar.get(HOUR);                                //获取小时值
```

（2）根据毫秒值、秒值、分值和小时值计算出秒针、分针和时针的角度，代码如下：

```
double secAngle = (60 - sec) * 6 - (millisecond / 150);        //秒针角度
int minutesAngle = (60 - minutes) * 6;                         //分针角度
int hoursAngle = (12 - hours) * 360 / 12 - (minutes / 2);      //时针角度
```

（3）根据秒针、分针和时针的角度计算出秒针、分针和时针指向点的坐标，代码如下：

```
int secX = (int) (secLen * Math.sin(Math.toRadians(secAngle)));                  //秒针指向点的 x 坐标
int secY = (int) (secLen * Math.cos(Math.toRadians(secAngle)));                  //秒针指向点的 y 坐标
int minutesX = (int) (minuesLen * Math.sin(Math.toRadians(minutesAngle)));       //分针指向点的 x 坐标
int minutesY = (int) (minuesLen * Math.cos(Math.toRadians(minutesAngle)));       //分针指向点的 y 坐标
int hoursX = (int) (hoursLen * Math.sin(Math.toRadians(hoursAngle)));            //时针指向点的 x 坐标
int hoursY = (int) (hoursLen * Math.cos(Math.toRadians(hoursAngle)));            //时针指向点的 y 坐标
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JPanel 类的 ClockPanel 面板类，重写 paint()方法实现石英钟的绘制，paint()方法的关键代码如下：

```
//分别绘制时针、分针、秒针
g2.setStroke(HOURS_POINT_WIDTH);                               //设置时针的宽度
g2.setColor(Color.BLACK);                                      //设置时针的颜色
g2.drawLine(centerX, centerY, centerX - hoursX, centerY - hoursY);   //绘制时针
```

```
g2.setStroke(MINUETES_POINT_WIDTH);                                    //设置分针的宽度
if (minutesAngle != hoursAngle)                                        //分针、时针重叠变色
    g2.setColor(new Color(0x2F2F2F));                                  //设置未重叠时的颜色
else {
    g2.setColor(Color.GREEN);                                          //设置重叠时的颜色
}
g2.drawLine(centerX, centerY, centerX - minutesX, centerY - minutesY); //绘制分针
g2.setStroke(SEC_POINT_WIDTH);                                         //设置秒针的宽度
if (secAngle != hoursAngle && secAngle != minutesAngle)                //分针、时针、秒针重叠变色
    g2.setColor(Color.ORANGE);                                         //设置未重叠时的颜色
else {
    g2.setColor(Color.GREEN);                                          //设置重叠时的颜色
}
//绘制 3 个指针的中心圆和秒针
g2.fillOval(centerX - 5, centerY - 5, 10, 10);                         //绘制中心圆
g2.drawLine(centerX, centerY, centerX - secX, centerY - secY);         //绘制秒针
g2.drawLine(centerX + 1, centerY + 1, centerX - secX + 1, centerY - secY + 1); //绘制秒针
```

（3）创建继承 JDialog 类的 ClockFrame 对话框窗体类，在该类的构造方法中取消窗体装饰，使其不显示标题栏和边框，并总在最前显示，代码如下：

```
setUndecorated(true);                                                  //取消窗体修饰
setAlwaysOnTop(true);                                                  //窗体置顶
```

说明：由于篇幅原因只给出了部分代码，如果需要对本实例有更深入的了解，请查看源程序代码。

秘笈心法

心法领悟 140：为石英钟添加系统托盘。

为了使本实例更完善，可以为石英钟添加系统托盘，方法是使用 SystemTray 类的 getSystemTray()方法获得系统托盘，然后使用 TrayIcon 类创建托盘图标对象，并使用 SystemTray 类的 add()方法将托盘图标添加到系统托盘。

实例 141 画图程序

光盘位置：光盘\MR\141

高级

趣味指数：★★★★★

实例说明

本实例使用 Java 的绘图技术实现一个画图程序。运行程序，可以通过菜单和工具栏两种方式选择背景色、前景色、画笔的粗细，还可以使用橡皮进行擦除，也可以清除整个画板的内容，如图 6.29 所示就是本实例中绘制的图片。

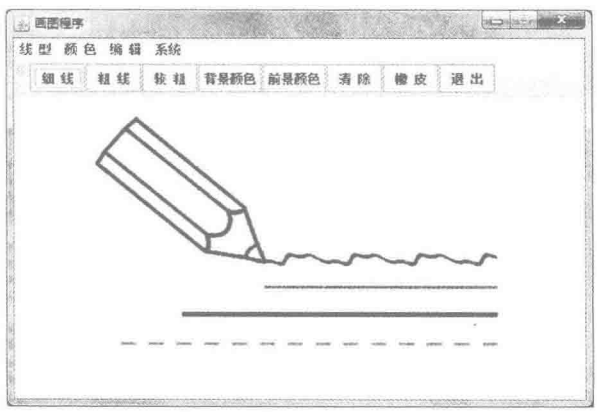


图 6.29 画图程序

关键技术

本实例通过 BufferedImage 对象进行画图，然后在自定义的 Canvas 对象上显示绘制的内容，从而实现了画图功能。

（1）在 BufferedImage 对象上画图，是通过鼠标的拖动事件实现的，其中的关键代码如下：

```
if (x > 0 && y > 0) {
    g.drawLine(x, y, e.getX(), e.getY());                    //在鼠标指针经过处画直线
}
x = e.getX();                                                //上一次鼠标绘制点的横坐标
y = e.getY();                                                //上一次鼠标绘制点的纵坐标
```

说明：在拖动鼠标的过程中，会重复执行上述代码，由于鼠标指针移动的起始点和终点距离非常小，相当于一个点，所以可以实现沿着鼠标指针的轨迹画图的功能。

（2）创建 Canvas 类的子类 DrawPictureCanvas，用于显示 BufferedImage 对象上绘制的内容，其中的关键代码如下：

```
public void setImage(Image image) {                         //传递 BufferedImage 对象
    this.image = image;                                      //为成员变量赋值
}
public void paint(Graphics g) {                              //重写 paint()方法，在画布上绘制 BufferedImage 对象
    g.drawImage(image, 0, 0, null);                          //在画布上绘制图像
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 Canvas 类的 DrawPictureCanvas 画布类，用于在其上绘制缓冲图像对象，该缓冲图像对象就是用户绘制的内容，从而实现显示所绘制内容的功能，该类的关键代码如下：

```
public class DrawPictureCanvas extends Canvas {
    private Image image = null;                              //定义 Image 对象的引用
    public void setImage(Image image) {
        this.image = image;                                  //为成员变量赋值
    }
    public void paint(Graphics g) {
        g.drawImage(image, 0, 0, null);                      //在画布上绘制图像
    }
}
```

（3）在项目中创建一个继承 Canvas 类的 DrawPictureFrame 窗体类，用于显示完成绘图的各种操作，其中实现绘图的关键代码，是画布类 DrawPictureCanvas 的实例 canvas 的 3 个鼠标事件，即拖动、移动和释放。

☑　拖动事件的代码

```
public void mouseDragged(final MouseEvent e) {
    if (rubber) {                                            //橡皮标识为 true，表示使用橡皮
        if (x > 0 && y > 0) {
            g.setColor(backgroundColor);                     //用背景色设置绘图上下文对象的颜色
            g.fillRect(x, y, 10, 10);                        //擦除鼠标指针经过位置的图像
        }
        x = e.getX();                                        //获得鼠标指针在画布上的横坐标
        y = e.getY();                                        //获得鼠标指针在画布上的纵坐标
    } else {                                                 //橡皮标识为 false，表示画图
        if (x > 0 && y > 0) {
            g.drawLine(x, y, e.getX(), e.getY());            //在鼠标指针经过处画直线
        }
        x = e.getX();                                        //上一次鼠标绘制点的横坐标
        y = e.getY();                                        //上一次鼠标绘制点的纵坐标
    }
    canvas.repaint();                                        //更新画布
}
```

☑　移动事件的代码

```
public void mouseMoved(final MouseEvent arg0) {
    if (rubber) {
```

```
            setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));          //设置鼠标指针的形状
        } else {
            setCursor(Cursor.getPredefinedCursor(Cursor.CROSSHAIR_CURSOR));     //设置鼠标指针的形状
        }
    }
```

☑ 释放事件的代码

```
public void mouseReleased(final MouseEvent arg0) {
    x = -1;                                                                     //上一次鼠标绘制点的横坐标
    y = -1;                                                                     //上一次鼠标绘制点的纵坐标
}
```

秘笈心法

心法领悟 141：完善画图程序。

本实例只能通过线条进行绘图，为此可以完善该画图程序，使其可以绘制圆形、方形等内容，甚至可以绘制图像，这可以通过定义多个标记变量来实现。例如定义一个画圆形的 boolean 变量，当其为 true 时，可以绘制圆形并将其他标记变量赋值为 false，不可以用鼠标绘制其他内容；当其为 false 时，才可以绘制其他内容。

实例 142 屏幕抓图程序

光盘位置：光盘\MR\142　　高级　　趣味指数：★★★★★

实例说明

本实例实现了屏幕抓图功能。运行程序，可以用鼠标选择屏幕上需要抓取的内容，并以 zzkkee.jpg 保存到 C 盘根目录。抓取 Windows 7 桌面上时钟和日历的效果，如图 6.30 所示，如果需要退出程序，只需在屏幕上右击即可。

图 6.30　抓取屏幕上时钟和日历的效果

关键技术

本程序使用窗体透明技术把窗体设置为透明，然后使用 Robot 类的 createScreenCapture()方法捕获屏幕，实现屏幕抓图的功能。

（1）程序首先取消窗体装饰，然后使用 AWTUtilities 类的 setWindowOpacity()方法将窗体的透明度设置为 0.01f，即使窗体接近完全透明，可以透出桌面，因此可以响应鼠标事件，并且可以抓取桌面图片。取消窗体装饰和设置窗体透明度的代码如下：

```
setUndecorated(true);                                                           //取消窗体修饰
AWTUtilities.setWindowOpacity(this, 0.01f);                                     //设置窗体透明
```

（2）使用 Robot 类的 createScreenCapture()方法，可以根据需要捕获屏幕上的指定区域。捕获屏幕指定区域的代码如下：

```
buffImage = robot.createScreenCapture(rect);                                    //获得缓冲图像对象
```

说明：上面代码中的 robot 是 Robot 对象；rect 表示需要捕获屏幕区域的 Rectangle 对象；buffImage 用于存储捕获的屏幕区域。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 CaptureScreenImageFrame 窗体类，在构造方法中添加取消窗体装饰、设置窗体透明度和捕获桌面图片的功能，主要代码如下：

```
setUndecorated(true);                                          //取消窗体修饰
AWTUtilities.setWindowOpacity(this, 0.01f);                    //设置窗体透明
getContentPane().add(partZoomInPanel, BorderLayout.CENTER);
robot = new Robot();                                           //创建 Robot 对象
rect = new Rectangle(0, 0, dim.width, dim.height);             //创建 Rectangle 对象
buffImage = robot.createScreenCapture(rect);                   //获得缓冲图像对象
```

（3）在 CaptureScreenImageFrame 窗体类的鼠标释放事件中，完成抓取屏幕图像并保存到磁盘的操作。鼠标释放事件的主要代码如下：

```
releaseX = e.getXOnScreen() - 1;                               //鼠标释放点在屏幕上的 x 坐标减 1，即去除选择线
releaseY = e.getYOnScreen() - 1;                               //鼠标释放点在屏幕上的 y 坐标减 1，即去除选择线
try {
    if (releaseX - pressX > 0 && releaseY - pressY > 0) {
        Rectangle rect = new Rectangle(pressX, pressY, releaseX
                - pressX, releaseY - pressY);                  //创建 Rectangle 对象
        buffImage = robot.createScreenCapture(rect);           //获得缓冲图像对象
        FileOutputStream out = new FileOutputStream("c:/zzkkee.jpg"); //保存位置的输出流对象
        ImageIO.write(buffImage, "jpg", out);                  //写入磁盘
        out.flush();
        out.close();
    }
} catch (FileNotFoundException e2) {
    e2.printStackTrace();
} catch (IOException e3) {
    e3.printStackTrace();
}
```

说明：由于篇幅原因，这里没有将所有代码一一给出，如果需要对本实例有更进一步的了解，可以参考源程序代码。

秘笈心法

心法领悟 142：手动保存抓取的屏幕图像。

本实例将抓取的图像直接保存到磁盘中指定的位置，并且每次都以相同的文件名存储，所以只能看到最后捕获的图片。如果希望将抓取的屏幕图像全部保存起来，可以在鼠标释放事件中完成抓取图像操作后，弹出文件保存对话框，手动指定保存位置。

实例 143　屏幕放大镜

光盘位置：光盘\MR\143　　高级　　趣味指数：★★★★★

实例说明

本实例实现了一个屏幕放大镜程序。运行程序，通过鼠标选择屏幕上需要放大的内容，即可放大所选择的内容。例如选择 Windows 7 桌面上的日历，将放大显示该日历，效果如图 6.31 所示，如果需要退出程序，只需要在屏幕上右击即可。

图 6.31　屏幕上的日历放大之前和之后的效果

关键技术

本实例通过 Robot 类捕获屏幕图像，然后使用绘图上下文的 drawImage()方法，对在屏幕上捕获的图像进行放大，并绘制到窗体上。

（1）使用 Robot 类的 createScreenCapture()方法，可以根据需要捕获屏幕上的指定区域，捕获屏幕指定区域的代码如下：

```
Rectangle rect = new Rectangle(pressX, pressY, releaseX - pressX, releaseY - pressY);        //创建 Rectangle 对象
zoomBuffImage = robot.createScreenCapture(rect);                                            //获得缓冲图像对象
```

说明：上面代码中的 pressX 和 pressY 是鼠标在屏幕上按下点的横、纵坐标值；releaseX 和 releaseY 是鼠标释放点的横、纵坐标值；zoomBuffImage 用于存储捕获到的屏幕图像。

（2）使用绘图上下文的 drawImage()方法，对在屏幕上捕获的图像进行放大，并绘制到窗体上，代码如下：

```
//绘制放大后的内容，即在水平和垂直方向分别放大 1.5 倍
g2.drawImage(zoomBuffImage, zoomX, zoomY, (int) ((releaseX - pressX) * 1.5f), (int) ((releaseY - pressY) * 1.5f), this);
```

说明：这段代码中，zoomX 和 zoomY 是放大后图像左上角绘制点的横、纵坐标值；(int) ((releaseX - pressX) * 1.5f)和(int) ((releaseY - pressY) * 1.5f)分别表示对图像在水平和垂直方向上各放大 1.5 倍，其余参数与本实例关键技术中（1）的内容相同。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ScreenZoomInFrame 窗体类，在该类的成员声明区定义如下变量，用于标识是否显示选择区域的矩形，以及是否放大选择的图像，代码如下：

```
private boolean flag = false;                                   //声明标记变量，为 true 时显示选择区域的矩形，否则不显示
private boolean mouseFlag = false;                              //进行放大图像的标记变量，为 true 时进行放大，否则不放大
```

（3）在 ScreenZoomInFrame 窗体类中创建一个内部面板类 PartZoomInPanal，用于在屏幕上绘制放大后的图像，该内部类的代码如下：

```
class PartZoomInPanel extends JPanel {                          //创建面板类
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        g2.drawImage(buffImage, 0, 0, this);                    //绘制图像
        g2.setColor(Color.BLACK);
        if (flag) {
            float[] arr = { 5.0f };                             //创建虚线模式的数组
            BasicStroke stroke = new BasicStroke(1, BasicStroke.CAP_BUTT,
                    BasicStroke.JOIN_BEVEL, 1.0f, arr, 0);      //创建宽度是 1 的平头虚线笔画对象
            g2.setStroke(stroke);                               //设置笔画对象
            g2.drawRect(pressPanelX, pressPanelY, releaseX - pressX,
                    releaseY - pressY);                         //绘制矩形选区
        }
        if (mouseFlag) {                                        //条件为真
            int zoomX = pressPanelX - (releaseX - pressX) / 4;  //放大内容绘制点的 x 坐标
            int zoomY = pressPanelY - (releaseY - pressY) / 4;  //放大内容绘制点的 y 坐标
            if (zoomX <= 0) {
                zoomX = 0;                                      //坐标值小于等于 0，让坐标值为 0
```

```
                }
                if (zoomY <= 0) {
                    zoomY = 0;                                    //坐标值小于等于 0，让坐标值为 0
                }
                g2.drawImage(zoomBuffImage, zoomX, zoomY,
                        (int) ((releaseX - pressX) * 1.5f),
                        (int) ((releaseY - pressY) * 1.5f), this);    //绘制放大后的内容
            }
        }
    }
```

（4）在窗体类 ScreenZoomInFrame 的鼠标释放事件中，捕获在屏幕上选择的图像，该图像就是在内部面板中放大的图像。鼠标释放事件的代码如下：

```
public void mouseReleased(final MouseEvent e) {                  //鼠标释放事件
    releaseX = e.getXOnScreen() - 1;                              //鼠标释放点在屏幕上的 x 坐标减 1，即去除选择线
    releaseY = e.getYOnScreen() - 1;                              //鼠标释放点在屏幕上的 y 坐标减 1，即去除选择线
    if (releaseX - pressX > 0 && releaseY - pressY > 0) {
        Rectangle rect = new Rectangle(pressX, pressY, releaseX
                - pressX, releaseY - pressY);                     //创建 Rectangle 对象
        zoomBuffImage = robot.createScreenCapture(rect);          //获得缓冲图像对象
    }
    flag = false;                                                 //为标记变量赋值为 false
    mouseFlag = true;                                             //标记为 true，进行放大
    partZoomInPanel.repaint();                                    //调用 paint()方法，实现放大
}
```

秘笈心法

心法领悟 143：局部放大并保存为图片。

通过本实例，可以实现对图片进行局部放大，并绘制到 BufferedImage 对象上，然后使用 ImageIO 类的 write() 方法，将绘制有局部放大效果的 BufferedImage 对象保存到磁盘，实现局部放大并保存为图片的功能。

JFreeChart 图表篇

第 7 章

JFreeChart 基本操作

- JFreeChart 基础操作
- 设置图表背景
- 处理图表的边框
- 修改图表的图示

7.1　JFreeChart 基础操作

实例 144	饼图	初级
	实例位置：光盘\MR\144	趣味指数：★★★

实例说明

JFreeChart 是一款开源的 Java 图表绘制工具，其图表种类丰富、接口通俗易懂、支持多种显示方式，如 Application、Applets、Servlet 和 JSP 等。本实例创建一个简单的饼图，图中有 A、B、C 3 个分类，运行效果如图 7.1 所示。

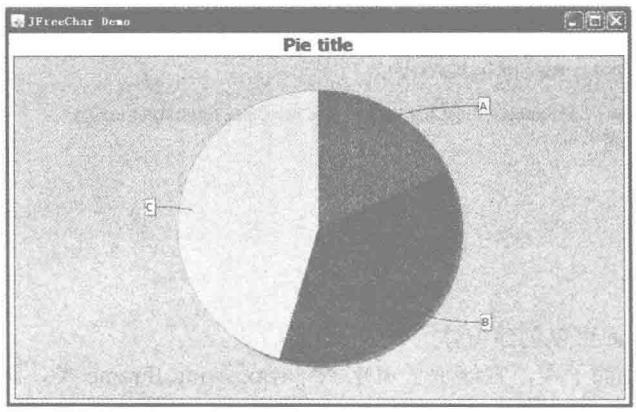


图 7.1　简单饼图

关键技术

（1）DefaultPieDataset 可以创建饼图的数据集合，使用 setValue()方法可以为数据集合添加数据，其语法如下：

```
public void setValue(Comparable key, double value)
```

参数说明

❶ key：指要为饼图添加的类别。

❷ value：表示与类别相对应的值。

（2）ChartFactory 是一个图形工厂，它有很多方法可以把各种数据集合转换成 JFreeChart 对象，这里使用的 createPieChart()方法使用饼图数据集合创建一个 JFreeChart 对象，其语法如下：

```
public static JFreeChart createPieChart(String title, PieDataset dataset, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：表示饼图的标题。

❷ dataset：表示饼图的数据集合。

❸ legend：表示是否使用图示。

❹ tooltips：表示是否生成工具栏提示。

❺ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码如下：

```
private PieDataset getPieDataset() {
        //创建一个饼图表的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图表的数据集添加数据
        dataset.setValue("A", 200);
        dataset.setValue("B", 400);
        dataset.setValue("C", 500);
        return dataset;
}
```

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，代码如下：

```
public JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("Pie title",dataset, false, false, false);
        return chart;
}
```

（4）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体，代码如下：

```
public static void main(String[] args) {
        PieChartDemo1 pieChartDemo1 = new PieChartDemo1();
        //生成窗体
        ChartFrame chartFrame = new ChartFrame("JFreeChar Demo",pieChartDemo1.getJFreeChart());
        //调整窗口的大小，以适合图表显示
        chartFrame.pack();
        //显示窗体
        chartFrame.setVisible(true);
}
```

秘笈心法

心法领悟 144：ChartFrame 可以创建窗体。

ChartFrame 是 JFreeChart 的子类，它继承了 JDK 中 javax.swing.JFrame 类，同时又有自己的实现，只要把 JFreeChart 的对象传入 ChartFrame 构造方法，ChartFrame 类就可以自动地为 JFreeChart 创建一个展示的窗体。

实例 145　显示图示

光盘位置：光盘\MR\145　　初级　　趣味指数：★★★

实例说明

一般情况下，图表都带有图示，JFreeChart 也支持图示的生成。本实例在实例 144 的基础上进行了改动，实现在饼图中显示图示。程序运行效果如图 7.2 所示。

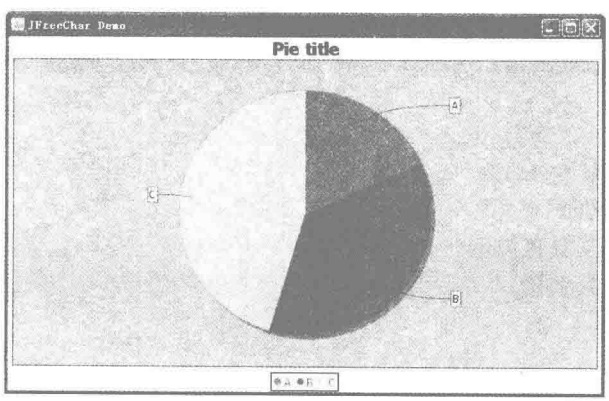


图 7.2　含有图示的饼图

关键技术

在生成 JFreeChart 时，可以使用 ChartFactory 的 createPieChart()方法设置图示是否显示，语法如下：

```
public static JFreeChart createPieChart(String title, PieDataset dataset, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：表示饼图的标题。

❷ dataset：表示饼图的数据集合。

❸ legend：表示是否使用图示。

❹ tooltips：表示是否生成工具栏提示。

❺ urls：表示是否生成 URL 链接。

其中，参数 legend 设置为 true，图片上就可以显示出图示。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 144。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并设置 legend 参数为 true，代码如下：

```
public JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("Pie title", dataset,true, false, false);
        return chart;
}
```

（4）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体，代码如下：

```
public static void main(String[] args) {
        ChartDemo2 demo = new ChartDemo2();
        //生成窗体
        ChartFrame chartFrame = new ChartFrame("JFreeChar Demo",demo.getJFreeChart());
        //调整窗口的大小，以适合图表显示
        chartFrame.pack();
        chartFrame.setVisible(true);
}
```

秘笈心法

心法领悟 145：createPieChart()方法中的 legend 参数表示是否生成图示。

ChartFactory 中还有很多类似于 createPieChart(String title，PieDataset dataset，boolean legend，boolean tooltips，boolean urls)的方法可以创建各种图表，其中，所有 legend 的参数都是用来表示是否生成图示。

实例 146　工具栏提示

光盘位置：光盘\MR\146　　初级　　趣味指数：★★★

实例说明

不管是在网页还是在应用程序中，有时把鼠标指针放在某一位置上，鼠标指针附近会弹出一个工具栏提示，介绍该位置上的详细情况。JFreeChart 也可以为图表生成这种提示，提示的内容是图表上具体的数值和百分比，运行效果如图 7.3 所示。

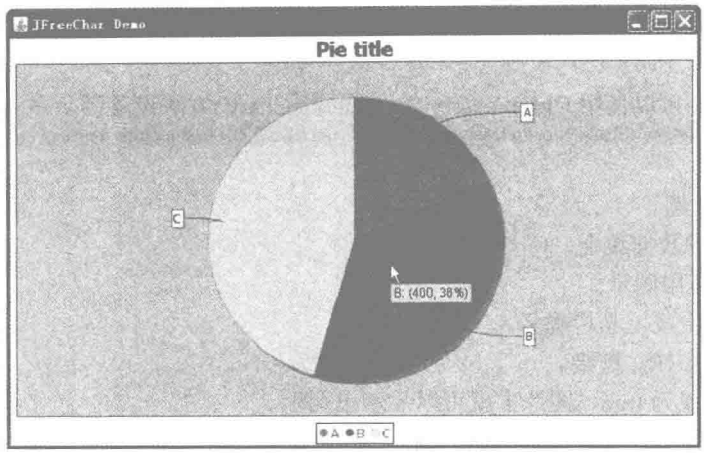


图 7.3　图表工具栏提示

关键技术

在 ChartFactory 类的 createPieChart()方法中，可以设置是否显示工具栏的提示，createPieChart()方法的语法见实例 144，其中，当参数 tooltips 为 true 时，显示工具栏提示；为 false 时不显示，代码如下：

```
public JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("Pie title", dataset,true, true, false);
        return chart;
}
```

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 144。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并设置 tooltips 参数为 true，代码如下：

```
public JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("Pie title", dataset,true, true, false);
        return chart;
}
```

（4）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体，代码如下：

```
public static void main(String[] args) {
        ChartDemo3 demo = new ChartDemo3();
        //生成窗体
        ChartFrame chartFrame = new ChartFrame("JFreeChar Demo",demo.getJFreeChart());
        //调整窗口的大小，以适合图表显示
        chartFrame.pack();
        chartFrame.setVisible(true);
}
```

秘笈心法

心法领悟 146：createPieChart()方法中的 tooltips 参数表示是否生成工具栏提示。

在 ChartFactory 中，createPieChart(String title, PieDataset dataset, boolean legend, boolean tooltips, boolean urls)方法用于创建饼图，参数 tooltips 为 true 时可以显示饼图的工具栏提示，ChartFactory 中其他的柱形图、线形图、区域图等都有 tooltips 参数，都可以用来控制工具栏提示是否启用。

实例 147　乱码问题

光盘\MR\147

中级

趣味指数：★★★

实例说明

JFreeChart 是国外的开源框架，图表中使用的默认字体是 SansSerif，在生成图表时，如果使用汉字，就会产生乱码，运行效果如图 7.4 所示。

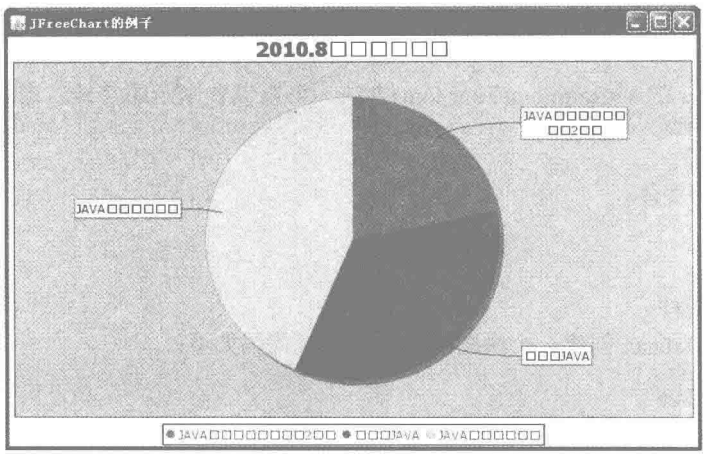


图 7.4　饼图中的乱码

如果希望在 JFreeChart 中使用汉字，需要重新设置汉字的字体，如“宋体”“黑体”“楷体”等。本实例的饼图中，图表标题、图表本身还有图示处需要输入文字，将这 3 个地方的字体都设置为“宋体”，运行效果如图 7.5 所示。

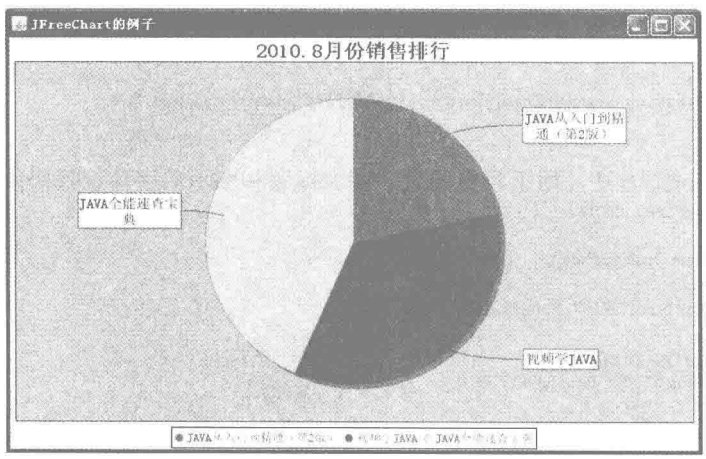


图 7.5　饼图中的汉字

关键技术

（1）使用 JFreeChart 的 getPlot()方法可以获取 PiePlot 对象，语法如下：

```
public Plot getPlot()
```

（2）使用 PiePlot 的 setLabelFont()方法，可以设置图表本身的字体，语法如下：

```
public void setLabelFont(Font font)
```

参数说明

font：表示图表的字体。

（3）使用 JFreeChart 的 getTitle()方法，可以获取 TextTitle 对象，语法如下：

```
public TextTitle getTitle()
```

（4）使用 TextTitle 的 setFont()方法，可以设置图表标题的字体，语法如下：

```
public void setFont(Font font)
```

参数说明

font：表示图表标题的字体。

（5）使用 JFreeChart 的 getLegend()方法，可以获取图示，语法如下：

```
public LegendTitle getLegend()
```

（6）使用 LegendTitler 的 setItemFont(Font font)方法，可以设置图示的字体，语法如下：

```
public void setItemFont(Font font)
```

参数说明

font：表示图表图示的字体。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码如下：

```
private PieDataset getPieDataset() {
        //创建一个饼图表的数据集
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向饼图表的数据集添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1000);
        return dataset;
}
```

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，代码如下：

```
public JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        return chart;
}
```

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码如下：

```
public void setPiePoltFont(JFreeChart chart) {
        //图表（饼图）
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置图表字体
        piePlot.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.BOLD, 20));
        //图示
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 12));
}
```

（5）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 147：Font 类中为字体定义各种样式的常量。

设置字体时，使用了 new Font("宋体", Font.PLAIN, 12)构造字体对象，其中，“宋体”表示要设置的字体名

称；12 表示字体的大小；Font.PLAIN 是一个常量，表示字体使用普通的样式，若使用 Font.BOLD 和 Font.ITALIC 来代替该常量，分别表示字体为粗体和斜体。

实例 148　显示数值

光盘位置：光盘\MR\148　　初级　　趣味指数：★★★

实例说明

一般图表中都会显示数值，可能是具体的值，也可能是百分比。在 JFreeChart 的图表中，数值的显示很方便，显示方式也非常灵活。本实例把《Java 从入门到精通（第 2 版）》、《视频学 Java》和《Java 全能速查宝典》的 8 月份销售量统计生成饼图，在图中显示出具体销售量。运行效果如图 7.6 所示。

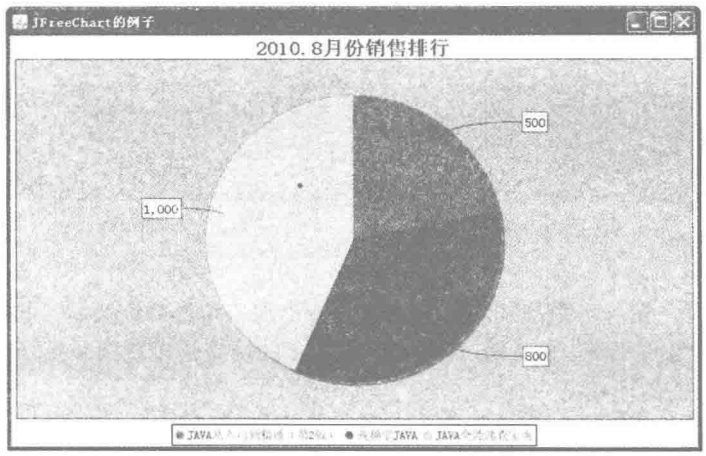


图 7.6　含有数值的图表

关键技术

在默认情况下，JFreeChart 在图表中只显示类别名称而不显示数值，使用 StandardPieSectionLabelGenerator 的构造函数可以重新生成图表标签，语法如下所示：

```
public StandardPieSectionLabelGenerator(String labelFormat)
```

参数说明

labelFormat：表示标签显示的样式。其中：

{0}是 JFreeChart 使用的默认值，使用该值时，表示图表中显示类别名称。

{1}表示图表中显示类别的具体数值。

{2}表示图表中显示当前类别占总数的百分比。

{3}表示图表中所有类别相加的总值。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，代码见实例 147。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setPiePoltNum()方法，修改 JFreeChart 图表，显示数值，代码如下：

```
public void setPiePoltNum(JFreeChart chart) {
        //图表（饼图）
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置饼图标签显示
        piePlot.setLabelGenerator(new StandardPieSectionLabelGenerator("{1}"));
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 148：在 StandardPieSectionLabelGenerator 类中使用{0}、{1}、{2}等字符串参数生成不同样式。

在使用 StandardPieSectionLabelGenerator 构造函数重写 JFreeChart 的标签时，构造函数的字符串参数中可以填写{0}、{1}、{2}、{3}，这些有格式化功能的字符都有其自己的含义。但是该构造函数的字符串参数的使用方法不只这些，还可以把格式化的字符组合起来产生不同的效果。如希望显示“类别名称：类别数值”，可以使用“{0}：{1}”；希望显示“类别百分比：类别总额”，可以使用“{2}：{3}”；希望显示“类别数值（单位）”，可以使用“{1}本”。

实例 149　抗锯齿设置

光盘位置：光盘\MR\149　　中级　　趣味指数：★★★

实例说明

图像在计算机中显示时，受分辨率的制约，物体周围会出现三角形的锯齿状，抗锯齿就是指对图像边缘进行柔化处理。为了使图像边缘更平滑，默认情况下，JFreeChart 的抗锯齿设置都是打开的，如果关闭抗锯齿设置，图像边缘会参差不齐，如图 7.7 所示。

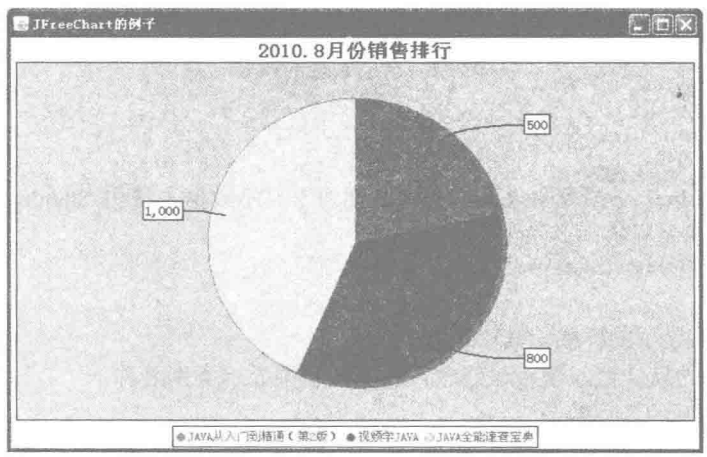


图 7.7　关闭抗锯齿设置

关键技术

获取 JFreeChart 对象，可以使用 JFreeChart 类的 setAntiAlias()方法决定是否设置抗锯齿，语法如下：

```
public void setAntiAlias(boolean flag)
```

参数说明

flag：表示是否设置抗锯齿，当其值为 true 时，表示打开抗锯齿设置，图表各颜色边缘会显示得比较平滑；当其值为 false 时，表示关闭抗锯齿设置，图表各颜色边缘会显示出锯齿。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias() 为 false，关闭抗锯齿设置，代码如下：

```
public JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        //关闭抗锯齿
        chart.setAntiAlias(false);
        return chart;
}
```

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 149：图像的抗拒齿技术原理。

抗拒齿是一种图像处理技术，先对图像物体周围进行采点，再根据采点情况进行处理，不同的抗拒齿技术采点的方式和方法以及处理图像的方法都不同。使用抗拒齿技术虽然可以让图像更美观，但是经过一系列的技术运算，自然会消耗一部分系统资源。

7.2　设置图表背景

实例 150	设置背景图片	初级
	光盘位置：光盘\MR\150	趣味指数：★★★★

实例说明

JFreeChart 的默认背景为灰色，为了使图表在显示时更美观，本实例为图表添加一个背景图片，运行效果如图 7.8 所示。

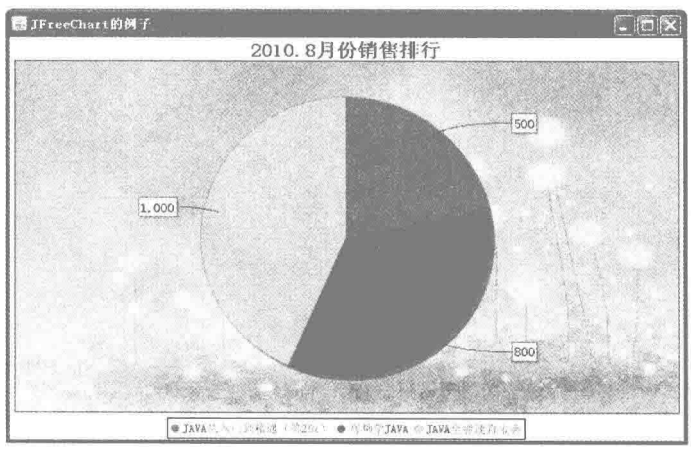


图 7.8　设置图表背景图片

关键技术

使用 PiePlot 类的 setBackgroundImage()方法，可以设置图表的背景图，语法如下：

```
public void setBackgroundImage(Image image)
```

参数说明

image：表示要设置的图表背景图片。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setBackgroundImage()方法，设置背景图片，代码如下：

```
public void setBackgroundImage(JFreeChart chart) {
        Image image = null;
        try {
                //读取图片
                image = ImageIO.read(new File("backgroundImage.jpg"));
        } catch (IOException e) {
                e.printStackTrace();
        }
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置饼图背景图
        piePlot.setBackgroundImage(image);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 150：JFreeChart 中使用 setBackgroundImage 设置背景图。

在本实例中，设置的是图表的背景图，还可以根据 JFreeChart 的 setBackgroundImage(Image image)方法为 JFreeChart 的窗体设置背景图，运行效果如图 7.9 所示。

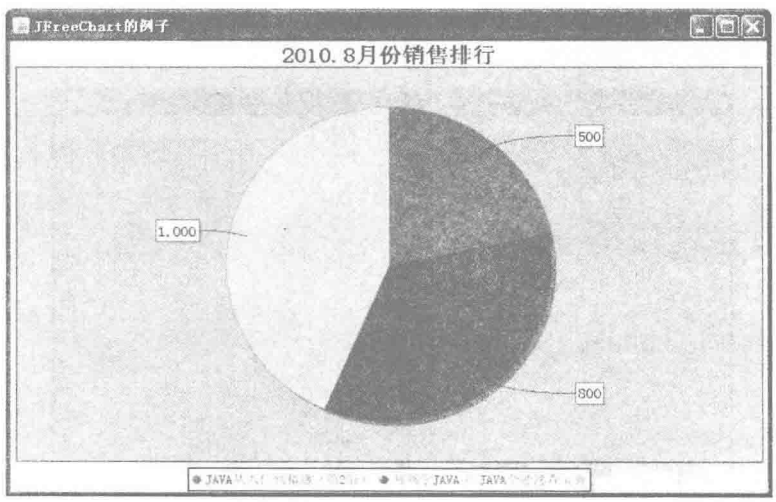


图 7.9 JFreeChart 窗体背景图

实例 151　设置图片对齐方式

光盘位置：光盘\MR\151　　初级　　趣味指数：★★★

实例说明

为 JFreeChart 设置背景图片时由于图片的大小不一定能和窗体完全一致，很多时候需要根据背景图片的特点设置图片的对齐方式，本实例设置饼图的背景图片居中对齐，运行效果如图 7.10 所示。

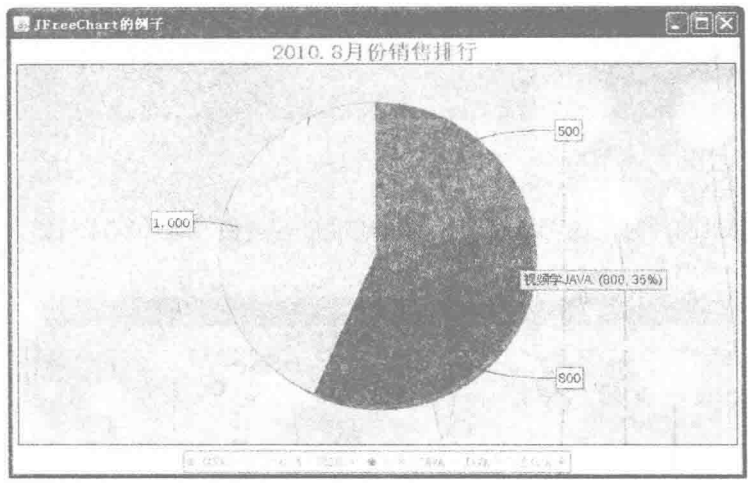


图 7.10　设置图表背景图片对齐方式

关键技术

使用 PiePlot 的 setBackgroundImageAlignment()方法，可以设置背景图对齐方式，语法如下：

```
public void setBackgroundImageAlignment(int alignment)
```

参数说明

alignment：表示要设置的背景图对齐方式。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias() 为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setBackgroundImage()方法，设置背景图片，同时设置背景图居中显示，代码如下：

```
public void setBackgroundImage(JFreeChart chart) {
    Image image = null;
    try {
        image = ImageIO.read(new File("backgroundImage.jpg"));
    } catch (IOException e) {
        e.printStackTrace();
    }
    PiePlot piePlot = (PiePlot) chart.getPlot();
```

```
        piePlot.setBackgroundImage(image);
        //设置背景对齐方式
        piePlot.setBackgroundImageAlignment(Align.CENTER);
    }
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 151：JFreeChart 中图表背景图的设置。

在实例中，虽然只定义了图表的背景图，但是如果为图表的背景图设置对齐方式以后，背景图片的作用域会扩展到 JFreeChart 窗体，不只是图表的背景。

实例 152 [illegible]图片透明度

[illegible]\MR\152

初级

趣味指数：★★★★

实例说明

为 JFreeChart 设置背景图片时，还可以对图片的透明度进行调整，本实例演示如何设置背景图的透明度，运行效果如图 7.11 所示。

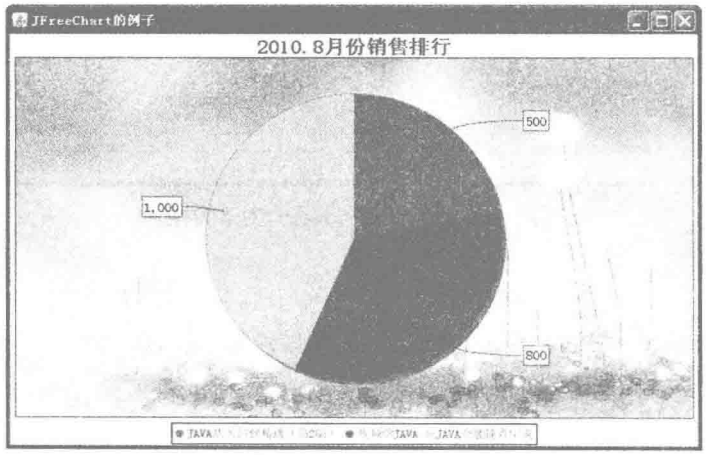


图 7.11　设置图表背景图片透明度

关键技术

使用 PiePlot 的 setBackgroundImageAlpha()方法可以设置背景图的透明度，语法如下：

```
public void setBackgroundImageAlpha(float alpha)
```

参数说明

alpha：表示要设置背景图的透明度，如果没有背景图，该设置不起作用。参数 alpha 的值为 0~1 之间的 float 类型值。值越小，透明度越小，背景图越模糊；反之，值越大，透明度越大，背景图越清晰。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias() 为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setBackgroundImage()方法，设置背景图片，同时设置背景图的透明度为 1，代码如下：

```
public void setBackgroundImage(JFreeChart chart) {
        Image image = null;
        try {
                image = ImageIO.read(new File("backgroundImage.jpg"));
        } catch (IOException e) {
                e.printStackTrace();
        }
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setBackgroundImage(image);
        //设置背景透明度
        piePlot.setBackgroundImageAlpha(1f);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 152：背景图的透明度设置。

实例中设置背景图的透明度为 1，表示背景图达到清晰的效果。如果透明度为 0，背景图将无法显示，只能显示图表的背景颜色。

实例 153　设置背景颜色

初级

趣味指数：★★★★

实例位置：光盘\MR\153

实例说明

JFreeChart 可以为图表设置背景颜色，从而让图表更美观，该功能可以单独使用，也可以与背景图和背景图片的透明度组合起来使用。本实例将图表的背景颜色设置为橙色，运行效果如图 7.12 所示。

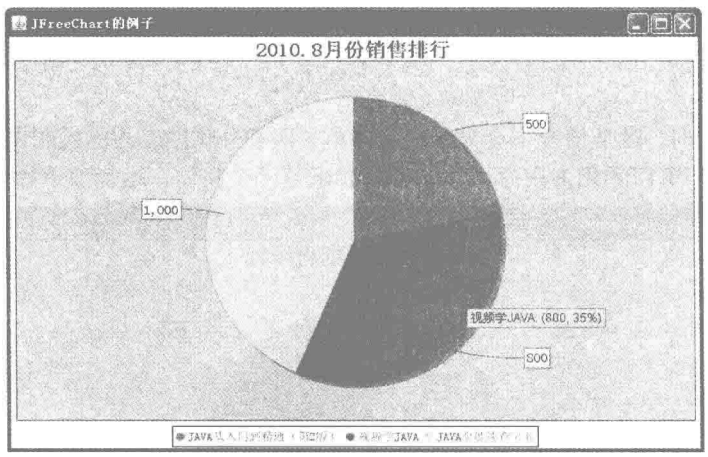


图 7.12　设置图表背景色

关键技术

使用 PiePlot 的 setBackgroundPaint()方法，可以为饼图图表设置背景颜色，语法如下：

```
public void setBackgroundPaint(Paint paint)
```

参数说明

paint：表示要设置背景的颜色值。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setBackgroundColor()方法，修改 JFreeChart 的默认颜色，使用 Color.orange 设置图表的背景色为橙色，代码如下：

```
public void setBackgroundColor(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置饼图背景色
        piePlot.setBackgroundPaint(Color.orange);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 153：使用数值为 Color 类设置颜色。

在 Color 类中定义的常量只为使用者提供基本的颜色，如果希望使用更丰富的颜色，可以使用 Color 的构造方法直接填写色值，如 new Color(200,220,202)、new Color(200,220,202,255)等。

7.3 处理图表的边框

实例 154	隐藏图表边框	中级
	实例位置：光盘\MR\154	趣味指数：★

实例说明

JFreeChart 在显示图表时，默认情况下会显示一个边框，用于分隔图表与标题和图示。本实例隐藏图表的边框，同时设置图表背景色和窗口颜色为白色，让二者看起来是一个整体，运行效果如图 7.13 所示。

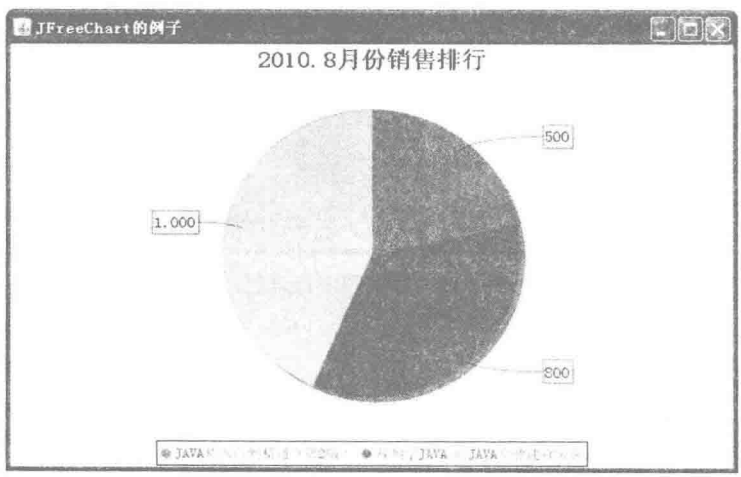


图 7.13 隐藏边框

关键技术

使用 PiePlot 的 setOutlineVisible()方法可以设置是否显示饼图边框，语法如下：

```
public void setOutlineVisible(boolean visible)
```

参数说明

visible：表示是否显示饼图边框。如果设置为 true，表示显示边框；如果设置为 false，表示隐藏边框。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias() 为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setOutline()方法，取消饼图边框，同时设置饼图背景色为白色（与窗体背景色一致），代码如下：

```
public void setOutline(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setBackgroundPaint(Color.white);
        //取消边框
        piePlot.setOutlineVisible(false);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 154：JFreeChart 图表边框的设置。

JFreeChart 的图表边框默认状态是显示的，本实例中如果只把窗体背景色调成白色，可以非常清晰地看到窗体边框，运行效果如图 7.14 所示。

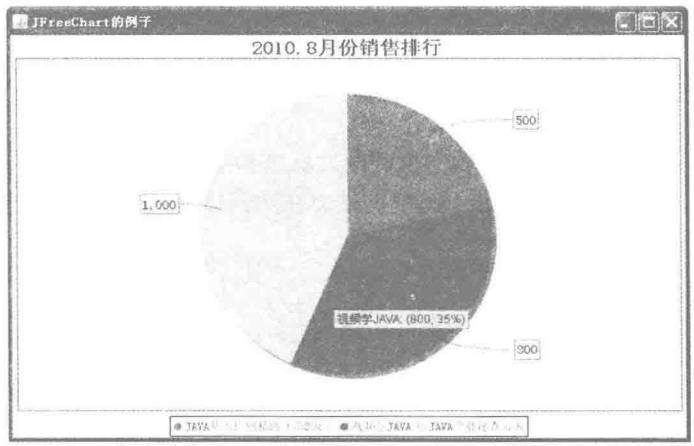


图 7.14　显示边框

实例 155　图表边框笔触

光盘位置：光盘\MR\155　　初级　　趣味指数：★★★

实例说明

笔触是画笔留下的痕迹，通过设置图表边框的笔触可以调节边框的宽度。本实例中把边框宽度调粗，运行

效果如图 7.15 所示。

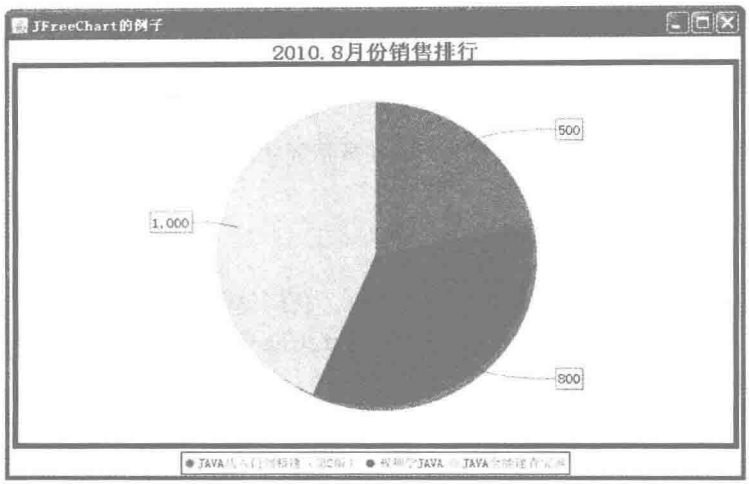


图 7.15　加粗边框

关键技术

使用 PiePlot 的 setOutlineStroke()方法可以为边框设置笔触，语法如下：

```
public void setOutlineStroke(Stroke stroke)
```

参数说明

stroke：表示边框的笔触，可以用 BasicStroke 进行实例化。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setOutline()方法，使用 BaseStroke 的构造方法实例化 Stroke，并且设置笔触的宽度为 5，再使用 setOutlineStroke()方法设置边框笔触，代码如下：

```
public void setOutline(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置饼图背景色
        piePlot.setBackgroundPaint(Color.white);
        Stroke stroke = new BasicStroke(5);
        //设置饼图笔触
        piePlot.setOutlineStroke(stroke);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 155：使用 BasicStroke 类中的笔触。

BasicStroke 的构造方法很多，不但可以调整边框的宽度，还可以调整边框的其他样式。在 BasicStroke 类中还定义了一些关于笔触的常量，用来设置 Stroke 的装饰。

实例 156　图表边框颜色

初级

趣味指数：★★★

实例说明

本实例中，饼图和窗体的背景色都为白色，同时设置图表边框为橙色，为了看起来更清晰，可以把边框的笔触加粗，运行效果如图 7.16 所示。

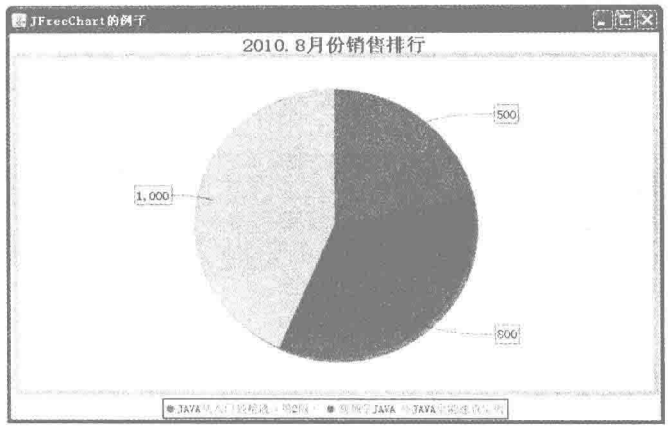


图 7.16　设置饼图边框颜色

关键技术

使用 PiePlot 类的 setOutlinePaint()方法可以为边框设置颜色，语法如下：

```
public void setOutlinePaint(Paint paint)
```

参数说明

paint：表示边框的颜色，可以使用 Paint 的实现类 Color 或者其常量实现，代码如下：

```
public void setOutline(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setBackgroundPaint(Color.white);
        Stroke stroke = new BasicStroke(5);
        //设置边框笔触
        piePlot.setOutlineStroke(stroke);
        //设置边框颜色
        piePlot.setOutlinePaint(Color.orange);
}
```

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias() 为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setOutline()方法，使用 PiePlot 的 setOutlinePaint()方法为边框设置橙色，代码如下：

```
public void setOutline(JFreeChart chart) {
        PiePlot piePlot = (PiePlot) chart.getPlot();
        piePlot.setBackgroundPaint(Color.white);
        Stroke stroke = new BasicStroke(5);
```

```
        //设置边框笔触
        piePlot.setOutlineStroke(stroke);
        //设置边框颜色
        piePlot.setOutlinePaint(Color.orange);
    }
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 156：图表背景色设置的特点。

本实例中，图表边框为橙色，图表和窗体背景色都为白色，如果把图表边框的颜色也设置成白色，可以达到隐藏图表边框的效果。

7.4 修改图表的图示

实例 157　设置图示背景色　　初级

光盘位置：光盘\MR\157　　趣味指数：★★★★

实例说明

JFreeChart 的图示背景色默认为白色，本实例中，将图示的背景色设置为橙色，运行效果如图 7.17 所示。

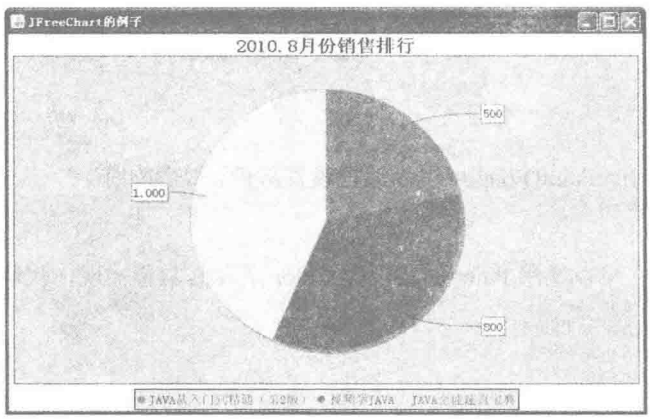


图 7.17　设置图示背景色

关键技术

JFreeChart 中，LegendTitle 类用于处理图示，使用 LegendTitle 类的 setBackgroundPaint()方法可以设置图示的背景颜色，语法如下：

```
public void setBackgroundPaint(Paint paint)
```

参数说明

paint：表示图示的背景颜色。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()

为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setLegendTitle()方法，使用 LegendTitle 类的 setBackgroundPaint()方法设置图示的背景色为橙色，代码如下：

```
public void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        //设置图示背景色
        legendTitle.setBackgroundPaint(Color.orange);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 157：图示颜色设置技巧。

图示中，常使用颜色代表某一种类别，在设置背景色时，不要把背景色与图示中的颜色设置为同一颜色，这样不容易区分。

实例 158　设置图示边框

光盘位置：光盘\MR\158　　初级　　趣味指数：★★

实例说明

JFreeChart 图示的边框与图表边框不同，主要通过设置边框的距离来确定边框的粗细。本实例把饼图图示的边框去掉，运行效果如图 7.18 所示。

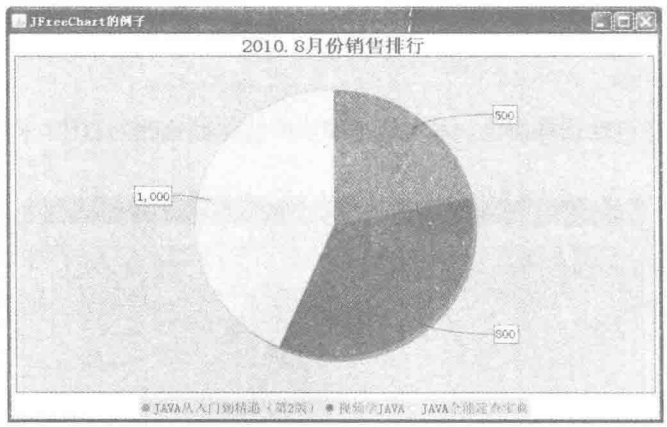


图 7.18　设置图示边框

关键技术

使用 LegendTitle 类的 setBorder()方法可以设置图示的边框，语法如下：

```
public void setBorder(double top, double left, double bottom,double right)
```

参数说明

❶ top：表示图示上边的宽度。

❷ left：表示图示左边的宽度。

❸ bottom：表示图示底边的宽度。

❹ right：表示图示右边的宽度。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias() 为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setLegendTitle()方法，使用 LegendTitle 类的 setBackgroundPaint()方法设置图示的背景色为橙色，同时设置图示的边框为 0，代码如下：

```
public void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setBackgroundPaint(Color.orange);
        //设置图示边框
        legendTitle.setBorder(0, 0, 0, 0);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 158：图示边框的设置。

图示的边框默认为显示状态，使用 JFreeChart 生成图形时，读者可以根据不同的需要对图示边框进行设置显示或者隐藏。

实例 159　设置图示边框颜色

大盘位置：光盘\MR\159　　中级　　趣味指数：★★

实例说明

JFreeChart 图示的边框颜色默认是黑色，本实例把图示的边框颜色改为蓝色，同时把边框加粗，运行效果如图 7.19 所示。

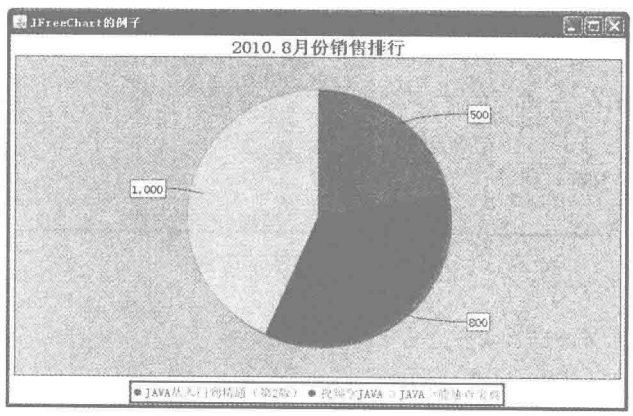


图 7.19　设置图示边框颜色

关键技术

使用 LegendTitle 的 setFrame()方法可以设置 BlockFrame 实例，通过 BlockBorder 的实例可以设置边框的颜色、宽度等，setFrame()方法的语法如下：

```
public void setFrame(BlockFrame frame)
```

参数说明

frame：表示 BlockBorder 的实例。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setLegendTitle()方法，然后使用 BlockBorder 类实例化 BlockFrame 接口，然后使用 LegendTitle 类的 setFrame()方法保存 BlockFrame 实例，代码如下：

```
public void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        //设置图示边框颜色
        BlockFrame blockFrame = new BlockBorder(2,2,2,2,Color.blue);
        legendTitle.setFrame(blockFrame);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 159：setBorder()方法修改图示边框颜色。

LegendTitle 的 setBorder(double top, double left, double bottom, double right)方法用于设置图示边框，而 BlockBorder 类不但可以设置图示边框，还可以改变图示边框的颜色，其功能更强大。

实例 160　图示边缘间距

源代码\MR\160

高级

趣味指数：★★★

实例说明

图示的边缘与边框不同，它确定的是整个图示（包括边框）与周边的距离。本实例设置图示与左边间距为 10、上边间距为 0、右边间距为 246、下边间距为 10，运行效果如图 7.20 所示。

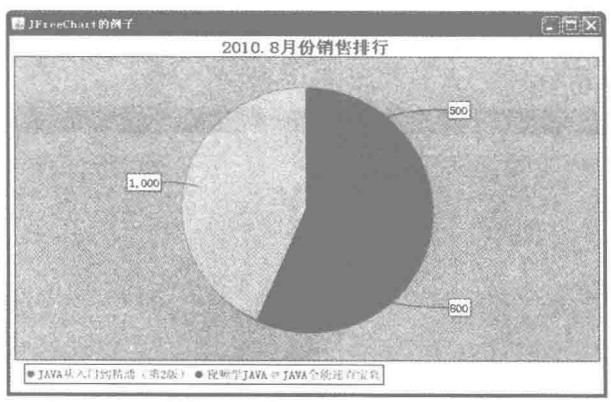


图 7.20　设置图示边缘间距

关键技术

LegendTitle 类的 setMargin()方法用于设置图示边缘的距离，其语法如下：

public void setMargin(double top, double left, double bottom,double right)

参数说明

❶ top：表示图示上边的距离。

❷ left：表示图示左边的距离。

❸ bottom：表示图示底边的距离。

❹ right：表示图示右边的距离。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias() 为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setLegendTitle()方法，使用 LegendTitle 类的 setMargin()方法设置图示与四边的距离，代码如下：

```
public void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        //设置图示间距
        legendTitle.setMargin(0, 10, 10, 246);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 160：控制图示边缘间距来移动图示的位置。

可以通过设置图示的边缘间距来移动图示的位置，但根据图表所在的窗体布局来看，通过这种方式，不管怎样移动，图示都只能显示在图表下方。

实例 161　设置图示字体颜色

光盘位置：光盘\MR\161　　中级　　趣味指数：★★★

实例说明

JFreeChart 的图示的字体颜色默认为黑色，有时需要对字体颜色进行调整。本实例中，将图示的字体颜色调整为紫色，运行效果如图 7.21 所示。

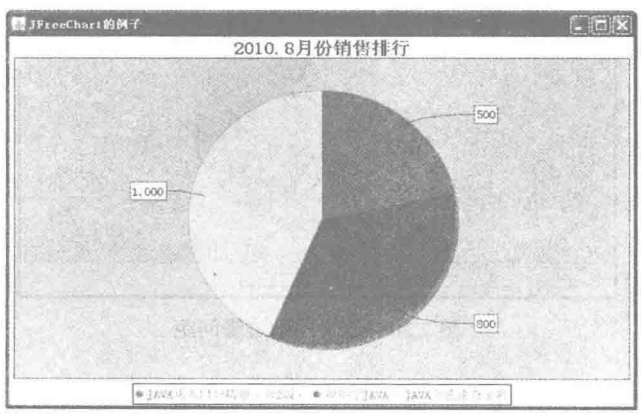


图 7.21　设置图示字体颜色

关键技术

LegendTitle 类的 setItemPaint()方法用于为图示设置字体颜色，语法如下：

```
public void setItemPaint(Paint paint)
```

参数说明

paint：表示图示字体的颜色。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setLegendTitle()方法，使用 LegendTitle 类的 setItemPaint()方法设置图示字体的颜色为紫色，代码如下：

```
public void setLegendTitle(JFreeChart chart) {
        LegendTitle legendTitle = chart.getLegend();
        //设置图示字体颜色
        legendTitle.setItemPaint(Color.MAGENTA);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 161：图示字体颜色设置技巧。

图示的字体颜色可以根据用户需求设置，在设置字体颜色时需要注意，切忌与图示的背景色相同或者相近，否则分辨不出图示上的文字。

实例 162　设置图示位置

光盘位置：光盘\MR\162　　中级　　趣味指数：★★★

实例说明

图示的默认位置在图表下方，有时根据不同需求，需要把图示调整到其他方位。本实例把图示位置调整到图表右侧，运行效果如图 7.22 所示。

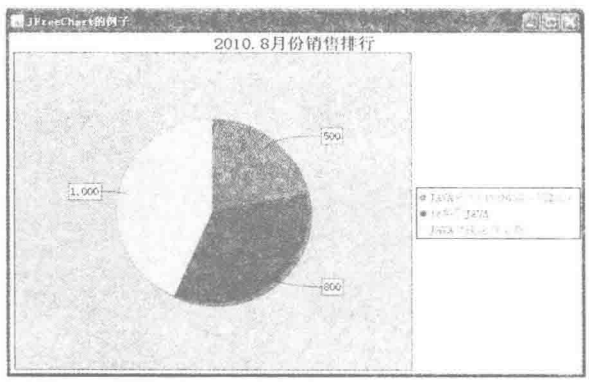


图 7.22　设置图示位置

关键技术

LegendTitle 类的 setPosition()方法可以设置图示在图表中的方位，语法如下：

```
public void setPosition(RectangleEdge position)
```

参数说明

position：表示图示在图表中的方位。

设计过程

（1）新建一个 Java 文件。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 147。

（3）ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，设置 JFreeChart 的方法 setAntiAlias()为 false，关闭抗锯齿设置，代码见实例 149。

（4）创建 setPiePoltFont()方法，用于设置图表、图表标题和图示的字体，代码见实例 147。

（5）创建 setLegendTitle()方法，在方法内获取 LegendTitle 对象，使用 LegendTitle 的 setPosition()方法把图示显示在图表右侧，代码如下：

```
public void setLegendTitle(JFreeChart chart) {
    LegendTitle legendTitle = chart.getLegend();
    //设置图示位置
    legendTitle.setPosition(RectangleEdge.RIGHT);
}
```

（6）创建一个 main()方法，使用 ChartFrame 的构造方法为 JFreeChart 生成一个窗体。

秘笈心法

心法领悟 162：使用 RectangleEdge 定义的常量设置图示方位。

JFreeChart 图示的方位默认使用的是 RectangleEdge.BOTTOM，即位于图表下方。RectangleEdge 还有另外 3 个常量用于标识图示方位，除了本实例用到的 RectangleEdge.RIGHT 之外，还有 RectangleEdge.LEFT 和 RectangleEdge.TOP。

第 8 章

基础图表技术

- 普通饼图
- 3D 饼图
- 多饼图
- 基本柱形图
- X 坐标轴
- Y 坐标轴
- 高级柱形图

8.1 普 通 饼 图

实例 163	分离饼图	高级
	光盘位置：光盘\MR\163	趣味指数：★★★★★

实例说明

制作图表时，某一个或者几个类别可能非常重要，需要将其突出显示。如本实例突出显示 8 月份销售最多和最少的图书，即把《Java 范例完全自学手册（1DVD）》和《Java 全能速查宝典》的销售情况分离出来，运行效果如图 8.1 所示。

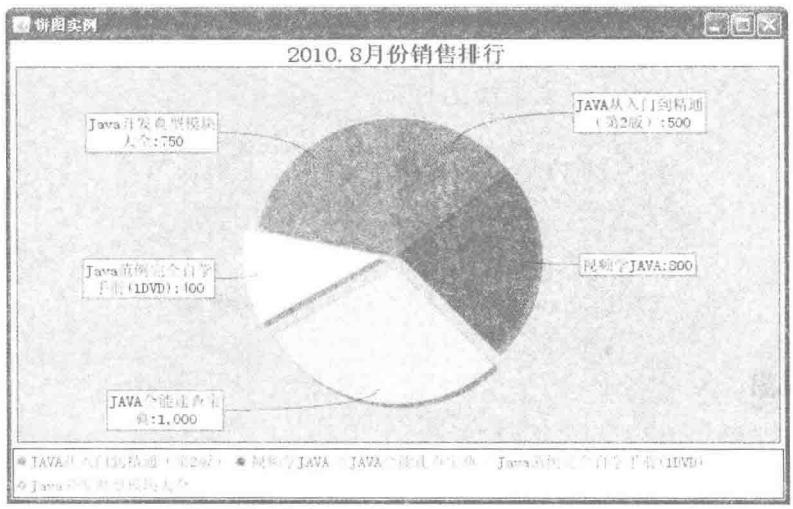


图 8.1 分离饼图

关键技术

使用 PiePlot 类的 setExplodePercent()方法可以指定要分离的饼图扇形，语法如下：

```
public void setExplodePercent(Comparable key, double percent)
```

参数说明

❶ key：表示要分离的名称。

❷ percent：表示要分离的距离。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，再使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，然后创建 createPiePlot()方法，在该方法中获取 JFreeChart 的实例。从 JFreeChart 中获取 PiePlot 实例后，设置需要分离的分类，最后调用父类的方法 setContentPane()把 JFreeChart 添加到面板中。部分代码如下：

```
/**
 * 设置 Pie
 * @param chart
 */
```

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //需要分离的图书
        piePlot.setExplodePercent("Java 范例完全自学手册(1DVD)", 0.1);
        piePlot.setExplodePercent("JAVA 全能速查宝典", 0.1);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（3）在 main()方法中生成 PieDemo1 的实例，调用 createPiePlot()方法，同时使用 RefineryUtilities 的 centerFrameOnScreen()方法把实例窗体显示到屏幕中央。代码如下：

```
public static void main(String[] args) {
        PieDemo1 pieChartDemo1 = new PieDemo1("饼图实例");
        pieChartDemo1.createPiePlot();
        pieChartDemo1.pack();
        RefineryUtilities.centerFrameOnScreen(pieChartDemo1);
        pieChartDemo1.setVisible(true);
}
```

秘笈心法

心法领悟 163：直接使用 JFrame 中的方法重写 WindowListener 接口。

本实例继承了 org.jfree.ui.ApplicationFrame 类，而 ApplicationFrame 又继承了 javax.swing.JFrame 类和 java.awt.event.WindowListener 接口，所以本实例可以直接使用 JFrame 中的方法，还可以重写 WindowListener 接口。

实例 164　椭圆形饼图

光盘位置：光盘\MR\164　　中级　　趣味指数：★★★★

实例说明

一般的饼图是正圆形的，但有时也会使用椭圆形的饼图。JFreeChart 提供了这样一个功能，可以很方便地绘制一个椭圆形的饼图，运行效果如图 8.2 所示。

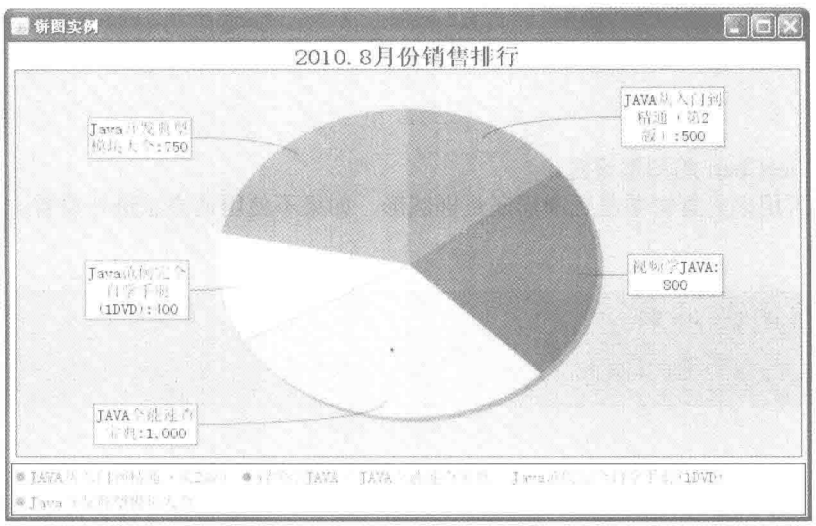


图 8.2　椭圆形饼图

关键技术

PiePlot 类提供的方法 setCircular()可以指定是否绘制圆形，其语法如下：

```
public void setCircular(boolean flag)
```

参数说明

flag：表示要绘制的图表是否为正圆形。当 flag 为 true 时，表示为正圆形；为 false 时，表示为椭圆形。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 org.jfree.ui.ApplicationFrame 类。

（2）获取 DefaultPieDataset 创建的饼图的数据集合，然后使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，再修改饼图字体。代码如下：

```
private JFreeChart getJFreeChart() {
        //获取数据集
        PieDataset dataset = getPieDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createPieChart("2010.8 月份销售排行", dataset,true, true, false);
        //设置图表字体
        setPiePoltFont(chart);
        return chart;
}
```

（3）创建 createPiePlot()方法，在该方法中使用 PiePlot 的 setCircular()方法，设置图表为椭圆形。代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //是否椭圆
        piePlot.setCircular(false);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（4）在 main()方法中调用 createPiePlot()方法创建图表，然后使用 RefineryUtilities 类的 centerFrameOnScreen()方法把图表窗体显示到屏幕中央。代码如下：

```
public static void main(String[] args) {
        PieDemo2 demo = new PieDemo2("饼图实例");
        demo.createPiePlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 164：JFreeChart 的图形设置。

SetCircular()方法可用来设置饼图是正圆形还是椭圆形，如果不使用该方法进行设置，JFreeChart 的默认情况下都是正圆形饼图。

实例 165 饼图的阴影

光盘位置：光盘\MR\165

中级

趣味指数：★★★

实例说明

本实例实现在 JFreeChart 饼图产生时，同时在其下面产生一个图表阴影，并且在图表的阴影区域中为不同分类绘制了分隔线，运行效果如图 8.3 所示。

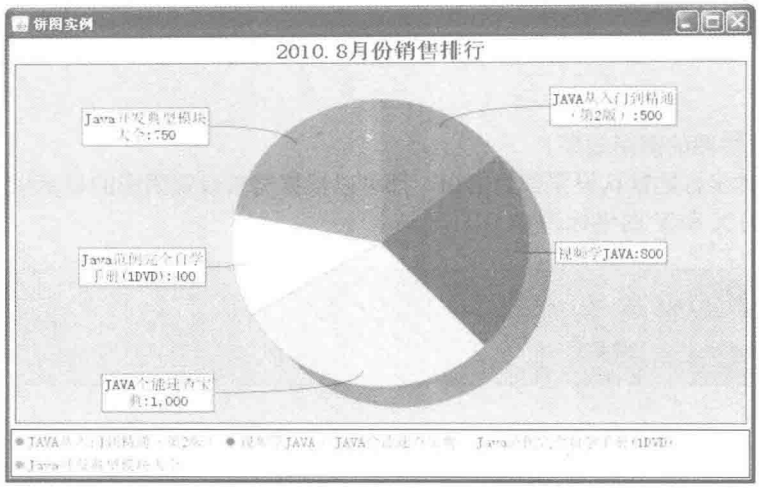


图 8.3　图形阴影

关键技术

JFreeChart 产生的图表阴影使用两个方法设置偏移。

（1）使用 setShadowXOffset()方法设置 X 轴方向的偏移，语法如下：

```
public void setShadowXOffset(double offset)
```

参数说明

offset：表示 X 轴方向上的偏移量。

（2）使用 setShadowYOffset()方法设置 Y 轴方向的偏移，语法如下：

```
public void setShadowYOffset(double offset)
```

参数说明

offset：表示 Y 轴方向上的偏移量。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 org.jfree.ui.ApplicationFrame 类。

（2）获取 DefaultPieDataset 创建的饼图的数据集合，然后使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，再修改饼图字体。代码见实例 164。

（3）创建 createPiePlot()方法，在方法中使用 PiePlot 的 setShadowXOffset(double x)和 setShadowYOffset(double y)方法设置阴影坐标为(20,20)。代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置阴影效果
        piePlot.setShadowXOffset(20);
        piePlot.setShadowYOffset(20);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（4）在 main()方法中调用 createPiePlot()方法创建图表，然后使用 RefineryUtilities 类的 centerFrameOnScreen()方法把图表窗体显示到屏幕中央。代码如下：

```
public static void main(String[] args) {
        PieDemo3 demo = new PieDemo3("饼图实例");
        demo.createPiePlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
```

```
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 165：设置饼图的阴影效果。

饼图阴影的 X、Y 轴坐标的默认设置都为 4.0f，都可以根据需求设置阴影的显示坐标，如果不希望使用阴影效果，可以设置阴影的 X 和 Y 的坐标为 0。

实例 166 饼图的分类边框颜色

光盘位置：光盘\MR\166　　中级　　趣味指数：★★★

实例说明

饼图的分类是有边框的，而且分类的边框可以与分类本身的颜色不一致。饼图分类边框的默认颜色为灰色。本实例把所有分类的边框颜色改成黑色，运行效果如图 8.4 所示。

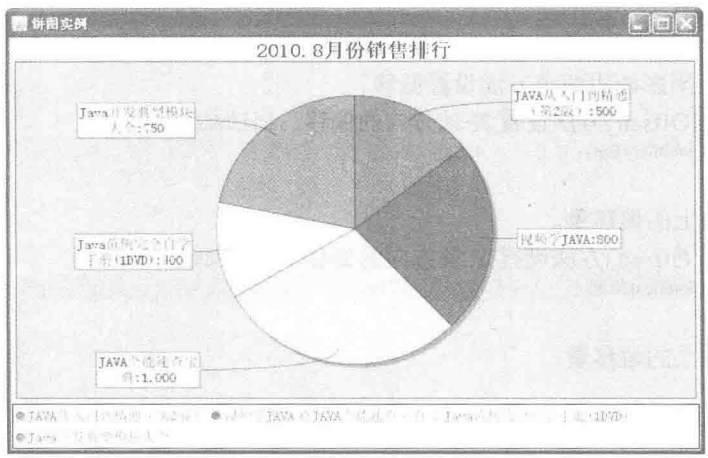


图 8.4　黑色的分类边框

关键技术

PiePlot 类中的 setSectionOutlinePaint()方法用于设置饼图的分类边框的颜色，语法如下：

```
public void setSectionOutlinePaint(Comparable key, Paint paint)
```

参数说明

❶ key：表示要赋予颜色的分类名称。

❷ paint：表示要赋予的颜色。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private PieDataset getPieDataset() {
        //创建数据集合实例
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向数据集合添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
```

```
        dataset.setValue("JAVA 全能速查宝典", 1,000);
        dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
        dataset.setValue("Java 开发典型模块大全", 750);
        return dataset;
    }
```

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并且设置饼图字体，代码见实例 164。

（4）创建 createPiePlot()方法，在方法中使用 PiePlot 的 setSectionOutlinePaint()方法为所有分类设置边框颜色为黑色，代码如下：

```
    public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置饼图边框的颜色
        piePlot.setSectionOutlinePaint("JAVA 从入门到精通（第 2 版）", Color.black);
        piePlot.setSectionOutlinePaint("视频学 JAVA", Color.black);
        piePlot.setSectionOutlinePaint("JAVA 全能速查宝典", Color.black);
        piePlot.setSectionOutlinePaint("Java 范例完全自学手册(1DVD)", Color.black);
        piePlot.setSectionOutlinePaint("Java 开发典型模块大全", Color.black);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
    }
```

（5）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把图表窗体显示到屏幕中央。

秘笈心法

心法领悟 166：饼图的分类边框。

一个完整的饼图由各个分类组成，每个分类自成一个扇形。饼图的分类边框并不单指扇形的弧线部分，还包括弧线部分两侧的直线，分类边框的颜色指的就是这 3 条线的颜色。

实例 167　加粗饼图分类边框

光盘位置：光盘\MR\167　　中级　　趣味指数：★★★

实例说明

饼图分类的边框可以根据需要进行改变。本实例把所有分类的边框加粗，通过修改边框笔触达到预想的效果，运行效果如图 8.5 所示。

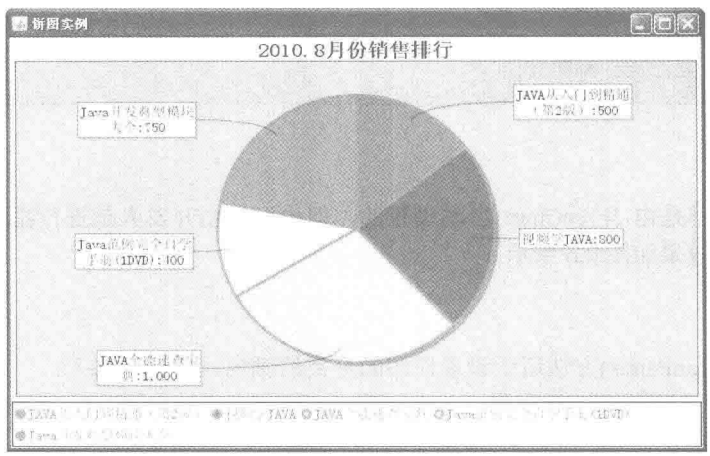


图 8.5　加粗分类边框

关键技术

PiePlot 类中的 setSectionOutlineStroke()方法用于设置饼图的笔触，其语法如下：

```
public void setSectionOutlineStroke(Comparable key, Stroke stroke)
```

参数说明

❶ key：表示要设置笔触的分类名称。

❷ stroke：表示要设置的笔触对象。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合。代码见实例 166。

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并且设置饼图字体，代码见实例 164。

（4）创建 createPiePlot()方法，在方法中使用 PiePlot 的 setSectionOutlineStroke()方法为所有分类设置边框笔触为 3，代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置饼图边框笔触
        piePlot.setSectionOutlineStroke("JAVA 从入门到精通（第 2 版）",new BasicStroke(3f));
        piePlot.setSectionOutlineStroke("视频学 JAVA", new BasicStroke(3f));
        piePlot.setSectionOutlineStroke("JAVA 全能速查宝典", new BasicStroke(3f));
        piePlot.setSectionOutlineStroke("Java 范例完全自学手册(1DVD)", new BasicStroke(3f));
        piePlot.setSectionOutlineStroke("Java 开发典型模块大全", new BasicStroke(3f));
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（5）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把图表窗体显示到屏幕中央。

秘笈心法

心法领悟 167：设置分类的边框笔触。

分类的内容和图示是关联在一起的，在实例中，设置分类的边框笔触为 3，那么分类图示的边框笔触也会变为 3。

实例 168 设置饼图颜色

光盘位置：光盘\MR\168　　中级　　趣味指数：★★★★

实例说明

饼图分类的颜色一般都是由 JFreeChart 自动设置的，但也可以由开发人员进行指定。本实例重新设置了饼图分类的图表颜色，运行效果如图 8.6 所示。

关键技术

PiePlot 类中的 setSectionPaint()方法用于设置饼图的分类的颜色，其语法如下：

```
public void setSectionPaint(Comparable key, Paint paint)
```

参数说明

❶ key：表示要设置颜色的分类名称。

❷ paint：表示要设置的颜色。

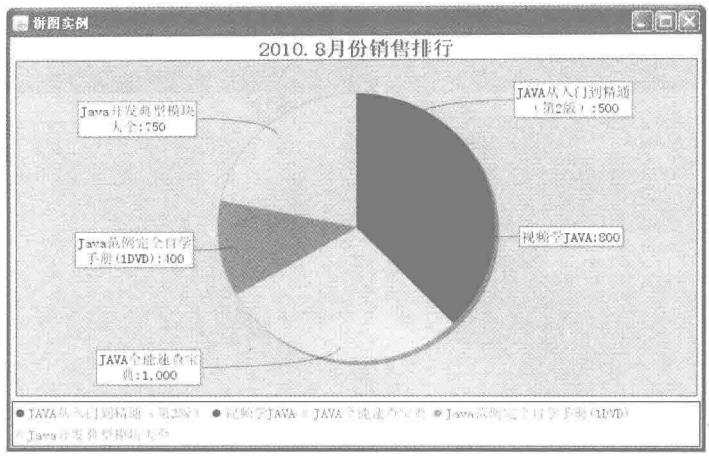


图 8.6　设置饼图颜色

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 166。

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并且设置饼图字体，代码见实例 164。

（4）创建 createPiePlot()方法，在方法中使用 PiePlot 的 setSectionPaint()方法为所有分类设置颜色，代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置饼图的颜色
        piePlot.setSectionPaint("JAVA 从入门到精通（第 2 版）", Color.black);
        piePlot.setSectionPaint("视频学 JAVA", Color.blue);
        piePlot.setSectionPaint("JAVA 全能速查宝典", Color.cyan);
        piePlot.setSectionPaint("Java 范例完全自学手册(1DVD)", Color.gray);
        piePlot.setSectionPaint("Java 开发典型模块大全", Color.orange);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（5）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把图表窗体显示到屏幕中央。

秘笈心法

心法领悟 168：设置饼图分类颜色。

为饼图设置分类颜色时，可以为一部分分类设置颜色，也可以为所有分类设置颜色。如果只为其中一部分分类设置颜色，那么剩余分类的颜色 JFreeChart 会自动进行设置。

实例 169　饼图旋转角度

光盘位置：光盘\MR\169　　中级　　趣味指数：★★★

实例说明

饼图是一个圆形区域，每个分类为一块扇形区域，各区域在图形中的位置可以因饼图的旋转而改变，运行

效果如图 8.7 所示。

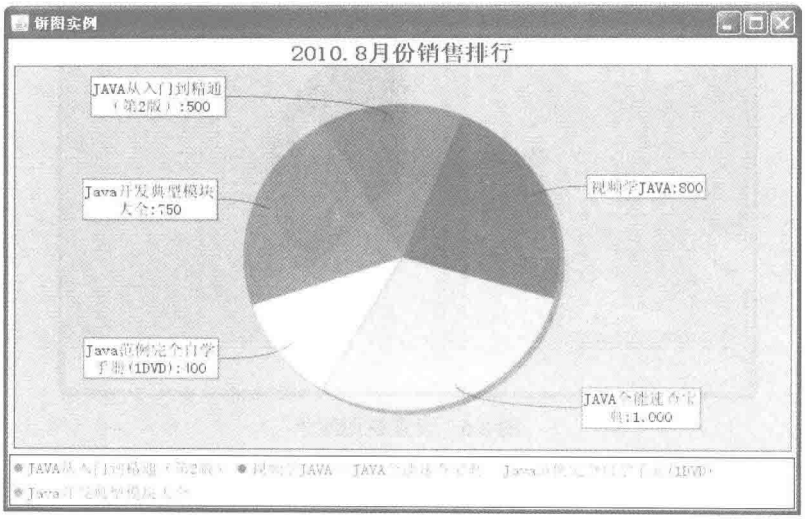


图 8.7　饼图旋转角度

关键技术

饼图默认情况下的初始角度为 90°，使用 PiePlot 类的 setStartAngle()方法可以设置旋转角度，语法如下：

```
public void setStartAngle(double angle)
```

参数说明

、angle：表示要改变的饼图初始角度。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 166。

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并且设置饼图字体，代码见实例 164。

（4）创建 createPiePlot()方法，在方法中使用 PiePlot 类的 setStartAngle()方法，设置初始角度为 120°，代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置旋转角度
        piePlot.setStartAngle(120);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（5）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把图表窗体显示到屏幕中央。

秘笈心法

心法领悟 169：JFreeChart 产生的图表角度。

JFreeChart 产生的图表角度是以数据集中添加的第一个分类的扇形左边线为基准的，并以过饼图中心的水平线为参照物。本实例中的角度是指基准线与参照物逆时针所成的角度，如图 8.8 所示。

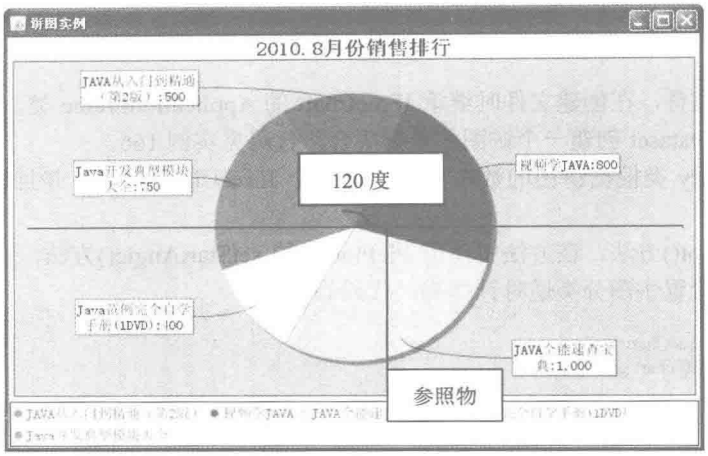


图 8.8　饼图旋转角度参照物

实例 170　饼图旋转顺序

光盘位置：光盘\MR\170　　中级　　趣味指数：★★★

实例说明

饼图类别的排列是按照数据集添加的顺序显示的，默认情况下，饼图类别按顺时针方向根据数据集添加的顺序依次向下排列。本实例修改了默认顺序，使饼图类别能够逆时针排列，运行效果如图 8.9 所示。

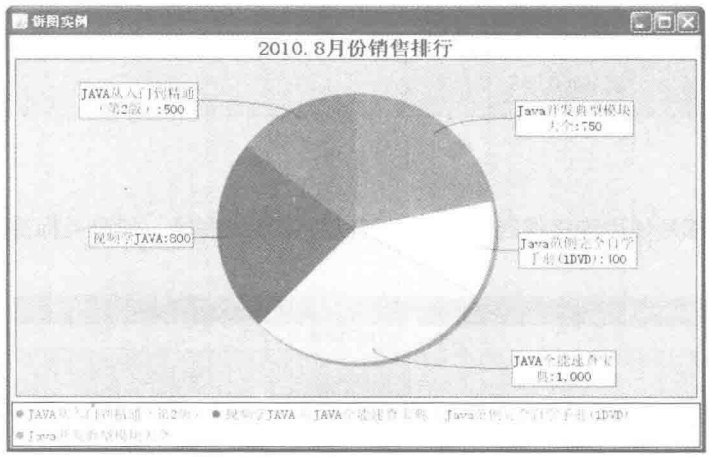


图 8.9　饼图逆时针排列

关键技术

使用 PiePlot 类的 setDirection()方法可以设置饼图的排列顺序，语法如下：

public void setDirection(Rotation direction)

参数说明

direction：表示饼图的排列顺序，它是 Rotation 类的实例，在 Rotation 中已经定义了两个常量用于设置饼图分类的排列顺序。

（1）Rotation.CLOCKWISE：定义了饼图顺时针排列分类。

（2）Rotation.ANTICLOCKWISE：定义了饼图逆时针排列分类。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 166。

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，并且设置饼图字体，代码见实例 164。

（4）创建 createPiePlot()方法，在方法中使用 PiePlot 类的 setStartAngle()方法，设置初始角度为 90°，同时使用 setDirection()方法设置饼图分类逆时针排列，代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置旋转角度
        piePlot.setStartAngle(90);
        //设置逆时针
        piePlot.setDirection(Rotation.ANTICLOCKWISE);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（5）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把图表窗体显示到屏幕中央。

秘笈心法

心法领悟 170：饼图顺时针排列与逆时针排列。

使用常量 Rotation.CLOCKWISE 时，说明饼图分类按顺时针排列，旋转角度以第一个分类的左边轴为起点；使用常量 Rotation.ANTICLOCKWISE 时，说明饼图分类按逆时针排列，旋转角度以第一个分类的右边轴为起点。

实例 171　隐藏分类标签连接线

光盘位置：光盘\MR\171　　中级　　趣味指数：★★★

实例说明

饼图的分类标签默认都是使用连接线连接的，也可以将连接线去掉，则分类标签将直接显示在分类图上，运行效果如图 8.10 所示。

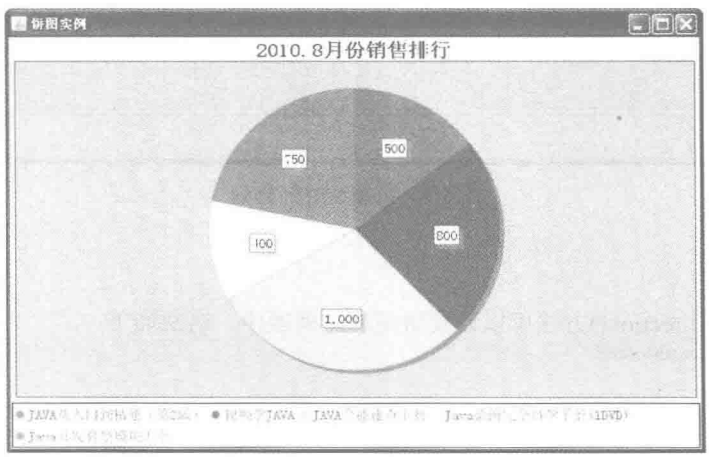


图 8.10　隐藏分类标签连接线

关键技术

饼图的分类标签连接线可以使用 piePlot.setSimpleLabels()方法进行设置，语法如下：

```
public void setSimpleLabels(boolean simple)
```

参数说明

simple：当 simple 为 true 时，表示使用简单的分类标签，连接线不会显示；当 simple 为 false 时，表示不使用简单的分类标签，连接线会显示出来。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合。代码见实例 166。

（3）使用 ChartFactory 类根据饼图的数据集合创建一个 JFreeChart 对象，代码见实例 164。

（4）创建 createPiePlot()方法，在方法中设置 PiePlot 类的 setSimpleLabels()方法为 true。代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot piePlot = (PiePlot) chart.getPlot();
        //设置标签模式
        piePlot.setSimpleLabels(true);
        //把 JFreeChart 对象保存到面板中
        setContentPane(new ChartPanel(chart));

}
```

（5）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把图表窗体显示到屏幕中央。

秘笈心法

心法领悟 171：使用 setSimpleLabels()方法隐藏分类标签连接线。

本实例使用 setSimpleLabels()方法隐藏了分类标签连接线，将分类标签直接显示在分类图形上，这适用于分类标签的内容比较少的情况，如果分类标签名称较长，一个标签内容可以会占用多个图形，则不容易区分标签真正代表的分类。

8.2　3D 饼图

实例 172　创建 3D 饼图　　高级

光盘位置：光盘\MR\172　　趣味指数：★★★★★

实例说明

普通的饼图都是平面的，可能会给人一种死板的感觉，JFreeChart 提供了一个很方便地创建 3D 饼图的功能，让图表更生动。本实例实现的 3D 饼图运行效果如图 8.11 所示。

关键技术

使用 ChartFactory 的 createPieChart3D()方法可以创建 3D 饼图，语法如下：

```
public static JFreeChart createPieChart3D(String title, PieDataset dataset, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：表示饼图的标题。

❷ dataset：表示饼图的数据集合。

❸ legend：表示是否使用图示。
❹ tooltips：表示是否生成工具栏提示。
❺ urls：表示是否生成 URL 链接。

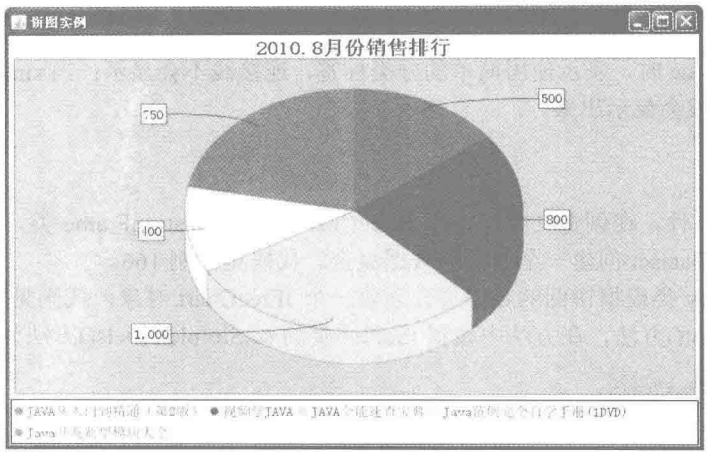


图 8.11　3D 饼图

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。
（2）使用 DefaultPieDataset 创建一个饼图的数据集合。代码如下：

```
private PieDataset getPieDataset() {
        //创建数据集合实例
        DefaultPieDataset dataset = new DefaultPieDataset();
        //向数据集合添加数据
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1,000);
        dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
        dataset.setValue("Java 开发典型模块大全", 750);
        return dataset;
}
```

（3）使用 ChartFactory 类根据饼图的数据集合创建有 3D 效果的 JFreeChart 对象。代码如下：

```
private JFreeChart getJFreeChart() {
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createPieChart3D("2010.8 月份销售排行", dataset,true, true, false);
        //设置饼图使用的字体
        setPiePoltFont(chart);
        return chart;
}
```

（4）创建 createPiePlot()方法，在方法中把 JFreeChart 对象保存到面板中。代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        //把 JFreeChart 对象保存到面板中
        setContentPane(new ChartPanel(chart));
}
```

（5）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把窗体显示到屏幕中央。

秘笈心法

心法领悟 172：3D 饼图与普通饼图的区别。

3D 饼图和普通饼图有很多效果是通用的，例如分离饼图、旋转饼图初始角度、设置简单的分类标签等，但是还有一部分效果是普通饼图特有的，如 3D 饼图不能设置阴影效果。

实例 173　3D 饼图透明度

中级

光盘位置：光盘\MR\173　　趣味指数：★★★★

实例说明

3D 饼图与普通饼图相比，立体感很强。本实体通过给 3D 饼图增加透明度，让其更具有立体感，运行效果如图 8.12 所示。

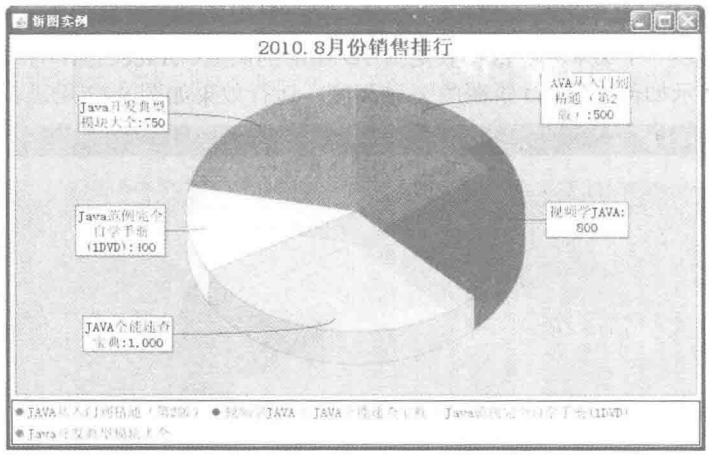


图 8.12　3D 饼图透明度

关键技术

设置饼图的透明度使用 PiePlot 的 setForegroundAlpha()方法，语法如下：

```
public void setForegroundAlpha(float alpha)
```

参数说明

alpha：表示 3D 饼图的透明度，范围为 0~1f，值越小，透明度越高；反之，值越大，透明度越低。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 172。

（3）使用 ChartFactory 类根据饼图的数据集合创建有 3D 效果的 JFreeChart 对象，代码见实例 172。

（4）创建 createPiePlot()方法，设置透明度，然后把 JFreeChart 对象保存到面板中。代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot plot   = (PiePlot)chart.getPlot();
        //设置饼图透明度
        plot.setForegroundAlpha(0.7f);
        //把 JFreeChart 对象保存到面板中
        setContentPane(new ChartPanel(chart));
}
```

（5）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把图表窗体显示到屏幕中央。

秘笈心法

心法领悟 173：使用 setForegroundAlpha()方法设置图表的透明度。

PiePlot 类的 setForegroundAlpha()方法用于设置图表的透明度，该方法不只适用于 3D 图表，还可以用在普通图表中，不过为普通的图表设置透明度的效果没有 3D 图表的效果突出。

实例 174　3D 饼图的 Z 轴

光盘位置：光盘\MR\174　　高级　　趣味指数：★★★

实例说明

3D 图形要比普通图形多一个 Z 轴，Z 轴一般是指 3D 图形的高度。JFreeChart 的 3D 饼图已经设置了默认的饼图高度为 1.2。本实例演示如何修改 3D 饼图的 Z 轴高度，运行效果如图 8.13 所示。

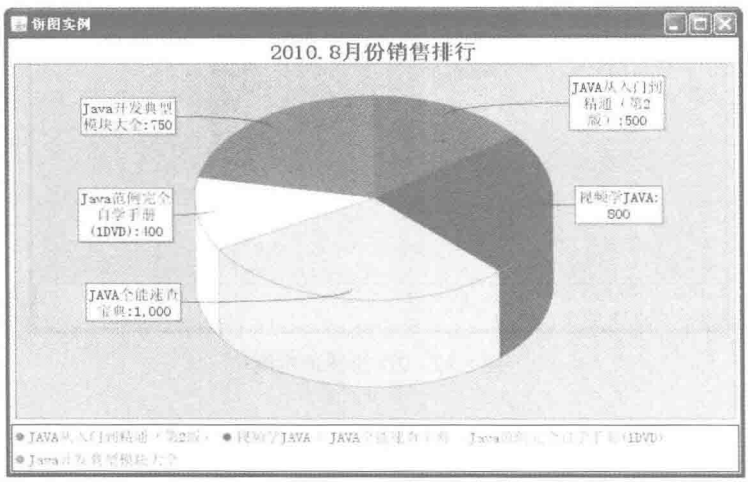


图 8.13　调整 3D 饼图的高度

关键技术

饼图 Z 轴的高度使用 PiePlot3D 的 setDepthFactor()方法设置，语法如下：

```
public void setDepthFactor(double factor)
```

参数说明

factor：表示 Z 轴的高度，范围为 0f~1f，值越小，Z 轴的高度越低；反之，值越大，Z 轴高度越高。

设计过程

（1）新建一个 Java 文件。

（2）在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（3）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 172。

（4）使用 ChartFactory 类根据饼图的数据集合创建有 3D 效果的 JFreeChart 对象，代码见实例 172。

（5）创建 createPiePlot()方法，在方法中获取 3D 饼图的对象，然后设置 Z 轴的高度为 0.3。代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot3D plot  = (PiePlot3D)chart.getPlot();
        //设置 3D 饼图 Z 轴的高度
```

```
        plot.setDepthFactor(0.3f);
        //把 JFreeChart 对象保存到面板中
        setContentPane(new ChartPanel(chart));
    }
```

（6）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把图表窗体显示到屏幕中央。

秘笈心法

心法领悟 174：设置 3D 饼图的特殊立体效果。

一般情况下，3D 饼图的侧面颜色与上面的颜色一致，但是使用 setDarkerSides(boolean darker)方法可以让图形的侧面颜色深于上面的颜色，以增强立体效果。当 darker 为 false 时，表示不使用特殊的立体效果；当 darker 为 true 时，则使用特殊效果。JfreeChart 中，默认情况下 darker 为 false。

实例 175　逆时针旋转 3D 饼图

光盘位置：光盘\MR\175　　高级　　趣味指数：★★★★★

实例说明

JFreeChart 的图表都是静态图像。本实例制作一个可以旋转的 3D 饼图，只要打开统计图，该图就可以逆时针自动旋转，运行效果如图 8.14 所示。

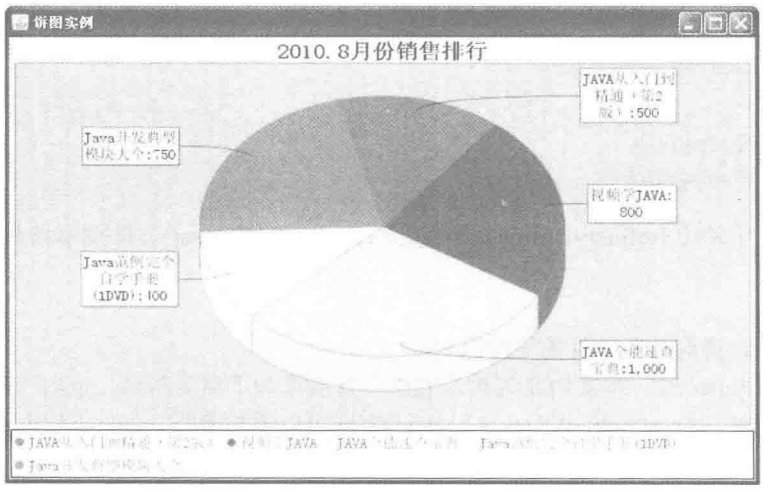


图 8.14　逆时针旋转 3D 饼图

关键技术

使用 javax.swing.Timer 可以让监听在某一个时间开始执行，调用其 start()方法可以执行这个定时器，Timer 类的构造方法如下：

```
public Timer(int delay, ActionListener listener)
```

参数说明

❶ delay：表示开始运行的时间，单位为毫秒。

❷ listener：表示要添加的监听。

设计过程

（1）新建一个 Java 文件，继承 ActionListener 接口，在 ActionListener 接口中实现 actionPerformed()方法，

让饼图的角度增加 1。代码如下：

```
private PiePlot plot;
//饼图的角度
private int angle = 90;
public PieDemo9Listener(PiePlot plot) {
        this.plot =plot;
}
public void actionPerformed( ActionEvent event) {
        this.plot.setStartAngle(this.angle);
        //设置饼图的角度，然后加 1，
        this.angle = this.angle + 1;
        //如果饼图的角度是 360°，把角度设置为 0
        if (this.angle == 360) {
                this.angle = 0;
        }
}
```

（2）新建另一个 Java 文件，继承 JFreeChart 的 ApplicationFrame 类。

（3）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 172。

（4）使用 ChartFactory 类根据饼图的数据集合创建有 3D 效果的 JFreeChart 对象，代码见实例 172。

（5）创建 createPiePlot()方法，在方法中使用定时器执行监听事件。代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        PiePlot3D plot = (PiePlot3D) chart.getPlot();
        ActionListener actionListener = new PieDemo9Listener(plot);
        //添加监听
        Timer timer = new Timer(100, actionListener);
        //启动 timer 时间器
        timer.start();
        //把 JFreeChart 对象保存到面板中
        setContentPane(new ChartPanel(chart));
}
```

（6）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把窗体显示到屏幕中央。

秘笈心法

心法领悟 175：Timer 类对监听事件的支持。

在使用 javax.swing.Timer 时，本实例在其构造方法中直接添加了监听事件，同时 Timer 类还支持对监听事件单独处理，如 addActionListener(ActionListener listener)方法可以添加监听、removeActionListener(ActionListener listener)方法可以删除监听等。

实例 176 顺时针旋转 3D 饼图

高级

光盘位置：光盘\MR\176

趣味指数：★★★★★

实例说明

饼图可以逆时针旋转，也可以顺时针旋转。本实例演示如何让一个 3D 饼图顺时针旋转，运行效果如图 8.15 所示。

关键技术

使用实例 ActionListener 接口的 actionPerformed()方法可以在事件中处理事件，语法如下：

```
public void actionPerformed(ActionEvent event)
```

参数说明

event：表示具体的事件。

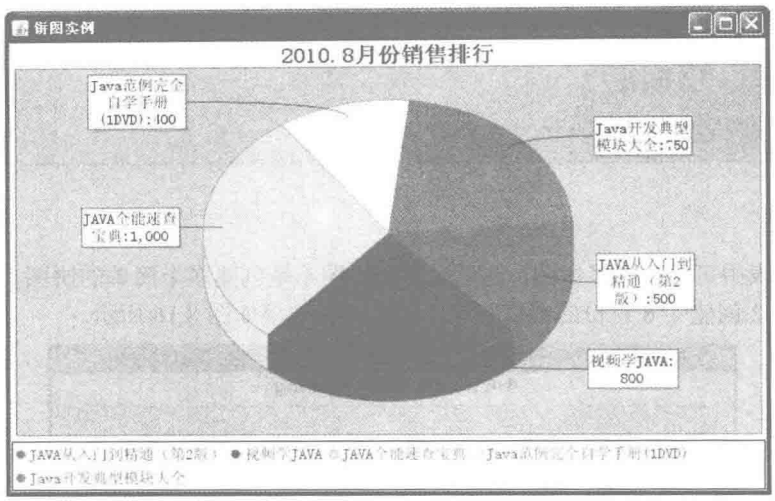


图 8.15　顺时针旋转 3D 饼图

设计过程

（1）新建一个 Java 文件，继承 ActionListener 接口，在 ActionListener 接口中实现 actionPerformed()方法，让饼图的角度增加 1。代码如下：

```
private PiePlot plot;
//饼图的角度
private int angle = 90;
public PieDemo10Listener(PiePlot plot) {
        this.plot =plot;
}
public void actionPerformed( ActionEvent event) {
        this.plot.setStartAngle(this.angle);
        //设置饼图的角度，顺时针旋转则减 1
        this.angle = this.angle - 1;
        //如果饼图的角度减到 0，把角度重新设置为 360
        if (this.angle == 0) {
                this.angle = 360;
        }
}
```

（2）新建另一个 Java 文件，然后继承 JFreeChart 的 ApplicationFrame 类。

（3）使用 DefaultPieDataset 创建一个饼图的数据集合，代码见实例 172。

（4）使用 ChartFactory 类根据饼图的数据集合创建有 3D 效果的 JFreeChart 对象，代码见实例 172。

（5）创建 createPiePlot()方法，在方法中使用定时器执行监听事件，代码见实例 175。

（6）在 main()方法中使用 RefineryUtilities 类的 centerFrameOnScreen()方法把窗体显示到屏幕中央。

秘笈心法

心法领悟 176：实现饼图的旋转。

饼图在旋转时，在监听中改变饼图的角度，从而达到旋转的效果。在监听中，要定义一个初始的角度，最好与默认的角度一致（为 90° ）；否则，在程序开始执行时会有跳跃的感觉。

8.3 多 饼 图

实例 177	实现多饼图	高级
	光盘位置：光盘\MR\177	趣味指数：★★★★★

实例说明

在 JFreeChart 的图表中可以建立多饼图，这里说的多饼图不是创建多个简单的饼图，而是创建一个饼图组。本实例为部门 1 和部门 2 创建 4~6 月份的销售统计图表，运行效果如图 8.16 所示。

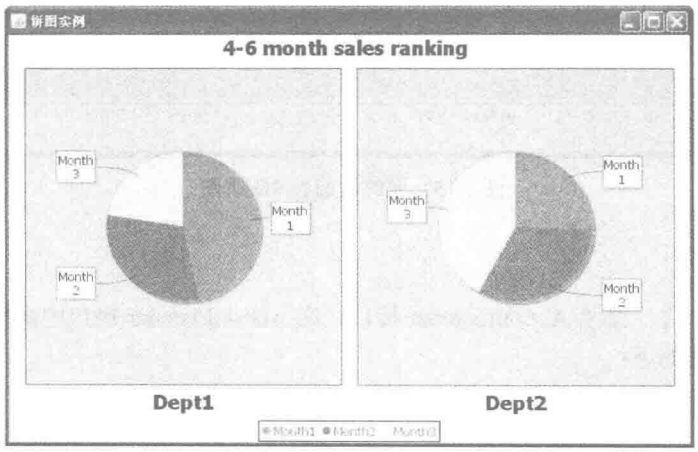


图 8.16 多饼图

关键技术

（1）DatasetUtilities 类含有创建分类数据集的方法，创建完成以后返回 CategoryDataset，createCategoryDataset()方法的语法如下：

```
public static CategoryDataset createCategoryDataset(String rowKeyPrefix, String columnKeyPrefix, double[][] data)
```

参数说明

❶ rowKeyPrefix：表示分类数据集的行名称。

❷ columnKeyPrefix：表示分类数据集的列名称。

❸ data：是一个二维数组，根据行、列的名称存储数据。

（2）ChartFactory 类中有很多方法，用于创建不同的 JFreeChart，如 createMultiplePieChart()方法可以创建一个多饼图的 JFreeChart 对象，其语法如下：

```
createMultiplePieChart(String title,CategoryDataset dataset, TableOrder order, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：表示多饼图的窗体的标题。

❷ dataset：表示多饼图的数据集合。

❸ order：设置多饼图的显示方式。

❹ legend：表示是否使用图示。

❺ tooltips：表示是否生成工具栏提示。

❻ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，然后继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 createDataset()方法，在方法中设置图表数据，创建一个 CategoryDataset 对象，代码如下：

```
private CategoryDataset createDataset() {
        double[ ][ ] data = new double[ ][ ] {
                        { 620, 410, 300 },
                        { 300, 390, 500 } };
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset(
                        "Dept",                         //行名称
                        "Month",                        //列名称
                        data);
        return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中根据数据集生成多饼图的 JFreeChart 对象，并且指定多表图排序方法为行排列，代码如下：

```
private JFreeChart getJFreeChart() {
        //获取数据集
        CategoryDataset dataset = createDataset();
        //生成 JFreeChart 对象
        JFreeChart chart = ChartFactory.createMultiplePieChart(
                        "4-6 month sales ranking ",     //饼图标题
                        dataset,                        //数据集
                        TableOrder.BY_ROW,              //排序方式
                        true, true, false);
        return chart;
}
```

（4）创建 createPiePlot()方法，在方法中获取 JFreeChart 的实例，最后调用父类的方法 setContentPane()把 JFreeChart 添加到面板中，代码如下：

```
public void createPiePlot() {
        JFreeChart chart = getJFreeChart();
        //把 JFreeChart 添加到面板中
        setContentPane(new ChartPanel(chart));
}
```

（5）创建 main()方法，使用 RefineryUtilities 类的 centerFrameOnScreen()方法把窗体显示到屏幕中央，代码如下：

```
public static void main(final String[] args) {
        final PieDemo11 demo = new PieDemo11("饼图实例");
        demo.createPiePlot();
        demo.pack();
        //把窗体显示到屏幕中央
        RefineryUtilities.centerFrameOnScreen(demo);
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 177：创建多饼图的数据集。

在多饼图中创建数据集时，需要指定数据集的行、列名称，生成多饼图以后，JFreeChart 会根据该名称自动为饼图的分类以及各个饼图生成自己的名称，命名规则为以指定的行、列名称为基础，后面添加相应数字。

多饼图乱码

光盘位置：光盘\MR\178

高级

趣味指数：★★★

实例说明

JFreeChart 的多饼图中的中文乱码也是因为字体的原因造成的，普通的饼图乱码解决方式只能解决饼图窗体

和图示的乱码，饼图内部还是会有乱码出现，效果如图 8.17 所示。

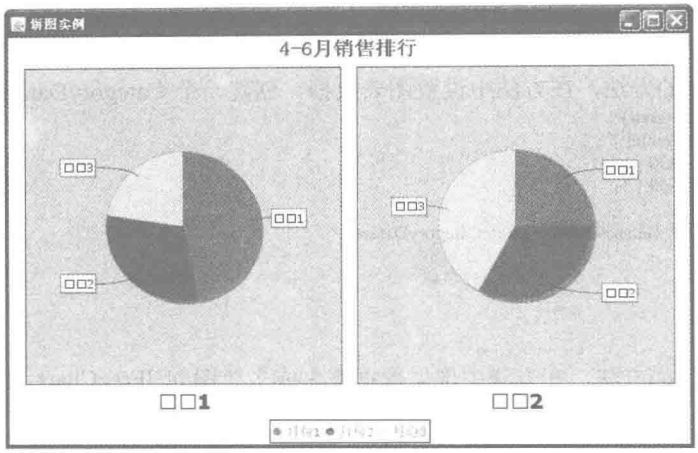


图 8.17　多饼图的乱码

这说明普通的饼图乱码解决方式不适用于多饼图。本实例解决多饼图内部的乱码问题，运行效果如图 8.18 所示。

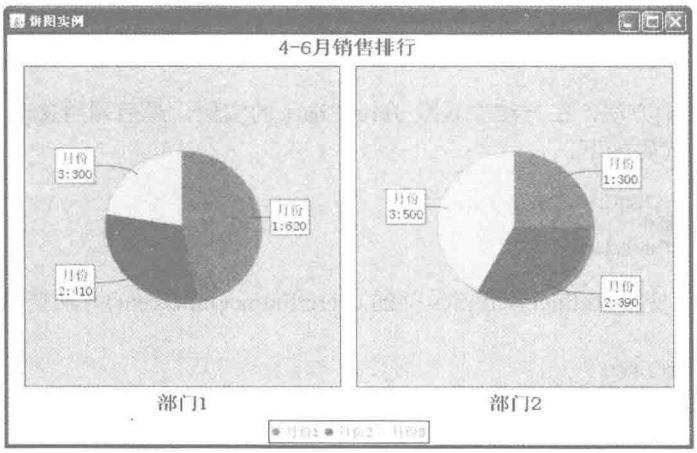


图 8.18　多饼图

关键技术

普通饼图使用 JFreeChart 的 getPlot()方法得到的实例是 PiePlot 的对象，而多饼图使用该方法得到的实例是 MultiplePiePlot 的对象。如果希望得到多饼图的饼图对象，要使用 MultiplePiePlot 的 getPieChart()方法获取 JfreeChart 对象，然后再从 JFreeChart 的对象中获取 PiePlot 的实例，代码如下：

```
JFreeChart chart = getJFreeChart();
//获取多饼图
MultiplePiePlot multiplePiePlot = (MultiplePiePlot) chart.getPlot();
//获取多饼图 JFreeChart 对象
JFreeChart jFreeChart = multiplePiePlot.getPieChart();
PiePlot piePlot = (PiePlot) jFreeChart.getPlot();
```

设计过程

（1）新建一个 Java 文件，然后继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 createDataset()方法，在方法中设置图表数据，创建一个 CategoryDataset 对象，代码见实例 177。

（3）创建 getJFreeChart()方法，在方法中根据数据集生成多饼图的 JFreeChart 对象，并且指定多表图排序方法为行排列，代码见实例 177。

（4）创建 createPiePlot()方法，在方法中获取 JFreeChart 的实例，再使用 JFreeChart 的实例分别获取 MultiplePiePlot、JFreeChart 和 PiePlot 实例。通过修改相应的字体来解决乱码问题，代码如下：

```
public void createPiePlot() {
    JFreeChart chart = getJFreeChart();
    //窗体标题
    TextTitle textTitle = chart.getTitle();
    textTitle.setFont(new Font("宋体", Font.BOLD, 20));
    //图例
    LegendTitle legendTitle = chart.getLegend();
    legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
    //获取多饼图
    MultiplePiePlot multiplePiePlot = (MultiplePiePlot) chart.getPlot();
    JFreeChart jFreeChart = multiplePiePlot.getPieChart();
    //图表标签
    PiePlot piePlot = (PiePlot) jFreeChart.getPlot();
    piePlot.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    piePlot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0}:{1}"));
    //图表标题
    TextTitle textTitle2 = jFreeChart.getTitle();
    textTitle2.setFont(new Font("宋体", Font.BOLD, 20));
    setContentPane(new ChartPanel(chart));
}
```

（5）创建 main()方法，使用 RefineryUtilities 类的 centerFrameOnScreen()方法把窗体显示到屏幕中央，代码如下：

```
public static void main(final String[] args) {
    final PieDemo12 demo = new PieDemo12("饼图实例");
    demo.createPiePlot();
    demo.pack();
    //把窗体显示到屏幕中央
    RefineryUtilities.centerFrameOnScreen(demo);
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 178：处理多饼图乱码主要的几个方面。

多饼图的结构实际分为两部分，一部分是窗体，另一部分是饼图。所以使用解决普通饼图乱码的办法不能完全解决多饼图的乱码问题。多饼图的窗体部分分为窗体的标题和图示，饼图部分又分为饼图标题和标签，所以字体乱码问题也是从这几个方面处理的。

实例 179　多饼图的展示方式

光盘位置：光盘\MR\179　　高级　　趣味指数：★★★★★

实例说明

JFreeChart 的多饼图有两种展示形式，同样的数据分别从不同角度进行统计，根据多饼图的排序方式决定从哪个角度进行图表展示。本实例以多饼图的列为排序依据进行排序，运行效果如图 8.19 所示。

关键技术

在 TableOrder 类中定义了两个常量，用于设置多饼图的排序方法，说明如下。

（1）TableOrder.BY_COLUMN：表示多饼图以列的形式进行排序。

（2）TableOrder.BY_ROW：表示多饼图以行的形式进行排序。

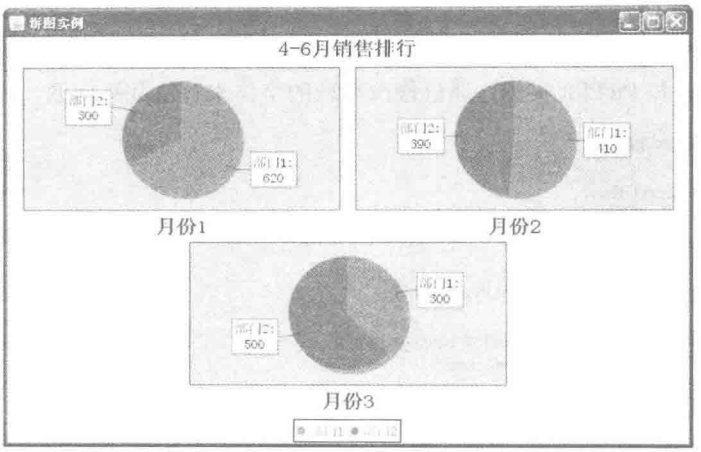


图 8.19　多饼图列排序

设计过程

（1）新建一个 Java 文件，然后继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 createDataset()方法，在方法中设置图表数据，创建一个 CategoryDataset 对象，代码见实例 177。

（3）创建 getJFreeChart()方法，在方法中根据数据集生成多饼图的 JFreeChart 对象，并且指定多饼图排序方法为列排列，代码如下：

```
private JFreeChart getJFreeChart() {
    CategoryDataset dataset = createDataset();
    JFreeChart chart = ChartFactory.createMultiplePieChart(
            "4-6 month sales ranking ",          //饼图标题
            dataset,                             //数据集
            TableOrder. BY_COLUMN,               //排序方式
            true, true, false);
    return chart;
}
```

（4）创建 createPiePlot()方法，在方法中获取 JFreeChart 的实例，再使用 JFreeChart 的实例分别获取 MultiplePiePlot、JFreeChart 和 PiePlot 实例。通过修改相应的字体来解决乱码问题，代码见实例 178。

（5）创建 main()方法，使用 RefineryUtilities 类的 centerFrameOnScreen()方法把窗体显示到屏幕中央。

秘笈心法

心法领悟 179：多饼图的排序。

当多饼图以列为排序依据时，统计图中则以数据集的列为顺序绘制饼图，这时，行的数据被作为饼图中的分类；当多饼图以行为排序依据时，统计图中则以数据集的行为顺序绘制饼图，此时，列的数据被作为饼图中的分类。

实例 180　3D 多饼图

光盘位置：光盘\MR\180　　高级　　趣味指数：★★★★

实例说明

多饼图同普通饼图一样，也可以绘制成 3D 的效果。本实例绘制一个具有 3D 效果的多饼图，运行效果

如图 8.20 所示。

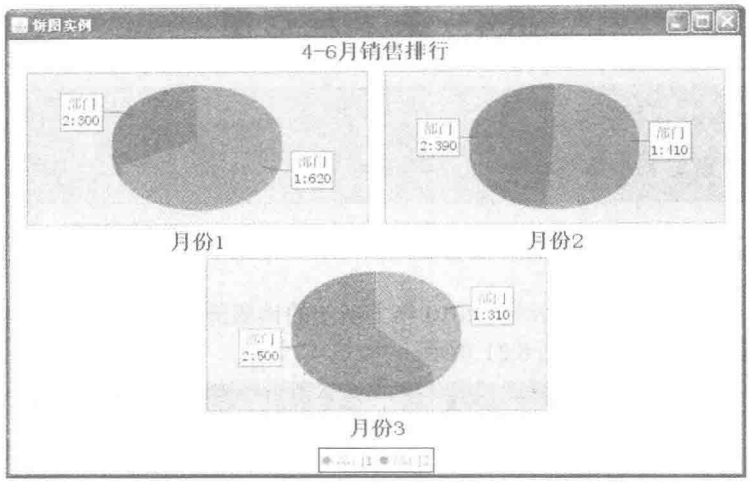


图 8.20　3D 多饼图

关键技术

JFreeChart 已经在 ChartFactory 类中创建了可以绘制 3D 多饼图的方法，该方法的语法如下：

public static JFreeChart createMultiplePieChart3D(String title,CategoryDataset dataset, TableOrder order, boolean legend, boolean tooltips, boolean urls)

参数说明

❶ title：表示 3D 多饼图的窗体的标题。

❷ dataset：表示 3D 多饼图的数据集合。

❸ order：设置 3D 多饼图的显示方式。

❹ legend：表示是否使用图示。

❺ tooltips：表示是否生成工具栏提示。

❻ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，然后继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 createDataset()方法，在方法中设置图表数据，创建一个 CategoryDataset 对象，代码见实例 177。

（3）创建 getJFreeChart()方法，在方法中根据数据集生成多饼图的 JFreeChart 对象，并且指定多饼图排序方法为列排列，代码见实例 179。

（4）创建 createPiePlot()方法，在方法中获取 JFreeChart 的实例，使用该实例分别获取 MultiplePiePlot、JFreeChart 和 PiePlot 实例。通过修改相应的字体来解决乱码问题，代码见实例 178。

（5）创建 main()方法，使用 RefineryUtilities 类的 centerFrameOnScreen()方法把窗体显示到屏幕中央。

秘笈心法

心法领悟 180：设置 3D 多饼图的 Z 轴高度。

在 3D 多饼图中，如果需要设置多饼图的 Z 轴高度，必须首先获取 MultiplePiePlot 对象，再从 MultiplePiePlot 对象中获取 JFreeChart 对象，然后从 JFreeChart 对象中获取 PiePlot3D 实例，使用 PiePlot3D 的 setDepthFactor()方法修改 Z 轴。

8.4 基本柱形图

实例 181	简单柱形图 光盘位置：光盘\MR\181	中级 趣味指数：★★★

实例说明

本实例使用 JFreeChart 的简单柱形图绘制 2010 年上半年的销售情况，X 轴（即横轴）表示销售月份，Y 轴（即竖轴）表示销售数量，运行效果如图 8.21 所示。

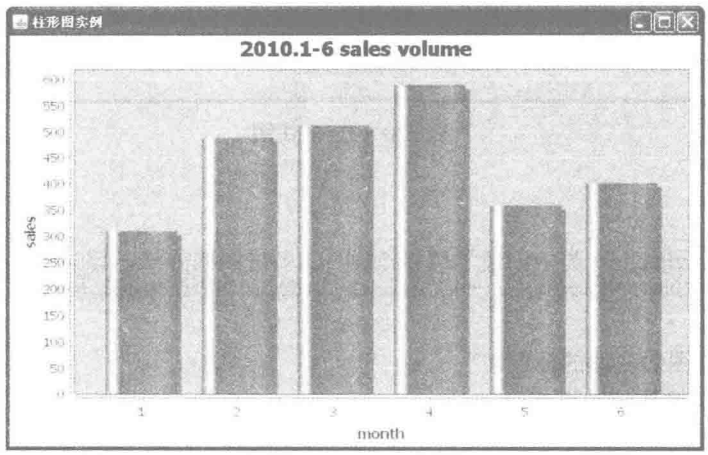


图 8.21 简单柱形图

关键技术

（1）DatasetUtilities 类可以创建一个简单的柱形图数据集，其方法如下：

public static CategoryDataset createCategoryDataset(Comparable rowKey, KeyedValues rowData)

参数说明

❶ rowKey：X 轴的含义，可以作为图示。

❷ rowData：数据集的内容，可以使用 DefaultKeyedValues 类创建数据集。

（2）DefaultKeyedValues 类可以为柱形图提供一个数据集的操作，为数据集添加数据的方法如下：

addValue(Comparable key, double value)

参数说明

❶ key：X 轴的名称。

❷ value：与 X 轴名称相对应的数值。

（3）JFreeChart 提供了创建普通柱形图的方法，创建完成后会返回一个 JFreeChart 对象，createBarChart() 方法的语法如下：

public static JFreeChart createBarChart(String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)

参数说明

❶ title：柱形图的标题。

❷ categoryAxisLabel：柱形图类别标签，即 X 轴的名称。

❸ valueAxisLabel：柱形图数据标签，即 Y 轴的名称。

❹ dataset：柱形图的数据集合。
❺ orientation：柱形图的图表方向。
❻ legend：表示是否使用图示。
❼ tooltips：表示是否生成工具栏提示。
❽ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。
（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码如下：

```
private CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1", 310);
        keyedValues.addValue("2", 489);
        keyedValues.addValue("3", 512);
        keyedValues.addValue("4", 589);
        keyedValues.addValue("5", 359);
        keyedValues.addValue("6", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JfreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",      //图表标题
                        "month",                                                     //X 轴标签
                        "sales",                                                     //Y 轴标签
                        dataset,                                                     //数据集
                        PlotOrientation.VERTICAL,                                    //图表方向：垂直
                        false,                                                       //是否显示图例（对于简单的柱状图必须是 false）
                        false,                                                       //是否生成工具
                        false                                                        //是否生成 URL 链接
                        );

        return chart;
}
```

（4）创建 createPlot()方法，在方法中获取 JFreeChart 对象，调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（5）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo1 barDemo = new BarDemo1("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 181：使用 addValue()方法添加数据时需要注意的问题。

创建柱形图的数据集时使用 DefaultKeyedValues 类，在使用 addValue()方法添加数据时需要注意先后顺序。添加数据集的顺序和 X 轴显示的顺序是一致的。

实例 182 柱形图角度

光盘位置：光盘\MR\182　　中级　　趣味指数：★★★

实例说明

本实例创建一个简单的 JFreeChart 柱形图，用于显示 2010 年上半年销售情况，为了丰富图表的展示效果，将柱形图横向展示出来，运行效果如图 8.22 所示。

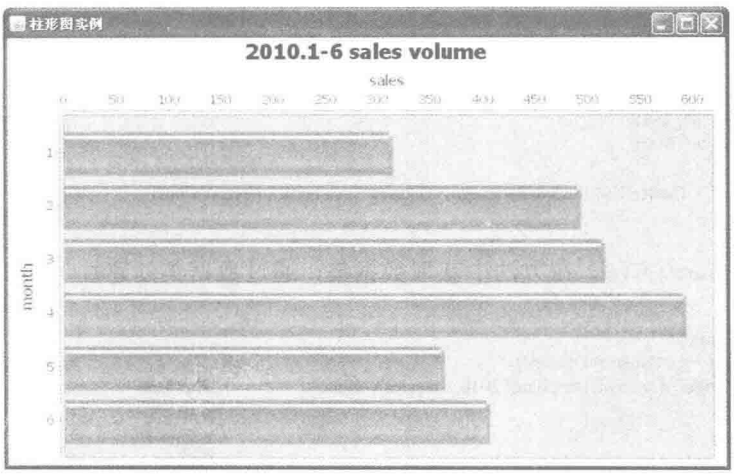


图 8.22　横向柱形图

关键技术

JFreeChart 的 PlotOrientation 类定义了柱形图的显示效果，说明如下：

（1）PlotOrientation.HORIZONTAL：表示柱形图水平显示。

（2）PlotOrientation.VERTICAL：表示柱形图垂直显示。

使用方法如下所示：

```
JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",      //图表标题
        "月份",                                                          //X 轴标签
        "销售量（单位：本）",                                            //Y 轴标签
        dataset,                                                         //数据集
        PlotOrientation. HORIZONTAL,                                     //图表方向：水平
        false,                                                           //是否显示图例（对于简单的柱形图必须是 false）
        false,                                                           //是否生成工具
        false                                                            //是否生成 URL 链接
        );
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 181。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，同时设置柱形图水平显示，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",      //图表标题
                        "month",                                                     //X 轴标签
                        "sales",                                                     //Y 轴标签
                        dataset,                                                     //数据集
                        PlotOrientation. HORIZONTAL,                                 //图表方向：水平
                        false,                                                       //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                       //是否生成工具
                        false                                                        //是否生成 URL 链接
                        );
        return chart;
}
```

（4）创建 createPlot()方法，在方法中获取 JFreeChart 和 TextTitle 对象，修改标题字体，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（5）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo19 barDemo = new BarDemo19("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 182：设置柱形图的角度的常量。

柱形图的角度只有两种，都使用 PlotOrientation 的常量表示，默认情况下使用 PlotOrientation.VERTICAL 常量，表示柱形图以垂直角度显示；还可以使用 PlotOrientation.HORIZONTAL 常量，表示柱形图以水平角度显示。

实例 183　柱形图负值

光盘位置：光盘\MR\183　　中级　　趣味指数：★★★

实例说明

在数据的图表统计中，JFreeChart 的柱形图还支持负值或者空值的显示。本实例显示了柱形图的负值与空值情况，运行效果如图 8.23 所示。

关键技术

JFreeChart 的 DefaultKeyedValues 类允许数据集接收负值，使用 addValue()方法可以向数据集添加数据，语法如下：

```
public void addValue(Comparable key, double value)
```

参数说明

❶ key：表示要添加的关键字。

❷ value：表示要添加的具体数据。

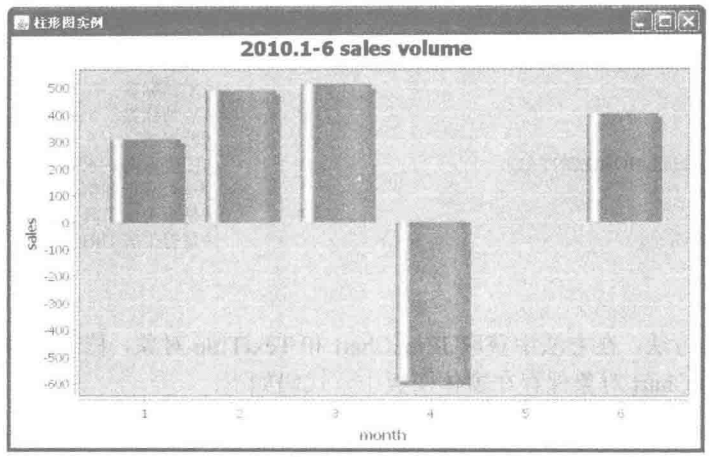


图 8.23　柱形图的负值

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码如下：

```
private CategoryDataset getCategoryDataset() {
    DefaultKeyedValues keyedValues = new DefaultKeyedValues();
    //添加数据
    keyedValues.addValue("1", 310);
    keyedValues.addValue("2", 489);
    keyedValues.addValue("3", 512);
    keyedValues.addValue("4", -589);
    keyedValues.addValue("5", null);
    keyedValues.addValue("6", 402);
    //创建数据集合实例
    CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
    return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，同时设置柱形图竖直显示，代码见实例 181。

（4）创建 createPlot()方法，在方法中获取 JFreeChart 和 TextTitle 对象修改标题字体，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 182。

（5）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    BarDemo20 barDemo = new BarDemo20("柱形图实例");
    barDemo.createPlot();
    barDemo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(barDemo);
    //设置可以显示
    barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 183：柱形图的数据集中保存不同值所达到的不同效果。

柱形图的数据集中允许保存正数、负数、0 和空值。如果保存正数，柱形图显示在 X 轴上方；如果保存负数，柱形图显示在 X 轴下方；如果是 0 值或者是空值，柱形图则不会显示。

8.5　X 坐标轴

实例 184	X 轴字体 光盘位置：光盘\MR\184	中级 趣味指数：★★★

实例说明

本实例创建了一个简单的 JFreeChart 柱形图，绘制 2010 年上半年销售情况，在 X 轴中设置的 1~6 月份的销售额度，如果使用默认字体添加汉字会产生乱码，把字体设为“宋体”即可解决该问题，运行效果如图 8.24 所示。

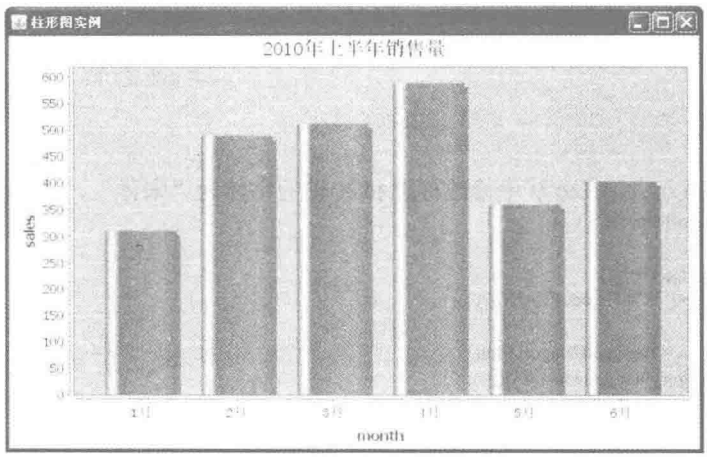


图 8.24　X 轴字体

关键技术

使用 CategoryAxis 类的 setTickLabelFont()方法可以设置图表 X 轴的字体，其语法如下：

```
public void setTickLabelFont(Font font)
```

参数说明

font：表示要设置的 X 轴的字体。

方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
CategoryAxis axis = categoryPlot.getDomainAxis();
//X 轴字体
axis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码如下：

```
private CategoryDataset getCategoryDataset() {
    DefaultKeyedValues keyedValues = new DefaultKeyedValues();
    //添加数据
    keyedValues.addValue("1 月", 310);
    keyedValues.addValue("2 月", 489);
```

```
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
    }
```

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码如下：

```
    private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",      //图表标题
                "month",                                                             //X 轴标签
                "sales",                                                             //Y 轴标签
                dataset,                                                             //数据集
                PlotOrientation.VERTICAL,                                            //图表方向：垂直
                false,                                                               //是否显示图例（对于简单的柱形图必须是 false）
                false,                                                               //是否生成工具
                false                                                                //是否生成 URL 链接
                );
        return chart;
    }
```

（4）创建 updateFont()方法，在方法中修改标题和 X 轴的字体为“宋体”，代码如下：

```
    public void updateFont(JFreeChart chart){
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis axis = categoryPlot.getDomainAxis();
        //X 轴字体
        axis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
    }
```

（5）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码如下：

```
    public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
    }
```

（6）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        BarDemo3 barDemo = new BarDemo3("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 184：坐标轴字体设置特点。

在一个图表中，一般情况下坐标轴的字体要比标题的字体小一些，所以本实例中设置标题字体大小为 20，而 X 轴的字体大小为 14。

实例 185　X 轴标签字体

光盘位置：光盘\MR\185　　中级　　趣味指数：★★

实例说明

X 轴的标签字体与 X 轴字体并不是同时设置的。本实例将 X 轴的标签字体设置为“宋体”，使其能够支持汉字，运行效果如图 8.25 所示。

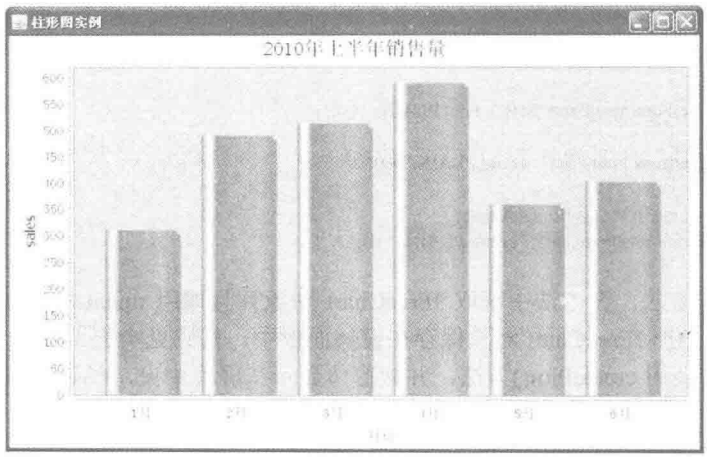


图 8.25　X 轴标签字体

关键技术

从 JFreeChart 的对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，使用 CategoryAxis 的 setLabelFont()方法可以设置 X 轴标签的字体，语法如下：

```
public void setLabelFont(Font font)
```

参数说明

font：表示要设置的 X 轴的标签字体。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
CategoryAxis categoryAxis= categoryPlot.getDomainAxis();
//X 轴标签字体
categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 184。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",      //图表标题
                        "月份",                                                      //X 轴标签
                        "sales",                                                     //Y 轴标签
                        dataset,                                                     //数据集
```

```
                PlotOrientation.VERTICAL,                    //图表方向：垂直
                false,                                       //是否显示图例（对于简单的柱形图必须是 false）
                false,                                       //是否生成工具
                false                                        //是否生成 URL 链接
                );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签和 Y 轴的字体为“宋体”，代码如下：

```
public void updateFont(JFreeChart chart){
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis= categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴字体
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（5）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 184。

（6）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo5 barDemo = new BarDemo5("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 185：灵活设置 X 轴标签字体的大小。

X 轴标签字体与 X 轴上的字体虽然使用两个方法进行设置，但一般情况下两者的字体大小一致。不过也有很多人喜欢把 X 轴标签的字体设置得稍大一些，这完全根据实际需要而定。

实例 186 X 轴标签角度

光盘位置：光盘\MR\186　　中级　　趣味指数：★★★

实例说明

X 轴的标签可以根据需要调整显示角度。本实例将 X 轴的标签设置为垂直显示，运行效果如图 8.26 所示。

关键技术

使用 Axis 类的 setLabelAngle()方法可以设置 X 轴标签旋转的角度，其语法如下：

```
public void setLabelAngle(double angle)
```

参数说明

angle：表示 X 轴标签旋转的弧度，根据弧度和 PI 值可以计算出旋转的角度。

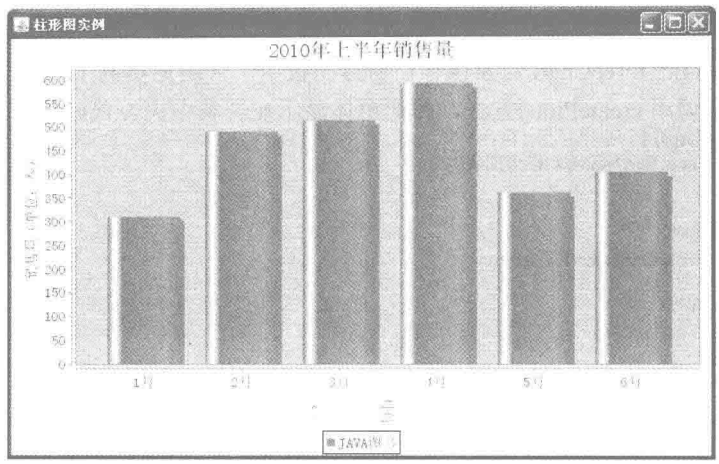


图 8.26　X 轴标签垂直显示

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 184。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",          //图表标题
                        "月份",                                                    //X 轴标签
                        "销售量（单位：本）",                                      //Y 轴标签
                        dataset,                                                   //数据集
                        PlotOrientation.VERTICAL,                                  //图表方向：垂直
                        true,                                                      //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                     //是否生成工具
                        false                                                      //是否生成 URL 链接
                        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴和 Y 轴标签的字体为“宋体”，代码如下：

```
private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（5）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 184。

（6）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo40 barDemo = new BarDemo40("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 186：通过 Axis 抽象类设置标签的旋转角度。

Axis 是一个抽象类，Y 轴与 X 轴分别使用 ValueAxis 和 CategoryAxis 继承了该抽象类，所以 Y 轴与 X 轴设置标签的旋转角度的方法相同，都是使用 Axis 类的 setLabelAngle()方法，不同的是 X 轴使用 CategoryAxis 的实例，而 Y 轴使用 ValueAxis 的实例。

实例 187　X 轴显示情况

光盘位置：光盘\MR\187　　中级　　趣味指数：★★★★

实例说明

在一些情况下，图表的 X 轴是不允许显示出来的。本实例隐藏 X 轴的内容，此时，与 X 轴相关的设置也就没有任何意义了，运行效果如图 8.27 所示。

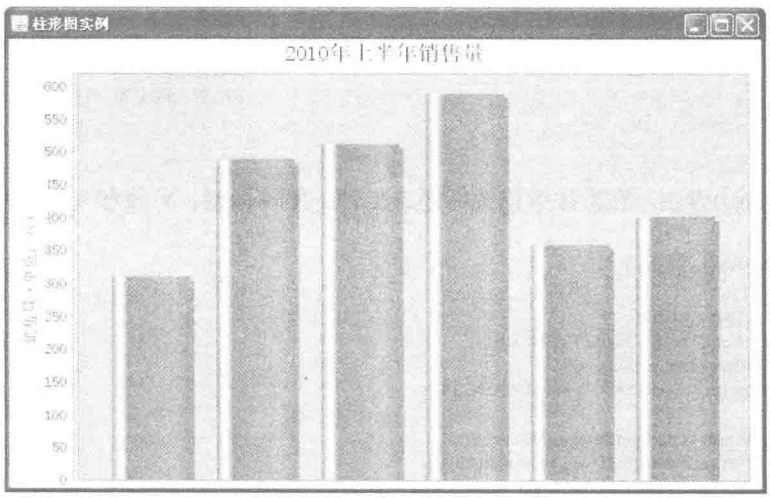


图 8.27　隐藏 X 轴

关键技术

从 JFreeChart 的对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用 setVisible()方法可以隐藏 X 轴，语法如下：

```
public void setVisible(boolean flag)
```

参数说明

flag：表示 X 轴是否显示，当 flag 为 true 时，显示 X 轴；当 flag 为 false 时，不显示 X 轴。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//X 轴（分类轴）
CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
//X 轴是否显示
categoryAxis.setVisible(false);
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 184。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 186。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴和 Y 轴标签的字体为“宋体”，代码见实例 186。

（5）创建 updatePlot()方法，在方法内部隐藏 X 轴，代码如下：

```
private void updatePlot(JFreeChart chart){
        //图表(柱形图)
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //X 轴（分类轴）
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴是否显示
        categoryAxis.setVisible(false);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并调用 updateFont()方法修改字体，再调用 updatePlot 方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //修改图表
        updatePlot(chart);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo7 barDemo = new BarDemo7("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 187：使用 JFreeChart 隐藏 X 轴。

使用 JFreeChart 隐藏 X 轴时，会同时隐藏 X 轴的尺度、尺度文字以及标签，所以在没有特别要求的情况下，X 轴是不需要隐藏的，其默认设置也处于显示的状态。

实例 188 X 轴尺度线颜色

光盘位置：光盘\MR\188

中级

趣味指数：★★★

实例说明

在 JFreeChart 中，柱形图中的 X 轴尺度线的颜色是可以改变的。本实例把 X 轴的尺度线设置为绿色，运行效果如图 8.28 所示。

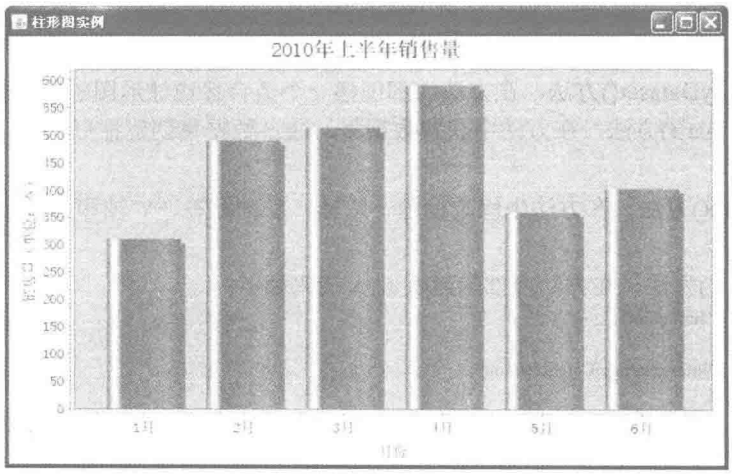


图 8.28　X 轴尺度颜色

关键技术

从 JFreeChart 对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用 setAxisLinePaint()方法为 X 轴的尺度设置颜色，语法如下：

```
public void setAxisLinePaint(Paint paint)
```

参数说明

paint：表示 X 轴尺度设置的颜色。

其方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//X 轴（分类轴）
CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
//X 轴尺度颜色
categoryAxis.setAxisLinePaint(Color.GREEN);
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 184。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 186。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴和 Y 轴标签的字体为“宋体”。代码见实例 186。

（5）创建 updatePlot()方法，将 X 轴尺度颜色设置为绿色，代码如下：

```
private void updatePlot(JFreeChart chart){
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //X 轴（分类轴）
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴尺度颜色
        categoryAxis.setAxisLinePaint(Color.GREEN);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 187。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo9 barDemo = new BarDemo9("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 188：通过 Color 类改变 X 轴与 X 轴字体的颜色。

JFreeChart 的 X 轴的尺度线颜色默认为灰色，对应于 Java 的 Color 类的常量 Color.gray，本实例设置 X 轴为绿色，使用了 Color 类的常量 Color.GREEN。如果希望修改 X 轴字体的颜色，那么可以使用 CategoryAxis 类的 setTickLabelPaint()方法。

实例 189　隐藏 X 轴尺度线

光盘位置：光盘\MR\189　　中级　　趣味指数：★★★

实例说明

X 轴的尺度线即图表的 X 轴下方的一条有刻度的线，在使用时可以根据需要将其隐藏。本实例隐藏了 X 轴的尺度线，运行效果如图 8.29 所示。

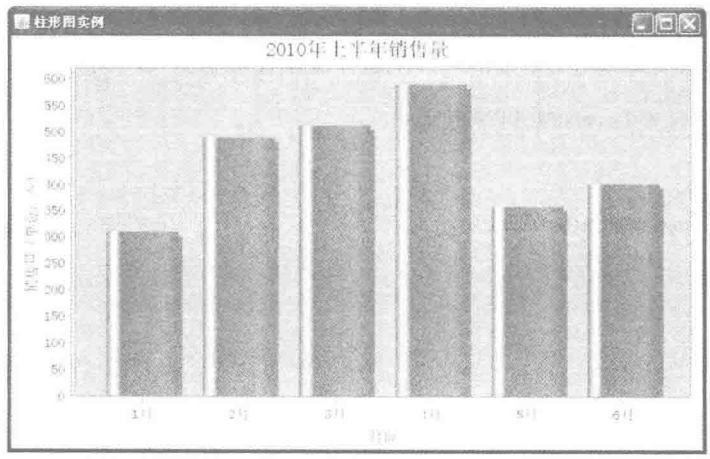


图 8.29　隐藏 X 轴尺度线

关键技术

从 JFreeChart 的对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用 setAxisLineVisible()方法隐藏 X 轴尺度线，语法如下：

```
public void setAxisLineVisible(boolean visible)
```

参数说明

visible：表示 X 轴尺度线是否显示，当 visible 为 true 时，显示尺度线；当 visible 为 false 时，不显示尺度线。方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//X 轴字体
CategoryAxis axis = categoryPlot.getDomainAxis();
//隐藏尺度线
axis.setAxisLineVisible(false);
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 184。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 186。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴和 Y 轴标签的字体为“宋体”，代码见实例 186。

（5）创建 updatePlot()方法，在方法中隐藏 X 轴尺度，代码如下：

```
private void updatePlot(JFreeChart chart){
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //x 轴字体
        CategoryAxis axis = categoryPlot.getDomainAxis();
        //隐藏尺度线
        axis.setAxisLineVisible(false);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 187。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo11 barDemo = new BarDemo11("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 189：X 轴上的尺度线。

X 轴的尺度默认情况下是显示的，用户可以根据需要决定是否使用尺度线。如果隐藏 X 轴的尺度，尺度线上的刻度会自动显示在图表的下边框上。

实例 190　X 轴尺度线笔触

光盘位置：光盘\MR\190　　中级　　趣味指数：★★

实例说明

X 轴的尺度线可以根据使用情况进行调整，例如加粗尺度线、修改尺度的样式等。本实例把 X 轴的尺度线加粗到 5，运行效果如图 8.30 所示。

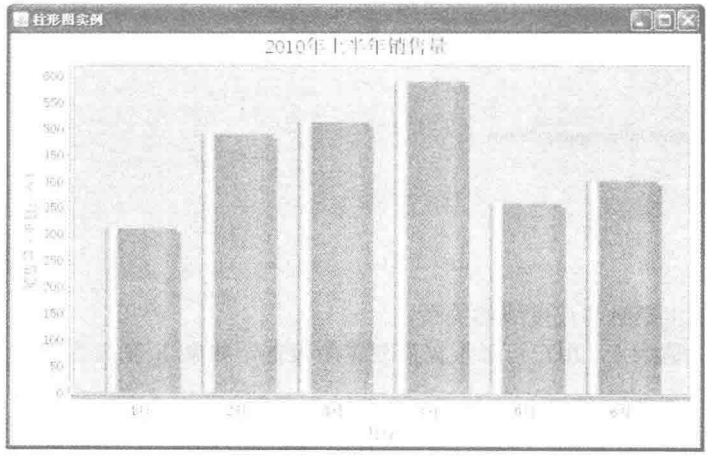


图 8.30　加粗 X 轴尺度线笔触

关键技术

从 JFreeChart 的对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用 BasicStroke()方法加粗 X 轴尺度线，语法如下：

```
public BasicStroke(float width)
```

参数说明

width：表示笔触要加粗的数值。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//X 轴
CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
//尺度笔触
categoryAxis.setAxisLineStroke(new BasicStroke(5));
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 184。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 186。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴和 Y 轴标签的字体为“宋体”，代码见实例 186。

（5）创建 updatePlot()方法，在方法中加粗 X 轴尺度线，代码如下：

```
private void updatePlot(JFreeChart chart) {
```

```
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //X 轴字体
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //尺度笔触
        categoryAxis.setAxisLineStroke(new BasicStroke(5));
    }
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 187。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo13 barDemo = new BarDemo13("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 190：在 X 轴上笔触与尺度线的关系。

X 轴尺度线的笔触默认值为 1，用户可以根据需要重新设置，如果 X 轴尺度线被设置为隐藏状态，那么尺度笔触的设置是不可用的。

实例 191　X 轴尺度标签角度

光盘位置：光盘\MR\191　　中级　　趣味指数：★★★

实例说明

如果 X 轴的尺度之间的距离较小，而标签名称又太长，则相邻的两个标签文字可能出现重叠现象，本实例对 X 轴的标签显示方向进行了调整（旋转了一个角度），这样就避免了上述情况的发生，运行效果如图 8.31 所示。

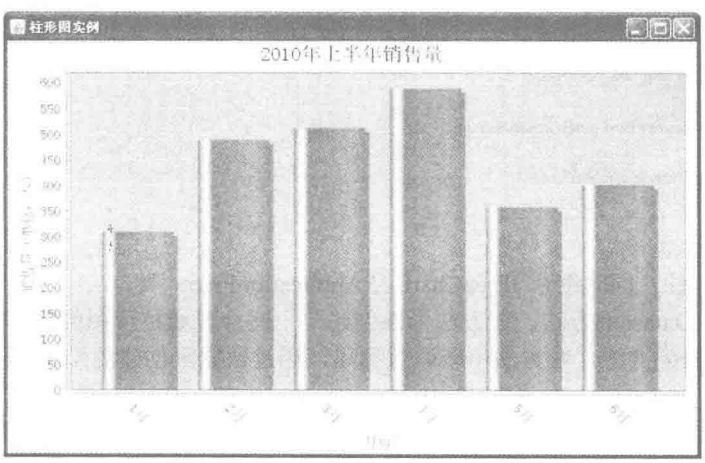


图 8.31　旋转 X 轴尺度标签

关键技术

从 JFreeChart 的对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用 setCategoryLabelPositions()方法可以修改 X 轴尺度标签的角度，语法如下：

```
public void setCategoryLabelPositions(CategoryLabelPositions positions)
```

参数说明

positions：表示要修改的尺度标签的角度，必须使用 CategoryLabelPositions 类实现。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 184。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 186。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴和 Y 轴标签的字体为“宋体”，代码见实例 186。

（5）创建 updatePlot()方法，在方法中把 X 轴尺度标签顺时针旋转 45°，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //X 轴
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //尺度间隔
        categoryAxis.setCategoryLabelPositions(CategoryLabelPositions.DOWN_45);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 187。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo15 barDemo = new BarDemo15("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 191：CategoryLabelPositions 类中定义角度的常量。

CategoryLabelPositions 类中定义了几个常用的常量分别用来表示不同的角度，具体如下。

（1）DOWN_45：表示分类轴尺度标签顺时针旋转 45°。

（2）DOWN_90：表示分类轴尺度标签顺时针旋转 90°。

（3）UP_45：表示分类轴尺度标签逆时针旋转 45°。

（4）UP_90：表示分类轴尺度标签逆时针旋转 90°。

实例 192 X 轴分类的间距

光盘位置：光盘\MR\192　　中级　　趣味指数：★★

实例说明

JFreeChart 自动生成的柱形图中，每个分类柱的宽度是由分类的数量决定的，如果分类较少，那么分类柱的宽度就会很宽，图表看起来很不协调。本实例通过调整分类之间的间隔，让整个图表更美观，运行效果如图 8.32 所示。

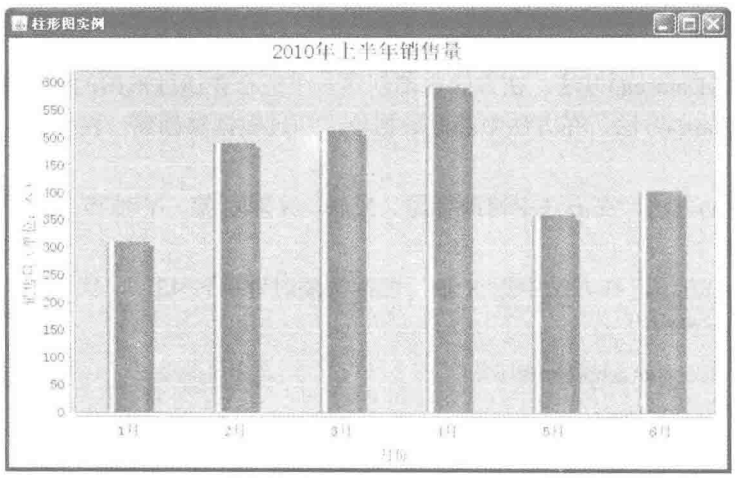


图 8.32　X 轴分类的间距

关键技术

从 JFreeChart 对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用 setCategoryMargin()方法可以调整 X 轴分类的间距，语法如下：

```
public void setCategoryMargin(double margin)
```

参数说明

margin：表示 X 轴分类之间的距离，数值越大，间距越大。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 184。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 186。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴和 Y 轴标签的字体为“宋体”，代码见实例 186。

（5）创建 updatePlot()方法，在方法中把 X 轴分类之间的距离设为 0.5，代码如下：

```
private void updatePlot(JFreeChart chart){
    //图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    //X 轴
    CategoryAxis axis = categoryPlot.getDomainAxis();
    //X 轴分类间距
```

```
        axis.setCategoryMargin(0.5);
    }
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 187。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo17barDemo = new BarDemo17("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 192：setCategoryMargin()方法设置 X 轴分类的间距。

setCategoryMargin()方法用于设置 X 轴分类的间距，JFreeChart 默认的间距值为 0.2，数值越小，分类之间的距离越小，则柱形的宽度越宽；反之，数值越大，分类之间的距离越大，柱形的宽度越窄。

实例 193　X 轴分类与原点的间距

中级

光盘位置：光盘\MR\193

趣味指数：★★★

实例说明

X 轴上柱形图可以根据实际需要向左或向右进行调整。本实例把 X 轴上的柱形图向右进行了调整，运行效果如图 8.33 所示。

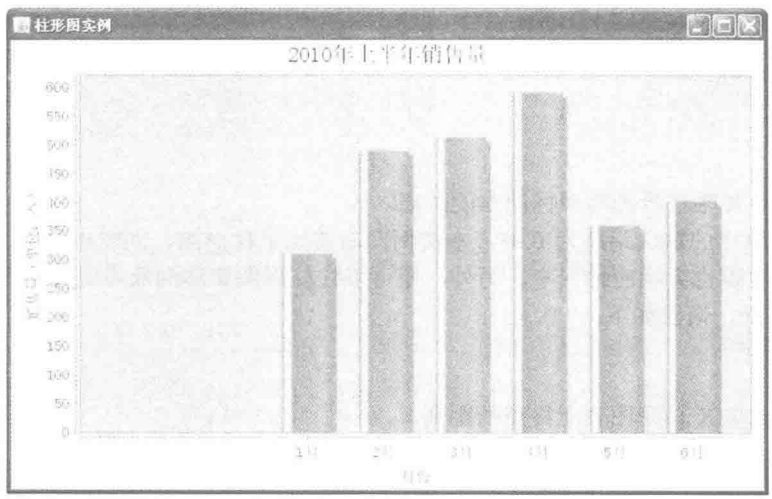


图 8.33　向右移动柱形图

关键技术

从 JFreeChart 对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 CategoryAxis 对象，在 CategoryAxis 对象中使用 setLowerMargin()方法可以修改 X 轴上柱形图与坐标轴原点的距

离，代码如下：

```
public void setLowerMargin(double margin)
```

参数说明

margin：表示 X 轴上柱形图与坐标轴原点的距离，数值越大，距离越大。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 184。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 186。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 186。

（5）创建 updatePlot()方法，在方法中把 X 轴上的柱形图与坐标原点的距离设置为 0.3，代码如下：

```
private void updatePlot(JFreeChart chart){
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //X 轴
        CategoryAxis axis = categoryPlot.getDomainAxis();
        //与原点的距离
        axis.setLowerMargin(0.3);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 187。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo18barDemo = new BarDemo18("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 193：调整 X 轴柱形图与坐标原点的距离。

X 轴柱形图与坐标原点的默认距离为 0.05，本实例向右移动了柱形图，同时也缩短了柱形图在图表上的展示长度，那么柱形图的宽度也会相应地变窄。另外，还可以让柱形图集体向原点靠拢，可使用 CategoryAxis 的 setUpperMargin()方法实现，语法如下：

```
setUpperMargin(double margin)。
```

参数说明

margin：表示柱形图与 X 轴在图表上终点的距离。

实例 194 X 轴的显示位置

光盘位置：光盘\MR\194　　中级　　趣味指数：★★★

实例说明

一般的柱形图都是垂直显示的，X 轴都在图表的下方，有时根据需要可能会对 X 轴进行调整。本实例演示

把 X 轴显示在图表上方，运行效果如图 8.34 所示。

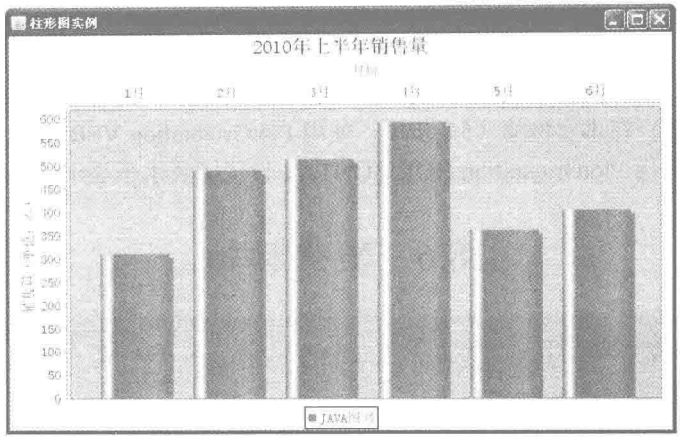


图 8.34　设置 X 轴显示位置

关键技术

在 CategoryPlot 中使用 setDomainAxisLocation()方法可以设置 X 轴在图表中的显示位置，setDomainAxisLocation()方法的语法如下：

```
public void setDomainAxisLocation(AxisLocation location)
```

参数说明

location：表示 X 轴的显示位置。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 184。

（3）创建 getJFreeChart()方法，在方法中获取数据集 CategoryDataset 实例，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 186。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 186。

（5）创建 updatePlot()方法，在方法中设置 X 轴显示在图表上方，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置 X 轴显示位置
        categoryPlot.setDomainAxisLocation(AxisLocation.TOP_OR_RIGHT);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 187。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo38 barDemo = new BarDemo38("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 194：使用 AxisLocation 类设置图表 X 轴显示位置。

使用 AxisLocation 类中的常量 AxisLocation.TOP_OR_RIGHT 可以为图表设置 X 轴显示位置，可以在图表的上方，也可以在图表右侧。在本实例第（5）步中，使用 PlotOrientation.VERTICAL 设置图表垂直显示，X 轴则显示在图表上方；如果使用 PlotOrientation.HORIZONTAL 设置图表水平显示，X 轴则显示在图表右侧。

8.6 Y 坐标轴

实例 195	Y 轴字体	中级
	光盘位置：光盘\MR\195	趣味指数：★★★

实例说明

Y 轴是数据轴，一般情况下只是用来展示数值，很少使用汉字，使用数值不会产生乱码，不过依然可以修改 Y 轴的字体，运行效果如图 8.35 所示。

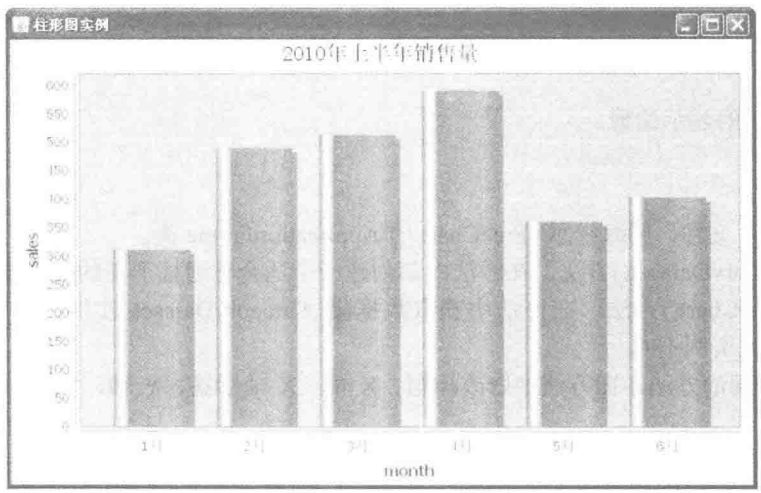


图 8.35 Y 轴字体

关键技术

从 JFreeChart 对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中可以设置 Y 轴的字体，语法如下：

```
public void setTickLabelFont(Font font)
```

参数说明

font：表示 Y 轴的字体。

其方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
ValueAxis valueAxis = categoryPlot.getRangeAxis();
//Y 轴字体
valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码如下：

```
private CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        //添加数据
        keyedValues.addValue("1 月", 310);
        keyedValues.addValue("2 月", 489);
        keyedValues.addValue("3 月", 512);
        keyedValues.addValue("4 月", 589);
        keyedValues.addValue("5 月", 359);
        keyedValues.addValue("6 月", 402);
        //创建数据集合实例
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
        return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",      //图表标题
                        "month",                                                     //X 轴标签
                        "sales",                                                     //Y 轴标签
                        dataset,                                                     //数据集
                        PlotOrientation.VERTICAL,                                    //图表方向：垂直
                        false,                                                       //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                       //是否生成工具
                        false                                                        //是否生成 URL 链接
                        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改标题和 Y 轴的字体为“宋体”，代码如下：

```
public void updateFont(JFreeChart chart){
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis axis = categoryPlot.getDomainAxis();
        //Y 轴字体
        axis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（5）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（6）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo4 barDemo = new BarDemo4("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 195：X 轴与 Y 轴之间的关系。

X 与 Y 轴是相辅相成的，一般情况下，X 轴和 Y 轴的字体及字体大小是一致的，所以当 X 轴字体设置为 14 时，Y 轴的字体也应该设置为 14。

实例 196　Y 轴标签字体

光盘位置：光盘\MR\196　　中级　　趣味指数：★★

实例说明

JFreeChart 的柱形图中，表示 Y 轴的含义的文字被称为 Y 轴的标签。本实例为 Y 轴的标签设置新字体，使其能够支持汉字，运行效果如图 8.36 所示。

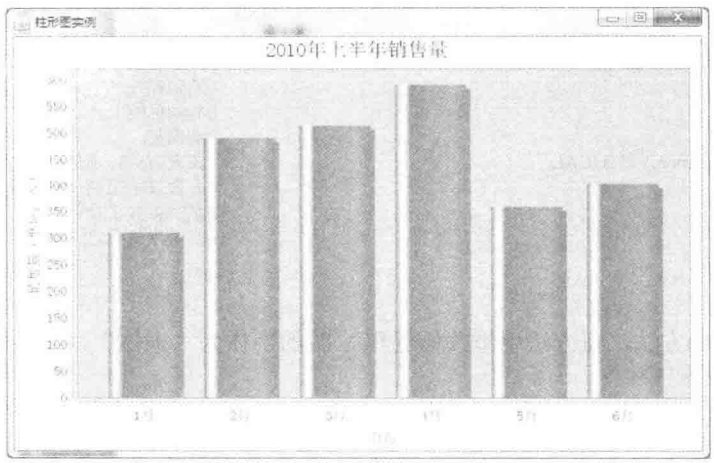


图 8.36　Y 轴标签字体

关键技术

从 JFreeChart 的对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中可以设置 Y 轴标签的字体，语法如下：

```
public void setLabelFont(Font font)
```

参数说明

font：表示 Y 轴标签的字体。

其方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
ValueAxis valueAxis = categoryPlot.getRangeAxis();
//Y 轴标签字体
valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",          //图表标题
                        "month",                                                         //X 轴标签
                        "销售量（单位：本）",                                            //Y 轴标签
                        dataset,                                                         //数据集
                        PlotOrientation.VERTICAL,                                        //图表方向：垂直
                        false,                                                           //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                           //是否生成工具
                        false                                                            //是否生成 URL 链接
                        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码如下：

```
public void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（5）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 195。

（6）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo6 barDemo = new BarDemo6("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 196：设置 Y 轴的标签使之更准确地表达图表的内容。

Y 轴是数值轴，Y 轴的标签一般用来表示 Y 轴的含义，可以在 Y 轴的标签后面为 Y 轴添加数值的单位，更准确地表达图表的内容。

实例 197　Y 轴显示情况

中级

光盘位置：光盘\MR\197　　趣味指数：★★

实例说明

在 JFreeChart 中，柱形图的 Y 轴也可以被隐藏。本实例把 Y 轴的内容隐藏起来，Y 轴被隐藏后，与其相关的设置也就没有任何意义了，运行效果如图 8.37 所示。

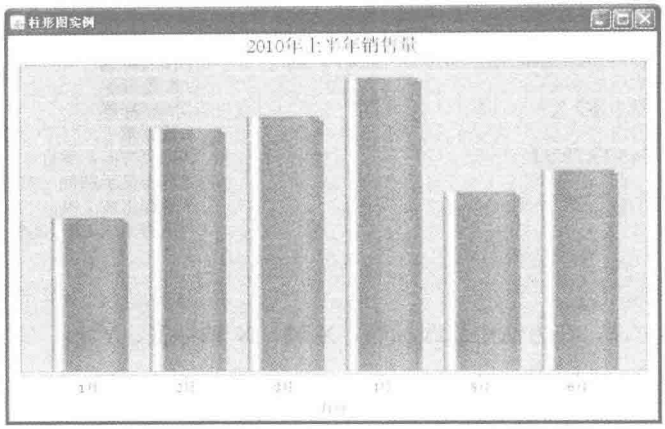


图 8.37　隐藏 Y 轴

关键技术

从 JFreeChart 对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中使用 setVisible()方法可以隐藏 Y 轴，语法如下：

```
public void setVisible(boolean flag)
```

参数说明

flag：表示 Y 轴是否显示，当 flag 为 true 时，显示 Y 轴；当 flag 为 false 时，不显示 Y 轴。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//Y 轴（数值轴）
ValueAxis valueAxis = categoryPlot.getRangeAxis();
//Y 轴是否显示
valueAxis.setVisible(false);
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",          //图表标题
                        "月份",                                                          //X 轴标签
                        "销售量（单位：本）",                                            //Y 轴标签
                        dataset,                                                         //数据集
                        PlotOrientation.VERTICAL,                                        //图表方向：垂直
                        false,                                                           //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                           //是否生成工具
                        false                                                            //是否生成 URL 链接
                        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，获取 ValueAxis 的实例，使用 ValueAxis 的实例隐藏 Y 轴，代码如下：

```
private void updatePlot(JFreeChart chart){
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴（数值轴）
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
```

```
        //Y 轴是否显示
        valueAxis.setVisible(false);
    }
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码如下：

```
    public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //修改图表
        updatePlot(chart);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
    }
```

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        BarDemo8 barDemo = new BarDemo8("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 197：使用 JFreeChart 隐藏 Y 轴。

JFreeChart 在隐藏 Y 轴时，会同时隐藏 Y 轴的尺度、尺度文字以及标签，所以在没有特别要求的情况下，Y 轴是不需要隐藏的，Y 轴的默认设置也是处于显示的状态。

实例 198　Y 轴尺度线颜色

光盘位置：光盘\MR\198　　中级　　趣味指数：★★★

实例说明

在 JFreeChart 中，柱形图中的 Y 轴尺度线的颜色应该和 X 轴尺度线颜色一致。本实例把 Y 轴的尺度线改成和 X 轴同样的绿色，运行效果如图 8.38 所示。

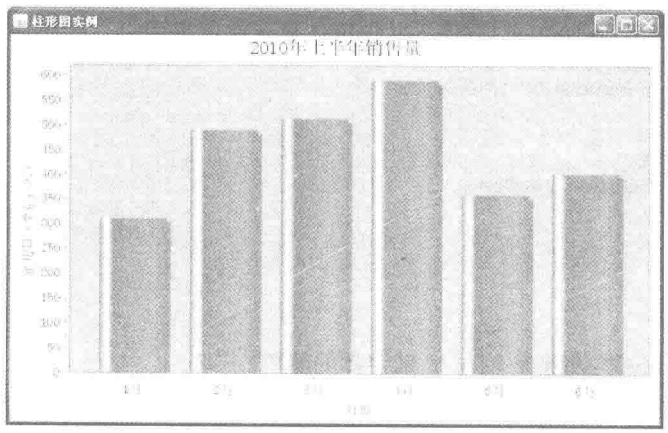


图 8.38　Y 轴尺度线颜色

关键技术

从 JFreeChart 对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中为 Y 轴尺度设置颜色，语法如下：

```
public void setAxisLinePaint(Paint paint)
```

参数说明

paint：表示 Y 轴尺度设置的颜色。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//Y 轴（数值轴）
ValueAxis valueAxis = categoryPlot.getRangeAxis();
//Y 轴尺度颜色
valueAxis.setAxisLinePaint(Color.GREEN);
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，将 X 轴和 Y 轴尺度颜色设置为绿色，代码如下：

```
private void updatePlot(JFreeChart chart){
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴（数值轴）
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴尺度颜色
        valueAxis.setAxisLinePaint(Color.GREEN);
        //X 轴（分类轴）
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴尺度颜色
        categoryAxis.setAxisLinePaint(Color.GREEN);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo10 barDemo = new BarDemo10("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 198：使用 Color 类设置 Y 轴与 Y 轴字体的颜色。

JFreeChart 的 Y 轴的尺度线的颜色默认为灰色，对应对 Java 的 Color 类的常量 Color.gray。本实例设置 Y 轴为绿色，使用了 Color 常量 Color.GREEN。如果希望修改 Y 轴字体的颜色，可以使用 ValueAxis 的 setTickLabelPaint()

方法。

实例 199　隐藏 Y 轴尺度线

光盘位置：光盘\MR\199　　中级　　趣味指数：★★

实例说明

Y 轴的尺度线是指在图表的 Y 轴左侧的一条有刻度的线，在使用时可以根据需要将其去掉。本实例把 Y 轴的尺度线去掉，运行效果如图 8.39 所示。

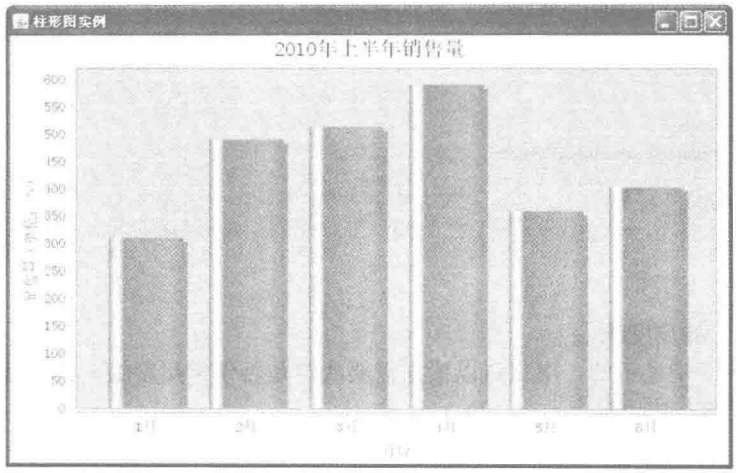


图 8.39　隐藏 Y 轴尺度线

关键技术

从 JFreeChart 对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中隐藏 Y 轴尺度线，语法如下：

```
public void setAxisLineVisible(boolean visible)
```

参数说明

visible：表示 Y 轴尺度线是否显示，当 visible 为 true 时，则显示尺度线；为 false 时，则不显示尺度线。

使用方法如下：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//Y 轴
ValueAxis valueAxis = categoryPlot.getRangeAxis();
//尺度线
valueAxis.setAxisLineVisible(false);
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中隐藏 Y 轴尺度线，代码如下：

```
private void updatePlot(JFreeChart chart){
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //尺度线
        valueAxis.setAxisLineVisible(false);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo12 barDemo = new BarDemo12("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 199：Y 轴上的尺度线。

Y 轴的尺度默认情况下是显示的，用户可以根据需要决定是否使用尺度线。尺度线被隐藏后，尺度线上的刻度会自动地显示在图表的左边边框上。

实例 200　Y 轴尺度线笔触

光盘位置：光盘\MR\200　　中级　　趣味指数：★★★

实例说明

Y 轴的尺度线可以根据使用情况进行调整，例如加粗尺度线、修改尺度的样式等。本实例把 Y 轴的尺度线加粗到 5，运行效果如图 8.40 所示。

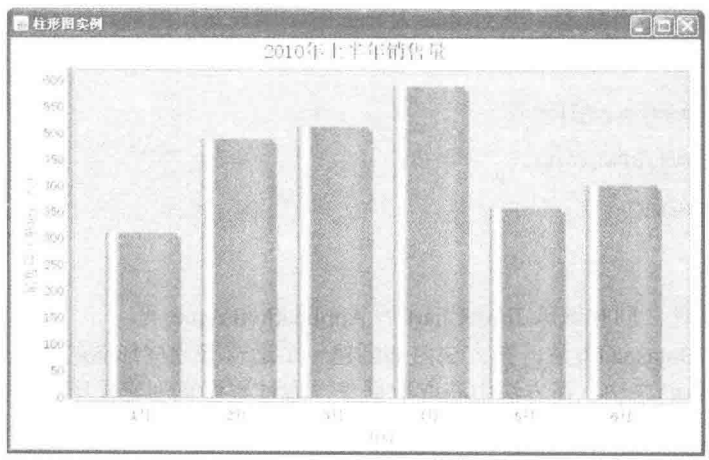


图 8.40　加粗 Y 轴尺度线笔触

关键技术

从 JFreeChart 对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中使用 BasicStroke()方法加粗 Y 轴尺度线，语法如下：

```
public BasicStroke(float width)
```

参数说明

width：表示笔触要加粗的数值。

使用方法如下所示：

```
//图表（柱形图）
CategoryPlot categoryPlot = chart.getCategoryPlot();
//Y 轴
ValueAxis valueAxis = categoryPlot.getRangeAxis();
//尺度笔触
valueAxis.setAxisLineStroke(new BasicStroke(5));
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中加粗 Y 轴尺度笔触，代码如下：

```
private void updatePlot(JFreeChart chart){
        //图表（柱形图）
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //尺度笔触
        valueAxis.setAxisLineStroke(new BasicStroke(5));
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot 方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo14 barDemo = new BarDemo14("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 200：在 Y 轴上笔触与尺度线的关系。

Y 轴尺度线的笔触默认值为 1，用户可以根据需要决定笔触值，如果 Y 轴尺度线被设置为隐藏状态，那么尺度笔触的设置即是没有必要的。

实例 201 Y 轴尺度标签角度

光盘位置：光盘\MR\201　　中级　　趣味指数：★★★

实例说明

Y 轴尺度的标签角度可以根据需要进行调整。本实例把 Y 轴尺度的标签逆时针旋转 90°，运行效果如图 8.41 所示。

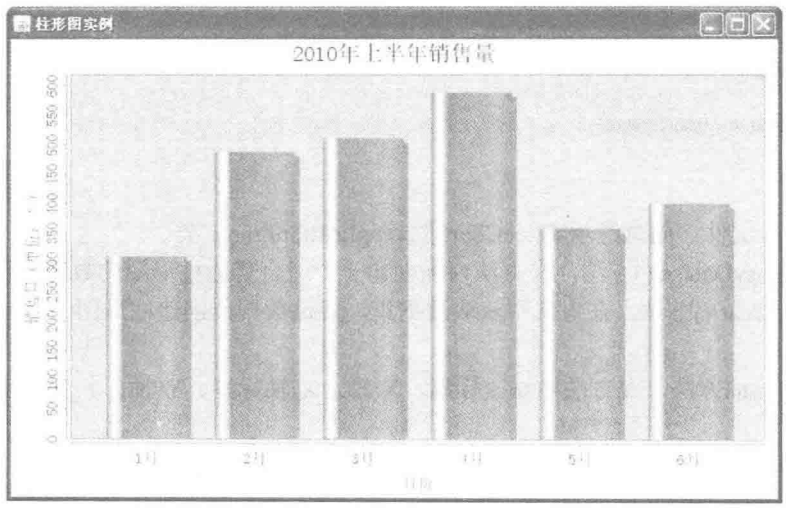


图 8.41　Y 轴尺度标签角度

关键技术

从 JFreeChart 对象的 getCategoryPlot()方法中可以获取 CategoryPlot 的实例，使用 CategoryPlot 实例能得到 ValueAxis 对象，在 ValueAxis 对象中使用 setVerticalTickLabels()方法可以修改 Y 轴尺度标签的角度，语法如下：

```
public void setVerticalTickLabels(boolean flag)
```

参数说明

flag：表示 Y 轴尺度标签是否垂直，当 flag 为 true 时，表示 Y 轴尺度标签是垂直的；为 false 时，表示 Y 轴尺度标签是水平的。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中设置 Y 轴尺度标签垂直，代码如下：

```
private void updatePlot(JFreeChart chart){
    //图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    //Y 轴
    ValueAxis valueAxis = categoryPlot.getRangeAxis();
```

```
        //Y 轴尺度标签
        valueAxis.setVerticalTickLabels(true);
    }
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo16barDemo = new BarDemo16("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 201：JFreeChart 控制 Y 轴尺度标签的角度。

Y 轴尺度标签的角度与 X 轴不同，X 轴尺度标签中 JFreeChart 提供了 5 个角度，分别使用不同的常量来表示，而 Y 轴尺度标签的角度控制中 JFreeChart 只提供了一个 boolean 类型的方法。

实例 202　Y 轴起始值

光盘位置：光盘\MR\202　　中级　　趣味指数：★★★

实例说明

一般情况下，柱形图的起始值默认为 0。本实例修改了柱形图中 Y 轴的起始值，让图表有更多的区域展示不同柱形之间的区别，运行效果如图 8.42 所示。

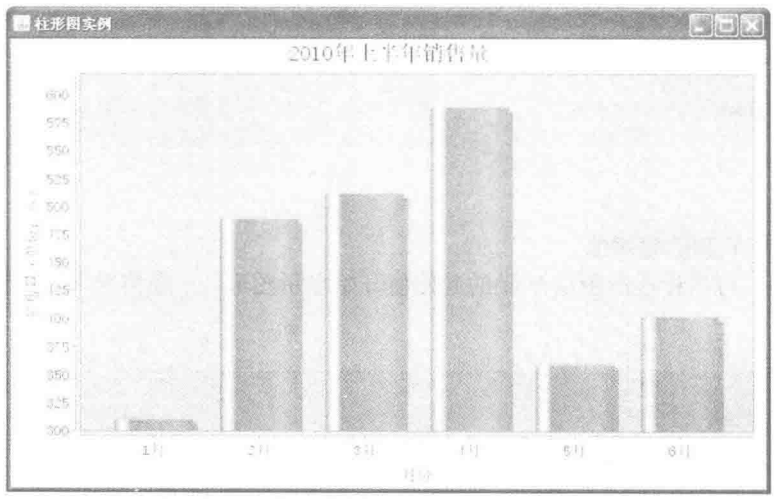


图 8.42　设置柱形图起始值

关键技术

设置柱形图的起始值使用 ValueAxis 类的 setLowerBound()方法，该方法可以指定柱形图 Y 轴的起始值。语

法如下：

```
public void setLowerBound(double min)
```

参数说明

min：表示 Y 轴的起始值。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 的对象，并且设置 Y 轴的起始值为 300，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //柱形图起始值
        valueAxis.setLowerBound(300);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo21barDemo = new BarDemo21("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 202：设置 Y 轴的起始值。

设置 Y 轴的起始值，可以让柱形图从 Y 轴的起始值开始显示图形。一般情况下，设置的起始值都不会大于最低的柱形的高度。

实例 203 Y 轴箭头

光盘位置：光盘\MR\203　　中级　　趣味指数：★★

实例说明

柱形图 Y 轴的上端和下端都可以添加箭头来表示数据延伸的方向。本实例为柱形图 Y 轴添加了上端箭头，运行效果如图 8.43 所示。

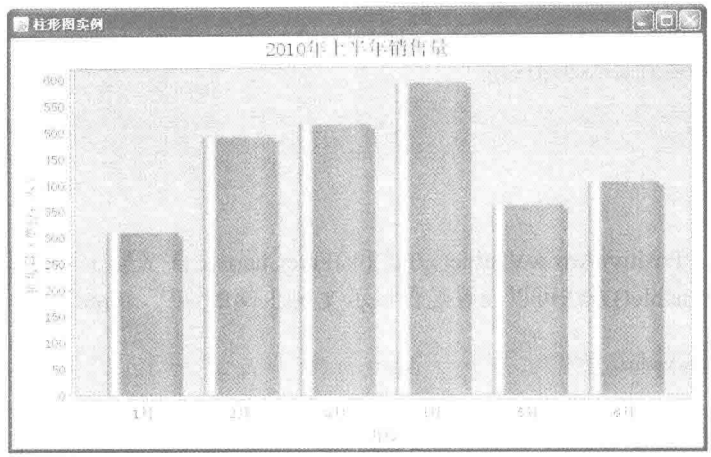


图 8.43　设置 Y 轴上端的箭头

关键技术

设置柱形图 Y 轴上端的箭头使用 ValueAxis 类的 setPositiveArrowVisible()方法，该方法通过 true 和 false 来表示是否显示 Y 轴上端的箭头，其语法如下：

```
public void setPositiveArrowVisible(boolean visible)
```

参数说明

visible：表示 Y 轴上端的箭头是否显示，默认值为 false。当 visible 为 true 时，表示显示 Y 轴上方的箭头；当 visible 为 false 时，表示不显示 Y 轴上方的箭头。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 的对象，并且显示出 Y 轴上方箭头，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //是否显示 Y 轴向上箭头
        valueAxis.setPositiveArrowVisible(true);
        //是否显示 Y 轴向下箭头
        valueAxis.setNegativeArrowVisible(false);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo23 barDemo = new BarDemo23("柱形图实例");
        barDemo.createPlot();
```

```
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 203：使用 setPositiveArrowVisible()方法和 JFreeChart 设置 Y 轴上端箭头。

使用 setPositiveArrowVisible()方法可以设置是否显示 Y 轴上端的箭头，JFreeChart 还可以设置下端的箭头是否显示，语法如下：

```
setNegativeArrowVisible(boolean visible)
```

参数说明

visible：表示 Y 轴下端的箭头是否显示，默认值为 false。当 visible 为 true 时，表示显示 Y 轴下方的箭头；当 visible 为 false 时，表示不显示 Y 轴下方的箭头。

实例 204　Y 轴主要刻度线

光盘位置：光盘\MR\204　　中级　　趣味指数：★★★

实例说明

此处的刻度线指的是主要刻度线，柱形图上 Y 轴的主要刻度线默认是显示状态。本实例通过 JFreeChart 的方法把柱形图 Y 轴的主要刻度线隐藏起来，运行效果如图 8.44 所示。

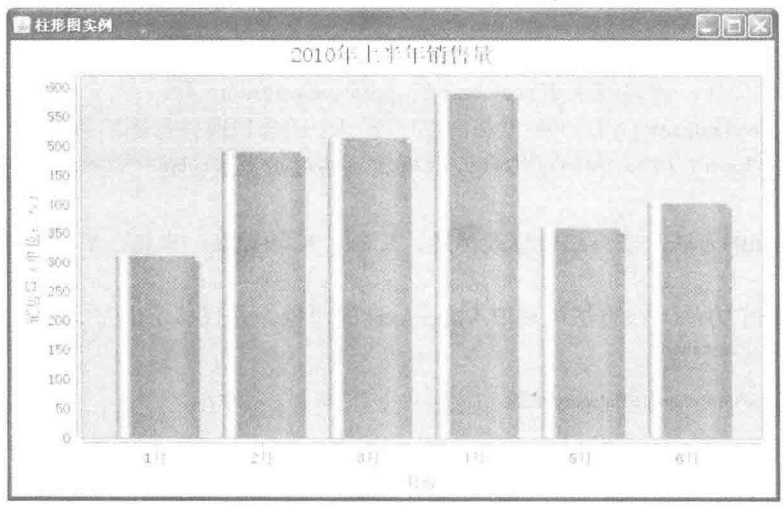


图 8.44　隐藏 Y 轴主要刻度线

关键技术

柱形图的 Y 轴刻度线可以根据需要显示或者隐藏，在 JFreeChart 中使用 Axis 类的 setTickMarksVisible()方法进行控制。语法如下：

```
public void setTickMarksVisible (boolean flag)
```

参数说明

flag：表示 Y 轴上是否显示主要刻度线，当 flag 为 true 时，表示显示刻度线；为 false 时，表示不显示刻度线。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 的对象，并且隐藏 Y 轴的主要刻度线，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //刻度线是否显示
        valueAxis.setTickMarksVisible(false);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo25 barDemo = new BarDemo25("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 204：表示 Y 轴的主要刻度线分布的白色虚线。

Y 轴的主要刻度线一般情况下都是显示的，柱形图的背景中有一些白色虚线，其分布以 Y 轴的主要刻度线为依据，即使隐藏了刻度线，这些白色虚线也会正常显示。

实例 205　Y 轴主要刻度线长度

光盘位置：光盘\MR\205　　中级　　趣味指数：★★★

实例说明

Y 轴的主要刻度线默认情况下只有外部刻度线，但实际上无论在图形内部还是图形外部，主要刻度线都可以进行长度设置。本实例显示出主要刻度线，并设置其长度，运行效果如图 8.45 所示。

关键技术

（1）设置柱形图 Y 轴内部主要刻度线长度使用 Axis 类的 setTickMarkInsideLength()方法，通过该方法可以根据主要刻度在柱形图内部显示线条。语法如下：

```
public void setTickMarkInsideLength (float length)
```

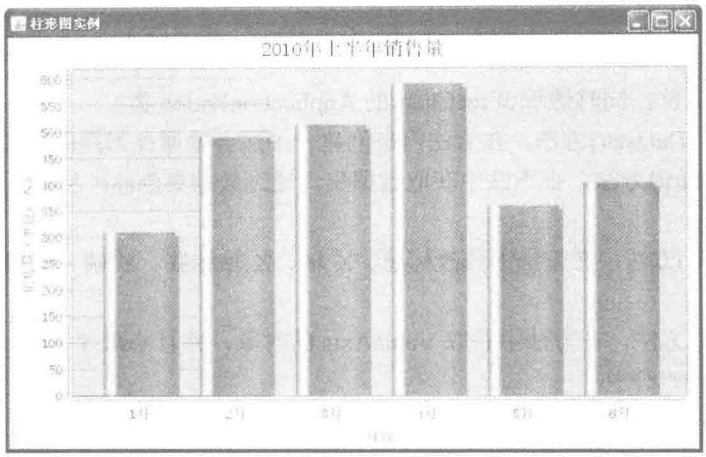


图 8.45　设置 Y 轴主要刻度线长度

参数说明

length：表示 Y 轴主要刻度线内部延长线的长度。

（2）通过 Axis 类的 setTickMarkOutsideLength()方法可以绘制 Y 轴主要刻度线外部的延长线。语法如下：

```
public void setTickMarkOutsideLength (float length)
```

参数说明

length：表示 Y 轴主要刻度线外部延长线的长度。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 的对象，然后设置 Y 轴主要刻度的内部延长线长度为 200，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //刻度线是否显示
        valueAxis.setTickMarksVisible(true);
        //内部刻度线
        valueAxis.setTickMarkInsideLength(200);
        //外部刻度线
        //valueAxis.setTickMarkOutsideLength(2);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo26 barDemo = new BarDemo26("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
```

```
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 205：设置 Y 轴主要刻度线是否显示。

Y 轴主要刻度线默认情况下是显示状态，它由两部分组成：一部分是外部主要刻度线，也就是 Y 轴左侧的线，默认情况下是显示的，其刻度线长度为 2；另外一部分是 Y 轴右侧的线，即内部主要刻度线，其默认值为 0，一般情况下不显示。

实例 206　Y 轴次要刻度线

光盘位置：光盘\MR\206　　中级　　趣味指数：★★★

实例说明

柱形图中，Y 轴的刻度线默认只显示主要刻度线。本实例通过 JFreeChart 的设置将柱形图 Y 轴的次要刻度线显示出来，运行效果如图 8.46 所示。

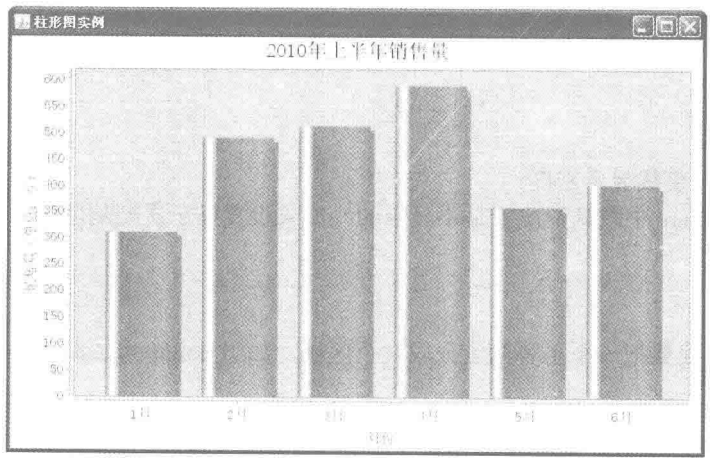


图 8.46　设置 Y 轴次要刻度线

关键技术

设置柱形图的 Y 轴上是否显示次要刻度线使用 Axis 类的 setMinorTickMarksVisible()方法，该方法通过 true 和 false 来标示是否显示次要刻度线，其语法如下：

```
public void setMinorTickMarksVisible(boolean flag)
```

参数说明

flag：表示 Y 轴上是否显示次要刻度线，当 flag 为 true 时，表示显示次要刻度线；当 flag 为 false 时，表示不显示次要刻度线。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 的对象，并且显示出 Y 轴的次要刻度线，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //是否显示次要刻度线
        valueAxis.setMinorTickMarksVisible(true);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo22 barDemo = new BarDemo22"柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 206：设置刻度线显示密度。

次要刻度线在 JFreeChart 中默认情况下是不显示的，如果设置显示次要刻度线，可以为次要刻度线设置显示密度，语法如下：

```
setMinorTickCount(int count)
```

参数说明

count：表示次要刻度线要把一个主要刻度分成几个区域，如当 count 为 2 时，会产生一条次要刻度线，把主要刻度分为两个区域。

实例 207　Y 轴次要刻度线长度

光盘位置：光盘\MR\207　　中级　　趣味指数：★★

实例说明

Y 轴的次要刻度线一般情况下只显示在 Y 轴外部，如果柱形图之间的差异很小，为了能明显地区别出各柱形图之间的差异，可以把次要刻度线在图表内部延长，运行效果如图 8.47 所示。

关键技术

（1）设置柱形图 Y 轴次要刻度线长度使用 Axis 类的 setMinorTickMarkInsideLength()方法，通过该方法可以根据次要刻度在柱形图内部显示线条。语法如下：

```
public void setMinorTickMarkInsideLength(float length)
```

参数说明

length：表示 Y 轴次要刻度线内部延长线的长度。

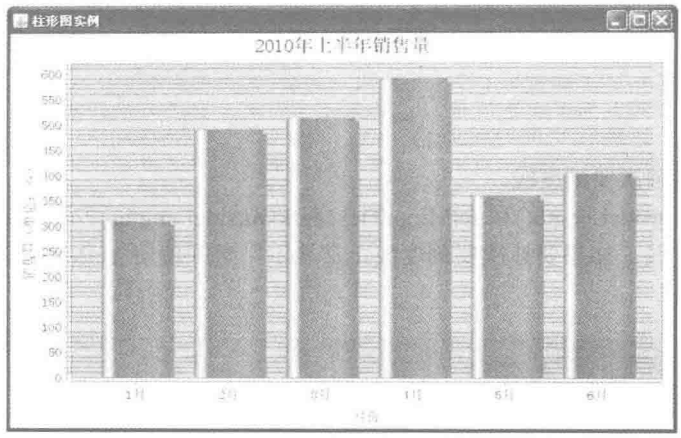


图 8.47　设置 Y 轴次要刻度线长度

（2）通过 Axis 类的 setMinorTickMarkOutsideLength()方法可以绘制 Y 轴次要刻度线外部的延长线。语法如下：

```
public void setMinorTickMarkOutsideLength (float length)
```

参数说明

length：表示 Y 轴次要刻度线外部延长线的长度。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 的对象，然后设置 Y 轴的内部延长线长度为 600，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //是否显示刻度线
        valueAxis.setMinorTickMarksVisible(true);
        //内部刻度延长线
        valueAxis.setMinorTickMarkInsideLength(600);
        //外部刻度延长线
        //valueAxis.setMinorTickMarkOutsideLength(4);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo24 barDemo = new BarDemo24("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
```

```
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 207：绘制 Y 轴次要刻度线。

使用 setMinorTickMarkInsideLength()和 setMinorTickMarkOutsideLength()方法都可以绘制 Y 轴次要刻度线，每条线在图表中都显示为实线。如果要让次要刻度线正常显示，必须使用 setMinorTickMarksVisible()方法设置次要刻度线为显示状态。

实例 208 设置 Y 轴最大值

光盘位置：光盘\MR\208　　中级　　趣味指数：★★★

实例说明

JFreeChart 的 Y 轴的最大值一般情况下是 JFreeChart 根据数据集中的数据动态设置的。本实例中屏蔽了 JFreeChart 自动设置的最大值，而进行人工设置，运行效果如图 8.48 所示。

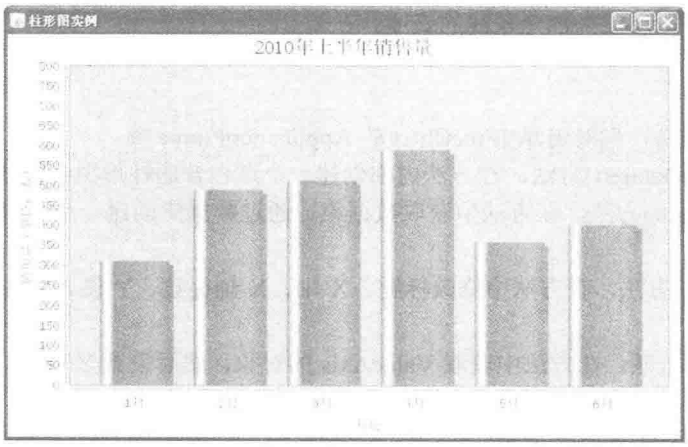


图 8.48　设置 Y 轴最大值

关键技术

如果已经确定了数据的最大值，那么可以使用 ValueAxis 类的 setUpperBound()方法对 Y 轴的最大值进行人工设置。语法如下：

```
public void setUpperBound(double max)
```

参数说明

max：表示 Y 轴的最大值。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，

代码见实例 196。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 的对象，然后设置 Y 轴的最大值为 800，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴最大值
        valueAxis.setUpperBound(800);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo27 barDemo = new BarDemo27("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 208：Y 轴最大值设置特点。

Y 轴最大值一定要超过柱形图中数据集中的最大值，否则在 JFreeChart 生成图像后，柱形图像会占满整个分类而无法分辨柱形图的高度。

实例 209　设置 Y 轴数据范围

光盘位置：光盘\MR\209　　中级　　趣味指数：★★★

实例说明

Y 轴的数据范围在默认情况下也是由 JFreeChart 设置的。本实例演示如何为 Y 轴设置数据范围，运行效果如图 8.49 所示。

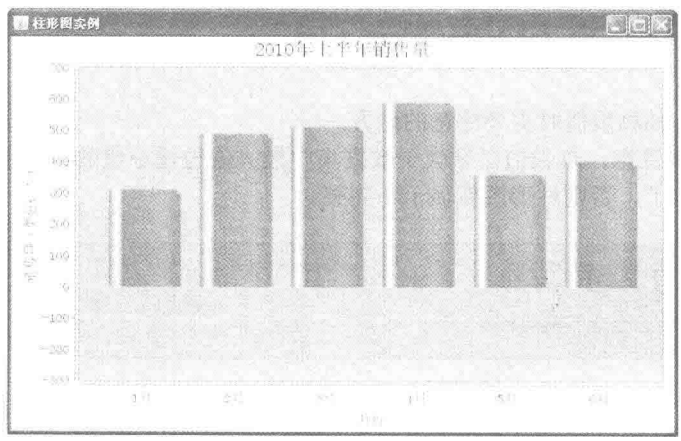


图 8.49　设置 Y 轴数据范围

关键技术

使用 ValueAxis 类的 setRangeAboutValue()方法可以设置 Y 轴的数据范围，语法如下：

```
public void setRangeAboutValue(double value, double length)
```

参数说明

❶ value：表示 Y 轴的中间值。

❷ length：表示 Y 轴的总数值。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 197。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 的对象，通过设置 Y 轴的中间值和总数值确定 Y 轴的数据范围，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //Y 轴
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴数据范围
        valueAxis.setRangeAboutValue(200,   1000);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo28 barDemo = new BarDemo28("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 209：设置 Y 轴总数值时需要注意的地方。

为 Y 轴设置总数值时要注意，总数值必须大于数据集中最大值与最小值的绝对值的和，而设置中间值时，一般取总数值的一半就可以了，否则柱形图像显示不完整。

实例 210　Y 轴的显示位置

光盘位置：光盘\MR\210　　中级　　趣味指数：★★★★

实例说明

一般的柱形图都是垂直显示，Y 轴一般都在图表的左侧，有时根据需要，可以对 Y 轴进行调整。本实例演

示把 Y 轴显示在图表右侧，运行效果如图 8.50 所示。

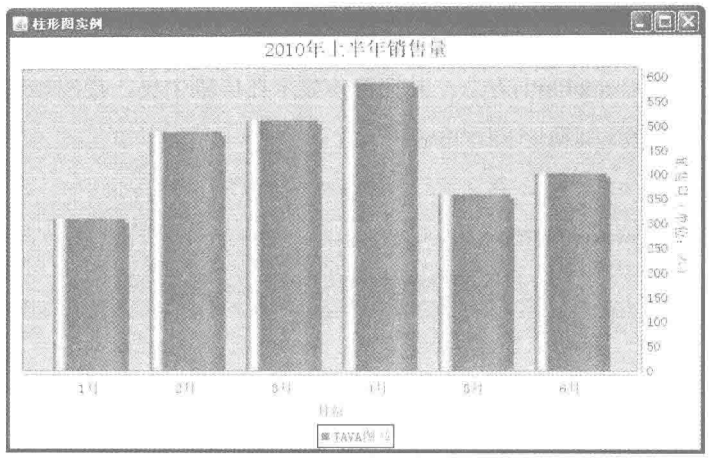


图 8.50　设置 Y 轴显示位置

关键技术

在 CategoryPlot 中使用 setRangeAxisLocation()方法可以设置 Y 轴在图表中的显示位置，setRangeAxisLocation()方法的语法如下：

```
Public void setRangeAxisLocation(AxisLocation location)
```

参数说明

location：表示 Y 轴的显示位置。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 195。

（3）创建 getJFreeChart()方法，在方法中获取数据集的 CategoryDataset 实例，通过数据集创建一柱形图的 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",          //图表标题
                        "月份",                                                    //X 轴标签
                        "销售量（单位：本）",                                      //Y 轴标签
                        dataset,                                                   //数据集
                        PlotOrientation.VERTICAL,                                  //图表方向：垂直
                        true,                                                      //是否显示图例（对于简单的柱形图必须是 false）
                        false,                                                     //是否生成工具
                        false                                                      //是否生成 URL 链接
                        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 196。

（5）创建 updatePlot()方法，在方法中设置 Y 轴显示在图表右侧，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置 Y 轴显示位置
        categoryPlot.setRangeAxisLocation(AxisLocation.TOP_OR_RIGHT);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 197。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo39 barDemo = new BarDemo39("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 210：使用 AxisLocation 类设置图表 Y 轴显示位置。

使用 AxisLocation 类中的常量 AxisLocation.TOP_OR_RIGHT 可以为图表设置 Y 轴显示位置，可以在图表的上方，也可以在图表右侧。在本实例第（3）步中使用 PlotOrientation.VERTICAL 设置图表垂直显示，Y 轴则显示在图表右侧；如果使用 PlotOrientation.HORIZONTAL 设置图表水平显示，Y 轴则显示在图表上方。

8.7 高级柱形图

实例 211	设置网格竖线	中级
	光盘位置：光盘\MR\211	趣味指数：★★★

实例说明

柱形图图像区默认情况下只显示网格横线，网格竖线一般是不显示的。本实例演示如何显示网格竖线，运行效果如图 8.51 所示。

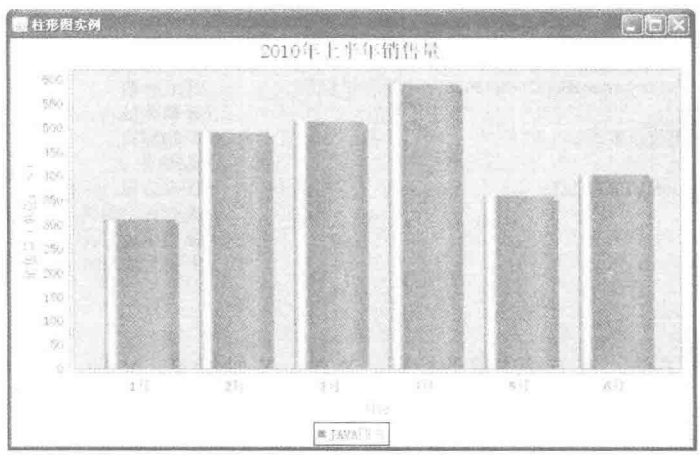


图 8.51 设置网格竖线

关键技术

使用 CategoryPlot 类的 setDomainGridlinesVisible()方法可以设置是否显示网格竖线，语法如下：

public void setDomainGridlinesVisible(boolean visible)

参数说明

visible：表示是否显示柱形图竖线，当 visible 为 true 时，则显示；当 visible 为 false 时，则不显示。柱形图中默认情况下 visible 为 false。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码如下：

```
private CategoryDataset getCategoryDataset() {
    DefaultKeyedValues keyedValues = new DefaultKeyedValues();
    //添加数据
    keyedValues.addValue("1 月", 310);
    keyedValues.addValue("2 月", 489);
    keyedValues.addValue("3 月", 512);
    keyedValues.addValue("4 月", 589);
    keyedValues.addValue("5 月", 359);
    keyedValues.addValue("6 月", 402);
    //创建数据集合实例
    CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
    return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset();
    JFreeChart chart = ChartFactory.createBarChart("2010.1-6 sales volume",      //图表标题
            "月份",                                                              //X 轴标签
            "销售量（单位：本）",                                                //Y 轴标签
            dataset,                                                             //数据集
            PlotOrientation.VERTICAL,                                            //图表方向：垂直
            true,                                                                //是否显示图例（对于简单的柱形图必须是 false）
            false,                                                               //是否生成工具
            false                                                                //是否生成 URL 链接
            );
    return chart;
}
```

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码如下：

```
public void updateFont(JFreeChart chart) {
    //标题
    TextTitle textTitle = chart.getTitle();
    textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
    //图表（柱形图）
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
    //X 轴字体
    categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
    //X 轴标签字体
    categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    ValueAxis valueAxis = categoryPlot.getRangeAxis();
    //Y 轴字体
    valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
    //Y 轴标签字体
    valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，设置图像区显示网格竖线，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    //设置网格竖线
```

```
        categoryPlot.setDomainGridlinesVisible(true);
    }
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码如下：

```
    public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //修改图表
        updatePlot(chart);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
    }
```

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        BarDemo29 barDemo = new BarDemo29("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 211：柱形图中网格竖线可看作 X 轴的延长线。

在柱形图的图像区，默认情况下只显示网格横向虚线而不显示竖向虚线，除非进行手工设置。网络竖线在显示时与 X 轴的刻度一致，可以看作是 X 轴刻度的延长线。

实例 212　设置网格竖线颜色

光盘位置：光盘\MR\212　　中级　　趣味指数：★★★

实例说明

柱形图网格竖线默认情况下与横线颜色相同，都为白色。本实例将网格竖线的颜色设置为蓝色，运行效果如图 8.52 所示。

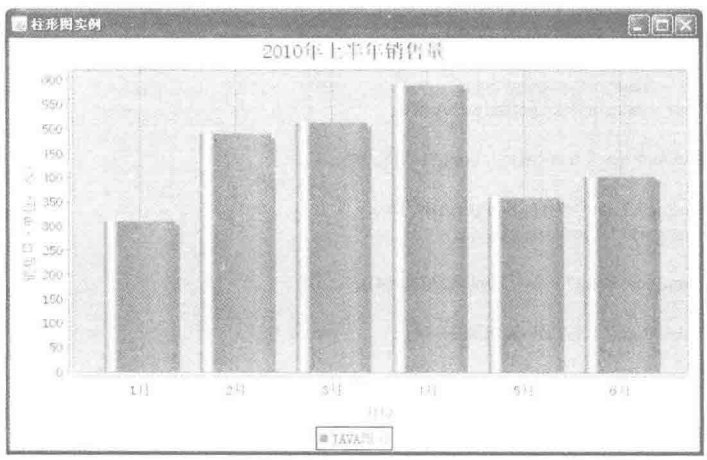


图 8.52　设置网格竖线颜色

关键技术

使用 CategoryPlot 类的 setDomainGridlinePaint()方法可以设置网格竖线的颜色，语法如下：

```
public void setDomainGridlinePaint(Paint paint)
```

参数说明

paint：表示柱形图竖线的颜色。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，并且设置网格竖线为蓝色，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置网格竖线
        categoryPlot.setDomainGridlinesVisible(true);
        categoryPlot.setDomainGridlinePaint(Color.blue);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo30 barDemo = new BarDemo30("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 212：图像区网格中竖线的设置。

图像区网格的竖线默认情况下是不显示的，如果希望修改竖线颜色，必须先将其设置为可见状态，同时还可以修改竖线的笔触，语法如下：

```
setDomainGridlineStroke(Stroke stroke)
```

参数说明

stroke：表示柱形图竖线的笔触。

实例 213　设置柱形图文本注解

光盘位置：光盘\MR\213　　中级　　趣味指数：★★★★

实例说明

在显示柱形图时，可以为图形添加文本注解。本实例将柱形图数据通过文本注解的方式显示在图形中，运

行效果如图 8.53 所示。

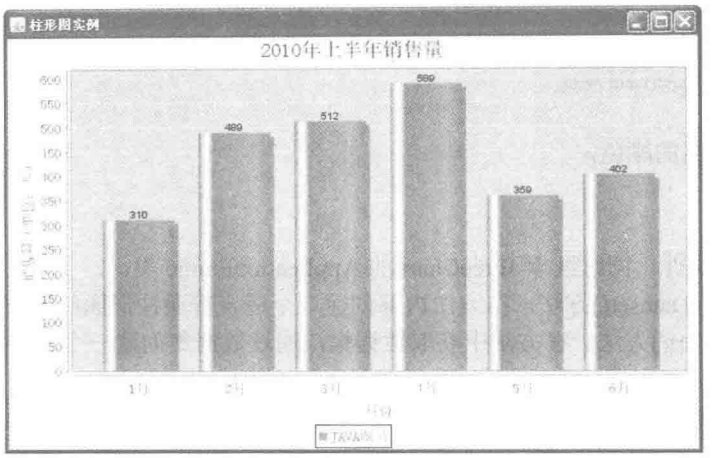


图 8.53　设置柱形图文本注解

关键技术

使用 CategoryPlot 类的 addAnnotation()方法可以为分类图形添加文本注解，语法如下：

```
public void addAnnotation(CategoryAnnotation annotation)
```

参数说明

annotation：表示柱形图的文本注解类。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，并且为每个分类添加文本注解，注解的内容为分类的数值，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置注解
        CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",320);
        CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",499);
        CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",522);
        CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",599);
        CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",369);
        CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",412);
        //添加注解
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    BarDemo31 barDemo = new BarDemo31("柱形图实例");
    barDemo.createPlot();
    barDemo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(barDemo);
    //设置可以显示
    barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 213：CategoryTextAnnotation 类的实现方法。

CategoryTextAnnotation 为柱形图的文本注解类，通过其构造方法创建 CategoryTextAnnotation 实例，语法如下：

```
CategoryTextAnnotation(String text, Comparable category, double value)
```

参数说明

❶ text：表示柱形图文本注解的内容。

❷ category：表示要添加文本注解的柱形图类别。

❸ value：表示要添加文本注解的位置。

实例 214　设置柱形图文本注解字体

光盘位置：光盘\MR\214　　中级　　趣味指数：★★★

实例说明

在为柱形图添加文本注解时，如果默认的字体不符合用户的要求，可根据实际情况对字体进行调整。本实例将注解字体设置为“宋体”，运行效果如图 8.54 所示。

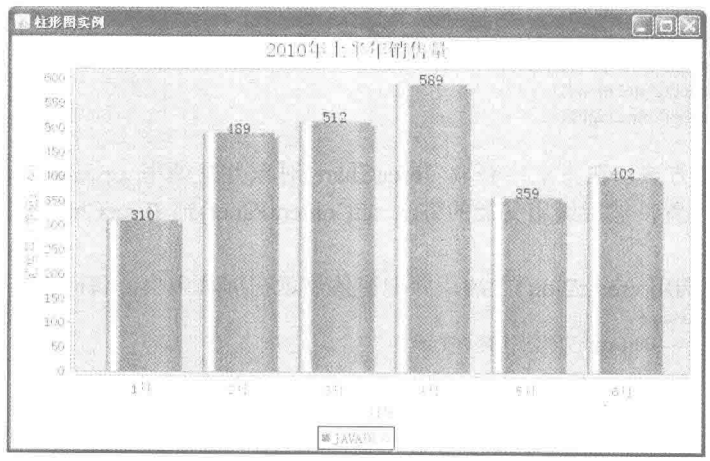


图 8.54　设置柱形图文本注解字体

关键技术

使用 TextAnnotation 类的 setFont()方法可以为分类图形设置文本注解字体，语法如下：

```
public void setFont(Font font)
```

参数说明

font：表示柱形图的文本注解字体。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，并且为每个柱形添加文本注解，注解的内容为分类的数值，然后设置注解字体为“宋体”，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置注解
        CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",320);
        CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",499);
        CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",522);
        CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",599);
        CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",369);
        CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",412);
        //设置注解字体
        annotation.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation1.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation2.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation3.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation4.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation5.setFont(new Font("宋体", Font.PLAIN, 15));
        //添加注解
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo32 barDemo = new BarDemo32("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 214：柱形图文本注解与图形位置的设置。

为柱形图文本注解设置字体时，修改字体大小可能会影响到字体与图形之间的距离，所以有时要根据字体调整文本注解在图形中的位置。

实例 215	设置柱形图文本注解颜色 光盘位置：光盘\MR\215	中级 趣味指数：★★★

实例说明

柱形图的文本注解颜色默认为黑色。本实例演示如何设置柱形图的文本注解颜色，运行效果如图 8.55 所示。

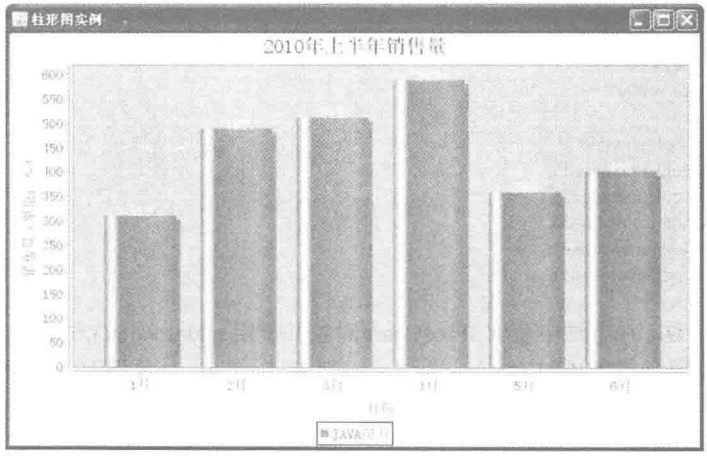


图 8.55　设置柱形图文本注解颜色

关键技术

使用 TextAnnotation 类的 setPaint()方法可以为分类图形设置文本注解颜色，语法如下：

```
public void setPaint(Paint paint)
```

参数说明

paint：表示柱形图的文本注解颜色。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，并且为每个柱形添加文本注解，注解的内容为分类的数值，然后设置注解字体为“宋体”、注解文字的颜色为白色，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //图表
```

```
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置注解
        CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",320);
        CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",499);
        CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",522);
        CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",599);
        CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",369);
        CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",412);
        //设置注解字体
        annotation.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation1.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation2.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation3.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation4.setFont(new Font("宋体", Font.PLAIN, 15));
        annotation5.setFont(new Font("宋体", Font.PLAIN, 15));
        //设置注解颜色
        annotation.setPaint(Color.WHITE);
        annotation1.setPaint(Color.WHITE);
        annotation2.setPaint(Color.WHITE);
        annotation3.setPaint(Color.WHITE);
        annotation4.setPaint(Color.WHITE);
        annotation5.setPaint(Color.WHITE);
        //添加注解
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
    }
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        BarDemo33 barDemo = new BarDemo33("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 215：柱形图文本注解的颜色设置。

柱形图文本注解一般显示在柱形上面或者图表的背景上，所以在设置文本注解颜色时一定要注意不能使用柱形的颜色或者图表的背景色，否则不易显示文本注解的内容。

实例 216　设置柱形图文本注解锚点

光盘位置：光盘\MR\216　　高级　　趣味指数：★★★

实例说明

柱形图的文本注解在实例化时可以设置垂直高度，还可以在实例化以后设置锚点，运行效果如图 8.56 所示。

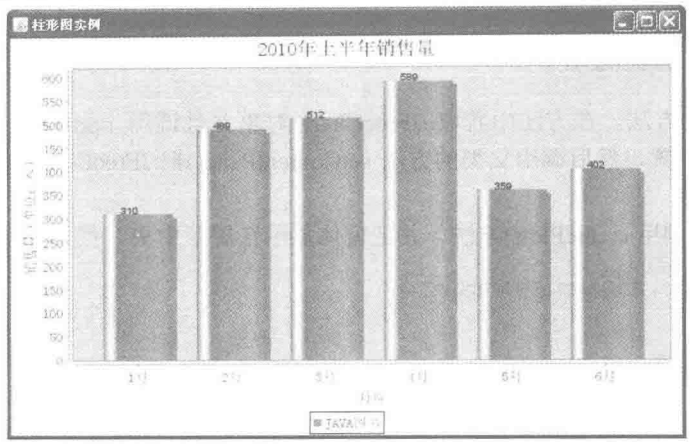


图 8.56　设置柱形图文本注解锚点

关键技术

使用 TextAnnotation 类的 setTextAnchor()方法可以为分类图形设置注解的文本锚点、调整文本的对齐方式，语法如下：

```
public void setTextAnchor(TextAnchor anchor)
```

参数说明

anchor：表示柱形图注解的文本锚点。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，并且为每个柱形注解设置文本锚点，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    //设置注解
    CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",310);
    CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",489);
    CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",512);
    CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",589);
    CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",359);
    CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",402);
    //设置文本注解锚点
    annotation.setTextAnchor(TextAnchor.BASELINE_RIGHT);
    annotation1.setTextAnchor(TextAnchor.BASELINE_RIGHT);
    annotation2.setTextAnchor(TextAnchor.BASELINE_RIGHT);
    annotation3.setTextAnchor(TextAnchor.BASELINE_RIGHT);
    annotation4.setTextAnchor(TextAnchor.BASELINE_RIGHT);
    annotation5.setTextAnchor(TextAnchor.BASELINE_RIGHT);
    //添加注解
    categoryPlot.addAnnotation(annotation);
    categoryPlot.addAnnotation(annotation1);
    categoryPlot.addAnnotation(annotation2);
```

```
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
    }
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo34 barDemo = new BarDemo34("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 216：TextAnchor 中锚点位置的常量。

JFreeChart 在 TextAnchor 类实现了很多注解文本锚点位置的常量，如 TextAnchor.CENTER、TextAnchor.BASELINE_CENTER、TextAnchor.BASELINE_LEFT、TextAnchor.BASELINE_RIGHT、TextAnchor.BOTTOM_CENTER 等。

实例 217　设置柱形图文本注解类别锚点

高级

光盘位置：光盘\MR\217

趣味指数：★★★

实例说明

柱形图的文本注解可以在实例化时设置垂直位置，还可以在实例化后设置类别锚点，运行效果如图 8.57 所示。

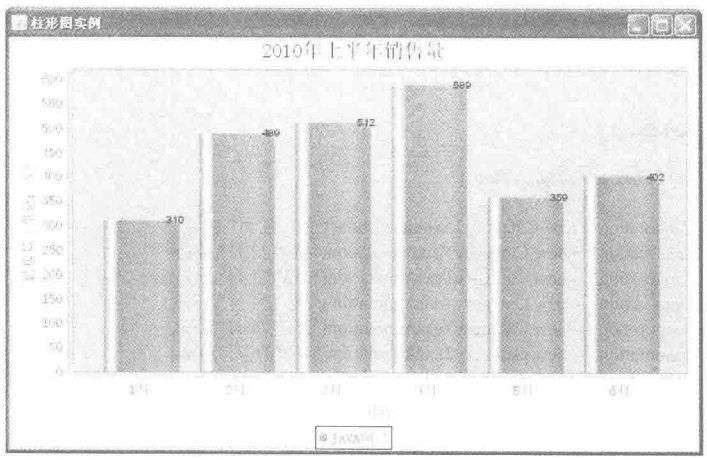


图 8.57　设置柱形图文本注解类别锚点

关键技术

TextAnnotation 类一般用于文本的注解，CategoryTextAnnotation 类一般用于类别注解，使用 CategoryTextAnnotation

的 setCategoryAnchor()方法可以设置类别锚点，语法如下：

```
public void setCategoryAnchor(CategoryAnchor anchor)
```

参数说明

anchor：表示柱形图文本注解的类别锚点。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，并且为每个柱形图文本注解设置类别锚点，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置注解
        CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",310);
        CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",489);
        CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",512);
        CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",589);
        CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",359);
        CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",402);
        //设置注解分类锚点
        annotation.setCategoryAnchor(CategoryAnchor.END);
        annotation1.setCategoryAnchor(CategoryAnchor.END);
        annotation2.setCategoryAnchor(CategoryAnchor.END);
        annotation3.setCategoryAnchor(CategoryAnchor.END);
        annotation4.setCategoryAnchor(CategoryAnchor.END);
        annotation5.setCategoryAnchor(CategoryAnchor.END);
        //添加注解
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo35 barDemo = new BarDemo35("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 217：用于表示柱形图文本注解分类锚点的常量。

JFreeChart 在 CategoryAnchor 类中定义了 CategoryAnchor.START、CategoryAnchor.MIDDLE 和 CategoryAnchor.

END 3 个常量，用于表示柱形图文本注解的分类锚点。

实例 218　设置柱形图文本注解旋转锚点

高级

光盘位置：光盘\MR\218

趣味指数：★★

实例说明

柱形图的文本注解可以在实例化时设置垂直位置，还可以在实例化后设置旋转锚点，运行效果如图 8.58 所示。

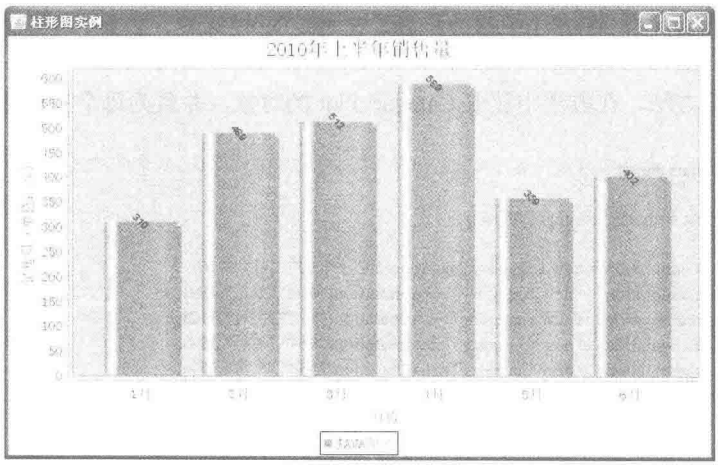


图 8.58　设置柱形图文本注解旋转锚点

关键技术

使用 TextAnnotation 的 setRotationAngle()方法可以设置文本注解的旋转锚点，语法如下：

```
public void setRotationAngle(double angle)
```

参数说明

angle：表示柱形图文本注解的旋转锚点。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，并为每个柱形文本注解设置旋转锚点，参数值为 Math.PI*0.2，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    //设置注解
    CategoryTextAnnotation annotation = new CategoryTextAnnotation("310","1 月",310);
    CategoryTextAnnotation annotation1 = new CategoryTextAnnotation("489","2 月",489);
```

```
        CategoryTextAnnotation annotation2 = new CategoryTextAnnotation("512","3 月",512);
        CategoryTextAnnotation annotation3 = new CategoryTextAnnotation("589","4 月",589);
        CategoryTextAnnotation annotation4 = new CategoryTextAnnotation("359","5 月",359);
        CategoryTextAnnotation annotation5 = new CategoryTextAnnotation("402","6 月",402);
        //设置注解角度锚点
        annotation.setRotationAngle(Math.PI*0.2);
        annotation1.setRotationAngle(Math.PI*0.2);
        annotation2.setRotationAngle(Math.PI*0.2);
        annotation3.setRotationAngle(Math.PI*0.2);
        annotation4.setRotationAngle(Math.PI*0.2);
        annotation5.setRotationAngle(Math.PI*0.2);
        //添加注解
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
        categoryPlot.addAnnotation(annotation5);
    }
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo36 barDemo = new BarDemo36("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 218：用于设置柱形图文本注解的旋转锚点的设置。

JFreeChart 柱形图文本注解使用数值设置的旋转锚点，通过数值的大小来表示旋转的角度，其中，锚点的数值指的是弧度数，PI 弧度角等于 180°，所以本实例使用 PI*0.2，即设置文本注解的角度为 36°。

实例 219　设置柱形图线条注解

光盘位置：光盘\MR\219　　高级　　趣味指数：★★★★

实例说明

柱形图的线条注解是一条直线，通过线条注解可以把任意两个柱形连接在一起，从而体现出折线图的效果，运行效果如图 8.59 所示。

关键技术

使用 TextAnnotation 的 addAnnotation()方法可以设置线条注解，语法如下：

```
public void addAnnotation(CategoryAnnotation annotation)
```

参数说明

annotation：表示要设置的柱形图的线条注解。

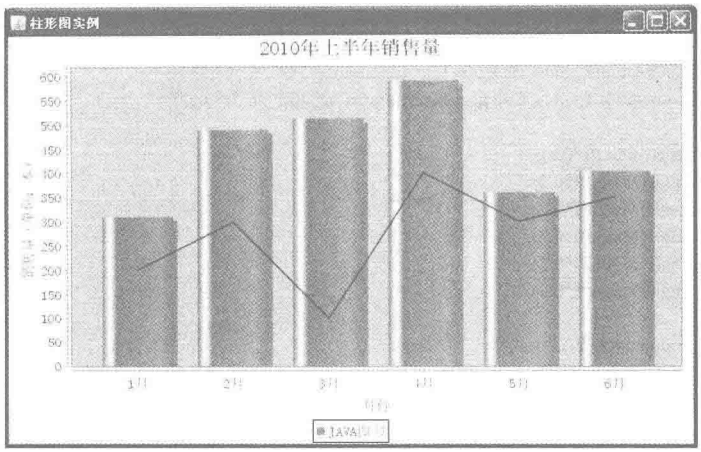


图 8.59　设置柱形图线条注解

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，并且为柱形图设置线条注解，绘制一个折线图，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置线条注解
        CategoryLineAnnotation annotation = new CategoryLineAnnotation("1 月",200,"2 月",300,Color.blue,new BasicStroke());
        CategoryLineAnnotation annotation1 = new CategoryLineAnnotation("2 月",300,"3 月",100,Color.blue,new BasicStroke());
        CategoryLineAnnotation annotation2 = new CategoryLineAnnotation("3 月",100,"4 月",400,Color.blue,new BasicStroke());
        CategoryLineAnnotation annotation3 = new CategoryLineAnnotation("4 月",400,"5 月",300,Color.blue,new BasicStroke());
        CategoryLineAnnotation annotation4 = new CategoryLineAnnotation("5 月",300,"6 月",350,Color.blue,new BasicStroke());
        categoryPlot.addAnnotation(annotation);
        categoryPlot.addAnnotation(annotation1);
        categoryPlot.addAnnotation(annotation2);
        categoryPlot.addAnnotation(annotation3);
        categoryPlot.addAnnotation(annotation4);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo37 barDemo = new BarDemo37("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 219：绘制线条注解的方法。

绘制线条注解要使用 CategoryLineAnnotation 类创建一个实例，通过该类的构造方法创建实例非常简单，语法如下：

CategoryLineAnnotation(Comparable category1, double value1,Comparable category2, double value2, Paint paint, Stroke stroke)

参数说明

❶ category1：表示前一个柱形图的分类名称。
❷ value1：表示前一个柱形图的线条注解的垂直高度。
❸ category2：表示后一个柱形图的分类名称。
❹ value2：表示后一个柱形图的线条注解的垂直高度。
❺ paint：表示线条注解的颜色。
❻ stroke：表示线条注解的笔触。

实例 220　绘制柱形效果

光盘位置：光盘\MR\220　　高级　　趣味指数：★★★

实例说明

JFreeChart 在生成柱形图时，可以根据需要显示柱形的效果。本实例使用 JFreeChart 提供的类，使用普通的平面效果显示柱形图，运行效果如图 8.60 所示。

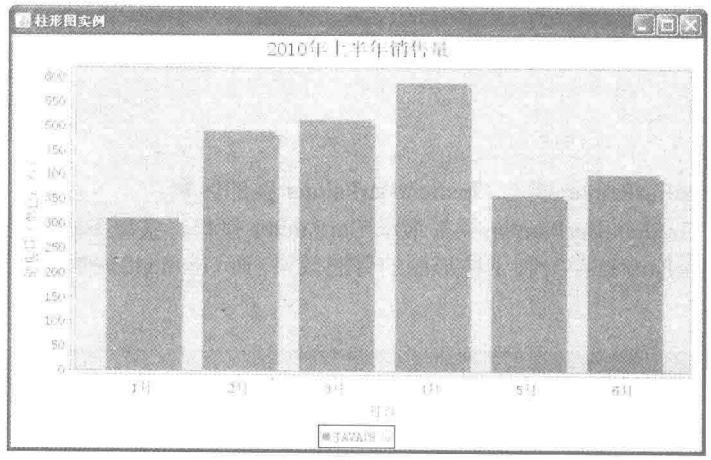


图 8.60　标准的柱形效果

关键技术

使用 BarRenderer 的 setBarPainter()方法可以设置柱形效果，语法如下：

public void setBarPainter(BarPainter painter)

参数说明

painter：表示要绘制的柱形效果。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，并绘制一个普通的柱形图效果图，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置 Y 轴显示位置
        BarRenderer renderer = (BarRenderer) categoryPlot.getRenderer();
        //普通效果
        StandardBarPainter barPainter = new StandardBarPainter();
        //梯形效果
        //GradientBarPainter barPainter = new GradientBarPainter();
        renderer.setBarPainter(barPainter);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo41 barDemo = new BarDemo41("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 220：StandardBarPainter 类与 GradientBarPainter 类的区别。

StandardBarPainter 与 GradientBarPainter 类都继承 BarPainter 接口，实现 BarPainter 中的抽象的方法，二者的不同之处在于 StandardBarPainter 类绘制了柱形图的普通效果，而 GradientBarPainter 类绘制了柱形图呈梯形的立体效果。

实例 221 柱形图阴影

光盘位置：光盘\MR\221　　中级　　趣味指数：★★

实例说明

为柱形图设置阴影效果可以使图表更生动，用户可以根据需要设置是否使用阴影效果。本实例演示如何取消阴影效果，运行效果如图 8.61 所示。

关键技术

使用 BarRenderer 的 setShadowVisible()方法可以设置柱形的阴影效果，语法如下：

```
public void setShadowVisible(boolean visible)
```

参数说明

visible：表示是否显示阴影效果。

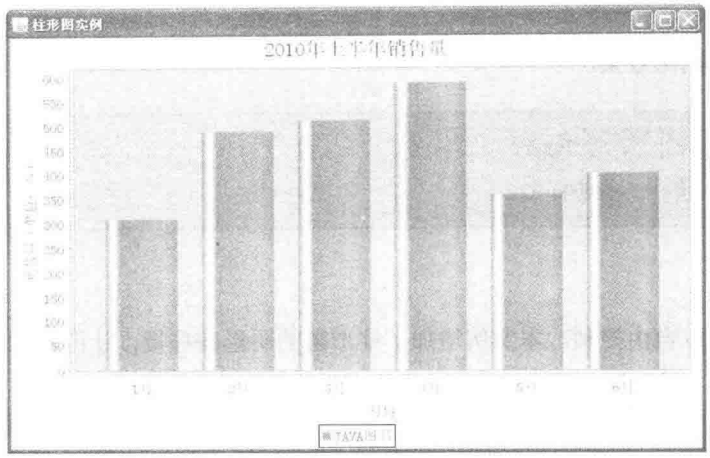


图 8.61　取消阴影效果

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，通过 CategoryPlot 的对象获取 BarRenderer 对象，然后设置隐藏阴影效果，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        BarRenderer renderer = (BarRenderer) categoryPlot.getRenderer();
        //阴影效果
        renderer.setShadowVisible(false);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo42 barDemo = new BarDemo42("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 221：柱形图阴影效果的显示。

默认情况下柱形图的阴影效果都是显示的，且显示颜色为灰色，可以使用 BarRenderer 类的 setShadowPaint()方法修改阴影的显示效果，语法如下：

```
setShadowPaint(Paint paint)
```

参数说明

paint：表示柱形图的阴影效果。

实例 222 柱形图阴影偏移

光盘位置：光盘\MR\222

中级

趣味指数：★★★

实例说明

柱形图的阴影效果可以自由控制。本实例增加了柱形图的阴影偏移量，让阴影的效果更明显，运行效果如图 8.62 所示。

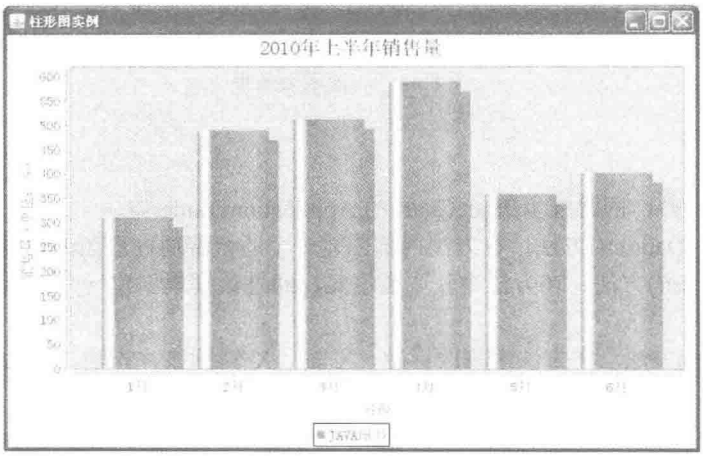


图 8.62 阴影偏移

关键技术

使用 BarRenderer 的 setShadowXOffset()和 setShadowYOffset()方法可以共同设置阴影偏移效果，语法如下：

```
public void setShadowXOffset(double offset)
```

参数说明

offset：表示柱形 X 轴方向阴影的偏移量。

```
public void setShadowYOffset(double offset)
```

参数说明

offset：表示柱形 Y 轴方向阴影的偏移量。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，通过 CategoryPlot 的对象获取 BarRenderer 对象，然后设置阴影的偏移效果，代码如下：

```
private void updatePlot(JFreeChart chart) {
```

```
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        BarRenderer renderer = (BarRenderer) categoryPlot.getRenderer();
        //阴影效果
        renderer.setShadowVisible(true);
        //X 轴偏移量
        renderer.setShadowXOffset(10);
        //Y 轴偏移量
        renderer.setShadowYOffset(10);
    }
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo43 barDemo = new BarDemo43("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
    }
```

秘笈心法

心法领悟 222：阴影的偏移值与阴影范围成正比。

阴影的偏移是以柱形图中柱形的位置为基础的，X 轴、Y 轴的偏移值默认情况下都为 4.0，偏移的值越大，阴影的范围越大；偏移的值越小，阴影的范围越小。

实例 223　设置柱形的颜色

光盘位置：光盘\MR\223　　中级　　趣味指数：★★★★

实例说明

JFreeChart 柱形图中柱形的颜色是可以设置的。本实例演示如何设置柱形图的柱形颜色，运行效果如图 8.63 所示。

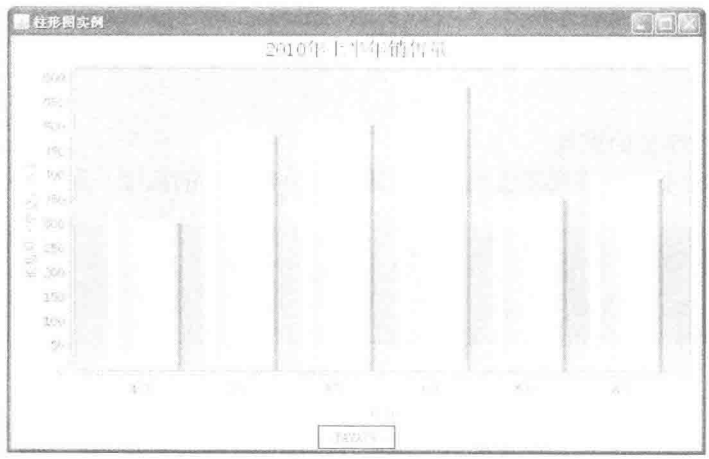


图 8.63　设置柱形图的柱形颜色

关键技术

使用 AbstractRenderer 的 setSeriesPaint()方法可以设置柱形的颜色，语法如下：

```
public void setSeriesPaint(int series, Paint paint)
```

参数说明

❶ series：表示要设置的柱形颜色的系列。

❷ paint：表示要设置的柱形的颜色。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建一个柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中修改标题、X 轴、X 轴标签、Y 轴、Y 轴标签的字体为“宋体”，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 的对象，通过 CategoryPlot 的对象获取 BarRenderer 对象，然后为柱形设置颜色，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        BarRenderer renderer = (BarRenderer) categoryPlot.getRenderer();
        //柱形图颜色
        renderer.setSeriesPaint(0, Color.orange);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，再调用 updatePlot()方法修改图表设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo44 barDemo = new BarDemo44("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 223：柱形图颜色的设置。

为柱形设置的颜色尽量不要与背景颜色相近，否则不易分辨柱形的高度，影响用户查看图表效果。

实例 224 绘制 3D 柱形图

光盘位置：光盘\MR\224　　高级　　趣味指数：★★★★

实例说明

不管如何绘制，普通柱形图的显示效果始终不够立体。本实例演示如何绘制 3D 效果的柱形图，运行效果如图 8.64 所示。

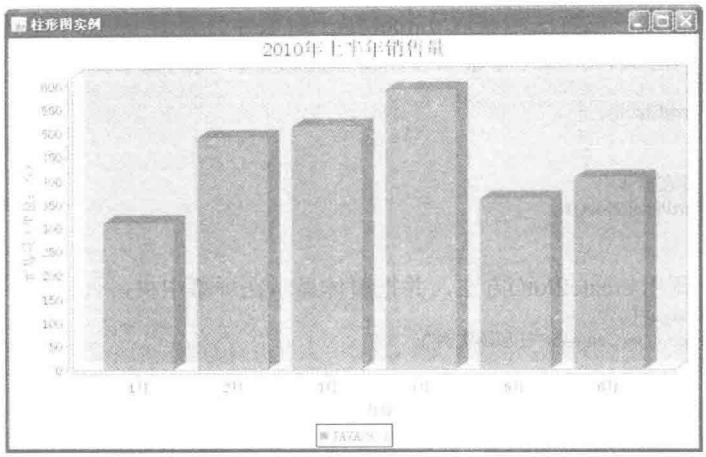


图 8.64　绘制 3D 柱形图

关键技术

在 JFreeChart 中使用 ChartFactory 的 createBarChart3D()方法可以创建 3D 柱形图，创建完成后会返回一个 JFreeChart 对象，语法如下：

```
public static JFreeChart createBarChart3D(String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：3D 柱形图的标题。

❷ categoryAxisLabel：3D 柱形图类别标签，即 X 轴的名称。

❸ valueAxisLabel：3D 柱形图数据标签，即 Y 轴的名称。

❹ dataset：柱形图的数据集合。

❺ orientation：3D 柱形图的图表方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个分类数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建 3D 柱形图的 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset();
    JFreeChart chart = ChartFactory.createBarChart3D("2010 年上半年销售量",      //图表标题
                "月份",                                                        //X 轴标签
                "销售量（单位：本）",                                          //Y 轴标签
                dataset,                                                       //数据集
                PlotOrientation.VERTICAL,                                      //图表方向：垂直
                true,                                                          //是否显示图例（对于简单的柱形图必须是 false）
                false,                                                         //是否生成工具
                false                                                          //是否生成 URL 链接
                );
    return chart;
}
```

（4）创建 updateFont()方法，在方法中重新设置图表标题、标签等字体，代码见实例 211。

（5）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));

}
```

（6）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo45 barDemo = new BarDemo45("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 224：3D 柱形图与普通柱形图的区别。

3D 柱形图与普通柱形图的一般属性都是相同的，如设置图表字体、修改图表颜色等。但有部分属性不能对两种柱形图都起作用，如普通柱形图可以设置阴影效果，而 3D 柱形图则没有阴影效果。

实例 225 标记柱形图区间

光盘位置：光盘\MR\225

高级

趣味指数：★★★★

实例说明

有时需要在柱形图的显示范围内划出一部分区域来标示一些特殊内容。本实例使用特殊颜色来显示划分出来的区域，并标示“超出历史最高销售”，运行效果如图 8.65 所示。

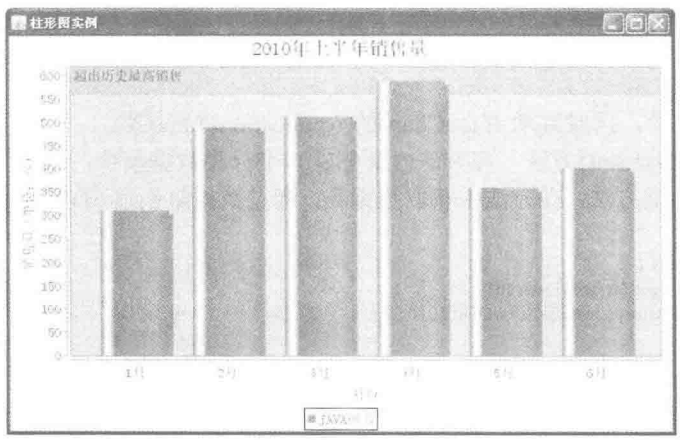


图 8.65 显示区域标记

关键技术

（1）使用 CategoryPlot 类的 addRangeMarker()方法可以创建一个范围标记区域，语法如下：

public void addRangeMarker(Marker marker)

参数说明

marker：表示范围标记区域的实现。

（2）IntervalMarker 类继承了抽像类 Marker，使用 IntervalMarker 的构造方法可以进行实例化，语法如下：

public IntervalMarker(double start, double end)

参数说明

❶ start：表示标记区域的开始值。

❷ end：表示标记区域的结束值。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个分类数据集合，代码见实例 211。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中重新设置图表标题、标签等字体，代码见实例 211。

（5）创建 updatePlot()方法，在方法中创建一个范围区域，并设置范围区域的各种属性，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        IntervalMarker target = new IntervalMarker(560.0, 700.0);
        //Marker 标签名称
        target.setLabel("超出历史最高销售");
        //Marker 标签字体
        target.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Marker 标签锚点
        target.setLabelAnchor(RectangleAnchor.LEFT);
        //Marker 标签文字锚点
        target.setLabelTextAnchor(TextAnchor.BASELINE_LEFT);
        //Marker 背景色
        target.setPaint(new Color(222, 122, 255, 128));
        //标记范围
        categoryPlot.addRangeMarker(target);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，调用 updatePlot()方法修改图表的设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    BarDemo46 barDemo = new BarDemo46("柱形图实例");
    barDemo.createPlot();
    barDemo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(barDemo);
    //设置可以显示
    barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 225：多图表在层上的支持。

创建范围区域时，JFreeChart 支持图表创建多个范围区域，CategoryPlot 类的 addRangeMarker()方法还支持层的使用，语法如下：

addRangeMarker(Marker marker, Layer layer)

参数说明

❶ marker：表示范围标记区域的实现。

❷ layer：表示层的使用情况。Layer 的类中有两个常量，Layer.BACKGROUND 表示范围区域显示在柱形下面一层；Layer.FOREGROUND 表示范围区域显示在柱形上面一层。

实例 226 多系列柱形图

光盘位置：光盘\MR\226　　高级　　趣味指数：★★★★★

实例说明

单系列的柱形图只能比较单一分类之间的不同，而多系列的柱形图可以比较多个分类中的不同以及各个分类之间的不同，运行效果如图 8.66 所示。

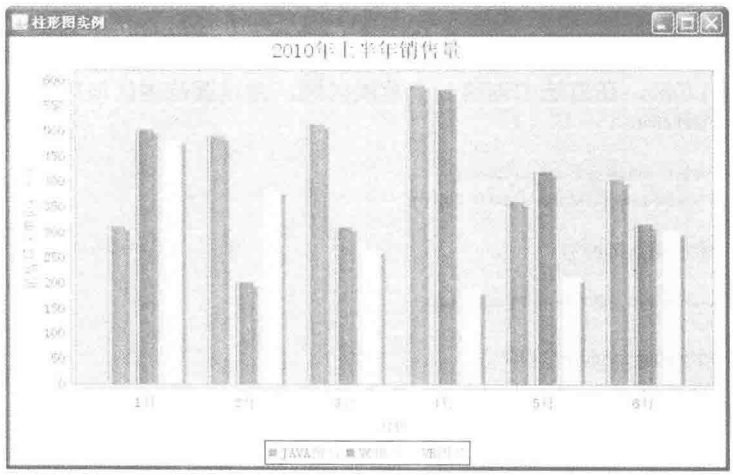


图 8.66　多系列柱形图

关键技术

这里把图表中不同的分类叫做系列，把 X 轴上的刻度叫做分类。要绘制多系列的柱形图，首先数据集要能支持多系列的数据，使用 DefaultCategoryDataset 类的 addValue()方法可以添加多系列的数据集，语法如下：

```
public void addValue(double value, Comparable rowKey, Comparable columnKey)
```

参数说明

❶ value：表示柱形图某一系列中某一分类的具体数据。

❷ rowKey：表示柱形图中的系列名称。

❸ columnKey：表示柱形图中的分类名称，即 X 轴上的刻度。

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建多系列的分类数据集合，代码如下：

```
private CategoryDataset getCategoryDataset() {

    //行关键字
    final String series1 = "JAVA 图书";
    final String series2 = "VC 图书";
    final String series3 = "VB 图书";
```

```
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);

        return dataset;

    }
```

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 211。

（4）创建 updateFont()方法，在方法中重新设置图表标题、标签等字体，代码见实例 211。

（5）创建 createPlot()方法，在方法中获取 JFreeChart 对象并调用 updateFont()方法修改字体，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 224。

（6）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo47 barDemo = new BarDemo47("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 226：使用 createCategoryDataset()方法创建多系列的柱形图数据。

创建 JFreeChart 的数据集的方法有很多，使用 DatasetUtilities 的 createCategoryDataset()方法也可以创建一个多系列的柱形图数据，语法如下：

```
createCategoryDataset(String rowKeyPrefix, String columnKeyPrefix, double[][] data)
```

参数说明

❶ rowKeyPrefix：表示柱形图中的系列名称。

❷ columnKeyPrefix：表示柱形图中的分类名称，即 X 轴上的刻度。

❸ data：表示具体的数据集合。

实例 227 多系列 3D 柱形图

光盘位置：光盘\MR\227　　高级　　趣味指数：★★★★★

实例说明

多系列的柱形图也可以绘制为 3D 效果。本实例演示如何绘制一个多系列的 3D 柱形图，运行效果如图 8.67 所示。

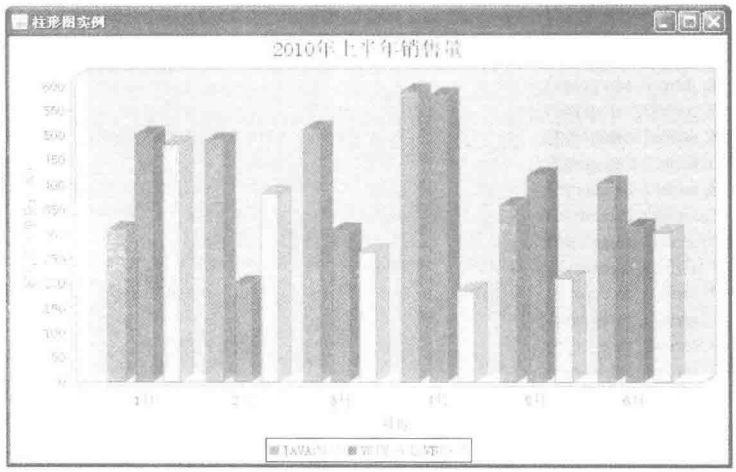


图 8.67　多系列 3D 柱形图

关键技术

在绘制多系列的 3D 柱形图时，还可以设置渲染效果。使用 CategoryPlot 类的 getRenderer()方法可以获取多系列 3D 的渲染效果，语法如下：

```
public CategoryItemRenderer getRenderer()
```

设计过程

（1）新建一个 Java 文件，同时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法内部创建多系列的分类数据集合，代码见实例 226。

（3）创建 getJFreeChart()方法，在方法中获取数据集，通过数据集创建柱形图的 JFreeChart 对象，代码见实例 224。

（4）创建 updateFont()方法，在方法中重新设置图表标题、标签等字体，代码见实例 211。

（5）创建 updatePlot()方法，在方法中获取 3D 的渲染效果实例，设置柱形图显示边线，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //分类图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    BarRenderer3D renderer = (BarRenderer3D) categoryPlot.getRenderer();
    //显示边线
    renderer.setDrawBarOutline(true);
}
```

（6）创建 createPlot()方法，在方法中获取 JFreeChart 对象并且调用 updateFont()方法修改字体，调用 updatePlot()方法更新图表的相关设置，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码见实例 211。

（7）在 main()方法中调用 createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BarDemo48 barDemo = new BarDemo48("柱形图实例");
        barDemo.createPlot();
        barDemo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(barDemo);
        //设置可以显示
        barDemo.setVisible(true);
}
```

秘笈心法

心法领悟 227：多系列的 3D 柱形图设置边线的方法。

在本实例中为多系列的 3D 柱形图设置了边线，使用了 BarRenderer 类的 setDrawBarOutline()方法实现，语法如下：

```
setDrawBarOutline(boolean draw)
```

参数说明

draw：表示是否显示柱形图的边线，当 draw 为 true 时，显示边线；当 draw 为 false 时，不显示边线。

第 9 章

扩展图表技术

- 区域图
- 气泡图
- 分步图
- 联合分类图
- 双轴图
- 折线图
- 环形图
- 堆积条形图
- 时序图

9.1　区　域　图

实例 228	基本区域图 光盘位置：光盘\MR\228	中级 趣味指数：★★★

实例说明

区域图是指根据基础数据在图表中绘制一定的区域，通过图表区域的形状和大小体现数据的变化。本实例演示如何绘制一个基本的区域图，运行效果如图 9.1 所示。

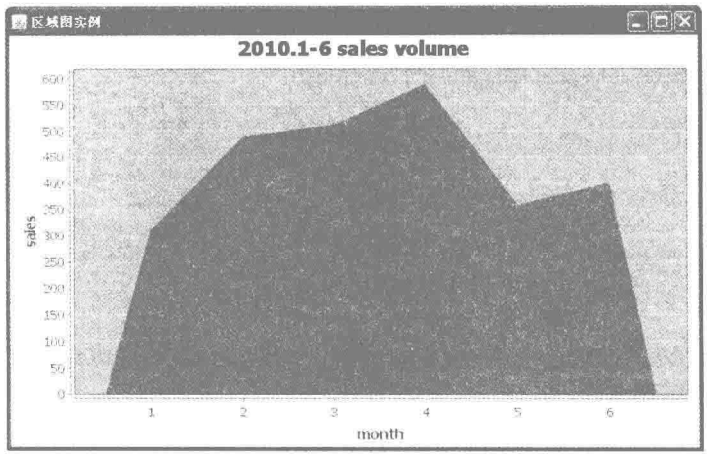


图 9.1　基本区域图

关键技术

ChartFactory 类的 createAreaChart()方法提供了创建基本区域图的方法，创建完成以后会返回一个 JFreeChart 对象，其语法如下：

```
public static JFreeChart createAreaChart(String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：图表的标题。

❷ categoryAxisLabel：图表类别标签，即 X 轴的名称。

❸ valueAxisLabel：图表数据标签，即 Y 轴的名称。

❹ dataset：区域图的数据集合。

❺ orientation：图表的显示方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultKeyedValues 类的实例中添加数据内容，然后通过

DatasetUtilities 类的 createCategoryDataset()方法创建一个数据集，代码如下：

```
private CategoryDataset getCategoryDataset() {
    DefaultKeyedValues keyedValues = new DefaultKeyedValues();
    //添加数据
    keyedValues.addValue("1", 310);
    keyedValues.addValue("2", 489);
    keyedValues.addValue("3", 512);
    keyedValues.addValue("4", 589);
    keyedValues.addValue("5", 359);
    keyedValues.addValue("6", 402);
    //创建数据集
    CategoryDataset dataset = DatasetUtilities.createCategoryDataset("java book", keyedValues);
    return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset();
    JFreeChart chart = ChartFactory.createAreaChart("2010 年上半年销售量",      //图表标题
    "month",                                                                    //X 轴标签
    "sales",                                                                    //Y 轴标签
    dataset,                                                                    //数据集
    PlotOrientation.VERTICAL,                                                   //图表方向：垂直
    true,                                                                       //显示图例
    false,                                                                      //不生成工具栏提示
    false                                                                       //不生成 URL 链接
    );
    return chart;
}
```

（4）创建 createPlot()方法，获取一个 JFreeChart 对象，然后调用父类的方法 setContentPane()把 JFreeChart 对象保存在窗体面板中，代码如下：

```
/**
 * 创建图表
 *
 * @param chart
 */
public void createPlot() {
    JFreeChart chart = getJFreeChart();
    //把 JFreeChart 面板保存在窗体中
    setContentPane(new ChartPanel(chart));

}
```

（5）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    AreaDemo1 demo = new AreaDemo1("区域图实例");
    demo.createPlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 228：使用 addValue()方法添加数据。

本实例使用 DefaultKeyedValues 类的 addValue()方法向实例中添加数据，语法如下：

```
addValue(Comparable key, double value)
```

参数说明

❶ key：图表的关键字，在本实例中是区域图中的分类。

❷ value：图表类别的值，在本实例中指区域图中的分类的数值。

实例 229　显示多分类区域图

光盘位置：光盘\MR\229　　中级　　趣味指数：★★★★

实例说明

普通区域图只显示一个分类的区域信息，而多分类区域图能更清楚地显示出不同分类之间的差别，运行效果如图 9.2 所示。

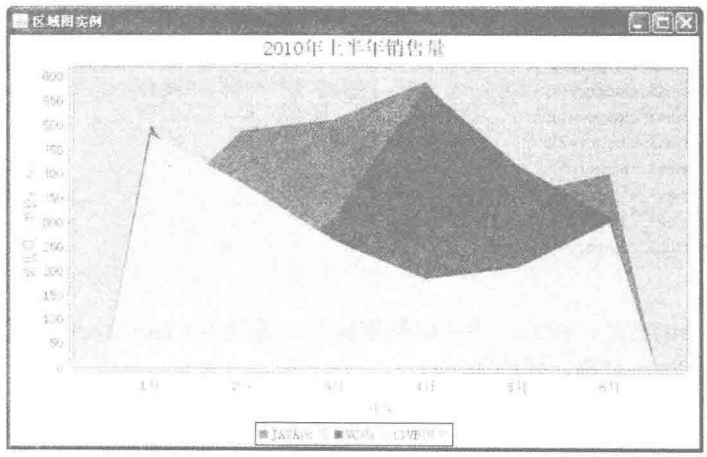


图 9.2　多分类区域图

关键技术

使用 DefaultCategoryDataset 类的 addValue()方法可以添加多类别的数据，为 JFreeChart 提供多类别的数据集合，语法如下：

```
public void addValue(double value, Comparable rowKey, Comparable columnKey)
```

参数说明

❶ value：表示某一分类的数据值。

❷ rowKey：数据集的行关键字，用来表示图表的系列名称。

❸ columnKey：数据集的列关键字，用来表示图表的类别名称。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集，代码如下：

```
private CategoryDataset getCategoryDataset() {

    //行关键字
    final String series1 = "JAVA 图书";
    final String series2 = "VC 图书";
    final String series3 = "VB 图书";
    //列关键字
    final String category1 = "1 月";
    final String category2 = "2 月";
    final String category3 = "3 月";
    final String category4 = "4 月";
```

```
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
    }
```

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象，代码如下：

```
    private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createAreaChart("2010 年上半年销售量",      //图表标题
        "月份",                                                                    //X 轴标签
        "销售量（单位：本）",                                                      //Y 轴标签
        dataset,                                                                   //数据集
        PlotOrientation.VERTICAL,                                                  //图表方向：垂直
        true,                                                                      //显示图例
        false,                                                                     //不生成工具栏提示
        false                                                                      //不生成 URL 链接
        );
        return chart;
    }
```

（4）创建 updateFont()方法，在方法中修改图表标题、轴线、标签等字体，代码如下：

```
    private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    }
```

（5）创建 createPlot()方法，获取一个 JFreeChart 对象，修改 JFreeChart 的默认字体，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        setContentPane(new ChartPanel(chart));
}
```

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        AreaDemo2 demo = new AreaDemo2("区域图实例");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 229：显示多分类区域的区域覆盖。

显示多分类区域时，每个系列都形成一个区域，不同的区域显示的内容可能会形成交叉，如果某一系列的区域值较小，则会被其他区域完全覆盖，所以有时在图表中找不到某一区域图时，并不是没有显示，而是被较大区域覆盖了。

实例 230　设置区域图透明度

中级

趣味指数：★★★★

实例说明

区域图中某一系列的区域值较小时，可能会被其他区域完全覆盖而无法看到。本实例为区域图设置一定的透明度，让图表更方便查看，运行效果如图 9.3 所示。

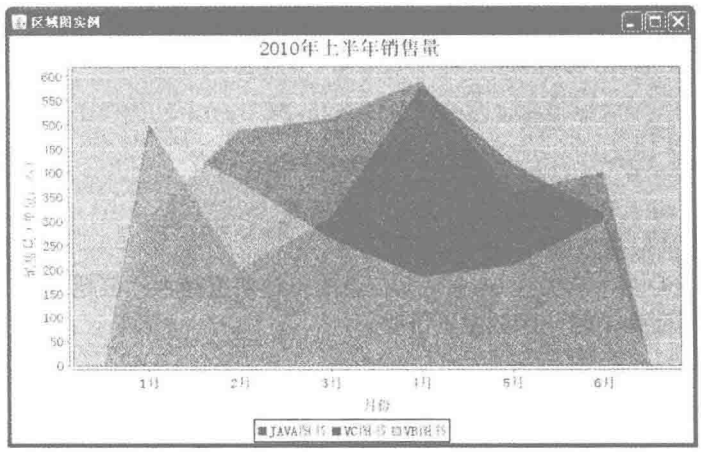


图 9.3　显示透明区域图

关键技术

使用 CategoryPlot 类的 setForegroundAlpha()方法可以设置图表的透明度，语法如下：

```
public void setForegroundAlpha(float alpha)
```

参数说明

alpha：表示图表的透明度，数值范围为 0f~1f。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集，代码见实例 229。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 229。

（4）创建 updateFont()方法，在方法中修改图表标题、轴线、标签等字体，代码见实例 229。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 类，然后设置透明度为 0.5f，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //分类图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    //设置透明度
    categoryPlot.setForegroundAlpha(0.5f);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，修改 JFreeChart 的默认字体，更新图表内容，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
    JFreeChart chart = getJFreeChart();
    //修改字体
    updateFont(chart);
    //更新图表
    updatePlot(chart);
    setContentPane(new ChartPanel(chart));
}
```

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    AreaDemo3 demo = new AreaDemo3("区域图实例");
    demo.createPlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 230：透明度的取值范围。

区域图的透明度默认值为 1f，范围为 0f~1f，其值越小，透明度越大；值越大，透明度越小。当透明度为 0 时，区域图被隐藏；当透明度为 1 时，区域图不透明。

实例 231 添加说明文字

光盘位置：光盘\MR\231　　中级　　趣味指数：★★★

实例说明

在区域图中添加说明文字，可以用来解释图表的含义，还可以用来表达图表的意义等。本实例演示如何添加说明文字，运行效果如图 9.4 所示。

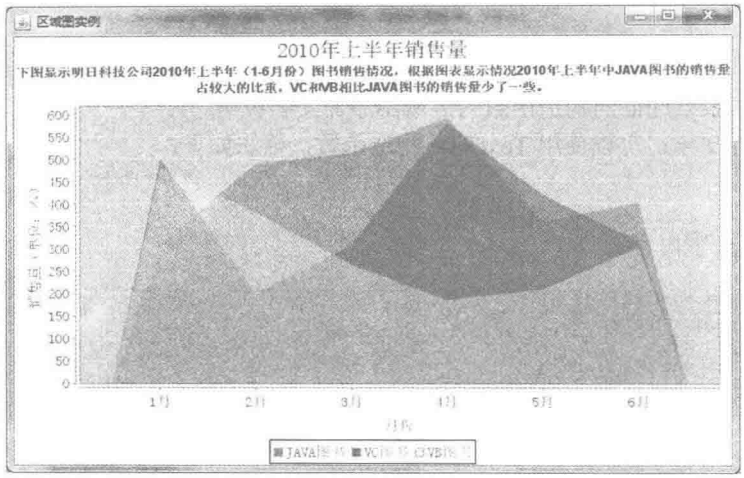


图 9.4　添加说明文字

关键技术

使用 JFreeChart 类的 addSubtitle()方法可以为区域图添加标题，同时也可添加一些说明性的文字，语法如下：

```
public void addSubtitle(Title subtitle)
```

参数说明

subtitle：表示为图表添加 Title 实例。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据生成分类数据集，代码见实例 229。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 229。

（4）创建 updateFont()方法，在方法中修改图表标题、轴线、标签等字体，代码见实例 229。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 类，为图表添加说明文字，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        categoryPlot.setForegroundAlpha(0.5f);
        //添加说明文字
        TextTitle subtitle = new TextTitle("下图显示明日科技公司 2010 上半年（1-6 月份）图书销售情况，根据图表显示情况，2010 年上半年中
JAVA 图书的销售量占较大的比重，VC 和 VB 相比 JAVA 图书的销售量少了一些。");
        chart.addSubtitle(subtitle);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，修改 JFreeChart 的默认字体，更新图表内容，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 230。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        AreaDemo4 demo = new AreaDemo4("区域图实例");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 231：通过 TextTitle 的构造函数为区域图添加文字说明。

为区域图添加说明性文字，可以使用 TextTitle 构造函数，语法如下：

```
TextTitle(String text)
```

参数说明

text：表示要添加的文字内容。

实例 232 设置说明文字位置

光盘位置：光盘\MR\232

中级

趣味指数：★★

实例说明

区域图中的说明文字默认情况下都显示在图表上方，字体为黑色。本实例将区域图说明文字设置成绿色，并且显示在图表左侧，运行效果如图 9.5 所示。

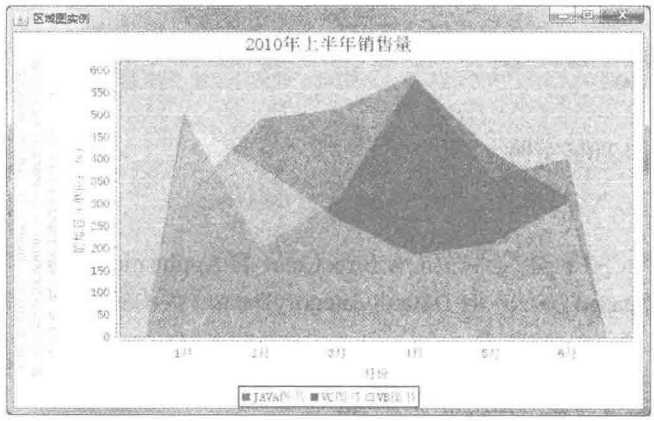


图 9.5 说明文字位置

关键技术

使用 Title 类的 setPosition()方法可以为区域图的说明文字设置方位，一般设置在图表的上、下、左、右 4 个方位，语法如下：

```
public void setPosition(RectangleEdge position)
```

参数说明

position：表示为图表的说明文字设置方位。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集，代码见实例 229。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 229。

（4）创建 updateFont()方法，在方法中修改图表标题、轴线、标签等字体，代码见实例 229。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 类，将图表说明文字置于图表左侧，同时设置文字

颜色为绿色，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        categoryPlot.setForegroundAlpha(0.5f);
        TextTitle subtitle = new TextTitle("下图显示明日科技公司 2010 上半年（1-6 月份）图书销售情况，根据图表显示情况，2010 年上半年中
JAVA 图书的销售量占较大的比重，VC 和 VB 相比 JAVA 图书的销售量少了一些。");
        //设置显示位置
        subtitle.setPosition(RectangleEdge.LEFT);
        subtitle.setPaint(Color.GREEN);
        chart.addSubtitle(subtitle);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，修改 JFreeChart 的默认字体，更新图表内容，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 230。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        AreaDemo5 demo = new AreaDemo5("区域图实例");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 232：通过 RectangleEdge 中的常量设置图表说明文字的位置。

设置图表说明文字的方位时可以使用 RectangleEdge 作为参数。RectangleEdge 中有 4 个常量用于表示说明文字的 4 个位置，其中，RectangleEdge.TOP 表示位于图表上部；RectangleEdge.BOTTOM 表示位于图表底部；RectangleEdge.LEFT 表示位于图表左侧；RectangleEdge.RIGHT 表示位于图表右侧。

实例 233　区域图 X 轴显示位置

光盘位置：光盘\MR\233　　中级　　趣味指数：★★★

实例说明

区域图的 X 坐标轴默认显示在图表底部。本实例演示如何将区域图 X 轴显示在图表顶部，运行效果如图 9.6 所示。

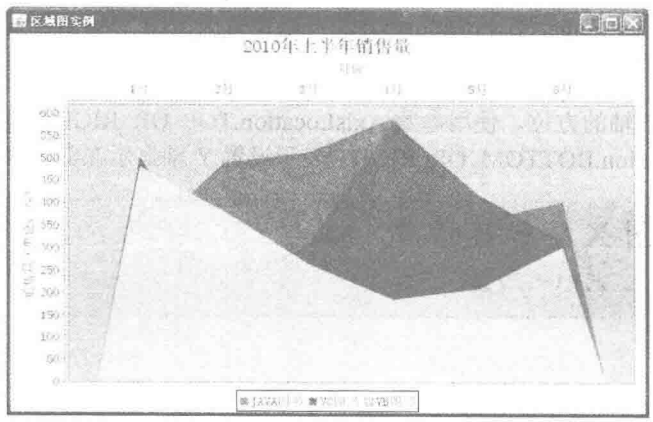


图 9.6　X 轴显示在顶部

关键技术

使用 CategoryPlot 类的 setDomainAxisLocation()方法可以设置区域图 X 轴的位置，语法如下：

```
public void setDomainAxisLocation(AxisLocation location)
```

参数说明

location：表示区域图 X 轴的方位，使用参数 AxisLocation.TOP_OR_LEFT 表示设置 X 轴显示在图表顶部或左侧；使用参数 AxisLocation.BOTTOM_OR_LEFT 表示设置 X 轴显示在图表底部或左侧。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集，代码见实例 229。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 229。

（4）创建 updateFont()方法，在方法中修改图表标题、轴线、标签等字体，代码见实例 229。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 类，为图表 X 轴设置显示位置，本实例把 X 轴显示在图表顶部，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //区域图 X 轴显示位置
        categoryPlot.setDomainAxisLocation(AxisLocation.TOP_OR_LEFT);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，修改 JFreeChart 的默认字体，更新图表内容，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 230。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        AreaDemo6 demo = new AreaDemo6("区域图实例");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 233：通过 setRangeAxisLocation()方法设置区域图 Y 轴的位置。

使用 CategoryPlot 类的 setRangeAxisLocation()方法可以设置区域图 Y 轴的位置，语法如下：

```
setRangeAxisLocation(AxisLocation location)
```

参数说明

location：表示区域图 Y 轴的方位，使用参数 AxisLocation.TOP_OR_RIGHT 表示设置 Y 轴显示在图表顶部或右侧；使用参数 AxisLocation.BOTTOM_OR_RIGHT 表示设置 Y 轴显示在图表底部或右侧。

实例 234 区域图 X 轴标签角度

光盘位置：光盘\MR\234

中级

趣味指数：★★★

实例说明

区域图的坐标轴标签角度可以根据需要进行调整。本实例演示如何调整区域图 X 轴的标签角度，运行效果如图 9.7 所示。

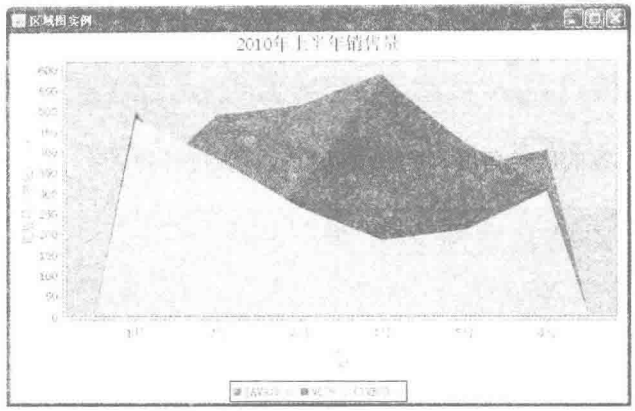


图 9.7　设置 X 轴标签角度

关键技术

使用 CategoryAxis 类的 setLabelAngle()方法可以设置区域图 X 轴标签的角度，语法如下：

```
public void setLabelAngle(double angle)
```

参数说明

angle：表示区域图 X 轴的标签旋转角度，此参数角度根据弧度计算。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集，代码见实例 229。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 229。

（4）创建 updateFont()方法，在方法中修改图表标题、轴线、标签等字体，代码见实例 229。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 类，为图表 X 轴标签设置弧度为 Math.PI*0.3，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //区域图 X 轴标签角度
        categoryAxis.setLabelAngle(Math.PI*0.3);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，修改 JFreeChart 的默认字体，更新图表内容，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 230。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        AreaDemo7 demo = new AreaDemo7("区域图实例");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 234：通过 setLabelAngle()方法设置区域图 Y 轴标签的角度。

区域图 Y 轴标签角度也可以根据需要自由设置，ValueAxis 类的 setLabelAngle()方法用于设置区域图 Y 轴标签的角度，语法如下：

```
setLabelAngle(double angle)
```

参数说明

angle：表示区域图 Y 轴标签的旋转角度，此参数角度根据弧度计算。

实例 235 区域图 X 轴尺度标签角度

光盘位置：光盘\MR\235　　中级　　趣味指数：★★★

实例说明

区域图坐标轴的尺度标签可以进行自由调整。本实例演示如何调整区域图 X 轴尺度标签的角度，运行效果如图 9.8 所示。

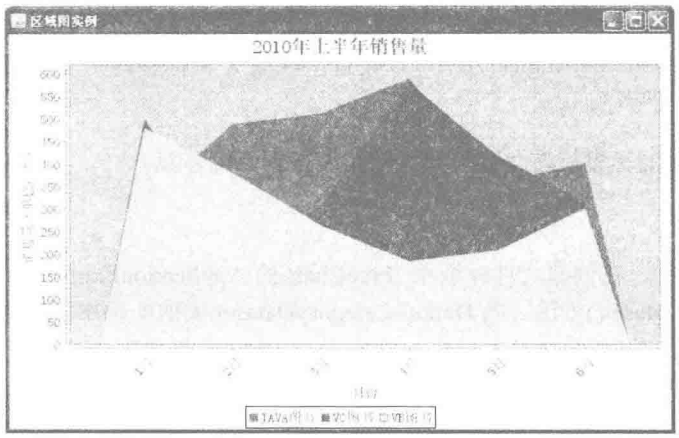


图 9.8　设置 X 轴尺度标签角度

关键技术

使用 CategoryAxis 类的 setCategoryLabelPositions()方法可以设置区域图 X 轴尺度标签的角度，语法如下：

```
public void setCategoryLabelPositions(CategoryLabelPositions positions)
```

参数说明

positions：表示区域图 X 轴尺度标签的旋转角度，参数使用 CategoryLabelPositions 类中定义的常量进行设置。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据，生成分类数据集，代码见实例 229。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 229。

（4）创建 updateFont()方法，在方法中修改图表标题、轴线、标签等字体，代码见实例 229。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 类，为图表 X 轴尺度标签设置角度为向上 45°，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //分类图表
```

```
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //区域图 X 轴尺度标签角度
        categoryAxis.setCategoryLabelPositions(CategoryLabelPositions.UP_45);
    }
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，修改 JFreeChart 的默认字体，更新图表内容，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 230。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        AreaDemo8 demo = new AreaDemo8("区域图实例");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 235：通过 setVerticalTickLabels()方法设置区域图 Y 轴尺度标签的角度。

区域图 Y 轴尺度标签的角度可以使用 ValueAxis 类的 setVerticalTickLabels()方法进行设置，语法如下：

```
setVerticalTickLabels(boolean flag)
```

参数说明

flag：表示区域图 Y 轴的尺度标签是否垂直。

实例 236　设置区域颜色

中级

趣味指数：★★★★

实例说明

区域图的颜色可以根据需要进行修改。本实例演示如何修改区域图指定系列的颜色，运行效果如图 9.9 所示。

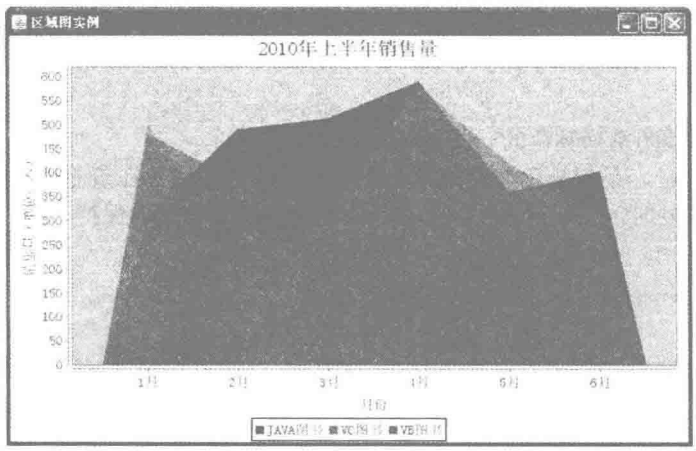


图 9.9　设置区域颜色

关键技术

使用 AreaRenderer 类的 setSeriesPaint()方法可以设置区域图指定系列的颜色，语法如下：

```
public void setSeriesPaint(int series, Paint paint)
```

参数说明

❶ series：表示指定区域图的系列。

❷ paint：表示指定区域图的颜色。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据生成分类数据集，代码见实例 229。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createAreaChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 229。

（4）创建 updateFont()方法，在方法中修改图表标题、轴线、标签等字体，代码见实例 229。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 类，通过 CategoryPlot 类获取 AreaRenderer 实例，为区域图指定系列颜色为黑色，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //设置透明度
        categoryPlot.setForegroundAlpha(0.6f);
        AreaRenderer renderer = (AreaRenderer) categoryPlot.getRenderer();
        //设置区域颜色
        renderer.setSeriesPaint(0, Color.BLACK);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，修改 JFreeChart 的默认字体，更新图表内容，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 230。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        AreaDemo9 demo = new AreaDemo9("区域图实例");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 236：设置区域图系列颜色的一些技巧。

在设置区域图系列颜色时，系列颜色与背景颜色不能一致，否则用户在查看图表时无法区分区域图中的颜色。在设置图表系列颜色的同时，JFreeChart 会让图示与图表中的颜色自动保持一致。

9.2 气 泡 图

实例 237　基本气泡图　　中级

光盘位置：光盘\MR\237　　趣味指数：★★★★

实例说明

气泡图会根据基础数据在图表中绘制一定的气泡形状，通过气泡的大小与位置体现数据的变化。本实例演示如何绘制一个基本的气泡图，运行效果如图 9.10 所示。

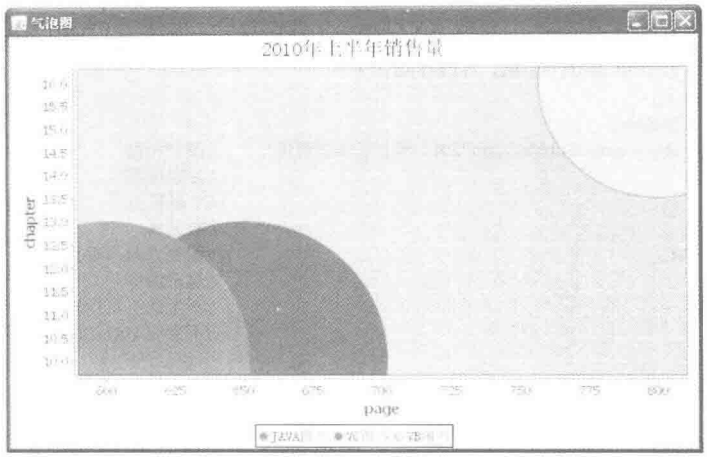


图 9.10　基本气泡图

关键技术

使用 ChartFactory 类的 createBubbleChart()方法可创建基本气泡图，创建完成后会返回一个 JFreeChart 对象，语法如下：

```
createBubbleChart(String title, String xAxisLabel, String yAxisLabel, XYZDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：图表的标题。

❷ xAxisLabel：表示 X 轴标签的名称。

❸ yAxisLabel：表示 Y 轴标签的名称。

❹ dataset：气泡图的数据集合。

❺ orientation：图表的显示方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 DefaultXYZDataset 类创建数据集，代码如下：

```
private XYZDataset getDataset() {
        //系列关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //创建分类数据集
        DefaultXYZDataset dataset = new DefaultXYZDataset();
        double data1 [] [] = new double[][] {{600}, {10}, {600/(10*10)}};
        double data2 [] [] = new double[][] {{650}, {10}, {650/(10*10)}};
        double data3 [] [] = new double[][] {{800}, {16}, {800/(16*10)}};
        dataset.addSeries(series1, data1);
        dataset.addSeries(series2, data2);
        dataset.addSeries(series3, data3);
        return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中获取 XYZDataset 数据集合，再使用 ChartFactory 类的 createBubbleChart()方法根据数据集合生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        XYZDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createBubbleChart("2010 年上半年销售量",    //图表标题
        "page",                                                                    //X 轴标签
        "chapter",                                                                 //Y 轴标签
        dataset,                                                                   //数据集
        PlotOrientation.VERTICAL,                                                  //图表方向：垂直
        true,                                                                      //显示图例
        false,                                                                     //不生成工具栏提示
        false                                                                      //不生成 URL 链接
        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改图表、图示标题等字体，代码如下：

```
private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图示
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
}
```

（5）创建 createPlot()方法，获取一个 JFreeChart 对象，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        setContentPane(new ChartPanel(chart));
}
```

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BubbleDemo1 demo = new BubbleDemo1("气泡图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 237：使用 addSeries()方法向集合添加数据。

本实例使用 DefaultXYZDataset 类创建数据集合，使用其 addSeries()方法向集合添加数据，语法如下：

```
addSeries(Comparable seriesKey, double[][] data)
```

参数说明

❶ seriesKey：表示指定气泡图的系列。

❷ data：表示指定气泡图系列的数据，其中 data 为 double[][]类型。在气泡图中，double 数组中含有 3 个元素，示例代码如下：

```
double data1 [] [] = new double[][] {{600}, {10}, {6}};
```

第 1 个元素为气泡图中某一系列的 x 坐标；第 2 个元素为气泡图中某一系列的 y 坐标；第 3 个元素为气泡图中某一系列的半径长度。

实例 238　气泡图 X 轴标签

光盘位置：光盘\MR\238　　中级　　趣味指数：★★

实例说明

用户可以根据需要设置气泡图 X 轴的标签。本实例演示如何修改气泡图默认的 X 轴标签，运行效果如图 9.11 所示。

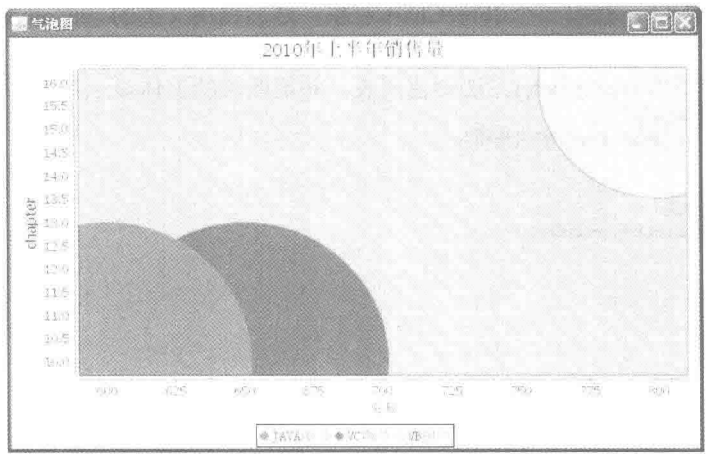


图 9.11　X 轴标签字体

关键技术

通过 XYPlot 类的 getDomainAxis()方法可以获取 NumberAxis 实例，使用 NumberAxis 类的 setLabelFont()方法可以修改气泡图 X 轴标签的默认字体，语法如下：

```
setLabelFont(Font font)
```

参数说明

font：表示气泡图 X 轴的标签字体。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 DefaultXYZDataset 类创建数据集，代码见实例 237。

（3）创建 getJFreeChart()方法，在方法中获取 XYZDataset 数据集合，再使用 ChartFactory 类的 createBubbleChart()方法根据数据集合生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        XYZDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createBubbleChart("2010 年上半年销售量",     //图表标题
        "页数",                                                                    //X 轴标签
        "chapter",                                                                 //Y 轴标签
        dataset,                                                                   //数据集
        PlotOrientation.VERTICAL,                                                  //图表方向：垂直
        true,                                                                      //显示图例
        false,                                                                     //不生成工具栏提示
        false                                                                      //不生成 URL 链接
        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改图表标题、图示标题、X 轴等字体，代码如下：

```
private void updateFont(JFreeChart chart) {
        //图表
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图示
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        XYPlot plot = chart.getXYPlot();
        //设置 X 轴字体
        NumberAxis numberAxis = (NumberAxis) plot.getDomainAxis();
        numberAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（5）创建 createPlot()方法，获取一个 JFreeChart 对象，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 237。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BubbleDemo2 demo = new BubbleDemo2("气泡图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 238：通过 setLabelPaint()方法设置标签颜色。

设置 X 轴标签字体时，同时可以使用 NumberAxis 类的 setLabelPaint()方法设置标签颜色，语法如下：

```
setLabelPaint(Paint paint)
```

参数说明

paint：表示气泡图 X 轴的标签字体颜色。

实例 239　气泡图 Y 轴标签

光盘位置：光盘\MR\239　　中级　　趣味指数：★★

实例说明

用户可以根据需要设置气泡图 Y 轴的标签。本实例演示如何修改气泡图默认的 Y 轴标签，运行效果如图 9.12 所示。

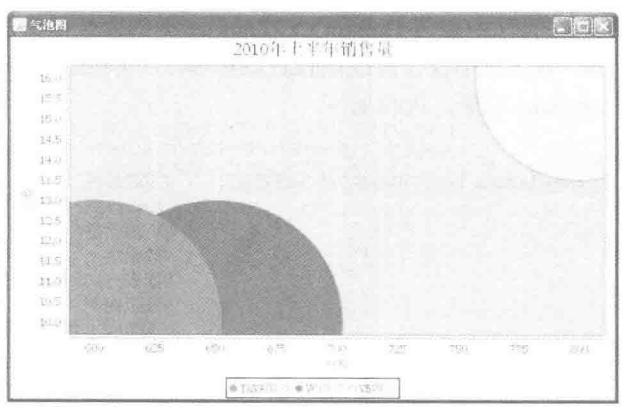


图 9.12　Y 轴标签字体

关键技术

通过 XYPlot 类的 getRangeAxis()方法可以获取 NumberAxis 实例，使用 NumberAxis 类的 setLabelFont()方法可以修改气泡图 Y 轴标签的默认字体，语法如下：

```
setLabelFont(Font font)
```

参数说明

font：表示气泡图 Y 轴的标签字体。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 DefaultXYZDataset 类创建数据集，代码见实例 237。

（3）创建 getJFreeChart()方法，在方法中获取 XYZDataset 数据集合，再使用 ChartFactory 类的 createBubbleChart()方法根据数据集合生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        XYZDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createBubbleChart("2010 年上半年销售量",     //图表标题
        "页数",                                                                    //X 轴标签
        "章数",                                                                    //Y 轴标签
        dataset,                                                                   //数据集
        PlotOrientation.VERTICAL,                                                  //图表方向：垂直
        true,                                                                      //显示图例
        false,                                                                     //不生成工具栏提示
        false                                                                      //不生成 URL 链接
        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改图表标题、图示标题、X 轴、Y 轴等字体，代码如下：

```
private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图示
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        XYPlot plot = chart.getXYPlot();
        //X 轴字体
        NumberAxis domainAxis = (NumberAxis) plot.getDomainAxis();
        domainAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴字体
        NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
        rangeAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
}
```

（5）创建 createPlot()方法，获取一个 JFreeChart 对象，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 237。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BubbleDemo3 demo = new BubbleDemo3("气泡图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 239：通过 setLabel()方法设置初始化的标签内容。

Y 轴的标签内容在初始化时就已经设置，使用 NumberAxis 类的 setLabel()方法可以重新设置初始化的标签内容，语法如下：

```
setLabel(String label)
```

参数说明

label：表示气泡图 Y 轴的标签内容。

实例 240 设置坐标范围

光盘位置：光盘\MR\240　　中级　　趣味指数：★★

实例说明

生成气泡图时，JFreeChar 会为坐标轴的范围设置默认值。本实例演示如何修改气泡图 X 轴、Y 轴坐标的默认范围，运行效果如图 9.13 所示。

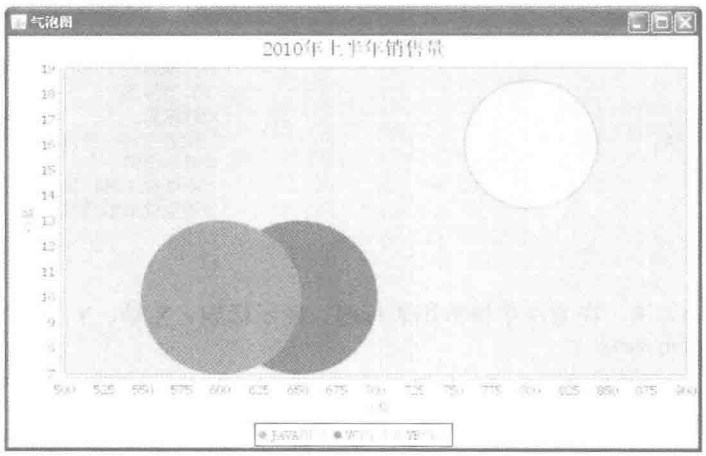


图 9.13　设置坐标轴范围

关键技术

通过 NumberAxis 类的 setLowerBound()方法可以设置坐标轴的最小值，使用 setUpperBound()方法可以设置坐标轴的最大值，语法如下：

```
setLowerBound(double min)
```

参数说明

min：表示坐标轴最小值。

```
setUpperBound (double max)
```

参数说明

max：表示坐标轴最大值。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 DefaultXYZDataset 类创建数据集，代码见实例 237。

（3）创建 getJFreeChart()方法，在方法中获取 XYZDataset 数据集合，再使用 ChartFactory 类的 createBubbleChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 239。

（4）创建 updateFont()方法，在方法中修改图表标题、图示标题、X 轴、Y 轴等字体，代码见实例 239。

（5）创建 updatePlot()方法，在方法中获取 XYPlot 类，然后分别设置 X、Y 坐标轴的显示范围，代码如下：

```
private void updatePlot(JFreeChart chart) {
        XYPlot plot = chart.getXYPlot();
        //X 轴
        NumberAxis domainAxis = (NumberAxis) plot.getDomainAxis();
        //设置范围
        domainAxis.setLowerBound(500);
        domainAxis.setUpperBound(900);
        //Y 轴
        NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
        //设置范围
        rangeAxis.setLowerBound(7);
        rangeAxis.setUpperBound(19);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，然后分别使用 updateFont()和 updatePlot()方法修改图表字体与图表设置，再调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //修改图表
        updatePlot(chart);
        setContentPane(new ChartPanel(chart));
}
```

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BubbleDemo4 demo = new BubbleDemo4("气泡图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 240：通过 setLowerBound()方法设置 X 轴与 Y 轴的范围。

设置坐标轴范围时，X 轴与 Y 轴的范围设置都是使用 NumberAxis 类的 setLowerBound()方法，其中 X 轴实例从 XYPlot 类的 getDomainAxis()方法中获取；Y 轴实例从 XYPlot 类的 getRangeAxis()方法中获取。

实例 241　设置透明度

光盘位置：光盘\MR\241　　中级　　趣味指数：★★★★

实例说明

气泡图在显示时有可能会发生重叠，通过设置气泡图的透明度可以更清晰地查看气泡图。本实例演示如何设置气泡图的透明度，运行效果如图 9.14 所示。

关键技术

通过 XYPlot 类的 setForegroundAlpha()方法可以设置气泡图的透明度，语法如下：

```
setForegroundAlpha(float alpha)
```

参数说明

alpha：表示气泡图的透明度。

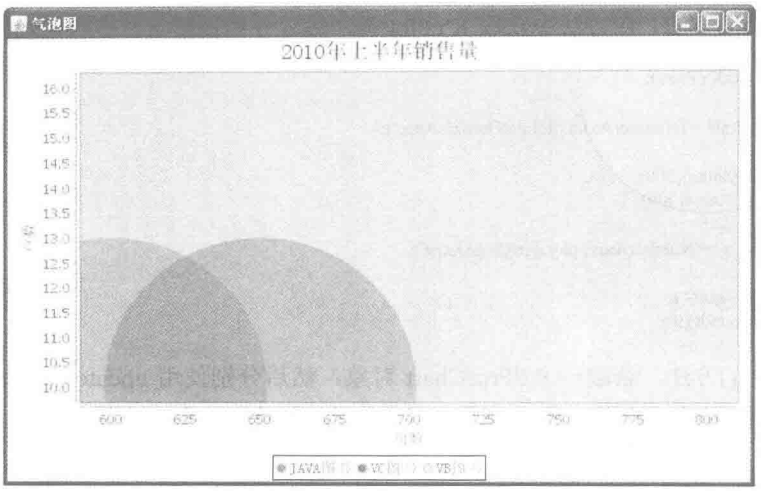


图 9.14　设置气泡图透明度

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 DefaultXYZDataset 类创建数据集，代码见实例 237。

（3）创建 getJFreeChart()方法，在方法中获取 XYZDataset 数据集合，再使用 ChartFactory 类的 createBubbleChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 239。

（4）创建 updateFont()方法，在方法中修改图表标题、图示标题、X 轴、Y 轴等字体，代码见实例 239。

（5）创建 updatePlot()方法，在方法中获取 XYPlot 类，然后为图表设置透明度为 0.4f，代码如下：

```
private void updatePlot(JFreeChart chart) {
        XYPlot plot = chart.getXYPlot();
        //设置透明度
        plot.setForegroundAlpha(0.4f);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，然后分别使用 updateFont()和 updatePlot()方法修改图表字体与图表设置，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 240。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BubbleDemo5 demo = new BubbleDemo5("气泡图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 241：通过 setBackgroundAlpha()方法设置气泡图背景透明度。

如果气泡图设置了背景，还可以为其背景设置透明度。背景透明度使用 XYPlot 类的 setBackgroundAlpha()方法设置，语法如下：

```
setBackgroundAlpha(float alpha)
```

参数说明：

alpha：表示气泡图背景的透明度。

实例 242　设置气泡颜色

光盘位置：光盘\MR\242　　中级　　趣味指数：★★★★

实例说明

可以根据需要对气泡图的颜色进行修改。本实例演示设置指定气泡的颜色为黄色，运行效果如图 9.15 所示。

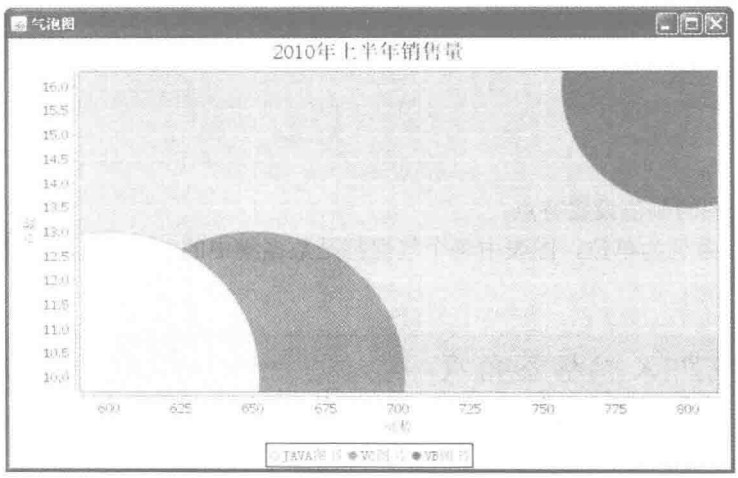


图 9.15　设置气泡颜色

关键技术

通过 XYBubbleRenderer 类的 setSeriesPaint()方法可以设置气泡图中指定气泡的颜色，语法如下：

```
setSeriesPaint(int series, Paint paint)
```

参数说明

❶ series：表示要设置的气泡的系列索引。

❷ paint：表示指定系列的气泡的颜色。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 DefaultXYZDataset 类创建数据集，代码见实例 237。

（3）创建 getJFreeChart()方法，在方法中获取 XYZDataset 数据集合，再使用 ChartFactory 类的 createBubbleChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 239。

（4）创建 updateFont()方法，在方法中修改图表标题、图示标题、X 轴、Y 轴等字体，代码见实例 239。

（5）创建 updatePlot()方法，在方法中获取 XYPlot 类并为图表设置透明度，然后为指定系列气泡设置颜色，代码如下：

```
private void updatePlot(JFreeChart chart) {
    XYPlot plot = chart.getXYPlot();
    //设置透明度
    plot.setForegroundAlpha(0.8f);
    //设置气泡颜色
    XYBubbleRenderer renderer =(XYBubbleRenderer) plot.getRenderer();
    renderer.setSeriesPaint(0, Color.YELLOW);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，然后分别使用 updateFont()和 updatePlot()方法修改图表字体与图表设置，调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 240。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BubbleDemo6 demo = new BubbleDemo6("气泡图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 242：气泡图的颜色设置特点。

气泡图的颜色设置以系列为单位，图表中每个气泡都是数据集中的一个系列，设置气泡颜色的顺序与系列在数据集中的顺序一致。

实例 243　气泡图 X 轴标签角度

光盘位置：光盘\MR\243　　中级　　趣味指数：★★★

实例说明

气泡图坐标轴的标签角度可以随意调整。本实例演示如何调整气泡图 X 轴的标签角度，运行效果如图 9.16 所示。

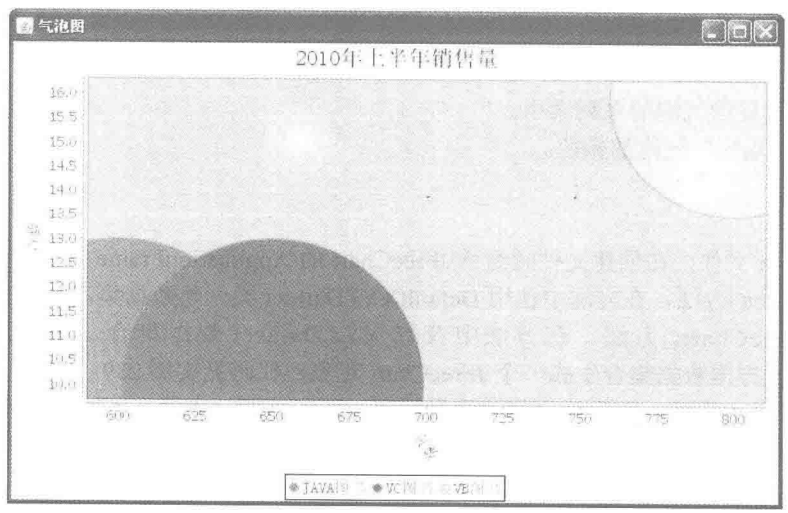


图 9.16　设置 X 轴标签角度

关键技术

使用 ValueAxis 类的 setLabelAngle()方法可以设置气泡图 X 轴标签的角度，语法如下：

```
setLabelAngle(double angle)
```

参数说明

angle：表示气泡图 X 轴的标签角度，此参数角度根据弧度计算。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 DefaultXYZDataset 类创建数据集，代码见实例 237。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createBubbleChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 239。

（4）创建 updateFont()方法，在方法中修改图表标题、轴线、标签等字体，代码见实例 239。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 类，为图表 X 轴标签设置弧度为 Math.PI*0.3，代码如下：

```
private void updatePlot(JFreeChart chart) {
        XYPlot plot = chart.getXYPlot();
        //设置透明度
        plot.setForegroundAlpha(0.8f);
        ValueAxis valueAxis = plot.getDomainAxis();
        //气泡图 X 轴标签角度
        valueAxis.setLabelAngle(Math.PI*0.3);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，修改 JFreeChart 的默认字体，更新图表内容，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 240。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BubbleDemo7 demo = new BubbleDemo7("气泡图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 243：通过 XYPlot 类设置气泡图 Y 轴标签角度。

气泡图 Y 轴的标签角度也可以根据需要进行自由设置。先使用 XYPlot 类的 getRangeAxis()方法获取 ValueAxis 类实例，再使用 ValueAxis 的 setLabelAngle()方法进行设置，语法如下：

```
setLabelAngle(double angle)
```

参数说明

angle：表示气泡图 Y 轴标签的旋转角度，以弧度计算。

实例 244　气泡图 X 轴尺度标签角度

中级

光盘位置：光盘\MR\244

趣味指数：★★★

实例说明

气泡图的分类名称如果过长，X 轴的尺度标签可能会重叠在一起，可以把尺度标签设为垂直来解决该问题。本实例演示如何调整气泡图 X 轴尺度标签的角度，运行效果如图 9.17 所示。

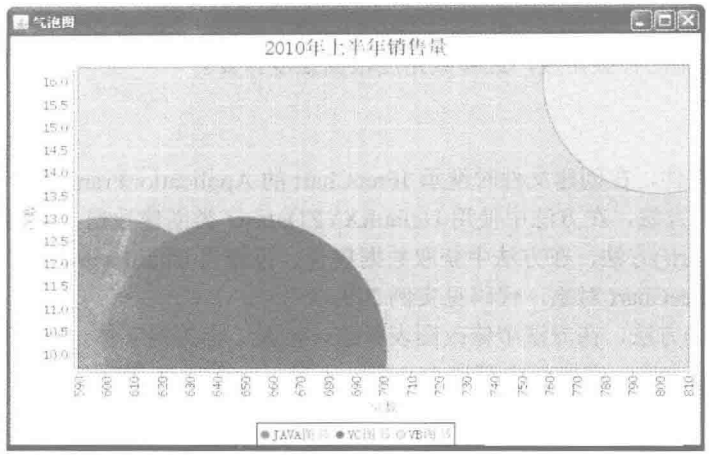


图 9.17　设置 X 轴尺度标签角度

关键技术

使用 ValueAxis 类的 setVerticalTickLabels()方法可以设置气泡图 X 轴尺度标签的角度，语法如下：

```
setVerticalTickLabels(boolean flag)
```

参数说明

flag：表示气泡图 X 轴尺度标签是否是垂直显示，当 flag 为 true 时，垂直显示；当 flag 为 false 时，水平显示。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，向 DefaultXYZDataset 类的实例中添加数据，生成 XYZDataset 数据集，代码见实例 237。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createBubbleChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 239。

（4）创建 updateFont()方法，在方法中修改图表标题、轴线、标签等字体，代码见实例 239。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 类，为图表 X 轴尺度标签设置垂直角度，代码如下：

```
private void updatePlot(JFreeChart chart) {
        XYPlot plot = chart.getXYPlot();
        //设置透明度
        plot.setForegroundAlpha(0.8f);
        ValueAxis valueAxis = plot.getDomainAxis();
        //气泡图 X 轴尺度标签角度
        valueAxis.setVerticalTickLabels(true);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，修改 JFreeChart 的默认字体，更新图表内容，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 240。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        BubbleDemo8 demo = new BubbleDemo8("气泡图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 244：通过 setVerticalTickLabels()方法设置 X 轴尺度标签的角度。

气泡图 Y 轴尺度标签的角度与 X 轴一样，可使用 ValueAxis 类的 setVerticalTickLabels()方法进行角度设置。标签角度只有水平和垂直两种。

9.3　分　步　图

实例 245　基本分步图　　中级

光盘位置：光盘\MR\245　　趣味指数：★★★★

实例说明

分步图通过图线模拟每个步骤，使用步骤的顺序与高低来体现业务量。本实例演示如何绘制一个基本分步图，运行效果如图 9.18 所示。

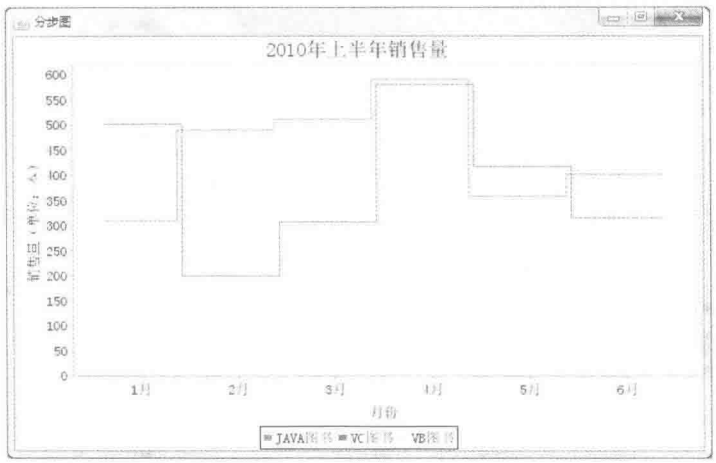


图 9.18　基本分步图

关键技术

（1）使用 CategoryStepRenderer 类的构造函数可以创建分步图渲染的实例，语法如下：

```
CategoryStepRenderer(boolean stagger)
```

参数说明

stagger：表示分步图是否交错显示，当 stagger 为 true 时，图线交错显示；当 stagger 为 false 时，图线不交错显示。

（2）使用 CategoryPlot 类的构造函数可以创建一个图表实例，根据渲染的实例创建不同的图表，语法如下：

```
CategoryPlot(CategoryDataset dataset, CategoryAxis domainAxis, ValueAxis rangeAxis, CategoryItemRenderer renderer)
```

参数说明

❶ dataset：表示分类图表的数据集。

❷ domainAxis：表示 X 轴实例。

❸ rangeAxis：表示 Y 轴实例。

❹ renderer：表示图表分类的渲染。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码如下：

```
private CategoryDataset getCategoryDataset() {
    //行关键字
    final String series1 = "JAVA 图书";
    final String series2 = "VC 图书";
    final String series3 = "VB 图书";
    //列关键字
    final String category1 = "1 月";
    final String category2 = "2 月";
    final String category3 = "3 月";
    final String category4 = "4 月";
    final String category5 = "5 月";
    final String category6 = "6 月";
    //创建分类数据集
    final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
    dataset.addValue(310, series1, category1);
    dataset.addValue(489, series1, category2);
    dataset.addValue(512, series1, category3);
    dataset.addValue(589, series1, category4);
    dataset.addValue(359, series1, category5);
    dataset.addValue(402, series1, category6);
    dataset.addValue(501, series2, category1);
    dataset.addValue(200, series2, category2);
    dataset.addValue(308, series2, category3);
    dataset.addValue(580, series2, category4);
    dataset.addValue(418, series2, category5);
    dataset.addValue(315, series2, category6);
    dataset.addValue(480, series3, category1);
    dataset.addValue(381, series3, category2);
    dataset.addValue(264, series3, category3);
    dataset.addValue(185, series3, category4);
    dataset.addValue(209, series3, category5);
    dataset.addValue(302, series3, category6);
    return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中获取 CategoryDataset 数据集合，再分别使用分类轴 CategoryAxis 类、数值轴 ValueAxis 类和分步渲染 CategoryStepRenderer 类创建 CategoryPlot 实例，然后使用 JFreeChart 类的构造函数生成 JFreeChart 实例，代码如下：

```
private JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset();
    //设置分步图渲染
    CategoryStepRenderer renderer = new CategoryStepRenderer(true);
    //创建 X 轴
    CategoryAxis domainAxis = new CategoryAxis("月份");
    //创建 Y 轴
    ValueAxis rangeAxis = new NumberAxis("销售量（单位：本）");
    //生成图表
    CategoryPlot plot = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer);
    final JFreeChart chart = new JFreeChart("2010 年上半年销售量", plot);
    return chart;
}
```

（4）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码如下：

```
private void updateFont(JFreeChart chart) {
    //标题
    TextTitle textTitle = chart.getTitle();
    textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
    LegendTitle legendTitle = chart.getLegend();
    legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
```

```
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    }
```

（5）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体，调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        setContentPane(new ChartPanel(chart));
    }
```

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        CategoryStepDemp1 demo = new CategoryStepDemp1("分步图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 245：JFreeChart 的构造函数。

JFreeChart 的构造函数可以根据 Plot 的实例创建一个 JFreeChart 对象，语法如下：

```
JFreeChart(String title, Plot plot)
```

参数说明

❶ title：表示图表的名称。

❷ plot：表示图表 Plot 的实例。

实例 246　加粗分步图

光盘位置：光盘\MR\246　　中级　　趣味指数：★★★★

实例说明

分步图显示时，图线的默认笔触比较细。本实例演示如何加粗指定分步图图线，运行效果如图 9.19 所示。

关键技术

使用 CategoryStepRenderer 类的 setSeriesStroke()方法可以修改分步图图线的笔触，语法如下：

```
setSeriesStroke(int series, Stroke stroke)
```

参数说明

❶ series：表示分步图的指定系列。

❷ stroke：表示指定系列的笔触。

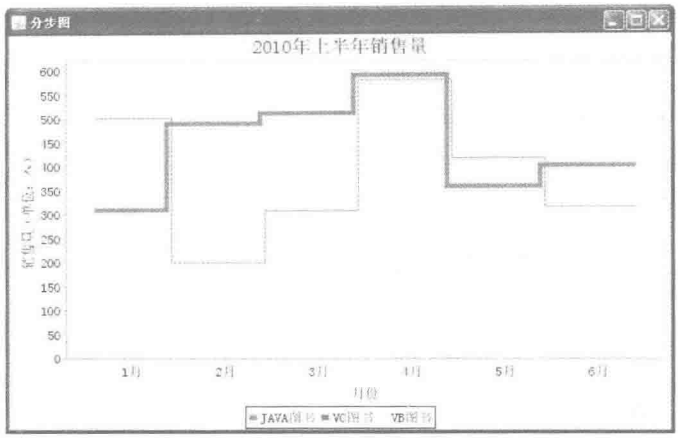


图 9.19　加粗分步图

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码见实例 245。

（3）创建 getJFreeChart()方法，在方法中获取 CategoryDataset 数据集合，再分别使用分类轴 CategoryAxis 类、数值轴 ValueAxis 类和分步渲染 CategoryStepRenderer 类创建 CategoryPlot 实例，然后使用 JFreeChart 类的构造函数生成 JFreeChart 实例，代码见实例 245。

（4）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 245。

（5）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 类，通过 CategoryPlot 的实例获取 CategoryStepRenderer 对象，然后加粗第一个系列图线的笔触，代码如下：

```
private void updatePlot(JFreeChart chart) {
        CategoryPlot plot = chart.getCategoryPlot();
        CategoryStepRenderer renderer =   (CategoryStepRenderer) plot.getRenderer();
        //加粗分步图
        renderer.setSeriesStroke(0,new BasicStroke(5));
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体、更新图表内容，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //更新图表
        updatePlot(chart);
        setContentPane(new ChartPanel(chart));
}
```

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        CategoryStepDemp2 demo = new CategoryStepDemp2("分步图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 246：通过 BasicStroke 类的构造方法修改笔触的粗细。

使用 BasicStroke 类的构造方法可以修改笔触的粗细，语法如下：

BasicStroke(float width)

参数说明

width：表示图表的粗细，数值越大，笔触越粗；数值越小，笔触越细。

实例 247　显示虚线效果

光盘位置：光盘\MR\247　　中级　　趣味指数：★★★★

实例说明

通过设置分步图图线的笔触，可以修改图线的显示效果。本实例演示如何把分步图的图线设置为虚线，运行效果如图 9.20 所示。

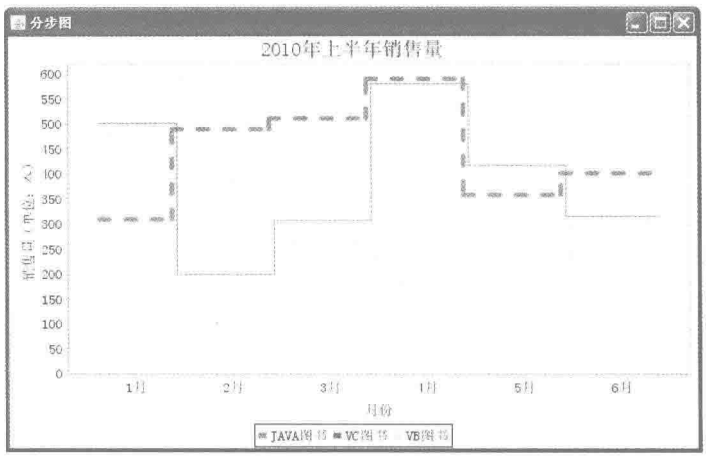


图 9.20　显示虚线效果

关键技术

BasicStroke 类有多个构造方法，使用 BasicStroke 类的构造方法可以进一步修改分步图图线的笔触效果，语法如下：

BasicStroke(float width, int cap, int join, float miterlimit, float dash[], float dash_phase)

参数说明

❶ width：表示笔触宽度。

❷ cap：表示笔触线条的端点样式。

❸ join：表示应用在路径线段交汇处的样式。

❹ miterlimit：表示斜接处的剪裁限制，其值必须大于或等于 1.0f。

❺ dash：表示虚线模式的数组。

❻ dash_phase：表示开始虚线模式的偏移量。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码见实例 245。

（3）创建 getJFreeChart()方法，在方法中获取 CategoryDataset 数据集合，再分别使用分类轴 CategoryAxis 类、数值轴 ValueAxis 类和分步渲染 CategoryStepRenderer 类创建 CategoryPlot 实例，然后使用 JFreeChart 类的

构造函数生成 JFreeChart 实例，代码见实例 245。

（4）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 245。

（5）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 类，通过 CategoryPlot 的实例获取 CategoryStepRenderer 对象，然后创建虚线笔触，代码如下：

```
private void updatePlot(JFreeChart chart) {
        CategoryPlot plot = chart.getCategoryPlot();
        CategoryStepRenderer renderer =  (CategoryStepRenderer) plot.getRenderer();
        //显示虚线效果
        renderer.setSeriesStroke(0,new BasicStroke(5,BasicStroke.CAP_ROUND,BasicStroke.JOIN_ROUND,1, new float[] {10.0f, 16.0f}, 0.0f));
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体、更新图表，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 246。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        CategoryStepDemp3 demo = new CategoryStepDemp3("分步图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 247：设置笔触线条的端点样式。

表示笔触线条端点样式的参数有 CAP_BUTT、CAP_ROUND 和 CAP_SQUARE，其中，CAP_BUTT 表示没有封闭的线条端点不使用样式；CAP_ROUND 表示没有封闭的线条端点使用圆形样式；CAP_SQUARE 表示没有封闭的线条端点使用方形样式。

实例 248　设置分步图颜色

光盘位置：光盘\MR\248　　中级　　趣味指数：★★★★★

实例说明

在使用分步图时，可以根据实际需要设置图线的颜色。本实例演示如何设置图线颜色为黑色，运行效果如图 9.21 所示。

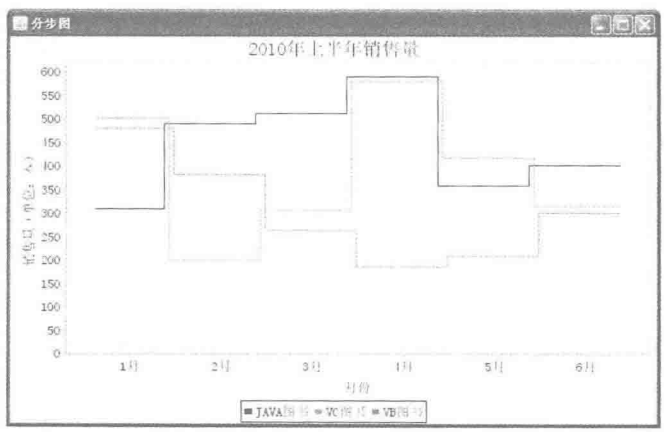


图 9.21　设置分步图颜色

关键技术

分步图的颜色根据系列设置，使用 CategoryStepRenderer 类的 setSeriesPaint()方法可以设置分步图指定系列颜色，语法如下：

```
setSeriesPaint(int series, Paint paint)
```

参数说明

❶ series：表示指定分步图的系列索引。

❷ paint：表示分步图中指定系列的颜色。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码见实例 245。

（3）创建 getJFreeChart()方法，在方法中获取 CategoryDataset 数据集合，再分别使用分类轴 CategoryAxis 类、数值轴 ValueAxis 类和分步渲染 CategoryStepRenderer 类创建 CategoryPlot 实例，然后使用 JFreeChart 类的构造函数生成 JFreeChart 实例，代码见实例 245。

（4）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 245。

（5）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 类，通过 CategoryPlot 的实例获取 CategoryStepRenderer 对象，然后设置分步图第一个索引的线条为黑色，代码如下：

```
private void updatePlot(JFreeChart chart) {
    CategoryPlot plot = chart.getCategoryPlot();
    CategoryStepRenderer renderer =  (CategoryStepRenderer) plot.getRenderer();
    //设置分步图颜色
    renderer.setSeriesPaint(0, Color.BLACK);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体、更新图表，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 246。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    CategoryStepDemp4 demo = new CategoryStepDemp4("分步图");
    demo.createPlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 248：设置分布图颜色的技巧。

默认情况下，分步图的颜色由 JFreeChart 自动设置，对分步图手动设置颜色时，尽量不要使用与背景相近的颜色，否则用户无法区分分步图中的线条。

实例 249　隐藏分步图

光盘位置：光盘\MR\249　　中级　　趣味指数：★★★★

实例说明

使用分步图时，有时需要只显示某一系列的图形，而将其他系列隐藏起来。本实例演示如何隐藏指定分步

图的线条，运行效果如图 9.22 所示。

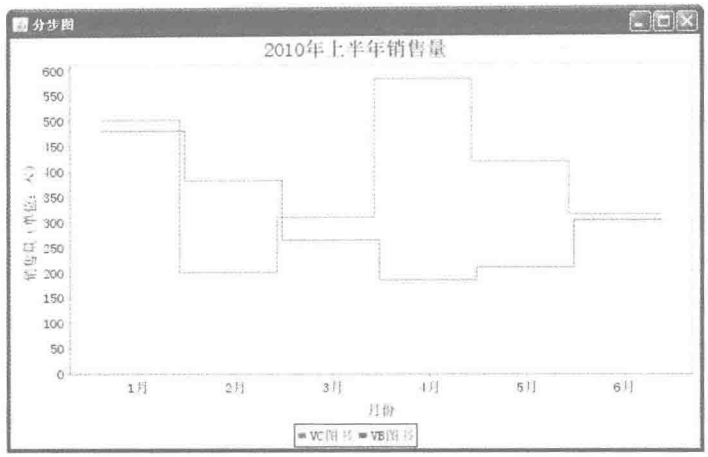


图 9.22　隐藏分步图

关键技术

分步图的隐藏和显示根据系列索引设置，使用 CategoryStepRenderer 类的 setSeriesVisible()方法可以隐藏或者显示分步图指定系列，语法如下：

```
setSeriesVisible(int series, Boolean visible)
```

参数说明

❶ series：表示指定分步图的系列索引。

❷ visible：表示分步图中指定系列是否显示。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset 方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码见实例 245。

（3）创建 getJFreeChart()方法，在方法中获取 CategoryDataset 数据集合，再分别使用分类轴 CategoryAxis 类、数值轴 ValueAxis 类和分步渲染 CategoryStepRenderer 类创建 CategoryPlot 实例，然后使用 JFreeChart 类的构造函数生成 JFreeChart 实例，代码见实例 245。

（4）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 245。

（5）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 类，通过 CategoryPlot 的实例获取 CategoryStepRenderer 对象，然后隐藏分步图第一个索引的线条，代码如下：

```
private void updatePlot(JFreeChart chart) {
        CategoryPlot plot = chart.getCategoryPlot();
        CategoryStepRenderer renderer = (CategoryStepRenderer) plot.getRenderer();
        //隐藏分步图
        renderer.setSeriesVisible(0, new Boolean(false));
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体、更新图表，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 246。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        CategoryStepDemp5 demo = new CategoryStepDemp5("分步图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
```

```
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 249：分布图的隐藏和显示。

在隐藏和显示分步图时，只能按图表中的系列索引进行操作，其中，系列索引与数据集中添加的系列顺序一致，并且索引数量与数据集中索引数量一致。

实例 250　分步图 X 轴显示位置

光盘位置：光盘\MR\250　　中级　　趣味指数：★★★

实例说明

在垂直分布图中，X 轴的显示位置可以显示在图表顶部或者底部。本实例演示如何把 X 轴显示在图表顶部，运行效果如图 9.23 所示。

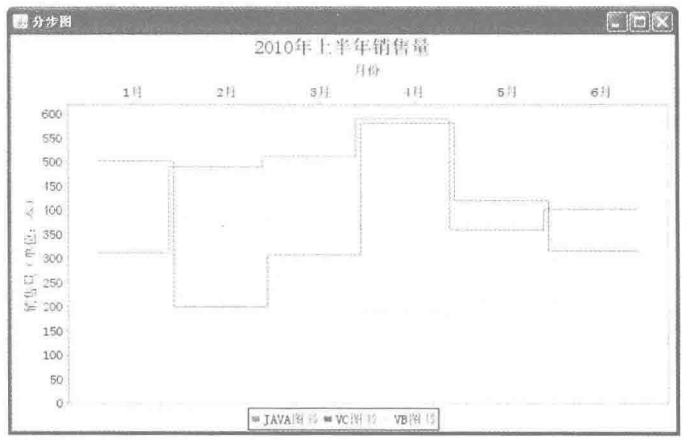


图 9.23　设置分步图 X 轴

关键技术

分步图的 X 轴在默认情况下显示在图表底部，使用 CategoryStepRenderer 类的 setDomainAxisLocation()方法可以修改 X 轴默认的显示位置，语法如下：

```
setDomainAxisLocation(AxisLocation location)
```

参数说明

location：表示 X 轴的显示位置，该参数使用 AxisLocation 类中的常量进行设置，AxisLocation.TOP_OR_LEFT 表示分步图 X 轴显示在顶部或者左侧；AxisLocation.BOTTOM_OR_RIGHT 表示分步图 X 轴显示在底部或者右侧。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码见实例 245。

（3）创建 getJFreeChart()方法，在方法中获取 CategoryDataset 数据集合，再分别使用分类轴 CategoryAxis 类、数值轴 ValueAxis 类和分步渲染 CategoryStepRenderer 类创建 CategoryPlot 实例，然后使用 JFreeChart 类的构造函数生成 JFreeChart 实例，代码见实例 245。

（4）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 245。

（5）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 类，通过 setDomainAxisLocation()方法设置 X 轴显示在图表顶部，代码如下：

```
private void updatePlot(JFreeChart chart) {
        CategoryPlot plot = chart.getCategoryPlot();
        //分步图 X 轴显示位置
        plot.setDomainAxisLocation(AxisLocation. TOP_OR_LEFT);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体、更新图表，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 246。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        CategoryStepDemp6 demo = new CategoryStepDemp6("分步图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 250：通过 setRangeAxisLocation()方法设置分步图 Y 轴显示位置。

使用 CategoryPlot 类的 setRangeAxisLocation()方法可以设置分步图 Y 轴的显示位置，语法如下：

```
setRangeAxisLocation(AxisLocation location)
```

参数说明

location：表示 Y 轴的显示位置，location 使用 AxisLocation 类中的常量进行设置，AxisLocation.TOP_OR_RIGHT 表示分步图 Y 轴显示在顶部或者右侧；AxisLocation.BOTTOM_OR_LEFT 表示分步图 Y 轴显示在底部或者左侧。

实例 251 分步图 X 轴标签角度

光盘位置：光盘\MR\251　　中级　　趣味指数：★★★

实例说明

设置分布图的 X 轴标签时，可以改变其显示角度。本实例演示如何改变 X 轴标签的角度，运行效果如图 9.24 所示。

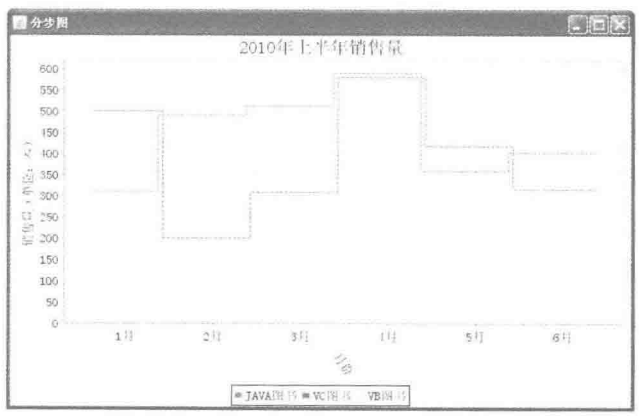


图 9.24　设置 X 轴标签角度

关键技术

使用 CategoryAxis 类的 setLabelAngle()方法可以设置分步图 X 轴的标签角度，语法如下：

```
setLabelAngle(double angle)
```

参数说明

angle：表示 X 轴标签的弧度，通过弧度计算 X 轴标签的角度。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码见实例 245。

（3）创建 getJFreeChart()方法，在方法中获取 CategoryDataset 数据集合，再分别使用分类轴 CategoryAxis 类、数值轴 ValueAxis 类和分步渲染 CategoryStepRenderer 类创建 CategoryPlot 实例，然后使用 JFreeChart 类的构造函数生成 JFreeChart 实例，代码见实例 245。

（4）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 245。

（5）创建 updatePlot()方法，在方法中获取一个 CategoryAxis 类，通过 setLabelAngle()方法设置 X 轴标签的角度，代码如下：

```
private void updatePlot(JFreeChart chart) {
        CategoryPlot plot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = plot.getDomainAxis();
        //分步图 X 轴标签角度
        categoryAxis.setLabelAngle(Math.PI*0.3);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体、更新图表，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 246。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        CategoryStepDemp7 demo = new CategoryStepDemp7("分步图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 251：通过 setLabelAngle()方法设置分步图 Y 轴的标签角度。

使用 ValueAxis 类的 setLabelAngle()方法可以设置分步图 Y 轴的标签角度，语法如下：

```
setLabelAngle(double angle)
```

参数说明

angle：表示 Y 轴标签的弧度，通过弧度计算 Y 轴标签的角度。

实例 252　分步图 X 轴尺度标签角度

中级

光盘位置：光盘\MR\252

趣味指数：★★★

实例说明

如果分步图的 X 轴的尺度标签内容过长，标签文字可能重叠在一起不便于辨认。本实例演示如何设置 X 轴的尺度标签角度，运行效果如图 9.25 所示。

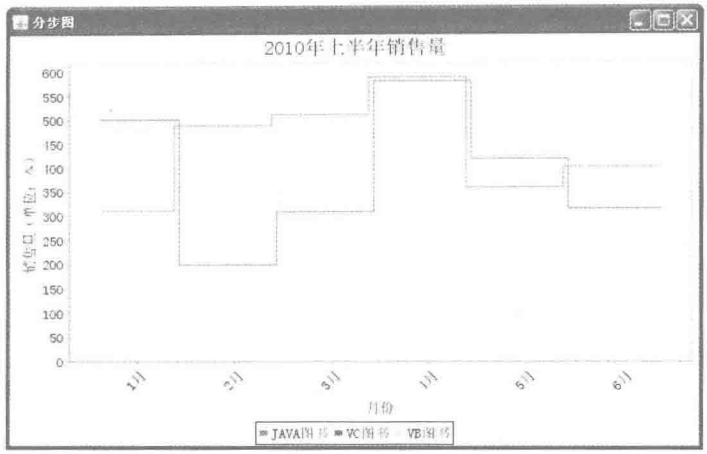


图 9.25　设置 X 轴尺度标签角度

关键技术

使用 CategoryAxis 类的 setCategoryLabelPositions()方法可以设置分步图 X 轴的尺度标签角度，语法如下：

```
setCategoryLabelPositions(CategoryLabelPositions positions)
```

参数说明

positions：表示 X 轴的尺度标签角度。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码见实例 245。

（3）创建 getJFreeChart()方法，在方法中获取 CategoryDataset 数据集合，再分别使用分类轴 CategoryAxis 类、数值轴 ValueAxis 类和分步渲染 CategoryStepRenderer 类创建 CategoryPlot 实例，然后使用 JFreeChart 类的构造函数生成 JFreeChart 实例，代码见实例 245。

（4）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 245。

（5）创建 updatePlot()方法，在方法中获取一个 CategoryAxis 类，通过 setCategoryLabelPositions()方法设置 X 轴尺度标签的角度，代码如下：

```
private void updatePlot(JFreeChart chart) {
        CategoryPlot plot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = plot.getDomainAxis();
        //区域图 X 轴尺度标签角度
        categoryAxis.setCategoryLabelPositions(CategoryLabelPositions.UP_45);
}
```

（6）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体、更新图表，接着调用父类 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 246。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        CategoryStepDemp8 demo = new CategoryStepDemp8("分步图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 252：CategoryLabelPositions 类中尺度标签的角度常量。

在 CategoryLabelPositions 类中定义了尺度标签的角度常量。CategoryLabelPositions.UP_45 表示角度向上旋转 45°；CategoryLabelPositions.UP_90 表示角度向上旋转 90°；CategoryLabelPositions.DOWN_45 表示角度向下旋转 45°；CategoryLabelPositions.DOWN_90 表示角度向下旋转 90°。

9.4　联合分类图

实例 253　生成线形图与柱形图　　高级

光盘位置：光盘\MR\253　　趣味指数：★★★★★

实例说明

联合分类图由多种图表共同组成，通过不同的图表展示同一个数据集。本实例演示如何生成线形图与柱形图的联合分类图，运行效果如图 9.26 所示。

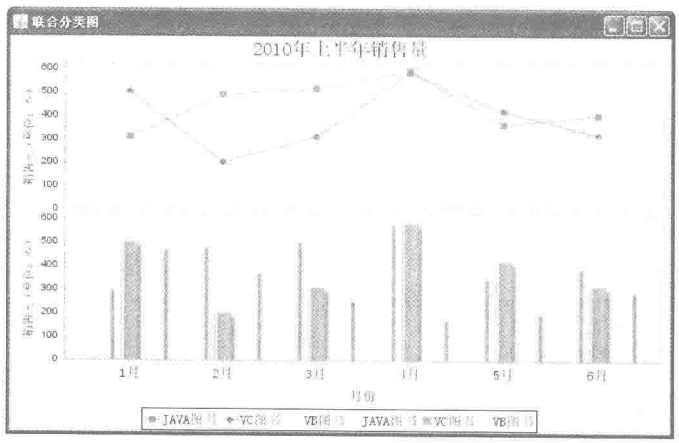


图 9.26　生成线形图与柱形图

关键技术

利用 CombinedDomainCategoryPlot 类的 add()方法可以添加多个 Plot，通过 CombinedDomainCategoryPlot 类可以让一个图表展示多个图形，语法如下：

```
add(CategoryPlot subplot)
```

参数说明

subplot：表示需要添加的 Plot 实例。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码如下：

```
private CategoryDataset getCategoryDataset() {
    //行关键字
    final String series1 = "JAVA 图书";
```

```
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
    }
```

（3）创建 getJFreeChart()方法，在方法中创建 LineAndShapeRenderer 和 BarRenderer 实例，分别使用这两个渲染生成两个 Plot，再把这两个 Plot 添加到 CombinedDomainCategoryPlot 实例中，然后生成 JFreeChart 对象，代码如下：

```
    private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        //生成线形图渲染
        LineAndShapeRenderer renderer1 = new LineAndShapeRenderer();
        //生成柱形图渲染
        BarRenderer renderer2 = new BarRenderer();
        //设置 X 轴
        CategoryAxis domainAxis = new CategoryAxis("月份");
        //设置 Y 轴
        NumberAxis rangeAxis = new NumberAxis("销售量（单位：本）");
        //设置图表
        CategoryPlot plot1 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer1);
        CategoryPlot plot2 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer2);
        //设置联合分类图表
        final CombinedDomainCategoryPlot plot = new CombinedDomainCategoryPlot(domainAxis);
        plot.add(plot1);
        plot.add(plot2);
        JFreeChart chart = new JFreeChart("2010 年上半年销售量", plot);
        return chart;
    }
```

（4）创建 updateFont()方法，在方法中修改图表标题和分类轴标签等字体，代码如下：

```
    private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //分类轴字体
```

```
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //分类轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    }
```

（5）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
    public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        setContentPane(new ChartPanel(chart));
    }
```

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        CombinedCategoryDemo1 demo = new CombinedCategoryDemo1("联合分类图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 253：通过 CombinedDomainCategoryPlot 类的构造方法创建联合分类图。

使用 CombinedDomainCategoryPlot 类的构造方法可以创建联合分类图的实例，语法如下：

```
CombinedDomainCategoryPlot(CategoryAxis domainAxis)
```

参数说明

domainAxis：表示联合分类图中的 X 分类轴。在联合分类图中，多个图表共用一个分类轴。

实例 254　设置图表高度

光盘位置：光盘\MR\254　　中级　　趣味指数：★★★★

实例说明

本实例分为上下两个图表，默认情况下两个图表的高度一致，根据实际需要，可以修改指定图表的默认高度值，运行效果如图 9.27 所示。

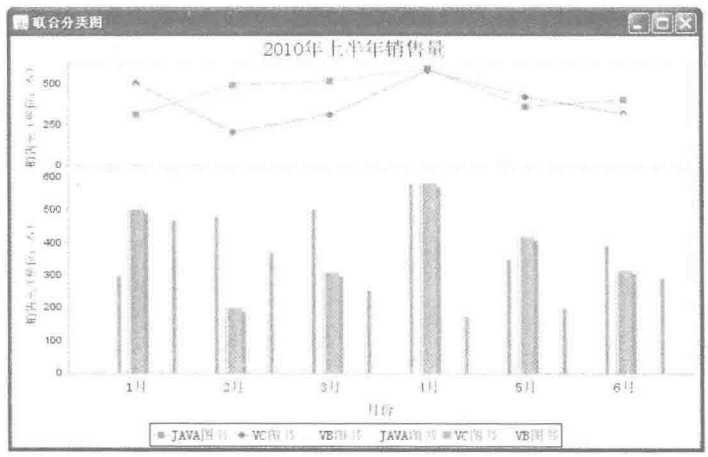


图 9.27　修改线形图与柱形图高度

关键技术

使用 CombinedDomainCategoryPlot 类的 add()方法可以添加多个 Plot，同时可以设置指定图表的高度，语法如下：

```
add(CategoryPlot subplot, int weight)
```

参数说明

❶ subplot：表示准备添加的 Plot 实例。

❷ weight：表示添加的 Plot 实例的高度。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码见实例 253。

（3）创建 getJFreeChart()方法，在方法中创建 LineAndShapeRenderer 和 BarRenderer 实例，分别使用这两个渲染生成两个 Plot，再把这两个 Plot 添加到 CombinedDomainCategoryPlot 实例中，然后生成 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        //生成线形图渲染
        LineAndShapeRenderer renderer1 = new LineAndShapeRenderer();
        //生成柱形图渲染
        BarRenderer renderer2 = new BarRenderer();
        //设置 X 轴
        CategoryAxis domainAxis = new CategoryAxis("月份");
        //设置 Y 轴
        NumberAxis rangeAxis = new NumberAxis("销售量（单位：本）");
        //设置图表
        CategoryPlot plot1 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer1);
        CategoryPlot plot2 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer2);
        //设置联合分类图表
        final CombinedDomainCategoryPlot plot = new CombinedDomainCategoryPlot(
                        domainAxis);
                plot.add(plot1, 1);
                plot.add(plot2, 2);
        JFreeChart chart = new JFreeChart("2010 年上半年销售量", plot);
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改图表标题和分类轴标签等字体，代码见实例 253。

（5）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 253。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        CombinedCategoryDemo2 demo = new CombinedCategoryDemo2("联合分类图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 254：上下图表的高度设置。

向 CombinedDomainCategoryPlot 类中添加图表实例时，设置的高度数值是与总图表高度的比例值。例如，线形图的高度设置为 1，柱形图的高度设置为 2 时，表示图表的总高度为 3。那么线形图的高度占总图表的 1/3，柱形图的高度占总图表的 2/3。

实例 255　设置图表位置

光盘位置：光盘\MR\255　　中级　　趣味指数：★★★★

实例说明

可以根据实际情况，设置联合分类图中多个图表的显示位置。本实例把柱形图显示在图表上面，把线形图显示在图表下面，运行效果如图 9.28 所示。

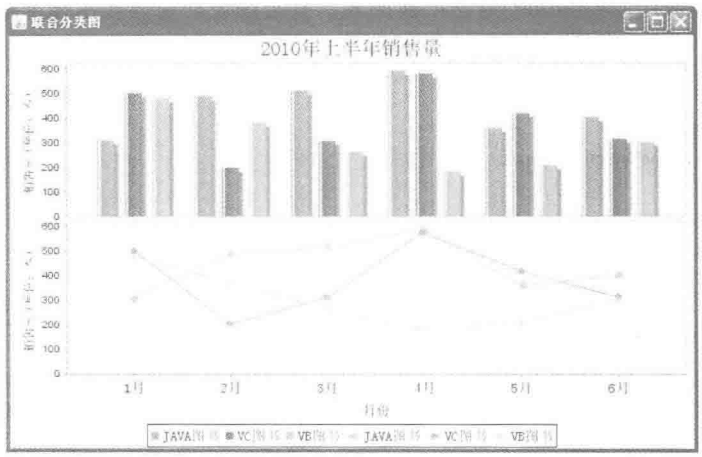


图 9.28　设置图表位置

关键技术

图表的显示顺序与添加 Plot 的实例顺序一致，先添加的 Plot 实例显示在图表上面，后添加的 Plot 实例显示在图表下面，代码如下：

```
final CombinedDomainCategoryPlot plot = new CombinedDomainCategoryPlot(domainAxis);
//设置联合图表
plot.add(plot2, 1);
plot.add(plot1, 1);
```

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码见实例 253。

（3）创建 getJFreeChart()方法，在方法中创建 LineAndShapeRenderer 和 BarRenderer 实例，分别使用这两个实例渲染生成两个 Plot，再把这两个 Plot 添加到 CombinedDomainCategoryPlot 实例中，然后生成 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset();
    //生成线形图渲染
    LineAndShapeRenderer renderer1 = new LineAndShapeRenderer();
    //生成柱形图渲染
    BarRenderer renderer2 = new BarRenderer();
    //设置 X 轴
    CategoryAxis domainAxis = new CategoryAxis("月份");
    //设置 Y 轴
    NumberAxis rangeAxis = new NumberAxis("销售量（单位：本）");
    //设置图表
    CategoryPlot plot1 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer1);
```

```
        CategoryPlot plot2 = new CategoryPlot(dataset, domainAxis, rangeAxis, renderer2);
        //设置联合分类图表
        final CombinedDomainCategoryPlot plot = new CombinedDomainCategoryPlot(domainAxis);
                plot.add(plot2, 1);
                plot.add(plot1, 1);
        JFreeChart chart = new JFreeChart("2010 年上半年销售量", plot);
        return chart;
    }
```

（4）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 253。

（5）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 253。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        CombinedCategoryDemo3 demo = new CombinedCategoryDemo3("联合分类图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 255：对 CategoryPlot 的初始化。

绘制联合分类图时，把多个 CategoryPlot 实例添加到 CombinedDomainCategoryPlot 类中，创建 CategoryPlot 实例时使用的渲染类型决定绘制图表的类型。本实例创建了线形图与柱形图，所以必须使用 LineAndShapeRenderer 和 BarRenderer 类初始化 CategoryPlot。

实例 256　线形图与分布图

光盘位置：光盘\MR\256　　高级　　趣味指数：★★★★

实例说明

在联合分类图中，可以添加多种分类图表。本实例演示如何在联合分类图中绘制线形图与分步图，运行效果如图 9.29 所示。

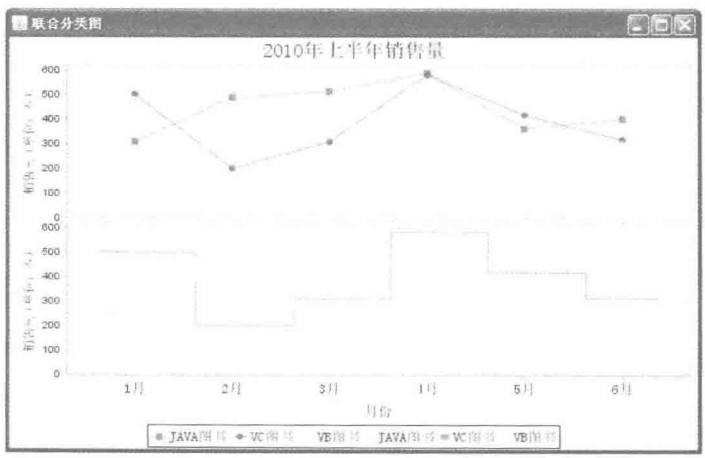


图 9.29　在联合分类图中绘制线形图与分步图

关键技术

使用 LineAndShapeRenderer 类和 CategoryStepRenderer 类的渲染可以创建线形图和分步图，语法如下：

```
//生成线形图渲染
LineAndShapeRenderer renderer1 = new LineAndShapeRenderer();
//生成分步图渲染
CategoryStepRenderer renderer2 = new CategoryStepRenderer();
```

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建数据集，代码见实例 253。

（3）创建 getJFreeChart()方法，在方法中创建 LineAndShapeRenderer 和 CategoryStepRenderer 实例，分别使用这两个渲染生成两个 Plot，再把这两个 Plot 添加到 CombinedDomainCategoryPlot 实例中，然后生成 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        //生成线形图渲染
        LineAndShapeRenderer renderer1 = new LineAndShapeRenderer();
        //生成分步图渲染
        CategoryStepRenderer renderer2 = new CategoryStepRenderer();
        //设置 X 轴
        CategoryAxis domainAxis = new CategoryAxis("月份");
        //设置 Y 轴
        NumberAxis rangeAxis = new NumberAxis("销售量（单位：本）");
        //设置图表
        CategoryPlot plot1 = new CategoryPlot(dataset, domainAxis, rangeAxis,
                        renderer1);
        CategoryPlot plot2 = new CategoryPlot(dataset, domainAxis, rangeAxis,
                        renderer2);
          //设置联合分类图表
          final CombinedDomainCategoryPlot plot = new CombinedDomainCategoryPlot(domainAxis);
          plot.add(plot1);
          plot.add(plot2);
          JFreeChart chart = new JFreeChart("2010 年上半年销售量", plot);
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 253。

（5）创建 createPlot()方法，获取一个 JFreeChart 对象，然后修改字体，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 253。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        CombinedCategoryDemo4 demo = new CombinedCategoryDemo4("联合分类图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 256：通过 add()方法向 CombinedDomainCategoryPlot 类添加多个 Plot 实例。

联合分类图用于展示多图表，如果希望联合分类图中展示多个图表，可以使用 add()方法向 CombinedDomainCategoryPlot 类添加多个 Plot 实例。

9.5 双 轴 图

实例 257	基本双轴图	高级
	光盘位置：光盘\MR\257	趣味指数：★★★★★

实例说明

双轴图一般在一张图表中把多种图表类型混合显示，可以含有两个 Y 轴，如果多个图表的 Y 轴数据一致，也可以把这些 Y 轴合并在一起显示。本实例绘制了线形图和柱形图混合的双轴图，运行效果如图 9.30 所示。

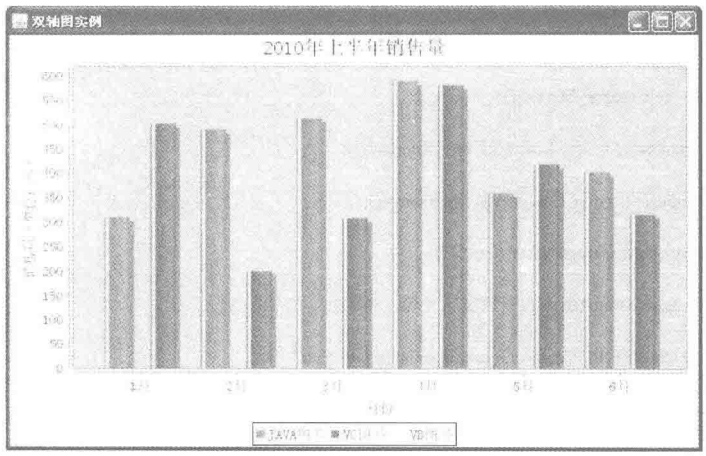


图 9.30 基本双轴图

关键技术

创建双轴图时需要使用两个数据集，使用 CategoryPlot 类的 setDataset()方法可以向 CategoryPlot 类添加数据集，语法如下：

```
setDataset(int index, CategoryDataset dataset)
```

参数说明

❶ index：表示需要添加的数据集索引。

❷ dataset：表示需要添加的分类数据集。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset1()方法，在方法中使用 DefaultCategoryDataset 类创建一个多系列数据集，用于绘制柱形图，代码如下：

```
private CategoryDataset getCategoryDataset1() {
    //行关键字
    final String series1 = "JAVA 图书";
    final String series2 = "VC 图书";
    //列关键字
    final String category1 = "1 月";
    final String category2 = "2 月";
    final String category3 = "3 月";
    final String category4 = "4 月";
```

```
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        return dataset;
    }
```

（3）创建 getCategoryDataset2()方法，在方法中使用 DefaultCategoryDataset 类创建一个单系列数据集，用于绘制线形图，代码如下：

```
    private CategoryDataset getCategoryDataset2() {
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
    }
```

（4）创建 getJFreeChart()方法，在方法中获取柱形图数据集，使用数据集创建一个 JFreeChart 实例，代码如下：

```
    private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset1();
        JFreeChart chart = ChartFactory.createBarChart("2010 年上半年销售量",          //图表标题
                "月份",                                                              //X 轴标签
                "销售量（单位：本）",                                                 //Y 轴标签
                dataset,                                                             //数据集
                PlotOrientation.VERTICAL,                                            //图表方向：垂直
                true,                                                                //显示图例
                false,                                                               //不生成工具栏提示
                false                                                                //不生成 URL 链接
                );
        return chart;
    }
```

（5）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码如下：

```
    private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
```

```
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    }
```

（6）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 实例，并为实例添加线形数据集和 LineAndShapeRenderer 实例，代码如下：

```
    private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        categoryPlot.setDataset(1,getCategoryDataset2());
        //线形渲染
        LineAndShapeRenderer renderer2 = new LineAndShapeRenderer();
        //设置线形图
        categoryPlot.setRenderer(1, renderer2);
    }
```

（7）创建 createPlot()方法，在方法中更新图表字体和内容，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
    public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //修改图表
        updatePlot(chart);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
    }
```

（8）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        DualAxisDemo1 demo = new DualAxisDemo1("双轴图实例");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 257：通过 setRenderer()方法为指定的索引添加渲染。

使用 LineAndShapeRenderer 类的 setRenderer()方法可以为指定的索引添加渲染，语法如下：

```
setRenderer(int index, CategoryItemRenderer renderer)
```

参数说明

❶ index：表示需要添加渲染的索引。

❷ renderer：表示需要添加的渲染实例。

实例 258 基本双 Y 轴图

光盘位置：光盘\MR\258　　中级　　趣味指数：★★★★★

实例说明

双轴图一般在一张图表中把多种图表类型混合显示，可以含有两个 Y 轴。本实例演示如何添加另外一个 Y

轴，运行效果如图 9.31 所示。

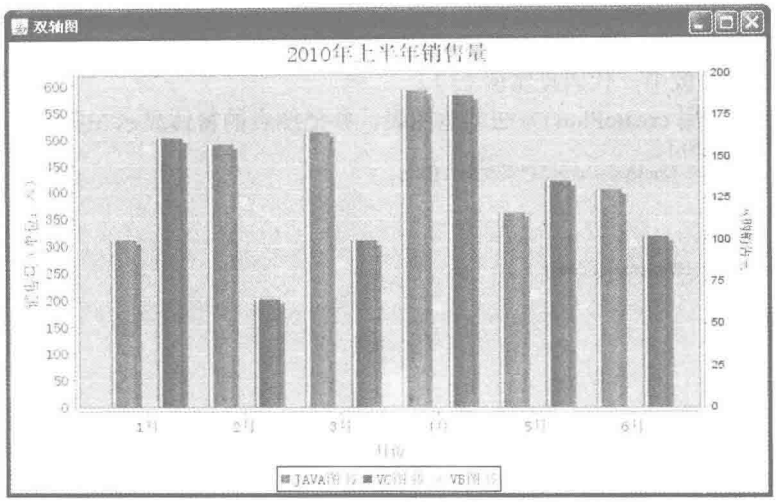


图 9.31　显示双 Y 轴

关键技术

使用 CategoryPlot 类的 setRangeAxis()方法可以向 Plot 实例中新增一个 Y 轴，语法如下：

```
setRangeAxis(int index, ValueAxis axis)
```

参数说明

❶ index：表示需要添加的数据集索引。

❷ axis：表示需要添加的 Y 轴实例。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset1()方法，在方法中使用 DefaultCategoryDataset 类创建一个多系列数据集，用于绘制柱形图，代码见实例 257。

（3）创建 getCategoryDataset2()方法，在方法中使用 DefaultCategoryDataset 类创建一个单系列数据集，用于绘制线形图，代码见实例 257。

（4）创建 getJFreeChart 方法，在方法中获取柱形图数据集，使用数据集创建一个 JFreeChart 实例，代码见实例 257。

（5）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 257。

（6）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 实例，并为 CategoryPlot 添加数据集、渲染和 Y 轴，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        categoryPlot.setDataset(1,getCategoryDataset2());
          LineAndShapeRenderer renderer2 = new LineAndShapeRenderer();
          //设置线形图
          categoryPlot.setRenderer(1, renderer2);
          //设置双 Y 轴
          final ValueAxis axis2 = new NumberAxis("网购销售量");
          //设置 Y 轴最大值
          axis2.setUpperBound(200);
```

```
        categoryPlot.setRangeAxis(1, axis2);
    }
```

（7）创建 createPlot()方法，在方法中更新图表字体和内容，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 257。

（8）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    DualAxisDemo2 demo = new DualAxisDemo2 ("双轴图实例");
    demo.createPlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 258：通过 setUpperBound()方法为数值轴设置最大值。

使用 ValueAxis 类的 setUpperBound()方法可以为数值轴设置最大值，语法如下：

```
setUpperBound(double max)
```

参数说明

max：表示数值轴的最大值。

实例 259 双 Y 轴字体

光盘位置：光盘\MR\259　　中级　　趣味指数：★★★★

实例说明

双轴图一般含有两个 Y 轴，每个数值轴的字体都可以单独设置。本实例演示如何修改新增的 Y 轴标签字体，运行效果如图 9.32 所示。

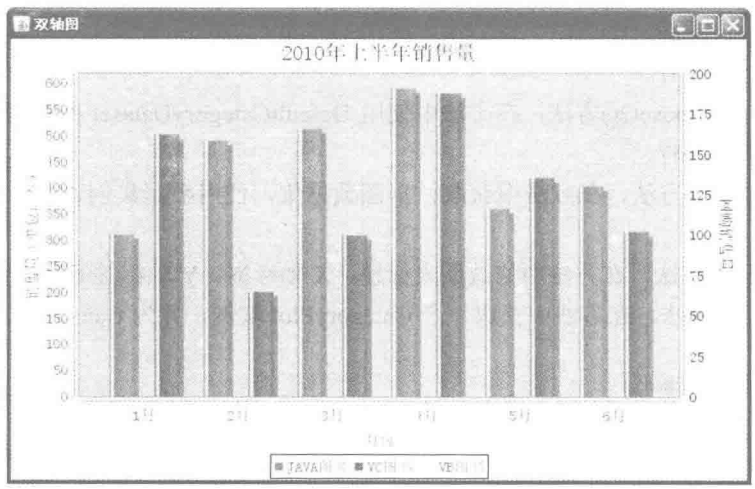


图 9.32　双 Y 轴字体

关键技术

使用 ValueAxis 类的 setLabelFont()方法可以向当前数值轴标签添加字体，语法如下：

setLabelFont(Font font)

参数说明

font：表示当前的数值轴的标签字体。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset1()方法，在方法中使用 DefaultCategoryDataset 类创建一个多系列数据集，用于绘制柱形图，代码见实例 257。

（3）创建 getCategoryDataset2()方法，在方法中使用 DefaultCategoryDataset 类创建一个单系列数据集，用于绘制线形图，代码见实例 257。

（4）创建 getJFreeChart()方法，在方法中获取柱形图数据集，使用数据集创建一个 JFreeChart 实例，代码见实例 257。

（5）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 257。

（6）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 实例，并为 CategoryPlot 添加数据集、渲染和 Y 轴，同时为 Y 轴设置标签字体，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        categoryPlot.setDataset(1,getCategoryDataset2());
          LineAndShapeRenderer renderer2 = new LineAndShapeRenderer();
          //设置线形图
          categoryPlot.setRenderer(1, renderer2);
          final ValueAxis axis2 = new NumberAxis("网购销售量");
          //Y 轴字体
          axis2.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
          //Y 轴标签字体
          axis2.setLabelFont(new Font("宋体", Font.PLAIN, 14));
          //设置最大值
          axis2.setUpperBound(200);
          categoryPlot.setRangeAxis(1, axis2);

}
```

（7）创建 createPlot()方法，在方法中更新图表字体和内容，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 257。

（8）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        DualAxisDemo3 demo = new DualAxisDemo3("双轴图实例");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 259：通过 setTickLabelFont()方法设置数值轴尺度标签字体。

使用 ValueAxis 类的 setTickLabelFont()方法可以设置数值轴尺度标签字体，语法如下：

setTickLabelFont(Font font)

参数说明

font：表示数值轴的尺度标签字体。

实例 260 3D 双轴图

光盘位置：光盘\MR\260

高级

趣味指数：★★★★★

实例说明

使用 3D 双轴图可以让图表更生动。本实例演示如何绘制 3D 双轴图，即将 3D 柱形图和线形图混合显示，运行效果如图 9.33 所示。

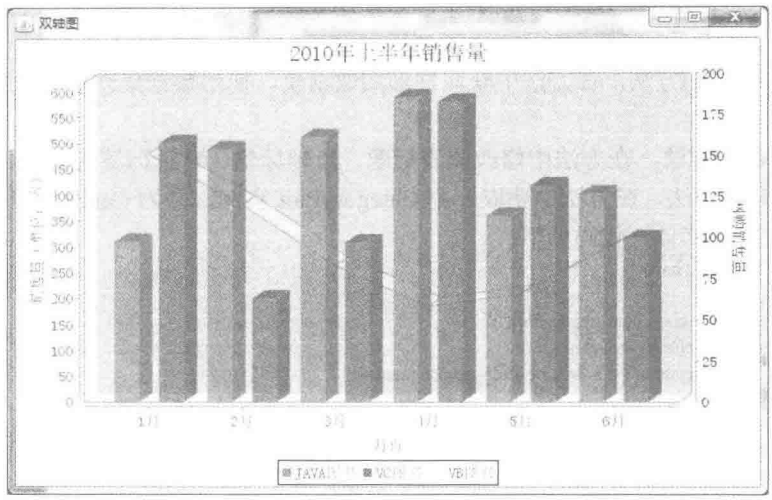


图 9.33 3D 双轴图

关键技术

使用 ChartFactory 类的 createBarChart3D()方法可以绘制 3D 柱形图，创建完成以后会返回一个 JFreeChart 对象，语法如下：

createBarChart3D(String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)

参数说明

❶ title：图表的标题。

❷ categoryAxisLabel：图表类别标签，即 X 轴的名称。

❸ valueAxisLabel：图表数据标签，即 Y 轴的名称。

❹ dataset：区域图的数据集合。

❺ orientation：图表的显示方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset1()方法，在方法中使用 DefaultCategoryDataset 类创建一个多系列数据集，用于绘制柱形图，代码见实例 257。

（3）创建 getCategoryDataset2()方法，在方法中使用 DefaultCategoryDataset 类创建一个单系列数据集，用于绘制线形图，代码见实例 257。

（4）创建 getJFreeChart()方法，在方法中获取柱形图数据集，使用数据集创建一个 JFreeChart 实例，代码如下：

```
private JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset1();
    JFreeChart chart = ChartFactory.createBarChart3D("2010 年上半年销售量",      //图表标题
            "月份",                                                              //X 轴标签
            "销售量（单位：本）",                                                //Y 轴标签
            dataset,                                                             //数据集
            PlotOrientation.VERTICAL,                                            //图表方向：垂直
            true,                                                                //是否显示图例
            false,                                                               //是否生成工具
            false                                                                //是否生成 URL 链接
            );
    return chart;
}
```

（5）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 257。

（6）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 实例，并且为 CategoryPlot 添加数据集、渲染和 Y 轴，代码见实例 259。

（7）创建 createPlot()方法，在方法中更新图表字体和图表内容，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 257。

（8）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    DualAxisDemo4  demo = new DualAxisDemo4("双轴图");
    demo.createPlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 260：渲染线形图实现 3D 效果。

本实例实现 3D 双轴图，线形图使用 LineAndShapeRenderer 类进行渲染并没有实现 3D 效果，如果使用 LineRenderer3D 类渲染线形图，那么线形图也可以实现 3D 效果。

实例 261　设置双轴图颜色

光盘位置：光盘\MR\261　　中级　　趣味指数：★★★★

实例说明

双轴图由多种图形混合组成，各个图的颜色都要分别设置。本实例绘制了柱形图与线形图，并且演示如何将线形图设置为黑色，运行效果如图 9.34 所示。

关键技术

使用 LineAndShapeRenderer 类的 setSeriesPaint()方法可以为线形图系列指定设置颜色，语法如下：

```
setSeriesPaint(int series, Paint paint)
```

参数说明

❶ series：表示指定图表的系列。

❷ paint：表示指定图表系列的颜色。

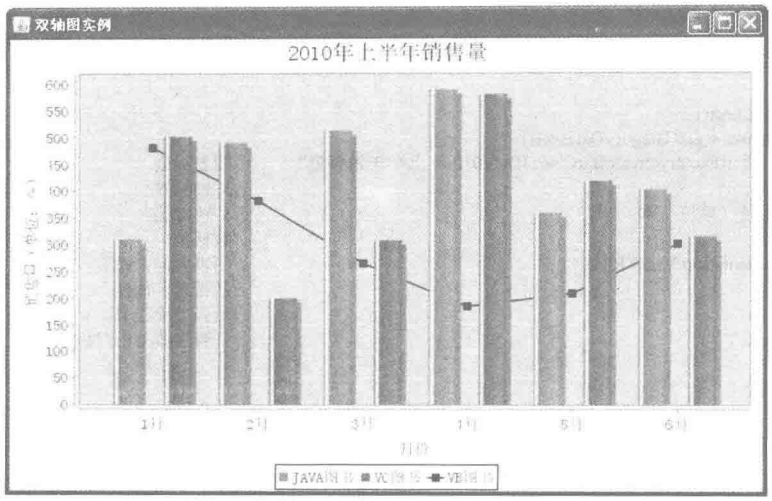


图 9.34　双轴图颜色

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset1()方法，在方法中使用 DefaultCategoryDataset 类创建一个多系列数据集，用于绘制柱形图，代码见实例 257。

（3）创建 getCategoryDataset2()方法，在方法中使用 DefaultCategoryDataset 类创建一个单系列数据集，用于绘制线形图，代码见实例 257。

（4）创建 getJFreeChart()方法，在方法中获取柱形图数据集，使用数据集创建一个 JFreeChart 实例，代码见实例 257。

（5）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 257。

（6）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 实例，并且为 CategoryPlot 添加数据集、渲染和 Y 轴，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        categoryPlot.setDataset(1,getCategoryDataset2());
        //线形渲染
          LineAndShapeRenderer renderer2 = new LineAndShapeRenderer();
          //设置线形图颜色
          renderer2.setSeriesPaint(0, Color.black);
          //设置线形图
          categoryPlot.setRenderer(1, renderer2);
}
```

（7）创建 createPlot()方法，在方法中更新图表字体和内容，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 257。

（8）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        DualAxisDemo5 demo = new DualAxisDemo5 ("双轴图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
```

```
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 261：通过 getRendererCount()方法获取双轴图中渲染的总数。

双轴图由多种图形混合组成，使用 CategoryPlot 类的 getRendererCount()方法可以获取双轴图中渲染的总数，返回值为 int 类型，语法如下：

```
getRendererCount()
```

实例 262　双轴图 X 轴标签角度

光盘位置：光盘\MR\262　　中级　　趣味指数：★★★

实例说明

双轴图中 X 轴的标签可以根据实例进行角度设置。本实例演示如何设置图表的 X 轴标签角度，运行效果如图 9.35 所示。

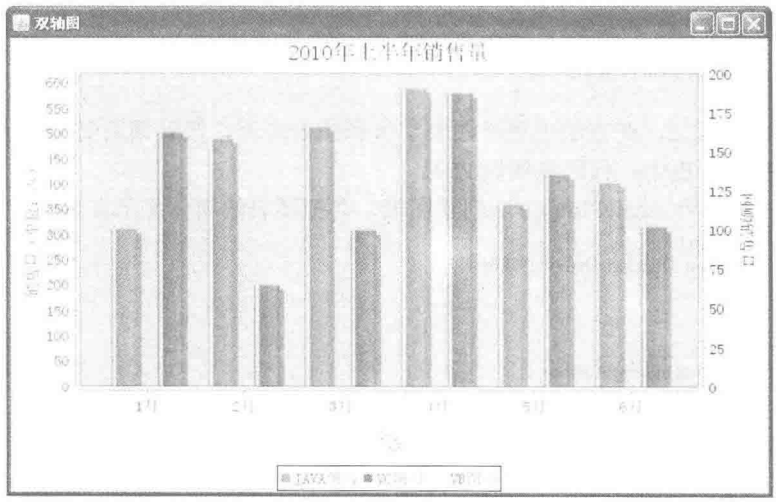


图 9.35　双轴图 X 轴标签角度

关键技术

使用 CategoryAxis 类的 setLabelAngle()方法可以为 X 轴标签设置显示角度，语法如下：

```
setLabelAngle(double angle)
```

参数说明

angle：指定图表的 X 轴标签旋转弧度，X 轴标签的角度根据此弧度参数进行计算。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset1()方法，在方法中使用 DefaultCategoryDataset 类创建一个多系列数据集，用于绘制柱形图，代码见实例 257。

（3）创建 getCategoryDataset2()方法，在方法中使用 DefaultCategoryDataset 类创建一个单系列数据集，用于绘制线形图，代码见实例 257。

（4）创建 getJFreeChart()方法，在方法中获取柱形图数据集，使用数据集创建一个 JFreeChart 实例，代码见实例 257。

（5）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 257。

（6）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 实例，并且为 CategoryPlot 添加数据集、渲染和 Y 轴，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        categoryPlot.setDataset(1,getCategoryDataset2());
          LineAndShapeRenderer renderer2 = new LineAndShapeRenderer();
          //设置线形图
          categoryPlot.setRenderer(1, renderer2);
          final ValueAxis axis2 = new NumberAxis("网购销售量");
          //Y 轴字体
          axis2.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
          //Y 轴标签字体
          axis2.setLabelFont(new Font("宋体", Font.PLAIN, 14));
          //设置最大值
          axis2.setUpperBound(200);
          categoryPlot.setRangeAxis(1, axis2);
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //双轴图 X 轴标签角度
        categoryAxis.setLabelAngle(Math.PI*0.3);
}
```

（7）创建 createPlot()方法，在方法中更新图表字体和图表内容，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 257。

（8）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        DualAxisDemo6 demo = new DualAxisDemo6 ("双轴图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 262：通过 getRangeAxis()方法获取 Y 轴实例。

双轴图一般含有两个 Y 轴，使用 getRangeAxis()方法可以获取 Y 轴实例，语法如下：

```
getRangeAxis(int index)
```

参数说明

index：表示获取 Y 轴的实例。通过 Y 轴的实例设置标签的旋转角度。

实例 263　双轴图 X 轴尺度标签角度

中级

光盘位置：光盘\MR\263　　趣味指数：★★★

实例说明

显示双轴图中 X 轴的尺度标签时，可以为尺度标签设置角度。本实例演示如何设置双轴图 X 轴尺度标签的角度，运行效果如图 9.36 所示。

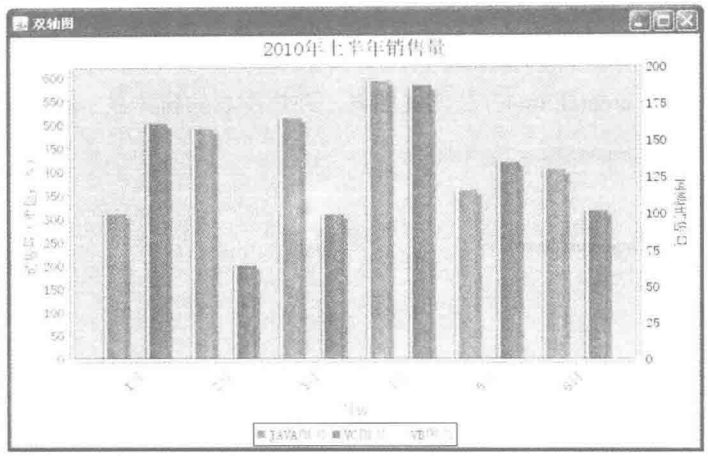


图 9.36　双轴图 X 轴尺度标签角度

关键技术

使用 CategoryAxis 类的 setCategoryLabelPositions()方法可以设置 X 轴的尺度标签角度，语法如下：

```
setCategoryLabelPositions(CategoryLabelPositions positions)
```

参数说明

positions：表示 X 轴尺度标签的角度。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset1()方法，在方法中使用 DefaultCategoryDataset 类创建一个多系列数据集，用于绘制柱形图，代码见实例 257。

（3）创建 getCategoryDataset2()方法，在方法中使用 DefaultCategoryDataset 类创建一个单系列数据集，用于绘制线形图，代码见实例 257。

（4）创建 getJFreeChart()方法，在方法中获取柱形图数据集，使用数据集创建一个 JFreeChart 实例，代码见实例 257。

（5）创建 updateFont()方法，在方法中修改图表标题、X 轴标签、Y 轴标签等字体，代码见实例 257。

（6）创建 updatePlot()方法，在方法中获取一个 CategoryPlot 实例，并且为 CategoryPlot 添加数据集、渲染和 Y 轴，然后设置 X 轴的尺度标签角度，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        categoryPlot.setDataset(1,getCategoryDataset2());
           LineAndShapeRenderer renderer2 = new LineAndShapeRenderer();
           //设置线形图
           categoryPlot.setRenderer(1, renderer2);
           final ValueAxis axis2 = new NumberAxis("网购销售量");
           //Y 轴字体
           axis2.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
           //Y 轴标签字体
           axis2.setLabelFont(new Font("宋体", Font.PLAIN, 14));
           //设置最大值
           axis2.setUpperBound(200);
           categoryPlot.setRangeAxis(1, axis2);
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //双轴图 X 轴尺度标签角度
        categoryAxis.setCategoryLabelPositions(CategoryLabelPositions.UP_45);
}
```

（7）创建 createPlot()方法，在方法中更新图表字体和图表内容，接着调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 257。

（8）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        DualAxisDemo7 demo = new DualAxisDemo7 ("双轴图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 263：通过 setVerticalTickLabels()方法设置 Y 轴尺度标签的角度。

使用 ValueAxis 类的 setVerticalTickLabels()方法可以设置 Y 轴尺度标签的角度，语法如下：

```
setVerticalTickLabels(boolean flag)
```

参数说明

flag：用于设置 Y 轴的尺度标签角度，当 flag 为 true 时，尺度标签为垂直角度；当 flag 为 false 时，尺度标签为水平角度。

9.6 折 线 图

实例 264 基本折线图　　高级

光盘位置：光盘\MR\264　　趣味指数：★★★★

实例说明

折线图通过数据的线形走势来体现情况的变化趋势，X 轴表示分类情况，Y 轴表示具体数值。本实例演示如何创建一个基本的折线图，运行效果如图 9.37 所示。

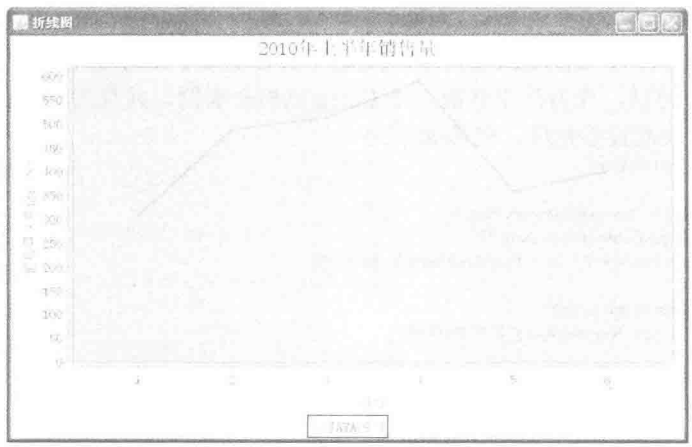


图 9.37 基本折线图

关键技术

使用 ChartFactory 类的 createLineChart()方法可创建基本折线图，创建完成以后会返回一个 JFreeChart 对象，

语法如下：

createLineChart (String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)

参数说明

❶ title：图表的标题。

❷ categoryAxisLabel：图表类别标签，即 X 轴的名称。

❸ valueAxisLabel：图表数据标签，即 Y 轴的名称。

❹ dataset：折线图的数据集合。

❺ orientation：图表的显示方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultKeyedValues 类的实例中添加数据内容，然后通过 DatasetUtilities 类的 createCategoryDataset()方法创建一个数据集，代码如下：

```
private CategoryDataset getCategoryDataset() {
    DefaultKeyedValues keyedValues = new DefaultKeyedValues();
    keyedValues.addValue("1", 310);
    keyedValues.addValue("2", 489);
    keyedValues.addValue("3", 512);
    keyedValues.addValue("4", 589);
    keyedValues.addValue("5", 359);
    keyedValues.addValue("6", 402);
    CategoryDataset dataset = DatasetUtilities.createCategoryDataset("JAVA 图书", keyedValues);
    return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset();
    JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",      //图表标题
    "月份",                                                                    //X 轴标签
    "销售量（单位：本）",                                                      //Y 轴标签
    dataset,                                                                   //数据集
    PlotOrientation.VERTICAL,                                                  //图表方向：垂直
    true,                                                                      //显示图例
    false,                                                                     //不生成工具栏提示
    false                                                                      //不生成 URL 链接
    );
    return chart;
}
```

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码如下：

```
private void updateFont(JFreeChart chart) {
    //标题
    TextTitle textTitle = chart.getTitle();
    textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
    LegendTitle legendTitle = chart.getLegend();
    legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
    //图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
    //X 轴字体
    categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
    //X 轴标签字体
    categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
```

```
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    }
```

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后调用 updateFont()方法修改字体，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
    public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        setContentPane(new ChartPanel(chart));
    }
```

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        LineDemo1 demo = new LineDemo1("折线图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 264：在折线图中设置 X 轴、Y 轴的字体。

折线图的 X 轴、Y 轴字体使用不同的类进行设置，X 轴使用 CategoryAxis 类；Y 轴使用 ValueAxis 类，二者都可以使用 setLabelFont()方法修改标签字体。

实例 265　多条折线图

光盘位置：光盘\MR\265　　中级　　趣味指数：★★★★★

实例说明

折线图支持多系列图表，通过折线的走势与不同折线的比较体现情况的变化趋势。本实例演示如何创建多条折线图，运行效果如图 9.38 所示。

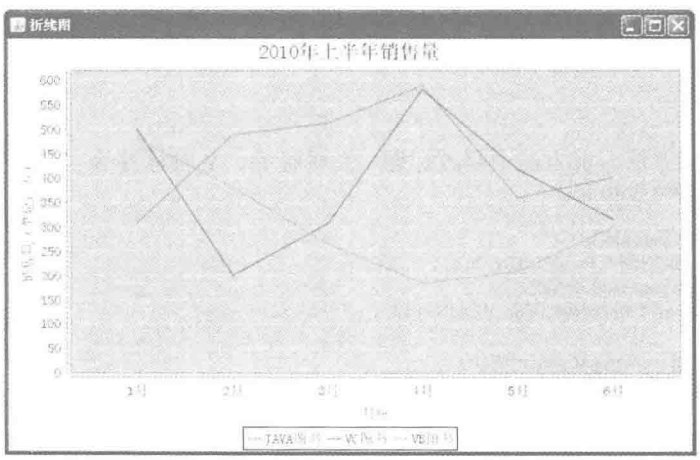


图 9.38　多条折线图

关键技术

使用 DefaultCategoryDataset 类的 addValue()方法可以创建多系列数据集，使用多系列数据集可以创建多条折线图，语法如下：

addValue(double value, Comparable rowKey, Comparable columnKey)

参数说明

❶ value：表示具体数据值。

❷ rowKey：表示 X 轴系列名称。

❸ columnKey：表示 X 轴分类名称。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，代码如下：

```
private CategoryDataset getCategoryDataset() {
        //系列关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //分类关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 264。

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后调用 updateFont()方法修改字体，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 264。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        LineDemo2 demo = new LineDemo2("折线图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
```

```
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 265：通过 DefaultKeyedValues 类创建基本线形图。

DefaultKeyedValues 类只能创建基本线形图，因为在该类中无法添加多系列数据；使用 DefaultCategoryDataset 类可以创建多折线图，使用 DefaultCategoryDataset 类可以创建多系列数据集。

实例 266 水平折线图

光盘位置：光盘\MR\266　　高级　　趣味指数：★★★★

实例说明

折线图可以绘制成垂直样式和水平样式。本实例绘制水平样式的折线图，其 X 轴显示在图表左侧，Y 轴显示在图表上部，运行效果如图 9.39 所示。

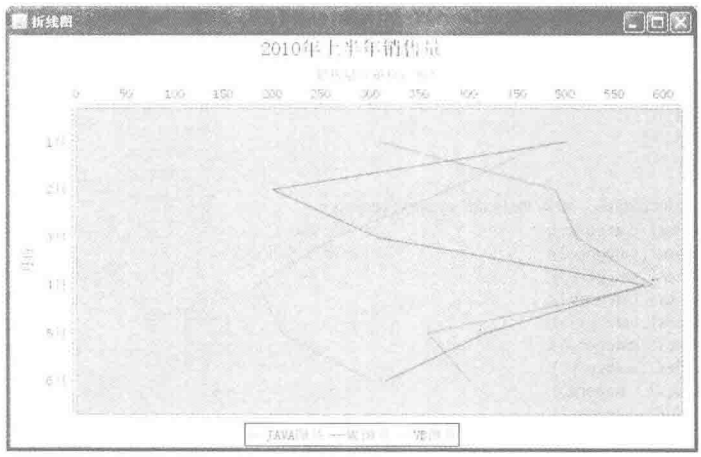


图 9.39　水平折线图

关键技术

使用 PlotOrientation 类可以设置折线图的显示状态，PlotOrientation 类有两个常量：PlotOrientation.HORIZONTAL 表示折线图显示为水平状态；PlotOrientation.VERTICAL 表示折线图显示为垂直状态。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，代码见实例 265。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createLineChart("2010 年上半年销售量",      //图表标题
        "月份",                                                                    //X 轴标签
        "销售量（单位：本）",                                                      //Y 轴标签
        dataset,                                                                   //数据集
        PlotOrientation.HORIZONTAL,                                                //图表方向：水平
```

```
            true,                                                    //显示图例
            false,                                                   //不生成工具栏提示
            false                                                    //不生成 URL 链接
            );
            return chart;
    }
```

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后调用 updateFont()方法修改字体，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 264。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        LineDemo3 demo = new LineDemo3("折线图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 266：通过 setDomainAxisLocation()方法设置折线图 X 轴的位置。

使用 CategoryPlot 类的 setDomainAxisLocation()方法可以设置折线图 X 轴的位置，语法如下：

```
setDomainAxisLocation(AxisLocation location)
```

参数说明

location：表示折线图 X 轴的方位，使用参数 AxisLocation.TOP_OR_RIGHT 表示设置 X 轴显示在图表顶部或右侧；使用参数 AxisLocation.BOTTOM_OR_RIGHT，表示设置 X 轴显示在图表底部或右侧。

实例 267　隐藏折线图

光盘位置：光盘\MR\267　　中级　　趣味指数：★★★

实例说明

绘制多条折线图时，在数据集中可能有一部分系列不需要显示在图表中。本实例演示如何隐藏指定折线图，运行效果如图 9.40 所示。

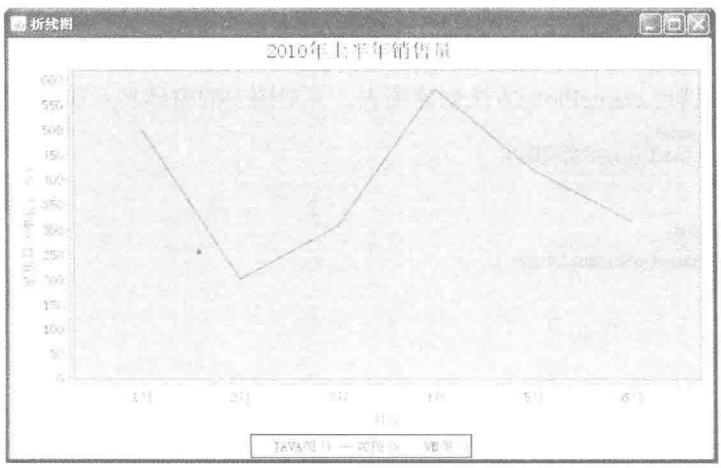


图 9.40　隐藏折线图

关键技术

使用 LineAndShapeRenderer 类的 setSeriesLinesVisible()方法，可以隐藏折线图中指定系列的折线，语法如下：

```
setSeriesLinesVisible(int series, boolean visible)
```

参数说明

❶ series：表示折线图中的系列。

❷ visible：表示折线图中指定系列是否显示，当 visible 为 true 时，则显示折线；当 visible 为 false 时，则隐藏折线。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，代码见实例 265。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 264。

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 类，根据 CategoryPlot 获取 LineAndShapeRenderer 实例，然后隐藏第一个系列的折线，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryPlot.getRenderer();
        //隐藏指定线形
        renderer.setSeriesLinesVisible(0, false);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //修改图表
        updatePlot(chart);
        setContentPane(new ChartPanel(chart));
}
```

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        LineDemo4 demo = new LineDemo4("折线图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 267：通过 setSeriesLinesVisible()方法隐藏折线图。

绘制折线图时，如果使用 setSeriesLinesVisible()方法隐藏折线图，折线则不会在图表中显示，同时当前系列的图示也在图表中隐藏。

实例 268　加粗折线

光盘位置：光盘\MR\268　　中级　　趣味指数：★★★★

实例说明

折线图中的折线图形笔触默认情况下比较细，用户可以根据需要加粗折线的笔触。本实例演示如何加粗折线，运行效果如图 9.41 所示。

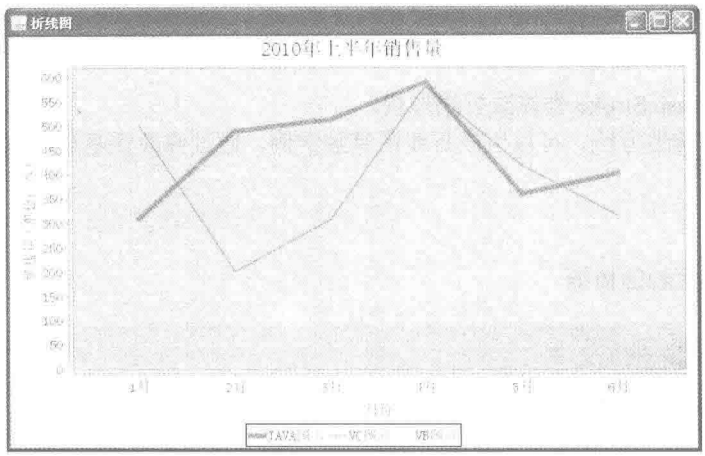


图 9.41　加粗折线

关键技术

通过修改折线图的笔触可以重新绘制折线图形，使用 LineAndShapeRenderer 类的 setSeriesStroke()方法可以设置折线图的笔触，语法如下：

```
setSeriesStroke(int series, Stroke stroke)
```

参数说明

❶ series：表示折线图中的系列。

❷ stroke：表示需要修改的折线图的笔触。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，代码见实例 265。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 264。

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 updatePlot()方法，在方法中获取 CategoryPlot 类，根据 CategoryPlot 获取 LineAndShapeRenderer 实例，然后加粗第一个系列的折线，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //分类图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryPlot.getRenderer();
    //加粗线形
    renderer.setSeriesStroke(0,new BasicStroke(5));
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 267。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        LineDemo5 demo = new LineDemo5("折线图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 268：通过 BasicStroke 类修改折线形状。

使用 BasicStroke 类的构造方法，可以生成折线图笔触实例，通过修改该笔触，可以修改折线的形状，语法如下：

```
BasicStroke(float width)
```

参数说明

width：表示折线图中笔触的粗细。

实例 269 显示折线节点

光盘位置：光盘\MR\269　　中级　　趣味指数：★★★★

实例说明

在绘制折线图时，可以为折线图生成节点，把折线图的关键节点体现在图表上。本实例演示如何显示折线节点，运行效果如图 9.42 所示。

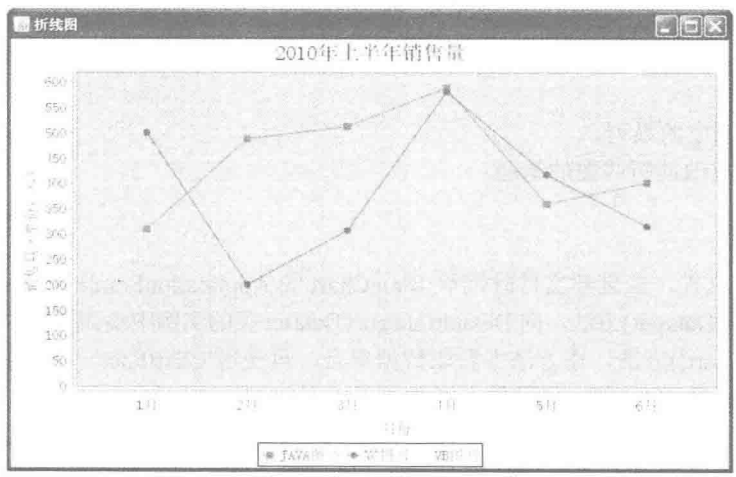


图 9.42　显示折线节点

关键技术

使用 LineAndShapeRenderer 类的 setSeriesShapesVisible()方法可以设置是否显示折线图的节点，语法如下：

```
setSeriesShapesVisible(int series, boolean visible)
```

参数说明

❶ series：表示折线图中的系列。

❷ visible：表示是否显示节点，当 visible 为 true 时，则显示节点；当 visible 为 false 时，则不显示节点。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，代码见实例 265。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 264。

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 updatePlot()方法，在方法中获取 LineAndShapeRenderer 实例，通过 LineAndShapeRenderer 实例显示折线图节点，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryPlot.getRenderer();
        //显示节点
        renderer.setSeriesShapesVisible(0, true);
        renderer.setSeriesShapesVisible(1, true);
        renderer.setSeriesShapesVisible(2, true);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 267。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        LineDemo6 demo = new LineDemo6("折线图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 269：折线图节点特点。

显示折线图的节点时，不但每条折线图颜色不同，各个折线图的节点样式也不一样，这是为了在查看折线图时，可以更明显地区分每条折线和折线之间的节点。

实例 270　生成节点图

光盘位置：光盘\MR\270　　高级　　趣味指数：★★★

实例说明

折线图中，折线和节点的显示与隐藏可以根据需要自由控制。本实例把折线隐藏起来，同时显示折线节点，这时图表中只显示出节点之间的对比情况，运行效果如图 9.43 所示。

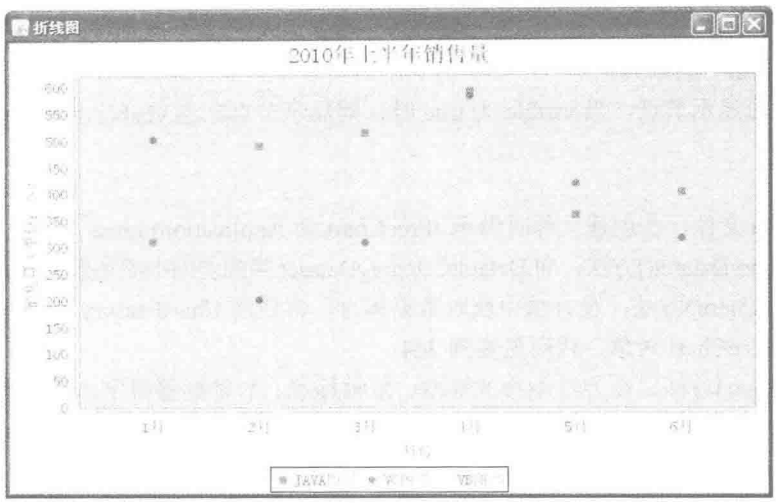


图 9.43　只显示节点

关键技术

使用 LineAndShapeRenderer 类的 setSeriesLinesVisible()方法可以设置是否显示折线图中的折线，语法如下：

```
setSeriesLinesVisible(int series, boolean visible)
```

参数说明

❶ series：表示折线图中的系列。

❷ visible：表示是否显示折线，当 visible 为 true 时，则显示折线；当 visible 为 false 时，则不显示折线。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，代码见实例 265。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 264。

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 updatePlot()方法，在方法中获取 LineAndShapeRenderer 实例，通过 LineAndShapeRenderer 实例隐藏折线，然后显示折线的节点，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //分类图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryPlot.getRenderer();
    //显示节点
    renderer.setSeriesLinesVisible(0, false);
    renderer.setSeriesLinesVisible(1, false);
    renderer.setSeriesLinesVisible(2, false);
    renderer.setSeriesShapesVisible(0, true);
    renderer.setSeriesShapesVisible(1, true);
    renderer.setSeriesShapesVisible(2, true);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 267。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    LineDemo7 demo = new LineDemo7("折线图");
    demo.createPlot();
```

```
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 270：折线图节点与连线的设置。

本实例使用 LineAndShapeRenderer 类的 setSeriesLinesVisible()方法把折线图的全部折线隐藏起来，同时使用 LineAndShapeRenderer 类的 setSeriesShapesVisible()方法显示出所有折线图的节点，这样图表中就只能展示各个节点之间的对比情况了。

实例 271　设置虚线图

光盘位置：光盘\MR\271　　中级　　趣味指数：★★★★

实例说明

折线图默认的笔触是实线，可以根据实际需要设置不同的折线图笔触。本实例演示如何绘制虚线折线图，运行效果如图 9.44 所示。

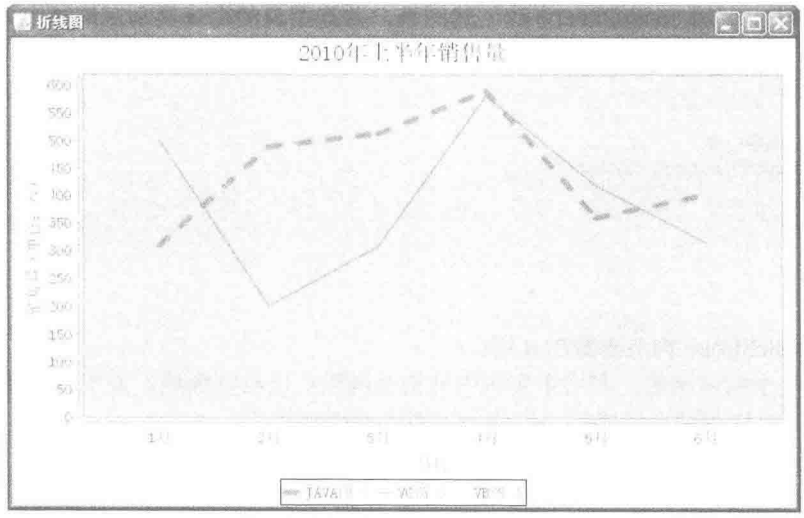


图 9.44　虚线折线图

关键技术

BasicStroke 类有多个构造方法，使用 BasicStroke 类的构造方法可以修改折线图图线的笔触样式，语法如下：

```
BasicStroke(float width, int cap, int join, float miterlimit, float dash[], float dash_phase)
```

参数说明

❶ width：表示笔触宽度。

❷ cap：表示笔触线条的端点样式。

❸ join：表示应用在路径线段交汇处的样式。

❹ miterlimit：表示斜接处的剪裁限制，其值必须大于或等于 1.0f。

❺ dash：表示虚线模式的数组。

❻ dash_phase：表示开始虚线模式的偏移量。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，代码见实例 265。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 264。

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 updatePlot()方法，在方法中获取 LineAndShapeRenderer 实例，通过 LineAndShapeRenderer 实例绘制折线为虚线，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //分类图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
    LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryPlot.getRenderer();
    //虚线
    renderer.setSeriesStroke(0, new BasicStroke(5, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND, 1, new float[] { 10.0f, 16.0f }, 0.0f));
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 267。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    LineDemo8 demo = new LineDemo8("折线图");
    demo.createPlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 271：BasicStroke 构造函数的使用。

BasicStroke 类有几个构造函数，利用本实例中的构造函数可以绘制虚线，如果只需要修改折线图的一些其他样式，例如加粗笔触、修改端点样式等，可使用下面的构造函数：

```
BasicStroke(float width, int cap, int join)
```

参数说明

❶ width：表示笔触宽度。

❷ cap：表示笔触线条的端点样式。

❸ join：表示应用在路径线段交汇处的样式。

实例 272 设置折线颜色

光盘位置：光盘\MR\272　　中级　　趣味指数：★★★★

实例说明

折线图的颜色很丰富，用户可以根据实际情况修改折线颜色。本实例演示如何修改折线图的默认颜色，运行效果如图 9.45 所示。

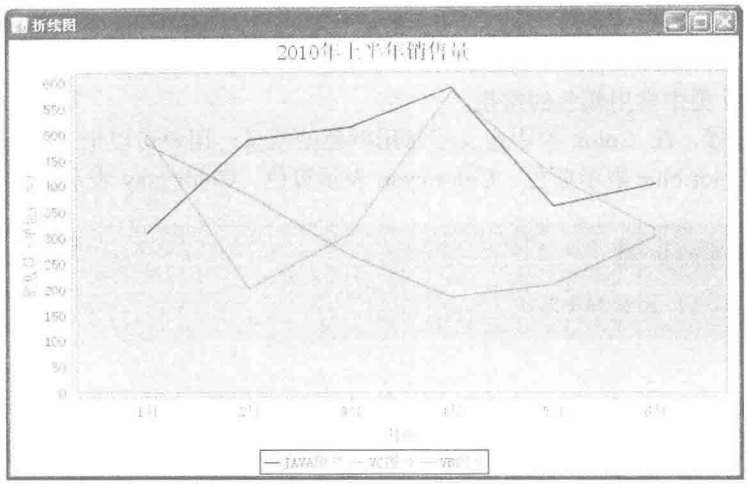


图 9.45　设置折线颜色

关键技术

使用 LineAndShapeRenderer 类的 setSeriesPaint()方法可以修改折线图的颜色，语法如下：

```
setSeriesPaint(int series, Paint paint)
```

参数说明

❶ series：表示折线图当前的系列索引。

❷ paint：表示准备设置系列的颜色。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，代码见实例 265。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createLineChart()方法根据数据集合生成一个 JFreeChart 对象，代码见实例 264。

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 updatePlot()方法，在方法中获取 LineAndShapeRenderer 实例，通过 LineAndShapeRenderer 实例设置第一个索引的折线为黑色，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        LineAndShapeRenderer renderer = (LineAndShapeRenderer) chart.getCategoryPlot().getRenderer();
        //设置显示颜色
        renderer.setSeriesPaint(0, Color.black);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 267。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        LineDemo9 demo = new LineDemo9("折线图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 272：Color 类中常用颜色的常量。

Color 类用于颜色设置，在 Color 类中定义了常用颜色的常量，用户可以把 Color 常量用于折线图中，如 Color.black 表示黑色、Color.blue 表示蓝色、Color.cyan 表示青色、Color.gray 表示灰色等。

实例 273　3D 折线图

光盘位置：光盘\MR\273　　高级　　趣味指数：★★★★★

实例说明

利用 JFreeChart 不但可以绘制普通折线图，还可以绘制 3D 折线图，从而让图表更有立体感。本实例演示如何绘制 3D 折线图，运行效果如图 9.46 所示。

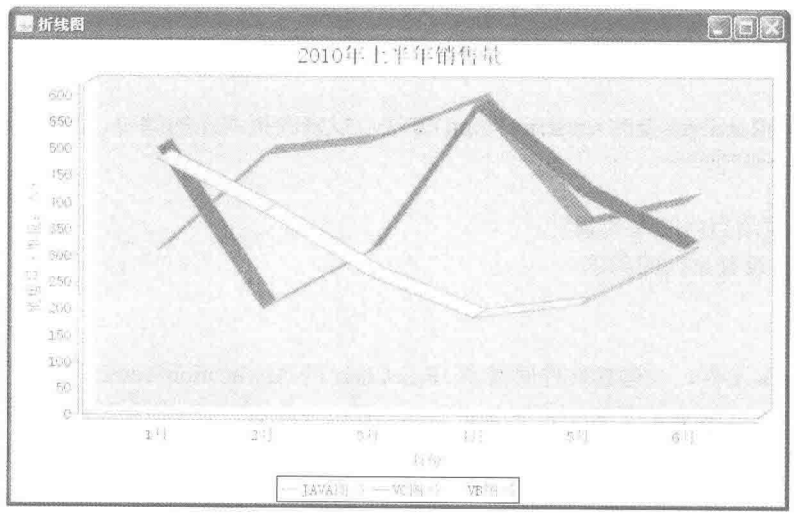


图 9.46　3D 折线图

关键技术

使用 ChartFactory 类的 createLineChart3D()方法可以创建 3D 折线图，创建完成后会返回一个 JFreeChart 对象，语法如下：

createLineChart3D (String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)

参数说明

❶ title：图表的标题。

❷ categoryAxisLabel：图表类别标签，即 X 轴的名称。

❸ valueAxisLabel：图表数据标签，即 Y 轴的名称。

❹ dataset：折线图的数据集合。

❺ orientation：图表的显示方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，向 DefaultCategoryDataset 类的实例中添加数据内容，代码见实例 265。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的 createLineChart3D()方法根据数据集合生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory. createLineChart3D ("2010 年上半年销售量",    //图表标题
        "月份",                                                                    //X 轴标签
        "销售量（单位：本）",                                                      //Y 轴标签
        dataset,                                                                   //数据集
        PlotOrientation.VERTICAL,                                                  //图表方向：垂直
        true,                                                                      //显示图例
        false,                                                                     //不生成工具栏提示
        false                                                                      //不生成 URL 链接
        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 267。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        LineDemo10 demo = new LineDemo10("折线图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 273：通过 setWallPaint()方法设置 3D 折线图的颜色。

在 3D 折线图中，可以使用 LineRenderer3D 类的 setWallPaint()方法设置 3D 折线图墙的颜色，语法如下：

```
setWallPaint(Paint paint)
```

参数说明

paint：表示 3D 折线图墙的颜色。

实例 274　XY 折线图

光盘位置：光盘\MR\274　　高级　　趣味指数：★★★★★

实例说明

利用 JFreeChart 可以绘制多种折线图，如分类折线图、XY 折线图等。本实例演示如何绘制 XY 折线图，并通过 XY 折线图体现数据的变化情况，运行效果如图 9.47 所示。

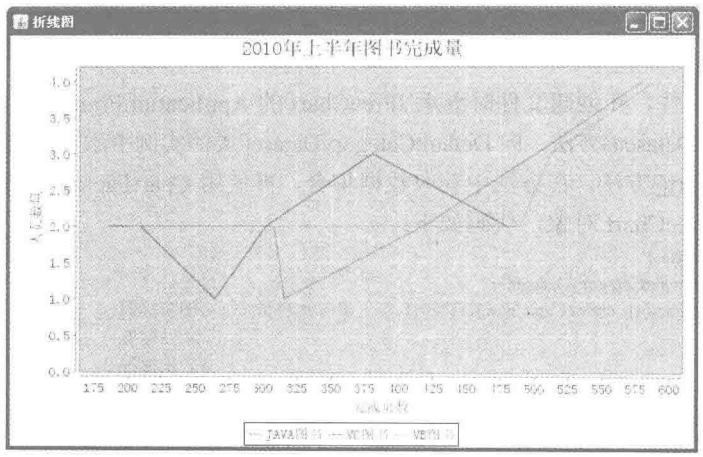


图 9.47　XY 折线图

关键技术

ChartFactory 类的 createXYLineChart()方法提供了创建基本 XY 折线图的方法，创建完成以后会返回一个 JFreeChart 对象，语法如下：

```
createXYLineChart(String title, String xAxisLabel, String yAxisLabel, XYDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)
```

参数说明

❶ title：图表的标题。

❷ xAxisLabel：图表 X 轴的标签内容。

❸ yAxisLabel：图表 Y 轴的标签内容。

❹ dataset：XY 折线图的数据集合。

❺ orientation：图表的显示方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，使用 XYSeries 类向 XYSeriesCollection 类的实例中添加数据内容，代码如下：

```
private IntervalXYDataset getDataset() {
        final XYSeries series1 = new XYSeries("JAVA 图书");
        final XYSeries series2 = new XYSeries("VC 图书");
        final XYSeries series3 = new XYSeries("VB 图书");
        series1.add(501, 3);
        series1.add(200, 2);
        series1.add(308, 2);
        series1.add(580, 4);
        series1.add(418, 2);
        series1.add(315, 1);
        series2.add(480, 2);
        series2.add(381, 3);
        series2.add(264, 1);
        series2.add(185, 2);
        series2.add(209, 2);
        series2.add(302, 2);
        series3.add(310, 2);
        series3.add(489, 2);
```

```
        series3.add(512, 3);
        series3.add(589, 4);
        series3.add(359, 2);
        series3.add(402, 2);
        final XYSeriesCollection dataset = new XYSeriesCollection();
        dataset.addSeries(series1);
        dataset.addSeries(series2);
        dataset.addSeries(series3);
        return dataset;
    }
```

（3）创建 getJFreeChart()方法，在方法中获取 IntervalXYDataset 数据集合，再使用 ChartFactory 类的 createXYLineChart()方法根据数据集合生成一个 JFreeChart 对象，代码如下：

```
    private JFreeChart getJFreeChart() {
        IntervalXYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createXYLineChart("2010 年上半年图书完成量",      //图表标题
        "完成页数",                                                                    //X 轴标签
        "人员数量",                                                                    //Y 轴标签
        dataset,                                                                       //数据集
        PlotOrientation.VERTICAL,                                                      //图表方向：垂直
        true,                                                                          //显示图例
        false,                                                                         //不生成工具栏提示
        false                                                                          //不生成 URL 链接
        );
        return chart;
    }
```

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，再修改字体，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 264。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        LineDemo11 demo = new LineDemo11("折线图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 274：通过 add()方法向 XYSeries 实例中添加 XY 折线图数据。

使用 XYSeries 类的 add()方法向 XYSeries 实例中添加 XY 折线图数据，再向 XYSeries 实例中添加数据集，语法如下：

```
add(double x, double y)
```

参数说明

❶ x：表示 X 轴对应的数据值。

❷ y：表示 Y 轴对应的数据值。

实例 275　排序折线图

光盘位置：光盘\MR\275　　中级　　趣味指数：★★★

实例说明

在绘制折线图时，首先要准备数据集填充折线图表，很多时候需要对数据集进行排序显示。本实例演示如何对 X 轴进行升序排列，运行效果如图 9.48 所示。

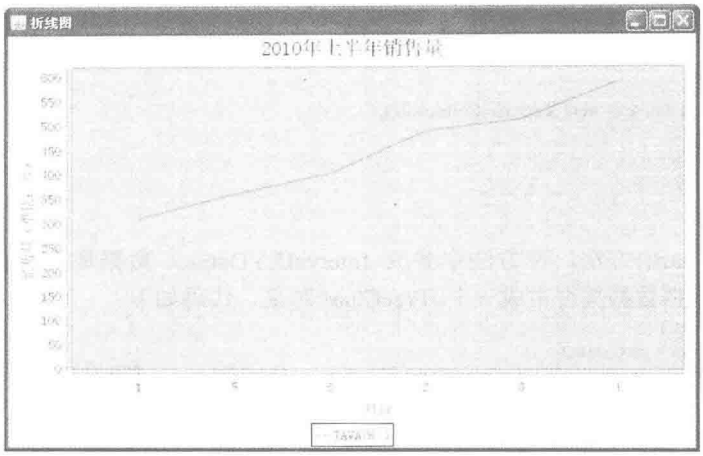


图 9.48　排序折线图

关键技术

使用 DefaultKeyedValues 类的 sortByValues()方法可以对数据集进行排序，语法如下：

```
sortByValues(SortOrder order)
```

参数说明

order：表示 DefaultKeyedValues 数据内容的排序方式。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，使用 DefaultKeyedValues 类创建分类数据集，代码如下：

```
private CategoryDataset getCategoryDataset() {
        DefaultKeyedValues keyedValues = new DefaultKeyedValues();
        keyedValues.addValue("1", 310);
        keyedValues.addValue("2", 489);
        keyedValues.addValue("3", 512);
        keyedValues.addValue("4", 589);
        keyedValues.addValue("5", 359);
        keyedValues.addValue("6", 402);
        //排序
        keyedValues.sortByValues(SortOrder.ASCENDING);
        CategoryDataset dataset = DatasetUtilities.createCategoryDataset("JAVA 图书", keyedValues);
        return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中获取 CategoryDataset 数据集合，再使用 ChartFactory 类的 createLineChart()方法生成 JFreeChart 对象，代码见实例 264。

（4）创建 updateFont()方法，在方法中修改标题、X 轴标签、Y 轴标签等字体，代码见实例 264。

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象修改字体，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 264。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        LineDemo12 demo = new LineDemo12("折线图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 275：使用 sortByValues()方法对数据集进行排序。

DefaultKeyedValues 类使用 sortByValues()方法对数据集进行排序，该方法的参数为 SortOrder 类，SortOrder 类中定义了两个常量，SortOrder.ASCENDING 表示数据集升序排列；SortOrder.DESCENDING 表示数据集降序排列。

9.7　环　形　图

实例 276　基本环形图　　中级

光盘位置：光盘\MR\276　　趣味指数：★★★★

实例说明

环形图与饼形图类似，但是在环形图的图表中是一个空心的环的形状。本实例演示如何创建一个基本的环形图，运行效果如图 9.49 所示。

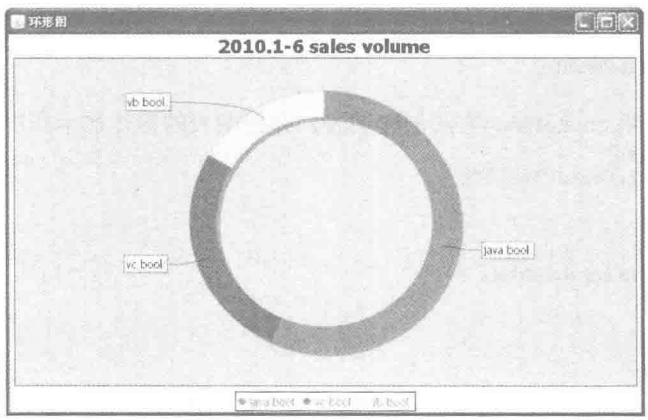


图 9.49　基本环形图

关键技术

使用 ChartFactory 类的 createRingChart()方法可以创建基本环形图，创建完成以后会返回一个 JFreeChart 对象，语法如下：

createRingChart(String title, PieDataset dataset, boolean legend, boolean tooltips, boolean urls)

参数说明

❶ title：表示图表的标题。

❷ dataset：表示环形图的数据集合。

❸ legend：表示是否使用图示。

❹ tooltips：表示是否生成工具栏提示。

❺ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getPieDataset()方法，在方法中使用 DefaultPieDataset 类创建一个数据集，代码如下：

```
private PieDataset getPieDataset() {
        DefaultPieDataset dataset = new DefaultPieDataset();
        dataset.setValue("java book", 1689);
        dataset.setValue("vc book", 810);
        dataset.setValue("vb book", 490);
        return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中使用 getPieDataset()方法获取数据集合，再使用 ChartFactory 类的 createRingChart()方法生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createRingChart(
        "2010.1-6 sales volume",                    //图表标题
        dataset,                                    //数据集
        true,                                       //显示图例
        false,                                      //不生成工具栏提示
        false                                       //不生成 URL 链接
        );
        return chart;
}
```

（4）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（5）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        RingDeom1 demo = new RingDeom1("环形图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 276：通过 setValue()方法向数据集添加数据。

DefaultPieDataset 类可以用于创建环形图的数据集，使用 DefaultPieDataset 类的 setValue()方法可以向数据集添加数据，语法如下：

```
setValue(Comparable key, double value)
```

参数说明

❶ key：表示环形图的分类名称。

❷ value：表示环形图当前分类数值。

实例 277　环形图字体

光盘位置：光盘\MR\277　　中级　　趣味指数：★★★★

实例说明

环形图的字体默认情况下不支持汉字，要让环形图支持汉字，必须修改默认字体。本实例演示如何修改环

形图的默认字体，运行效果如图 9.50 所示。

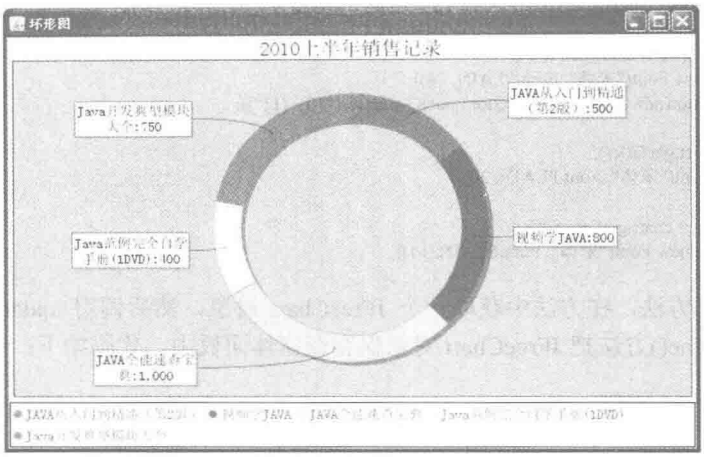


图 9.50　环形图字体

关键技术

（1）使用 RingPlot 类的 setLabelFont()方法可以修改环形图标签字体，语法如下：

```
setLabelFont(Font font)
```

参数说明

font：表示环形图的标签字体。

（2）使用 TextTitle 类的 setFont()方法可以修改环形图的标题字体，语法如下：

```
setFont(Font font)
```

参数说明

font：表示标题的字体。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getPieDataset()的方法，在方法中使用 DefaultPieDataset 类创建一个数据集，代码如下：

```
private PieDataset getPieDataset() {
        DefaultPieDataset dataset = new DefaultPieDataset();
        dataset.setValue("JAVA 从入门到精通（第 2 版）", 500);
        dataset.setValue("视频学 JAVA", 800);
        dataset.setValue("JAVA 全能速查宝典", 1,000);
        dataset.setValue("Java 范例完全自学手册(1DVD)", 400);
        dataset.setValue("Java 开发典型模块大全", 750);
        return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中使用 getPieDataset()方法获取数据集合，再使用 ChartFactory 类的 createRingChart()方法生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        PieDataset dataset = getPieDataset();
        JFreeChart chart = ChartFactory.createRingChart(
        "2010 上半年销售记录",                    //图表标题
        dataset,                                  //数据集
        true,                                     //是否显示图例
        false,                                    //是否生成工具栏提示
        false                                     //是否生成 URL 链接
        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改图表、标题、图示等字体，代码如下：

```
private void updateFont(JFreeChart chart) {
        //图表
        RingPlot ringPlot = (RingPlot) chart.getPlot();
        ringPlot.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ringPlot.setLabelGenerator(new StandardPieSectionLabelGenerator("{0}:{1}"));
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        //图示
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
}
```

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后调用 updateFont()方法修改字体，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //设置字体
        updateFont(chart);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        RingDeom2 demo = new RingDeom2("环形图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 277：通过 setItemFont()方法修改环形图示字体。

使用 LegendTitle 类的 setItemFont()方法可以修改环形图示字体，语法如下：

```
setItemFont(Font font)
```

参数说明

font：表示环形图图示字体。

实例 278 设置环形深度

光盘位置：光盘\MR\278　　中级　　趣味指数：★★★★

实例说明

JFreeChart 可以根据需要设置环形图的深度，修改深度可以改变环形图面积。本实例演示如何设置环形图的深度，运行效果如图 9.51 所示。

关键技术

环形图深度的默认值为 0.20，使用 RingPlot 类的 setSectionDepth()方法可以设置环形图的深度，语法如下：

```
setSectionDepth(double sectionDepth)
```

参数说明

sectionDepth：表示环形图的深度，范围为 0~1。

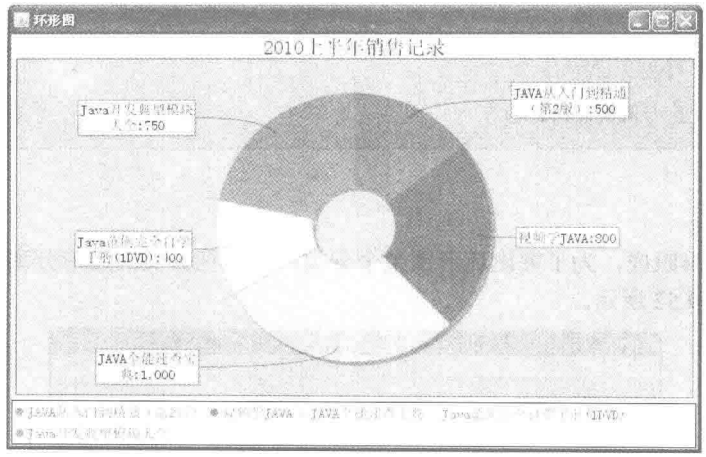


图 9.51　设置环形图深度

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getPieDataset()方法，在方法中使用 DefaultPieDataset 类创建一个数据集，代码见实例 277。

（3）创建 getJFreeChart()方法，在方法中使用 getPieDataset()方法获取数据集合，再使用 ChartFactory 类的 createRingChart()方法生成一个 JFreeChart 对象，代码见实例 277。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示等字体，代码见实例 277。

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，设置环形图深度，然后调用 updateFont()方法修改字体，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        RingPlot ringPlot = (RingPlot) chart.getPlot();
        //设置环形深度
        ringPlot.setSectionDepth(0.7);
        //设置字体
        updateFont(chart);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
}
```

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        RingDeom3 demo = new RingDeom3 ("环形图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 278：通过 setSectionDepth()方法设置环形图的深度。

利用 RingPlot 类的 setSectionDepth()方法可以设置环形图的深度，深度越小，环形图的面积越小；深度越大，环形图的面积越大，当深度为 1 时，环形图则变成圆形图。

实例 279 分离环形图

光盘位置：光盘\MR\279

中级

趣味指数：★★★★

实例说明

整个环形图由多个扇形组成，为了突出环形图某个分类，可以把分类的扇形分离出来。本实例演示如何分离环形图，运行效果如图 9.52 所示。

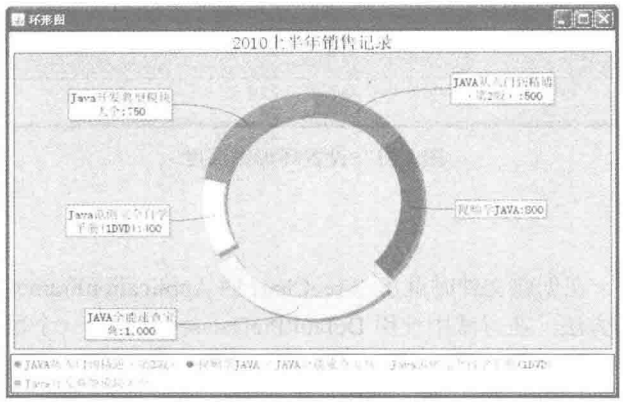


图 9.52 分离环形图

关键技术

使用 RingPlot 类的 setExplodePercent()方法可以分离环形图指定的分类，语法如下：

```
setExplodePercent(Comparable key, double percent)
```

参数说明

❶ key：表示环形图的分类。

❷ percent：表示环形图指定分类的分离数值，percent 值越小，分离的距离越小；percent 值越大，分离的距离越大。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getPieDataset()方法，在方法中使用 DefaultPieDataset 类创建一个数据集，代码见实例 277。

（3）创建 getJFreeChart()方法，在方法中使用 getPieDataset()方法获取数据集合，再使用 ChartFactory 类的 createRingChart()方法生成一个 JFreeChart 对象，代码见实例 277。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示等字体，代码见实例 277。

（5）创建 updatePlot()方法，在方法中设置环形图“JAVA 全能速查宝典”的类别分离值为 0.1，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //图表
    RingPlot plot = (RingPlot) chart.getPlot();
    plot.setExplodePercent("JAVA 全能速查宝典", 0.1);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体和图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
    JFreeChart chart = getJFreeChart();
    //设置字体
    updateFont(chart);
```

```
        //设置图表
        updatePlot(chart);
        //把 JFreeChart 面板保存在窗体中
        setContentPane(new ChartPanel(chart));
    }
```

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        RingDeom4 demo = new RingDeom4("环形图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 279：通过 getExplodePercent()方法获取某分类的分离值。

使用 RingPlot 类的 getExplodePercent()方法可以获取某分类的分离值，语法如下：

```
getExplodePercent(Comparable key)
```

参数说明

key：表示环形图的分类。

实例 280　椭圆环形图

光盘位置：光盘\MR\280　　中级　　趣味指数：★★★★★

实例说明

环形图默认情况下显示为正圆形，也可以根据实际需要将其设置为椭圆形。本实例演示如何设置环形图为椭圆形，运行效果如图 9.53 所示。

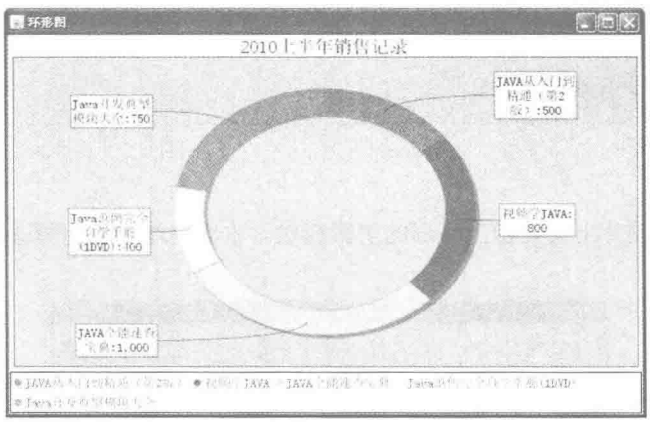


图 9.53　椭圆环形图

关键技术

使用 RingPlot 类的 setCircular()方法可以设置环形图为椭圆状，语法如下：

```
setCircular(boolean flag)
```

参数说明

flag：表示环形图是否显示为椭圆，当 flag 为 true 时，表示环形图为正圆形；当 flag 为 false 时，表示环形图为椭圆形。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getPieDataset()方法，在方法中使用 DefaultPieDataset 类创建一个数据集，代码见实例 277。

（3）创建 getJFreeChart()方法，在方法中使用 getPieDataset()方法获取数据集合，再使用 ChartFactory 类的 createRingChart()方法生成一个 JFreeChart 对象，代码见实例 277。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示等字体，代码见实例 277。

（5）创建 updatePlot()方法，在方法中设置环形图为椭圆形，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        RingPlot plot = (RingPlot) chart.getPlot();
        plot.setCircular(false);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体和图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 279。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        RingDeom5 demo = new RingDeom5("环形图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 280：通过 isCircular()方法判断获取的环形图是否是正圆形。

使用 RingPlot 类的 isCircular()方法可以获取当前的环形图是否是正圆形，如果 isCircular()方法的返回值为 true，表示当前环形图为正圆形；如果 isCircular()方法的返回值为 false 时，表示当前环形图为椭圆形。

实例 281　环形的阴影偏移

光盘位置：光盘\MR\281　　中级　　趣味指数：★★★★

实例说明

在显示环形图时，背景图片上会显示出环形的阴影形状。本实例演示如何修改环形图的阴影偏移，运行效果如图 9.54 所示。

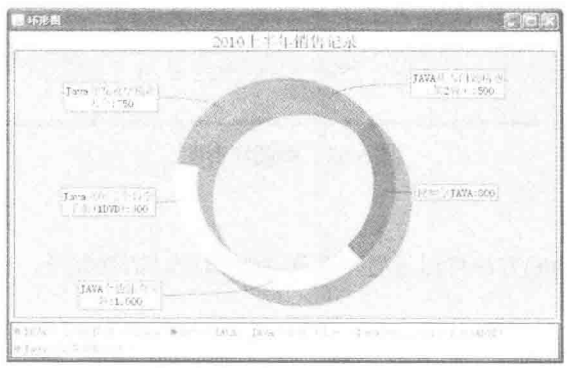


图 9.54　环形图阴影偏移

关键技术

使用 RingPlot 类的 setShadowXOffset()和 setShadowYOffset()方法可以设置环形图的 X 轴和 Y 轴的偏移量，语法如下：

```
//设置 X 轴偏移
setShadowXOffset(double offsetx)
//设置 Y 轴偏移
setShadowYOffset(double offsety)
```

参数说明

❶ offsetx：表示 X 轴的偏移量。

❷ offsety：表示 Y 轴的偏移量。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getPieDataset()方法，在方法中使用 DefaultPieDataset 类创建一个数据集，代码见实例 277。

（3）创建 getJFreeChart()方法，在方法中使用 getPieDataset()方法获取数据集合，再使用 ChartFactory 类的 createRingChart()方法生成一个 JFreeChart 对象，代码见实例 277。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示等字体，代码见实例 277。

（5）创建 updatePlot()方法，在方法中分别设置环形图 X、Y 的阴影偏移量为 20，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        RingPlot plot = (RingPlot) chart.getPlot();
        //阴影偏移
        plot.setShadowXOffset(20);
        plot.setShadowYOffset(20);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，修改图表字体和图表设置，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 279。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        RingDeom6 demo = new RingDeom6("环形图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 281：环形图阴影偏移的特点。

环形图阴影的 X 轴和 Y 轴的偏移量最小值都为 0，如果设置 X 轴和 Y 轴的偏移量为 0，阴影则被隐藏在环形图下面，即图表中只显示环形。

实例 282　环形的简单标签

中级

光盘位置：光盘\MR\282

趣味指数：★★★

实例说明

绘制环形图时，可以设置标签为简单状态，将标签直接显示在环形图上。本实例演示如何设置环形图使用

简单标签，运行效果如图 9.55 所示。

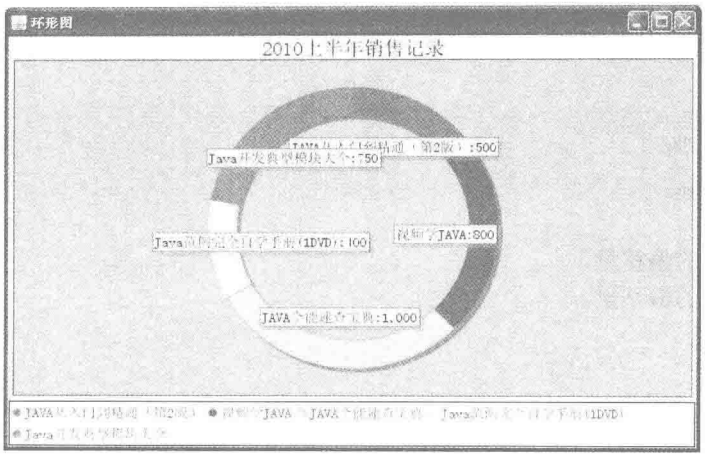


图 9.55　环形图的简单标签

关键技术

使用 RingPlot 类的 setSimpleLabels()方法可以设置环形图是否使用简单标签，语法如下：

```
setSimpleLabels(boolean simple)
```

参数说明

simple：表示是否使用简单标签。simple 为 true 时，表示使用简单标签；simple 为 false 时，表示不使用简单标签。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getPieDataset()方法，在方法中使用 DefaultPieDataset 类创建一个数据集，代码见实例 277。

（3）创建 getJFreeChart()方法，在方法中使用 getPieDataset()方法获取数据集合，再使用 ChartFactory 类的 createRingChart()方法生成一个 JFreeChart 对象，代码见实例 277。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示等字体，代码见实例 277。

（5）创建 updatePlot()方法，在方法中设置环形图使用简单标签，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        RingPlot plot = (RingPlot) chart.getPlot();
        //环形的简单标签
        plot.setSimpleLabels(true);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，修改图表字体和图表设置，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 279。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        RingDeom7 demo = new RingDeom7("环形图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 282：使用 setLabelLinksVisible()方法隐藏标签。

隐藏复杂标签的连线，图表上也只显示标签内容，此时标签显示与简单标签相似，可使用 RingPlot 类的 setLabelLinksVisible()方法隐藏标签连线，语法如下：

```
setLabelLinksVisible(boolean visible)
```

参数说明

visible：表示是否显示标签连线。visible 为 false 时，表示隐藏标签连线；visible 为 true 时，表示显示标签连线。

实例 283　环形图的旋转角度

光盘位置：光盘\MR\283　　中级　　趣味指数：★★

实例说明

绘制环形图时，可以设置旋转角度，让整个环形图按一定角度旋转。本实例演示如何设置环形图的旋转角度，运行效果如图 9.56 所示。

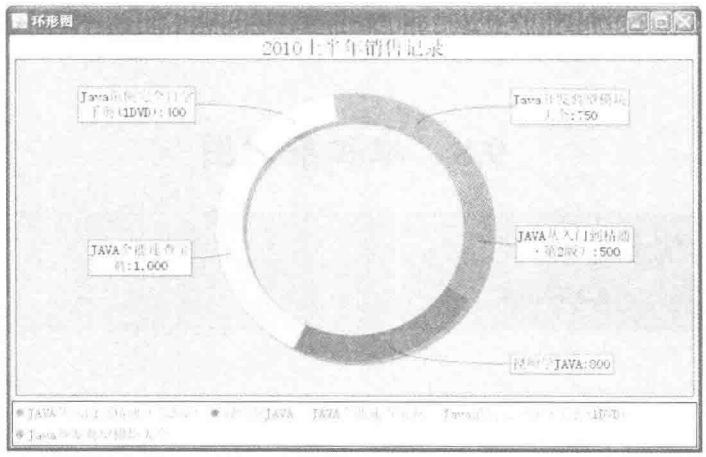


图 9.56　环形图的旋转角度

关键技术

使用 RingPlot 类的 setStartAngle()方法可以设置环形图的旋转角度，语法如下：

```
setStartAngle(double angle)
```

参数说明

angle：表示环形图的旋转角度。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getPieDataset()方法，在方法中使用 DefaultPieDataset 类创建一个数据集，代码见实例 277。

（3）创建 getJFreeChart()方法，在方法中使用 getPieDataset()方法获取数据集合，再使用 ChartFactory 类的 createRingChart()方法生成一个 JFreeChart 对象，代码见实例 277。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示等字体，代码见实例 277。

（5）创建 updatePlot()方法，在方法中设置环形图的旋转角度，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //图表
        RingPlot plot = (RingPlot) chart.getPlot();
        //环形图的旋转角度
        plot.setStartAngle(20);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，修改图表字体和图表设置，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 279。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        RingDeom8 demo = new RingDeom8("环形图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 283：通过 setStartAngle()方法设置环形图的旋转角度。

环形图的旋转角度以数据集中添加的第一个分类的左边线为基准，默认情况下与水平边框成 90°，setStartAngle()方法的参数为正数时，环形图逆时针旋转角度；setStartAngle()方法的参数为负数时，环形图顺时针旋转角度。

9.8 堆积条形图

实例 284 基本堆积条形图　　高级

光盘位置：光盘\MR\284　　趣味指数：★★★★

实例说明

堆积条形图按分类显示，每个分类按系列索引顺序进行累加。本实例演示如何绘制基本堆积条形图，运行效果如图 9.57 所示。

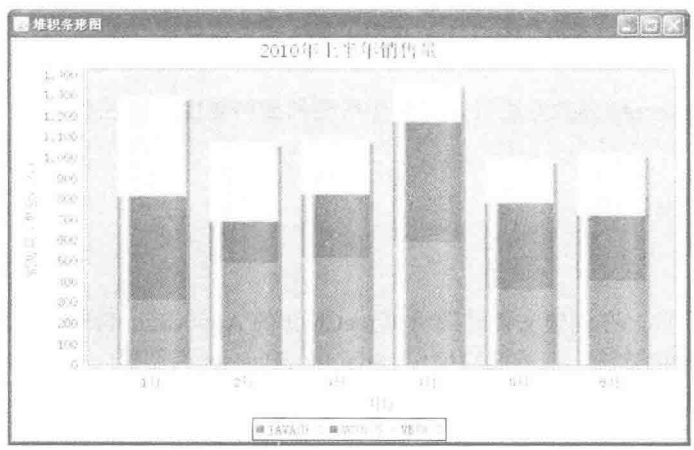


图 9.57　堆积条形图

关键技术

ChartFactory 类的 createStackedBarChart()方法提供了创建基本堆积条形图的方法，创建完成以后会返回一个 JFreeChart 对象，语法如下：

createStackedBarChart(String title, String domainAxisLabel, String rangeAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)

参数说明

❶ title：图表的标题。

❷ domainAxisLabel：表示区域标签，一般用于 X 轴的名称。

❸ rangeAxisLabel：表示范围标签，一般用于 Y 轴的名称。

❹ dataset：堆积条形图的数据集合。

❺ orientation：表示图表的显示方向。

❻ legend：表示是否使用图示。

❼ tooltips：表示是否生成工具栏提示。

❽ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建一个数据集，代码如下：

```
private CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书";
        final String series2 = "VC 图书";
        final String series3 = "VB 图书";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中使用 getCategoryDataset()方法获取数据集合，再使用 ChartFactory 类的 createStackedBarChart()方法生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
```

```
        JFreeChart chart = ChartFactory.createStackedBarChart("2010 年上半年销售量",      //图表标题
        "月份",                                                                          //X 轴标签
        "销售量（单位：本）",                                                            //Y 轴标签
        dataset,                                                                         //数据集
        PlotOrientation.VERTICAL,                                                        //图表方向：垂直
        true,                                                                            //是否显示图例
        false,                                                                           //是否生成工具栏提示
        false                                                                            //是否生成 URL 链接
        );
        return chart;
    }
```

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、数据轴等字体，代码如下：

```
    private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis = categoryPlot.getDomainAxis();
        //X 轴字体
        categoryAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
        //Y 轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    }
```

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，修改图表字体，然后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
    public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        setContentPane(new ChartPanel(chart));
    }
```

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        StackedBarDemo1 demo = new StackedBarDemo1("堆积条形图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 284：通过 setTickLabelFont()方法设置坐标轴的字体。

使用 CategoryPlot 类可以从 getDomainAxis()方法中获取 CategoryAxis 实例、从 getRangeAxis()方法中获取 ValueAxis 实例，CategoryAxis 与 ValueAxis 类有一个共同的方法——setTickLabelFont()，可以设置坐标轴的字体，语法如下：

```
setTickLabelFont(Font font)
```

参数说明

font：表示要设置的坐标轴字体。

实例 285　横向堆积条形图

光盘位置：光盘\MR\285　　高级　　趣味指数：★★★★

实例说明

堆积条形图默认情况下垂直显示，可根据需要将其设置为水平状态显示。本实例演示如何把堆积条形图设置为水平显示，运行效果如图 9.58 所示。

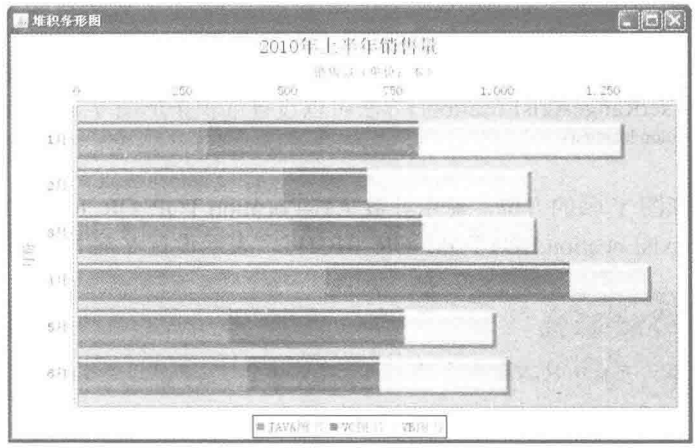


图 9.58　横向堆积条形图

关键技术

使用 PlotOrientation 类可以设置堆积条形图的显示状态，PlotOrientation 类有两个常量，PlotOrientation.HORIZONTAL 表示堆积条形图水平显示；PlotOrientation.VERTICAL 表示堆积条形图垂直显示。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建一个数据集，代码见实例 284。

（3）创建 getJFreeChart()方法，在方法中使用 getCategoryDataset()方法获取数据集合，再使用 ChartFactory 类的 createStackedBarChart()方法生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
    CategoryDataset dataset = getCategoryDataset();
    JFreeChart chart = ChartFactory.createStackedBarChart("2010 年上半年销售量",    //图表标题
    "月份",                                                                          //X 轴标签
    "销售量（单位：本）",                                                            //Y 轴标签
    dataset,                                                                         //数据集
    PlotOrientation.HORIZONTAL,                                                      //图表方向：水平
    true,                                                                            //是否显示图例
    false,                                                                           //是否生成工具栏提示
    false                                                                            //是否生成 URL 链接
    );
    return chart;
}
```

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、数据轴等字体，代码见实例 284。

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，修改图表字体，然后调用父类的

setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 284。

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    StackedBarDemo2 demo = new StackedBarDemo2("堆积条形图");
    demo.createPlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 285：使用 setRangeAxisLocation()方法可以设置堆积条形图 Y 轴的位置。

使用 CategoryPlot 类的 setRangeAxisLocation()方法可以设置堆积条形图 Y 轴的位置，语法如下：

```
setRangeAxisLocation(AxisLocation location)
```

参数说明

location：表示堆积条形图 Y 轴的方位，使用参数 AxisLocation.TOP_OR_RIGHT，表示设置 Y 轴显示在图表顶部或右侧；使用参数 AxisLocation.BOTTOM_OR_RIGHT，表示设置 Y 轴显示在图表底部或右侧。

实例 286　设置边线距离

光盘位置：光盘\MR\286　　中级　　趣味指数：★★★

实例说明

堆积条形图中的柱形顶部与图表上边线一般情况下都留有距离。本实例演示如何设置图表柱形的顶部与图表上边线的距离，运行效果如图 9.59 所示。

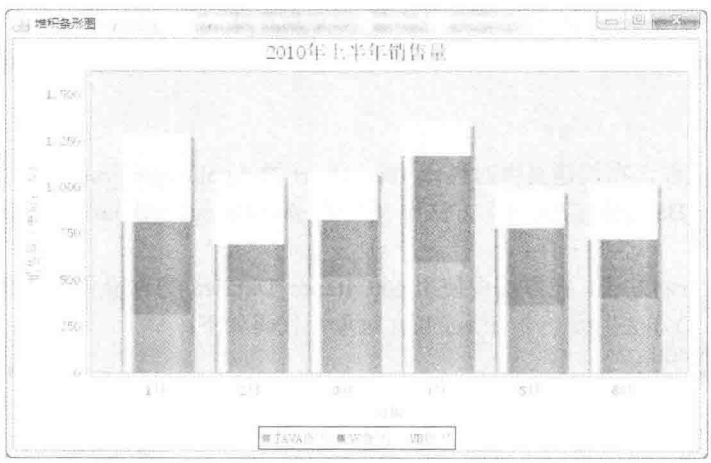


图 9.59　设置边线距离

关键技术

使用 ValueAxis 类的 setUpperMargin()方法可以设置堆积条形图中的柱形顶部与图表上边线的距离，语法如下：

```
setUpperMargin(double margin)
```

参数说明

margin：表示距离的高度，为堆积条形图的柱形图与图表上边线的距离与堆积条形图中柱形图高度的比例。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建一个数据集，代码见实例 284。

（3）创建 getJFreeChart()方法，在方法中使用 getCategoryDataset()方法获取数据集合，再使用 ChartFactory 类的 createStackedBarChart()方法生成一个 JFreeChart 对象，代码见实例 284。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、数据轴等字体，代码见实例 284。

（5）创建 updatePlot()方法，在方法中获取 ValueAxis 实例，然后设置堆积条形图中柱形顶部与上边线的距离为 0.8，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //分类图表
    CategoryPlot categoryPlot = chart.getCategoryPlot();
        final ValueAxis rangeAxis = categoryPlot.getRangeAxis();
        //上边线距离
        rangeAxis.setUpperMargin(0.8);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
    JFreeChart chart = getJFreeChart();
    //修改字体
    updateFont(chart);
    //修改图表
    updatePlot(chart);
    setContentPane(new ChartPanel(chart));
}
```

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    StackedBarDemo3 demo = new StackedBarDemo3("堆积条形图");
    demo.createPlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 286：通过 setLowerMargin()方法获取堆积条形图中柱形与图表下边线的距离。

使用 ValueAxis 类的 setLowerMargin()方法可以设置堆积条形图中柱形与图表下边线的距离，语法如下：

```
setLowerMargin(double margin)
```

参数说明

margin：表示距离的高度，为堆积条形图的柱形图与图表下边线的距离与堆积条形图中柱形图高度的比例。

实例 287　分组堆积条形图

光盘位置：光盘\MR\287　　高级　　趣味指数：★★★★

实例说明

堆积条形图可以把每个分类的数据进行分组显示。本实例演示如何把堆积条形图拆分为两组进行显示，运行效果如图 9.60 所示。

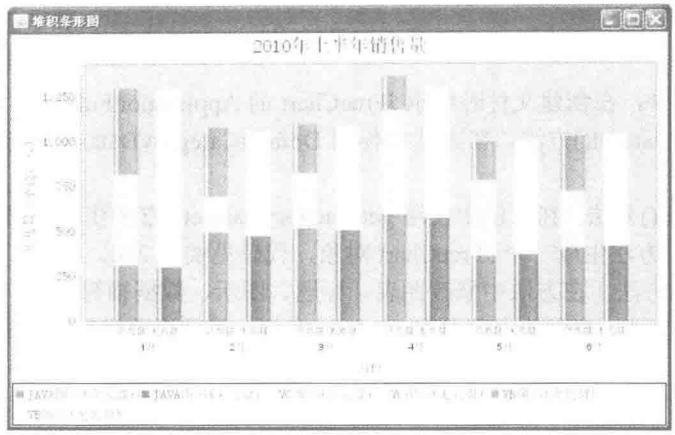


图 9.60　分组堆积条形图

关键技术

（1）使用 GroupedStackedBarRenderer 类的 setSeriesToGroupMap()方法可以为堆积条形图添加分组图表，语法如下：

```
setSeriesToGroupMap(KeyToGroupMap map)
```

参数说明

map：表示分组内容。

（2）使用 KeyToGroupMap 类的 mapKeyToGroup()方法可以向分组数据集中添加分组数据，语法如下：

```
mapKeyToGroup(Comparable key, Comparable group)
```

参数说明

❶ key：表示分组关键字。

❷ group：表示分组的名称。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getCategoryDataset()方法，在方法中使用 DefaultCategoryDataset 类创建一个数据集，代码如下：

```
private CategoryDataset getCategoryDataset() {
        //行关键字
        final String series1 = "JAVA 图书(含光盘)";
        final String series2 = "VC 图书(含光盘)";
        final String series3 = "VB 图书(含光盘)";
        final String series11 = "JAVA 图书(无光盘)";
        final String series21 = "VC 图书(无光盘)";
        final String series31 = "VB 图书(无光盘)";
        //列关键字
        final String category1 = "1 月";
        final String category2 = "2 月";
        final String category3 = "3 月";
        final String category4 = "4 月";
        final String category5 = "5 月";
        final String category6 = "6 月";
        //创建分类数据集
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(310, series1, category1);
        dataset.addValue(489, series1, category2);
        dataset.addValue(512, series1, category3);
        dataset.addValue(589, series1, category4);
        dataset.addValue(359, series1, category5);
        dataset.addValue(402, series1, category6);
        dataset.addValue(300, series11, category1);
        dataset.addValue(469, series11, category2);
```

```
        dataset.addValue(502, series11, category3);
        dataset.addValue(569, series11, category4);
        dataset.addValue(369, series11, category5);
        dataset.addValue(412, series11, category6);
        dataset.addValue(501, series2, category1);
        dataset.addValue(200, series2, category2);
        dataset.addValue(308, series2, category3);
        dataset.addValue(580, series2, category4);
        dataset.addValue(418, series2, category5);
        dataset.addValue(315, series2, category6);
        dataset.addValue(511, series21, category1);
        dataset.addValue(210, series21, category2);
        dataset.addValue(318, series21, category3);
        dataset.addValue(560, series21, category4);
        dataset.addValue(408, series21, category5);
        dataset.addValue(305, series21, category6);
        dataset.addValue(480, series3, category1);
        dataset.addValue(381, series3, category2);
        dataset.addValue(264, series3, category3);
        dataset.addValue(185, series3, category4);
        dataset.addValue(209, series3, category5);
        dataset.addValue(302, series3, category6);
        dataset.addValue(470, series31, category1);
        dataset.addValue(371, series31, category2);
        dataset.addValue(254, series31, category3);
        dataset.addValue(165, series31, category4);
        dataset.addValue(219, series31, category5);
        dataset.addValue(312, series31, category6);
        return dataset;
    }
```

（3）创建 getJFreeChart()方法，在方法中使用 getCategoryDataset()方法获取数据集合，再使用 ChartFactory 类的 createStackedBarChart()方法生成一个 JFreeChart 对象，代码见实例 284。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、数据轴等字体，代码见实例 284。

（5）创建 updatePlot()方法，在方法中添加分组数据，同时为坐标轴添加分组刻度，代码如下：

```
    private void updatePlot(JFreeChart chart) {
        //分类图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        //添加 X 轴子标签
          SubCategoryAxis domainAxis = new SubCategoryAxis("月份");
          domainAxis.addSubCategory("含光盘");
          domainAxis.addSubCategory("无光盘");
          categoryPlot.setDomainAxis(domainAxis);
          //分组堆积条形图渲染
          GroupedStackedBarRenderer renderer = new GroupedStackedBarRenderer();
          KeyToGroupMap map = new KeyToGroupMap("G1");
          //分为两组
          map.mapKeyToGroup("JAVA 图书(含光盘)", "G1");
          map.mapKeyToGroup("VC 图书(含光盘)", "G1");
          map.mapKeyToGroup("VB 图书(含光盘)", "G1");
          map.mapKeyToGroup("JAVA 图书(无光盘)", "G2");
          map.mapKeyToGroup("VC 图书(无光盘)", "G2");
          map.mapKeyToGroup("VB 图书(无光盘)", "G2");
          renderer.setSeriesToGroupMap(map);
          categoryPlot.setRenderer(renderer);
    }
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 286。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        StackedBarDemo4 demo = new StackedBarDemo4("堆积条形图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 287：通过 addSubCategory()方法向子分类轴添加分类名称。

使用 SubCategoryAxis 类的 addSubCategory()方法可以向子分类轴添加分类名称，语法如下：

addSubCategory(Comparable subCategory)

参数说明

subCategory：表示子分类轴的名称。

9.9 时 序 图

实例 288 基本时序图　　中级

光盘位置：光盘\MR\288　　趣味指数：★★★★★

实例说明

时序图一般情况下使用时间轴表示某个分类的进展状况。本实例演示如何绘制基本时序图，运行效果如图 9.61 所示。

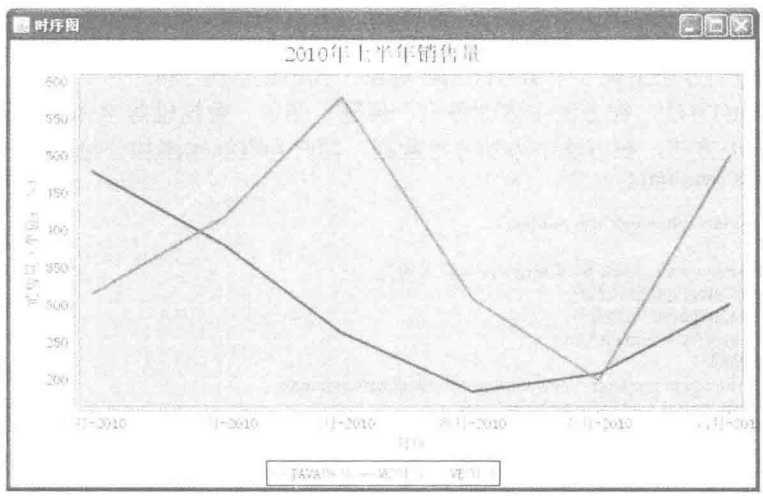


图 9.61　基本时序图

关键技术

使用 ChartFactory 类的 createTimeSeriesChart()方法可以创建一个基本时序图，创建完成以后会返回一个 JFreeChart 对象，语法如下：

createTimeSeriesChart(String title, String timeAxisLabel, String valueAxisLabel, XYDataset dataset, boolean legend, boolean tooltips, boolean urls)

参数说明

❶ title：图表的标题。

❷ timeAxisLabel：表示时间标签，一般用于 X 轴的名称。

❸ valueAxisLabel：表示范围标签，一般用于 Y 轴的名称。

❹ dataset：表示时序图的数据集合。

❺ legend：表示是否使用图示。

❻ tooltips：表示是否生成工具栏提示。

❼ urls：表示是否生成 URL 链接。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 TimeSeriesCollection 类创建一个数据集，代码如下：

```
private XYDataset getDataset() {
        final TimeSeries s1 = new TimeSeries("JAVA 图书");
        s1.add(new Month(1, 2010), 480);
        s1.add(new Month(2, 2010), 381);
        s1.add(new Month(3, 2010), 264);
        s1.add(new Month(4, 2010), 185);
        s1.add(new Month(5, 2010), 209);
        s1.add(new Month(6, 2010), 302);
        final TimeSeries s2 = new TimeSeries("VC 图书");
        s2.add(new Month(1, 2010), 315);
        s2.add(new Month(2, 2010), 418);
        s2.add(new Month(3, 2010), 580);
        s2.add(new Month(4, 2010), 308);
        s2.add(new Month(5, 2010), 200);
        s2.add(new Month(6, 2010), 501);
        final TimeSeries s3 = new TimeSeries("VB 图书");
        s3.add(new Month(1, 2010), 310);
        s3.add(new Month(2, 2010), 489);
        s3.add(new Month(3, 2010), 512);
        s3.add(new Month(4, 2010), 589);
        s3.add(new Month(5, 2010), 359);
        s3.add(new Month(6, 2010), 402);
        final TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        dataset.addSeries(s3);
        return dataset;
}
```

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象，代码如下：

```
private JFreeChart getJFreeChart() {
        XYDataset dataset = getDataset();
        JFreeChart chart = ChartFactory.createTimeSeriesChart("2010 年上半年销售量",      //图表标题
        "月份",                                                                          //X 轴标签
        "销售量（单位：本）",                                                            //Y 轴标签
        dataset,                                                                         //数据集
        true,                                                                            //是否显示图例
        false,                                                                           //是否生成工具栏提示
        false                                                                            //是否生成 URL 链接
        );
        return chart;
}
```

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体，代码如下：

```
private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN, 14));
        //图表
        XYPlot xyPlot = chart.getXYPlot();
        ValueAxis domainyAxis = xyPlot.getDomainAxis();
        //X 轴字体
        domainyAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //X 轴标签字体
        domainyAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
        ValueAxis rangeAxis = xyPlot.getRangeAxis();
        //Y 轴字体
```

```
        rangeAxis.setTickLabelFont(new Font("宋体", Font.PLAIN, 14));
        //Y 轴标签字体
        rangeAxis.setLabelFont(new Font("宋体", Font.PLAIN, 14));
    }
```

（5）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
    public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        setContentPane(new ChartPanel(chart));
    }
```

（6）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
        TimeSeriesDemo1 demo = new TimeSeriesDemo1("时序图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 288：通过 addSeries()方法向数据集添加时序数据。

使用 TimeSeriesCollection 的 addSeries()方法可以向数据集添加时序数据，语法如下：

```
addSeries(TimeSeries series)
```

参数说明

series：表示数据集中的时序数据。

实例 289 设置时间显示格式

光盘位置：光盘\MR\289　　中级　　趣味指数：★★★★

实例说明

时序图中时间刻度的样式可以根据需求进行调整。本实例演示如何格式化时序图中的时间样式，运行效果如图 9.62 所示。

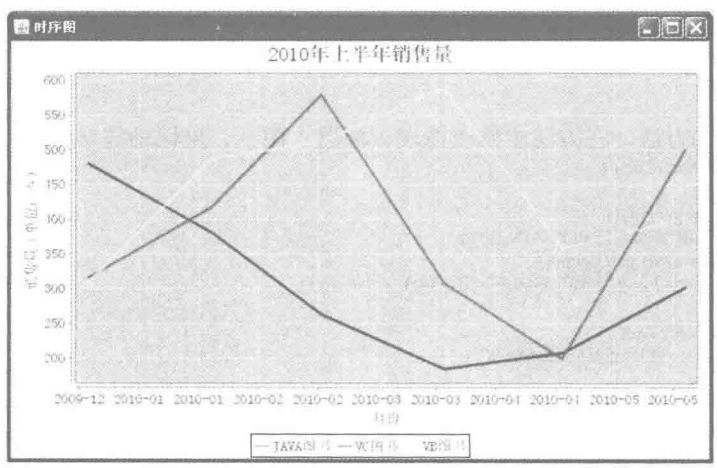


图 9.62　设置时间格式

关键技术

使用 DateAxis 类的 setDateFormatOverride()方法可以格式化时间轴显示格式，创建完成以后会返回一个 JFreeChart 对象，语法如下：

```
setDateFormatOverride(DateFormat formatter)
```

参数说明

formatter：表示时序图时间轴的显示格式。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 TimeSeriesCollection 类创建一个数据集，代码见实例 288。

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象，代码见实例 288。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体，代码见实例 288。

（5）创建 updatePlot()方法，在方法中获取一个 XYPlot 实例，然后重新格式化时间格式。

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码如下：

```
public void createPlot() {
        JFreeChart chart = getJFreeChart();
        //修改字体
        updateFont(chart);
        //修改图表
        updatePlot(chart);
        setContentPane(new ChartPanel(chart));
}
```

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        TimeSeriesDemo2 demo = new TimeSeriesDemo2("时序图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 289：重新设置时间格式。

使用 SimpleDateFormat 的构造方法可以重新设置时间格式，语法如下：

```
SimpleDateFormat(String pattern)
```

参数说明

pattern：表示时间格式。

实例 290　添加双时间轴

光盘位置：光盘\MR\290　　中级　　趣味指数：★★★★

实例说明

JFreeChart 可以为时序图添加第二个时间轴，用来表示另一个时间顺序。本实例演示如何添加双时间轴，运行效果如图 9.63 所示。

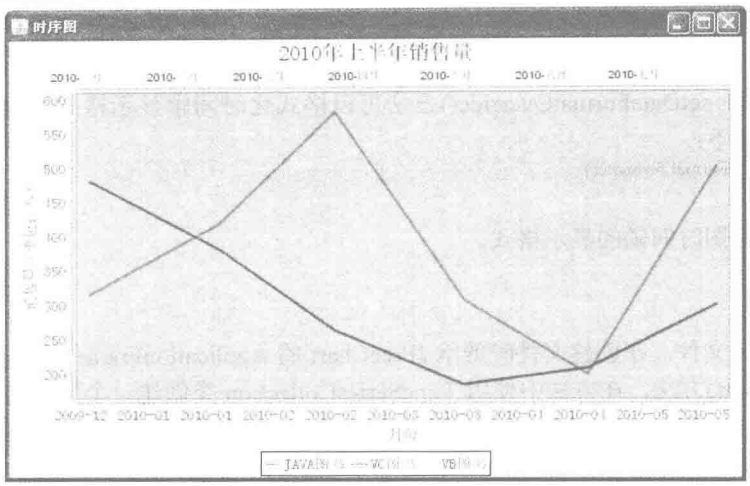


图 9.63　双时间轴

关键技术

使用 XYPlot 类的 setDomainAxis()方法可以为时序图新增一个时间轴，语法如下：

```
setDomainAxis(int index, ValueAxis axis)
```

参数说明

❶ index：表示时序图的索引。

❷ axis：表示要添加的指定索引的时间轴。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 TimeSeriesCollection 类创建一个数据集，代码见实例 288。

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象，代码见实例 288。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体，代码见实例 288。

（5）创建 updatePlot()方法，在方法中获取一个 XYPlot 实例，为图表新增一个时间轴，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        XYPlot xyPlot = chart.getXYPlot();
        DateAxis dateAxis = (DateAxis) xyPlot.getDomainAxis();
        //设置显示格式
        dateAxis.setDateFormatOverride(new SimpleDateFormat("yyyy-MM"));

        //添加时间轴
        DateAxis dateAxis1 = new DateAxis();
        //添加时间范围
        dateAxis1.setRange(new Month(1, 2010).getStart(), new Month(7, 2010).getEnd());
        //设置时间表格
        dateAxis1.setDateFormatOverride(new SimpleDateFormat("yyyy-MMM"));
        xyPlot.setDomainAxis(1,dateAxis1);

    }
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 289。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        TimeSeriesDemo3 demo = new TimeSeriesDemo3("时序图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 290：通过 setRange()方法为时间轴添加时间范围。

使用 DateAxis 类的 setRange()方法可以为时间轴添加时间范围，语法如下：

```
setRange(Date lower, Date upper)
```

参数说明

❶ lower：表示时间轴的开始时间。

❷ upper：表示时间轴的结束时间。

实例 291　双时间轴位置

光盘位置：光盘\MR\291　　中级　　趣味指数：★★★

实例说明

JFreeChart 可以新增时间轴，新增的时间轴位置可以根据需要进行设置。本实例演示如何将两个时间轴都显示在图表的底部，运行效果如图 9.64 所示。

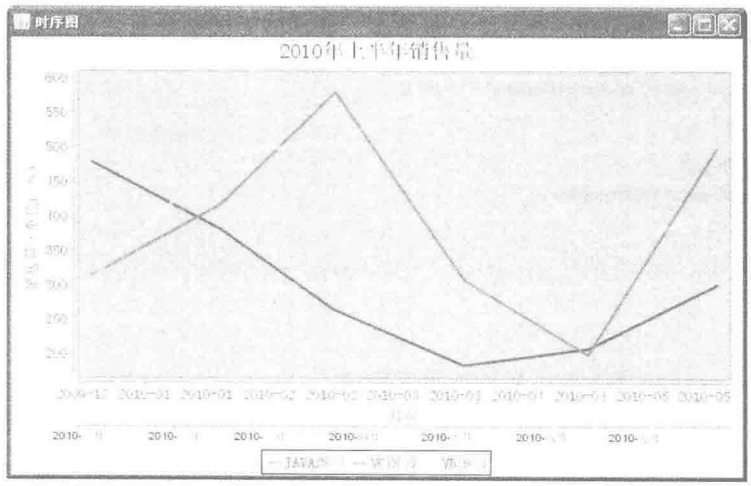


图 9.64　设置时间轴位置

关键技术

使用 XYPlot 类的 setDomainAxisLocation()方法可以为新增的时间轴设置显示位置，语法如下：

```
setDomainAxisLocation(int index, AxisLocation location)
```

参数说明

❶ index：表示时间轴的索引。

❷ location：表示要设置时间轴的显示位置。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 TimeSeriesCollection 类创建一个数据集，代码见实例 288。

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象，代码见实例 288。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体，代码见实例 288。

（5）创建 updatePlot()方法，在方法中获取一个 XYPlot 实例，为图表新增一个时间轴，并且设置时间轴在图表底部或者左侧，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //分类图表
    XYPlot xyPlot = chart.getXYPlot();
    DateAxis dateAxis = (DateAxis) xyPlot.getDomainAxis();
    //设置显示格式
    dateAxis.setDateFormatOverride(new SimpleDateFormat("yyyy-MM"));

    //添加时间轴
    DateAxis dateAxis1 = new DateAxis();
    dateAxis1.setRange(new Month(1, 2010).getStart(), new Month(7, 2010).getEnd());
    dateAxis1.setDateFormatOverride(new SimpleDateFormat("yyyy-MMM"));
    xyPlot.setDomainAxis(1,dateAxis1);
    //设置时间轴位置
    xyPlot.setDomainAxisLocation(1, AxisLocation.BOTTOM_OR_LEFT);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 289。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    TimeSeriesDemo4 demo = new TimeSeriesDemo4("时序图");
    demo.createPlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 291：AxisLocation 类中时间轴的位置常量。

在 AxisLocation 类中定义了时间轴的位置常量，AxisLocation.BOTTOM_OR_LEFT 表示时间轴位于图表底部或者左侧；AxisLocation.TOP_OR_RIGHT 表示时间轴位于图表顶部或者右侧。

实例 292 动态显示十字标记

光盘位置：光盘\MR\292

高级

趣味指数：★★★★

实例说明

绘制完时序图，单击 X 轴与 Y 轴刻度交汇处时，图表上会显示出十字标记。本实例演示如何在单击处显示 X、Y 轴交叉标记，运行效果如图 9.65 所示。

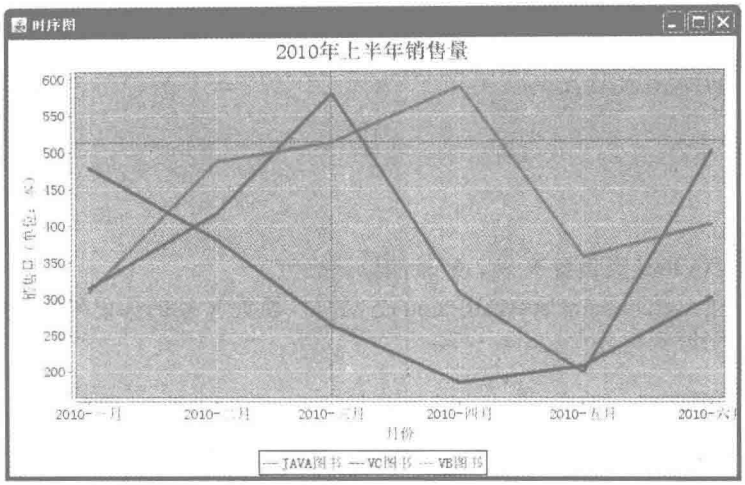


图 9.65　显示十字标记

关键技术

（1）使用 XYPlot 类的 setDomainCrosshairVisible()方法可以显示出时间轴上的标记，语法如下：

```
setDomainCrosshairVisible(boolean flag)
```

参数说明

flag：表示是否显示 X 轴标记，flag 为 true 时，表示显示 X 轴标记；flag 为 false 时，表示不显示 X 轴标记。

（2）使用 XYPlot 类的 setRangeCrosshairVisible()方法可以显示出范围轴上的标记，语法如下：

```
setRangeCrosshairVisible (boolean flag)
```

参数说明

flag：表示是否显示 Y 轴标记，flag 为 true 时，表示显示 Y 轴标记；flag 为 false 时，表示不显示 Y 轴标记。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 TimeSeriesCollection 类创建一个数据集，代码见实例 288。

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象，代码见实例 288。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体，代码见实例 288。

（5）创建 updatePlot()方法，在方法中获取一个 XYPlot 实例，绘制图表根据单击情况显示十字标记，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        XYPlot xyPlot = chart.getXYPlot();
        //显示十字
        xyPlot.setDomainCrosshairVisible(true);
        xyPlot.setRangeCrosshairVisible(true);
        DateAxis dateAxis = (DateAxis) xyPlot.getDomainAxis();
        //设置显示格式
        dateAxis.setDateFormatOverride(new SimpleDateFormat("yyyy-MMM"));
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 289。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        TimeSeriesDemo5 demo = new TimeSeriesDemo5("时序图");
```

```
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 292：通过 XYPlot 类设置 X 轴、Y 轴的标记颜色。

（1）使用 XYPlot 类的 setDomainCrosshairPaint()方法可以设置 X 轴的标记颜色，语法如下：

```
setDomainCrosshairPaint (Paint paint)
```

参数说明

paint：表示 X 轴标记的颜色。

（2）使用 XYPlot 类的 setRangeCrosshairPaint()方法可以设置 Y 轴的标记颜色，语法如下：

```
setRangeCrosshairPaint(Paint paint)
```

参数说明

paint：表示 Y 轴标记的颜色。

实例 293 添加 Y 轴标记

光盘位置：光盘\MR\293　　中级　　趣味指数：★★

实例说明

绘制时序图时，为 Y 轴添加标记，可通过标记定位图表数据。本实例演示如何为 Y 轴添加标记，运行效果如图 9.66 所示。

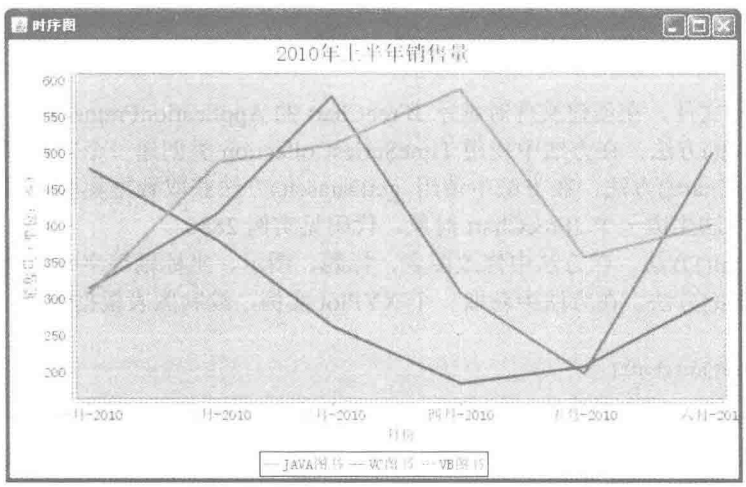


图 9.66　显示 Y 轴标记

关键技术

使用 XYPlot 类的 addRangeMarker()方法可以绘制 Y 轴上的标记，语法如下：

```
addRangeMarker(Marker marker)
```

参数说明

marker：表示 Y 轴上的标记。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 TimeSeriesCollection 类创建一个数据集，代码见实例 288。

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象，代码见实例 288。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体，代码见实例 288。

（5）创建 updatePlot()方法，在方法中获取一个 XYPlot 实例，并且在 Y 轴上添加标记，代码如下：

```
private void updatePlot(JFreeChart chart) {
    //分类图表
    XYPlot xyPlot = chart.getXYPlot();
    //添加 Y 轴标记
    ValueMarker marker = new ValueMarker(450.0);
    marker.setPaint(Color.orange);
    xyPlot.addRangeMarker(marker);
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 289。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
    TimeSeriesDemo6 demo = new TimeSeriesDemo6("时序图");
    demo.createPlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

秘笈心法

心法领悟 293：创建数据轴的标记。

使用 ValueMarker 的构造方法可以创建数据轴的标记实例，语法如下：

```
ValueMarker(double value)
```

参数说明

value：表示 Y 轴标记上的数据值。

实例 294　添加 X 轴标记

光盘位置：光盘\MR\294　　中级　　趣味指数：★★

实例说明

绘制时序图时，为 X 轴添加标记，可通过标记定位时间轴。本实例演示如何为 X 轴添加标记，运行效果如图 9.67 所示。

关键技术

使用 XYPlot 类的 addDomainMarker()方法可以绘制 X 轴上的标记，语法如下：

```
addDomainMarker(Marker marker)
```

参数说明

marker：表示 X 轴上的标记。

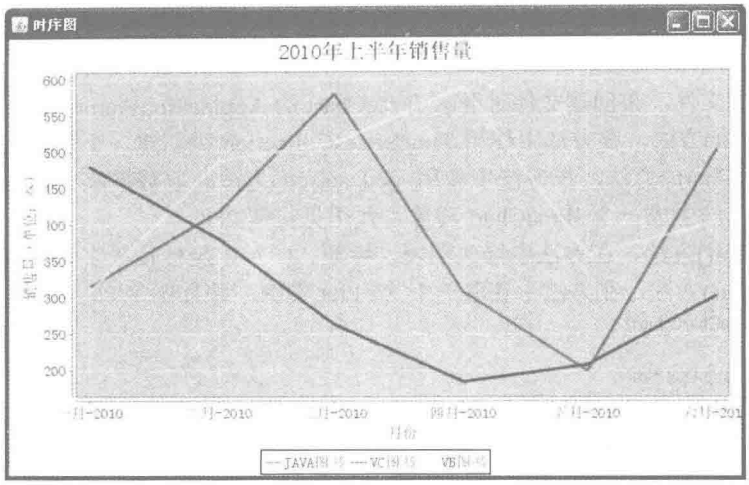


图 9.67　显示 X 轴标记

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 TimeSeriesCollection 类创建一个数据集，代码见实例 288。

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象，代码见实例 288。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体，代码见实例 288。

（5）创建 updatePlot()方法，在方法中获取一个 XYPlot 实例，并且在 X 轴上添加标记，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        XYPlot xyPlot = chart.getXYPlot();
        //在 X 轴添加季度标记
        Quarter quarter = new Quarter(2, 2010);
        ValueMarker marker = new ValueMarker(quarter.getFirstMillisecond());
        marker.setPaint(Color.ORANGE);
        //添加标记
        xyPlot.addDomainMarker(marker);

    }
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 289。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        TimeSeriesDemo7 demo = new TimeSeriesDemo7("时序图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 294：使用 Quarter()方法创建季度实例。

使用 Quarter()构造方法可以创建季度实例，语法如下：

```
Quarter(int quarter, int year)
```

参数说明

❶ quarter：表示季度数值。

❷ year：表示年度数值。

实例 295　设置刻度单位

光盘位置：光盘\MR\295　　中级　　趣味指数：★★★★

实例说明

绘制时序图时，可以根据不同需要为 X 轴设置不同的刻度单位。本实例演示如何设置 X 轴的刻度单位，运行效果如图 9.68 所示。

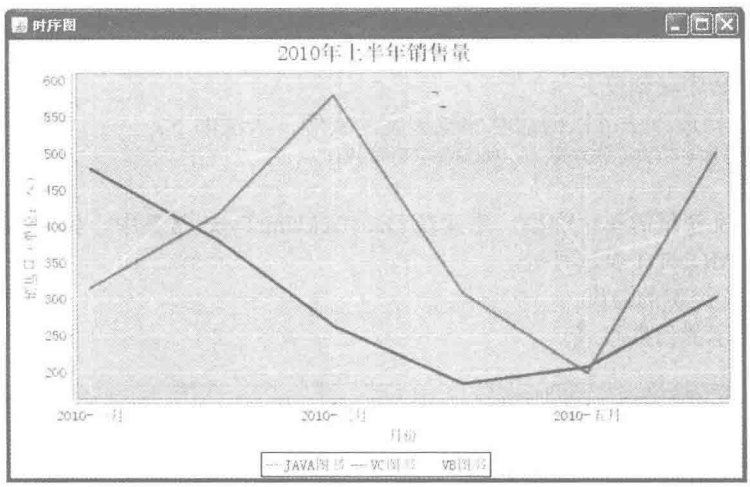


图 9.68　设置刻度单位

关键技术

使用 DateAxis 类的 setTickUnit()方法可以设置 X 轴上的刻度单位，语法如下：

```
setTickUnit(DateTickUnit unit)
```

参数说明

unit：是一个日期类型值，表示 X 轴刻度单位的实例。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 TimeSeriesCollection 类创建一个数据集，代码见实例 288。

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象，代码见实例 288。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体，代码见实例 288。

（5）创建 updatePlot()方法，在方法中获取一个 XYPlot 实例，同时设置刻度单位，代码如下：

```
private void updatePlot(JFreeChart chart) {
        //分类图表
        XYPlot xyPlot = chart.getXYPlot();
          //设置单位
          final DateAxis axis = (DateAxis) xyPlot.getDomainAxis();
          //设置刻度单位
```

```
            axis.setTickUnit(new DateTickUnit(DateTickUnitType.MONTH, 2,
                    new SimpleDateFormat("yyyy-MMM")));
    }
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 289。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        TimeSeriesDemo8 demo = new TimeSeriesDemo8("时序图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
    }
```

秘笈心法

心法领悟 295：创建时序轴刻度。

使用 DateTickUnit 的构造方法可以创建时序轴刻度的实例，语法如下：

```
DateTickUnit(DateTickUnitType unitType, int multiple, DateFormat formatter)
```

参数说明

❶ unitType：表示刻度使用的单位类型，定义在 DateTickUnitType 常量中，如 DateTickUnitType.YEAR 表示年、DateTickUnitType.MONTH 表示月等。

❷ multiple：表示单位类型的跨度。

❸ formatter：表示显示的日期格式。

实例 296 设置 X 轴范围

光盘位置：光盘\MR\296　　中级　　趣味指数：★★★

实例说明

绘制 JFreeChart 时序图时，可以根据需要设置 X 轴的取值范围，本实例演示如何设置 X 轴的范围，运行效果如图 9.69 所示。

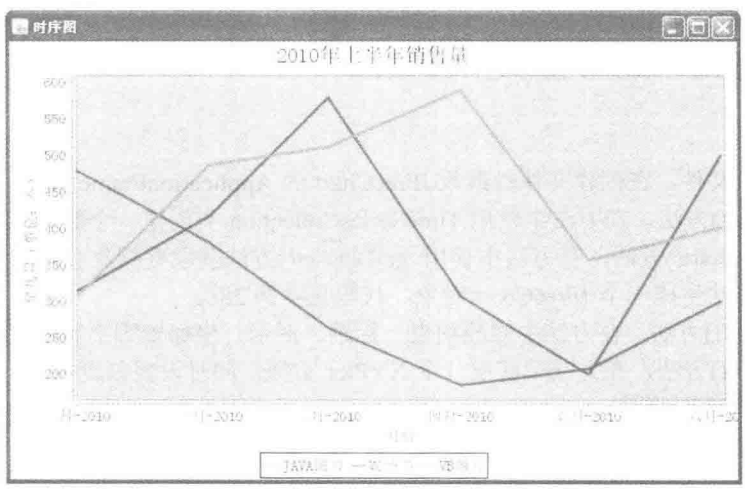


图 9.69　设置 X 轴范围

关键技术

使用 DateAxis 类的 setRange()方法可以设置 X 轴的显示范围，语法如下：

```
setRange(Date lower, Date upper)
```

参数说明

❶ lower：表示 X 轴的开始时间。

❷ upper：表示 X 轴的结束时间。

设计过程

（1）新建一个 Java 文件，在创建文件时继承 JFreeChart 的 ApplicationFrame 类。

（2）创建 getDataset()方法，在方法中使用 TimeSeriesCollection 类创建一个数据集，代码见实例 288。

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用 ChartFactory 类的 createTimeSeriesChart()方法生成一个 JFreeChart 对象，代码见实例 288。

（4）创建 updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体，代码见实例 288。

（5）创建 updatePlot()方法，在方法中创建两个 Day 类型，设置 X 轴的时间范围，代码如下：

```
private void updatePlot(JFreeChart chart) {
        Day endDay = new Day(1,6,2010);
                SerialDate serialDate = SerialDate.addMonths(-5, endDay.getSerialDate());
        Day beginDay = new Day(serialDate.toDate());
        DateAxis axis = (DateAxis) chart.getXYPlot().getDomainAxis();
        //设置 X 轴开始和结束位置
        axis.setRange(beginDay.getStart(), endDay.getEnd());
}
```

（6）创建 createPlot()方法，在方法中获取一个 JFreeChart 对象，然后修改图表字体与图表设置，最后调用父类的 setContentPane()方法把 JFreeChart 对象保存在窗体面板中，代码见实例 289。

（7）在 main()方法中调用 createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
        TimeSeriesDemo8 demo = new TimeSeriesDemo8("时序图");
        demo.createPlot();
        demo.pack();
        //把窗体显示到显示器中央
        RefineryUtilities.centerFrameOnScreen(demo);
        //设置可以显示
        demo.setVisible(true);
}
```

秘笈心法

心法领悟 296：使用 addMonths()方法做运算。

使用 SerialDate 类的 addMonths()方法可以对月进行加、减法计算，语法如下：

```
addMonths(int months, SerialDate base)
```

参数说明

❶ months：需要增加或者减少的月数，如果 months 为正数，则增加月数；如果 months 为负数，则减少月数。

❷ base：表示需要进行计算的日期。

XML 篇

第10章

初识 XML

- XML 语言基础
- XML 与 CSS
- XML 与 XSLT 的元素
- XML 与 XSLT 的内建函数
- DTD 的引用与验证
- 使用 DTD 定义 XML 元素
- 使用 DTD 定义 XML 属性

10.1　XML 语言基础

实例 297　简单的 XML

光盘位置：光盘\MR\297　　中级　　趣味指数：★★

实例说明

在项目开发阶段经常会用到 XML。XML 是一种简单的数据存储语言，使用一系列简单的标记描述数据，而这些标记可以用方便的方式建立。虽然 XML 占用的空间比二进制数据多，但 XML 易于掌握和使用。本实例创建一个简单的 XML，如图 10.1 所示。

图 10.1　简单的 XML

关键技术

本实例的主要目的是通过简单的方式建立一个简单的 XML，以让读者更快地熟悉 XML，知道 XML 是数据存储的一种有规律的方式。通过实例可以了解到在定义 XML 的格式时，元素必须配对、元素名称区分大小写等。不但如此，XML 还必须使用正确的嵌套，每个 XML 文档都必须要有根元素。

设计过程

（1）创建一个文本文档。

（2）在文本文档中输入 XML 代码，如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
    <book>
        <name>JAVA 范例完全自学手册</name>
        <publisher>人民邮电出版社</publisher>
        <author>李钟尉</author>
        <ISBN>9787115201027</ISBN>
        <price>59.80</price>
    </book>
</books>
```

（3）保存文本文档，并将其扩展名改为.xml。

秘笈心法

心法领悟 297：隐藏与显示文件的扩展名。

Windows 操作系统在默认情况下文件的扩展名是隐藏的，直接在文件上右击，在弹出的快捷菜单中选择“重

命名”命令是不能修改文件的扩展名的，需要通过修改 Windows 的设置，把文件的扩展名显示出来再修改，方法如下：

（1）双击“我的电脑”图标，打开一个 Windows 窗口，在菜单栏中选择“工具”/“文件夹选项”命令，弹出“文件夹选项”对话框。

（2）在“文件夹选项”对话框中选择“查看”选项卡，在“高级设置”列表框中取消选中“隐藏已知文件类型的扩展名”复选框，然后单击“确定”按钮即可。

说明：XML 中也可以添加注释，与 HTML 的注释类似，使用“<!--”作为开始标签，使用“-->”作为结束标签。

实例 298　验证 XML 的格式

光盘位置：光盘\MR\298　　中级　　趣味指数：★★★

实例说明

本实例实现验证 XML 的格式。使用 IE 浏览器打开错误的 XML 实例，会有如图 10.2 所示的错误提示。

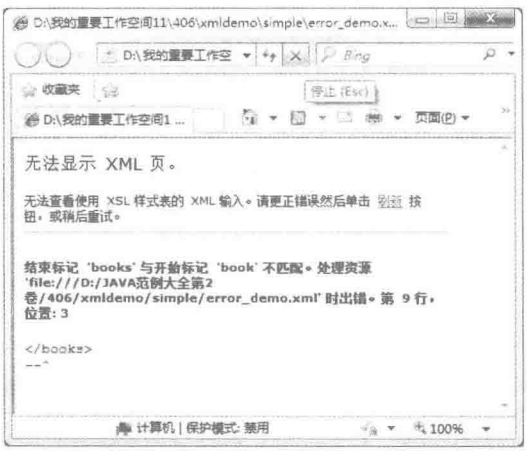


图 10.2　标签不匹配

关键技术

正确的 XML 格式的标签必须是成对出现的，而且每对标签的大小写都必须一致。为了检验 XML 的拼写格式是否有语法错误，最简单的方法是通过浏览器对其进行验证，也可以使用专业的 XML 编辑器进行验证。验证 XML 时，出现不同的格式错误，浏览器会有不同的提示。

设计过程

（1）创建一个 XML 文档。

（2）输入下面的 XML 内容，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
<book>
    <name>JAVA 范例完全自学手册</name>
    <publisher>人民邮电出版社</publisher>
    <author>李钟尉</author>
    <ISBN>9787115201027</ISBN>
```

```
    <price>59.80</price>
</books>
```

（3）打开 IE 浏览器，在菜单栏中选择“文件”/“打开”命令，弹出“打开”对话框。

（4）单击“浏览”按钮找到 XML 文档，选中后单击“确定”按钮打开 XML，在浏览器中可以看到错误提示，如图 10.2 所示。

秘笈心法

心法领悟 298：验证 XML 格式的编辑器。

验证 XML 的格式是否正确不只有使用浏览器一种方法，还可以通过 XML 的编辑器进行验证，网上有很多可视化的 XML 编辑器，如 XML Spy 、XMLEditor、XML Notepad 等，这里不再一一列举。

实例 299　XML 属性的使用

光盘位置：光盘\MR\299　　中级　　趣味指数：★★★★

实例说明

在 XML 的使用中，只含有元素远远满足不了业务需求，所以 XML 的标准中定义每个元素都可以有自己的属性，通过属性能更简洁、更完整地定义 XML。当然在定义 XML 时，更鼓励大家定义元素而不是属性，因为在符合逻辑的情况下尽可能定义 XML 元素，可以让 XML 有更好的格式。本实例定义一个拥有属性的 XML，如图 10.3 所示。

图 10.3　XML 的属性

关键技术

每个元素都可以定义多个属性，在使用属性时，需要用引号把属性值括起来，此时的引号必须在半角状态下输入，这是书写 XML 的规范，具体属性值可以根据实际情况填写。如为 price 定义一个属性 unit，属性值为 RBM 的代码如下：

```
<price unit="RBM" >65.00</price>
```

设计过程

（1）创建一个 XML 文档。

（2）在 XML 文档中输入以下代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
<book>
        <name>C#从入门到精通（第 2 版）</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
</book>
<book>
        <name>JavaScript 开发技术大全</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115179708</ISBN>
        <price unit="RBM" >65.00</price>
</book>
</books>
```

秘笈心法

心法领悟 299：XML 元素属性的特点。

在定义 XML 时，XML 的元素属性是针对当前元素而定义的，一般都是对元素的单一方面的特性进行定义，XML 属性在传输过程中消耗的资源比元素小很多。

实例 300　XML 中 CDATA 的使用

光盘位置：光盘\MR\300　　中级　　趣味指数：★★★

实例说明

使用浏览器解析 XML 时，XML 的元素内容中有时会含有“<”“&”等符号，这时浏览器会把这些特殊符号当作 XML 的定义符号来解析，这就可能造成 XML 的格式错乱。本实例使用 CDATA 把不希望浏览器解析的内容定义出来，如图 10.4 所示。

图 10.4　使用了 CDATA 的 XML

关键技术

CDATA 比较特殊，起始于“<![CDATA[”，终止于“]]>”，凡是使用这对符号修饰的 XML 内容，在解析时都会被跳过。需要注意，CDATA 必须是成对出现的，同时也不能嵌套使用。实例中的关键代码如下所示：

```
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
```

设计过程

（1）创建一个 XML 文档。

（2）在 XML 文档中输入以下代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
<book>
        <name>C#从入门到精通（第 2 版） </name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
<book>
        <name>JavaScript 开发技术大全</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115179708</ISBN>
        <price unit="RBM" >65.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=138&sid=5]]></url>
</book>
<book>
        <name>Java 全能速查宝典</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115214874</ISBN>
        <price unit="RBM" >59</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=191&sid=7]]></url>
</book>
<book>
        <name>视频学 Java Web(1DVD)</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

秘笈心法

心法领悟 300：特殊符号的显示方法。

如果不希望在 XML 内容中使用 CDATA，那么在遇到“<”“&”等特殊符号时，可以使用实体引用替代。如在 XML 元素的内容中遇到“<”时，可以使用“<”替代；遇到“&”时，可以使用“&”替代。各种符号的实体引用示例如表 10.1 所示。

表 10.1　实体引用示例表

实 体 引 用	符　　号	中 文 释 义
&lt;	<	小于
&gt;	>	大于

续表

实 体 引 用	符　　号	中 文 释 义
&amp;	&	和号
&apos;	'	单引号
&quot;	"	引号

说明：在 CDATA 内部使用实体引用，浏览器仍然不会对其进行解析。

10.2　XML 与 CSS

实例 301　在 XML 中使用 CSS

实例位置：光盘\MR\301　　中级　　趣味指数：★★★★

实例说明

XML 在编辑器和浏览器中以树形结构显示，可以通过 CSS 样式对这种展示形式进行修饰，甚至可以修饰成与 HTML 一样的页面。本实例在浏览器中使用 CSS 格式化 XML，效果如图 10.5 所示。

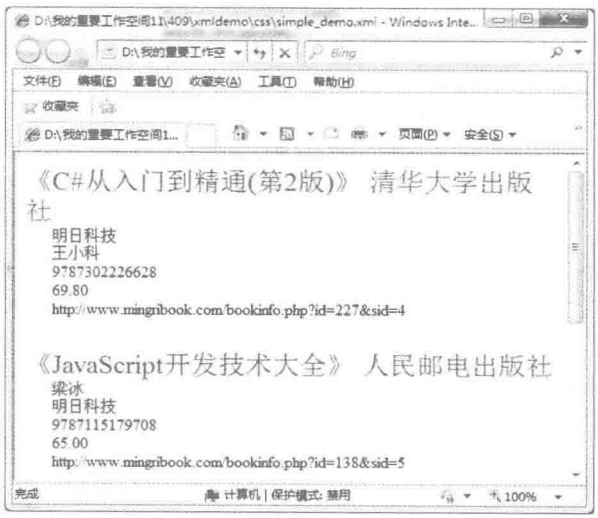


图 10.5　CSS 格式化后的 XML

关键技术

使用 CSS 格式化 XML 时，需要先在 XML 中引用 CSS 样式表，代码如下：

```
<?xml-stylesheet type="text/css" href="simple_demo.css"?>
```

设计过程

（1）分别创建 XML 文档 simple_demo.xml 和 CSS 文档 simple_deom.css。

（2）编辑 CSS 文档，在文档中添加关于 XML 的样式，代码如下：

```
books
{
background-color: #FFFFFF;
width: 100%;
```

```
}
book
{
    display: block;
    margin-bottom: 25pt;
    margin-left: 0;
}
name
{
    color: #FF0000;
    font-size: 20pt;
}
publisher
{
    color: #3333FF;
    font-size: 20pt;
}
company,author,isbn,price,url
{
    display: block;
    color: #000000;
    margin-left: 20pt;
}
```

（3）将 CSS 样式表引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="simple_demo.css"?>
<books>
<book>
    <name>《C#从入门到精通(第 2 版)》</name>
    <publisher>清华大学出版社</publisher>
    <company>明日科技</company>
    <author>王小科</author>
    <ISBN>9787302226628</ISBN>
    <price unit="RBM" >69.80</price>
    <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
……
</books>
```

秘笈心法

心法领悟 301：引入样式表的路径问题。

在 XML 中引用 CSS 样式表时，使用 href="simple_demo.css"表示 simple_demo.css 和 simple_demo.xml 在同一个路径下，href 标签支持相对路径、绝对路径和 URL 多种引用方式。

实例 302　CSS 格式化 XML 布局

光盘位置：光盘\MR\302　　中级　　趣味指数：★★★★

实例说明

本实例在 XML 中引入 CSS，通过 CSS 对 XML 文档的布局进行格式化，运行效果如图 10.6 所示。

图 10.6　CSS 格式化 XML 的布局

关键技术

使用 CSS 格式化 XML 的布局时，可以使用 display。display 的两种常用属性为 block 和 inline，使用方法如下所示：

```
book
{
    display: block|inline;
}
```

参数说明

❶ block：其作用是将要修饰的元素自成一个区域，被 block 修饰过的元素前后都有换行。

❷ inline：是 display 的默认值，被显示成一行元素，前后没有换行，被 inline 修饰的元素按原文件顺序依次排列下去。

设计过程

（1）先建立一个规范的 XML 文件，然后在 XML 文档中引入 CSS 文档，如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="display_demo.css"?>
<books>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
<book>
        <name>《JavaScript 开发技术大全》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115179708</ISBN>
        <price unit="RBM" >65.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=138&sid=5]]></url>
</book>
<book>
        <name>《Java 全能速查宝典》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115214874</ISBN>
        <price unit="RBM" >59.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=191&sid=7]]></url>
</book>
<book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

（2）根据 XML 引入的 CSS 文档名称和路径建立 CSS，在 CSS 中使用 display:block 修饰 book 元素，使 XML 文档中的各个 book 的内容自成一行分开显示，在整体上划分 XML 文档。

（3）使用 display:block 修饰 name 元素，这样在每个 book 内部的 name 也和其他元素相对独立出来，使 name 的元素也独自成一行。

（4）book 内部的其他元素都使用 display:inline 默认的样式修饰即可，使其都显示在一行，和 name 元素形成对比，以突出 name 的内容。

秘笈心法

心法领悟 302：display 属性的作用。

使用 display 属性可以修饰 XML 元素自成一个区域，也可以修饰 XML 中多个元素共成一个区域，通过使用 display，可以让 XML 的各种元素结构变得更加清晰。

实例 303　CSS 格式化 XML 背景色

光盘位置：光盘\MR\303　　中级　　趣味指数：★★★★

实例说明

为了更清晰地说明 CSS 是如何格式化 XML 元素的背景的，本实例将每一个元素都加上背景色。在修饰元素背景色时，先将 XML 的布局进行格式化，然后再对元素添加背景颜色，运行效果如图 10.7 所示。

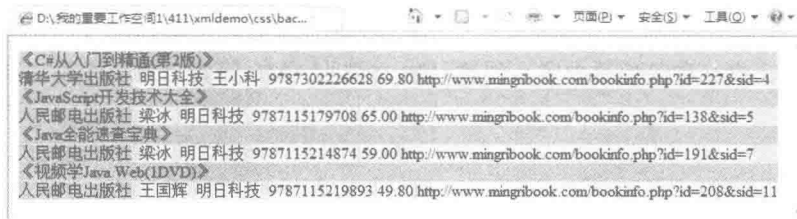


图 10.7　CSS 格式化 XML 的背景色

关键技术

使用 CSS 格式化 XML 背景色时，在需要使用背景的元素上使用 background-color 的属性，属性值可以是十六进制的颜色标识，如#CCCCFF 或#1CC 等。CSS 代码示例如下：

```
company,author,isbn,price,url
{
background-color: #CCCCFF;
}
```

本实例中，book 元素的背景色使用#3F0；name 元素的背景色使用#1CC；publisher 元素的背景色使用#CC9999；company、author、isbn、price 和 url 等元素的背景色使用#CCCCFF。

设计过程

（1）先建立 XML 文件，然后引入 CSS 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="background_demo.css"?>
<books>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
```

```
</book>
<book>
        <name>《JavaScript 开发技术大全》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115179708</ISBN>
        <price unit="RBM" >65.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=138&sid=5]]></url>
</book>
<book>
        <name>《Java 全能速查宝典》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115214874</ISBN>
        <price unit="RBM" >59.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=191&sid=7]]></url>
</book>
<book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

（2）根据 XML 的元素名称等信息内容，创建 CSS 文档为 XML 元素，并且设置背景色，代码如下：

```
book
{
    display: block;
}
name
{
    display: block;
}
publisher,company,author,ISBN,price,url
{
    display: inline;
}
book
{
background-color: #3F0;
}
name
{
background-color: #1CC;
}
publisher
{
background-color: #CC9999;
}
company,author,isbn,price,url
{
background-color: #CCCCFF;
}
```

秘笈心法

心法领悟 303：CSS 格式化背景色。

使用 CSS 格式化 XML 背景色时，如果子元素没有设置背景色，页面上会显示父元素的背景色；如果子元

素有背景色，则会显示子元素的背景色，同时在子元素的有效范围内子元素会覆盖父元素的背景色。

实例 304　CSS 格式化 XML 字体

光盘位置：光盘\MR\304　　中级　　趣味指数：★★★★

实例说明

页面里经常使用不同字体样式和字体大小，使用 CSS 格式化字体是很普遍的，XML 中的字体可以以元素为单位进行格式化，运行效果如图 10.8 所示。

图 10.8　CSS 格式化 XML 字体

关键技术

修饰字体经常用到 font-size、font-style 等 CSS 样式，通过这些样式来改变字体的显示形式。下面是 CSS 格式化字体样式的示例，字体大小默认单位是 px（像素）：

```
book
{
//设置字体大小
 font-size: 15px
}
name
{
 font-size: 30px;
//使用倾斜字体
 font-style: italic;
}
publisher
{
 font-size: 24px;
//字体下加下划线
 text-decoration: underline;
}
```

设计过程

（1）建立 XML 文件，在 XML 中引入 CSS 样式文件，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="font_demo.css"?>
<books>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <author>王小科</author>
        <company>明日科技</company>
```

```
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
    </book>
    <book>
        <name>《JavaScript 开发技术大全》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115179708</ISBN>
        <price unit="RBM" >65.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=138&sid=5]]></url>
    </book>
    <book>
        <name>《Java 全能速查宝典》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115214874</ISBN>
        <price unit="RBM" >59.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=191&sid=7]]></url>
    </book>
    <book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
    </book>
    </books>
```

（2）引入 CSS 以后，为 book 的元素添加 font-size: 15px，这时所有 book 下面的字体大小都为 15px。然后在 name 的元素上添加字体样式 font-size: 30px、font-style: italic，name 元素被单独修饰后，字体样式不再受 book 的约束，字体大了 15 像素，同时变为倾斜样式。最后 publisher 也被单独修饰为 font-size: 24px、text-decoration: underline，其样式也不受 book 的字体样式限制，字体比以前大了 9 像素，同时多了下划线，代码如下：

```
book
{
 display: block;
font-size: 15px
}
name
{
 display: block;
font-size: 30px;
font-style: italic;
}
publisher{
 display: inline;
font-size: 24px;
text-decoration: underline;
}
company,author,ISBN,price,url
{
    display: inline;
}
```

秘笈心法

心法领悟 304：CSS 格式化字体样式。

添加字体样式时，在子元素没有设置字体样式的情况下，页面上会套用父元素的字体样式；如果子元素有自己的字体样式时，XML 会显示子元素的字体样式，该特性和使用 CSS 格式化 XML 的背景色是一样的。

实例 305　CSS 改变 XML 中的鼠标手势

光盘位置：光盘\MR\305　　中级　　趣味指数：★★★★★

实例说明

鼠标手势在不同情况下表示不同的含义，在 XML 中也可以通过 CSS 让每个元素在网页里显示出不同的鼠标手势，效果如图 10.9 所示。

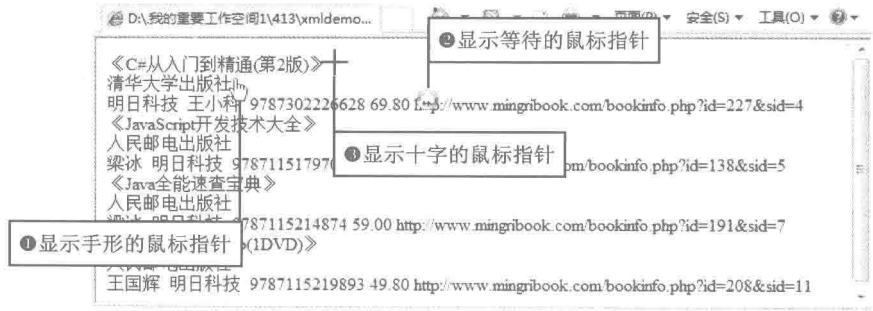


图 10.9　CSS 改变 XML 中的鼠标手势

关键技术

鼠标手势是以 XML 的元素为基础来定义的，可以为不同的元素定义不同的手势，如果子元素定义了手势，在子元素的覆盖范围内父元素定义的手势则不起作用，代码如下：

```
name
{
cursor:crosshair;
}
publisher
{
cursor:hand;
}
company,author,isbn,price,url
{
cursor:wait;
}
```

参数说明

❶ cursor:crosshair：表示鼠标指针放在该元素上显示十字形状。

❷ cursor:hand：表示鼠标指针放在该元素上显示手的形状。

❸ cursor:wait：表示鼠标指针放在该元素上显示等待的形状。

设计过程

（1）建立 XML 文件，并引入 CSS 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="cursor_demo.css"?>
<books>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <author>王小科</author>
        <company>明日科技</company>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
```

```
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
    </book>
    <book>
        <name>《JavaScript 开发技术大全》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115179708</ISBN>
        <price unit="RBM" >65.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=138&sid=5]]></url>
    </book>
    <book>
        <name>《Java 全能速查宝典》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115214874</ISBN>
        <price unit="RBM" >59.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=191&sid=7]]></url>
    </book>
    <book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
    </book>
    </books>
```

（2）引入 CSS 以后，在 name 的元素上面添加 cursor:crosshair，这时所有 name 元素的覆盖范围鼠标指针都是十字形状；在 publisher 的元素上添加 cursor:hand，则 publisher 元素所在的范围鼠标指针显示手的形状；在 company、author、ISBN、price 和 url 元素上添加 cousor:wait，那么鼠标指针放在这些元素上时都显示为沙漏状，代码如下：

```
book
{
    display: block;
}
name
{
    display: block;
}
publisher
{
    display: block;
}
company,author,ISBN,price,url
{
display: inline;
}
name
{
cursor:crosshair;
}
publisher
{
cursor:hand;
}
company,author,isbn,price,url
{
cursor:wait;
}
```

秘笈心法

心法领悟 305：CSS 修饰多个元素为相同样式。

在使用 CSS 添加样式时，如果多个元素使用相同的样式修饰，可以把元素名称写在一起，中间以半角的逗号分隔。

实例 306　CSS 在 XML 中添加背景图

光盘位置：光盘\MR\306　　中级　　趣味指数：★★★★★

实例说明

通过 CSS 为 XML 添加背景图片可以使页面更加丰富，给用户的感觉更直观，可以更有效地表达页面的主题思想，相比没有背景图片的页面，给用户更深层次的感受。本实例实现添加背景图片的页面，运行效果如图 10.10 所示。

图 10.10　CSS 为 XML 添加背景图片

关键技术

添加背景图片在 CSS 中也是以元素为单位来完成的，在设置背景图片时，要同时设置好图片的长度和宽度，使用 background-image:url()即可为 XML 元素指定图片，代码如下：

```
pic
{
background-image: url(background-image_demo.jpg);
width:153px;
height:193px;
}
```

参数说明

❶ background-image_demo.jpg：背景图片的路径和名称。

❷ width:153px；height:193px：pic 元素的宽度和长度，预留图片需要显示的位置。

设计过程

（1）建立 XML 文件，并引入 CSS 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="background-image_demo.css "?>
<books>
<book>
        <pic></pic>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <author>王小科</author>
        <company>明日科技</company>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
<book>
```

```
        <name>《JavaScript 开发技术大全》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115179708</ISBN>
        <price unit="RBM" >65.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=138&sid=5]]></url>
</book>
<book>
        <name>《Java 全能速查宝典》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115214874</ISBN>
        <price unit="RBM" >59.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=191&sid=7]]></url>
</book>
<book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

注意：此时在<name>《C#从入门到精通(第 2 版)》</name>元素上面添加了一个元素 pic。

（2）引入 CSS 以后，将准备好的图片放在相应的目录下，并查看图片的高度和宽度。然后设置 CSS 样式中 pic 的宽、高与图片的宽、高一致，设置完成以后，就可以在 pic 中把背景图片引入进来，代码如下：

```
book
{
    display: block;
}
name
{
    display: block;
}
publisher,company,author,ISBN,price,url
{
    display: inline;
}
pic
{
background-image: url(background-image_demo.jpg);
width:153px;
height:193px;
}
```

秘笈心法

心法领悟 306：元素中可以不添加元素内容。

在显示图片时，为了不让图片被文字遮住影响效果，pic 的元素中允许不添加内容。

实例 307 CSS 制作 XML 表格

中级

光盘位置：光盘\MR\307

趣味指数：★★★★★

实例说明

XML 本身的规则性很强，实际使用中，经常会把 XML 的数据以表格的形式展示出来，通过 CSS 把 XML

格式化成表格也是一种方式。本实例使用 CSS 把 XML 格式化成一个简单的表格，效果如图 10.11 所示。

图 10.11　CSS 制作 XML 表格

关键技术

使用 CSS 的 border-style:outset 可以显示表格的边框，代码如下：

```
book
{
border-style:outset;
}
```

设计过程

（1）建立 XML 文件，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="table_demo.css"?>
<books>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <author>王小科</author>
        <company>明日科技</company>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
<book>
        <name>《JavaScript 开发技术大全》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115179708</ISBN>
        <price unit="RBM" >65.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=138&sid=5]]></url>
</book>
<book>
        <name>《Java 全能速查宝典》</name>
        <publisher>人民邮电出版社</publisher>
        <author>梁冰</author>
        <company>明日科技</company>
        <ISBN>9787115214874</ISBN>
        <price unit="RBM" >59.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=191&sid=7]]></url>
</book>
<book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
```

```
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
    </book>
    </books>
```

（2）引入 CSS 以后，先在 book 的元素上使用 display:block 属性，让 book 的元素按 book 模块排开，然后把 book 的边框进行修饰，这里添加边框样式、设置边框颜色、调整边框宽度，代码如下：

```
book
{
display: block;
border-width: 1px;
border-color: #930;
border-style:outset;
background-color: #CCC
}
```

（3）为 name 元素添加 display:block 属性，使其单独成一行，并把 name 的文本居中显示。代码如下：

```
name
{
display: block;
text-align: center;
}
```

（4）把 publisher、company、author、ISBN 和 price 元素居中显示，同时设置各元素宽度，url 的内容比较多，如果和 publisher 等元素放在一行显示会影响美观，所以把 url 元素加上 display:block 属性，让其单独在一行显示。代码如下：

```
publisher,company,author,ISBN,price
{
text-align: center;
width: 18%
}
url
{
display: block;
}
```

秘笈心法

心法领悟 307：使用 CSS 制作表格时的距离设置。

使用 CSS 制作表格时，多个元素连在一行显示时很难把每个元素区分开来，所以在显示时让元素之间保持一定的距离，这个距离可以使用相对值 width:18%的方式来控制，同时也可以使用 px、pt 等带有单位的绝对数值来控制。

10.3 XML 与 XSLT 的元素

实例 308	在 XML 中使用 XSLT 光盘位置：光盘\MR\308	中级 趣味指数：★★★

实例说明

XSLT 是 XSL 的一部分，其本身也是一种 XML，同时又可以把普通的 XML 转换成 XHTML。使用支持 XSLT 的浏览器就可以把 XML 转换成 XHTML 显示出来，运行效果如图 10.12 所示。

关键技术

要想正常地用 XSLT 把 XML 转换成 XHTML，需要浏览器支持 XSLT，现在主流浏览器都能做到这一点；同时还要在 XML 中正确引用 XSL 文件，使用 stylesheet 和 transform 都可以声明 XSLT，在使用 XSLT 时，二

者没有区别。如下所示：

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

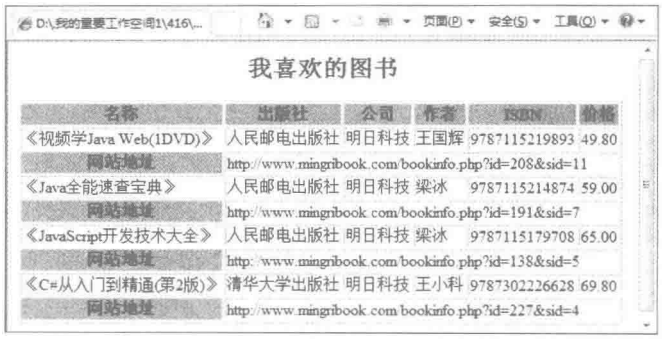


图 10.12　使用 XSLT 转换 XML

设计过程

（1）先创建一个 XSLT 文档，在 XSLT 文档中添加 XML 的样式，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
   <h2 align="center">我喜欢的图书</h2>
   <table border="1">
     <tr bgcolor="#55A0FF">
       <th>名称</th>
       <th>出版社</th>
       <th>公司</th>
       <th>作者</th>
       <th>ISBN</th>
       <th>价格</th>
     </tr>
     <xsl:for-each select="books/book">
     <xsl:sort select="price"/>
     <tr>
       <td><xsl:value-of select="name"/></td>
       <td><xsl:value-of select="publisher"/></td>
       <td><xsl:value-of select="company"/></td>
       <td><xsl:value-of select="author"/></td>
       <td><xsl:value-of select="ISBN"/></td>
       <td><xsl:value-of select="price"/></td>
     </tr>
     <tr>
       <th bgcolor="#55A0FF" align="center">网站地址</th>
       <td colspan="5"><xsl:value-of select="url"/></td>
     </tr>
     </xsl:for-each>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

（2）创建 XML 文档，把 XSLT 样式表引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="simple_demo.xsl"?>
<books>
<book>
      <name>《C#从入门到精通(第 2 版)》</name>
      <publisher>清华大学出版社</publisher>
```

```
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
    </book>
    //省略部分代码
    <book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
    </book>
    </books>
```

秘笈心法

心法领悟 308：引入 XSLT 文件时的路径问题。

在 XML 引用 XSLT 文件时使用 href="simple_demo.xsl"，表示 simple_demo. xsl 和 simple_demo.xml 在同一个路径下，href 标签支持相对路径、绝对路径和 URL 多种引用方式。

实例 309 使用 XSLT 中的 template

中级

光盘位置：光盘\MR\309

趣味指数：★★★★

实例说明

当希望某一个节点匹配一个模板时使用 template，match 属性用于把模板关联到某个 XML 元素，该元素就是一个节点，<xsl:template>与</xsl:template>之间的内容就是一个模板。本实例使用 template 创建一个表格的模板，运行效果如图 10.13 所示。

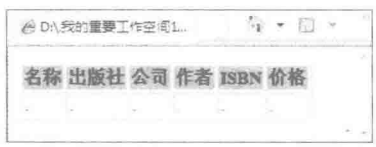


图 10.13　使用 template 创建一个模板

关键技术

使用<xsl:template>制定模板时，首先需要把<xsl:template>中的内容使用 match 与 XML 文档的元素联系起来，示例如下：

```
<xsl:template match="/">
```

参数说明

Match 中的路径就是 XML 元素的级别。

设计过程

（1）先创建一个 XSL 文档，引入<xsl:stylesheet>或<xsl:transform>标签，然后在 XSLT 中定义一个模板<xsl:template>，并在其内部添加模板的样式，同时在制定模板时可以使用 html 元素，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
```

```
<html>
 <body>
   <table border="1">
     <tr bgcolor="#55A0FF">
       <th>名称</th>
       <th>出版社</th>
       <th>公司</th>
       <th>作者</th>
       <th>ISBN</th>
       <th>价格</th>
     </tr>
     <tr>
       <td>.</td>
       <td>.</td>
       <td>.</td>
       <td>.</td>
       <td>.</td>
       <td>.</td>
     </tr>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

（2）创建 XML 文档，把 XSLT 样式表引入 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="simple_demo.xsl"?>
<books>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
//省略部分代码
<book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

秘笈心法

心法领悟 309：编辑 HTML 的经验。

在定义模板时，如果模板直接编辑 html 元素，错误率会很高，而且非常不方便。此时可以在可视化的编辑器（如 Adobe Dreamweaver 等）中先把模板编辑出来，再复制到<xsl:template>，以达到事半功倍的效果。

实例 310　使用 XSLT 中的 value-of

中级

光盘位置：光盘\MR\310

趣味指数：★★★

实例说明

使用 XSLT 转换 XML 最基本的目的是把 XML 文档的内容提取出来显示在浏览器中，这时就需要用到

<xsl:value-of />。本实例使用 value-of 取出 XML 中的元素，运行效果如图 10.14 所示。

图 10.14　使用 value-of 取出 XML 中的元素

关键技术

使用 XSLT 中的 value-of 可以根据 XML 的元素节点取值显示内容，每级节点中间使用“/”分开，示例如下：

```
<xsl:value-of select="books/book/name"/>>
```

属性说明

select：表示需要获取的 XML 节点的内容。

设计过程

（1）创建一个 XSLT 文档，在文档中绘制一个 1 行 6 列的表格，然后在<td>内部添加<xsl:value-of>标签，再把 XML 的元素写在<xsl:value-of>标签的 select 属性中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
   <table border="1">
     <tr>
       <td><xsl:value-of select="books/book/name"/></td>
       <td><xsl:value-of select="books/book/publisher"/></td>
       <td><xsl:value-of select="books/book/company"/></td>
       <td><xsl:value-of select="books/book/author"/></td>
       <td><xsl:value-of select="books/book/ISBN"/></td>
       <td><xsl:value-of select="books/book/price"/></td>
     </tr>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，再把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="value-of_demo.xsl"?>
<books>
<book>
      <name>《C#从入门到精通(第 2 版)》</name>
      <publisher>清华大学出版社</publisher>
      <company>明日科技</company>
      <author>王小科</author>
      <ISBN>9787302226628</ISBN>
      <price unit="RBM" >69.80</price>
      <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
</books>
```

秘笈心法

心法领悟 310：浏览器是否转义取出内容。

在使用<xsl:value-of>时，如果不希望浏览器对取出的内容进行转义，则使用 disable-output-escaping="yes"，否则使用 disable-output-escaping="no"。例如，如果设置 disable-output-escaping="yes"，则页面直接显示“<”符号不进行转换；如果设置 disable-output-escaping="no"，则 XSLT 会把“<”转换成“<”显示出来。

实例 311　使用 XSLT 中的 for-each

光盘位置：光盘\MR\311　　中级　　趣味指数：★★★★

实例说明

使用<xsl:for-each>可以遍历某个集合下面的节点，如果配合<xsl:value-of>等元素使用，即可把 XML 中的内容全部遍历出来显示在浏览器中，运行效果如图 10.15 所示。

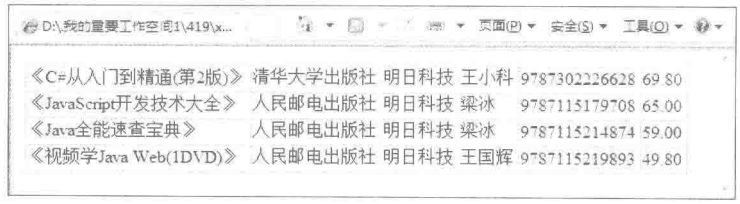


图 10.15　使用 for-each 循环取出 XML 文档内容

关键技术

使用 XML 文档中的 for-each 可以遍历 XML 的元素，代码如下：

```
<xsl:for-each select="books/book">
  <tr>
    <td><xsl:value-of select="name"/></td>
    <td><xsl:value-of select="publisher"/></td>
    <td><xsl:value-of select="company"/></td>
    <td><xsl:value-of select="author"/></td>
    <td><xsl:value-of select="ISBN"/></td>
    <td><xsl:value-of select="price"/></td>
  </tr>
 </xsl:for-each>
```

参数说明

select：表示指定需要遍历的 XML 元素名称，每级元素中间使用“/”分开。

设计过程

（1）创建一个 XSLT 文档，在其中绘制一个 1 行 6 列的表格，然后使用<xsl:for-each/>遍历标签，把<tr>标签包含进来，这样<xsl:for-each>每次遍历时，同时会重复<tr>中的内容。再在<td>内部添加<xsl:value-of>标签，并把 XML 的元素写在<xsl:value-of>标签的 select 属性中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
   <table border="1">
   <xsl:for-each select="books/book">
     <tr>
       <td><xsl:value-of select="name"/></td>
       <td><xsl:value-of select="publisher"/></td>
       <td><xsl:value-of select="company"/></td>
       <td><xsl:value-of select="author"/></td>
       <td><xsl:value-of select="ISBN"/></td>
       <td><xsl:value-of select="price"/></td>
     </tr>
    </xsl:for-each>
   </table>
 </body>
 </html>
```

```
</xsl:template>
    </xsl:stylesheet>
```

（2）建立一个 XML 文档，把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="for-each_demo.xsl"?>
<books>
<book>
    <name>《C#从入门到精通(第 2 版)》</name>
    <publisher>清华大学出版社</publisher>
    <company>明日科技</company>
    <author>王小科</author>
    <ISBN>9787302226628</ISBN>
    <price unit="RBM" >69.80</price>
    <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
//省略部分代码
<book>
    <name>《视频学 Java Web(1DVD)》</name>
    <publisher>人民邮电出版社</publisher>
    <author>王国辉</author>
    <company>明日科技</company>
    <ISBN>9787115219893</ISBN>
    <price unit="RBM" >49.80</price>
    <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

秘笈心法

心法领悟 311：<xsl:for-each/>与<xsl:value-of>配合使用。

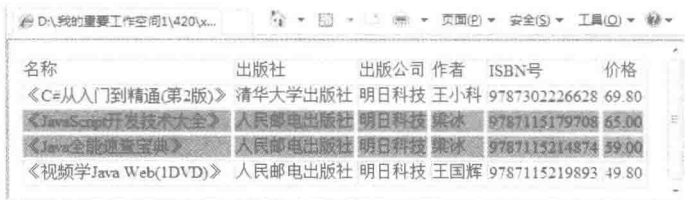
单独使用一个标签不能达到希望的效果时，就需要多个标签配合使用。如<xsl:for-each/>只能起到遍历的作用，如果希望在遍历的同时能把遍历的内容输出到浏览器，就需要配合使用<xsl:value-of>标签。

实例 312　使用 XSLT 中的 if

光盘位置：光盘\MR\312　　中级　　趣味指数：★★★★

实例说明

当要对某元素做一些特殊处理时，就需要把这些元素选择出来，这时可以使用<xsl:if>标签。本实例把所有作者为“梁冰”的一行文字选择出来，并把选择出来的行的背景颜色设置成蓝色，其他作者所在的行设为黄色，运行效果如图 10.16 所示。



名称	出版社	出版公司	作者	ISBN号	价格
《C#从入门到精通(第2版)》	清华大学出版社	明日科技	王小科	9787302226628	69.80
《JavaScript开发技术大全》	人民邮电出版社	明日科技	梁冰	9787115179708	65.00
《Java全能速查宝典》	人民邮电出版社	明日科技	梁冰	9787115214874	59.00
《视频学Java Web(1DVD)》	人民邮电出版社	明日科技	王国辉	9787115219893	49.80

图 10.16　使用 if 判断显示背景颜色

关键技术

使用<xsl:if>标签进行判断，需要在 test 中填写判断规则，而<xsl:if>内部是符合判断条件的内容，代码如下：

```
<xsl:if test="author = '梁冰'">
    <tr bgcolor="#7D7DFF">
      <td><xsl:value-of select="name"/></td>
```

```
        <td><xsl:value-of select="publisher"/></td>
        <td><xsl:value-of select="company"/></td>
        <td><xsl:value-of select="author"/></td>
        <td><xsl:value-of select="ISBN"/></td>
        <td><xsl:value-of select="price"/></td>
      </tr>
      </xsl:if>
```

设计过程

（1）创建一个 XSLT 文档，在其中绘制一个 3 行 6 列的表格。在第 1 个<tr>的<td>中依次填写名称、出版社、出版公司、作者、ISBN 和价格。使用<xsl:for-each/>遍历标签，把第 2 个<tr>和第 3 个<tr>包含进来，然后在第 3 个<tr>前加上判断 author = '梁冰'，将第 2 行的背景色设为蓝色；在第 3 行前加上判断 author!= '梁冰'，将第 3 行的背景色设为黄色，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
   <table border="1">
     <tr>
       <td>名称</td>
       <td>出版社</td>
       <td>出版公司</td>
       <td>作者</td>
       <td>ISBN 号</td>
       <td>价格</td>
     </tr>
   <xsl:for-each select="books/book">
<xsl:if test="author = '梁冰'">
     <tr bgcolor="#7D7DFF">
       <td><xsl:value-of select="name"/></td>
       <td><xsl:value-of select="publisher"/></td>
       <td><xsl:value-of select="company"/></td>
       <td><xsl:value-of select="author"/></td>
       <td><xsl:value-of select="ISBN"/></td>
       <td><xsl:value-of select="price"/></td>
     </tr>
     </xsl:if>
<xsl:if test="author != '梁冰'">
     <tr bgcolor="#FFFF6F">
       <td><xsl:value-of select="name"/></td>
       <td><xsl:value-of select="publisher"/></td>
       <td><xsl:value-of select="company"/></td>
       <td><xsl:value-of select="author"/></td>
       <td><xsl:value-of select="ISBN"/></td>
       <td><xsl:value-of select="price"/></td>
     </tr>
     </xsl:if>
   </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="if_demo.xsl"?>
<books>
<book>
     <name>《C#从入门到精通(第 2 版)》</name>
     <publisher>清华大学出版社</publisher>
     <company>明日科技</company>
     <author>王小科</author>
     <ISBN>9787302226628</ISBN>
```

```
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
    </book>
    //省略部分代码
    <book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
    </book>
    </books>
```

秘笈心法

心法领悟 312：XSTL 转换 XML 时元素的位置。

XSLT 转换 XML 时，XSLT 中显示的元素与其在 XML 中的位置无关，还可以根据需要自行取舍。如本实例的 XML 元素的顺序是 name、publisher、author、company、ISBN、price、url，而在 XSLT 转换时，输出的顺序却是 name、publisher、company、author、ISBN、price。

实例 313 使用 XSLT 中的 sort

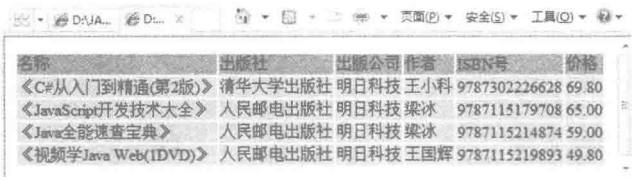
中级

光盘位置：光盘\MR\313

趣味指数：★★★

实例说明

在 XSLT 输出结果时，默认情况下输出的排列顺序和 XML 中存储的顺序一致，但是在实际应用中，有时希望显示结果可以按一定的规则有序地显示，通过<xsl:sort>可以对结果进行排序。本实例在图书列表中根据价格由高到低进行排列，运行效果如图 10.17 所示。



名称	出版社	出版公司	作者	ISBN号	价格
《C#从入门到精通(第2版)》	清华大学出版社	明日科技	王小科	9787302226628	69.80
《JavaScript开发技术大全》	人民邮电出版社	明日科技	梁冰	9787115179708	65.00
《Java全能速查宝典》	人民邮电出版社	明日科技	梁冰	9787115214874	59.00
《视频学Java Web(1DVD)》	人民邮电出版社	明日科技	王国辉	9787115219893	49.80

图 10.17　使用 sort 对结果进行排序

关键技术

<xsl:sort>可以对结果进行排序，但必须与<xsl:for-each>等标签联合使用，示例代码如下：

```
<xsl:sort select="price" order="ascending|descending" case-order="upper-first|lower-first" />
```

参数说明

❶ order：根据字母升、降序排列。使用 ascending，表示按升序排列；使用 descending，表示按降序排列。默认情况下，order 使用 ascending 排序。

❷ case-order：根据字母大、小写排列。使用 upper-first 时，表示大写字母优先排列；使用 lower-first 时，表示小写字母优先排列。

设计过程

（1）创建一个 XSLT 文档，在其中绘制一个 2 行 6 列的表格。在第 1 个<tr>中的<td>依次填写名称、出版社、出版公司、作者、ISBN、价格。然后使用<xsl:for-each>遍历标签，把第 2 个<tr>包含进来，然后在第 2 个

<tr>与<xsl:for-each>之间插入<xsl:sort>标签，在 order 属性上添加 descending，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
   <table>
     <tr bgcolor="#8E8E8E">
       <td>名称</td>
       <td>出版社</td>
       <td>出版公司</td>
       <td>作者</td>
       <td>ISBN 号</td>
       <td>价格</td>
     </tr>
    <xsl:for-each select="books/book">
      <xsl:sort select="price" order="descending" />
     <tr bgcolor="#C2C287">
       <td><xsl:value-of select="name"/></td>
       <td><xsl:value-of select="publisher"/></td>
       <td><xsl:value-of select="company"/></td>
       <td><xsl:value-of select="author"/></td>
       <td><xsl:value-of select="ISBN"/></td>
       <td><xsl:value-of select="price"/></td>
     </tr>
    </xsl:for-each>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，再把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="sort_demo.xsl"?>
<books>
<book>
      <name>《C#从入门到精通(第 2 版)》</name>
      <publisher>清华大学出版社</publisher>
      <company>明日科技</company>
      <author>王小科</author>
      <ISBN>9787302226628</ISBN>
      <price unit="RBM" >69.80</price>
      <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
//省略部分代码
<book>
      <name>《视频学 Java Web(1DVD)》</name>
      <publisher>人民邮电出版社</publisher>
      <author>王国辉</author>
      <company>明日科技</company>
      <ISBN>9787115219893</ISBN>
      <price unit="RBM" >49.80</price>
      <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

秘笈心法

心法领悟 313：XML 文件中的元素排序。

使用<xsl:sort>标签可以根据 XML 的某一个元素对结果内容排序，在日常使用过程中，仅使用一种规则排序往往是不够的，所以可以同时使用多个<xsl:sort>标签，在一个结果内容中对不同的 XML 元素排序。在一个<xsl:for-each>内部，可以有两个<xsl:sort>标签，写在前面的<xsl:sort>的排序方式被优先处理。

实例 314 使用 XSLT 中的 choose

光盘位置：光盘\MR\314　　高级　　趣味指数：★★★★

实例说明

如果在使用 XSLT 时需要有多个判断，可以使用<xsl:choose>标签。本实例根据图书价格的不同区间显示不同的背景颜色，运行效果如图 10.18 所示。

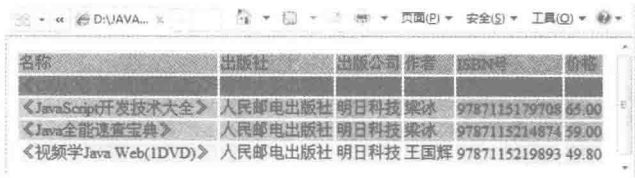


图 10.18　使用 XSLT 的 choose

关键技术

使用 XSLT 的 choose 标签可以进行条件选择，同时 choose 中还包含 when 和 otherwise 标签，代码如下：

```
<xsl:choose>
  <xsl:when test="(price &lt; 50)">
  ……
  </xsl:when>
<xsl:otherwise>
  ……
</xsl:otherwise>
</xsl:choose>
```

设计过程

（1）创建一个 XSLT 文档，在其中绘制一个 4 行 6 列的表格，然后用<xsl:choose>把第 2 个、第 3 个和第 4 个<tr>包含起来，再使用<xsl:choose>和<xsl:when>标签根据输出内容设置单元格的背景色，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
   <table>
     <tr bgcolor="#8E8E8E">
       <td>名称</td>
       <td>出版社</td>
       <td>出版公司</td>
       <td>作者</td>
       <td>ISBN 号</td>
       <td>价格</td>
     </tr>
   <xsl:for-each select="books/book">
     <xsl:choose>
       <xsl:when test="(price &lt; 50)">
     <tr bgcolor="#CDCD9A">
       <td><xsl:value-of select="name"/></td>
       <td><xsl:value-of select="publisher"/></td>
       <td><xsl:value-of select="company"/></td>
       <td><xsl:value-of select="author"/></td>
       <td><xsl:value-of select="ISBN"/></td>
       <td><xsl:value-of select="price"/></td>
     </tr>
       </xsl:when>
       <xsl:when test="(price &lt;= 65)" >
```

```
        <tr bgcolor="#A5A552">
          <td><xsl:value-of select="name"/></td>
          <td><xsl:value-of select="publisher"/></td>
          <td><xsl:value-of select="company"/></td>
          <td><xsl:value-of select="author"/></td>
          <td><xsl:value-of select="ISBN"/></td>
          <td><xsl:value-of select="price"/></td>
        </tr>
          </xsl:when>
         <xsl:otherwise>
        <tr bgcolor="#616130">
          <td><xsl:value-of select="name"/></td>
          <td><xsl:value-of select="publisher"/></td>
          <td><xsl:value-of select="company"/></td>
          <td><xsl:value-of select="author"/></td>
          <td><xsl:value-of select="ISBN"/></td>
          <td><xsl:value-of select="price"/></td>
        </tr>
         </xsl:otherwise>
         </xsl:choose>
      </xsl:for-each>
    </table>
  </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，再把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="choose_demo.xsl"?>
<books>
 <book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
//省略部分代码
<book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

秘笈心法

心法领悟 314：XML 文件中特殊符号的使用。

使用<xsl:when>对条件进行判断时，test 中使用小于和小于等于的判断，实际上应该使用“<”和“<=”，但是 XSLT 本身也是一种 XML，如果在其内部使用这些符号会发生错误，所以需要通过实体引用来代替它们。

实例 315　使用 XSLT 中的 copy-of

高级

光盘位置：光盘\MR\315

趣味指数：★★★★

实例说明

如果一个表格的行列数较多，则不易分辨行列的内容。这时就希望在每行的上面都有一个表头和表里的内

容相对应。本实例使用 copy-of 复制多个表头，运行效果如图 10.19 所示。

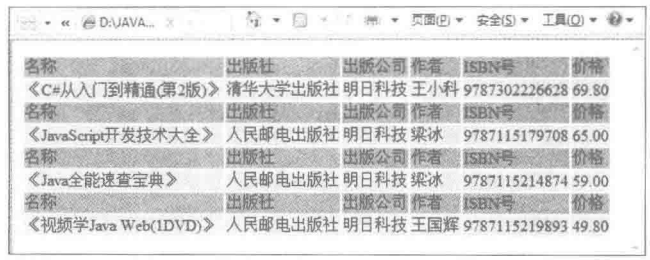


图 10.19　使用 copy-of 复制多个表头

关键技术

使用 XSLT 的 variable 可以为一部分内容定义变量，同时使用 copy-of 可以对变量的内容进行复制，代码如下：

```
<xsl:variable name="title">
      <tr bgcolor="#8E8E8E">
        <td>名称</td>
        <td>出版社</td>
        <td>出版公司</td>
        <td>作者</td>
        <td>ISBN 号</td>
        <td>价格</td>
      </tr>
</xsl:variable>
……
<xsl:copy-of select="$title" />
```

设计过程

（1）创建一个 XSLT 文档，在 XSLT 文档中使用<xsl:variable>声明一个变量，<xsl:variable>中写变量的具体内容。然后使用<xsl:copy-of>把变量声明的内容复制到需要的位置，代码如下：

```
 <?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="title">
      <tr bgcolor="#8E8E8E">
        <td>名称</td>
        <td>出版社</td>
        <td>出版公司</td>
        <td>作者</td>
        <td>ISBN 号</td>
        <td>价格</td>
      </tr>
</xsl:variable>
<xsl:template match="/">
<html>
 <body>
   <table>
    <xsl:for-each select="books/book">
<xsl:copy-of select="$title" />
      <tr bgcolor="#CDCD9A">
        <td><xsl:value-of select="name"/></td>
        <td><xsl:value-of select="publisher"/></td>
        <td><xsl:value-of select="company"/></td>
        <td><xsl:value-of select="author"/></td>
        <td><xsl:value-of select="ISBN"/></td>
        <td><xsl:value-of select="price"/></td>
      </tr>
    </xsl:for-each>
```

```
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，再把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="copy-of_demo.xsl"?>
<books>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
//省略部分代码
<book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

秘笈心法

心法领悟 315：select 元素的用法。

本实例中，<xsl:copy-of>的 select 属性填写的内容是事先已经声明过的变量，引用变量时在变量名前面添加“$”符号。select 里的内容也可以填写 XML 元素，如果填写 XML 元素，最终显示的则是元素的内容。

实例 316　使用 XSLT 中的 apply-templates

光盘位置：光盘\MR\316　　高级　　趣味指数：★★★★

实例说明

使用 apply-templates 可以把定义好的模板应用到某一个元素上，方便用户使用 XSLT 转换 XML，通过使用 apply-templates 可以制作出更加丰富的展示效果，如图 10.20 所示。

图 10.20　使用 apply-templates

关键技术

使用<xsl:template>可以定义一个模板，在需要时可以用<xsl:apply-templates>引用<xsl:template>。<xsl:apply-templates>一般应用在某一个元素上，示例代码如下：

```
<xsl:apply-templates select="name"/>
```

参数说明

select：表示填写需要处理的元素或节点的名称。

设计过程

（1）创建一个 XSLT 文档，使用<xsl:template>创建 3 个模板，在参数 match 中分别填写 name、price、ISBN，在<xsl:template>内部定义各模板的样式，并通过字体颜色加以区分。再使用<xsl:templae>创建一个 book 的模板，然后使用<xsl:apply-templates>引入 name、price、ISBN 这 3 个模板并放在 book 的模板中。设置好后，把<xsl:apply-templaes />应用到<xsl:template match="/">中，让 XML 的每个元素节点应用刚才定义的模板，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
   <xsl:apply-templates/>
 </body>
 </html>
</xsl:template>

<xsl:template   match="book">
<xsl:apply-templates select="name"/>
<xsl:apply-templates select="ISBN"/>
<xsl:apply-templates select="price"/>
<br/>
</xsl:template>

<xsl:template match="name">
<span style="color:#00008b">名称:<xsl:value-of select="."/></span>
<br/>
</xsl:template>
<xsl:template match="price">
<span style="color:#8b008b">价格: <xsl:value-of select="."/></span>
<br/>
</xsl:template>
<xsl:template match="ISBN">
<span style="color:#2f4f4f">ISBN: <xsl:value-of select="."/></span>
<br/>
</xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，再把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="apply-templates_demo.xsl"?>
<books>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
//省略部分代码
<book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
```

```
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
    </book>
    </books>
```

秘笈心法

心法领悟 316：<xsl:template>与<xsl:apply-templates>的关系。

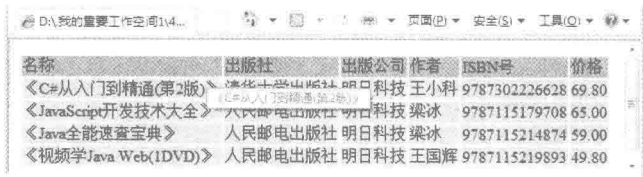
<xsl:apply-templates>的使用有一个前提条件，即需要先使用<xsl:template>定义好一个模板，而<xsl:apply-templates>又可以在<xsl:template>的内部使用，可见二者之间可以相互配合使用。

实例 317　使用 XSLT 中的 attribute　　高级

光盘位置：光盘\MR\317　　趣味指数：★★★★

实例说明

<xls:attribute>可以为某一个元素添加属性。现在有一个表示图书信息的 XML，使用 XSLT 可以将其转换成一个 XHTML 表格。但是这时用户希望把鼠标指针放在图书名称上时，页面上可以同时弹出一个标签，用于显示当前图书的名称，这就需要在 XHTML 中填写图书名称的<td>中动态地添加一个 title 的属性，这可以使用<xls:attribute>来实现，运行效果如图 10.21 所示。



名称	出版社	出版公司	作者	ISBN号	价格
《C#从入门到精通(第2版)》	清华大学出版社	明日科技	王小科	9787302226628	69.80
《JavaScript开发技术大全》	人民邮电出版社	明日科技	梁冰	9787115179708	65.00
《Java全能速查宝典》	人民邮电出版社	明日科技	梁冰	9787115214874	59.00
《视频学Java Web(1DVD)》	人民邮电出版社	明日科技	王国辉	9787115219893	49.80

图 10.21　使用 attribute 为<td>添加 title 属性

关键技术

使用<xsl:attribute>可以在某一元素内部为元素添加属性，示例代码如下：

```
<xsl:attribute name="title">……</xsl:attribute>
```

参数说明

name：表示添加的属性名称。

设计过程

（1）创建一个 XSLT 文档，在图书名称的<td>中额外添加一个<xsl:attribute>，在属性 name 上填写 title，使用浏览器把 XML 文件打开，把鼠标指针移到图书名称上时，可以看见 title 的值，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
   <table>
     <tr bgcolor="#8E8E8E">
       <td>名称</td>
       <td>出版社</td>
       <td>出版公司</td>
       <td>作者</td>
       <td>ISBN 号</td>
```

```
        <td>价格</td>
      </tr>
    <xsl:for-each select="books/book">
      <tr bgcolor="#C2C287">
            <td><xsl:attribute name="title"><xsl:value-of select="name"/></xsl:attribute><xsl:value-of select="name"/></td>
        <td><xsl:value-of select="publisher"/></td>
        <td><xsl:value-of select="company"/></td>
        <td><xsl:value-of select="author"/></td>
        <td><xsl:value-of select="ISBN"/></td>
        <td><xsl:value-of select="price"/></td>
      </tr>
    </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，再把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="attribute_demo.xsl"?>
<books>
<book>
      <name>《C#从入门到精通(第 2 版)》</name>
      <publisher>清华大学出版社</publisher>
      <company>明日科技</company>
      <author>王小科</author>
      <ISBN>9787302226628</ISBN>
      <price unit="RBM" >69.80</price>
      <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
//省略部分代码
<book>
      <name>《视频学 Java Web(1DVD)》</name>
      <publisher>人民邮电出版社</publisher>
      <author>王国辉</author>
      <company>明日科技</company>
      <ISBN>9787115219893</ISBN>
      <price unit="RBM" >49.80</price>
      <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

秘笈心法

心法领悟 317：<xsl:attribute>的标签内容特点。

<xsl:attribute>标签的中间可以填写文字，在转换时，浏览器会直接把<xsl:attribute>标签中间的内容赋在属性中，也可以填写类似<xsl:value-of>的 XSLT 语句，如果是 XSLT 的语法浏览器，在转换时会先把 XSLT 语句转换成普通文字，然后再把文字赋在属性中。

实例 318　使用 XSLT 中的 elements

高级

光盘位置：光盘\MR\318

趣味指数：★★★★

实例说明

<xls:elements>在转换时可以动态地创建一个元素。如现在有一个完整的表格，表格的内容都是从 XML 中获取的，想为该表格添加一个表头，于是使用<xls:elements>创建一个名称为<h2>的元素，该元素在 XHTML 中即可用作表格的表头，<xls:elements>标签的内容就是表头的内容，运行效果如图 10.22 所示。

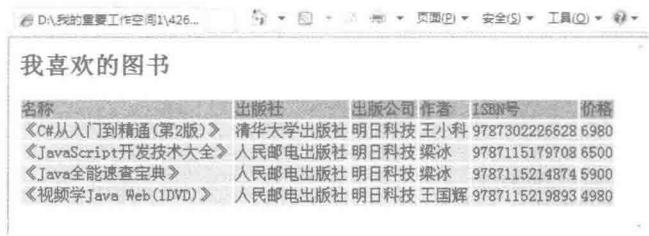


图 10.22　使用 elements 创建标题

关键技术

使用<xsl:elements>为输出文档创建一个元素，示例代码如下：

```
<xsl:element name="h2" >我喜欢的图书 </xsl:element>
```

参数说明

name：表示创建的元素名称。

设计过程

（1）创建一个 XSLT 文档，在表格的上面添加一个<xsl:elements>标签，在输出文档上创建一个元素<h2>，<xsl:elements>的内部添写“我喜欢的图书”作为<h2>的内容，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <body>
      <xsl:element name="h2" >我喜欢的图书</xsl:element>
      <table>
        <tr bgcolor="#8E8E8E">
          <td>名称</td>
          <td>出版社</td>
          <td>出版公司</td>
          <td>作者</td>
          <td>ISBN 号</td>
          <td>价格</td>
        </tr>
        <xsl:for-each select="books/book">
          <xsl:sort select="price" order="descending" />
          <tr bgcolor="#C2C287">
            <td><xsl:value-of select="name"/></td>
            <td><xsl:value-of select="publisher"/></td>
            <td><xsl:value-of select="company"/></td>
            <td><xsl:value-of select="author"/></td>
            <td><xsl:value-of select="ISBN"/></td>
            <td><xsl:value-of select="price"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，再把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="elements_demo.xsl"?>
<books>
<book>
    <name>《C#从入门到精通(第 2 版)》</name>
    <publisher>清华大学出版社</publisher>
    <company>明日科技</company>
```

```
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
    </book>
    //省略部分代码
    <book>
        <name>《视频学 Java Web(1DVD)》 </name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
    </book>
    </books>
```

秘笈心法

心法领悟 318：<xsl:elements>与 html 语言配合使用。

使用<xsl:elements>把文字包含起来设置 name 是 h2，可以把<xsl:elements>内部的文字当作抬头使用，也可以对抬头做其他处理，例如在<xsl:elements>中嵌套一层 或者标签，让抬头看起来更美观。

10.4 XML 与 XSLT 的内建函数

实例 319 使用 XSLT 中的 generate-id()　　高级

光盘位置：光盘\MR\319　　趣味指数：★★★★

实例说明

当一个网页较长时，通过 HTML 的锚点可以准确地定位到页面的各个位置，XSLT 利用 generate-id()函数能很轻松地在 XHTML 中实现该功能。本实例展示了通过 4 本书的名称定位到 4 本书的详细介绍上，运行效果如图 10.23 所示。

图 10.23　使用 generate-id()函数定位锚点

关键技术

使用 generate-id()函数可以为元素生成一个唯一的字符串 id，用来标识此元素在文档中的唯一性。generate-id()函数不带参数，表示为当前节点生成上下文唯一的字符串 id；如果 generate-id()含有参数，表示为此参数生成唯一的字符串 id。代码如下：

```
generate-id(name)
```

参数说明

name：表示 XML 元素名称。

设计过程

（1）创建一个 XSLT 文档，使用 generate-id()函数为 name 生成一个锚点，并且生成与之相对应的锚点，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" version="4.0" encoding="iso-8859-1" indent="yes" />
 <xsl:template match="/">
        <html>
                <body>
                        <xsl:element name="h2">
                                我喜欢的图书
                        </xsl:element>
                        <xsl:for-each select="books/book">
                                <p><xsl:value-of select="name" /> <a href="#{generate-id(name)}">查看详细</a></p>
                        </xsl:for-each>
                                <hr/>
                        <xsl:for-each select="books/book">
                                <p><a name="#{generate-id(name)}"><xsl:value-of select="name" /></a></p>
                                <p><xsl:value-of select="publisher" /></p>
                                <p><xsl:value-of select="company" /></p>
                                <p><xsl:value-of select="author" /></p>
                                <p><xsl:value-of select="ISBN" /></p>
                                <p><xsl:value-of select="price" /></p>
                                <p><xsl:value-of select="url" /></p>
                                <hr/>
                        </xsl:for-each>
                </body>
        </html>
 </xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，再把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="generate-id_demo.xsl"?>
<books>
 <book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
 </book>
 //省略部分代码
 <book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
```

```
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
    </book>
    </books>
```

秘笈心法

心法领悟 319：generate-id()的用法。

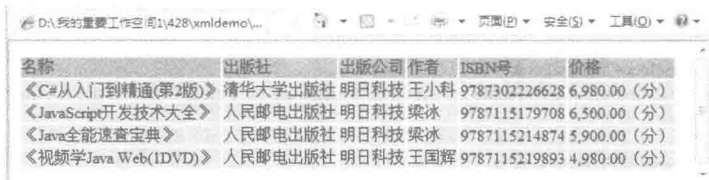
在实例中，使用 generate-id()函数为元素生成了一个唯一的字符串，XSLT 根据元素的名称生成，函数名称不变，无论在何处调用 generate-id()函数，只要是在同一个文档中，generate-id()生成的值都是一致的。使用 generate-id()函数时，每次打开文档都会随机生成不同的字符串。

实例 320 使用 XSLT 中的 format-number()

光盘位置：光盘\MR\320　　高级　　趣味指数：★★★★

实例说明

format-number()函数用来格式化数字。本实例中，书的价格单位是元，为了能明显地看到 format-number()的实现效果，将单位换成分，这样即使图书的价格要扩大 100 倍，看起来依然方便，运行效果如图 10.24 所示。



名称	出版社	出版公司	作者	ISBN号	价格
《C#从入门到精通(第2版)》	清华大学出版社	明日科技	王小科	9787302226628	6,980.00（分）
《JavaScript开发技术大全》	人民邮电出版社	明日科技	梁冰	9787115179708	6,500.00（分）
《Java全能速查宝典》	人民邮电出版社	明日科技	梁冰	9787115214874	5,900.00（分）
《视频学Java Web(1DVD)》	人民邮电出版社	明日科技	王国辉	9787115219893	4,980.00（分）

图 10.24　使用 format-number()函数格式化数字

关键技术

使用 formate-number()函数可格式化 XML 中数值的显示方法，代码如下：

```
format-number(price,"###,###.00")
```

参数说明

❶ price：表示 XML 数值的元素名称。

❷ ###,###.00：表示数值的显示方式。

设计过程

（1）创建一个 XSLT 文档，并在其中创建一个 2 行 6 列的表格，第 1 个<tr>作为表头，第 2 个<tr>使用<xsl:for-each>遍历 XML 文档内容。在输出 XML 内容时，把 price 元素使用 format-number()格式化输出，然后再定义 price 的输出样式，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
   <table>
     <tr bgcolor="#8E8E8E">
       <td>名称</td>
       <td>出版社</td>
       <td>出版公司</td>
       <td>作者</td>
       <td>ISBN 号</td>
       <td>价格</td>
```

```
        </tr>
      <xsl:for-each select="books/book">
        <tr bgcolor="#C2C287">
          <td><xsl:value-of select="name"/></td>
          <td><xsl:value-of select="publisher"/></td>
          <td><xsl:value-of select="company"/></td>
          <td><xsl:value-of select="author"/></td>
          <td><xsl:value-of select="ISBN"/></td>
          <td><xsl:value-of select='format-number(price,"###,###.00")'/>（分）</td>
        </tr>
      </xsl:for-each>
    </table>
  </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，再把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="formate-number_demo.xsl"?>
<books>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
//省略部分代码
<book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

秘笈心法

心法领悟 320：format-number()语句中特殊字符的含义。

格式化 format-number()函数时，通过一些特殊字符来表示。如“#”表示具体数字的位置，“.”表示小数点的位置，“0”表示小数点前后的零，“,”表示千位的分隔符，“%”表示使用百分比来显示。

实例 321　使用 XSLT 中的 document()

高级

光盘位置：光盘\MR\321

趣味指数：★★★★

实例说明

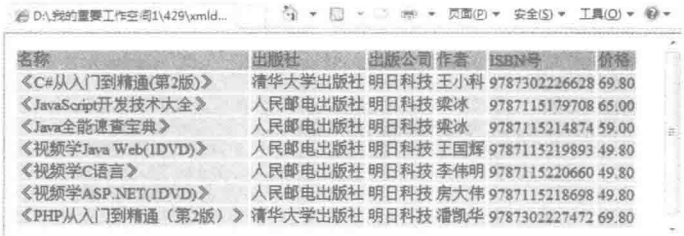
通常一个 XML 的内容不能满足需要，需要引用另外一个 XML，并把引用的 XML 和当前的 XML 合并在一起使用，运行效果如图 10.25 所示。

关键技术

使用 document()函数可以引用外部的 XML 文档，可以引入整个 XML 文档，也可以引用 XML 文档中的某

些元素，代码如下：

document('document_ext_demo.xml')



D:\我的重要工作空间1\429\xmld... 页面(P) 安全(S) 工具(O)

名称	出版社	出版公司	作者	ISBN号	价格
《C#从入门到精通(第2版)》	清华大学出版社	明日科技	王小科	9787302226628	69.80
《JavaScript开发技术大全》	人民邮电出版社	明日科技	梁冰	9787115179708	65.00
《Java全能速查宝典》	人民邮电出版社	明日科技	梁冰	9787115214874	59.00
《视频学Java Web(1DVD)》	人民邮电出版社	明日科技	王国辉	9787115219893	49.80
《视频学C语言》	人民邮电出版社	明日科技	李伟明	9787115220660	49.80
《视频学ASP.NET(1DVD)》	人民邮电出版社	明日科技	房大伟	9787115218698	49.80
《PHP从入门到精通（第2版）》	清华大学出版社	明日科技	潘凯华	9787302227472	69.80

图 10.25　使用 document()函数引用外部 XML

参数说明

document_ext_demo.xml：表示希望引入的 XML 文档内容。

本实例中，通过 document()函数引用 document_ext_demo.xml 的 book 元素，然后使用<xsl:for-each>对 document_ext_demo.xml 进行遍历。

设计过程

（1）创建一个 XSLT 文档，用<xsl:for-each>遍历内部的 XML 内容。再在<xsl:for-each>下面建立一个<xsl:for-each>标签，用于遍历外部的 XML 文档，在该标签中使用 select 属性调用 document()方法。document()方法引用了外部的 XML 文档，其后跟外部 XML 需要遍历的元素，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
   <table>
     <tr bgcolor="#8E8E8E">
       <td>名称</td>
       <td>出版社</td>
       <td>出版公司</td>
       <td>作者</td>
       <td>ISBN 号</td>
       <td>价格</td>
     </tr>
   <xsl:for-each select="books/book">
     <tr bgcolor="#C2C287">
       <td><xsl:value-of select="name"/></td>
       <td><xsl:value-of select="publisher"/></td>
       <td><xsl:value-of select="company"/></td>
       <td><xsl:value-of select="author"/></td>
       <td><xsl:value-of select="ISBN"/></td>
       <td><xsl:value-of select="price"/></td>
     </tr>
    </xsl:for-each>
   <xsl:for-each select="document('document_ext_demo.xml')/books/book">
     <tr bgcolor="#C2C287">
       <td><xsl:value-of select="name"/></td>
       <td><xsl:value-of select="publisher"/></td>
       <td><xsl:value-of select="company"/></td>
       <td><xsl:value-of select="author"/></td>
       <td><xsl:value-of select="ISBN"/></td>
       <td><xsl:value-of select="price"/></td>
     </tr>
    </xsl:for-each>
   </table>
 </body>
</html>
```

```
</xsl:template>
</xsl:stylesheet>
```

（2）建立一个 XML 文档，再把 XSLT 引入到 XML 中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="document_demo.xsl"?>
<books>
<book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
//省略部分代码
<book>
        <name>《视频学 Java Web(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>王国辉</author>
        <company>明日科技</company>
        <ISBN>9787115219893</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=208&sid=11]]></url>
</book>
</books>
```

（3）建立另外一个需要引用的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
<book>
        <name>《视频学 C 语言》</name>
        <publisher>人民邮电出版社</publisher>
        <company>明日科技</company>
        <author>李伟明</author>
        <ISBN>9787115220660</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo-207.html]]></url>
</book>
<book>
        <name>《视频学 ASP.NET(1DVD)》</name>
        <publisher>人民邮电出版社</publisher>
        <author>房大伟</author>
        <company>明日科技</company>
        <ISBN>9787115218698</ISBN>
        <price unit="RBM" >49.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo-210.html]]></url>
</book>
<book>
        <name>《PHP 从入门到精通（第 2 版）》</name>
        <publisher>清华大学出版社</publisher>
        <author>潘凯华</author>
        <company>明日科技</company>
        <ISBN>9787302227472</ISBN>
        <price unit="RBM" >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo-223.html]]></url>
</book>

</books>
```

秘笈心法

心法领悟 321：引入外部文档的方法。

使用 document()函数可以引用外部文档以及文档中的元素，本实例引用的是同一目录下的 XML 文档，直接在参数中输入 XML 文件的名称即可，如果引用上级目录或下级目录，需要使用目录的访问形式，如 document ('.../document_deom.xml')或者 document('docu/document_demo.xml')。

10.5 DTD 的引用与验证

实例 322 在 XML 内部定义 DTD

光盘位置：光盘\MR\322

中级

趣味指数：★★★★

实例说明

XML 的使用规则是由使用者自己来定义的，其他用户如果想使用这个 XML，就要遵守其使用规则，这个规则可以通过 DTD 来定义。本实例使用 DTD 定义了一个 XML 的文档结构，并且在 XML 中使用了该 DTD，运行效果如图 10.26 所示。

图 10.26 在 XML 中定义 DTD

关键技术

在定义 XML 文档结构时，使用 DOCTYPE 可以声明 DTD，代码如下：

```
<!DOCTYPE  book [
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
]>
```

参数说明

❶ DOCTYPE：表示声明 DTD 的关键字。

❷ ELEMENT：用于声明 XML 元素。

设计过程

（1）建立一个空的 XML 文档，根据 XML 文档的格式声明 DTD 内容，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE  book [
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
]>
```

（2）根据 DTD 的定义情况创建一个 XML 文档，内容附加在 DTD 内容下面，代码如下：

```
<book>
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
```

```
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price>69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 322：验证 DTD 文档的错误。

一般在编写 XML 时，最简单的验证方法是通过浏览器打开 XML，验证编写是否有书写上的错误，在 XML 中定义了 DTD 后，同样可以使用该方法验证 DTD 是否有拼写等错误。

实例 323　在 XML 外部引用 DTD

光盘位置：光盘\MR\323　　中级　　趣味指数：★★★★

实例说明

使用 DTD 时，可以在 XML 文档内部定义，但为了引用方便和书写规范，一般都把 DTD 单独定义成一个 DTD 文档，XML 可以通过引用的方式使用 DTD 的定义。本实例实现引用外部 DTD 的 XML 文档，运行效果如图 10.27 所示。

图 10.27　引用外部 DTD 文档

关键技术

使用 DOCTYPE 表示 DTD 时，SYSTEM 表示引用一个外部的 DTD，在 SYSTEM 后面添写 DTD 的 URL 地址，即可在 XML 引用一个 DTD 文档，示例代码如下：

```
<!DOCTYPE book SYSTEM "simple_demo.dtd">
```

参数说明

❶ book：表示是 XML 的根节点。

❷ simple_demo.dtd：表示 XML 的文件名称。

关键技术

（1）根据需要创建一个 DTD 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

（2）根据 DTD 的定义创建一个 XML 文档，在 XML 文档中引入 DTD 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "simple_demo.dtd">
<book>
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price>69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 323：引入 DTD 文档的具体过程。

在 XML 中引用 DTD 时，首先使用 DOCTYPE 声明，然后定义 XML 文档中的根元素，接着使用 SYSTEM 来标示 XML 需要引用的外部 DTD 资源，SYSTEM 的位置既可以定义为 SYSTEM，又可以定义为 PUBLIC，当定义为 PUBLIC 时，表示引用的 DTD 是由权威机构制订的，供特定行业或公众使用。

实例 324　验证 XML 是否符合 DTD 的定义　高级

光盘位置：光盘\MR\324　趣味指数：★★★★

实例说明

使用 DTD 可以定义 XML 的文档结构，但是该 XML 文档结构是否遵守了 DTD 的定义规则，则可以通过程序来验证。本实例使用 IE 浏览器验证 XML 是否符合 DTD 的定义，运行效果如图 10.28 所示。

图 10.28　验证 XML 是否符合 DTD 定义

关键技术

（1）使用 JavaScript 中的 ActiveXObject()函数可以获取一个 XMLDOM 对象，语法如下：

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
```

参数说明

Microsoft.XMLDOM：表示在 IE 浏览器获取 XMLDOM 对象。

（2）使用 JavaScript 中 ActiveXObject 的 load()方法可以加载一个 XML 文档，语法如下：

```
load(filename);
```

参数说明

filename：表示要加载 XML 文档的路径和名称。

设计过程

（1）创建一个 DTD 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

（2）根据 DTD 文档创建一个 XML 文档，同时引入 DTD 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "simple_demo.dtd">
<book>
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price>69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

（3）设计一个 HTML 文档，添加一个<text>标签接收需要验证的 XML；添加一个 button 按钮，在按钮上添加 onclick()方法，每次单击按钮时，按钮都会触发验证 XML 文档的 JavaScript 方法；添加一个 id 为 showError 的 标签，用于显示输出信息到页面上。完成验证 DTD 的方法，在方法中把错误的信息输出到 showError 的位置上。

声明一个 XMLDOM 对象，使用 load()方法加载需要验证的 XML 文档，解析器在加载 XML 时也会加载 DTD 文档，同时完成验证工作。如果解析器验证没有错误，那么 parseError.errorCode 的值就为 0；如果有错误，parseError.errorCode 的值则是与之相对应的错误值。同时使用 parseError.reason 和 xmlDoc.parseError.line 还可以准确地输出错误的原因和行数，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<style type="text/css">
<!--
#showError {
color: #F00;
font-size: 16px;
}
-->
</style></head>
<script type="text/javascript">
function validateXML(filename){
var txt="";
if (window.ActiveXObject){
        document.getElementById("showError").innerText ="";
        var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
        xmlDoc.async = false;
        xmlDoc.validateOnParse= true;
        xmlDoc.load(filename);

        if(xmlDoc.parseError.errorCode!=0){
        txt="Error Code: " + xmlDoc.parseError.errorCode + "\n";
        txt=txt+"Error Reason: " + xmlDoc.parseError.reason ;
        txt=txt+"Error Line: " + xmlDoc.parseError.line;
      }else{
        txt="没有错误";
      }
        document.getElementById("showError").innerText = txt;
```

```
}else{
            alert("此浏览器不支持验证!");
}
}
</script>
<body>
<center>
<h2>通过 DTD 验证 XML 文件</h2>
<p>
<input id="xmlfile" type="text" />
<input type="button" name="submit" value="验 证" onClick="validateXML(document.getElementById('xmlfile').value);" />
</p>
</center>
<div id="showError"></div>
</body>
</html>
```

秘笈心法

心法领悟 324：输出 DTD 文档的错误信息。

在输出 DTD 的错误信息时，先获取<showError>标签，并在页上把错误信息输出到<showError>标签上，这里可以使用 innerText()和 innerHtml()两种方法。两者的区别在于 innerText()可以把信息完整地输出到页面上，而 innerHtml()会把信息按着 HTML 语言的方式输出到页面上。如果页面输出一个
，innerText()输出以后就是
的字样，使用 innerHtml()输出时，页面上只会输出一个回车符。

10.6 使用 DTD 定义 XML 元素

实例 325 在 DTD 中声明元素　　中级

光盘位置：光盘\MR\325　　趣味指数：★★★

实例说明

XML 的基本单位是元素，所以在 DTD 中声明元素也是最基本的，ELEMENT 用于 DTD 元素的声明，在声明的同时还可以定义元素的各种使用情况。XML 文档一般根据 DTD 的声明进行创建，如图 10.29 所示。

图 10.29　根据 DTD 声明创建 XML 元素

关键技术

使用 DTD 的 ELEMENT 关键字可以声明 XML 元素，示例代码如下：

```
<!ELEMENT name (#PCDATA)>
```

参数说明

name：表示 XML 元素的名称。

设计过程

（1）建立一个 DTD 文档，首先在文档中使用 ELEMENT 定义根元素和其他内容，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

（2）根据 DTD 的定义，建立相应的 XML 文件，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "simple_demo.dtd">
<book>
 <name>《C#从入门到精通(第 2 版)》</name>
 <publisher>清华大学出版社</publisher>
 <company>明日科技</company>
 <author>王小科</author>
 <ISBN>9787302226628</ISBN>
 <price>69.80</price>
 <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 325：DTD 文档中对元素的约束。

在定义 DTD 时，如果不想给某一元素定义的约束太强，让其可以包含任意内容，可以把元素内容定义成 ANY，如上面定义 book 元素只能包含 name、publisher、company、author、ISBN、privce 和 url 7 个子元素，如果想使其包含任意内容，可以这样定义：

```
<!ELEMENT book ANY>
```

如果希望元素内容没有任何值，则可以定义为 EMPTY，代码如下：

```
<!ELEMENT book EMPTY>
```

实例 326　在 DTD 中声明重复元素

光盘位置：光盘\MR\326　　中级　　趣味指数：★★★

实例说明

定义 DTD 时，在父元素的内容中声明子元素的名称，如果不加任何条件，该子元素只能在父元素中出现一次，若在 XML 中定义了多个子元素，使用验证工具进行检验时会出现如图 10.30 所示的错误。

图 10.30　在 XML 文档中添加重复元素的错误

关键技术

如果希望父元素内部能出现多个子元素，声明时要在子元素后面添加“+”，示例代码如下：

```
<!ELEMENT book (name,publisher,company,author+,ISBN,price,url)>
```

参数说明

author：在 author 后面加“+”，表示可以有多个 author。

设计过程

（1）建立一个 DTD 文档，首先在文档中使用 ELEMENT 定义根元素及其子元素，在可以重复的子元素后添加“+”，然后分别定义子元素和子元素的内容，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book (name,publisher,company,author+,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

（2）根据定义的 DTD 文档，建立相应的 XML 文件，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "author_demo.dtd">
<book>
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<author>徐薇</author>
<ISBN>9787302226628</ISBN>
<price>69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 326：定义子元素在父元素中出现的次数。

使用“+”表示 XML 子元素在其父元素中至少要出现一次，也可以出现多次，说明子元素必须要在父元素中出现，若在使用 XML 时，允许子元素可以不在父元素中出现，也可以出现多次，可使用“*”，如下所示：

```
<!ELEMENT book (name,publisher,company,author*,ISBN,price,url)>
```

在元素后面使用“?”，表示该元素可以不出现，如果出现也只能出现一次，代码如下：

```
<!ELEMENT book (name,publisher,company,author?,ISBN,price,url)>
```

实例 327 在 DTD 中声明选择性元素

光盘位置：光盘\MR\327

中级

趣味指数：★★★

实例说明

为了满足 XML 数据的多样性，DTD 提供了各种声明。有这样一种情况，XML 的父元素中有多个子元素，其中有两个子元素必须要在父元素中出现一个，但不能同时出现。如在 book 元素中，必须要有 tel 或 phone 元素，但是二者不能同时出现，如果没有按 DTD 的声明使用 XML，验证时就会出现错误，运行效果如图 10.31

所示。

图 10.31　选择性元素同时存在的错误情况

实例说明

使用“|”定义元素时，“|”前后的两个元素在父元素中有且只有一个出现，示例代码如下：

```
<!ELEMENT book (name,publisher,company,author,ISBN,price,url,(tel|phone))>
```

设计过程

（1）建立一个 DTD 文档，首先在文档中使用 ELEMENT 定义根元素及其子元素 name、publisher、company、author、ISBN、price、url 和（telephone），然后分别定义子元素和子元素内容，同时定义 tel 和 phone 两者不能同时出现在 XML 父元素中，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

（2）根据定义的 DTD 文档，建立相应的 XML 文件，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "tel_demo.dtd">
<book>
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price>69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
<tel>13812345678</tel>
</book>
```

秘笈心法

心法领悟 327：DTD 中元素出现的位置。

在 DTD 中把“|”和“*”混合在一起使用，可使定义的元素更灵活，如下所示：

```
<!ELEMENT book (name|publisher|company|author|ISBN|price|url)*>
```

上述代码表示 book 中的子元素可以在任意位置出现，并且没有次数的限制，这种 DTD 声明让用户在使用 XML 时更自由。

实例 328 在 DTD 中使用 ENTITY

光盘位置：光盘\MR\328

高级

趣味指数：★★★

实例说明

使用 ENTITY 关键字可以声明一个实体，在 XML 文档中可以重用公共内容、提高代码效率，对于一个可能出现多次的内容而言十分有用。在 XML 中通过引用实体名称实现对 ENTITY 的调用，如图 10.32 所示。

图 10.32 在 DTD 中使用 ENTITY

关键技术

使用 DTD 的 ENTITY 关键字可以声明一个实体，示例代码如下：

```
<!ENTITY info "软件工程师入门丛书" >
```

参数说明

❶ info：表示声明实体的名称。

❷ 软件工程师入门丛书：表示声明实体的内容。

设计过程

（1）使用 ELEMENT 定义根元素 books 及其子元素 book，同时允许 book 出现多次，所以要在 book 后面添加“+”。使用 ELEMENT 为 book 声明子元素 name、publisher、company、author、ISBN、price、url 和 type，并为每个子元素设定元素类型。然后使用 ENTITY 声明实体名称 info，在实体名称 info 后面定义实体内容“软件工程师入门丛书”，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT books (book+)>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url,type)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ENTITY info "软件工程师入门丛书" >
```

（2）在 XML 文档中使用实体时，要在实体前后添加“&”和“;”符号，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "entity_demo.dtd">
<books>
 <book>
        <name>《Java 从入门到精通（第 2 版）》</name>
        <publisher>清华大学出版社</publisher>
```

```
        <company>明日科技</company>
        <author>李钟蔚</author>
        <ISBN>9787302227465</ISBN>
        <price>59.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=222]]></url>
        <type>&info;</type>
    </book>
    </books>
```

秘笈心法

心法领悟 328：引入外部实体的方法。

本实例声明的是内部实体，在其他应用中还可以使用 ENTITY 声明外部实体。在引用外部实体时必须添加关键字 SYSTEM 或 PUBLIC，如下所示：

```
<!ENTITY info SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
```

10.7　使用 DTD 定义 XML 属性

实例 329　在 DTD 中声明元素属性　　中级

光盘位置：光盘\MR\329　　趣味指数：★★★

实例说明

在 XML 的使用中，有时只使用元素还不能声明完整的 XML 结构，还要使用 DTD 定义 XML 属性。例如，一本图书的价格是 69.80 元，那么采用的币种是什么？人民币、美元还是欧元？这时就可以通过属性的定义完善价格定义的不足，如图 10.33 所示。

图 10.33　在 XML 中声明元素属性

关键技术

使用 DTD 的 ATTLIST 关键字可以为 XML 声明元素的属性，示例代码如下：

```
<!ATTLIST price unit CDATA "RMB">
```

参数说明

❶ price：表示要声明属性所在元素名称。

❷ unit：表示属性名称。

❸ CDATA：表示属性类型。

❹ RMB：表示属性默认值。

设计过程

（1）建立一个 DTD 文档，首先在文档中使用 ELEMENT 定义根元素 book 及其子元素 name、publisher、

company、author、ISBN、price 和 url。然后使用 ATTLIST 为 price 元素定义一个属性 unit，并设定 unit 的类型为 CDATA、默认值是 RMB，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST price unit CDATA "RMB"></book>
```

（2）根据定义的 DTD 文档，建立相应的 XML 文档，在文档中为 price 定义属性 unit="RMB"，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "attlist_demo.dtd">
<book>
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price unit="RMB">69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 329：元素定义时的默认值。

本实例中为元素的属性定义了默认值 RMB，但是在 XML 的实际应用中，属性的内容可以是任何 CDATA 值，unit 属性省略时才会默认为 RMB。

实例 330　在 DTD 中声明带有#IMPLIED 的属性

中级

光盘位置：光盘\MR\330

趣味指数：★★★

实例说明

XML 文档中，有些元素的属性比较特殊，例如同样一个元素，有的地方必须使用属性，而有的地方不能使用属性，这样 DTD 声明就会出现矛盾。DTD 中如果声明了属性，XML 默认会把属性加到元素中；如果 DTD 中不定义属性，XML 使用时对其验证会出现如图 10.34 所示的错误。为了解决这种问题，声明 XML 属性时需要使用#IMPLIED 对其进行修饰，表示该 XML 元素的属性不是必须使用的。

图 10.34　没有定义 unit 的验证

关键技术

在 DTD 文档中定义属性时，在正常的属性定义上面加上参数#IMPLIED，表示该元素的属性不是必须使用的，示例代码如下：

```
<!ATTLIST price unit CDATA #IMPLIED>
```

设计过程

（1）建立一个 DTD 文档，首先在文档中使用 ELEMENT 定义根元素 book 及其子元素 name、publisher、company、author、ISBN、price 和 url。然后使用 ATTLIST 为 price 元素声明一个属性 unit，并设定 unit 的类型为 CDATA，设置属性参数为#IMPLIED，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST price unit CDATA #IMPLIED>
```

（2）创建 XML 文档，这时 price 的属性即可根据需要使用，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "attlist_implied_demo.dtd">
<book>
 <name>《C#从入门到精通(第 2 版)》</name>
 <publisher>清华大学出版社</publisher>
 <company>明日科技</company>
 <author>王小科</author>
 <ISBN>9787302226628</ISBN>
 <price unit="RMB">69.80</price>
 <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 330：定义元素时可以为元素添加默认值。

定义属性时，若没有添加#IMPLIED 参数，可以为 unit 添加一个默认值，但在使用#IMPLIED 参数后，不可以再为属性添加默认值，否则在验证时会出现如图 10.35 所示的错误。

图 10.35　使用#IMPLIED 参数以后为属性添加默认值

实例 331　在 DTD 中声明带有#REQUIRED 的属性　中级

光盘位置：光盘\MR\331　趣味指数：★★★★

实例说明

在定义 DTD 的属性时，有些 XML 文档要求属性值必须明确地写出来，如果没有按照 DTD 的声明使用 XML，解析器在对其进行验证时会出现如图 10.36 所示的错误。

图 10.36　添加#REQUIRED 参数后没有使用 unit 属性

关键技术

在 DTD 文档中声明属性时，在属性定义的末尾加上#REQUIRED 关键字，这时的 XML 文档在使用时必须填写属性值，示例代码如下：

```
<!ATTLIST price unit CDATA #REQEUIRED>
```

设计过程

（1）建立一个 DTD 文档，首先在文档中使用 ELEMENT 声明根元素 book 及其子元素 name、publisher、company、author、ISBN、price 和 url。然后使用 ATTLIST 为 price 元素声明一个属性 unit，并设定 unit 的类型为 CDATA，设置属性参数为#REQUIRED，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST price unit CDATA #REQUIRED>
```

（2）在使用 XML 文档时，price 元素必须填写属性值，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "attlist_implied_demo.dtd">
<book>
 <name>《C#从入门到精通(第 2 版)》</name>
 <publisher>清华大学出版社</publisher>
 <company>明日科技</company>
 <author>王小科</author>
 <ISBN>9787302226628</ISBN>
 <price unit="RMB">69.80</price>
 <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 331：定义元素时可添加默认值。

定义属性时，若没有添加#REQUIRED 参数，可以为 unit 添加一个默认值，但在使用#REQUIRED 参数后，不可以再为属性添加默认值，否则在验证时会出现如图 10.37 所示的错误。

图 10.37　使用#REQUIRED 参数以后为属性添加默认值

实例 332　在 DTD 中声明带有#FIXED 的属性　中级

光盘位置：光盘\MR\332　　趣味指数：★★★★

实例说明

在定义 DTD 属性时，有些 XML 文档要求属性是定义者事先规定好的，不允许使用者随意改动，如果没有按照 DTD 的声明使用，在对 XML 进行验证时会出现如图 10.38 所示的错误。

图 10.38　添加 unit 属性值使用不正确

关键技术

在 DTD 文档中定义属性时，在元素类型的后面添加#FIXED 关键字，并在#FIXED 参数后面声明固定的属性值，示例代码如下：

```
<!ATTLIST price unit CDATA #FIXED "RMB">
```

设计过程

（1）建立一个 DTD 文档，首先在文档中使用 ELEMENT 定义根元素 book 及其子元素 name、publisher、company、author、ISBN、price 和 url。然后使用 ATTLIST 为 price 元素声明属性 unit，设定 unit 类型为 CDATA，设置属性参数为#FIXED，并填写固定的属性值，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name          (#PCDATA)>
<!ELEMENT publisher     (#PCDATA)>
<!ELEMENT company       (#PCDATA)>
<!ELEMENT author        (#PCDATA)>
<!ELEMENT ISBN          (#PCDATA)>
<!ELEMENT price         (#PCDATA)>
<!ELEMENT url       (#PCDATA)>
<!ATTLIST price unit CDATA #FIXED "RMB">
```

（2）在使用 XML 文档时，为 price 填写属性值 RMB，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "attlist_implied_demo.dtd">
<book>
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price unit="RMB">69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 332：元素定义时属性的固定值。

在 DTD 定义属性时，为元素 price 的属性 unit 添加#FIXED 参数，表示如果 XML 中 price 使用属性值，必须使用已经规定的属性值。这只在 XML 使用属性的情况下有效，如果 prcie 没有使用属性，也就用不到固定的属性值，这样也不会出现问题。

实例 333　在 DTD 中声明列举属性值　　中级

光盘位置：光盘\MR\333　　趣味指数：★★★★

实例说明

如果希望属性值为一系列固定的值之一，可以使用列举属性值。例如，一本图书的价格是 69.80，其单位可能是元、角或者分，但是只可能是其中一项。本实例定义图书价格单位为元，运行效果如图 10.39 所示。

图 10.39　price 属性 unit 的值是元

关键技术

使用 DTD 的 ATTLIST 关键字声明属性时，“|”可以为属性列举具体的属性值，在 XML 中使用该属性时，必须使用 DTD 文档中列举的值，“|”的使用方法如下：

```
<!ATTLIST price unit (元|角|分) "元">
```

参数说明

❶ (元|角|分)：表示列举的属性值为元、角、分。

❷ "元"：表示属性的默认值是元。

设计过程

（1）建立一个 DTD 文档，首先在文档中使用 ELEMENT 定义根元素 book 及其子元素 name、publisher、company、author、ISBN、price 和 url。然后使用 ATTLIST 为 price 元素定义一个属性 unit，并列举出 unit 的值为元、角和分，设置属性的默认值为元，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST price unit (元|角|分) "元">
```

（2）在使用 XML 文档时，可以为 price 的属性 unit 定义元、角、分中的一种，也可以省略，系统会使用默认值，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "attlist_many_demo.dtd">
<book>
 <name>《C#从入门到精通(第 2 版)》</name>
 <publisher>清华大学出版社</publisher>
 <company>明日科技</company>
 <author>王小科</author>
 <ISBN>9787302226628</ISBN>
 <price unit="元" >69.80</price>
 <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
 </book>
```

秘笈心法

心法领悟 333：DTD 定义属性值的错误用法。

DTD 中如果枚举出属性值，在 XML 中就只能使用枚举中的内容，例如 price 的属性 unit 只能使用元、角、分，而不能使用其他内容，否则在验证 XML 时会出现如图 10.40 所示的错误。

图 10.40　属性内容使用枚举以外的值

实例 334　类型为 ID 的属性实例

高级

趣味指数：★★★

光盘位置：光盘\MR\334

实例说明

属性类型声明为 ID 时，表示该属性的值在 XML 中必须是唯一的，具有唯一标识的功能，而且必须遵守

XML 名称定义的规则，ID 属性可以作为元素的唯一标识符，每个元素只有一个 ID 类型的属性。例如，一本书通过 UID 来标示其在书架上的唯一性，不可能有两个同样的 UID="AAAA"。若 XML 属性出现重复，解析文档时就会出现如图 10.41 所示的错误。

图 10.41　ID 属性值重复

关键技术

使用 ATTLIST 关键字为元素定义了一个属性时，如果属性的类型是 ID，则属性必须设置为#IMPLIED 或#REQUIRED 类型，示例代码如下：

```
<!ATTLIST book UID ID #REQUIRED>
```

设计过程

（1）使用 ELEMENT 定义根元素 books，在其内部声明子元素 book，同时为了使 books 中可有多个 book 子元素，定义时在 book 后面添加“+”；使用 ELEMENT 为 book 声明其子元素 name、publisher、company、author、ISBN、price 和 url，并为每个子元素设定类型；使用 ATTLIST 为元素 book 定义一个属性 UID，同时设定 UID 的类型是 ID，设置属性参数为#REQUIRED，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT books (book+)>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST book UID ID #REQUIRED>
```

（2）在 XML 文档中可以使用多个 book，每个 book 的 UID 属性必须填写并且是唯一的，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "attlist_id_demo.dtd">
<books>
 <book UID="AAAA">
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
 </book>
 <book UID="AAAB">
        <name>《JavaScript 开发技术大全》</name>
        <publisher>人民邮电出版社</publisher>
        <company>明日科技</company>
        <author>梁冰</author>
```

```
        <ISBN>9787115179708</ISBN>
        <price>65.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=138&sid=5]]></url>
    </book>
    </books>
```

秘笈心法

心法领悟 334：定义属性的唯一性。

XML 中所有的元素都可以定义属性，UID 作为 book 的属性，必须在整个 XML 中唯一，即使是 name、publisher、company 等 book 的子元素的属性值也不可以和 UID 的值一致。

实例 335　类型为 IDREF 的属性实例

光盘位置：光盘\MR\335　　高级　　趣味指数：★★★

实例说明

IDREF 是 DTD 的又一种属性类型，与属性类型为 ID 的声明联合使用，一般用在文档中创建链接和交叉引用的地方。使用 IDREF 可以把两个相关的元素体现在 XML 文档中，属性为 IDREF，和 ID 类型一样，必须遵守 XML 名称定义的规则。IDREF 一般用来引用 ID 类型的值来表示两个元素之间的关系。例如，现有 3 个 book 元素，其 UID 分别是 AAAA、AAAB 和 AAAC，在 UID="AAAC"的 book 中有一个 relevance 元素，其属性 relUID 就是一个 IDREF 类型的属性，这里使用 relUID="AAAA"来表示 UID 为 AAAA 和 AAAC 之间存在的一种关系，如图 10.42 所示。

图 10.42　IDREF 类型的属性

关键技术

使用 ATTLIST 关键字为元素 book 定义一个属性 UID，同时设定 UID 的类型是 ID，ID 类型的属性必须设置为#IMPLIED 或#REQUIED。然后为元素 relevace 定义 relUID 属性，设定 relUID 的类型为 IDREF、参数是

#REQUIRED。代码如下：

```
<!ATTLIST book UID ID #REQUIRED>
<!ATTLIST relevance relUID IDREF #REQUIRED>
```

设计过程

（1）使用 ELEMENT 定义根元素 books，并在其内部定义子元素 book，同时允许 books 中可以有多个 book，所以这样声明 books 的内容：(book+)；使用 ELEMENT 为 book 声明其子元素 name、publisher、company、author、ISBN、price、url 和 relevance，并为每个子元素设定类型；使用 ATTLIST 为元素 book 定义一个属性 UID，同时设定 UID 的类型是 ID，设置属性参数为#REQUIRED。relevace 表示和当前图书是一个套系的图书，relUID 表示同一套系图书的 UID 值，这样就把两个有关联的图书绑定在一起了。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT books (book+)>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url,relevance*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT relevance (#PCDATA)>
<!ATTLIST book UID ID #REQUIRED>
<!ATTLIST relevance relUID IDREF #REQUIRED>
```

（2）在 XML 文档中可以定义 3 个 book，其 UID 分别为 AAAA、AAAB、AAAC，其中 AAAC 和 AAAA 是同一套图书，那么 AAAC 中 relevace 的属性 relUID 的值就是 AAAA，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "attlist_idref_demo.dtd">
<books>
<book UID="AAAA">
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price >69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
<book UID="AAAB">
        <name>《JavaScript 开发技术大全》</name>
        <publisher>人民邮电出版社</publisher>
        <company>明日科技</company>
        <author>梁冰</author>
        <ISBN>9787115179708</ISBN>
        <price>65.00</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=138&sid=5]]></url>
</book>
<book UID="AAAC">
        <name>《Java 从入门到精通（第 2 版）》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>李钟蔚</author>
        <ISBN>9787302227465</ISBN>
        <price>59.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=222]]></url>
        <relevance relUID="AAAA">《C#从入门到精通(第 2 版)》</relevance>
</book>
</books>
```

秘笈心法

心法领悟 335：IDREF 与 IDREFS 之间的比较。

在学习时要知道，使用 IDREF 强调的不是两个元素之间有什么样的关系，而是重点标注哪二者存在关系。如一个元素和多个元素之间产生关联使用 IDREF 无法表达，这就需要使用 IDREFS，表示一个元素和多个元素的关系，其定义方式相似，唯一的不同是在 XML 使用时，IDREFS 定义的属性值的多个值中间使用空格分开，如下所示：

```
<relevance relUIDS="AAAA AAAB">《C#从入门到精通(第 2 版)》 《JavaScript 开发技术大全》</relevance>
```

实例 336　类型为 NMTOKEN 的属性实例

高级

光盘位置：光盘\MR\336

趣味指数：★★★★

实例说明

属性的命名规则都是在 DTD 中定义的，使用 NMTOKEN 可以对属性名称加以限制，从而使命名更规范。如在 XML 文档中声明了一本图书及其各个子元素 name、publisher、company、author、ISBN、price 和 url。在价格元素 price 的 unit 属性上设定 NMTOKEN 类型，就表现在使用 XML 时，unit 要符合 NMTOKEN 的类型，否则对 XML 进行验证时会出现如图 10.43 所示的错误。

图 10.43　不符合 NMTOKEN 类型

关键技术

使用 NTOKEN 可以定义 XML 的属性内容只能设置合法的 XML 名称，如字母、数字、冒号、点号等，示例代码如下：

```
<!ATTLIST price unit NMTOKEN #REQUIRED>
```

设计过程

（1）使用 ELEMENT 定义根元素 books，在其内部定义子元素 book，同时允许 books 中可以有多个 book；使用 ELEMENT 为 book 声明子元素 name、publisher、company、author、ISBN、price 和 url，并为每个子元素设定类型；然后用 ATTLIST 为元素 price 定义一个属性 unit，同时设定 unit 的类型是 NMTOKEN，设置属性参数为#REQUIRED。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT books (book+)>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST price unit NMTOKEN #REQUIRED>
```

（2）在使用 XML 文档时，price 的属性值必须是 NTOKEN 类型。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "attlist_nmtoken_demo.dtd">
<books>
 <book>
        <name>《C#从入门到精通(第 2 版)》</name>
        <publisher>清华大学出版社</publisher>
        <company>明日科技</company>
        <author>王小科</author>
        <ISBN>9787302226628</ISBN>
        <price unit="RMB">69.80</price>
        <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
</books>
```

秘笈心法

心法领悟 336：NMTOKEN 与 NMTOKENS 的区别。

使用 NMTOKEN 定义的属性的类型，表示当前属性的值不能是任意字符，而是只能设置一个合法的 XML 名称。属性值只有一个时，可以用 NMTOKEN 定义；当属性中可以同时存在多个值时，就要用 NMTOKENS 声明属性类型。每个属性值的命名规范和 NMTOKEN 是一致的，不同属性值中间使用空格分开。如果使用 NMTOKENS 定义 unit，如下所示：

```
<!ATTLIST price unit NMTOKEN #REQUIRED>
```

那么在 XML 中可以这样使用：

```
<price unit="RMB USD">69.80</price>
```

第11章

XML Schema

- XSD 的引用与验证
- 使用 XSD 简单类型定义 XML 元素
- 使用 XSD 复杂类型定义 XML 元素
- 使用 XSD 的普通类型

11.1 XSD 的引用与验证

实例 337	在 XML 中使用 XML Schema	中级
	光盘位置：光盘\MR\337	趣味指数：★★★

实例说明

XML Schema 是 DTD 的替代者，也用于描述 XML 文档的结构，比 DTD 的功能更加强大。XML Schema 的文件扩展名是.xsd，如图 11.1 所示为在 XML 文档中引用了 XSD 文档。

图 11.1 在 XML 文档中引用了 XSD 文档

关键技术

在 XML 文档中引用 XSD 文档要规定一个默认命名空间，例如：

```
xmlns=http://www.mingrisoft.com
```

表示在当前 XML 文档中所使用的 XML 元素声明都来自 http://www.mingrisoft.com，然后规定 xsi 的命名空间，如下所示：

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

其中，xsi 表示 XML 文档中使用的元素声明来自 http://www.w3.org/2001/XMLSchema-instance，然后使用 xsi 的 schemaLocation 属性声明 XSD 的位置，例如：

```
xsi:schemaLocation="http://www.mingrisoft.com simple_demo.xsd"
```

其中，schemaLocation 属性有 http://www.mingrisoft.com 和 simple_demo.xsd 两个值。http://www.mingrisoft.com 表示使用的命名空间，simple_demo.xsd 表示命名空间提供的 XSD 的位置。

设计过程

（1）根据需求构思 XML 文档的格式，创建扩展名为.xsd 的 Schema 文档。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">
<xs:element name="book" type="bookType" />
<xs:complexType name="bookType">
        <xs:sequence>
                <xs:element name="name" type="xs:string" />
                <xs:element name="publisher" type="xs:string" />
                <xs:element name="company" type="xs:string" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="ISBN" type="xs:string" />
                <xs:element name="price" type="xs:string" />
```

```
            <xs:element name="url" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
    </xs:schema>
```

（2）根据 Schema 的文档结构创建需要的 XML 文档，然后引用 Schema 文档。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com simple_demo.xsd">
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price>69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 337：DTD 与 Schema 的关系。

DTD 相比 Schema 功能较弱，对 XML 的文档描述差，但是使用 Schema 和 DTD 都可以对 XML 文档结构进行描述，有时也可以同时使用两种方式对同一个 XML 文档做出描述，用户可根据个人习惯选择需要的方式。

实例 338　验证 XML 是否符合 Schema 的描述

光盘位置：光盘\MR\338　　中级　　趣味指数：★★★

实例说明

定义 Schema 后，开发人员即可根据定义的 Schema 建立 XML 文档，在编写 XML 文档时，很可能由于拼写错误导致使用 XML 时出错。本实例实现在 IE 浏览器中验证 XML 是否符合 Schema 的定义，运行效果如图 11.2 所示。

图 11.2　验证 XML 文档是否符合 Schema 的定义

关键技术

使用 JavaScript 中的 ActiveXObject()函数可以获取一个 XMLDOM 对象，语法如下：

```
var xmlDoc = new ActiveXObject("MSXML2.DOMDocument.4.0");
```

参数说明

MSXML2.DOMDocument.4.0：表示在 IE 浏览器获取 XMLDOM 对象。

设计过程

（1）创建一个 XSD 文档，代码见实例 337。

（2）根据 XSD 文档创建一个 XML 文档，同时引入 XSD 文档，代码见实例 337。

（3）先设计一个 HTML 文档并声明一个 MSXML2.DOMDocument.4.0 对象，然后使用 load()方法加载需要验证的 XML 文档，在加载时，系统会自动加载 XSD 文件，同时完成验证工作，如果验证没有错误，parseError.errorCode 的值为 0；如果验证有错误，parseError.errorCode 的值就是相对应的错误代码。另外，还可以通过输出 parseError.reason 和.parseError.line 准确地定位错误的原因和行数。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<style type="text/css">
<!--
#showError {
 color: #F00;
 font-size: 16px;
}
-->
</style></head>
<script type="text/javascript">
function validateXML(filename){
 var txt="";
 if (window.ActiveXObject){
        document.getElementById("showError").innerText ="";
        var xmlDoc = new ActiveXObject("MSXML2.DOMDocument.4.0");
        xmlDoc.async = false;
        xmlDoc.validateOnParse= true;
        xmlDoc.load(filename);
        if(xmlDoc.parseError.errorCode!=0){
        txt="Error Code: " + xmlDoc.parseError.errorCode + "\n";
        txt=txt+"Error Reason: " + xmlDoc.parseError.reason ;
        txt=txt+"Error Line: " + xmlDoc.parseError.line;
      }else{
        txt="没有错误";
      }
        document.getElementById("showError").innerText = txt;
}else{
            alert("此浏览器不支持验证!");
}

}
</script>
<body>
<center>
<h2>通过 XSD 验证 XML 文件</h2>
<p>
<input id="xmlfile" type="text" />
<input type="button" name="submit" value="验 证" onClick="validateXML(document.getElementById('xmlfile').value);" />
</p>
</center>
<div id="showError"></div>
</body>
</html>
```

秘笈心法

心法领悟 338：文档的验证方法。

本实例中，使用 MSXML2.DOMDocument.4.0 对文档进行验证，这需要操作系统和浏览器的支持。有些 XML 编辑器也可以做到这一点，例如 Altova XMLSpy、XML Copy Editor 等，读者可以根据实际情况进行选择。

实例 339　XSD 文档根元素的引用

光盘位置：光盘\MR\339　　中级　　趣味指数：★★★

实例说明

XSD 本身也是一种 XML 文档，也遵守 XML 的语法规范，在定义 XSD 时，也要注意大小写等问题，尽量少用特殊标识符。本实例定义了一个空的 XSD 文档。

关键技术

本实例在 XSD 文档中使用了 xs 作为前缀，声明 xs 的命名空间为 http://www.w3.org/001/XMLSchema，在文档中使用的元素和数据类型都需要以 xs 开头，例如 xs:schema。代码如下：

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

设计过程

（1）建立一个 XSD 文档，为文档定义 XML 版本和编码格式。

（2）创建 XSD 的根节点 xs:schema。

（3）在根节点开始的元素上定义文档的命名空间，使 XSD 下所定义的元素和默认的命名空间都来自 http://www.mingrisoft.com，并使用刚刚定义的命名空间限定 XML 实例文档所使用的所有元素和类型。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
 elementFormDefault="qualified">
</xs:schema>
```

秘笈心法

心法领悟 339：声明命名空间时的参数设置。

实例声明 XSD 的命名空间时，如果使用 elementFormDefault="unqualified"，表示使用刚刚定义的命名空间限定 XML 实例文档使用的全局元素和类型，对局部元素不做限定。

实例 340　在 XSD 中使用注释

光盘位置：光盘\MR\340　　中级　　趣味指数：★★★

实例说明

当 XSD 的内容较多、编写的时间较长时，使用者在理解时可能会有一定困难，为了防止这种情况出现，可以在声明 XSD 时添加一些有助于理解的注释。本实例演示如何设置 XSD 注释。

关键技术

xs:annotation 包含两个子元素，即 xs:appinfo 和 xs:documentation。二者都用于注释，其中，xs:appinfo 一般为工具、样式表或者其他应用程序提供说明信息；xs:documentation 一般用于帮助理解文档内容。示例代码如下：

```
<xs:annotation>
        <xs:documentation>这是一个图书的数据结构，定义了图书的基本信息</xs:documentation>
</xs:annotation>
```

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。

（2）在 XSD 文档中添加 xs:annotation 元素，在其内部声明 xs:documentation，然后添加注释信息。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">

<xs:annotation>
      <xs:documentation>这是一个图书的数据结构，定义了图书的基本信息</xs:documentation>
</xs:annotation>

</xs:schema>
```

秘笈心法

心法领悟 340：设置注释的语言种类。

在使用 xs:documentation 定义注释时，可以使用其 xml:lang 属性来设置内容使用的语言种类，例如 xml:lang="en"。

11.2 使用 XSD 简单类型定义 XML 元素

实例 341 在 XSD 中定义 XML 元素　　中级

光盘位置：光盘\MR\341　　趣味指数：★★★

实例说明

一个最基本的 XML 文档，就是必须要有一个根元素。本实例使用 xs:elmenet 声明一个元素，然后定义一个 XML 文档，如图 11.3 所示。

图 11.3　定义 XML 元素

关键技术

使用 xs:element 可以为 XML 定义一个元素，示例代码如下：

```
<xs:element name="book" type="xs:string" fixed="明日科技图书"></xs:element>
```

参数说明

❶ book：表示 XML 文件中元素的名称。

❷ xs:stirng：表示 XML 文件的元素类型是 xs:string。

❸ fixed：是 XML 元素的选填项，如果使用 fixed 参数，那么 book 的内容就必须是 fixed 定义的值；如果没有使用 fixed 参数，book 中的内容则不受限制。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。使用 xs:element 定义 XML 元素的根节点 book，然后限制 XML 元素节点的类型和字的内容。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
 elementFormDefault="qualified">
 <xs:element name="book" type="xs:string" fixed="明日科技图书"></xs:element>
</xs:schema>
```

（2）在 XML 文档中引用 XSD 文件，然后添加 book 元素。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.mingrisoft.com element_demo.xsd">明日科技图书</book>
```

秘笈心法

心法领悟 341：XSD 中元素属性值的设定。

本实例在 book 上使用 fixed="明日科技图书"限定了 book 元素已经确定的、不可更改的值。xs:element 还有另外一个属性 default，当元素内容为空时，解析器取 XML 中 default 中定义的默认值。同时还要注意，由于这两个属性的特性，fixed 和 default 相互矛盾，不能同时使用在一个 xs:element 中。

实例 342　使用 xs:simpleType 和 xs:restriction

光盘位置：光盘\MR\342　　高级　　趣味指数：★★★★

实例说明

xs:simpleType 表示在 XSD 中可以使用简单类型约束 XML 的元素，它只能起到声明的作用，要对元素内容进行限定，还必须和 xs:restriction 配合使用。如果元素内容超过限定约束，验证 XML 时就会出现错误，效果如图 11.4 所示。

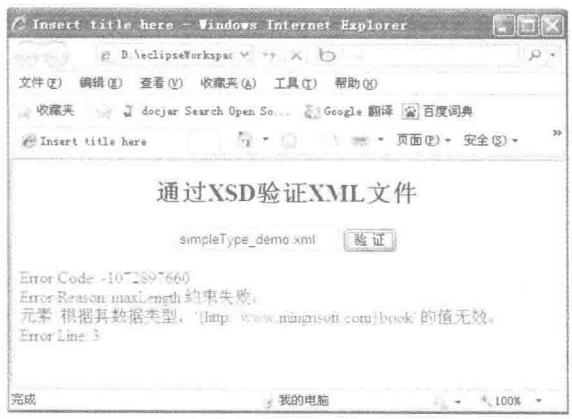


图 11.4　使用 xs:maxLength 限制元素长度

关键技术

使用 xs:simpleType 定义一个简单类型，可以声明要约束的 XML 元素的属性或元素的内容，但 xs:simpleType 只起到声明的作用，使用 xs:simpleType 的子节点 xs:restriction 才可以对元素进行限定。xs:restriction 表示要对当前的 xs:element 元素添加一个约束，属性 base 限制了 xs:element 内容的类型。代码如下：

```
<xs:simpleType>
        <xs:restriction base="xs:string">
        <xs:maxLength value="5"></xs:maxLength>
        </xs:restriction>
</xs:simpleType>
```

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。使用 xs:element 定义 XML 元素的根节点 book，在 element 内部定义一个简单类型 xs:simpleType，在 xs:simpleType 中限制 book 的类型是 xs:string，并在 xs:restriction 内部添加 xs:maxLength value="5"，从而限制 book 的内容不能超过 5 个字符。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">

<xs:element name="book">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                <xs:maxLength value="5"></xs:maxLength>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
</xs:schema>
```

（2）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com simpleType_demo.xsd">12345</book>
```

提示：在编写 XML 文档时，当元素内容为汉字时，解析器认为一个汉字和一个英文字母都是一个字符。

秘笈心法

心法领悟 342：XML 文档中字符串长度的限定。

实例中使用 xs:maxLengthvalue="5"对字符串限定了 5 个字符，表示在 XML 文档中，book 元素的内容最大值只能是 5 个字符，即使在元素内容中间插入回车、换行或者空格，都会占有相应的字符数量。

实例 343 使用 xs:list

光盘位置：光盘\MR\343

高级

趣味指数：★★★

实例说明

一般情况下，一个 XML 元素内只有一个内容，使用 xs:list 可以限定 XML 元素在使用时，可以是多个内容的集合。例如，正常情况下，一本图书只有一个价格，但是图书通过网络销售可以降低一部分成本，从而会在网络上产生另一个价格，这时一本图书就可以有两个价格，在使用 XML 定义图书时就可以填写两个价格，并用空格分开，运行效果如图 11.5 所示。

关键技术

使用 xs:simpleType 可以声明一个简单类型，在内部使用 xs:list 定义类型 itemType="xs:double"可定义一个 xs:double 集合类型。在 xs: restriction 中通过 base 引用集合类型，表示当前的 xs:element 是一个 xs:double 的集合类型。代码如下：

```
<xs:restriction base="pricetype">
        <xs:maxLength value="2"></xs:maxLength>
</xs:restriction>
……
```

```
<xs:simpleType name="pricetype">
        <xs:list itemType="xs:double"></xs:list>
</xs:simpleType>
```

图 11.5　使用 list 限定 price 的集合类型

参数说明

❶ 当 xs:restriction 的 base 是一个基本类型（如 xs:string 等）时，使用 xs:maxLength 可以约束 xs:element 中内容的长度。

❷ 当 xs:restriction 的 base 的类型是一个集合类型 xs:list 时，xs:maxLength 用来约束 xs:element 内容的个数，多个内容之间使用空格分开。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点以及其下的子元素，并为 name、publisher、company、author、ISBN 和 url 子元素添加 xs:string 类型限定。然后声明一个 name="pricetype"独立的 xs:simpleType，定义内部元素 xs:list 的 itemType 类型是 xs:double。最后为 price 添加一个简单类型并且定义 xs:maxLength value="2"，再引用外部定义好的简单类型 base="pricetype"。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">
<xs:element name="book" type="bookType" />
<xs:complexType name="bookType">
        <xs:sequence>
                <xs:element name="name" type="xs:string" />
                <xs:element name="publisher" type="xs:string" />
                <xs:element name="company" type="xs:string" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="ISBN" type="xs:string" />
                <xs:element name="price">
                        <xs:simpleType>
                                <xs:restriction base="pricetype">
                                        <xs:maxLength value="2"></xs:maxLength>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:element>
                <xs:element name="url" type="xs:string" />
        </xs:sequence>
</xs:complexType>
<xs:simpleType name="pricetype">
        <xs:list itemType="xs:double"></xs:list>
</xs:simpleType>
</xs:schema>
```

（2）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.mingrisoft.com list_demo.xsd">

 <name>《C#从入门到精通(第 2 版)》</name>
 <publisher>清华大学出版社</publisher>
 <company>明日科技</company>
 <author>王小科</author>
 <ISBN>9787302226628</ISBN>
 <price>69.80 69.00</price>
 <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>

 </book>
```

秘笈心法

心法领悟 343：XSD 中对集合的类型及数量的限定。

实例中不但在 XSD 中限定了集合类型，同时还限定了集合的最大数量，这需要联合使用 xs:restriction 和 xs:maxLength。在 xs:restriction 内部还有一个 xs:minLength 元素，可用于限定集合的最小数量，如果有需要，也可以将 xs:minLength 联合其他限定一起使用。

实例 344　使用 xs:enumeration

光盘位置：光盘\MR\344　　高级　　趣味指数：★★★

实例说明

大多数情况下，XML 元素的内容都是由用户自己定义的，XSD 中只限定了 XML 元素的类型。使用 xs:enumeration 可以限定 XML 元素的内容。本实例中，author 的元素内容被 XSD 所限定，如果没有使用 XSD 限定的内容，在验证时会出现如图 11.6 所示的错误。

图 11.6　author 使用错误

关键技术

xs:restriction 表示声明一个约束，xs:enumeration 用在 xs:restriction 中，确定一个枚举类型的约束。示例代码如下：

```
<xs:restriction base="xs:string">
      <xs:enumeration value="王小科"></xs:enumeration>
      <xs:enumeration value="徐薇"></xs:enumeration>
</xs:restriction>
```

参数说明

❶ xs:restriction 的 base="xs:string"声明 XML 的内容必须是一个字符串。

❷ xs:enumeration 的属性 value 声明 XML 中要使用的枚举值，而且该值必须符合 xs:restriction 中的 base 约束，即为 xs:string 类型。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点以及其下的子元素，并为 name、publisher、company、ISBN 和 url 子元素添加 xs:string 类型限定。然后声明一个 name="prictype"独立的 xs:simpleType，定义内部元素 xs:list 的 itemType 类型是 xs:double。再为 price 添加一个简单类型，定义 xs:maxLength value="2"，并引用外部定义好的简单类型 base="pricetype"。最后为 author 声明一个简单类型，在 xs:restriction 内部声明 xs:enumeration，并添加固定枚举类型的值。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">

<xs:element name="book">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="publisher" type="xs:string" />
            <xs:element name="company" type="xs:string" />
            <xs:element name="author">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="王小科"></xs:enumeration>
                        <xs:enumeration value="徐薇"></xs:enumeration>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="ISBN" type="xs:string" />
            <xs:element name="price">
                <xs:simpleType>
                    <xs:restriction base="pricetype">
                        <xs:maxLength value="2"></xs:maxLength>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="url" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:simpleType name="pricetype">
    <xs:list itemType="xs:double"></xs:list>
</xs:simpleType>
</xs:schema>
```

（2）根据 XSD 的限定，xs:restriction 中含有两个 xs:enumeration，其值分别是“王小科”和“徐薇”，表示在 XML 元素 author 中的内容必须是“王小科”或者“徐薇”，本实例 XML 中 author 的内容为“王小科”，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com enumeration_demo.xsd">
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price>69.80 69.00</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 344：声明 XSD 文档时对类型的声明。

使用 xs:enumeration 声明 XSD 文档时，其值要与 xs:restriction 声明的类型一致，若 xs:restriction 声明的类型是 xs:string，xs:enumeration 的 value 值也必须是 xs:string 类型的；若 xs:restriction 声明的类型是 xs:double，xs:enumeration 内容也必须是 xs:double 类型的，二者相互关联。

实例 345 使用 xs:pattern

光盘位置：光盘\MR\345　　高级　　趣味指数：★★★

实例说明

可以通过 xs:enumeration 将 XML 的元素内容定义成枚举值，也可以使用 xs:pattern 将 XML 内容定义成一定的样式，使其符合一定的规则，如果违反了这种规则，解析器检测时会出现如图 11.7 所示的错误。

图 11.7　违反 xs:pattern 使用规则

关键技术

xs:pattern 用在 xs:restriction 内部，用来限定 XML 中元素使用固定的格式，格式的写法使用正则表达式来表示，示例代码如下：

```
<xs:pattern value="[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]" />
```

其中，value="[0-9]"表示 0~9 之间的任意一个数字，那么 value="[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]"就表示应该有 13 个数字，且每个数字只能在 0~9 之间。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点以及其下的子元素，并为 name、publisher、company、author、price 和 url 子元素添加 xs:string 类型限定；为 ISBN 声明一个简单类型，定义 xs:restriction 为 xs:integer。在 xs:restriction 内定义一个 xs:parttern，设置 value="[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]"。在 XML 中使用 ISBN 时，元素内部必须填写 13 个连续的数字，每个数字为 0~9 之间的任意一个即可。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
 elementFormDefault="qualified">
 <xs:element name="book" type="bookType" />
 <xs:complexType name="bookType">
        <xs:sequence>
                <xs:element name="name" type="xs:string" />
                <xs:element name="publisher" type="xs:string" />
                <xs:element name="company" type="xs:string" />
                <xs:element name="author" type="xs:string" />
```

```
            <xs:element name="ISBN">
                <xs:simpleType>
                    <xs:restriction base="xs:integer">
                        <xs:pattern value="[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]" />
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="price" type="xs:string" />
            <xs:element name="url" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
    </xs:schema>
```

（2）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com pattern_demo.xsd">
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price>69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 345：XSD 文档中正则表达式的运用。

在 xs:pattern 中使用的正则表达式，可通过某种模式去匹配一类字符串的一个公式。使用正则表达式来表示的字符串非常灵活，而且表达起来简单、功能强大，有兴趣的读者可以查阅相关资料，这里不再讲述。

11.3　使用 XSD 复杂类型定义 XML 元素

实例 346　使用 xs:complexType 和 xs:sequence　　高级

光盘位置：光盘\MR\346　　趣味指数：★★★

实例说明

xs:sequence 要求 XML 中的元素以指定的顺序排列在 XML 中，在 xs:sequence 内声明的 xs:element 顺序就是在 XML 文档中使用的顺序。如果 XML 中没有按 xs:sequence 规定的顺序使用，验证时会出现如图 11.8 所示的错误。

图 11.8　publisher 没有按规定顺序使用

关键技术

xs:complexType 可以声明一个复杂类型，在 XSD 中使用复杂类型约束 XML 的元素，需要用 xs:complexType 与其他元素联合使用。例如在 xs:complexType 内部定义一个 xs:sequence，可共同对 XML 元素或属性进行限定。示例代码如下：

```
<xs:complexType>
    <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="publisher" type="xs:string" />
        <xs:element name="company" type="xs:string" />
        <xs:element name="author" type="xs:string" />
        <xs:element name="ISBN" type="xs:string" />
        <xs:element name="price" type="xs:string" />
        <xs:element name="url" type="xs:string" />
    </xs:sequence>
</xs:complexType>
```

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。使用 xs:element 定义 XML 元素的根节点 book，在 element 内部定义一个复杂类型 xs:complexType。然后在 xs:complextpType 内部添加 xs:sequence，声明一个有顺序的元素定义，再在 xs:sequence 内部使用 xs:element 声明 book 的子元素 name、publisher、company、author、ISBN、price 和 url，并且定义这几个元素为 xs:string 类型。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
 elementFormDefault="qualified">
 <xs:element name="book">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="publisher" type="xs:string" />
            <xs:element name="company" type="xs:string" />
            <xs:element name="author" type="xs:string" />
            <xs:element name="ISBN" type="xs:string" />
            <xs:element name="price" type="xs:string" />
            <xs:element name="url" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
 </xs:element>
</xs:schema>
```

（2）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.mingrisoft.com simple_demo.xsd">
 <name>《C#从入门到精通(第 2 版)》</name>
 <publisher>清华大学出版社</publisher>
 <company>明日科技</company>
 <author>王小科</author>
 <ISBN>9787302226628</ISBN>
 <price>69.80</price>
 <url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 346：xs:complexType 定义公用复杂类型。

xs:complexType 可以在 xs:element 中定义一个元素的复杂类型，若需要定义多个一样的复杂类型，可以把 xs:complexType 提取出来，声明一个公共的复杂类型供其他 xs:element 使用。提取 xs:complexType 定义后，要为其添加一个属性 name，且 name 在文档中必须是唯一的，这样才能保证在引用时不会有冲突。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">
<xs:element name="book" type="bookType" />
<xs:complexType name="bookType">
    <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="publisher" type="xs:string" />
        <xs:element name="company" type="xs:string" />
        <xs:element name="author" type="xs:string" />
        <xs:element name="ISBN" type="xs:string" />
        <xs:element name="price" type="xs:string" />
        <xs:element name="url" type="xs:string" />
    </xs:sequence>
</xs:complexType>
</xs:schema>
```

实例 347　使用 xs:choice

光盘位置：光盘\MR\347　　高级　　趣味指数：★★★

实例说明

一本图书的元素有 name、publisher、company 和 author 等，其中，company 表示出版图书的公司，author 表示出版图书的作者，有时只需知道作者名字或者出版图书的公司名称即可，不希望二者同时出现，这时可以使用 xs:choice 定义 XML，一旦同时使用，验证时会出现如图 11.9 所示的错误。

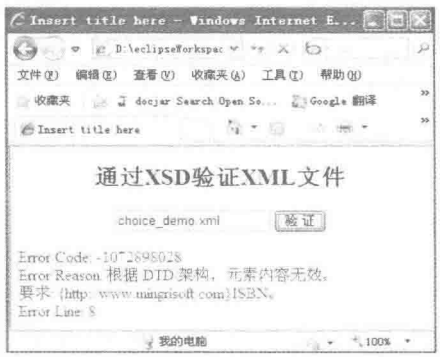


图 11.9　XML 元素没有按定义使用

关键技术

在 XML 中，不希望两个或者两个以上的元素同时出现时，可以使用 xs:choice 进行限定。xs:choice 可以用在多个地方，如用在 xs:sequence、xs:complexType、xs:restriction 和 xs:extension 中，对这些元素进行扩展；也可以放在 xs:choice 内部对其本身进行扩展。

xs:choice 内部可以有多个 xs:element，每一个 xs:element 在 XML 文档中都不能同时出现，用户根据需要选则其中一个 xs:element 定义的元素使用。xs:choice 的使用代码如下：

```
<xs:choice>
    <xs:element name="company" type="xs:string" />
    <xs:element name="author" type="xs:string" />
</xs:choice>
```

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点以及其下的子元素，并为 name、

publisher、ISBN、price 和 url 子元素添加 xs:string 类型限定；在 publisher 下面声明一个 xs:choice，在 xs:choice 中添加两个 xs:element，分别声明 company 和 author，并为定义的元素添加使用类型 xs:string。这样在使用 XML 时，company 和 author 就不可以同时出现在 book 元素内部了，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">

<xs:element name="book" type="bookType" />

<xs:complexType name="bookType">
    <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="publisher" type="xs:string" />
        <xs:choice>
            <xs:element name="company" type="xs:string" />
            <xs:element name="author" type="xs:string" />
        </xs:choice>
        <xs:element name="ISBN" type="xs:string" />
        <xs:element name="price" type="xs:double" />
        <xs:element name="url" type="xs:string" />
    </xs:sequence>
</xs:complexType>
</xs:schema>
```

（2）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com choice_demo.xsd">
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<ISBN>9787302226628</ISBN>
<price>69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 347：xs:chioce 的相关用法。

xs:chioce 的定义声明了元素在 XML 中的使用情况，xs:chioce 还有两个属性：maxOccurs 和 minOccurs，其中，maxOccurs 表示 xs:chioce 声明的元素在 XML 中使用时出现的最大次数；minOccurs 表示 xs:chioce 声明的元素在 XML 中使用时出现的最小次数。

实例 348 使用 xs:all

光盘位置：光盘\MR\348　　高级　　趣味指数：★★★

实例说明

在 XML 元素中，有时用户可能对元素的使用情况限制的比较严格，甚至不能调换 XML 中元素的上下位置，但有时对 XML 的文档要求比较宽松，允许随意调换元素的位置。本实例建立一个各个元素的位置可以相互调换的 XML 文档，如图 11.10 所示。

关键技术

xs:all 一般用在 xs:complexType 内部，用来声明一个复杂元素，声明时，需要声明一组 xs:element，每个 xs:element 定义的元素在 XML 使用时是没有先后顺序的，代码如下：

```
<xs:all>
    <xs:element name="name" type="xs:string" />
```

```
        <xs:element name="publisher" type="xs:string" />
        <xs:element name="company" type="xs:string" />
        <xs:element name="author" type="xs:string" />
        <xs:element name="ISBN" type="xs:string" />
        <xs:element name="price" type="xs:double" />
        <xs:element name="url" type="xs:string" />
</xs:all>
```

图 11.10　元素之间的位置可以相互调换

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。然后声明一个复杂类型，设定 xs:complexType 的 name="bookType"。再在 xs:complexType 内部声明 xs:all，准备为复杂元素添加 xs:element，分别声明 name、publisher、company、author、ISBN、price 和 url 等元素，同时设定其类型。最后为声明 book 的 xs:element 添加 type 属性，设置属性为刚刚定义完的复杂类型 bookType。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com" elementFormDefault="qualified">
<xs:element name="book" type="bookType" />
<xs:complexType name="bookType">
        <xs:all>
                <xs:element name="name" type="xs:string" />
                <xs:element name="publisher" type="xs:string" />
                <xs:element name="company" type="xs:string" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="ISBN" type="xs:string" />
                <xs:element name="price" type="xs:double" />
                <xs:element name="url" type="xs:string" />
        </xs:all>
</xs:complexType>
</xs:schema>
```

（2）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com all_demo.xsd">

<publisher>清华大学出版社</publisher>
<name>《C#从入门到精通(第 2 版)》</name>
<company>明日科技</company>
<author>王小科</author>
<price>69.80</price>
<ISBN>9787302226628</ISBN>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>

</book>
```

秘笈心法

心法领悟 348：使用 xs:all 声明的 xs:element 的使用规则。

使用 xs:all 声明的 xs:element 在 XML 中的顺序是没有限制的，但是使用时，不可以随意减少或者增加元素，否则解析器在验证时会出现如图 11.11 所示的错误。

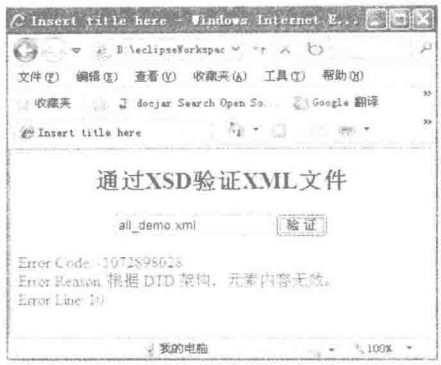


图 11.11　xs:all 声明的元素没有正确使用

实例 349　使用 xs:group

光盘位置：光盘\MR\349　　高级　　趣味指数：★★★★

实例说明

在定义 XML 格式时，会声明很多重复的内容，一个最好的解决办法就是把重复的内容提取出来，在需要时进行调用，但是不管 XML 是怎样声明的，使用时都是一样的，不会有差别。本实例创建使用 xs:group 限制的 XML 文档，如图 11.12 所示。

图 11.12　使用 xs:group 限制的 XML 文档

关键技术

xs:group 内部可以声明 xs:choice、xs:sequence 和 xs:all 元素，在 xs:choice、xs:sequence 或者 xs:all 内部都可以定义多个 xs:element 元素，为每个 xs:element 声明独自的名称和类型，就形成一个 xs:element 组。代码如下：

```
<xs:group name="otherType">
    <xs:sequence>
        <xs:element name="publisher" type="xs:string" />
        <xs:element name="company" type="xs:string" />
```

```
            <xs:element name="author" type="xs:string" />
            <xs:element name="ISBN" type="xs:string" />
            <xs:element name="price" type="xs:double" />
            <xs:element name="url" type="xs:string" />
        </xs:sequence>
    </xs:group>
```

其中，name="otherType"表示该 xs:element 组的名字。需要在文档的其他地方使用时，定义 xs:group 并且使用 ref="otherType"即可引用该组。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。定义一个 xs:element 的 name="book"为一个根节点。在根节点下声明一个复杂类型 xs:complexType，并使用 xs:sequnce 扩展 xs:complexType。在 xs:sequnce 下定义一个 xs:element 元素，设置 name="name"表示在根元素下面定义一个子节点，并设置其 type="xs:string"。

（2）独立声明一个 xs:group，设置其属性 name="otherType"，扩展 xs:sequence 元素，在 xs:sequence 下面定义一组元素 publisher、company、author、ISBN、price 和 url。然后在声明 book 根元素的里面、xs:element 的上面插入定义好的元素组 ref="otherType"。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">
<xs:element name="book">
    <xs:complexType>
        <xs:sequence>
            <xs:group ref="otherType"></xs:group>
            <xs:element name="name" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:group name="otherType">
    <xs:sequence>
        <xs:element name="publisher" type="xs:string" />
        <xs:element name="company" type="xs:string" />
        <xs:element name="author" type="xs:string" />
        <xs:element name="ISBN" type="xs:string" />
        <xs:element name="price" type="xs:double" />
        <xs:element name="url" type="xs:string" />
    </xs:sequence>
</xs:group>

</xs:schema>
```

（3）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com group_demo.xsd">
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price>69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
<name>《C#从入门到精通(第 2 版)》</name>
</book>
```

秘笈心法

心法领悟 349：XML 文件中对 xs:group 的限制。

在使用 xs:group 时，把 xs:group 引用放在了 xs:element 上面，由于它们都被包含在 xs:sequnce 内部，所以说明了 xs:group 元素在 XML 文档中是有顺序限制的。同时在 xs:group 内部也使用了 xs:sequnce，说明 xs:group 内部也是有顺序限制的。

实例 350　使用 xs:extension 和 xs:simpleContent

光盘位置：光盘\MR\350　　高级　　趣味指数：★★★★

实例说明

XSD 可以为 XML 的元素定义属性及其属性的类型，可以更完整地定义 XML 文档，以方便用户使用。如果 XML 属性被定义而不使用，在验证时则会报错，如图 11.13 所示。

图 11.13　缺少 RMB 属性

关键技术

使用 xs:simpleContext 和 xs:extension 可以限制元素类型、属性名称和属性类型。首先在 xs:element 内部声明一个复杂元素 xs:complexType，然后在 xs:complexType 中声明 xs:simpleContent，并且在其内部声明 xs:extension，使用 xs:extension 来限制 xs:element 定义的元素类型。另外，使用 xs:attribute 可以在 xs:extension 内部声明当前元素的属性，代码如下：

```
<xs:complexType>
    <xs:simpleContent>
        <xs:extension base="xs:double">
            <xs:attribute name="RMB" type="xs:string" use="required"></xs:attribute>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

参数说明

❶ base="xs:double"：表示 xs:extension 所在的元素属性是一个 xs:double 类型。

❷ name="RMB"：表示元素的属性名称为 RMB。

❸ type="xs:string"：表示属性的类型。

❹ use="required"：表示属性是必须填写的，否则验证时会报错。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点以及其下的子元素，并为 name、publisher、company、author、ISBN 和 url 子元素添加 xs:string 类型限定。再为 price 声明一个复杂类型，在 xs:complexType 扩展 xs:simpleContent、xs: extension 声明，定义 xs:extension 属性 base="xs:double"。然后在 xs:extension 基础上扩展 xs:attribute 属性，同时定义 name="RMB"、type="xs:string"和 use="required"。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com" elementFormDefault="qualified">
 <xs:element name="book" type="bookType" />
```

```
<xs:complexType name="bookType">
    <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="publisher" type="xs:string" />
        <xs:element name="company" type="xs:string" />
        <xs:element name="author" type="xs:string" />
        <xs:element name="ISBN" type="xs:string" />
        <xs:element name="price">
            <xs:complexType>
                <xs:simpleContent>
                    <xs:extension base="xs:double">
                        <xs:attribute name="RMB" type="xs:string" use="required"></xs:attribute>
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
        </xs:element>
        <xs:element name="url" type="xs:string" />
    </xs:sequence>
</xs:complexType>
</xs:schema>
```

（2）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com extension_demo.xsd">
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price RMB="yuan">69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 350：XSD 中 use 参数的用法。

使用 xs:attribute 扩展 xs:extension 声明 XSD 的属性，定义 use="required"表示该属性在 XML 中必须使用，use 还可以填写另外两个值：optional 和 prohibited，use="optional"时，表示 RMB 属性在 XML 中是可以选择使用的；use="prohibited"时，表示 RMB 属性在 XML 文档中是被禁止使用的。

实例 351　使用 xs:extension 和 xs:complexContent

高级

光盘位置：光盘\MR\351

趣味指数：★★★★

实例说明

在一本图书的 XML 中，price 表示图书的价格，在某些特殊的情况下，图书的价格可能会有两个：一个是原价；另一个是打折以后的价格。那么 price 的元素中就会有 orginal 和 dicount 两个元素分别记录原价和打折以后的价格。通过 xs:complexContent 和 xs:extension，可以使用外部定义的元素和属性。在 XSD 定义中，orginal 表示原价，可以直接定义在 xs:extension 内部；discount 表示折扣价格，一般都是后期根据库存等情况制定的，所以把它定义在 price 元素外部，在使用时需要用 xs:extension 的 base 属性来引用，效果如图 11.14 所示。

关键技术

在 xs:extension 内部定义元素 original，同时设置 xs:extension 的属性 base="discountPric"来引用外部定义的 discount 元素，这样既保证了原有的 original 元素，还引用了 discount 元素，代码如下：

```
<xs:extension base="discountPrice">
    <xs:sequence>
```

```
            <xs:element name="original" type="xs:double" />
        </xs:sequence>
        <xs:attribute name="RMB" type="xs:string" use="required" />
</xs:extension>
.....
<xs:complexType name="discountPrice">
        <xs:sequence>
            <xs:element name="discount" type="xs:double" />
        </xs:sequence>
</xs:complexType>
```

图 11.14　使用扩展元素

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点以及其下的子元素，并为 name、publisher、company、author、ISBN 和 url 子元素添加 xs:string 类型限定。再为 price 声明一个复杂类型，在 xs:complexType 基础上扩展 xs:complexContent、xs: extension 声明，设置 xs:extension 属性 base="discountPric"。然后在 xs:extension 基础上扩展 xs:attribute 属性，同时定义 name="RMB"、type="xs:string"和 use="required"。最后在声明 book 元素的外部定义一个 xs:complexType，扩展 xs:complexType 定义 discount 元素，设置 xs:complexType 的 name="discountPric"。代码如下：

```
    <?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">

<xs:element name="book">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="publisher" type="xs:string" />
            <xs:element name="company" type="xs:string" />
            <xs:element name="author" type="xs:string" />
            <xs:element name="ISBN" type="xs:string" />
            <xs:element name="price">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="discountPrice">
                            <xs:sequence>
                                <xs:element name="original" type="xs:double" />
                            </xs:sequence>
                            <xs:attribute name="RMB" type="xs:string" use="required" />
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="url" type="xs:string" />
        </xs:sequence>
```

```
        </xs:complexType>
    </xs:element>

    <xs:complexType name="discountPrice">
        <xs:sequence>
            <xs:element name="discount" type="xs:double" />
        </xs:sequence>
    </xs:complexType>

        </xs:schema>
```

（2）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com extension_complexContent_demo.xsd">
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price RMB="yuan">
        <discount>69.80</discount>
        <original>69.00</original>
</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

秘笈心法

心法领悟 351：xs:complexContent 和 xs:extension 的用法。

使用 xs:complexContent 和 xs:extension 定义元素 original 时，同时也会引用一个复杂类型，在引入的这个复杂类型中定义了 discount 元素，这两个元素分别定义在不同的 xs:sequnce 内部，但是在 XML 中使用时，要先使用 xs:extension 通过 base 引用的元素（即 discount），再使用 xs:extension 内部定义的元素，否则在验证时会出现如图 11.15 所示的错误。

图 11.15　通过 xs:extension 引入的元素顺序使用错误

实例 352　使用 xs:restriction 和 xs:simpleContent

高级

光盘位置：光盘\MR\352

趣味指数：★★★

实例说明

一个合理的图书价格应当在一定的区间范围内，所以在定义 XSD 时，可以对元素值的范围进行限制，如果超出该范围，验证时会出现如图 11.16 所示的错误。

图 11.16　XML 中 price 低于限定的最小值

关键技术

使用 xs:simpleContent 和 xs:restriction 可以限定元素值的内容，示例代码如下：

```
<xs:simpleContent>
    <xs:restriction base="basePriceType">
        <xs:minExclusive value="0" />
        <xs:maxExclusive value="1000.00" />
    </xs:restriction>
</xs:simpleContent>
```

参数说明

❶ xs:minExclusive：表示限定元素的最小值。

❷ xs:maxExclusive：表示限定元素的最大值。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点以及其下的子元素，并为 name、publisher、company、author、ISBN 和 url 子元素添加 xs:string 类型限定。声明一个复杂类型 basePriceType，使用 xs:simpleContent 和 xs:extension 对 xs:complexType 进行扩展，定义 xs:extension 的基本类型为 xs:double，同时声明一个属性 RMB。这样就限定了元素 price 的值必须是一个 xs:double 类型，值在 0~1000 之间，同时 price 元素还有一个必填的属性 RMB。

声明另一个复杂类型 priceType，并扩展 xs:simpleContent、xs:restriction，使用 xs:restriction 引用刚才声明的 basePriceType 类型，并在 xs:restriction 内部限定一个最大值和最小值。然后在 book 的元素内部定义一个元素 price，设置类型 type="priceType"，这样 price 既有了 basePriceType 的特性，同时又被 priceType 进行了约束。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">
<xs:element name="book">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="publisher" type="xs:string" />
            <xs:element name="company" type="xs:string" />
            <xs:element name="author" type="xs:string" />
            <xs:element name="ISBN" type="xs:string" />
            <xs:element name="price" type="priceType" />
            <xs:element name="url" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:complexType name="basePriceType">
    <xs:simpleContent>
```

```
            <xs:extension base="xs:double">
                <xs:attribute name="RMB" use="required"></xs:attribute>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="priceType">
        <xs:simpleContent>
            <xs:restriction base="basePriceType">
                <xs:minExclusive value="0" />
                <xs:maxExclusive value="1000.00" />
            </xs:restriction>
        </xs:simpleContent>
    </xs:complexType>
    </xs:schema>
```

（2）根据 XSD 的定义，普通的图书价格定义 price 元素为 69.80，它在 0~1000 元之间符合 XSD 的定义，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com restriction_simpleContent_demo.xsd">
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price RMB="yuan">69.80</price>
<url>http://www.mingribook.com/bookinfo.php?id=227</url>
</book>
```

秘笈心法

心法领悟 352：XSD 文档中对元素内容的约束。

使用 xs: minExclusive 和 xs: maxExclusive 对元素的内容进行约束时，xs:minExclusive 表示内容的最小值，xs:maxExclusive 表示内容的最大值，说明 XML 元素的内容必须在这两个值之间而不包含这两个值，例如设置 xs:minExclusive value="0"时，若在 XML 中元素内容使用 0，验证时解析器同样会报出约束失败的错误。

实例 353　使用 xs:restriction 和 xs:complexContent　高级

光盘位置：光盘\MR\353　　趣味指数：★★★

实例说明

在定义 XSD 时，有时先声明一个基本的元素对象，然后根据不同的需要，对这个基本的元素声明进行扩展，即在原有的元素声明基础上，不需要改变原来元素的定义，对定义进一步完善，这时可以使用 xs:complexContent 和 xs:restriction。本实例的 XML 就使用了这种定义方式的 XSD，如图 11.17 所示。

图 11.17　使用 xs:complexContent 和 xs:restriction 定义的 XML

关键技术

使用 xs:complexContent 和 xs:restriction 可以对原有元素的声明进行扩展，代码如下：

```
<xs:complexType name="bookType">
    <xs:complexContent>
        <xs:restriction base="baseBookType">
            <xs:sequence>
                <xs:element name="name" type="xs:string" />
                <xs:element name="publisher" type="xs:string" />
                <xs:element name="company" type="xs:string" fixed="明日科技" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="ISBN" type="xs:string" />
                <xs:element name="price" type="xs:double" />
                <xs:element name="url" type="xs:string" />
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
```

参数说明

❶ base="baseBookType"：表示原有的元素定义的名称。

❷ name="bookType"：表示对原有元素定义扩展的名称。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 books 根节点，在其下面定义子元素 book，声明一个复杂类型 baseBookType 作为基本的元素声明，再定义 book 的子元素，并为 name、publisher、company、author、ISBN 和 url 子元素添加 xs:string 类型限定，为 price 添加 xs:double 类型。然后声明另外一个复杂类型 priceType，扩展 xs:complexContent 和 xs:restriction 元素并设置 xs:restriction 属性 base="baseBookType"，覆盖 baseBookType 的声明；添加元素 company 的属性 fixed="明日科技"，表示 company 的内容已经被限定为"明日科技"。在根元素的内部添加 xs:element 元素，为 book 设置类型 bookType。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">
<xs:element name="books">
    <xs:complexType>
        <xs:all>
            <xs:element name="book" type="bookType" />
        </xs:all>
    </xs:complexType>
</xs:element>
<xs:complexType name="baseBookType">
    <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="publisher" type="xs:string" />
        <xs:element name="company" type="xs:string" />
        <xs:element name="author" type="xs:string" />
        <xs:element name="ISBN" type="xs:string" />
        <xs:element name="price" type="xs:double" />
        <xs:element name="url" type="xs:string" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="bookType">
    <xs:complexContent>
        <xs:restriction base="baseBookType">
            <xs:sequence>
                <xs:element name="name" type="xs:string" />
                <xs:element name="publisher" type="xs:string" />
                <xs:element name="company" type="xs:string" fixed="明日科技" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="ISBN" type="xs:string" />
                <xs:element name="price" type="xs:double" />
```

```
                    <xs:element name="url" type="xs:string" />
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    </xs:schema>
```

（2）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.mingrisoft.com restriction_complexContent_demo.xsd">
 <name>《C#从入门到精通(第 2 版)》</name>
 <publisher>清华大学出版社</publisher>
 <company>明日科技</company>
 <author>王小科</author>
 <ISBN>9787302226628</ISBN>
 <price RMB="yuan">69.80</price>
 <url>http://www.mingribook.com/bookinfo.php?id=227</url>
 </book>
```

秘笈心法

心法领悟 353：扩展原有的基本元素声明的方法。

在使用 xs:complexContent 和 xs:restriction 对原有的基本元素声明进行扩展时，要把 xs:restriction 引用定义的基本声明进行重写覆盖，然后在此基础上进行扩展，否则不符合 xs:complexContent 和 xs:restriction 的使用规范。

实例 354　使用 xs:attributeGroup 和 xs:attribute

光盘位置：光盘\MR\354　　高级　　趣味指数：★★★

实例说明

定义 XSD 时，除了定义元素，还要定义属性，在使用 XML 时，属性也占有很大部分。定义一本图书的 XML 元素包括图书名称、出版社、作者和价格等。以作者为例，可以为作者添加其联系方式，以此作为一种属性，如 qq、msn、tel，如图 11.18 所示。

图 11.18　含有多个属性的 XML

关键技术

使用 xs:attribute 和 xs:attributeGroup 可以定义一个属性组，代码如下：

```
<xs:attributeGroup name="att">
        <xs:attribute name="tel" type="xs:string" use="required"></xs:attribute>
        <xs:attribute name="qq" type="xs:integer"></xs:attribute>
        <xs:attribute name="msn" type="xs:string"></xs:attribute>
</xs:attributeGroup>
```

参数说明

❶ name="att"：表示属性组名。在 xs:attributeGroup 内部定义属性组的成员时，成员数量没有限制，声明完的属性组一般都在 xs:extenstion 内部使用。

❷ name="tel"：name="tel"、name="qq"等表示属性名称。

❸ type="xs:string"：type="xs:string"、type="xs:integer"等表示属性类型。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点，添加复杂类型 xs:cimplexType 以及 book 下的子元素，为 name、publisher、company、ISBN、price 和 url 子元素添加 xs:string 类型限定。在 company 下面添加一个元素 author，因为要为 author 声明属性，在 xs:simpleType 上扩展 xs:simpleType 和 xs:extension 元素，并在 xs:extension 上为 author 内容声明属性 xs:string。然后在 xs:schema 内部和 book 根节点平级处声明一个 xs:attributeGroup，设置其 name="att"，再在 xs:attributeGroup 内部声明 3 个属性，分别为 tel、qq 和 msn，形成一个属性组。在声明 author 处扩展 xs:extension 元素，并在其内部声明一个 xs:attributeGroup 引用，引用刚才声明的属性组。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">

<xs:element name="book">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="publisher" type="xs:string" />
            <xs:element name="company" type="xs:string" />
            <xs:element name="author">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attributeGroup ref="att"></xs:attributeGroup>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="ISBN" type="xs:string" />
            <xs:element name="price" type="xs:double" />
            <xs:element name="url" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:attributeGroup name="att">
    <xs:attribute name="tel" type="xs:string" use="required"></xs:attribute>
    <xs:attribute name="qq" type="xs:integer"></xs:attribute>
    <xs:attribute name="msn" type="xs:string"></xs:attribute>
</xs:attributeGroup>

</xs:schema>
```

（2）根据 XSD 的定义建立相应的 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com attributeGroup_demo.xsd">

<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author msn="wangxiaoke@mingrisoft.com" qq="200958602" tel="0431-84978981">王小科
</author>
<ISBN>9787302226628</ISBN>
<price>2.0</price>
<url>http://www.mingribook.com/bookinfo.php?id=227</url>
</book>
```

秘笈心法

心法领悟 354：XSD 文档中属性组的添加方法。

在本实例中，为 author 添加了一个属性组，其内部有 3 个属性 tel、qq 和 msn，这样做是为了更好地说明 xs:attributeGroup 的使用方式，但一般不建议这样定义 XML，最好的方式是把 tel、qq 和 msn 定义成 author 的子元素。

11.4　使用 XSD 的普通类型

实例 355　在 XSD 中对字符进行限制　　高级

光盘位置：光盘\MR\355　　趣味指数：★★★

实例说明

在 XML 中，可能使用不同的字符串表达不同的含义，XSD 为了更好地支持 XML，定义了很多字符串的类型，以满足不同需要。在本实例中，使用 XSD 定义了不同的字符类型描述一本图书，并通过建立 XML 来使用这些字符类型，如图 11.19 所示。

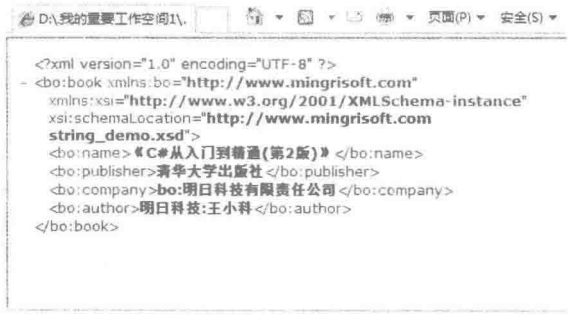


图 11.19　对字符进行限制

关键技术

使用 xs:token、xs:NCName 和 xs:QName 等字符类型可以对字符进行限制，代码如下：

```
<xs:element name="name" type="xs:token" />
<xs:element name="publisher" type="xs:NCName" />
<xs:element name="company" type="xs:QName" />
<xs:element name="author" type="xs:Name" />
```

参数说明

❶ type="xs:token"：表示定义的 name 的类型为 xs:token，这限制了 name 元素中不能包含换行符、回车或制表符、开头或结尾空格或者多个连续空格的字符串，即在使用 XML 时，图书的名称不应该含有这些符号。

❷ type="xs:NCName"：定义 publisher 的类型为 xs:NCName，说明 publisher 元素在使用时，必须是一个不包含冒号的 xs:NCName 类型。

❸ type="xs:QName"：定义 company 的类型为 xs:QName，说明 company 元素在 XML 中使用时，可以由前缀名和局部名组成，中间使用冒号分开。

❹ type="xs:Name"：定义 author 的类型为 xs:Name，说明 author 元素是一个只能包含数字、字母、下划线、冒号及其他名字字符的 Name 类型，而且 author 的第一个字符不可以是数字。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点，添加复杂类型 xs:cimplexType。在根元素 book 内部声名子元素 name、publisher、company 和 author，分别定义其数据类型为 xs:token、xs:NCName、xs:QName 和 xs:Name。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">
<xs:element name="book">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="name" type="xs:token" />
                        <xs:element name="publisher" type="xs:NCName" />
                        <xs:element name="company" type="xs:QName" />
                        <xs:element name="author" type="xs:Name" />
                </xs:sequence>
        </xs:complexType>
</xs:element>
</xs:schema>
```

（2）在 XML 中分别根据这几种不同的类型定义使用不同的元素。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<bo:book xmlns:bo="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com string_demo.xsd">
<bo:name>《C#从入门到精通(第 2 版)》</bo:name>
<bo:publisher>清华大学出版社</bo:publisher>
<bo:company>bo:明日科技有限责任公司</bo:company>
<bo:author>明日科技:王小科</bo:author>
</bo:book>
```

秘笈心法

心法领悟 355：XSD 文档中对字符串的定义。

本实例中使用的字符串定义符都是 xs:string 的子集，xs:string 表示所有的字符串都适用，字符串的内容可以是回车、空格、制表符等。xs:string 还有一个子集，即 xs:normalizedString，xs:normalizedString 是一个规范化子集，是表示 xs:string 中除了回车、换行和制表符以外的所有字符串的一个类型。

实例 356　在 XSD 中对数值进行限制

光盘位置：光盘\MR\356　　高级　　趣味指数：★★★★

实例说明

数值的类型在 XSD 中非常丰富，无论是数字正负、数字位数、数字的使用区间，还是数值的精确位置，都有很多类型供选择。在本实例中，使用 XSD 定义了不同的数字类型来描述一本图书的 ISBN、price 和 pageNum。XML 文档如图 11.20 所示。

图 11.20　对数值进行限制

关键技术

使用 type="xs:positiveInteger"、type="xs: nonNegativeInteger"和 type="xs:double"等可以对数值进行限制，代码如下：

```
<xs:element name="ISBN" type="xs:positiveInteger" />
<xs:element name="pageNum" type="xs:nonNegativeInteger" />
<xs:element name="price" type="xs:double" />
```

参数说明

❶ type="xs:positiveInteger"：限制了 ISBN 元素中的数据必须是正整数，数据内容不能包含 0。

❷ type="xs: nonNegativeInteger"：限制了 pageNum 元素中的数据是非负值的数，数据内容可以包含 0。

❸ type="xs:double"：可以设置精度比较高的数值数据类型。

设计过程

（1）建立 XSD 文档，声明文档根元素和命名空间。声明一个 book 根节点，声名复杂类型 xs:cimplexType。在根元素 book 声明子元素 ISBN、pageNum 和 price，分别为它们定义 xs: positiveInteger、xs: nonNegativeInteger 和 xs: double 类型。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.mingrisoft.com" xmlns="http://www.mingrisoft.com"
elementFormDefault="qualified">
<xs:element name="book">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="ISBN" type="xs:positiveInteger" />
                        <xs:element name="pageNum" type="xs:nonNegativeInteger" />
                        <xs:element name="price" type="xs:double" />
                </xs:sequence>
        </xs:complexType>
</xs:element>
</xs:schema>
```

（2）在 XML 中分别根据这几种不同的类型定义使用不同的元素内容。代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.mingrisoft.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mingrisoft.com integer_demo.xsd">
<ISBN>9787302226628</ISBN>
<pageNum>650</pageNum>
<price>69.80</price>
</book>
```

秘笈心法

心法领悟 356：XSD 中数值的数据类型表示形式。

本实例中使用的数值类型只是一小部分比较有代表性的，在 XSD 中还有很多数值类型值得学习，例如，xs: byte 表示具有正负值的 8 位数；xs:int 表示有正负值的 32 位数；xs:long 表示有正负值的 64 位数等。

第12章

解析 XML 文件

- 使用 SAX 读取 XML
- 使用 SAX 解析 XML
- 使用 DOM 读取 XML
- 使用 DOM 解析 XML
- 使用 DOM 操作 XML

12.1　使用 SAX 读取 XML

实例 357	从文件中读取 XML 光盘位置：光盘\MR\357	高级 趣味指数：★★★

实例说明

使用 SAX 解析 XML 文件的第一步，就是要使用 SAX 取读 XML 内容。SAX 读取 XML 的方法有很多种，如果 XML 被存储成文档，并且文档的内容不是特别大时，一般都通过读取 File 进行操作。读取的 XML 文件如图 12.1 所示。

图 12.1　XML 文件

关键技术

使用 SAXParserFactory 的 parse()方法可以从文件中读取 XML，语法如下：

```
public void parse(File f, DefaultHandler dh) throws SAXException, IOException
```

参数说明

❶ f：表示要读取的 XML 文件。

❷ dh：表示读取的 XML 文件处理机制。

设计过程

（1）创建一个 XML 文档。

（2）创建一个 Java 文件。在文件中创建 parseReadFile()方法，首先通过 SAXParseFactory 的 newInstance()方法创建一个 SAXParseFactory 的实例 factory，然后从 factory 可以获取到 SAXParser 的实例 parser。本实例中使用了 parser 的 parse()方法读取 XML 文件，代码如下：

```
public void parseReadFile(String pathname) {
    SAXParser parser;
    //生成 SAXParserFactory 对象
    SAXParserFactory factory = SAXParserFactory.newInstance();
    try {
        parser = factory.newSAXParser();
        //获取文件
        File file = new File(pathname);
        //解析文件
```

```
            parser.parse(file, new DefaultHandler());
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

（3）创建 main()方法，parser 可以利用该方法读取 XML 的文件内容。代码如下：

```
public static void main(String[] arg) {
    String pathname = " xmldemo/books.xml";
    new ParserFile().parseReadFile(pathname);
}
```

秘笈心法

心法领悟 357：parser()方法的参数。

parser 的 parse()方法使用了两个参数，一个参数是 File 的实例 file，其中 file 是 SAX 需要读取的 XML 文件；另一个参数是 DefaultHandler 的实例。DefaultHandler 是 SAX 的一个没有任何实现的类，作用是调用 parser 的 parse()方法读取 XML 文件时对文件进行解析，它继承了 EntityResolver、DTDHandler、ContentHandler 和 ErrorHandler 4 个接口，虽然使用了 parser 的 parse()方法，但 DefaultHandler 是一个没有实现的类，所以如果希望使用 SAX 解析，还要重新实现 DefaultHandler。

实例 358　从数据流中读取 XML

高级

光盘位置：光盘\MR\358　　趣味指数：★★★

实例说明

如果 XML 被存储成文件的格式，可以按文件的方式读取 XML；如果 XML 被保存到数据库中或者使用其他方式存储，可以使用数据流的方式读取。在本实例中，SAX 将借助 InputStream 读取 XML，为了更方便地操作 InputStream，可以在 main()方法中使用 File 得到 InputStream。读取的 XML 文件如图 12.1 所示。

关键技术

使用 SAXParserFactory 的 parse()方法可以从数据流中读取 XML，语法如下：

```
public void parse(InputStream is, DefaultHandler dh) throws SAXException, IOException
```

参数说明

❶ is：表示要读取的 XML 数据流。

❷ dh：表示读取的 XML 文件处理机制。

设计过程

（1）创建一个 XML 文档用于 SAX 进行读取。

（2）创建一个 Java 文件。在 Java 文件中创建 parseInputStream()方法，首先通过 SAXParseFactory 类的 newSAXParser()方法获取 SAXParser 的实例，然后获取 InputStream 和 DefaultHandler 的对象，再使用 SAXParser 的 parse()方法对数据流进行读取解析。代码如下：

```
public void parseInputStream(InputStream inputStream) {
    SAXParser parser;
    //生成 SAXParserFactory 实例
    SAXParserFactory factory = SAXParserFactory.newInstance();
    try {
        //获取 SAXParser 实例
        parser = factory.newSAXParser();
```

```
            parser.parse(inputStream, new DefaultHandler());
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

（3）创建 main()方法，先通过 File()和 FileInputStream()方法创建 InputStream 的对象，然后调用 parseInputStream()方法实现 SAX 通过数据流读取 XML 内容。代码如下：

```
    public static void main(String[] arg) {
        String pathname = "xmldemo/books.xml";
        InputStream inputStream = null;
        try {
            //获取数据流
            inputStream = new FileInputStream(new File(pathname));
        } catch (FileNotFoundException e1) {
            e1.printStackTrace();
        }
        //读取文件流
        new ParserInputStream().parseInputStream(inputStream);
        try {
            //关闭文件流
            inputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 358：使用流的方式读取文件后要关闭数据流。

使用 XML 读取 InputStream 时是以流的方式进行的，在使用完 InputStream 以后，要调用 close()方法关闭数据流，避免程序占用系统资源。

实例 359　从数据源中读取 XML

高级

光盘位置：光盘\MR\359

趣味指数：★★★

实例说明

本实例中，使用数据源 InputSource 对象作为 SAX 解析 XML 的输入，InputSource 允许使用字节数据流 InputStream、字符数据流 Reader 和 URL 资源 systemId，不过在使用完 InputStream 和 Reader 以后，要关闭数据源。读取的 XML 文件如图 12.1 所示。

关键技术

使用 SAXParserFactory 的 parse()方法可以从数据源中读取 XML，语法如下：

```
public void parse(InputSource is, DefaultHandler dh) throws SAXException, IOException
```

参数说明

❶ is：表示要读取的 XML 数据源。

❷ dh：表示读取的 XML 文件处理机制。

设计过程

（1）创建一个 XML 文档，用于 SAX 进行读取。

（2）创建一个 Java 文件。在 Java 文件中创建 parseInputSource()方法，首先获取 InputSource 和 DefaultHandler 的对象，然后创建 SAXParseFactory 实例，通过 SAXParseFactory 的 newSAXParser()方法获取 SAXParser 的实例，然后使用 SAXParser 的方法 parse()对数据源进行读取解析。代码如下：

```
public void parseInputSource(InputSource inputSource) {
    SAXParser parser;
    //生成 SAXParserFactory 实例
    SAXParserFactory factory = SAXParserFactory.newInstance();
    try {
        //获取 SAXParser 实例
        parser = factory.newSAXParser();
        parser.parse(inputSource, new DefaultHandler());
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
    } catch (SAXException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（3）创建 main()方法，通过 File()和 FileInputStream()方法创建 InputSource，然后把 InputSource 作为数据源传入到 parseInputSource()方法，实现 SAX 通过数据源读取 XML 内容。代码如下：

```
public static void main(String[] arg) {
    String pathname = "xmldemo/books.xml";
    InputStream inputStream = null;
    try {
        //获取文件流
        inputStream = new FileInputStream(new File(pathname));
    } catch (FileNotFoundException e1) {
        e1.printStackTrace();
    }
    //转换成数据源
    InputSource inputSource = new InputSource(inputStream);
    new ParserInputSource().parseInputSource(inputSource);
    try {
        //关闭数据源
        inputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 359：使用 InputSource 时传输方式的选择。

SAX 可以将 InputSource 对象作为读取 XML 的输入。使用 InputSource 时，发现有字符流 InputStream 可用，则解析器将直接读取该流，而忽略其他资源；如果没有发现字符流 InputStream，但却有字节流 Reader，则将使用该字节流；如果既没有字符流，也没有字节流，则解析器将尝试打开到由系统标识符 systemId 标识的资源的 URI 连接。

12.2 使用 SAX 解析 XML

实例 360 解析 XML 元素名称　　高级

光盘位置：光盘\MR\360　　趣味指数：★★★

实例说明

本实例讲解如何使用 SAX 解析 XML 元素名称，并把解析的 XML 元素名称打印到控制台，如图 12.2 所示。

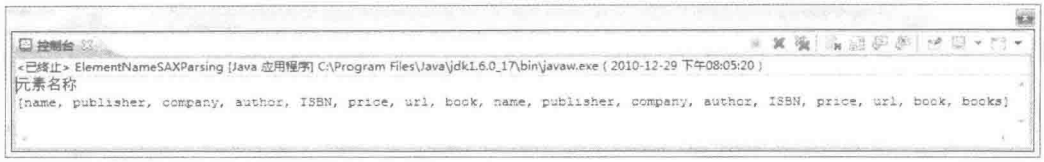


图 12.2　解析 XML 元素名称

关键技术

使用 SAX 解析 XML 时，按顺序以流的方式读取 XML，此处 ElementNameSAXParsing 继承了 DefaultHandler 类，并重写 endElement()方法。以后在每次调用 SAXParser 的 parse()方法时，向 parse()方法中传入 XML 文件和 ElementNameSAXParsing 的实例，在 SAX 读取 XML 到每个元素结束时，endElement()方法就会被执行。XML 中有几个元素，解析结束时 endElement()就会被调用几次。endElement()方法的语法如下：

```
public void endElement(String uri, String localName, String qName)  throws SAXException
```

参数说明

❶ uri：表示 XML 元素命名空间，在这里就是 http://www.mingrisoft.com。

❷ localName：表示 XML 元素的本地标识符，此处为 name、publisher、company、author 和 ISBN 等。

❸ qName：表示元素在 XML 文件中使用的名称，此处为 book:name、book:publisher、book:company、book:author 和 book:ISBN 等。

设计过程

（1）创建一个 XML 文档用于 SAX 进行解析。

（2）创建一个 Java 文件，定义 ElementNameSAXParsing 类，然后继承 DefaultHandler 类。在 Java 文件中创建 parseReadFile()方法用于解析 XML。代码如下：

```
public void parseReadFile(String pathname) {
    SAXParser parser;
    SAXParserFactory factory = SAXParserFactory.newInstance();
    try {
        //验证 XML 的格式是否正确
        factory.setValidating(true);
        //是否引入 XML 命名空间
        factory.setNamespaceAware(true);
        parser = factory.newSAXParser();
        File file = new File(pathname);
        parser.parse(file, this);
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
    } catch (SAXException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（3）在 ElementNameSAXParsing 类中重写 endElement()方法，在方法中获取 localName，并将其保存在 List 中。代码如下：

```
public void endElement(String uri, String localName, String qName) throws SAXException {
    list.add(localName);
}
```

（4）创建 main()方法，在方法中使用 ElementNameSAXParsing 解析 books.xml 文件，输出元素名称到控制台。代码如下：

```
public static void main(String[] arg) {
    String pathname = "xmldemo/books.xml";
    ElementNameSAXParsing elementSAXParsing = new ElementNameSAXParsing();
    elementSAXParsing.parseReadFile(pathname);
```

```
        System.out.println("元素名称");
        System.out.println(elementSAXParsing.getList());
    }
```

秘笈心法

心法领悟 360：setValidating()方法与 setNamespaceAware()方法的实现功能。

上面代码中 parseReadFile()方法中有如下两行代码：

```
//验证 XML 的格式是否正确
factory.setValidating(true);
//是否引入 XML 命名空间
factory.setNamespaceAware(true);
```

设置 setValidating 为 true 时，表示 SAX 在解析 XML 时会验证 XML 的格式是否正确，如果不正确，后台会报出相应的错误。

设置 setNamespaceAware 为 true 时，表示 SAX 在解析 XML 时会引入 XML 命名空间，只有引入命名空间，在解析 XML 时处理 endElement()等方法中，SAX 才会向 uri、localName 参数传值。

实例 361　解析 XML 元素名称和内容

光盘位置：光盘\MR\361　　高级　　趣味指数：★★★

实例说明

解析 XML 元素名称和内容的关键是当 SAX 解析 XML 时，把元素的名称和内容及时保存起来，同时 XML 中可能会有很多同名的元素，要把元素名称和内容对应起来。本实例在解析 XML 时，在 endElement()方法中把元素和内容拼成一个字符串，然后把字符串保存在 List 中。有了元素和内容的 List，想实现其他业务逻辑就不难了，在这里只是简单地输出，效果如图 12.3 所示。

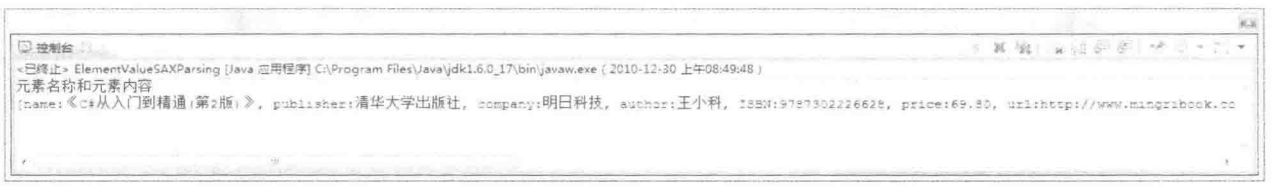


图 12.3　解析 XML 元素名称和内容

关键技术

（1）使用 ElementValueSAXParsing 类继承 DefaultHandler 类可以实现 characters()方法，通过实现该方法，可以获取 XML 元素的名称和内容，语法如下：

```
public void characters(char[] ch, int start, int length) throws SAXException
```

参数说明

❶ ch：表示 XML 的内容，其内容是整个 XML 文档。

❷ start：表示当前元素在整个 XML 文档中开始的字节数。

❸ length：表示当前元素本身的字节长度。

（2）SAX 解析 XML 时，ElementValueSAXParsing 继承了 DefaultHandler 类，并重写 endElement()方法。以后在每次调用 SAXParser 的 parse()方法时，向 parse()中传入 XML 文件和 ElementValueSAXParsing 的实例。SAX 读取每个元素的内容时，characters()方法就会被执行，读取 XML 元素结束时，endElement()方法就会被调用。代码如下：

```
package com.mingrisoft.SAX_demo;
```

```
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class ElementValueSAXParsing extends DefaultHandler {

private List<String> list = new ArrayList<String>();

private String value;

/**
 * 读取当前元素的内容，过滤制表符、空格符、回车符和换行符
 */
@Override
public void characters(char[] ch, int start, int length)
                throws SAXException {
        value = String.valueOf(ch, start, length);
        value = value.replace("\t", "");
        value = value.replace(" ", "");
        value = value.replace("\n", "");
        value = value.replace("\r", "");
}

/**
 * 读取元素结束，把元素名称和元素内容保存在 Map 中
 */
@Override
public void endElement(String uri, String localName, String qName)
                throws SAXException {
        list.add(localName + ":" + value);
}

public List<String> getList() {
        return this.list;
}

/**
 * 通过文件读取 XML
 *
 * @param pathname
 *                  文件路径
 */
public void parseReadFile(String pathname) {
        SAXParser parser;
        SAXParserFactory factory = SAXParserFactory.newInstance();
        try {
                factory.setValidating(true);
                factory.setNamespaceAware(true);
                parser = factory.newSAXParser();
                File file = new File(pathname);
                parser.parse(file, this);
        } catch (ParserConfigurationException e) {
                e.printStackTrace();
        } catch (SAXException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
```

```
}

public static void main(String[] arg) {

        String pathname = "xmldemo/books.xml";
        ElementValueSAXParsing elementSAXParsing = new ElementValueSAXParsing();
        elementSAXParsing.parseReadFile(pathname);
        System.out.println("元素名称和元素内容");
        System.out.println(elementSAXParsing.getList());
}
}
```

设计过程

（1）创建一个 XML 文档用于 SAX 进行解析。

（2）创建 ElementNameSAXParsing 类，然后继承 DefaultHandler 接口。在 Java 文件中创建 parseReadFile()方法用于解析 XML。

（3）实现 characters()方法，获取当前元素的内容，把它保存在临时变量 value 中。代码如下：

```
public void characters(char[] ch, int start, int length) throws SAXException {
        //读取当前元素的内容，过滤制表符、空格符、回车符和换行符
        value = String.valueOf(ch, start, length);
        value = value.replace("\t", "");
        value = value.replace(" ", "");
        value = value.replace("\n", "");
        value = value.replace("\r", "");
}
```

（4）实现 endElement()方法，在方法中获取 localName 和临时变量 value 值，并把元素的名称和内容一起保存在 List 中。代码如下：

```
public void endElement(String uri, String localName, String qName)
                throws SAXException {
        //读取元素结束，把元素名称和内容保存在 list 中
        list.add(localName + ":" + value);
}
```

（5）创建 main()方法，在方法中使用 ElementValueSAXParsing 解析 books.xml 文件，输出元素名称到控制台。

秘笈心法

心法领悟 361：SAX 读取文件时对字符的操作。

在 XML 文档中，book 是一个父元素，其内部包含了几个子元素，但是它本身并没有元素内容，但是在读取时，SAX 会把内部一些不可见的字符读出来，例如空格符、回车符等，可以通过字符串的替换把这些字符替换掉。或者使用 ignorableWhitespace()方法，此方法获取的元素内容都是没有空格符的，但是要想触发 ignorableWhitespace()方法，必须让 XML 文档引用 DTD，使用 XML Schema 限定 XML 时，ignorableWhitespace()方法是不会被触发的。

实例 362 解析 XML 元素属性和属性值

高级

光盘位置：光盘\MR\362

趣味指数：★★★★

实例说明

XML 的属性包含属性名称和属性值，在 XML 中，每个元素都可能含有属性，属性是针对元素而言的。本实例的 XML 文档中有两本图书，每本图书都有价格，即每个 book 元素中都包含一个 price 元素，但是它们的内容可能又是不一样的。在 price 元素中包含 unit 和 unitType 两个属性，每个 price 元素都可以有同样的属性，但是不同的 price 的属性值可能是不一样的。本实例讲解如何获取 XML 元素的属性和属性值，如图 12.4 所示。

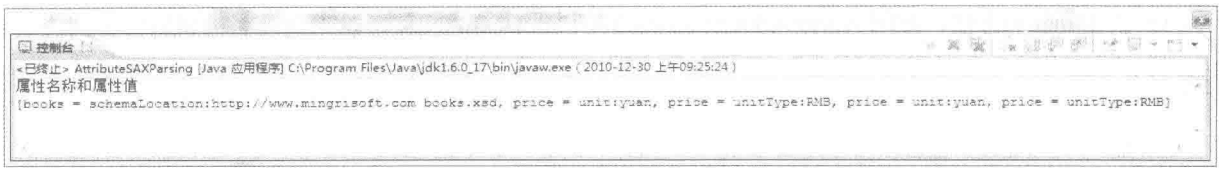


图 12.4　解析 XML 属性

关键技术

SAX 每次开始读取 XML 元素时，startElement()方法都会被执行，使用 AttributeSAXParsing 类重写 DefaultHandler 的 startElement()方法可以获取元素属性和属性值，语法如下：

```
public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException
```

参数说明

❶ uri：表示 XML 元素命名空间，此处为 http://www.mingrisoft.com。

❷ localName：表示 XML 元素的本地标识符，此处为 name、publisher、company、author 和 ISBN 等。

❸ qName：表示元素在 XML 文件中使用的名称，此处为 book:name、book:publisher、book:company、book:author 和 book:ISBN 等。

❹ attributes：表示当前元素的属性集合。

设计过程

（1）创建一个 XML 文档，用于 SAX 进行解析。

（2）创建 AttributeSAXParsing 类继承 DefaultHandler 类，重写 startElement()方法读取属性和属性值，并把它们合并为一个字符串保存在 List 中。一个元素可能有多个属性，所以使用 attributes 中的 getLength()方法获取当前元素属性的个数，在这个属性集合中，使用 getLocalName()方法能获取当前元素第几个属性的名称，getValue()方法中参数是几就表示当前元素的第几个属性值。代码如下：

```
public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
    //读取属性名称和属性值保存在 List 中
    for (int i = 0; i < attributes.getLength(); i++) {
        attribute.add(localName + " = " + attributes.getLocalName(i) + ":"+ attributes.getValue(i));
    }
}
```

（3）创建 parseReadFile()方法把 AttributeSAXParsing 实例传入解析器，实现 XML 的解析。代码如下：

```
public void parseReadFile(String pathname) {
    SAXParser parser;
    //获取 SAXParserFactory 实例
    SAXParserFactory factory = SAXParserFactory.newInstance();
    try {
        factory.setValidating(true);
        factory.setNamespaceAware(true);
        //获取 SAXParser 实例
        parser = factory.newSAXParser();
        //获取 XML 文件
        File file = new File(pathname);
        parser.parse(file, this);
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
    } catch (SAXException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（4）创建 main()方法，使用 AttributeSAXParsing 解析 books.xml 文件，输出属性名称和属性值到控制台。

秘笈心法

心法领悟 362：attributes 参数的含义。

本实例中，使用 startElement()方法获取了属性的名称和值，attributes 是 startElement()方法的一个参数，通过它可以获取当前元素的所有属性集合和属性相关信息，如使用 getIndex()方法获取某个属性的索引值，使用 getType()方法获取某个属性的类型，使用 getQName()方法获取某个属性的 XML 元素名称等。

实例 363　使用 VO 解析 XML 元素

光盘位置：光盘\MR\363　　高级　　趣味指数：★★★★

实例说明

XML 元素使用 DTD 或者 XML Schema 限定以后，结构一般是不会变的，可以根据 XML 的结构定义一个 VO，解析 XML 时，借助 VO 保存元素内容，使程序操作起来更方便，结构更简单。本实例讲解如何使用 VO 解析 XML 元素，如图 12.5 所示。

图 12.5　使用 VO 解析 XML 元素

关键技术

SAX 解析 XML 时，建立 BookElementSAXParsing 继承 DefaultHandler 类，并在 BookElementSAXParsing 中重写 startElement()方法。在 startElement()方法内，每次 SAX 检测到 book 元素都重新创建一个 BookElement 实例，示例代码如下：

```
//每次读取 book 元素时重新初始化 bookElement
if (bookElement == null || "book".equals(localName)) {
    bookElement = new BookElement();
}
```

在 endElement()方法中，SAX 检测到相应的元素就把元素内容置到 BookElement 的实例中，当检测到 book 元素时，就把 BookElement 实例保存在 List 中，这样，在 XML 中有几个 book 元素，在 List 中就会有几个 BookElement 实例。示例代码如下：

```
//检测当前元素，把元素内容保存在相应的实例属性中
if ("name".equals(localName)) {
    bookElement.setName(elementValue);
} else if ("publisher".equals(localName)) {
    bookElement.setPublisher(elementValue);
} else if ("company".equals(localName)) {
    bookElement.setCompany(elementValue);
```

```
} else if ("author".equals(localName)) {
        bookElement.setAuthor(elementValue);
} else if ("ISBN".equals(localName)) {
        bookElement.setIsbn(elementValue);
} else if ("price".equals(localName)) {
        bookElement.setPrice(Double.valueOf(elementValue));
} else if ("url".equals(localName)) {
        try {
                bookElement.setUrl(new URL(elementValue));
        } catch (MalformedURLException e) {
                e.printStackTrace();
        }
} else if ("book".equals(localName)) {
        bookList.add(bookElement);
}
```

设计过程

（1）创建一个 XML 文档，用于 SAX 进行解析。

（2）根据 XML 文档结构创建 BookElement 类。

（3）创建 BookElementSAXParsing 类继承 DefaultHandler 类，然后实现 startElement()方法，在方法中根据元素名称创建 VO 实体，代码如下：

```
public void startElement(String uri, String localName, String qName,
                Attributes attributes) throws SAXException {
        //每次读取 book 元素时重新初始化 bookElement
        if (bookElement == null || "book".equals(localName)) {
                bookElement = new BookElement();
        }
}
```

（4）创建 characters()方法，在方法中读取、解析 XML 元素内容。代码如下：

```
public void characters(char[] ch, int start, int length) throws SAXException {
        //解析元素内容
        elementValue = elementValue.valueOf(ch, start, length);
        elementValue = elementValue.replace("\t", "");
        elementValue = elementValue.replace(" ", "");
        elementValue = elementValue.replace("\n", "");
        elementValue = elementValue.replace("\r", "");
}
```

（5）创建 endElement()方法，把元素内容保存在 BookElement 实例中，再把 BookElement 实例保存在 List 中。

（6）创建 parseReadFile()方法，把 BookElementSAXParsing 实例传入解析器，实现 XML 的解析。

（7）创建 main()方法，在方法中使用 BookElementSAXParsing 解析 books.xml 文件，输出 BookElement 的 List 到控制台。

秘笈心法

心法领悟 363：BookElement 在控制台中的显示格式。

在控制台输出 BookElement 时，显示的是 BookElement 中保存的内容，并且通过一定的格式显示出来，这是因为重写了 toString()方法。在 toString()方法的内部，把 BookElement 类的变量名称和值一一对应起来，并把它们拼成一个字符串返回。

实例 364　使用 VO 解析 XML 元素和属性

高级

光盘位置：光盘\MR\364

趣味指数：★★★★

实例说明

在前面的实例中，解析 XML 元素时使用 VO 保存元素内容，使程序操作起来更方便。在本实例中，使用

VO 实现对 XML 元素和属性的解析，如图 12.6 所示。

图 12.6　使用 VO 解析 XML 元素和属性

关键技术

SAX 解析 XML 时，创建 BookSAXParsing 类并继承 DefaultHandler，在 BookSAXParsing 类中重写 startElement()方法，并使用该方法读取 price 元素的属性，保存在 bookPrice 中。代码如下：

```
//读取元素、属性，把元素、属性名称保存在 List 中
if (bookPrice == null || "price".equals(localName)) {
    bookPrice = new BookPrice();
    //读取属性名称和值
    Map<String, String> attributeMap = new HashMap<String, String>();
    for (int i = 0; i < attributes.getLength(); i++) {
        //把属性名称和值保存在 Map 中
        attributeMap.put(attributes.getLocalName(i),attributes.getValue(i));
    }
    if (!attributeMap.isEmpty()) {
        bookPrice.setAttributeMap(attributeMap);
    }
}
```

设计过程

（1）创建一个 XML 文档用于 SAX 进行解析。

（2）根据 XML 文档结构创建 Book 类和 BookPrice 类。

（3）创建 BookSAXParsing 类继承 DefaultHandler，然后实现 startElement()方法，在 startElement()方法中根据元素名称创建 VO 实体，同时解析 XML 元素属性，把属性名称和属性值保存在 VO 中，代码如下：

```
public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
    if (book == null || "book".equals(localName)) {
        book = new Book();
    }
    //读取元素、属性，把元素、属性名称保存在 List 中
    if (bookPrice == null || "price".equals(localName)) {
        bookPrice = new BookPrice();
        //读取属性名称和值
        Map<String, String> attributeMap = new HashMap<String, String>();
        for (int i = 0; i < attributes.getLength(); i++) {
            //把属性名称和值保存在 Map 中
            attributeMap.put(attributes.getLocalName(i),attributes.getValue(i));
        }
        if (!attributeMap.isEmpty()) {
```

```
                bookPrice.setAttributeMap(attributeMap);
            }
        }
    }
```

（4）创建 characters()方法，在方法中读取、解析 XML 元素内容。代码如下：

```
public void characters(char[] ch, int start, int length) throws SAXException {
    //解析元素内容
    elementValue = elementValue.valueOf(ch, start, length);
    elementValue = elementValue.replace("\t", "");
    elementValue = elementValue.replace(" ", "");
    elementValue = elementValue.replace("\n", "");
    elementValue = elementValue.replace("\r", "");
}
```

（5）创建 endElement()方法，把元素内容保存在 Book 实例中，再把 Book 实例保存在 List 中。代码如下：

```
public void endElement(String uri, String localName, String qName) throws SAXException {
    if ("name".equals(localName)) {
        book.setName(elementValue);
    } else if ("publisher".equals(localName)) {
        book.setPublisher(elementValue);
    } else if ("company".equals(localName)) {
        book.setCompany(elementValue);
    } else if ("author".equals(localName)) {
        book.setAuthor(elementValue);
    } else if ("ISBN".equals(localName)) {
        book.setIsbn(elementValue);
    } else if ("price".equals(localName)) {
        bookPrice.setValue(Double.valueOf(elementValue));
        book.setPrice(bookPrice);
    } else if ("url".equals(localName)) {
        try {
            book.setUrl(new URL(elementValue));
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
    } else if ("book".equals(localName)) {
        bookList.add(book);
    }
}
```

（6）创建 parseReadFile()方法，把 BookSAXParsing 实例传入解析器，实现 XML 的解析。

（7）创建 main()方法，在方法中使用 BookSAXParsing 解析 books.xml 文件，输出 Book 的 List 到控制台。

秘笈心法

心法领悟 364：使用 SAX 解析文件时获取参数的方法。

本实例使用 SAX 解析 XML 时，主要使用 startElement()、endElement()和 characters()方法获取元素、属性，另外还使用了 startDocument()方法，该方法在解析 XML 时调用了一次，所以可以把初始化变量的工作放在此处完成。相对应的还有 endDocument()方法，该方法在每个 XML 文件解析完时调用一次，读者可以根据它们的特性在实际情况下使用。

实例 365　使用 SAX 验证 DTD

高级

光盘位置：光盘\MR\365

趣味指数：★★★★

实例说明

使用 SAX 也可以验证 XML 是否符合 DTD 的规范。首先 XML 要引用 DTD 文件，然后通过 SAX 对当前的 XML 进行解析，解析时需要设置 factory.setValidating(true)。本实例要解析 XML 和 DTD，为了看到验证效果，将 XML 元素中的 price 元素改成 prices，解析 XML 时就会出现如图 12.7 所示的错误信息。

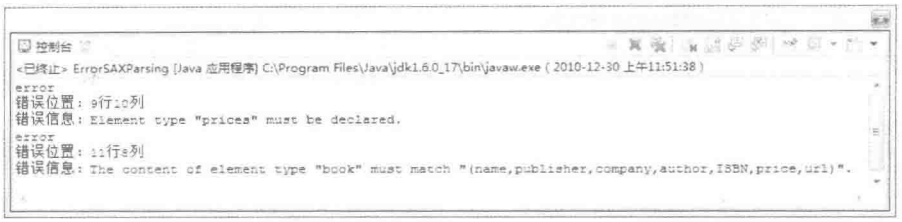


图 12.7　验证 XML 内容

关键技术

SAX 解析 XML 时，建立 ErrorSAXParsing 继承 DefaultHandler 类，并在 ErrorSAXParsing 上重写 warning()、error()和 fatalError()方法。3 个错误处理表示 3 个不同程度的错误，验证 XML 是否符合 DTD 规范需要覆盖 error()方法。error()方法的语法如下：

```
public void error(SAXParseException exception) throws SAXException
```

参数说明

exception：解析 XML 时发生的异常处理。

设计过程

（1）创建一个 DTD 文档，用来定义 XML 文档的格式，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book (name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST price unit CDATA "RMB" >
```

（2）根据 XSD 文档的结构定义 XML 文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "books.dtd">
<book>
<name>《C#从入门到精通(第 2 版)》</name>
<publisher>清华大学出版社</publisher>
<company>明日科技</company>
<author>王小科</author>
<ISBN>9787302226628</ISBN>
<price>69.80</price>
<url><![CDATA[http://www.mingribook.com/bookinfo.php?id=227&sid=4]]></url>
</book>
```

（3）创建 ErrorSAXParsing 类继承 DefaultHandler，实现 error()、warning()和 fatalError()方法。在 3 个方法中向控制台输出错误位置和错误信息。代码如下：

```
public void error(SAXParseException exception) throws SAXException {
        System.out.println("error");
        System.out.println("错误位置：" + exception.getLineNumber() + "行" + exception.getColumnNumber() + "列");
        System.out.println("错误信息：" + exception.getMessage());
}
```

（4）创建 parseReadFile()方法，把 ErrorSAXParsing 实例传入解析器，实现 XML 的解析。

（5）创建 main()方法，在方法中使用 ErrorSAXParsing 解析 books.xml 文件，输出错误信息到控制台。

秘笈心法

心法领悟 365：使用 SAX 解析文件时获取错误信息的方法。

使用 SAXParseException 类的 getLineNumber()、getColumnNumber()和 getMessage()方法可以获取错误发生

的行号、列号以及具体的错误内容。语法如下：

```
//行号
public int getLineNumber ()
//列号
public int getColumnNumber ()
//错误内容
public String getMessage ()
```

12.3　使用 DOM 读取 XML

实例 366	从文件中读取 XML	中级
	光盘位置：光盘\MR\366	趣味指数：★★★

实例说明

使用 DOM 解析 XML，首先要读取 XML 文件，一般使用 File 类对文件进行操作，把文件的访问路径通过 File 的构造方法传递进去得到一个 File 的对象，然后对该 File 对象进行读取。要读取的 XML 文件如图 12.1 所示。

关键技术

使用DOM解析XML文件前，先通过DocumentBuilderFactory的newInstance()方法创建一个DocumentBuilderFactory的实例 documentBuilderFactory，通过该实例可以获取 DocumentBuilder 的实例 dombuilder，然后使用 dombuilder 的 parse()方法解析 File 的对象。代码如下：

```
//创建 DocumentBuilderFactory 对象
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
//创建 DocumentBuilder 对象
DocumentBuilder dombuilder = documentBuilderFactory.newDocumentBuilder();
//获取文件
File file = new File(path);
dombuilder.parse(file);
```

设计过程

（1）创建一个 XML 文档用于 DOM 解析。

（2）创建 ParserFile 类，在类中创建 parseReadFile()方法，使用 DocumentBuilderFactory、DocumentBuilder 和 File 类读取 XML 文件，代码如下：

```
public void parseReadFile(String path) throws ParserConfigurationException,SAXException, IOException {
    //创建 DocumentBuilderFactory 对象
    DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
    //创建 DocumentBuilder 对象
    DocumentBuilder dombuilder = documentBuilderFactory.newDocumentBuilder();
    //获取文件
    File file = new File(path);
    dombuilder.parse(file);
}
```

（3）创建 main()方法，在方法中调用 ParserFile 类的 parseReadFile()方法读取 XML 文件。代码如下：

```
public static void main(String[] arg) {
    ParserFile parserFile = new ParserFile();
    String path = "xmldemo/books.xml";
    try {
        //读取 XML 文件
        parserFile.parseReadFile(path);
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
    } catch (SAXException e) {
        e.printStackTrace();
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 366：使用 parse()方法时产生异常的处理方法。

使用 parse()方法时，需要抛出 3 个异常处理：ParserConfigurationException、SAXException 和 IOException。异常处理有两种方式，可以使用 throws 直接抛出，留给后面需要调用 parseReadFile()的方法进行处理；也可以直接在本方法中使用 try/catch 把错误包容在本方法中。代码如下：

```
try {
    dombuilder = documentBuilderFactory.newDocumentBuilder();
    dombuilder.parse(file);
} catch (ParserConfigurationException e) {
    e.printStackTrace();
} catch (SAXException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

实例 367 从数据流中读取 XML

高级

光盘位置：光盘\MR\367

趣味指数：★★★

实例说明

DOM 也可以从数据流中读取 XML，当 XML 被保存到数据库中或者使用其他方式存储时，可以使用数据流的方式读取。在本实例中，将以 InputStream 的形式读取 XML 文档。要读取的 XML 文件如图 12.1 所示。

关键技术

使用 DocumentBuilderFactory 的 parse()方法可以从数据流中读取 XML 文档，语法如下：

```
public Document parse(InputStream is)  throws SAXException, IOException
```

参数说明

is：表示 XML 文件的数据流。

设计过程

（1）创建一个 XML 文档用于 DOM 解析。

（2）创建 ParserInputStream 类，在类中创建 parseInputStream()方法，用于从数据流中读取 XML。使用 DocumentBuilderFactory 类的 parse()方法读取数据流，代码如下：

```
public void parseInputStream(String path) throws ParserConfigurationException, SAXException, IOException {
    //创建 DocumentBuilderFactory 实例
    DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
    //创建 DocumentBuilder 实例
    DocumentBuilder dombuilder = documentBuilderFactory.newDocumentBuilder();
    //创建 InputStream 实例
    InputStream is = new FileInputStream(path);
    dombuilder.parse(is);
    //关闭数据流
    is.close();
}
```

（3）创建 main()方法，在方法中调用 ParserInputStream 类的 parseInputStream()方法，以数据流的方式读取 XML 的文件，代码如下：

```
public static void main(String[] arg) {
        ParserInputStream parserFile = new ParserInputStream();
        String path = "xmldemo/books.xml";
        try {
                //读取 XML 文件
                parserFile.parseInputStream(path);
        } catch (ParserConfigurationException e) {
                e.printStackTrace();
        } catch (SAXException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

秘笈心法

心法领悟 367：实现 InputStream 接口的实体类可以获取 InputStream。

DOM 解析 XML 文件可以读取数据流 InputStream，实例中使用 FileInputStream 实例化 InputStream 只是其中的一个途径，因为 FileInputSteam 实现了 InputStream 接口。任何实现 InputStream 接口的实体类都可以获取 InputStream。

实例 368　从数据源中读取 XML

光盘位置：光盘\MR\368　　高级　　趣味指数：★★★

实例说明

在实例中使用 InputSource 作为 DOM 解析 XML 的输入源，DOM 在解析 XML 时，有多种输入方式。InputSource 允许使用字节数据流 InputStream、字符数据流 Reader 和系统标示的资源 systemId，是一个非常灵活的输入接口。本实例使用 systemId 标示一个 XML 的数据源。要读取的 XML 文件如图 12.1 所示。

关键技术

使用 DocumentBuilder 的 parse()方法可以从数据源中读取 XML 文档，语法如下：

```
public abstract Document parse(InputSource is) throws SAXException, IOException;
```

参数说明

is：表示 XML 文件的数据源。

设计过程

（1）创建一个 XML 文档用于 DOM 解析。

（2）创建 ParserInputSource 类，在类中创建 parseInputSource()方法，用于从数据源中读取 XML。使用 DocumentBuilder 类的 parse()方法读取数据源，代码如下：

```
public void parseInputSource(InputSource is) throws ParserConfigurationException, SAXException, IOException {
        //创建 DocumentBuilderFactory 实例
        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
        //创建 DocumentBuilder 实例
        DocumentBuilder dombuilder = documentBuilderFactory.newDocumentBuilder();
        //从数据源解析 XML
        dombuilder.parse(is);
}
```

（3）创建 main()方法，在方法中调用 ParserInputSource 类的 parseInputSource()方法，从数据源中读取 XML 文件。代码如下：

```
public static void main(String[] arg) {
        ParserInputSource parserFile = new ParserInputSource();
```

```
        String systemId = "xmldemo/books.xml";
        InputSource is = new InputSource(systemId);
        try {
            //读取数据源
            parserFile.parseInputSource(is);
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 368：DOM 解析 XML 时编码类型的设置。

DOM 可以将 InputSource 对象作为读取 XML 的输入。在使用 InputSource 时，可以使用 setEncoding()方法设置 XML 的编码类型，如 is.setEncoding("UTF-8")指定了 XML 的编码格式为 UTF-8。

12.4 使用 DOM 解析 XML

实例 369	解析 XML 元素名称	中级
	光盘位置：光盘\MR\369	趣味指数：★★★

实例说明

解析 XML 时，首先要获取元素名称，DOM 把元素信息保存在 Node 中，通过 Node 类的 getNodeName()方法可以获取元素名称。本实例演示如何使用 DOM 解析 XML 元素名称，解析的 XML 元素如图 12.8 所示。

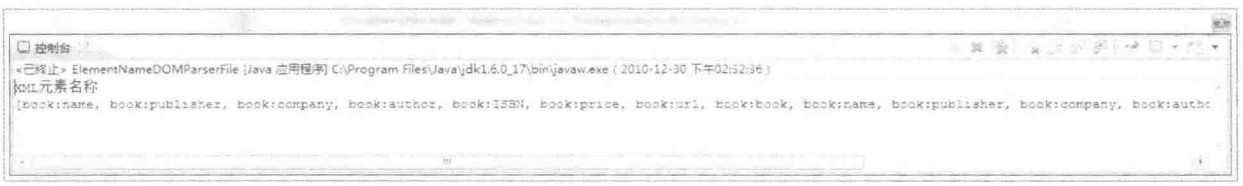


图 12.8 XML 文件

关键技术

（1）使用 Node 类的 getChildNodes()方法可以获取 XML 文件中的子元素列表，语法如下：

```
public NodeList getChildNodes()
```

（2）使用 Node 类的 hasChildNodes()方法可以判断当前 XML 元素是否有子节点，语法如下：

```
public boolean hasChildNodes()
```

设计过程

（1）创建一个 XML 文档用于 DOM 解析。

（2）创建 ElementNameDOMParserFile 类的 parseReadFile()方法，用于读取 XML 文件，同时返回一个 Document，代码如下：

```
public Document parseReadFile(String path) throws ParserConfigurationException, SAXException, IOException {
    //创建 DocumentBuilderFactory 实例
    DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
    //创建 DocumentBuilder 实例
```

```
        DocumentBuilder dombuilder = documentBuilderFactory.newDocumentBuilder();
        File file = new File(path);
        //解析 XML 文件
        return dombuilder.parse(file);
    }
```

（3）创建 getElementName()方法，该方法有一个参数 Node，通过 Node 的 hasChildNodes()和 getChildNodes()方法遍历 Node 的内容。当 Node 存在子节点时，则使用 getNodeName()方法得到当前节点的名称，也就是元素的名称，同时保存在 List 中，然后递归调用 getElementName()方法获取下级子节点的元素名称。因为 XML 文档是一个树形结构，只有这样才能得到所有 XML 元素的名称。代码如下：

```
    public List<String> getElementName(Node parentNode) {
        //是否有子节点
        if (parentNode.hasChildNodes()) {
            //获取子节点
            NodeList nodeList = parentNode.getChildNodes();
            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);
                //如果有子节点则递归调用
                if (node.hasChildNodes()) {
                    getElementName(node);
                    elementList.add((node.getNodeName()));
                }
            }
        }
        return elementList;
    }
```

（4）创建 main()方法，在内部调用 parseReadFile()和 getElementName()方法，最后把结果输出到控制台。

秘笈心法

心法领悟 369：获取 XML 元素名称的方法。

在 main()方法中，根据 parseReadFile()方法得到一个 Document，因为 Document 继承了 Node 接口，而 getElementName()方法的参数是 Node，所以可以把 Document 对象传入到 getElementName()方法中，由 getElementName()方法负责获取 XML 元素名称。

实例 370　解析 XML 元素名称和内容　　中级

光盘位置：光盘\MR\370　　趣味指数：★★★

实例说明

XML 元素的名称和内容都可以使用 DOM 解析出来。本实例演示如何使用 DOM 解析 XML 元素名称和内容，然后把它们输出到控制台，如图 12.9 所示。

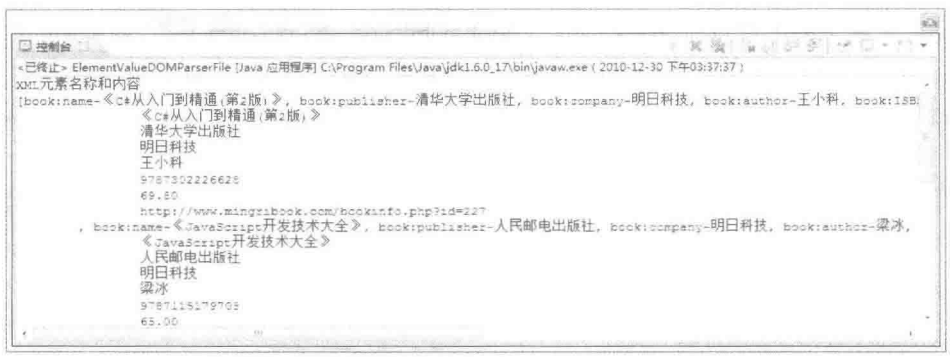


图 12.9　DOM 解析 XML 元素名称和内容

关键技术

（1）使用 Node 类的 getNodeName()方法可以获取元素节点的名称，语法如下：

```
public String getNodeName()
```

（2）使用 Node 类的 getTextContent()方法可以获取元素内容，语法如下：

```
public String getTextContent() throws DOMException;
```

设计过程

（1）创建一个 XML 文档用于 DOM 解析。

（2）创建 ElementValueDOMParserFile 类，在类中创建 parseReadFile()方法用于读取 XML 文件，同时返回一个 Document。

（3）建立一个 List 变量，用于保存 XML 文件的元素和内容，代码如下：

```
private List<String> elementList = new ArrayList<String>();
```

（4）创建一个 getElementName()方法，在方法内部读取 XML 元素和内容，然后拼成一个字符串，保存到 List 的变量中。代码如下：

```
public void getElementName(Node parentNode) {
    if (parentNode.hasChildNodes()) {
        NodeList nodeList = parentNode.getChildNodes();
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            //判断是否有子节点
            if (node.hasChildNodes()) {
                getElementName(node);
                //拼成字符串保存在 List 中
                elementList.add(node.getNodeName() + "-"+ node.getTextContent());
            }
        }
    }
}
```

（5）创建 getElementList()方法，返回保存在 List 变量中的结果，代码如下：

```
public List<String> getElementList() {
    return this.elementList;
}
```

（6）创建 main()方法，在内部调用 getElementList()方法，最后把结果输出到控制台。

秘笈心法

心法领悟 370：通过 DOM 获取父节点的内容是子节点集合。

如果通过 DOM 获取一个父节点的内容，那么该父节点的内容就是其所有子节点内容的集合，例如实例中第一个 book:book 元素下的子节点有 book:name、book:publisher、book:company、book:author、book:ISBN、book:price 和 book:url，这些元素的内容分别是《C#从入门到精通(第 2 版)》、清华大学出版社、明日科技、王小科、9787302226628、69.80 和 http://www.mingribook.com/bookinfo.php?id=227，那么 book:book 元素的内容就是它们的集合，如下所示：

```
《C#从入门到精通(第 2 版)》
    清华大学出版社
    明日科技
    王小科
    9787302226628
    69.80
    http://www.mingribook.com/bookinfo.php?id=227
```

实例 371　解析 XML 元素属性和属性值

光盘位置：光盘\MR\371　　中级　　趣味指数：★★★

实例说明

DOM 解析完 XML 以后，每个元素都会转换成一个 Node 对象，Node 内不但存储着元素内容，同时还保存着元素的属性。使用 getAttributes()方法可以获取当前元素的所有属性。在 XML 中，不同父节点的同名子节点的属性有可能相同，例如本实例中的两个 book:book 元素，它们都有 book:price 子节点，两个子节点的属性都是 unit="yuan"和 unitType="RMB"，使用 DOM 解析 XML 属性，如图 12.10 所示。

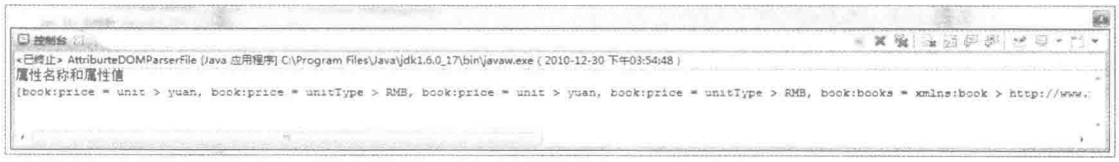


图 12.10　DOM 解析 XML 属性

关键技术

（1）使用 Node 类的 getAttributes()方法可以获取属性列表，语法如下：

```
public NamedNodeMap getAttributes()
```

（2）使用 NamedNodeMap 类的 item()方法可以获取指定的节点，语法如下：

```
public Node item(int index)
```

参数说明

index：表示当前节点的索引。

设计过程

（1）创建一个 XML 文档用于 DOM 解析。

（2）创建 AttriburteDOMParserFile 类，在类中创建 parseReadFile()方法，用于读取 XML 文件，同时返回一个 Document。

（3）建立一个 List 变量，用于保存 XML 文件的元素和内容，代码见实例 370。

（4）创建一个 getElementName()方法，在方法内部读取 XML 文件元素、属性和属性值，然后拼成一个字符串，保存到 List 的变量中。代码如下：

```
public void getElementName(Node parentNode) {
    if (parentNode.hasChildNodes()) {
        NodeList nodeList = parentNode.getChildNodes();
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.hasChildNodes()) {
                getElementName(node);
                //获取属性列表
                NamedNodeMap namedNodeMap = node.getAttributes();
                for (int j = 0; j < namedNodeMap.getLength(); j++) {
                    Node node2 = namedNodeMap.item(j);
                    //获取属性值
                    elementList.add(node.getNodeName() + "= "+ node2.getNodeName() + " > "+ node2.getNodeValue());
                }
            }
        }
    }
}
```

（5）创建 getElementList()方法，返回保存在 List 变量中的结果，代码如下：

```
public List<String> getElementList() {
    return this.elementList;
}
```

（6）创建 main()方法，在内部调用 AttriburteDOMParserFile 的 parseReadFile()、getElementName()和 getElementList()方法，然后把结果输出到控制台。

秘笈心法

心法领悟 371：获取属性值的方法。

在实例中，根据 Node 的 getNodeValue()方法获取属性的值，如果只是针对属性而言，使用 getTextContent()方法也可以获取属性值，且 getTextContent()和 getNodeValue()方法的内容是一样的。

实例 372 使用 VO 解析 XML 元素

高级

光盘位置：光盘\MR\372

趣味指数：★★★★

实例说明

使用 DOM 解析 XML 可以根据 XML 元素建立一个 VO 类，把 XML 的子元素都看作是 VO 类的一个属性，元素的内容存储在 VO 中，如图 12.11 所示。

图 12.11 使用 VO 解析 XML 元素

关键技术

使用 Node 类的 getElementsByTagName()方法可以获取指定的元素内容，语法如下：

```
public NodeList getElementsByTagName(String name)
```

参数说明

name：表示 XML 元素中的名称。

设计过程

（1）创建一个 XML 文档用于 DOM 解析。

（2）创建 BookElement 类用于解析 XML 时的 VO。

（3）创建 BookElementDOMParsing 类，并且创建 parseReadFile()方法用于读取 XML 文件。

（4）创建 getBook()方法，把读取的 XML 文件元素和内容保存在 BookElement 中，代码如下：

```
public List<BookElement> getBook(Element element) {
    NodeList list = element.getElementsByTagName("book:book");
    for (int i = 0; i < list.getLength(); i++) {
        bookElement = new BookElement();
        NodeList name = element.getElementsByTagName("book:name");
        NodeList publisher = element.getElementsByTagName("book:publisher");
```

```
                NodeList company = element.getElementsByTagName("book:company");
                NodeList author = element.getElementsByTagName("book:author");
                NodeList ISBN = element.getElementsByTagName("book:ISBN");
                NodeList price = element.getElementsByTagName("book:price");
                NodeList url = element.getElementsByTagName("book:url");
                bookElement.setName(name.item(i).getTextContent());
                bookElement.setPublisher(publisher.item(i).getTextContent());
                bookElement.setCompany(company.item(i).getTextContent());
                bookElement.setAuthor(author.item(i).getTextContent());
                bookElement.setIsbn(ISBN.item(i).getTextContent());
                bookElement.setPrice(new Double(price.item(i).getTextContent()));
                try {
                        bookElement.setUrl(new URL(url.item(i).getTextContent()));
                } catch (MalformedURLException e) {
                        e.printStackTrace();
                } catch (DOMException e) {
                        e.printStackTrace();
                }
                bookList.add(bookElement);
        }
        return bookList;
}
```

（5）建立一个 List 变量，把遍历的每一个 BookElement 保存在 List 中，代码如下：

```
private List<BookElement> bookList = new ArrayList<BookElement>();
```

（6）返回一个 List<BookElement>，在 main()方法中把刚才保存的结果输出到控制台，代码如下：

```
public static void main(String[] arg) {
        String pathname = "xmldemo/books.xml";
        BookElementDOMParsing elementSAXParsing = new BookElementDOMParsing();
        Document document = null;
        document = elementSAXParsing.parseReadFile(pathname);
        List<BookElement> bookElements = elementSAXParsing.getBook(document.getDocumentElement());
        System.out.println("自定义 JAVA 类封装元素名称和内容");
        System.out.println(bookElements);
}
```

秘笈心法

心法领悟 372：遍历 NodeList 时的方法。

遍历 NodeList 时，把元素内容保存在 BookElement 中，所以在每次遍历 NodeList 前，都要重新创建一个 BookElement 对象，避免保存在 List 中的 BookElement 数据不是想要的结果。如果没有在 NodeList 遍历的内部创建 BookElement 对象，最后 List 中的数据都将是一个 XML 元素节点的内容。

实例 373　使用 VO 解析 XML 元素和属性

高级

光盘位置：光盘\MR\373　　趣味指数：★★★★

实例说明

本实例需要使用 VO 解析 XML 元素和属性，可以根据 XML 元素和属性的特点建立一个 VO 类，使每一个元素都对应 VO 的一个属性，并把它们输出到控制台，效果如图 12.12 所示。

关键技术

获取 XML 元素的属性比较繁琐，特别是元素有多个属性时。先根据 getAttributes()方法获取属性列表，然后把属性内容保存在 Map 中，等所有属性解析完成后再把内容保存在 VO 中，代码如下：

```
//获取价格
BookPrice bookPrice = new BookPrice();
Map<String, String> attributeMap = new HashMap<String, String>();
NamedNodeMap namedNodeMap = price.item(i).getAttributes();
for (int j = 0; j < namedNodeMap.getLength(); j++) {
```

```
        //获取价格属性
        Node node = namedNodeMap.item(j);
        attributeMap.put(node.getNodeName(), node.getNodeValue());
    }
    bookPrice.setAttributeMap(attributeMap);
    bookPrice.setValue(new Double(price.item(i).getTextContent()));
    book.setPrice(bookPrice);
```

图 12.12　使用 VO 解析 XML 元素和属性

设计过程

（1）创建一个 XML 文档用于 DOM 解析。

（2）分别创建 Book 类和 BookPrice 类用于保存 XML 元素和属性。

（3）创建 BookDOMParsing 类，在类中创建 parseReadFile()方法，用于读取 XML 文件，代码如下：

```
public Document parseReadFile(String path) {
    //创建 DocumentBuilderFactory 对象
    DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dombuilder = null;
    try {
        //创建 DocumentBuilder 对象
        dombuilder = documentBuilderFactory.newDocumentBuilder();
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
    }
    File file = new File(path);
    try {
        return dombuilder.parse(file);
    } catch (SAXException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}
```

（4）创建 getBook()方法，把读取的 XML 文件元素和内容保存在 Book 中。对于 book:price 元素要特殊处理，先把 book:price 元素属性取出来保存在 BookPrice 类的 attributeMap 中，然后把值也保存在 BookPrice 中，再把 BookPrice 对象存放在 Book 中，最后把遍历的每个 Book 保存在一个 List 变量中，代码如下：

```
public List<Book> getBook(Element element) {
    NodeList list = element.getElementsByTagName("book:book");
    for (int i = 0; i < list.getLength(); i++) {
        book = new Book();
        NodeList name = element.getElementsByTagName("book:name");
        NodeList publisher = element.getElementsByTagName("book:publisher");
        NodeList company = element.getElementsByTagName("book:company");
        NodeList author = element.getElementsByTagName("book:author");
        NodeList ISBN = element.getElementsByTagName("book:ISBN");
```

```
            NodeList price = element.getElementsByTagName("book:price");
            NodeList url = element.getElementsByTagName("book:url");
            book.setName(name.item(i).getTextContent());
            book.setPublisher(publisher.item(i).getTextContent());
            book.setCompany(company.item(i).getTextContent());
            book.setAuthor(author.item(i).getTextContent());
            book.setIsbn(ISBN.item(i).getTextContent());
            //获取价格
            BookPrice bookPrice = new BookPrice();
            Map<String, String> attributeMap = new HashMap<String, String>();
            NamedNodeMap namedNodeMap = price.item(i).getAttributes();
            for (int j = 0; j < namedNodeMap.getLength(); j++) {
                //获取价格属性
                Node node = namedNodeMap.item(j);
                attributeMap.put(node.getNodeName(), node.getNodeValue());
            }
            bookPrice.setAttributeMap(attributeMap);
            bookPrice.setValue(new Double(price.item(i).getTextContent()));
            book.setPrice(bookPrice);
            //获取 URL
            try {
                book.setUrl(new URL(url.item(i).getTextContent()));
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (DOMException e) {
                e.printStackTrace();
            }
            bookList.add(book);
        }
        return bookList;
    }
```

（5）返回一个 List<Book>，在 main()方法中把刚才保存的结果输出到控制台，代码如下：

```
public static void main(String[] arg) {

    String pathname = "xmldemo/books.xml";
    BookDOMParsing elementSAXParsing = new BookDOMParsing();
    Document document = null;
    document = elementSAXParsing.parseReadFile(pathname);
    List<Book> bookElements = elementSAXParsing.getBook(document
            .getDocumentElement());
    System.out.println("自定义 JAVA 类封装元素名称和内容");
    System.out.println(bookElements);

}
```

秘笈心法

心法领悟 373：复杂的 XML 文档结构的处理方法。

根据不同的需要，XML 的文档结构也不一样，在使用过程中可能很简单也可能很复杂。在遇到复杂的 XML 文档结构时，可以根据文档的结构特点把 XML 元素拆分成最小单位，设计符合需要的 VO 类，从而为解析和操作 XML 带来方便。

12.5　使用 DOM 操作 XML

实例 374　创建简单的 XML 文件　　中级

光盘位置：光盘\MR\374　　趣味指数：★★★

实例说明

XML 文件很多时候是需要动态生成的，本实例通过 DOM 创建了一个简单的 XML 文件，运行效果如图 12.13

所示。

图 12.13　一个简单的 XML 文件

关键技术

（1）使用 DocumentBuilderFactory 类的 newDocumentBuilder()方法可以创建一个 DocumentBuilder 实例，语法如下：

```
public abstract DocumentBuilder newDocumentBuilder() throws ParserConfigurationException
```

（2）使用 DocumentBuilder 类的 newDocument()方法可以创建一个空白的 XML 文件，语法如下：

```
public abstract Document newDocument()
```

（3）使用 DocumentBuilder 类的 createElement()方法可以创建一个 XML 元素，语法如下：

```
public Element createElement(String tagName) throws DOMException
```

参数说明

tagName：表示要创建的元素名称。

设计过程

（1）创建一个 Java 文件，在文件中创建 bulid()方法，在方法内部构建一个符合 DOM 规范的 Document 对象，代码如下：

```
public Document bulid() {
        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dombuilder = null;
        try {
                dombuilder = documentBuilderFactory.newDocumentBuilder();
        } catch (ParserConfigurationException e) {
                e.printStackTrace();
        }
        Document document = dombuilder.newDocument();
        Element book = document.createElement("book:book");
        book.setAttribute("xmlns:book", "http://www.mingrisoft.com");
        document.appendChild(book);
        return document;
}
```

（2）创建 writeFile()方法，获取 Node 对象和 XML 文件的生成路径，根据 Node 对象和 XML 的生成路径生成 XML 文档，代码如下：

```
public void writeFile(Node node, String url) {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        DOMSource domSource = new DOMSource(node);
        StreamResult streamResult = new StreamResult(new File(url));
        try {
                Transformer transformer = transformerFactory.newTransformer();
                transformer.transform(domSource, streamResult);
        } catch (TransformerConfigurationException e) {
                e.printStackTrace();
        } catch (TransformerException e) {
                e.printStackTrace();
        }

}
```

（3）创建 main()方法，设置 XML 生成的路径，调用 bulid()和 writeFile()方法完成 XML 文档的创建。

秘笈心法

心法领悟 374：定义 XML 的命名空间。

在 bulid()方法中创建的 Element 对象 book 是 XML 的根元素，其名称是 book:book，该元素冒号前部分表示命名空间，冒号后部分才是真正的元素名称，所以在创建 XML 时要使用 setAttribute()方法为 XML 指定命名空间。

实例 375　创建基本的 XML 文件　中级

光盘位置：光盘\MR\375　　趣味指数：★★★

实例说明

本实例通过 DOM 创建了一个基本的 XML 文件，在 XML 中含有元素、元素的级别和元素的内容。通过浏览器查看 XML 文档的结构，如图 12.14 所示。

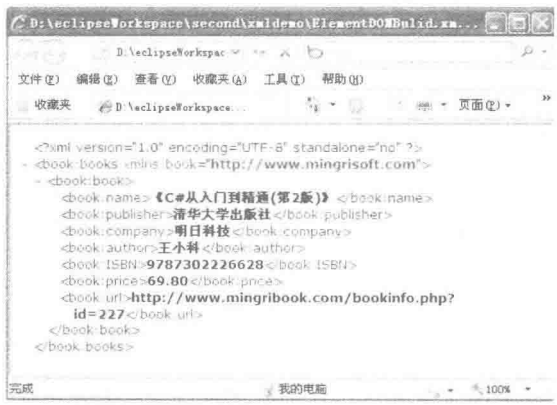


图 12.14　通过 DOM 创建的基本的 XML

关键技术

通过 DocumentBuilderFactory 和 DocumentBuilder 类可以创建一个新的 Document，再使用 Document 的 createElement()方法创建相应的元素 book:books、book:book、book:name、book:publisher、book:company、book:author、book:ISBN、book:price 和 book:url，然后根据 XML 元素的对应关系，使用 appendChild()方法添加 XML 节点，最后返回 document。代码如下：

```
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dombuilder = null;
try {
        dombuilder = documentBuilderFactory.newDocumentBuilder();
} catch (ParserConfigurationException e) {
        e.printStackTrace();
}

Document document = dombuilder.newDocument();
Element books = document.createElement("book:books");
books.setAttribute("xmlns:book", "http://www.mingrisoft.com");

Element book = document.createElement("book:book");
Element name = document.createElement("book:name");
……
name.setTextContent("《C#从入门到精通(第 2 版)》");
……
```

```
book.appendChild(name);
……

document.appendChild(books);
```

设计过程

（1）创建一个 Java 文件 ElementDOMBulid。

（2）在 ElementDOMBulid 中创建 bulid()方法，在方法内部构建一个符合 DOM 规范的 Document 对象。先使用 createElement()方法生成元素，再为生成的元素设置相应的内容或者属性，如 name.setTextContent("《C#从入门到精通(第 2 版)》");，然后根据元素之间的父子关系添加元素节点。代码如下：

```
public Document bulid() {
        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dombuilder = null;
        try {
                dombuilder = documentBuilderFactory.newDocumentBuilder();
        } catch (ParserConfigurationException e) {
                e.printStackTrace();
        }
        Document document = dombuilder.newDocument();
        Element books = document.createElement("book:books");
        books.setAttribute("xmlns:book", "http://www.mingrisoft.com");
        Element book = document.createElement("book:book");
        Element name = document.createElement("book:name");
        Element publisher = document.createElement("book:publisher");
        Element company = document.createElement("book:company");
        Element author = document.createElement("book:author");
        Element isbn = document.createElement("book:ISBN");
        Element price = document.createElement("book:price");
        Element url = document.createElement("book:url");
        name.setTextContent("《C#从入门到精通(第 2 版)》");
        publisher.setTextContent("清华大学出版社");
        company.setTextContent("明日科技");
        author.setTextContent("王小科");
        isbn.setTextContent("9787302226628");
        price.setTextContent("69.80");
        url.setTextContent("http://www.mingribook.com/bookinfo.php?id=227");
        book.appendChild(name);
        book.appendChild(publisher);
        book.appendChild(company);
        book.appendChild(author);
        book.appendChild(isbn);
        book.appendChild(price);
        book.appendChild(url);
        books.appendChild(book);
        document.appendChild(books);
        return document;
}
```

（3）创建 writeFile()方法，获取 Node 对象和 XML 文件的生成路径，生成 XML 文档。代码如下：

```
public void writeFile(Node node, String url) {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        DOMSource domSource = new DOMSource(node);
        StreamResult streamResult = new StreamResult(new File(url));
        try {
                Transformer transformer = transformerFactory.newTransformer();
                transformer.transform(domSource, streamResult);
        } catch (TransformerConfigurationException e) {
                e.printStackTrace();
        } catch (TransformerException e) {
                e.printStackTrace();
        }
}
```

（4）创建 main()方法，设置 XML 生成的路径，调用 bulid()和 writeFile()方法完成 XML 文档的创建。

秘笈心法

心法领悟 375：创建 XML 文档时的层次结构设置。

在创建 XML 文档时，可以把 document 看作是文档的第一级节点，把 books 元素看作第二级节点，book 元素是第三级节点，name 等元素是第四级节点。所以在设定 XML 节点关系时，使用这样的层次结构：document.appendChild(books)、books.appendChild(book)、book.appendChild(name)。

实例 376　使用 VO 创建 XML 文件

光盘位置：光盘\MR\376　　高级　　趣味指数：★★★★

实例说明

为了更方便地创建 XML 文件，可以根据要创建的 XML 文件的格式，建立符合 XML 结构特性的 VO，让 VO 中的每个属性都对应 XML 的一个元素，同时为复杂的元素创建单独的 VO。创建的 XML 文档的运行效果如图 12.15 所示。

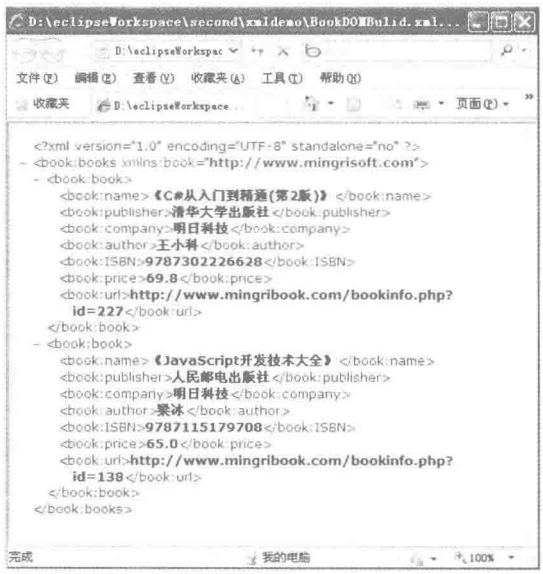


图 12.15　使用 VO 生成 XML 文档

关键技术

使用 TransformerFactory 类的 newInstance()方法可以创建 TransformerFactory 的一个实例，用于创建 XML 文件时产生一个转换器，语法如下：

```
public static TransformerFactory newInstance() throws TransformerFactoryConfigurationError
```

设计过程

（1）创建 initData()方法，在方法内部生成一组数据，数据通过 Book、BookPrice 等 VO 存储在 List 中。代码如下：

```
protected List<Book> initData() {
    List<Book> bookList = new ArrayList<Book>();
    Book book;
    BookPrice bookPrice;
```

```
        Map<String, String> attributeMap;
        //第一本书
        book = new Book();
        bookPrice = new BookPrice();
        book.setName("《C#从入门到精通(第 2 版)》");
        book.setPublisher("清华大学出版社");
        book.setCompany("明日科技");
        book.setAuthor("王小科");
        book.setIsbn("9787302226628");
        attributeMap = new HashMap<String, String>();
        attributeMap.put("unit", "yuan");
        attributeMap.put("unitType", "RMB");
        bookPrice.setAttributeMap(attributeMap);
        bookPrice.setValue(Double.parseDouble("69.80"));
        book.setPrice(bookPrice);
        try {
            book.setUrl(new URL("http://www.mingribook.com/bookinfo.php?id=227"));
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        bookList.add(book);
        //第二本书
        book = new Book();
        bookPrice = new BookPrice();
        book.setName("《JavaScript 开发技术大全》");
        book.setPublisher("人民邮电出版社");
        book.setCompany("明日科技");
        book.setAuthor("梁冰");
        book.setIsbn("9787115179708");

        attributeMap = new HashMap<String, String>();
        attributeMap.put("unit", "yuan");
        attributeMap.put("unitType", "RMB");
        bookPrice.setAttributeMap(attributeMap);
        bookPrice.setValue(Double.parseDouble("65.00"));
        book.setPrice(bookPrice);
        try {
            book.setUrl(new URL("http://www.mingribook.com/bookinfo.php?id=138"));
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        bookList.add(book);
        return bookList;
    }
```

（2）创建 bulid()方法，在方法内调用 initData()方法产生的数据，解析初始化数据的同时，可以转换成 DOM 的规范格式存储在 Document 中。代码如下：

```
    public Document bulid() {
        //创建 DocumentBuilderFactory 对象
        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dombuilder = null;
        Document document = null;
        try {
            //创建 DocumentBuilder 对象
            dombuilder = documentBuilderFactory.newDocumentBuilder();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        }
        List<Book> bookList =initData();
        System.out.println(bookList);
        if (bookList != null) {
            document = dombuilder.newDocument();
            Element books = document.createElement("book:books");
            books.setAttribute("xmlns:book", "http://www.mingrisoft.com");
            for (Iterator iterator = bookList.iterator(); iterator.hasNext();) {
                Book bookEntity = (Book) iterator.next();
```

```
                Element book = document.createElement("book:book");
                Element name = document.createElement("book:name");
                Element publisher = document.createElement("book:publisher");
                Element company = document.createElement("book:company");
                Element author = document.createElement("book:author");
                Element isbn = document.createElement("book:ISBN");
                Element price = document.createElement("book:price");
                Element url = document.createElement("book:url");
                //设置 XML 内容
                name.setTextContent(bookEntity.getName());
                publisher.setTextContent(bookEntity.getPublisher());
                company.setTextContent(bookEntity.getCompany());
                author.setTextContent(bookEntity.getAuthor());
                isbn.setTextContent(bookEntity.getIsbn());
                price.setTextContent(bookEntity.getPrice().getValue().toString());
                url.setTextContent(bookEntity.getUrl().toString());
                //添加 XML 节点
                book.appendChild(name);
                book.appendChild(publisher);
                book.appendChild(company);
                book.appendChild(author);
                book.appendChild(isbn);
                book.appendChild(price);
                book.appendChild(url);
                books.appendChild(book);
            }
            document.appendChild(books);
        }
        return document;
    }
```

（3）使用 writeFile()方法接收 bulid()方法产生的 Document，然后将其转换成 XML 文件，根据参数传递的 XML 文件路径写在指定的位置上。

（4）在 main()方法中设置 XML 生成的路径，调用 bulid()和 writeFile()方法完成 XML 文档的创建。

秘笈心法

心法领悟 376：initData 方法对数据的操作。

initData()方法用于产生一批规则的数据，然后把这批数据封装以后交给 DOM 处理，DOM 会创建一个 XML 文件存储这些数据。实际上，在整个过程中 initData 模拟的是一个数据源。initData()方法产生的数据可以是从数据库中读取的，也可以是从其他文件中读取的，尽管数据的来源多种多样，但是都可以通过这种方式转换成 XML 文件。

实例 377　使用 DOM 添加 XML 元素

高级

光盘位置：光盘\MR\377

趣味指数：★★★

实例说明

在操作 XML 时，会经常需要为 XML 添加元素。在本实例中，原始 XML 文件只有两本图书内容，如图 12.16 所示。下面通过 DOM 为 XML 添加一个 book:book 元素，产生新的 XML 文件，运行效果如图 12.17 所示。

通过图 12.16 和图 12.17 可以很清楚地看到，最下面的 XML 元素是新增的 book:book 元素，该元素含有元素名称、元素内容和属性的相关信息。

关键技术

使用 DOMSource 类的构造方法，可以创建一个 DOM 的数据源，生成 XML 文件时，从该数据源获取数据，语法如下：

public DOMSource(Node n)

参数说明

n：表示 DOM 的 XML 节点。

图 12.16　原始 XML 文件

图 12.17　添加元素以后的 XML 文件

设计过程

（1）创建 BookDOMAdd 类，然后创建 parseReadFile()方法解析 XML 文件，文件解析完以后存储在 DOM 的 document 对象中，代码如下：

```
private void parseReadFile() {
        //创建 DocumentBuilderFactory 对象
        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dombuilder = null;
        try {
                //创建 DocumentBuilder 对象
                dombuilder = documentBuilderFactory.newDocumentBuilder();
        } catch (ParserConfigurationException e) {
                e.printStackTrace();
        }
        File file = new File(path);
        try {
                //解析 XML
                document = dombuilder.parse(file);
        } catch (SAXException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

（2）新建 addBook()方法，创建新的 book:book、book:name、book:publisher、book:company、book:author、book:isbn、book:price、book:url 等元素，然后为 book 添加子元素 name、publisher、company、author、isbn、price 和 url，再把 book 作为一个子元素添加到 document 中，代码如下：

```
public void addBook() {
        parseReadFile();
        Element book = document.createElement("book:book");
        Element name = document.createElement("book:name");
        Element publisher = document.createElement("book:publisher");
        Element company = document.createElement("book:company");
        Element author = document.createElement("book:author");
        Element isbn = document.createElement("book:ISBN");
        Element price = document.createElement("book:price");
        Element url = document.createElement("book:url");

        name.setTextContent("《Java 全能速查宝典》");
        publisher.setTextContent("人民邮电出版社");
        company.setTextContent("明日科技");
        author.setTextContent("梁冰");
        isbn.setTextContent("9787115213794");
        price.setTextContent("59.00");
        price.setAttribute("unit", "yuan");
        price.setAttribute("unitType", "RMB");
        url.setTextContent("http://www.mingribook.com/bookinfo.php?id=191");
        book.appendChild(name);
        book.appendChild(publisher);
        book.appendChild(company);
        book.appendChild(author);
        book.appendChild(isbn);
        book.appendChild(price);
        book.appendChild(url);
        document.getDocumentElement().appendChild(book);
}
```

（3）创建 writeFile()方法，把 document 中存储的内容保存在 XML 文件中。

秘笈心法

心法领悟 377：XML 元素中多个属性值的设置方法。

添加 XML 元素时，使用 setTextContent()方法为指定的 XML 元素添加内容，如果 XML 元素含有属性，可以使用 setAttribute()方法添加属性名称和属性值，同时 setAttribute()方法还支持属性的叠加，如果一个元素有两个属性，可以使用两次 setAttribute()方法，第二次添加的属性不会覆盖第一次添加的属性。

实例 378 使用 DOM 修改 XML 元素

光盘位置：光盘\MR\378

高级

趣味指数：★★★★

实例说明

在操作 XML 时，如果 XML 元素的内容存储错误或者由于其他原因需要更改时，也可以使用 DOM 对 XML 进行操作。例如，有一个原始的 XML，如图 12.16 所示，现需要将作者是“王小科”的图书名称由“《C#从入门到精通（第 2 版）》”改成“《C#从入门到精通（第 1 版）》”，这就需要修改 book:name 中的内容。修改完以后的内容运行效果如图 12.18 所示。

图 12.18 修改 book:name 以后的 XML

关键技术

使用 StreamResult 类的构造函数可以创建一个 StreamResult 实例，通过该实例能把一个 XML 文件转换成数据流的结果集，语法如下：

```
public StreamResult(File f)
```

参数说明

f：表示 XML 文件的实例。

设计过程

（1）创建 BookDOMUpdate 类，在类中创建 parseReadFile()方法解析 XML 文件，文件解析完以后存储在 DOM 的 document 对象中，代码如下：

```
private void parseReadFile() {
    DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dombuilder = null;
    try {
        dombuilder = documentBuilderFactory.newDocumentBuilder();
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
    }
    File file = new File(path);
    try {
        document = dombuilder.parse(file);
    } catch (SAXException e) {
```

```
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

（2）新建 updateBook()方法，在方法中修改 XML 元素。使用 DOM 修改 XML 元素的内容前，先把 XML 文件读取出来保存在 document 中，根据条件查找 document 内部需要修改的 XML 元素，然后再修改元素内容。根据本实例的 XML 的结构，需要逐级锁定要找的元素，即先找到 book:book 级别的元素，再找出 book:book 内的子元素名称，才能做修改操作。

book:book 元素在 document 中可能存在多个，所以查找结果是一个 NodeList 列表，使用 for 语句对 list 列表循环搜索，获取的每个 Node 就是一个 book:book 元素。

因为 book:book 元素还有子节点，所以使用 bookNode.getChildNodes 获取 book:book 的子元素列表，再对子元素的列表 list1 循环搜索，获取的 node1 就是 book:book 的子元素。然后根据指定的条件 node1.getNodeName().equals(nodeName)和 node1.getTextContent().equals(text)进行判断，如果循环取得的元素名称和元素内容和 nodeName、text 中的一致，就设置变量 flag=true。如果 falg=true，就锁定了要找的 book:book 元素，可以针对当前的 book:book 进行修改操作。在修改之前，还要对 list1 循环搜索，若 node1.getNodeName().equals(replNodeName)判断成立，表示当前的 node1 就是想要修改的元素，然后使用 setTextContent(repltext)设置修改以后的元素内容。具体代码如下：

```
/**
 * 根据 book 子节点指定的条件更新元素的内容
 * @param nodeName 指定条件元素名称
 * @param text 指定条件元素内容
 * @param replNodeName 指定更新的节点
 * @param repltext 设置更新的内容
 */
public void updateBook(String nodeName, String text, String replNodeName,String repltext) {
    parseReadFile();
    NodeList list = document.getElementsByTagName("book:book");
    for (int i = 0; i < list.getLength(); i++) {
        boolean flag = false;
        Node bookNode = list.item(i);
        NodeList list1 = bookNode.getChildNodes();
        //查询出需要修改的 book 节点
        for (int j = 0; j < list1.getLength(); j++) {
            Node node1 = list1.item(j);
            if (node1.getNodeName().equals(nodeName)
                    && node1.getTextContent().equals(text)) {
                flag = true;
            }
        }
        //查询到需要的 book 节点，设置指定的元素内容
        if (flag) {
            for (int j = 0; j < list1.getLength(); j++) {
                Node node1 = list1.item(j);
                //查询出需要修改的节点
                if (node1.getNodeName().equals(replNodeName)) {
                    node1.setTextContent(repltext);
                }
            }
            flag = false;
        }

    }
}
```

（3）创建 writeFile()方法，把修改后的 document 内容保存在 XML 文件中。

秘笈心法

心法领悟 378：修改 XML 元素中临时变量的用法。

在修改 XML 元素时，使用了一个临时变量 flag，用来判断需要修改的元素是否是当前 book:book 元素的子元素，所以其有效范围在 list 的循环语句中，在 list 中的每一个对象都可以看作是一个 book:book 元素。临时变量 flag 的初始状态是 false，在 list1 的 for 循环语句中，如果检测到与条件相符的元素，则改变 flag 状态为 true，只有当 flag 为 true 时，表示搜索到了需要修改的元素，再对其内容进行修改。当修改完指定的元素内容时，一定要把 flag 设为初始状态 false，否则在后面的循环步骤中会改变所有的元素内容。

实例 379　使用 DOM 删除 XML 元素

光盘位置：光盘\MR\379　　高级　　趣味指数：★★★★

实例说明

XML 文件的内容是不断变化的，有时需要把 XML 中不需要的元素删除，使用 DOM 删除元素时，也同样要先把 XML 文件读取到 DOM 对象中，删除以后再把 DOM 中的内容保存到 XML 文件。现有原始的 XML 文件如图 12.16 所示，本实例要删除 book:name，即《C#从入门到精通（第 2 版）》的 book:book 元素，删除后 XML 的运行效果如图 12.19 所示。

图 12.19　删除元素后的 XML

关键技术

使用 Node 类的 removeChild()方法可以删除不需要的节点，语法如下：

```
public Node removeChild(Node oldChild) throws DOMException
```

参数说明

oldChild：表示希望删除的节点。

设计过程

（1）创建 BookDOMDelete 类，然后创建 parseReadFile()方法解析 XML 文件，文件解析后存储在 DOM 的 document 对象中，代码如下：

```
private void parseReadFile() {
    //创建 DocumentBuilderFactory 对象
    DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dombuilder = null;
    try {
        //创建 DocumentBuilder 对象
        dombuilder = documentBuilderFactory.newDocumentBuilder();
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
    }
    File file = new File(path);
```

```
        try {
            //解析 XML
            document = dombuilder.parse(file);
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

（2）在 deleteBook()方法内部找到要删除的 XML 元素，并存放在临时变量中，然后使用 removeChild()方法删除临时变量中的元素值。deleteBook()方法有 nodeName 和 text 两个参数，根据这两个参数传递的值可以删除指定的 book:book 元素。在 deleteBook()方法内部使用 getElementsByTagName()方法获取 NodeList，循环 list 变量可以获取每个 book:book 元素，将其保存在 node1 中，再获取 nod1.getChildNodes 循环 list1 的内容，遍历出 book:book 下面的所有子元素。

在 list1 的循环中根据元素名称和节点锁定该元素的父节点 book:book 元素，把 book:book 的对象赋值给 node，然后等循环结束使用根元素的 removeChild 方法删除 node。代码如下：

```
//根据 book 子节点的条件删除 book 节点
public void deleteBook(String nodeName, String text) {
    parseReadFile();

    Element books = document.getDocumentElement();
    NodeList list = document.getElementsByTagName("book:book");
    Node node = null;
    for (int i = 0; i < list.getLength(); i++) {
        NodeList list1 = list.item(i).getChildNodes();
        for (int j = 0; j < list1.getLength(); j++) {
            Node node1 = list1.item(j);
            //根据条件查询要删除的节点
            if (node1.getNodeName().equals(nodeName)
                    && node1.getTextContent().equals(text)) {
                node = list.item(i);
            }
        }
    }
    books.removeChild(node);
}
```

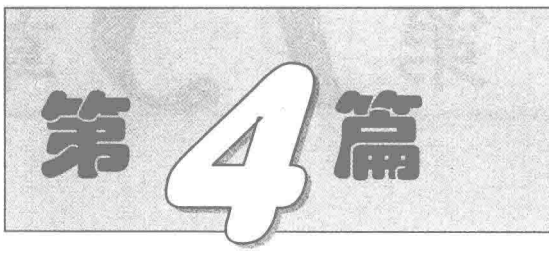
（3）删除元素后，document 中的内容发生了变化，需要使用 writeFile()方法把 document 存储在 XML 中。

秘笈心法

心法领悟 379：removeChild 方法的具体实现。

使用 removeChild()方法只能删除当前元素的下一级元素，例如，当前的实例 books 是根元素，其 removeChild()方法只对 book:book 级别的元素有效，如果希望删除 book:book 的子节点，如 book:name 等，则不能使用该方法，否则会报出如下错误：

```
org.w3c.dom.DOMException: NOT_FOUND_ERR: An attempt is made to reference a node in a context where it does not exist.
```



操作 PDF 篇

第13章

操作 PDF 文档

- 文档和文档属性
- 初始化操作
- 字体与中文处理
- 块、短语、段落、章节和区域
- 读取 PDF 文档

13.1　文档和文档属性

实例 380	创建 PDF 文档 光盘位置：光盘\MR\380	初级 趣味指数：★★★★

实例说明

本实例演示了如何通过 Java 应用程序创建 PDF 文档。运行程序，将在本地系统的 C 盘根目录下创建一个名为“创建第一个 PDF 文档.pdf”的文件，双击该文件可以看到所创建 PDF 文档的显示效果，如图 13.1 所示。

图 13.1　创建的 PDF 文档

关键技术

本实例主要是通过 Document 类创建文档对象，然后使用 PdfWriter 类的 getInstance()方法将该文档对象与磁盘文件关联，再使用 Document 类的 open()方法打开文档对象，然后通过 Document 类的 add()方法向文档中添加段落文本，最后使用 Document 类的 close()方法关闭文档对象，完成 PDF 文档的创建。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 CreateDocumentDemo 类，该类中只有一个 main()主方法，在应用程序启动后，将执行该方法的代码，完成文档的创建。CreateDocumentDemo 类的完整代码如下：

```
public class CreateDocumentDemo {
    public static void main(String[] args) {
        try {
            Document document = new Document();                                          //创建文档对象
            PdfWriter.getInstance(document, new FileOutputStream("c:\\创建第一个 PDF 文档.pdf")); //关联文档对象与输出流
            document.open();                                                             //打开文档
            document.add(new Paragraph("First Document."));                              //向文档中添加内容
            document.add(new Paragraph("Success."));                                     //向文档中添加内容
            document.close();                                                            //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        }

    }
}
```

秘笈心法

心法领悟 380：创建 PDF 文档时需要注意的事项。

在创建 PDF 文档时，必须在使用 Docement 类的 open()方法打开文档后，向文档中写入内容，否则程序将发生异常，导致无法完成文档的创建。

实例 381 添加 PDF 文档标题

光盘位置：光盘\MR\381　　初级　　趣味指数：★★★★

实例说明

许多用户在发布电子文档时都会向文档属性窗口中输入一些信息，例如作品的作者、标题、主题等。本实例向 PDF 文档属性中添加作品标题，效果如图 13.2 所示。

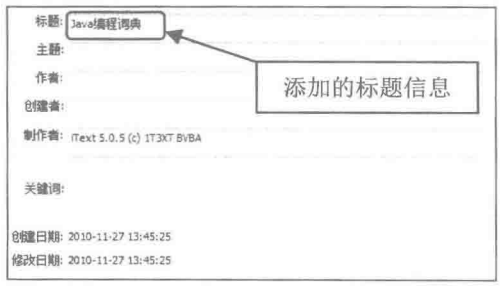


图 13.2　添加 PDF 文档属性标题

关键技术

本实例主要是通过调用 iText 类库中的 Document 类的 addTitle()方法来实现的，该方法的语法格式如下：

```
public boolean addTitle(String title)
```

其中，参数 title 为字符串类型，表示设置的标题名称。如果方法调用成功，返回值为 true，否则为 false。

设计过程

（1）创建一个普通的 Java 工程。

（2）向工程中添加一个 AddTitle 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
Document document=new Document();                                          //创建文档对象
try {
      PdfWriter.getInstance(document, new FileOutputStream("c:\\添加标题.pdf"));   //关联文档对象与输出流
      document.addTitle("Java 编程词典");                                    //向文档中添加标题
      document.open();                                                     //打开文档
      document.add(new Paragraph("Add Title"));                            //向文档中添加内容
      document.close();                                                    //关闭文档
} catch (FileNotFoundException e) {
      e.printStackTrace();
} catch (DocumentException e) {
      e.printStackTrace();
}
}
```

秘笈心法

心法领悟 381：添加标题属性的时机。

为了能够成功添加标题属性，必须在使用 Document 类的 open()方法打开文档之前进行添加，否则将无法成功添加。

实例 382　添加 PDF 文档主题

初级

趣味指数：★★★★

光盘位置：光盘\MR\382

实例说明

本实例向 PDF 文档属性中添加作品的主题，效果如图 13.3 所示。

图 13.3　添加 PDF 文档属性主题

关键技术

本实例主要是通过调用 iText 类库中的 Document 类的 addSubject()方法来实现的，该方法的语法格式如下：

```
public boolean addSubject(String subject)
```

其中，参数 subject 为字符串类型，表示设置的主题名称。如果方法调用成功，返回值为 true，否则为 false。

设计过程

（1）创建一个普通的 Java 工程。

（2）向工程中添加一个 AddSubject 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                          //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\添加 PDF 文档主题.pdf"));   //关联文档对象与输出流
        document.addSubject("学习 iText 的使用");                                  //向文档中添加主题
        document.open();                                                         //打开文档
        document.add(new Paragraph("Subject"));                                  //向文档中添加内容
        document.close();                                                        //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 382：进行文件操作时使用异常捕捉。

在程序中对文件进行操作时，由于各种原因经常会导致某一个文件函数调用失败。为了在文件操作失败时能够进行相应的处理，在调用有关文件函数时，建议使用 try…catch 异常处理语句对文件异常进行处理。这样可使程序更加健壮。

实例 383 添加 PDF 文档关键词

光盘位置：光盘\MR\383

初级

趣味指数：★★★★

实例说明

PDF 文档的关键字能够记录一段摘要信息，文档的设计者可以利用该属性来记录文档的主要作用、功能或有关文档的一些辅助信息。在本实例中就实现了这样的一个功能，效果如图 13.4 所示。

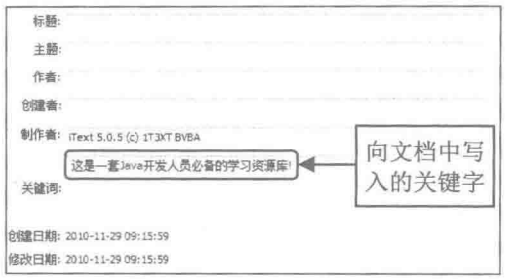


图 13.4 添加 PDF 文档关键字

关键技术

本实例主要是通过调用 iText 类库中的 Document 类的 addKeywords()方法来实现的，该方法的语法格式如下：

```
public boolean addKeywords(String keywords)
```

其中，参数 keywords 是字符串类型，用于表示设置的关键字文本信息。如果方法调用成功，返回值为 true，否则为 false。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddKeywords 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document=new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\Java 编程词典.pdf"));   //关联文档对象与输出流
        document.addKeywords("这是一套 Java 开发人员必备的学习资源库!");          //向文档中添加关键字
        document.open();                                                 //打开文档
        document.add(new Paragraph("Keywords"));                         //向文档中添加内容
        document.close();                                                //关闭文档
    }
    catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 383：字符串中的目录表示方法。

在应用程序中经常需要使用字符串表示一个目录。例如，使用“c:\Java\Java 编程词典.exe”表示 C 盘 Java 目录下的“Java 编程词典.exe”文件，但是，用户在编写 Java 代码时却不能直接使用“c:\Java\Java 编程词典.exe”来表示目录，因为在 Java 语言中，字符“\”表示转义字符，用户需要使用“\\”来表示真正的“\”字符，即“c:\\Java\\

Java 编程词典.exe”。

实例 384　添加 PDF 文档作者

光盘位置：光盘\MR\384　　中级　　趣味指数：★★★★

实例说明

用户在发表电子文档时，通常会在文档属性窗口中添加自己的名字，以表示作品的作者。本实例实现了为文档添加作者的功能，效果如图 13.5 所示。

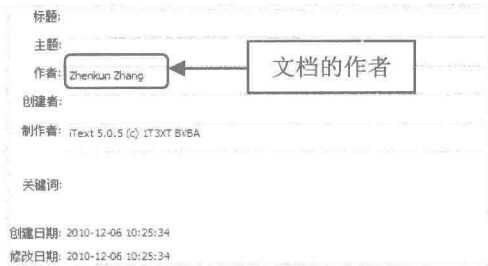


图 13.5　添加 PDF 文档作者

关键技术

本实例主要是通过调用 iText 类库中 Document 类的 addAuthor()方法来实现的，该方法的语法格式如下：

```
public boolean addAuthor(String author)
```

其中，参数 author 是字符串类型，表示设置的作者名称。如果方法调用成功，返回值为 true，否则为 false。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddAuthor 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\添加 PDF 文档作者.pdf"));   //关联文档对象与输出流
        document.addAuthor("Zhenkun Zhang");                               //向文档中添加作者
        document.open();                                                   //打开文档
        document.add(new Paragraph("Add Author."));                        //向文档中添加内容
        document.close();                                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 384：应及时关闭文档对象。

在使用 Document 类创建文档对象后，如果使用 open()方法打开了文档，该文档就会占用系统资源，因此在执行完对文档的操作后，一定要及时使用 close()方法关闭文档，以释放所占用的系统资源。

实例 385 添加 PDF 文档创建者

光盘位置：光盘\MR\385　　中级　　趣味指数：★★★★

实例说明

用户在发表电子文档时，通常会在文档属性窗口中标识自己的名字，以表示对作品拥有最终的所有权。本实例实现了指定文档创建者的功能，效果如图 13.6 所示。

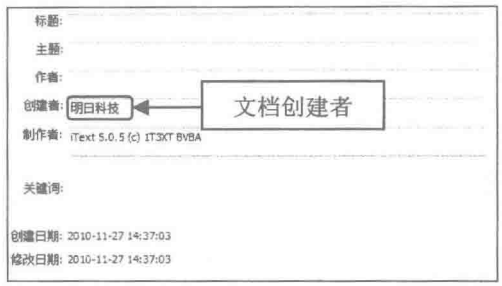


图 13.6　添加 PDF 文档创建者

关键技术

本实例主要是通过调用 iText 类库中 Document 类的 addCreator()方法来实现的，该方法的语法格式如下：

```
public boolean addCreator(String creator)
```

其中，参数 creator 是字符串类型，表示设置的创建者名称。如果方法调用成功，返回值为 true，否则为 false。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddCreator 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\Java 编程词典.pdf"));                  //关联文档对象与输出流
        document.addCreator("明日科技");                    //添加创建者
        document.open();                                   //打开文档
        document.add(new Paragraph("Creator"));            //向文档中添加内容
        document.close();                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 385：捕捉所有的异常。

在程序中使用 try…catch 语句捕捉异常时，用户需在 catch 语句部分列举需要处理的类型，但是在实际开发中，用户想要列举出所有可能出现的异常是比较困难的，而且需要编写大量的代码。对于一些需要进行相同处理的异常，可以在 catch 语句部分添加对 Exception 异常的处理，但是该异常的 catch 语句需要放置在最后，以防止截获后面 catch 语句部分的异常。

实例 386　添加 PDF 文档制作者

光盘位置：光盘\MR\386　　中级　　趣味指数：★★★★

实例说明

用户在发表电子文档时，通常会在文档属性窗口中标识制作者，即创建文档所使用的工具包。本实例实现了指定文档制作者的功能，效果如图 13.7 所示，从图中可以看到，制作者是 iText 5.0.5，即创建 PDF 文档使用的工具包是 iText 5.0.5。

图 13.7　添加 PDF 文档制作者

关键技术

本实例主要是通过调用 iText 类库中的 Document 类的 addProducer()方法来实现的，该方法的语法格式如下：

```
public boolean addProducer( )
```

参数说明

返回值：如果该方法调用成功，返回值为 true，否则为 false。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddProducer 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\添加 PDF 文档制作者.pdf"));           //关联文档对象与输出流
        document.addProducer();                            //向文档中添加制作者
        document.open();                                   //打开文档
        document.add(new Paragraph("Add Producer."));      //向文档中添加内容
        document.close();                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 386：为什么没有添加制作者，却在文档属性中显示出制作者呢？

这是因为当使用 iText 工具包创建 PDF 文档时，制作者是自动添加的，所以虽然没有为 PDF 文档添加制作者，在文档属性中仍然会显示出来。

实例 387　添加 PDF 文档创建日期

光盘位置：光盘\MR\387　　中级　　趣味指数：★★★

实例说明

在 PDF 文档的属性窗口中有一项创建日期选项，描述了文档的创建日期。文档的设计人员通过该选项可以了解文档的版本信息。本实例实现了向 PDF 文档添加创建日期的功能，效果如图 13.8 所示。

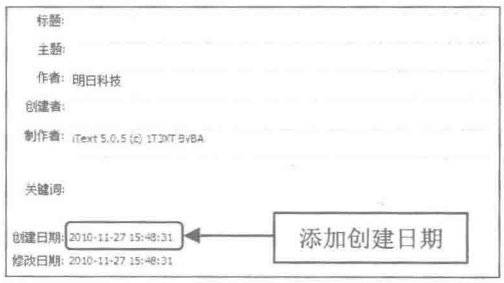


图 13.8　添加 PDF 文档创建日期

关键技术

向 PDF 文档中添加日期主要是通过调用 iText 类库中 Document 类的 addCreationDate()方法实现的，该方法的语法格式如下：

```
public boolean addCreationDate()
```

该方法不需要任何参数，它会将系统当前日期写入到文档的属性中。如果方法调用成功，返回值为 true，否则为 false。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddandCreateDate 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String args[]){
Document document = new Document();                                          //创建文档对象
try {
      PdfWriter.getInstance(document, new FileOutputStream("c:\\Java 编程词典.pdf"));   //关联文档对象与输出流
      document.addAuthor("明日科技");                                          //添加作者
      document.addCreationDate();                                            //创建日期
      document.open();                                                       //打开文档
      document.add(new Paragraph("CreateDate"));                             //向文档中添加内容
      document.close();                                                      //关闭文档
} catch (FileNotFoundException e) {
      e.printStackTrace();
} catch (DocumentException e) {
      e.printStackTrace();
}
}
```

秘笈心法

心法领悟 387：保证文件路径的存在。

在使用 FileOutputStream 文件输出流构建一个新文件时，要保证文件的路径一定存在，否则文件的创建和写入操作都将会失败。例如，在构建 FileOutputStream 对象时使用“"c:\\mr\\Java 编程词典.pdf"”作为参数，如果

“c:\\mr”路径不存在，则其后对文件流的各种操作都将失败。

13.2　初始化操作

实例 388	设置页面大小 光盘位置：光盘\MR\388	中级 趣味指数：★★★★

实例说明

对于一些具有特殊性质的文件来说，其文件的用纸大小是有严格限制的。在设计这类文档时，要求自定义页面的大小。本实例实现了在 PDF 文档中设置页面大小的功能，效果如图 13.9 所示。

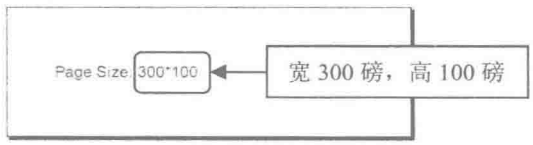


图 13.9　设置页面大小

关键技术

设置 PDF 文档页面大小主要通过调用 Document 类的 setPageSize()方法实现，该方法的语法格式如下：

```
public boolean setPageSize(Rectangle)
```

其中，参数为 Rectangle 类型，表示一个矩形区域，即页面的范围。Rectangle 类构造函数可以接收两个实数作为参数，分别用于表示矩形的宽度和高度，单位为磅，此时的起点坐标为(0, 0)。Rectangle 类构造函数也可以接收 4 个实数作为参数，这 4 个参数用于确定矩形区域的大小。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SetPageSize 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
        Document document = new Document();                                    //创建文档对象设置文档大小
        try {
                PdfWriter.getInstance(document, new FileOutputStream("c:\\Java 资料库.pdf")); //关联文档对象与输出流
                Rectangle pageSize= new Rectangle(300, 100);                   //定制页面范围
                document.setPageSize(pageSize);                                //设置页面大小
                document.open();                                               //打开文档
                document.add(new Paragraph("Page Size: 300*100"));             //向文档中添加内容
                document.close();                                              //关闭文档
        } catch (FileNotFoundException e1) {
                e1.printStackTrace();
        } catch (DocumentException e1) {
                e1.printStackTrace();
        }
}
```

秘笈心法

心法领悟 388：Java 中为何只有 new 运算符而没有 delete 运算符。

熟悉 C++语言的用户都知道，在堆中构建一个对象使用 new 运算符，这与 Java 是相同的，但是对象使用后需要释放，也就是需要调用 delete 运算符来释放对象占用的空间，否则会出现内存泄露。但是在 Java 中却没有

delete 运算符。这是因为 Java 提供了垃圾回收机制，开发人员不需要自己释放内存空间。因此，也就不需要使用 delete 运算符。

实例 389　横向显示页面

光盘位置：光盘\MR\389　　中级　　趣味指数：★★★★

实例说明

在生成 PDF 文档时，可以根据需要使页面横向显示，这样就可以在水平方向上添加更多的内容。本实例实现了在 PDF 文档中横向显示页面的功能，效果如图 13.10 所示。

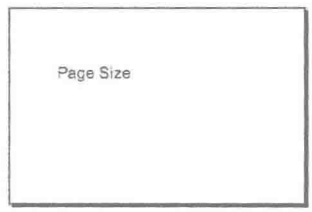


图 13.10　横向显示页面的效果

关键技术

本实例主要使用 Rectangle 类创建表示页面大小的矩形对象，然后调用 Rectangle 类的 rotate()方法将页面转换为横向，并将转换后的矩形对象设置为文档对象的页面大小，从而实现了页面横向显示的功能，关键代码如下：

```
Rectangle pageSize= new Rectangle(150,220);          //创建表示页面大小的矩形对象
pageSize = pageSize.rotate();                        //页面转换为横向
document.setPageSize(pageSize);                      //设置页面大小
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 TransverseShowPage 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                          //创建文档对象设置文档大小
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\横向显示页面.pdf"));   //关联文档对象与输出流
        Rectangle pageSize= new Rectangle(150,220);                              //创建表示页面大小的矩形对象
        pageSize = pageSize.rotate();                                            //页面转换为横向
        document.setPageSize(pageSize);                                          //设置页面大小
        document.open();                                                         //打开文档
        document.add(new Paragraph("Page Size"));                                //向文档中添加内容
        document.close();                                                        //关闭文档
    } catch (FileNotFoundException e1) {
        e1.printStackTrace();
    } catch (DocumentException e1) {
        e1.printStackTrace();
    }
}
```

秘笈心法

心法领悟 389：页面转换为横向显示需要注意的事项。

在使用 Rectangle 类的 rotate()方法将页面转换为横向时，一定要在文档打开之前进行设置，否则将发生错误，从而导致无法实现将页面转换为横向的操作。

实例 390　纵向显示页面

光盘位置：光盘\MR\390　　中级　　趣味指数：★★★★

实例说明

在生成 PDF 文档时，也可以根据需要设置页面纵向显示，这样就可以在垂直方向上添加更多的内容。本实例实现了将横向页面转换为纵向显示的功能，效果如图 13.11 所示。

图 13.11　纵向显示页面的效果

关键技术

本实例也是使用 Rectangle 类创建表示页面大小的矩形对象，只不过该矩形对象是横向的，即宽度比高度大，这种情况下可以使用 Rectangle 类的 rotate()方法将页面转换为纵向的，并将转换后的矩形对象设置为文档对象的页面大小，从而实现了页面纵向显示的功能，关键代码如下：

```
Rectangle pageSize= new Rectangle(220,150);          //创建表示页面大小的矩形对象，该矩形对象是横向显示的
pageSize = pageSize.rotate();                        //转换为纵向
document.setPageSize(pageSize);                      //设置页面大小
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 PortraitShowPage 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                    //创建文档对象设置文档大小
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\纵向显示页面.pdf"));                  //关联文档对象与输出流
        Rectangle pageSize = new Rectangle(220, 150);      //创建表示页面大小的矩形对象，该矩形对象是横向显示的
        pageSize = pageSize.rotate();                      //转换为纵向
        document.setPageSize(pageSize);                    //设置页面大小
        document.open();                                   //打开文档
        document.add(new Paragraph("Page Size"));          //向文档中添加内容
        document.close();                                  //关闭文档
    } catch (FileNotFoundException e1) {
        e1.printStackTrace();
    } catch (DocumentException e1) {
        e1.printStackTrace();
    }
}
```

秘笈心法

心法领悟 390：PDF 文档默认的显示方式。

在创建 PDF 文档时，文档页面默认是纵向显示的，但是如果手动设置了页面大小，并且页面是横向的，则可以用本实例的方法，将横向页面转换为纵向页面。

实例 391　添加水印

光盘位置：光盘\MR\391　　中级　　趣味指数：★★★★★

实例说明

PDF 文档创建后，除了可以对页面进行修饰外，还可以为其添加水印，并将添加有水印的内容存储为一个新的 PDF 文档。运行程序，可以看到添加水印之后的效果，如图 13.12 所示。

图 13.12　文档添加水印之后的效果

关键技术

本实例主要创建一个空白 PDF 文档的临时文件，然后使用 PdfReader 对象将已有的 PDF 空白文档的内容与输出流对象关联后创建 PdfStamper 对象，再使用该对象的 getUnderContent()方法获得要为其添加水印内容的 PdfContentByte 对象，并添加水印图片，然后向文档中添加内容，最后将空白的 PDF 临时文件删除，从而实现为 PDF 文档添加水印的功能。创建空白 PDF 文档的实现代码如下：

```
Document document = new Document();                                    //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream("c:\\tempWatermark.pdf"));  //关联文档对象与临时文件的输出流
document.open();                                                       //打开文档
document.add(new Paragraph(" "));                                      //向文档中添加内容
document.close();                                                      //关闭文档对象
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddWatermark 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\tempWatermark.pdf"));                             //关联文档对象与临时文件的输出流
        document.open();                                               //打开文档
        document.add(new Paragraph(" "));                              //向文档中添加内容
        document.close();                                              //关闭文档对象
        PdfReader reader = new PdfReader("c:\\tempWatermark.pdf");     //创建 tempWatermark.pdf 的 PdfReader 对象
        PdfStamper stamp = new PdfStamper(reader, new FileOutputStream("c:\\添加水印.pdf"));   //创建 PdfStamper 对象
        Image img = Image.getInstance("image/watermark.jpg");          //创建图像对象
        img.setAbsolutePosition(50, 385);                              //定位图片对象
        PdfContentByte under = stamp.getUnderContent(1);               //获得第一页的内容
        under.addImage(img);                                           //添加图片，完成水印功能
```

```
            BaseFont chinese = BaseFont.createFont("STSong-Light",
                    "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);          //定义字体
            under.beginText();                                       //标记文本开始
            under.setFontAndSize(chinese, 42);                       //设置字体和字号
            under.setTextMatrix(70, 550);                            //设置添加内容的显示位置
            under.showText("下面是添加的水印图片.");                  //添加内容
            under.endText();                                         //标记文本结束
            stamp.close();      //PdfStamper 对象，将从 tempWatermark.pdf 中读取的文档添加水印后写入“添加水印.pdf”
            File file = new File("c:\\tempWatermark.pdf");           //创建临时文件的 File 对象
            file.delete();                                           //删除临时文件
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 391：为文档的多页添加水印和文本内容。

本实例只为 PDF 文档的一页添加水印图片和文本内容，如果需要添加更多的页，并为每一页添加水印和文本内容，可以使用 PdfStamper 对象的 insertPage()方法插入新页，然后使用本实例的方法即可为 PDF 文档的多页添加水印和文本内容。插入新页的代码如下：

```
stamp.insertPage(2, PageSize.A4);          //使用 PdfStamper 对象 stamp 调用 insertPage()方法插入新页
```

实例 392　添加页眉和页脚

光盘位置：光盘\MR\392　　中级　　趣味指数：★★★★

实例说明

PDF 文档创建后，有时根据需要还可以向页面添加页眉和页脚信息。本实例实现了为页面添加页眉和页脚的功能，运行程序，可以看到添加页眉和页脚之后的效果，如图 13.13 和图 13.14 所示。

图 13.13　添加页眉的效果

图 13.14　添加页脚的效果

关键技术

本实例主要是使用 PdfReader 对象，将已有 PDF 文档的内容与输出流对象关联后创建 PdfStamper 对象，然后使用该对象的 getOverContent()和 getUnderContent()方法，获得要为其添加页眉和页脚内容的 PdfContentByte 对象，从而实现为 PDF 文档添加页眉和页脚的功能。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddPageFoot 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                              //创建文档对象
    try {
```

```
            PdrWriter.getInstance(document, new FileOutputStream("c:\\页眉页脚.pdf"));    //关联文档对象与输出流
            document.open();                                                              //打开文档
            document.add(new Paragraph("Add Page Top And Foot."));                        //向文档中添加内容
            document.close();                                                             //关闭文档对象
            PdfReader reader = new PdfReader("c:\\页眉页脚.pdf");                          //创建“页眉页脚.pdf”的 PdfReader 对象
            PdfStamper stamp = new PdfStamper(reader, new FileOutputStream("c:\\添加页眉页脚.pdf")); //创建 PdfStamper 对象
            PdfContentByte over = stamp.getOverContent(1);                                //获得第一页的内容
            over.setTextRise(810);                                                        //文本上移到 810 的位置
            over.beginText();                                                             //标记文本开始
            BaseFont chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);  //定义字体
            over.setFontAndSize(chinese, 18);                                             //设置字体和字号
            over.showText("页眉的内容");                                                   //添加页眉
            over.endText();                                                               //标记文本结束
            stamp.insertPage(2, PageSize.A4);                                             //增加新的一页，为新页添加页脚
            PdfContentByte under = stamp.getUnderContent(2);                              //获得第二页的内容
            under.setTextRise(15);                                                        //文本上移到 15 的位置
            under.beginText();                                                            //标记文本开始
            under.setFontAndSize(chinese, 18);                                            //设置字体和字号
            under.showText("页脚的内容");                                                  //添加页脚
            under.endText();                                                              //标记文本结束
            stamp.close();          //PdfStamper 对象，将从“页眉页脚.pdf”中读取的文档添加页眉页脚后写入“添加页眉页脚.pdf”
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 392：注意 PdfContentByte 类的 setTextRise()方法的参数。

通过 PdfContentByte 对象的 setTextRise()方法，可以设置页眉和页脚的显示位置，不过在使用该方法时，如果参数是小数，必须在数值后加上字符 f 或 F，以表示这是一个单精度浮点数，否则程序会出错，因为该方法只能接受单精度浮点数。

实例 393　创建新页

光盘位置：光盘\MR\393　　中级　　趣味指数：★★★★

实例说明

在创建 PDF 文档时，在一页写了部分内容后，往往还需要创建新页，以添加更多的内容。本实例实现了在 PDF 文档中添加新页的功能，运行程序，可以看到文档中有两页内容，效果如图 13.15 和图 13.16 所示。

图 13.15　文档第 1 页显示的内容

图 13.16　文档中新页的显示内容

关键技术

在 PDF 文档中创建新页主要是通过调用 Document 类的 newPage()方法实现的，该方法的语法格式如下：

```
public boolean newPage( )
```

该方法没有参数，如果新页创建成功，则返回 true，否则返回 false。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 CreateNewPage 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\创建新页.pdf"));  //关联文档与输出流
        document.open();                                                   //打开文档
        document.add(new Paragraph("Old Page"));                           //为第 1 页添加内容
        document.newPage();                                                //创建新的页
        document.add(new Paragraph("New Page"));                           //为新页添加内容
        document.close();                                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 393：为文档创建更多的新页。

本实例只有两页内容，如果希望创建更多的新页，可以再次调用 Document 对象的 newPage()方法，程序中每调用一次该方法，就会创建一个新页，这样即可实现创建更多新页的功能。

实例 394　为 PDF 文档添加页码

光盘位置：光盘\MR\394　　中级　　趣味指数：★★★

实例说明

在创建 PDF 文档时，可以为文档添加页码信息，例如“第 2 页/共 3 页”。本实例实现了为 PDF 文档添加页码的功能，运行程序，可以看到生成的文档中显示有页码信息，其中文档第 2 页的显示效果如图 13.17 所示。

图 13.17　文档第 2 页显示的页码信息

关键技术

本实例主要是通过继承 PdfPageEventHelper 类，并重写该类的 onOpenDocument()、onEndPage()和 onCloseDocument()方法，然后为 PdfWriter 对象添加事件监听器来实现为 PDF 文档添加页码的功能。

（1）onOpenDocument()方法在文档打开时执行创建模板和字体的初始化操作。

（2）onEndPage()方法是在一个页面结束时执行的操作，本实例中该方法主要是为页面添加页码和模板。

（3）onCloseDocument()方法在文档关闭时执行，用于向模板中添加总页数。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddPageNumber 类，该类继承自 PdfPageEventHelper 类，并重写了 onOpenDocument()、onEndPage()和 onCloseDocument()方法。

（3）在 onOpenDocument()方法中实现在文档打开时创建模板和字体的初始化操作，代码如下：

```
pdfTemplate = writer.getDirectContent().createTemplate(180, 180);          //创建模板对象
try {
baseFont = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H",
        BaseFont.NOT_EMBEDDED);                                             //创建基础字体
} catch (DocumentException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
```

（4）在 onEndPage()方法中实现在页面结束时为页面添加页码和模板的操作，代码如下：

```
PdfContentByte cb = writer.getDirectContent();                              //创建内容对象
cb.saveState();                                                             //保存状态
cb.beginText();                                                             //文本开始标记
cb.setFontAndSize(baseFont, 12);                                            //设置字体和字号
cb.setTextMatrix(260, 800);                                                 //设置文本显示位置
String page = "第" + writer.getPageNumber() + "页/共";                      //定义表示页码信息的变量
cb.showText(page);                                                          //显示文本
cb.endText();                                                               //文本结束标记
cb.addTemplate(pdfTemplate, 305, 800);                                      //添加模板对象
cb.stroke();                                                                //确认并保存操作
cb.restoreState();                                                          //恢复标记
cb.closePath();                                                             //关闭内容通道
```

（5）在 onCloseDocument()方法中实现在关闭文档时向模板中添加总页数的操作，代码如下：

```
pdfTemplate.beginText();                                                    //模板文本开始标记
pdfTemplate.setFontAndSize(baseFont, 12);                                   //设置模板字体和字号
int totalPages = writer.getPageNumber() - 1;                                //获得总页数
pdfTemplate.showText(totalPages + "页");                                    //显示总页数信息
pdfTemplate.endText();                                                      //模板文本结束标记
pdfTemplate.closePath();                                                    //关闭模板通道
```

秘笈心法

心法领悟 394：为文档每一页添加通用的内容。

在创建文档时，如果文档中每一页都有相同的内容，可以先创建一个模板对象，然后在模板对象上添加这些相同的内容，最后再将模板添加到文档中，这样就可以方便地向文档中的每一页添加相同的内容，而不必再手动重复添加，从而可以极大地提高工作效率。

13.3 字体与中文处理

实例 395 设置特殊的文本字体　　中级

光盘位置：光盘\MR\395　　趣味指数：★★★

实例说明

对于文档中的一些重要内容，为了进行突出显示，通常将其设置为特殊的字体形式。在本实例中，将设计一个具有特殊文本字体的 PDF 文档，效果如图 13.18 所示。

图 13.18　设置特殊的文本字体

关键技术

为 PDF 文档设置字体主要是在定义段落对象 Paragraph 时指定一个字体。那么如何定义字体对象呢？在 iText 类库中提供了 Font 类用于表示字体对象。在构建 Font 类对象时，通常需要使用 BaseFont 类对象作为参数，BaseFont 类表示了字体的基本信息，例如字体的名称、编码和风格等内容。下面的代码演示了如何设置文本字体：

```
BaseFont bfChinese = BaseFont.createFont(BaseFont.TIMES_ROMAN, BaseFont.CP1252, BaseFont.NOT_EMBEDDED); //定义基本字体
    Font contentFont = new Font(bfChinese, 20);                    //定义普通字体和大小
    document.add(new Paragraph("Java Function Classes", contentFont));    //向文档中添加内容并指定普通字体
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SetDocumentFontDemo 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c://Java 类库.pdf")); //关联文档对象与输出流
        document.open();                                                   //打开文档
        //定义基本字体
        BaseFont bfChinese = BaseFont.createFont(BaseFont.TIMES_ROMAN, BaseFont.CP1252, BaseFont.NOT_EMBEDDED);
        Font contentFont = new Font(bfChinese, 20);                        //定义字体和大小
        document.add(new Paragraph("Java Function Classes", contentFont)); //向文档中添加内容并指定普通字体
        document.close();                                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 395：类静态方法的使用。

在本实例中，构建 BaseFont 类对象 bfChinese 时是通过调用 createFont()方法实现的。该方法属于 BaseFont 类的静态方法，可以直接通过类名来调用，即 BaseFont.createFont。

实例 396　加粗字体　　中级

光盘位置：光盘\MR\396　　趣味指数：★★★

实例说明

在设置电子文档时，对于文档中的重要内容，通常使用特殊的效果加以标示。例如，将文档内容加粗显示。在本例中实现了对 PDF 文档中的内容加粗显示，效果如图 13.19 所示。

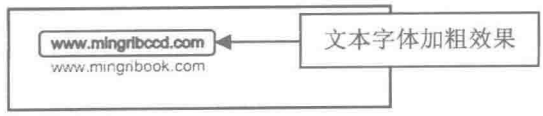


图 13.19　加粗字体

关键技术

实现文本的字体加粗非常简单，只要在构建字体对象时使字体风格中包含 Font.BOLD 风格即可。例如：

```
Font contentFont = new Font(bfChinese, 12, Font.BOLD);
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 BoldFontDocumentDemo 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\Java.pdf"));   //关联文档对象与输出流
        document.open();                                                   //打开文档
        BaseFont bfChinese = BaseFont.createFont(BaseFont.HELVETICA, BaseFont.CP1252, BaseFont.NOT_EMBEDDED);  //定义基本字体
        Font contentFont = new Font(bfChinese, 12, Font.BOLD);             //定义加粗字体
        document.add(new Paragraph("www.mingribccd.com",contentFont));     //向文档中添加内容并指定加粗字体
        document.add(new Paragraph("www.mingribook.com"));
        document.close();                                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
            e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 396：定义不确定参数个数的方法。

用户在开发应用程序时，有时需要动态确定方法参数的个数。此时该如何定义方法呢？可以在定义方法参数时使用“...”来表示方法的参数不确定。例如：

```
  Public int add(int ...arr){
for(int i=0; i<arr.length; i++){
  //代码省略
}
  }
```

实例 397　添加下划线

光盘位置：光盘\MR\397　　中级　　趣味指数：★★★

实例说明

通常用户在浏览文档时会对重点内容使用下划线进行标识，以便以后复习或者其他人阅读时可以重点看这些内容，可提高阅读效率。本实例实现了向 PDF 文档中内容添加下划线的功能，效果如图 13.20 所示。

图 13.20　添加下划线

关键技术

在 PDF 文档中设置下划线，主要是在构建段落对象 Paragraph 时使用的字体风格中包含 UNDERLINE 风格，例如：

```
new Paragraph("Welcome to GuangZhou！", FontFactory.getFont(FontFactory.HELVETICA,15,Font.UNDERLINE))
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddUnderline 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                                          //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\明日科技简介.pdf"));   //关联文档对象与输出流
        document.open();                                                         //打开文档
        document.add(new Paragraph("Generalize"));                               //向文档中添加内容
        //向文档中添加内容
        document.add(new Paragraph("Welcome to GuangZhou！", FontFactory.getFont(FontFactory.HELVETICA,15,Font.UNDERLINE)));
        document.close();                                                        //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 397：保证代码一定会被执行。

在 Java 的异常处理语句中有一个 finally 子句，用于标识无论 try 部分的代码是否发生异常，都将执行 finally 部分的代码。甚至用户在 try 部分使用 return 语句时也会执行 finally 部分的代码。

实例 398　添加删除线

光盘位置：光盘\MR\398

中级

趣味指数：★★★★

实例说明

用户在编辑或校订文档时，对于建议删除的内容，通常在文本上标识出删除线。本实例实现了在 PDF 文档中添加删除线的功能，效果如图 13.21 所示。

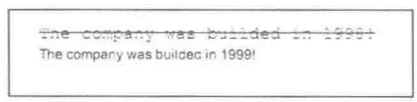


图 13.21　添加删除线

关键技术

在 PDF 中对文本添加下划线非常简单，只要在构建段落对象 Paragraph 时使用的字体风格中包含 STRIKETHRU 风格即可。例如：

```
new Paragraph("The company was builded in 1998!", FontFactory.getFont(FontFactory.COURIER, 15, Font.NORMAL | Font.STRIKETHRU))
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 DeleteUnderline 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
Document document = new Document();                                          //创建文档对象
try {
      PdfWriter.getInstance(document, new FileOutputStream("c:\\公司简介.pdf"));   //关联文档对象与输出流
      document.open();                                                       //打开文档
      document.add(new Paragraph("The company was builded in 1998!",
          FontFactory.getFont(FontFactory.COURIER, 15, Font.NORMAL | Font.STRIKETHRU))); //向文档中添加内容
      document.add(new Paragraph("The company was builded in 1999!"));       //向文档中添加内容
      document.close();                                                      //关闭文档
} catch (FileNotFoundException e) {
      e.printStackTrace();
} catch (DocumentException e) {
      e.printStackTrace();
}
}
```

秘笈心法

心法领悟 398：获取异常抛出点。

当程序中的代码发生异常时，系统通常会弹出错误提示。但是开发人员很难确定是哪一条语句出现了问题。为了准确地找出出现异常的代码，通常在异常捕捉语句的 catch 部分调用异常类的 printStackTrace()方法来输出抛出异常的“地点”。

实例 399　在 PDF 文档中显示中文

光盘位置：光盘\MR\399　　高级　　趣味指数：★★★★

实例说明

开发人员在使用 iText 开发包创建 PDF 文档时，当向文档中输入数据时会发现，英文可以正常显示，但是中文却显示不出来。本实例实现了在 PDF 文档中显示中文的功能，效果如图 13.22 所示。

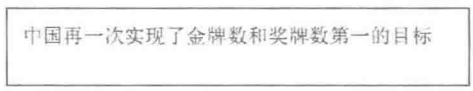


图 13.22　在 PDF 文档中显示中文

关键技术

在 PDF 文档中显示中文主要是在设置文本时指定一个支持中文的字体。例如，在定义字体时使用 STSong-Light 为字体名称，采用 UniGB-UCS2-H 字符编码，即可显示中文。代码如下：

```
BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);   //定义基础字体
Font FontChinese = new Font(Chinese, 20, Font.NORMAL);          //实例化字体类与设置字体大小属性
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 HandleChineseErrorCode 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
      Document document = new Document();                                          //创建文档对象
      try {
            PdfWriter.getInstance(document, new FileOutputStream("c:\\亚运速递.pdf"));   //关联文档对象与输出流
            document.open();                                                       //打开文档
            BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);   //定义基础字体
            Font FontChinese = new Font(Chinese, 20, Font.NORMAL);                 //实例化字体类与设置字体大小属性
            document.add(new Paragraph("中国再一次实现了金牌数和奖牌数第一的目标", FontChinese));   //向文档中添加内容并指定中文
```

```
            document.close();                                          //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 399：重新解压和打包.jar 文件。

本实例在使用 iText 和 iTextAsian 包时，出现了包名不一致的问题，导致系统无法发现 STSong-Light 和 UniGB-UCS2-H。在 iText5.x 以上版本中的 font 和 encoding 文件都位于 com/itextpdf/text/pdf/fonts 下，而 iTextAsian 中的 font 位于 com/lowagie/text/pdf/fonts 下，导致了访问 font 信息的路径错误。为了解决该问题，需要对 iTextAsian 包进行重新解压和打包。首先解压 iTextAsian 包，将 lowagie 目录修改为 itextpdf，然后在命令行输入 jar cvf iTextAsian.jar com/itextpdf/text/pdf/fonts/*，最后将生成的 iTextAsian.jar 放置在工程目录 lib 下，在工程中导入 iTextAsian.jar 即可。

实例 400　设置 PDF 文档密码

光盘位置：光盘\MR\400　　高级　　趣味指数：★★★★

实例说明

为了保证文档的安全，可以在生成 PDF 文档时为其添加文档打开密码，这样在打开文档时就会弹出如图 13.23 所示的“密码”对话框，只有输入正确的密码才能打开文档看到其中的内容。

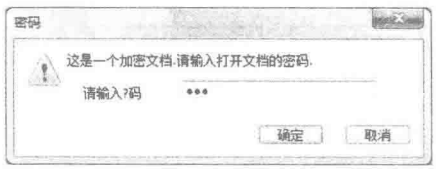


图 13.23　打开文档的“密码”对话框

关键技术

显示打开文档的密码对话框，主要是通过 PdfWriter 类的 setEncryption()方法实现的。例如：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\设置密码.pdf"));      //关联文档对象与输出流
writer.setEncryption("zzk".getBytes(),"123".getBytes(),PdfWriter.ALLOW_COPY,
                                        PdfWriter.STANDARD_ENCRYPTION_128);      //设置密码参数和常量
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SetPassWord 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                          //创建文档对象
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\设置密码.pdf"));                       //关联文档对象与输出流
        writer.setEncryption("zzk".getBytes(), "123".getBytes(),
                PdfWriter.ALLOW_COPY, PdfWriter.STANDARD_ENCRYPTION_128);        //设置密码参数和常量
        document.open();                                                         //打开文档
```

```
            document.add(new Paragraph("Set Encryption"));                    //向文档添加内容
            document.close();                                                 //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 400：加密的代码的放置位置。

在使用 PdfWriter 类的 setEncryption()方法为 PDF 文档加密时，一定要在文档打开之前进行设置，否则程序会发生异常。

13.4 块、短语、段落、章节和区域

实例 401 添加和创建块　　中级

光盘位置：光盘\MR\401　　趣味指数：★★★

实例说明

在 PDF 文档中，块（chunk）是最小的文本单位，其次是短语、段落、文档等。通过块来添加文本，用户可以控制指定文本的字体，即使这些文本位于一个段落中，效果如图 13.24 所示。

图 13.24　添加和创建块

关键技术

向文档中添加块，首先需要构建一个块对象，然后调用文档对象 Document 的 add()方法将块添加到文档中。下面重点介绍块对象的构建。

在 iText 类库中提供了 Chunk 类用于表示块，该类的构造函数格式为：

```
Chunk(String content, Font font)
```

其中，参数 content 表示块的文本，该文本将显示在文档中；font 表示块文本使用的字体。下面的代码演示了如何定义一个块对象：

```
Chunk chunk1 = new Chunk("Text chunk1",FontFactory.getFont(FontFactory.COURIER_BOLD,15,Font.ITALIC));
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 CreateAddChunk 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String args[]){
    Document document = new Document();                                       //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\Java.pdf"));  //关联文档对象与输出流
        document.open();                                                      //打开文档
        //创建块并定义字体属性和添加内容
        Chunk chunk1 = new Chunk("Text chunk1",FontFactory.getFont(FontFactory.COURIER_BOLD,15,Font.ITALIC)) ;
```

```
            document.add(chunk1);                                          //向文档中添加块
            //创建块并定义字体属性和添加内容
            Chunk chunk2 = new Chunk("Text chunk2",FontFactory.getFont(FontFactory.COURIER_BOLD,30,Font.BOLD));
            document.add(chunk2);                                          //向文档中添加块
            document.close();                                              //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 401：局部变量的声明和初始化。

用户在定义类的成员变量时，可以不必进行初始化，而在构造函数中为成员变量赋值。但是对于局部变量来说，在定义时则必须进行初始化，否则将会出现编译错误。

实例 402　设置上标和下标

光盘位置：光盘\MR\402　　中级　　趣味指数：★★★

实例说明

在生活中经常需要用到上标和下标，例如 X^2 和 Y_2 等。本实例实现了为 PDF 文档中的块添加上标和下标的功能，效果如图 13.25 所示。

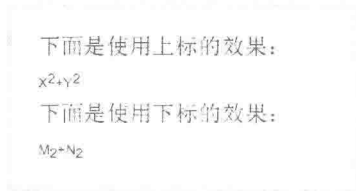


图 13.25　设置上标和下标的效果

关键技术

为 PDF 文档中的块添加上标和下标，主要用到 Chunk 类的 setTextRise()方法，其参数是一个 float 类型的单精度浮点数，为正值时表示提升块，即将块设置为上标；为负值时表示降低块，即将块设置为下标。例如，下面的代码将块 2 设置为上标，代码运行的结果为 X^2。

```
Chunk chunk = new Chunk("X");          //创建块
document.add(chunk);                   //向文档添加内容
chunk = new Chunk("2");                //创建块
chunk.setTextRise(4.0f);               //提升块文本
document.add(chunk);                   //添加块
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 CreateAddChunk 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\设置上标和下标.pdf"));                                 //关联文档对象与输出流
        document.open();
        BaseFont Chinese = BaseFont.createFont("STSong-Light",
```

```
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                    //定义基础字体
        Font fontChinese = new Font(Chinese, 20, Font.NORMAL);             //实例化字体类与设置字体大小属性
        document.add(new Paragraph("下面是使用上标的效果：", fontChinese));  //添加段落
        Chunk chunk = new Chunk("X");                                      //创建块
        document.add(chunk);                                               //向文档添加内容
        chunk = new Chunk("2");                                            //创建块
        chunk.setTextRise(4.0f);                                           //提升块文本
        document.add(chunk);                                               //添加块
        chunk = new Chunk("+Y");                                           //创建块
        document.add(chunk);                                               //添加块
        chunk = new Chunk("2");                                            //创建块
        chunk.setTextRise(4.0f);                                           //提升块文本
        document.add(chunk);                                               //添加块
        document.add(new Paragraph("下面是使用下标的效果：", fontChinese));  //添加段落
        chunk = new Chunk("M");                                            //创建块
        document.add(chunk);                                               //添加块
        chunk = new Chunk("2");                                            //创建块
        chunk.setTextRise(-3.0f);                                          //降低块文本
        document.add(chunk);                                               //添加块
        chunk = new Chunk("+N");                                           //创建块
        document.add(chunk);                                               //添加块
        chunk = new Chunk("2");                                            //创建块
        chunk.setTextRise(-3.0f);                                          //降低块文本
        document.add(chunk);                                               //添加块
        document.close();                                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 402：将添加到文档中的块换行。

在文档中添加块对象时，如果希望换一行添加块，可以在添加需要换行的块之前，先添加一个段落对象，然后再添加块对象即可实现块换行显示。

实例 403 设置文本背景颜色

光盘位置：光盘\MR\403　　中级　　趣味指数：★★★

实例说明

许多文本编辑器都能够设置文本的背景颜色，PDF 文档自然也不例外。本实例中将通过程序来设置 PDF 文档的文本背景颜色，效果如图 13.26 所示。

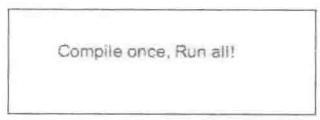


图 13.26　设置文本背景颜色

关键技术

本实例主要是通过调用 iText 类库中的 Chunk 类的 setBackground()方法来实现的，该方法的语法格式如下：

Chunk setBackground(Color color)

其中，参数 color 表示设置的背景颜色。例如，下面的代码将背景颜色设置为淡灰色。

```
chunk.setBackground(BaseColor.LIGHT_GRAY);
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SetChunkBackground 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                                          //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\Java 编程全能词典.pdf")); //关联文档对象与输出流
        document.open();                                                         //打开文档
        Chunk chunk = new Chunk("Compile once, Run all!");                       //定义块并添加内容
        chunk.setBackground(BaseColor.LIGHT_GRAY);                               //设置背景颜色
        document.add(chunk);                                                     //添加背景颜色
        document.close();                                                        //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 403：方法的返回值为何不作为方法重载的区分标识？

方法重载是指方法名称相同、参数类型或参数个数不相同的一组方法。在调用重载方法时，编译器会根据参数类型和参数个数进行区分，但是编译器不以方法的返回值类型作为区分重载方法的标志。例如：

```
void f();
int f();
```

当执行 f();语句时，编译器无法区分用户调用哪个方法，因此，方法的返回值不作为区分方法重载的标识。

实例 404　添加和创建短语

光盘位置：光盘\MR\404　　中级　　趣味指数：★★★★

实例说明

在 PDF 文档中，短语是块的集合。在一个短语中有一个主字体，但是短语中的块可以有自己的字体，一个短语中所有的块都具有相同的行距。本实例演示如何向 PDF 文档中添加短语，效果如图 13.27 所示。

图 13.27　添加和创建短语

关键技术

向文档中添加短语，首先需要构建一个短语对象，然后调用文档对象 Document 的 add()方法将短语添加到文档中。下面重点介绍短语对象的构建。

在 iText 类库中提供了 Phrase 类用于表示短语，该类的构造函数格式为：

Phrase(String string)

其中，参数 string 表示短语的文本，该文本将显示在文档中。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 CreateAddPhrase 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String args[]){
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\book.pdf")); //关联对象与输出流
        document.open();                                                   //打开文档
        Phrase phrase1 = new Phrase("BeiJing Olympics");                   //创建短语并添加内容
        document.add(phrase1);
        Phrase phrase2 = new Phrase("One world, one dream!");              //创建短语并添加内容
        document.add(phrase2);
        document.close();                                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 404：访问同名的外部类成员。

在设计复杂的类时，有时在类内部还会包含一个子类，并成为内部类。当这个内部类中的成员与外部类成员同名时，该如何在内部类中访问外部类成员呢？下面的代码实现了这样的功能。

```
public class One{
    private int x;
    private class Two{
        private int x;
        public void Insert(int x)
        {
            x++;            //访问参数 x
            this.x++;       //访问类 Two 的成员变量 x
            One.this.x++;   //访问类 One 的成员变量 x
        }
    }
}
```

为了访问外部类中的同名成员变量，需要使用外部类名作为前缀，然后是 this，最后是成员变量名，即以“One.this.x”的形式来访问。

实例 405　添加和创建段落

光盘位置：光盘\MR\405　　中级　　趣味指数：★★★★

实例说明

在 PDF 文档中，段落是短语的集合。用户在设计自己的 PDF 文档时，第一步通常是创建一个段落，然后向段落中添加各种样式的文本。本实例实现了向 PDF 文档中添加段落的功能，效果如图 13.28 所示。

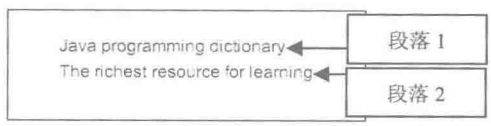


图 13.28　添加和创建段落

关键技术

实现向文档中添加段落，首先需要构建一个段落对象 Paragraph，然后将段落对象添加到文档 Document 中。下面重点介绍如何构建段落对象。

在 iText 类库中提供了 Paragraph 类用于描述 PDF 文档中的段落，该类的构造函数如下：

```
Paragraph(String string)
```

其中，参数 string 表示段落中显示的文本。下面的代码描述了如何创建段落，并将其添加到文本中。

```
Paragraph P1 = new Paragraph("Java programming dictionary");
document.add(P1);
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 CreateAddParagraph 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
Document document = new Document();                                          //创建文档对象
try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\Java 编程词典.pdf"));   //关联文档对象与输出流
        document.open();                                                     //打开文档
        Paragraph P1 = new Paragraph("Java programming dictionary");         //创建段落对象
        document.add(P1);                                                    //向文档添加段落
        Paragraph P2 = new Paragraph("The richest resource for learning");   //再次创建一个段落对象
        document.add(P2);                                                    //向文档添加段落
} catch (FileNotFoundException e) {
            e.printStackTrace();
} catch (DocumentException e) {
            e.printStackTrace();
}
finally{
        document.close();                                                    //关闭文档
}
}
```

秘笈心法

心法领悟 405：finally 语句的作用。

在程序中，开发人员经常使用 try 语句来捕捉可能出现的异常，并在 catch 语句部分处理异常。Java 程序在运行过程中如果触发了异常，那么触发异常代码后面的代码就不会被执行，程序将直接跳转到相应的 catch 语句部分处理异常，这将导致异常发生时，有些操作可能会被取消。一些重要的操作，例如关闭已经打开的文档，会要求无论是否发生异常都应该保证被执行。对于这部分操作，用户在编写异常处理语句时应将其放置在 finally 语句部分，以保证无论异常是否发生，操作都能够被执行。

实例 406　设置段落首行缩进

光盘位置：光盘\MR\406　　中级　　趣味指数：★★★★

实例说明

通常在写文章时一个段落的开始都会空出一小块空间，这样有利于阅读。这种形式已成为一种文档的书写规范，称为段落首行缩进。在本实例中，将通过程序来设置段落的首行缩进，效果如图 13.29 所示。

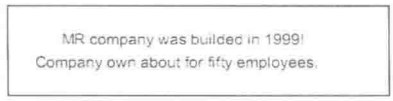


图 13.29　设置段落首行缩进

关键技术

实现段落的首行缩进主要是调用段落对象 Paragraph 的 setFirstLineIndent()方法来实现的，该方法的语法格式如下：

```
void setFirstLineIndent(float firstLineIndent)
```

其中，参数 firstLineIndent 标识首行缩进的大小，单位为磅。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SetParagraphFirstIndent 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                                          //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\公司简介.pdf")); //关联文档对象与输出流
        document.open();                                                         //打开文档
        Paragraph P1 = new Paragraph("MR company was builded in 1999!");         //创建段落并添加内容
        P1.setFirstLineIndent(20);                                               //设置段落首行缩进
        document.add(new Paragraph(P1));                                         //向文档添加段落
        Paragraph P2 = new Paragraph("Company own about for fifty employees."); //创建段落并添加内容
        document.add(P2);                                                        //向文档添加段落
        document.close();                                                        //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 406：数组的快速填充方法。

在程序中为数组赋值时，通常采用循环的方式遍历数组元素，依次对每一个元素赋值。如果要求数组中所有元素都具有相同的值，可以应用 Arrays 类的 fill()方法来整体对数组元素赋值。例如，下面的语句将数组所有元素赋值为 1。

```
int arr[] = new int[5];
Arrays.fill(arr, 1);
```

实例 407 设置段落的上下间距

光盘位置：光盘\MR\407　　中级　　趣味指数：★★★★

实例说明

在 Word 文档中，用户可以非常方便地调整段落间距，实际上，在 PDF 文档中用户也可以设置段落间距。本实例就实现了该功能，效果如图 13.30 所示。

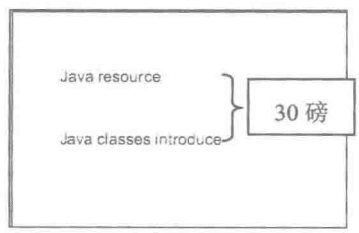


图 13.30　设置段落上下间距

关键技术

在 iText 类库中的 Paragraph 类提供了 setSpacingBefore()方法用于设置段落的上边距，提供了 setSpacingAfter()方法用于设置段落的下边距。其中，setSpacingBefore()方法的语法格式如下：

```
void setSpacingBefore(float spacing)
```

其中，参数 spacing 表示设置的上边距，单位为磅。

setSpacingAfter()方法的语法格式如下：

```
void setSpacingAfter(float spacing)
```

其中，参数 spacing 表示设置的下边距，单位为磅。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SetParagraphSpacingAfter 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\Java.pdf"));   //关联文档对象与输出流
        document.open();                                                   //打开文档
        Paragraph paragraph1 = new Paragraph("Java resource");             //创建段落添加内容
        paragraph1.setSpacingBefore(10);                                   //设置段落上边距
        paragraph1.setSpacingAfter(30);                                    //设置段落下边距
        document.add(paragraph1);                                          //向文档添加段落
        Paragraph paragraph2 = new Paragraph("Java classes introduce");    //创建段落添加内容
        paragraph2.setSpacingAfter(30);                                    //设置段落下边距
        document.add(paragraph2);                                          //向文档添加段落
        document.close();                                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 407：只导入包中的静态成员。

用户在使用其他人开发的类库时，程序中有时只使用一部分静态成员，如果导入整个类库或整个类，都会使程序变得臃肿。可以在调入包时使用 static 关键字，表示只导入某一静态成员。例如：

```
Import static java.lang.Math.max;
```

实例 408　设置段落左右缩进

光盘位置：光盘\MR\408　　中级　　趣味指数：★★★★

实例说明

使用 PDF 设置电子文档时，用户不但可以设置段落的上下间距，还可以调整段落的左右间距。本实例就实现了这样的功能，效果如图 13.31 所示。

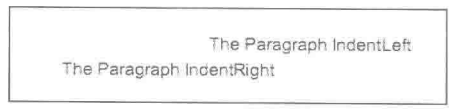


图 13.31　设置段落左右缩进

关键技术

设置段落的左右缩进主要是调用了段落对象 Paragraph 的 setIndentationLeft()和 setIndentationRight()方法来实现。其中，setIndentationLeft()方法用于设置段落的左缩进，该方法的语法格式如下：

```
void setIndentationLeft(float indentation)
```

其中，参数 indentation 表示设置的段落左缩进量，单位为磅。

setIndentationRight()方法用于设置段落的右缩进，该方法的语法格式如下：

```
void setIndentationRight(float indentation)
```

其中，参数 indentation 表示设置的段落右缩进量，单位为磅。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SetParagraphLeftRightIndent 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                                          //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\Java 编程词典.pdf"));   //关联文档对象与输出流
        document.open();                                                         //打开文档
        Paragraph paragraph1 = new Paragraph("The Paragraph IndentLeft");        //实例化段落并添加内容
        paragraph1.setIndentationLeft(100);                                      //段落左缩进
        document.add(paragraph1);                                                //向文档中添加段落
        Paragraph paragraph2 = new Paragraph("The Paragraph IndentRight");       //实例化段落并添加内容
        paragraph2.setIndentationRight(100);                                     //段落右缩进
        document.add(paragraph2);                                                //向文档中添加段落
        document.close();                                                        //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 408：运行时类型识别。

在编写程序代码时，经常需要判断某一个对象是否是指定的类型。在 Java 中提供了 instanceof 关键字用于辨别对象的类型。例如，下面的代码用于验证对象 obj 是否为 JFrame 类型。

```
if (obj instanceof JFrame)
{
    //代码省略
}
```

实例 409 设置段落的对齐方式

光盘位置：光盘\MR\409　　中级　　趣味指数：★★★★

实例说明

在设置 PDF 文档时，用户不但可以调整段落的上、下、左、右间距，还可以调整段落的对齐方式。例如，将某个段落居左对齐、居中对齐、居右对齐等。本实例就实现了这样的功能，效果如图 13.32 所示。

关键技术

设置段落的对齐方式主要是通过调用 Paragraph 对象的 setAlignment()方法实现的，该方法的语法格式如下：

```
void setAlignment(int alignment)
```

其中，参数 alignment 表示段落的对齐方式。为 Element.ALIGN_LEFT，表示居左对齐；为 Element.ALIGN_CENTER，表示居中对齐；为 Element.ALIGN_RIGHT，表示居右对齐。

图 13.32　设置段落的对齐方式

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SetParagraphAlignment 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
        Document document = new Document();                                    //创建文档对象
        try {
                PdfWriter.getInstance(document, new FileOutputStream("c:\\Java.pdf"));  //关联文档对象与输出流
                document.open();                                               //打开文档
                Paragraph paragraph1 = new Paragraph("www.mingrosoft.com");    //创建段落并添加内容
                paragraph1.setAlignment(Element.ALIGN_LEFT);                   //左对齐
                document.add(new Paragraph(paragraph1));                       //向文档添加段落
                Paragraph paragraph2 = new Paragraph("www.mingribook.com");    //创建段落并添加内容
                paragraph2.setAlignment(Element.ALIGN_RIGHT);                  //右对齐
                document.add(new Paragraph(paragraph2));                       //向文档添加段落
                Paragraph paragraph3 = new Paragraph("www.mingribccd.com");    //创建段落并添加内容
                paragraph3.setAlignment(Element.ALIGN_CENTER);                 //居中对齐
                document.add(new Paragraph(paragraph3));                       //向文档添加段落
                document.close();                                              //关闭文档
        } catch (FileNotFoundException e) {
                e.printStackTrace();
        } catch (DocumentException e) {
                e.printStackTrace();
        }
 }
```

秘笈心法

心法领悟 409：在构造函数中调用其他构造函数。

用户在设计类时，通常需要定义几个构造函数，某一个构造函数可能需要使用其他构造函数的代码，如果采用复制的方式，会使构造函数的代码臃肿。其实，用户完全可以直接调用另一个构造函数。例如：

```
public class   Student{
        Student(int nAge){                                                     //构造函数
        //代码省略
 }
        Student(){                                                             //构造函数
            this(5);                                                           //调用上一个构造函数
        //代码省略
 }

}
```

实例 410　设置段落字体大小

光盘位置：光盘\MR\410　　中级　　趣味指数：★★★★

实例说明

用户在编写文档时，通常将文档中的标题设置为大一些的字体，以使标题更加突出和醒目。本实例实现了

向 PDF 写入文本时设置字体大小的功能，效果如图 13.33 所示。

图 13.33　设置段落字体大小

关键技术

设置段落字体大小主要是在创建段落对象时设置一个指定大小的字体。这里的关键是如何构建一个指定大小的字体。可以利用 iText 类库中提供的 FontFactory 类的 getFont()重载方法来实现，该方法的语法格式如下：

```
static Font getFont(String fontname, float size, int style)
```

其中，参数 fontname 表示字体的名称；size 表示字体的大小；style 表示字体的风格。方法返回值为创建的字体。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SetParagraphFontSize 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
        Document document = new Document();                                    //创建文档对象
        try {
                PdfWriter.getInstance(document, new FileOutputStream("c:\\个人简历.pdf"));    //关联文档对象与输出流
                document.open();                                               //打开文档
                //定义段落字体属性并添加内容
                document.add(new Paragraph("Personal Resumes",FontFactory.getFont(FontFactory.HELVETICA,50,Font.BOLDITALIC)));
                document.close();                                              //关闭文档
        } catch (FileNotFoundException e) {
                e.printStackTrace();
        } catch (DocumentException e) {
                e.printStackTrace();
        }
}
```

秘笈心法

心法领悟 410：在嵌套的复合语句中不允许定义重名的局部变量。

在 C++中，只要同名的两个局部变量位于不同的作用域中，它们是可以同时存在的，但是在 Java 中，则是不允许的。例如，下面的代码在 C++中是合法的，但是在 Java 中则是非法的。

```
{
        int nAge = 10;
        {
                int nAge = 30;
        }
}
```

Java 的设计者认为，同名的局部变量除了会给程序开发人员增加烦恼外，不会有任何好处，因此不允许在嵌套的复合语句中定义同名的局部变量。

实例 411　设置段落文本颜色

光盘位置：光盘\MR\411　　中级　　趣味指数：★★★★★

实例说明

在设计电子文档时，为了使页面更加美观和易于阅读，通常将某一类文本设置为特殊的颜色。例如，将文

档中所有标题的颜色设置为蓝色。本实例实现了设置 PDF 文档中的文本颜色的功能，效果如图 13.34 所示。

图 13.34　设置段落文本颜色

关键技术

设置段落文本颜色主要是在创建段落对象时设置一个指定颜色的字体。这里的关键是如何构建一个指定颜色的字体。可以利用 iText 类库中提供的 FontFactory 类的 getFont()重载方法来实现，该方法的语法格式如下：

```
static Font getFont(String fontname, float size, Color color)
```

其中，参数 fontname 表示字体的名称；size 表示字体的大小；color 表示字体颜色。方法返回值为创建的字体。

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SetParagraphColor 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                                          //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\Java 类库参考手册.pdf")); //关联文档对象与输出流
        document.open();                                                         //打开文档
        Paragraph paragraph = new Paragraph(new Paragraph("JFrame Class Member List",
            FontFactory.getFont(FontFactory.HELVETICA, 30, BaseColor.BLUE )));   //创建段落、定义字体并添加内容
        paragraph.setFirstLineIndent(100);                                       //设置段落首行缩进
        document.add(paragraph);                                                 //向文档添加段落
        document.close();                                                        //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 411：理解文件输入流与文件输出流。

在 Java 中，文件输入流用于从文件中读取数据，文件输出流用于向文件写入数据。那么用户如何理解输入和输出呢？其实，这里的输入和输出不是以文件为中心描述的，而是以内存为中心，即向内存中写入数据为输入流，例如加载文件数据到内存中；从内存中提取数据为输出流，例如将内存中的数据写入磁盘文件中。

实例 412　添加章节

光盘位置：光盘\MR\412　　中级　　趣味指数：★★★★

实例说明

为了方便对 PDF 文档的阅读，可以为 PDF 文档添加章节信息，这样就会在书签列表中显示章节信息，如图 13.35 所示，单击“书签”中的列表项，就会在文档页面显示对应的章节内容，如图 13.36 所示。

关键技术

本实例主要通过使用 Chapter 类创建章节对象，然后将该章节对象添加到 Document 对象中实现，向文档对象中添加章节的代码如下：

```
Chapter chapter = new Chapter("This is chapter 1", 1);          //创建与新章节对象关联的内容
document.add(chapter);                                          //向文档中添加章节
```

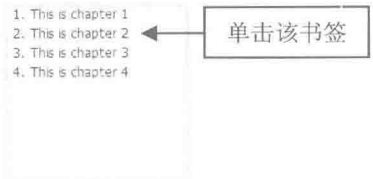


图 13.35　章节的书签列表

图 13.36　章节显示的页信息

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddChapter 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                         //创建文档对象
    try {
        PdfWriter.getInstance(document,
                new FileOutputStream("c:\\添加章节.pdf"));      //关联文档对象与输出流
        document.open();                                        //打开文档
        Chapter chapter = new Chapter("This is chapter 1", 1);  //创建与新章节对象关联的内容
        document.add(chapter);                                  //向文档中添加章节
        chapter = new Chapter("This is chapter 2", 2);          //创建与新章节对象关联的内容
        document.add(chapter);                                  //向文档中添加章节
        chapter = new Chapter("This is chapter 3", 3);          //创建与新章节对象关联的内容
        document.add(chapter);                                  //向文档中添加章节
        chapter = new Chapter("This is chapter 4", 4);          //创建与新章节对象关联的内容
        document.add(chapter);                                  //向文档中添加章节
        document.close();                                       //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 412：在添加章节的页添加多页段落。

在为文档添加章节时，还可以在该章节中添加多页段落文档，方法是在 PDF 文档中添加完章节，然后向这个章节中添加段落文本，然后创建新页，创建完新页再添加新的段落文本，例如：

```
Chapter chapter = new Chapter("This is chapter 1", 1);          //创建与新章节对象关联的内容
document.add(chapter);                                          //向文档中添加章节
document.add(new Paragraph("One"));                             //添加段落文本
document.newPage();                                             //分页
document.add(new Paragraph("Two"));                             //在新页中添加段落文本
```

实例 413　在章节中添加小节

初级

光盘位置：光盘\MR\413

趣味指数：★★★★

实例说明

为了方便对 PDF 文档的阅读，可以为 PDF 文档添加章节信息，同样也可以在章节中添加小节信息，本实例实现了在章节中添加小节信息的功能，效果如图 13.37 所示。

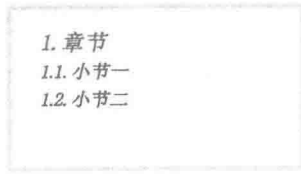


图 13.37　章节中添加小节后的效果

关键技术

本实例主要使用 Chapter 类创建章节对象，然后使用 Chapter 类的 addSection()方法向章节中添加小节，最后再将章节对象添加到 Document 对象中。向章节对象中添加小节的代码如下：

```
paragraph = new Paragraph("小节一",fontChinese2);                    //创建段落对象
chapter.addSection(paragraph);                                       //在章节中添加小节
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddSectionInChapter 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\在章节中添加小节.pdf"));    //关联文档对象与输出流
        document.open();                                                                   //打开文档
        BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);    //定义基础字体
        Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,BaseColor.RED);  //实例化字体类、设置字体大小和颜色
        Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,BaseColor.BLUE); //实例化字体类、设置字体大小和颜色
        Paragraph paragraph = new Paragraph("章节",fontChinese1);                  //创建段落对象
        Chapter chapter = new Chapter(paragraph,1);                                //创建章节对象
        paragraph = new Paragraph("小节一",fontChinese2);                          //创建段落对象
        chapter.addSection(paragraph);                                             //添加小节
        paragraph = new Paragraph("小节二",fontChinese2);                          //创建段落对象
        chapter.addSection(paragraph);                                             //添加小节
        document.add(chapter);                                                     //向文档添加章节
        document.close();                                                          //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 413：添加多个章节并为每个章节添加小节。

本实例只添加了一个章节，在实际工作中可能需要添加多个章节，并为每个章节添加小节，这可以通过多次创建章节对象，然后为每个章节对象添加小节，最后再把所有的章节对象都添加到文档中实现。

实例 414　在小节中添加列表

光盘位置：光盘\MR\414　　中级　　趣味指数：★★★★

实例说明

在 PDF 文档中，除了可以添加章节、小节以外，还可以为小节添加内容，如添加列表、段落、表格和图片

等。本实例实现了在 PDF 文档的小节中添加列表的功能，效果如图 13.38 所示。

图 13.38　小节中添加列表后的效果

关键技术

本实例主要使用 Chapter 类的 addSection()方法获得小节对象，然后使用 List 创建列表对象，并将列表对象添加到小节中，从而实现了在小节中添加列表的功能。向小节中添加列表的代码如下：

```
List list = new List(true, false, 10);                                    //创建列表
list.add(new ListItem("小节中的列表一",fontChinese3));                    //向列表添加内容
list.add(new ListItem("小节中的列表二",fontChinese3));                    //向列表添加内容
list.add(new ListItem("小节中的列表三",fontChinese3));                    //向列表添加内容
section.add(list);                                                        //向小节添加列表
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddListInSection 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                                   //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\在小节中添加列表.pdf"));   //关联文档对象与输出流
        document.open();                                                  //打开文档
        BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);   //定义基础字体
        Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,BaseColor.RED);   //实例化字体类、设置字体大小和颜色
        Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,BaseColor.BLUE);  //实例化字体类、设置字体大小和颜色
        Font fontChinese3 = new Font(Chinese, 12, Font.NORMAL,BaseColor.BLACK);     //实例化字体类、设置字体大小和颜色
        Paragraph paragraph = new Paragraph("章节",fontChinese1);         //创建段落对象
        Chapter chapter = new Chapter(paragraph,1);                       //创建章节对象
        paragraph = new Paragraph("小节",fontChinese2);                   //创建段落对象
        Section section = chapter.addSection(paragraph);                  //创建并加入小节对象
        paragraph = new Paragraph("\n 小节中添加的列表如下：\n\n",fontChinese3);   //创建段落对象
        section.add(paragraph);                                           //向小节添加段落
        List list = new List(true, false, 10);                            //创建列表
        list.add(new ListItem("小节中的列表一",fontChinese3));            //向列表添加内容
        list.add(new ListItem("小节中的列表二",fontChinese3));
        list.add(new ListItem("小节中的列表三",fontChinese3));
        section.add(list);                                                //向小节添加列表
        document.add(chapter);                                            //向文档添加章节
        document.close();                                                 //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 414：对列表进行分类并放到不同的小节中。

如果文档中的列表较多，可以对列表进行分类，然后把每一类列表分别放到不同的小节中，这样通过单击

小节即可查看其中的列表，极大地方便用户对列表的操作和查找。

实例 415　在小节中添加段落

光盘位置：光盘\MR\415　　中级　　趣味指数：★★★★

实例说明

本实例实现了在 PDF 文档的小节中添加段落的功能，效果如图 13.39 所示。

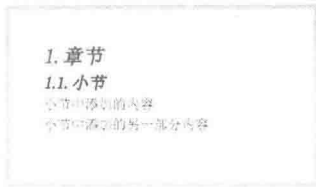


图 13.39　小节中添加段落后的效果

关键技术

本实例主要使用 Chapter 类的 addSection()方法获得小节对象，然后使用 Paragraph 类创建段落对象，并将段落对象添加到小节中，从而实现了在小节中添加段落的功能。向小节中添加段落的代码如下：

```
paragraph = new Paragraph("小节中添加的内容",fontChinese3);                    //创建段落
section.add(paragraph);                                                        //向小节添加段落内容
paragraph = new Paragraph("小节中添加的另一部分内容",fontChinese3);            //创建段落
section.add(paragraph);                                                        //向小节添加段落内容
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddParagraphInSection 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
    Document document = new Document();                                        //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\在小节中添加段落.pdf"));   //关联文档对象与输出流
        document.open();                                                       //打开文档
        BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);   //定义基础字体
        Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,BaseColor.RED);      //实例化字体类、设置字体大小和颜色
        Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,BaseColor.BLUE);     //实例化字体类、设置字体大小和颜色
        Font fontChinese3 = new Font(Chinese, 12, Font.NORMAL,BaseColor.BLACK);        //实例化字体类、设置字体大小和颜色
        Paragraph paragraph = new Paragraph("章节",fontChinese1);              //创建段落对象
        Chapter chapter = new Chapter(paragraph,1);                            //创建章节对象
        paragraph = new Paragraph("小节",fontChinese2);                        //创建段落对象
        Section section = chapter.addSection(paragraph);                       //创建并加入小节对象
        paragraph = new Paragraph("小节中添加的内容",fontChinese3);            //创建段落
        section.add(paragraph);                                                //向小节添加段落内容
        paragraph = new Paragraph("小节中添加的另一部分内容",fontChinese3);    //创建段落
        section.add(paragraph);                                                //向小节添加段落内容
        document.add(chapter);                                                 //向文档添加章节
        document.close();                                                      //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 415：解决 Chapter 类创建的章节不显示汉字的问题。

在使用 Chapter 类的重载构造方法 Chapter(String title,int num)创建章节时，如果为参数 title 指定汉字，则在文档中是不会显示的，这时可以使用另一个重载构造方法 Chapter(Paragraph paragraph,int num)创建章节对象，这样可以对参数 paragraph 进行处理使其显示汉字，本实例就使用了该构造方法，所以在文档中能够显示汉字。

实例 416　在小节中添加表格

光盘位置：光盘\MR\416　　高级　　趣味指数：★★★★

实例说明

本实例实现了在 PDF 文档的小节中添加表格的功能，效果如图 13.40 所示。

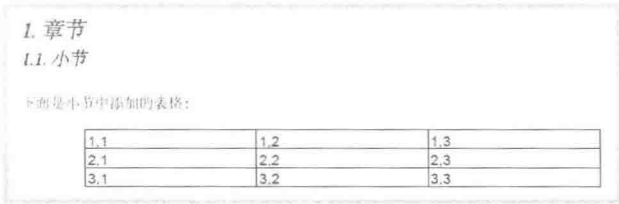


图 13.40　小节中添加表格后的效果

关键技术

本实例主要使用 Chapter 类的 addSection()方法获得小节对象，然后使用 PdfTable 类创建表格对象，并将表格对象添加到小节中，从而实现了在小节中添加表格的功能。向小节中添加表格的代码如下：

```
PdfPTable table = new PdfPTable(3);                    //创建表格对象
//将单元格顺次加入到表格，当一行充满时自动换行
table.addCell("1,1");
table.addCell("1,2");
table.addCell("1,3");
section.add(table);                                    //将表格添加到小节
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddTableInSection 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\在小节中添加表格.pdf"));          //关联文档对象与输出流
        document.open();                               //打开文档
        BaseFont Chinese = BaseFont.createFont("STSong-Light",
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED); //定义基础字体
        Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,
                BaseColor.RED);                        //实例化字体类、设置字体大小和颜色
        Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,
                BaseColor.BLUE);                       //实例化字体类、设置字体大小和颜色
        Font fontChinese3 = new Font(Chinese, 12, Font.NORMAL,
                BaseColor.BLACK);                      //实例化字体类、设置字体大小和颜色
        Paragraph paragraph = new Paragraph("章节", fontChinese1); //创建段落对象
        Chapter chapter = new Chapter(paragraph, 1);   //创建章节对象
        paragraph = new Paragraph("小节", fontChinese2); //创建段落对象
```

```
            Section section = chapter.addSection(paragraph);                    //创建并加入小节对象
            PdfPTable table = new PdfPTable(3);                                 //创建表格对象
            //将单元格顺次加入到表格，当一行充满时自动换行
            table.addCell("1,1");
            table.addCell("1,2");
            table.addCell("1,3");
            table.addCell("2,1");
            table.addCell("2,2");
            table.addCell("2,3");
            table.addCell("3,1");
            table.addCell("3,2");
            table.addCell("3,3");
            paragraph = new Paragraph("\n 下面是小节中添加的表格：\n\n", fontChinese3);   //创建段落
            section.add(paragraph);                                             //向小节添加段落内容
            section.add(table);                                                 //将表格添加到小节
            document.add(chapter);                                              //向文档添加章节
            document.close();                                                   //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 416：在表格的单元格中显示汉字。

在 PDF 表格的单元格中添加中文时，单元格中并不会显示，这时可以创建一个对中文进行处理的段落对象，然后将该段落对象添加到单元格中，这样就能在表格的单元格中显示汉字了。关键代码如下：

```
paragraph = new Paragraph("表格", fontChinese);                    //创建段落
table.addCell(paragraph);                                          //将单元格顺次加入到表格，当一行充满时自动换行
```

实例 417　在小节中添加图片

高级

光盘位置：光盘\MR\417

趣味指数：★★★★★

实例说明

本实例实现了在 PDF 文档的小节中添加图片的功能，效果如图 13.41 所示。

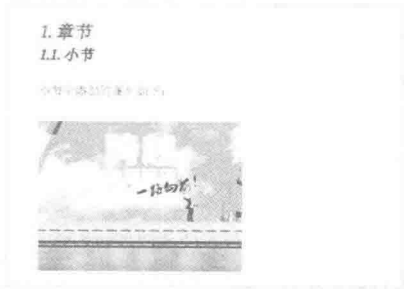


图 13.41　小节中添加图片后的效果

关键技术

本实例主要使用 Chapter 类的 addSection()方法获得小节对象，然后使用 Image 类创建指定图片的图像对象，并将图像对象添加到小节中，从而实现了在小节中添加图片的功能。向小节中添加图片的代码如下：

```
Image image = Image.getInstance("image/image.jpg");                    //定义图片信息
image.scalePercent(40);                                                //设置图片的显示百分比
section.add(image);                                                    //向小节添加图片
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 AddImageInSection 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\在小节中添加图片.pdf"));                          //关联文档对象与输出流
        document.open();                                               //打开文档
        BaseFont Chinese = BaseFont.createFont("STSong-Light",
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                //定义基础字体
        Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,
                BaseColor.RED);                                        //实例化字体类、设置字体大小和颜色
        Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,
                BaseColor.BLUE);                                       //实例化字体类、设置字体大小和颜色
        Font fontChinese3 = new Font(Chinese, 12, Font.NORMAL,
                BaseColor.BLACK);                                      //实例化字体类、设置字体大小和颜色
        Paragraph paragraph = new Paragraph("章节", fontChinese1);     //创建段落对象
        Chapter chapter = new Chapter(paragraph, 1);                   //创建章节对象
        paragraph = new Paragraph("小节", fontChinese2);               //创建段落对象
        Section section = chapter.addSection(paragraph);               //创建并加入小节对象
        paragraph = new Paragraph("\n 小节中添加的图片如下：\n\n", fontChinese3);   //创建段落对象
        section.add(paragraph);                                        //向小节添加段落
        Image image = Image.getInstance("image/image.jpg");            //定义图片信息
        image.scalePercent(40);                                        //设置图片的显示百分比
        section.add(image);                                            //向小节添加图片
        document.add(chapter);                                         //向文档添加章节
        document.close();                                              //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 417：在小节中混合添加表格和图像。

在 PDF 文档中，可以根据需要绘制内容，当需要图片时，可以将图像添加到小节中，如果该小节还需要使用表格，可以再向小节中添加表格，添加完图像和表格后，再将章节添加到 PDF 文档中，这样就实现了在小节中混合添加表格和图像的功能。

13.5 读取 PDF 文档

实例 418 文本文件转换为 PDF 文档　　中级

光盘位置：光盘\MR\418　　趣味指数：★★★

实例说明

在创建 PDF 文档时，如果要在文档中添加的内容已在某个文本文件中存在，可以直接将文本文件转换为 PDF

文档，这样在 PDF 文档中就会显示文本文件中的内容了。本实例实现了将文本文件转换为 PDF 文档的功能，假设文本文件中有如下 3 行内容：

这是一个美丽的故事
这是一个神奇的传说
在很久很久以前，有一个.....

则将文本文件转换为 PDF 文档后的效果如图 13.42 所示。

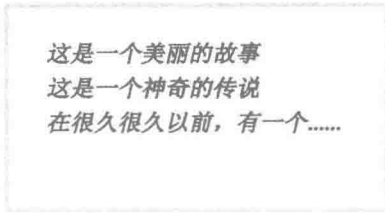


图 13.42　文本文件转换为 PDF 文档的效果

关键技术

本实例主要通过 BufferedReader 对象从文本文件中读取内容，然后将读取的内容添加到 PDF 文档中，从而实现了将文本文件转换为 PDF 文档的功能，关键代码如下：

```
String line = null;                                    //存储从文本文件中读取的内容
while ((line = read.readLine()) != null) {             //读取一行信息
doc.add(new Paragraph(line, fontChinese));             //将读取的信息添加到文档中
}
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 TextFileToPdf 类，在该类中创建一个 txtFileToPdfFile()方法，该方法实现将文本文件转换为 PDF 文档，该方法的代码如下：

```
/**
 * 将文本文件转换为 PDF 文件的方法
 * @param txtFile 原文本文件的路径
 * @param pdfFile 生成 PDF 文件的路径
 */
private static void txtFileToPdfFile(String txtFile, String pdfFile) {
        Document doc = new Document();                                  //创建文档对象
        try {
                FileReader fileRead = new FileReader(txtFile);          //创建字符流对象
                BufferedReader read = new BufferedReader(fileRead);     //创建字符缓冲流对象
                PdfWriter.getInstance(doc, new FileOutputStream(pdfFile));  //关联文档和输出流对象
                doc.open();                                             //打开文档
                BaseFont Chinese = BaseFont.createFont("STSong-Light",
                                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);  //定义基础字体
                Font fontChinese = new Font(Chinese, 18, Font.BOLDITALIC,
                                BaseColor.BLUE);                        //实例化字体类、设置字体大小和颜色
                String line = null;                                     //存储从文本文件中读取的内容
                while ((line = read.readLine()) != null) {              //读取一行信息
                        doc.add(new Paragraph(line, fontChinese));      //将读取的信息添加到文档中
                }
                doc.close();                                            //关闭文档对象
        } catch (Exception e) {
                e.printStackTrace();
        }
}
```

（3）在 TextFileToPdf 类中添加 main()方法，在该方法中调用 txtFileToPdfFile()方法，实现将文本文件转换为 PDF 文档的操作，代码如下：

```
public static void main(String[] args) {
    //将文本文件 oldTextFile.txt 转换为 PDF 文件 newPdfFile.pdf
    txtFileToPdfFile("textFile\\oldTextFile.txt", "c:\\newPdfFile.pdf");
}
```

秘笈心法

心法领悟 418：定义可以直接被主方法调用的方法。

为了操作方便，本实例将实现文本文件转换为 PDF 文档的方法定义为静态方法，即使用 static 关键字修饰该方法，这样就可以在主方法中直接调用该方法实现文件转换，用户只需要为其传递所需的参数即可。

实例 419 读取 PDF 文档

光盘位置：光盘\MR\419　　中级　　趣味指数：★★★

实例说明

PDF 文档创建后，可以通过程序读取其中的文本信息。本实例实现了通过程序读取 PDF 文档中文本信息的功能。运行程序，在控制台显示读取到的文本内容，如图 13.43 所示。

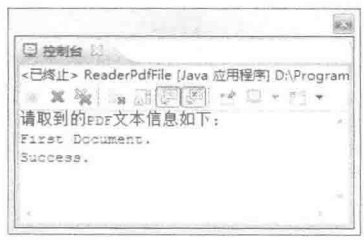


图 13.43　从 PDF 文档中读取的文本内容

关键技术

本实例主要通过 PDFBox 开源工具实现了对 PDF 文件中文本内容的读取，关键代码如下：

```
PDFParser parser = new PDFParser(in);                          //创建 PDF 解析器
parser.parse();                                                //解析 PDF 文档
PDDocument pdfdocument = parser.getPDDocument();               //获得解析后的 PDF 文档
PDFTextStripper stripper = new PDFTextStripper();              //创建 PDF 文本剥离器
String msg = stripper.getText(pdfdocument);                    //使用剥离器从 PDF 文档中剥离文本信息
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 ReaderPdfFile 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) throws MalformedURLException {
    Document document = new Document();                        //创建文档对象
    File file = new File("c:\\创建第一个 PDF 文档.pdf");          //创建 File 对象
    try {
        FileInputStream in = new FileInputStream(file);        //创建输入流对象
        PDFParser parser = new PDFParser(in);                  //创建 PDF 解析器
        parser.parse();                                        //解析 PDF 文档
        PDDocument pdfdocument = parser.getPDDocument();       //获得解析后的 PDF 文档
        PDFTextStripper stripper = new PDFTextStripper();      //创建 PDF 文本剥离器
        String msg = stripper.getText(pdfdocument);            //使用剥离器从 PDF 文档中剥离文本信息
        System.out.println("请取到的 PDF 文本信息如下:\n" + msg);  //输出信息
        in.close();                                            //关闭输入流对象
    } catch (Exception e) {
        e.printStackTrace();
```

```
        }
        document.close();                                                    //关闭文档
    }
```

秘笈心法

心法领悟 419：解决 PDFBox 读取 PDF 文档报错的问题。

在使用 PDFBox 读取 PDF 文档时，除了添加 PDFBox 开源工具包外，还需要添加 commons 工具包、fontbox 工具包和 iText 的所有工具包，否则在使用本实例读取 PDF 文档时就会报错。

实例 420　读取加密的 PDF 文档

光盘位置：光盘\MR\420　　中级　　趣味指数：★★★★

实例说明

在实例 400 中实现了对 PDF 文档加密的功能，那么该如何通过程序读取加密的 PDF 文档呢？本实例实现了该功能。运行程序，在控制台显示从加密的 PDF 文档中读取的文本内容，效果如图 13.44 所示。

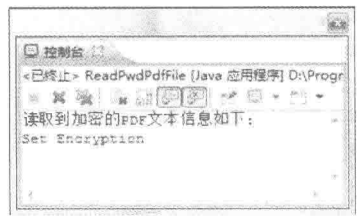


图 13.44　读取加密的 PDF 文档内容

关键技术

本实例主要通过 PdfReader 类对加密的 PDF 文档进行解密，然后将解密后的内容存储到一个临时的 PDF 文件中，并使用 PDFBox 开源工具从这个临时的 PDF 文件中读取文本内容。

使用 PDFReader 类将加密的 PDF 文档读取到临时文件中的代码如下：

```
PdfReader reader = new PdfReader("c:\\设置密码.pdf", "123".getBytes());                    //创建“水印.pdf”的 PdfReader 对象
PdfStamper stamp = new PdfStamper(reader, new FileOutputStream("c:\\TempFile.pdf"));       //创建 PdfStamper 对象
stamp.close();                    //关闭 PdfStamper 对象，并将从“设置密码.pdf”中读取的内容写入 TempFile.pdf
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 ReadPwdPdfFile 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) throws MalformedURLException {
    try {
        PdfReader reader = new PdfReader("c:\\设置密码.pdf", "123".getBytes());            //创建“水印.pdf”的 PdfReader 对象
        PdfStamper stamp = new PdfStamper(reader, new FileOutputStream("c:\\TempFile.pdf"));   //创建 PdfStamper 对象
        stamp.close();                    //关闭 PdfStamper 对象，并将从“设置密码.pdf”中读取的内容写入 TempFile.pdf
        Document document = new Document();                                            //创建文档对象
        File file = new File("c:\\TempFile.pdf");                                      //创建 File 对象
        try {
            FileInputStream in = new FileInputStream(file);                            //创建输入流对象
            PDFParser parser = new PDFParser(in);                                      //创建 PDF
            parser.parse();                                                            //解析 PDF 文档
            PDDocument pdfdocument = parser.getPDDocument();                           //获得解析后的 PDF 文档
            PDFTextStripper stripper = new PDFTextStripper();                          //创建 PDF 文本剥离器
```

```
            String msg = stripper.getText(pdfdocument);                    //使用剥离器从 PDF 文档中剥离文本信息
            System.out.println("读取到加密的 PDF 文本信息如下:\n" + msg);      //输出信息
            in.close();                                                    //关闭输入流对象
        } catch (Exception e) {
            e.printStackTrace();
        }
        document.close();                                                  //关闭文档
        file.delete();                                                     //删除 TempFile.pdf
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 420：使用文档打开密码还是拥有者密码？

在使用 PdfReader 类读取加密的 PDF 文档时，如果使用的是文档打开密码，程序会出错，因此需要使用文档拥有者密码，这样即可通过程序读取加密的 PDF 文档。

实例 421 编辑 PDF 文档

光盘位置：光盘\MR\421　　中级　　趣味指数：★★★★

实例说明

在 PDF 文档创建后，如果对文档内容不满意，可以对文档进行编辑，例如添加页码、文本内容等。本实例实现了编辑 PDF 文档的功能，如图 13.45 和图 13.46 所示分别是原文档的内容和对原文档进行编辑后的内容。

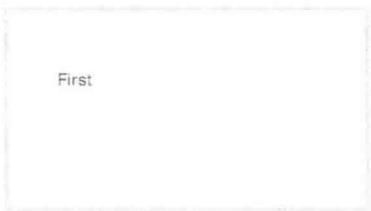


图 13.45　原文档编辑前的效果

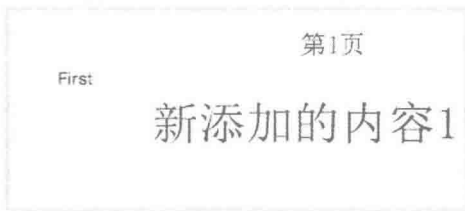


图 13.46　原文档编辑后的效果

关键技术

本实例将原文件中的内容保存到临时文件中，然后对临时文件进行编辑，编辑后删除原文件，并将临时文件重命名为原文件的名称，从而实现了对原文件进行编辑的功能。其中，删除原文件，并将临时文件重命名的实现代码如下：

```
File oldFile = new File("c:\\原文档.pdf");                    //创建原文件的 File 对象
oldFile.delete();                                             //删除原文件
File tempFile = new File("c:\\编辑后文档的临时文件.pdf");      //创建临时文件的 File 对象
tempFile.renameTo(oldFile);                                   //重命名临时文件为原文件名
tempFile.delete();                                            //删除临时文件
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 EditPdfDocumentDemo 类，在该类中创建一个 createOldFile()方法，用于创建“原文档.pdf”文件，其代码如下：

```
public static void createOldFile() {                                    //创建原文件的方法
    Document document = new Document();                                  //创建文档对象
    try {
        PdfWriter
                .getInstance(document, new FileOutputStream("c:\\原文档.pdf"));  //关联文档对象与输出流
        document.open();                                                 //打开文档
        document.add(new Paragraph("First"));                            //向文档中添加内容
        document.newPage();
        document.add(new Paragraph("Second"));                           //向文档中添加内容
        document.newPage();
        document.add(new Paragraph("Third"));                            //向文档中添加内容
        document.close();                                                //关闭文档对象
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

（3）在 EditPdfDocumentDemo 类中再创建一个 editOldFile()方法，用于对“原文档.pdf”文件进行编辑，本实例向“原文档.pdf”文件中添加了页码和文本内容，其代码如下：

```
public static void editOldFile() {                                      //编辑原文件的方法
    try {
        PdfReader reader = new PdfReader("c:\\原文档.pdf");              //创建“原文档.pdf”的 PdfReader 对象
        int totalPages = reader.getNumberOfPages();                      //获得总页数
        PdfStamper stamp = new PdfStamper(reader, new FileOutputStream(
                "c:\\编辑后文档的临时文件.pdf"));                        //创建 PdfStamper 对象
        BaseFont chinese = BaseFont.createFont("STSong-Light",
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                  //定义字体
        PdfContentByte under = null;
        for (int i = 1; i <= totalPages; i++) {
            under = stamp.getUnderContent(i);                            //获得每一页的内容
            under.beginText();                                           //标记文本开始
            under.setFontAndSize(chinese, 18);                           //设置字体和字号
            under.setTextMatrix(200, 810);                               //设置页码的显示位置
            under.showText("第" + i + "页");                             //添加页脚
            under.endText();                                             //标记文本结束
            under.beginText();                                           //标记文本开始
            under.setFontAndSize(chinese, 32);                           //设置字体和字号
            under.setTextMatrix(100, 750);                               //设置文本的显示位置
            under.showText("新添加的内容" + i);                          //添加新文本
            under.endText();                                             //标记文本结束
        }
        stamp.close();      //PdfStamper 对象，将从“原文档.pdf”中读取的文档添加页码后写入“编辑后文档的临时文件.pdf”
        File oldFile = new File("c:\\原文档.pdf");                       //创建原文件的 File 对象
        oldFile.delete();                                                //删除原文件
        File tempFile = new File("c:\\编辑后文档的临时文件.pdf");         //创建临时文件的 File 对象
        tempFile.renameTo(oldFile);                                      //重命名临时文件为原文件名
        tempFile.delete();                                               //删除临时文件
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 421：旋转文档中添加的文本信息的角度。

编辑文档时，除了可以添加页码和文本内容等信息外，还可以对添加的文本信息进行处理，较常用的处理

就是旋转添加的文本信息的角度，可以通过如下代码实现：

```
AffineTransform af= new AffineTransform();                    //定义 AffineTransform 对象
af.rotate(0.5f);                                              //指定旋转角度
under.setTextMatrix(af);                                      //设置旋转角度
```

实例 422　导入已有文档

光盘位置：光盘\MR\422　　中级　　趣味指数：★★★★

实例说明

在使用 PDF 文档时，可以根据需要导入已有文档。本实例实现了导入已有文档的功能。运行程序，将完成已有文档导入到新文档中的操作，并弹出如图 13.47 所示的消息框进行提示。

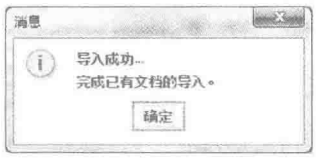


图 13.47　提示导入成功对话框

关键技术

本实例主要是通过 PdfReader 类和 PdfStamper 类将已有 PDF 文档与输出流对象关联，从而实现导入已有文档的功能，实现代码如下：

```
PdfReader reader = new PdfReader("c:\\newPdfFile.pdf");                    //创建已有文档的 PdfReader 对象
PdfStamper stamp = new PdfStamper(reader, new FileOutputStream("c:\\导入已有文档.pdf")); //关联已有文档与输出流
stamp.close();                                                             //关闭 PdfStamper 对象，完成文档导入功能
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 ImportAlreadyDocument 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    try {
        PdfReader reader = new PdfReader("c:\\newPdfFile.pdf");            //创建已有文档的 PdfReader 对象
        PdfStamper stamp = new PdfStamper(reader, new FileOutputStream(
                "c:\\导入已有文档.pdf"));                                  //关联已有文档与输出流
        stamp.close();                                                     //关闭 PdfStamper 对象，完成文档导入功能
        JOptionPane.showMessageDialog(null, "导入成功...\n 完成已有文档的导入。");
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 422：解决导入已有文件时发生的异常。

由于 PdfReader 类是对已有文件进行操作，所以在创建 PdfReader 类的实例时，其构造方法的参数必须是存在的 PDF 文档，并且该文档的路径必须是有效的，否则在导入已有文件时会发生异常。

实例 423　导入并添加页码

光盘位置：光盘\MR\423　　中级　　趣味指数：★★★★

实例说明

在 PDF 文档创建以后，如果没有为文档添加页码，可以导入已有文档并为其添加页码。本实例实现了导入已有文档并添加页码的功能，如图 13.48 所示是文档中第 2 页底部显示的页码效果。

图 13.48　导入并添加页码的效果

关键技术

本实例主要是通过 PdfContentByte 类实现对导入的文档添加页码的功能，实现代码如下：

```
PdfContentByte under = null;                                    //定义 PdfContentByte 对象
for (int i=1;i<=totalPages;i++){                                //totalPages 是已有文档的总页数
        under = stamp.getUnderContent(i);                       //获得每一页的内容
        under.beginText();                                      //标记文本开始
        under.setFontAndSize(chinese, 18);                      //设置字体和字号
        under.setTextMatrix(280, 15);                           //设置页码的显示位置
        under.showText("第"+i+"页");                            //添加页脚
        under.endText();                                        //标记文本结束
}
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 ImportAndAddPageNumber 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args){
        Document document = new Document();                     //创建文档对象
        try {
                PdfWriter.getInstance(document, new FileOutputStream("c:\\页码.pdf"));  //关联文档对象与输出流
                document.open();                                //打开文档
                document.add(new Paragraph("No. 1 page"));      //向文档中添加内容
                document.newPage();
                document.add(new Paragraph("No. 2 page"));      //向文档中添加内容
                document.close();                               //关闭文档对象
                PdfReader reader = new PdfReader("c:\\页码.pdf");   //创建“页码.pdf”的 PdfReader 对象
                int totalPages = reader.getNumberOfPages();
                PdfStamper stamp = new PdfStamper(reader, new FileOutputStream("c:\\导入并添加页码.pdf"));   //创建 PdfStamper 对象
                BaseFont chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);   //定义字体
                PdfContentByte under = null;
                for (int i=1;i<=totalPages;i++){
                        under = stamp.getUnderContent(i);       //获得每一页的内容
                        under.beginText();                      //标记文本开始
                        under.setFontAndSize(chinese, 18);      //设置字体和字号
                        under.setTextMatrix(280, 15);           //设置页码的显示位置
                        under.showText("第"+i+"页");            //添加页脚
                        under.endText();                        //标记文本结束
                }
```

```
            stamp.close();              //PdfStamper 对象，将从“页码.pdf”中读取的文档添加页码后写入“添加页码.pdf”
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 423：获得已有文档的总页数。

在对导入的 PDF 文档进行操作时，有时需要获得已有文档的总页数，可使用 PdfReader 类的 getNumberOfPages()方法实现，代码如下：

```
PdfReader reader = new PdfReader("c:\\页码.pdf");          //创建“页码.pdf”的 PdfReader 对象
int totalPages = reader.getNumberOfPages();                //获得已有文档的总页数
```

实例 424 导入并添加水印

光盘位置：光盘\MR\424　　中级　　趣味指数：★★★★★

实例说明

PDF 文档创建后，为了对页面进行修饰，还可以为其添加水印，然后将添加有水印的内容存储为一个新的 PDF 文档。运行程序，可以看到添加水印之前的原文档和导入并添加水印之后新文档的显示效果，如图 13.49 和图 13.50 所示。

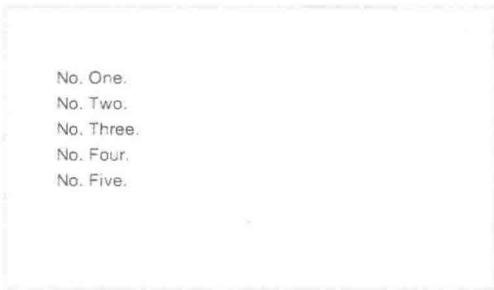


图 13.49　添加水印之前的效果

图 13.50　添加水印之后的效果

关键技术

本实例主要是使用 PdfReader 对象，将已有 PDF 文档的内容与输出流对象关联后创建 PdfStamper 对象，然后使用该对象的 getUnderContent()方法获得要为其添加水印内容的 PdfContentByte 对象，从而实现为 PDF 文档添加水印的功能。

为 PDF 文档添加水印的实现代码如下：

```
PdfReader reader = new PdfReader("c:\\水印.pdf");                                  //创建“水印.pdf”的 PdfReader 对象
PdfStamper stamp = new PdfStamper(reader, new FileOutputStream("c:\\添加水印.pdf")); //创建 PdfStamper 对象
Image img = Image.getInstance("image/watermark.jpg");                              //写上内容
img.setAbsolutePosition(30, 385);                                                  //定位图片对象
PdfContentByte under = stamp.getUnderContent(1);                                   //获得第一页的内容
under.addImage(img);                                                               //添加图片，完成水印功能
stamp.close();                  //PdfStamper 对象，将从“水印.pdf”中读取的文档添加水印后写入“添加水印.pdf”
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 ImportAndAddWatermark 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\水印.pdf"));  //关联文档对象与输出流
        document.open();                                                   //打开文档
        document.add(new Paragraph("No. One."));                           //向文档中添加内容
        document.add(new Paragraph("No. Two."));                           //向文档中添加内容
        document.add(new Paragraph("No. Three."));                         //向文档中添加内容
        document.add(new Paragraph("No. Four."));                          //向文档中添加内容
        document.add(new Paragraph("No. Five."));                          //向文档中添加内容
        document.close();                                                  //关闭文档对象
        PdfReader reader = new PdfReader("c:\\水印.pdf");                  //创建“水印.pdf”的 PdfReader 对象
        PdfStamper stamp = new PdfStamper(reader, new FileOutputStream("c:\\添加水印.pdf"));   //创建 PdfStamper 对象
        Image img = Image.getInstance("image/watermark.jpg");              //写上内容
        img.setAbsolutePosition(30, 385);                                  //定位图片对象
        PdfContentByte under = stamp.getUnderContent(1);                   //获得第 1 页的内容
        under.addImage(img);                                               //添加图片，完成水印功能
        stamp.close();          //PdfStamper 对象，将从“水印.pdf”中读取的文档添加水印后写入“添加水印.pdf”
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 424：添加水印时设置水印图片的旋转角度。

本实例为 PDF 文档添加水印图片，并没有指定旋转角度，如果需要指定水印图片的旋转角度，可以使用 PdfContentByte 对象重载的 addImage(Image image,AffineTransform transform)方法实现，该方法的第二个参数 transform 是使用 rotate()方法指定了图片旋转角度的 AffineTransform 对象。

实例 425　导入并添加新页和内容

光盘位置：光盘\MR\425　　中级　　趣味指数：★★★★

实例说明

本实例实现了导入已有 PDF 文档到新的文档中，并在新的文档中增加新页和内容的功能。运行程序，可以看到新文档中除了包含原文档中的内容，还增加了新页和新的内容，原文档和新文档的内容分别如图 13.51 和图 13.52 所示。

图 13.51　原文档中的内容

图 13.52　导入并添加新内容的效果

关键技术

本实例主要是使用 PdfReader 类和 PdfStamper 类将原文档的内容导入到新文档中，然后使用 PdfContentByte 类在新文档中添加内容和插入新页。其中，添加内容和增加新页的关键代码如下：

```
PdfContentByte cb = stamp.getOverContent(1);                          //获取第 1 页内容
cb.beginText();                                                       //写内容
cb.setFontAndSize(Chinese, 25);                                       //设置字体属性
cb.setTextMatrix(15, 15);                                             //设置矩阵（坐标）
cb.showText("第一页");                                                //矩阵处显示文本
cb.showTextAligned(Element.ALIGN_CENTER, "新增的内容。", 180, 760, 0);  //设置文本对齐、内容、位置和旋转角度
cb.endText();                                                         //内容结束
stamp.insertPage(2, PageSize.A4);                                     //增加新页
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 ImportAddNewPageContent 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    try {
        PdfReader reader = new PdfReader("c:\\创建第一个 PDF 文档.pdf");                    //导入文档
        PdfStamper stamp = new PdfStamper(reader, new FileOutputStream(
                "c:\\导入并添加新页和内容.pdf"));                                          //关联文档与输出流
        BaseFont Chinese = BaseFont.createFont("STSong-Light",
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                                  //定义基础字体
        PdfContentByte cb = stamp.getOverContent(1);                                     //获取第 1 页内容
        cb.beginText();                                                                  //写内容
        cb.setFontAndSize(Chinese, 25);                                                  //设置字体属性
        cb.setTextMatrix(15, 15);                                                        //设置矩阵（坐标）
        cb.showText("第一页");                                                           //矩阵处显示文本
        cb.showTextAligned(Element.ALIGN_CENTER, "新增的内容。", 180, 760, 0);           //设置文本对齐、内容、位置和旋转角度
        cb.endText();                                                                    //内容结束
        stamp.insertPage(2, PageSize.A4);                                                //增加新页
        cb = stamp.getOverContent(2);                                                    //获取第 2 页内容
        cb.beginText();                                                                  //写内容
        cb.setFontAndSize(Chinese, 20);                                                  //设置字体属性
        cb.showTextAligned(Element.ALIGN_LEFT, "在新增的页中添加的内容。", 100, 600, 0);  //设置文本对齐、内容、位置和旋转角度
        cb.endText();                                                                    //内容结束
        stamp.close();                                                                   //关闭
    } catch (IOException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 425：妙用 PdfContentByte 类的 showTextAligned()方法。

使用 PdfContentByte 类的 showTextAligned()方法，可以向新文档中添加文本内容，如果在添加文本内容时需要指定所添加文本的旋转角度，可以将该方法的最后一个参数设置为 0 以外的数值，例如，要使文本旋转 30°，可以使用如下代码实现：

```
cb.showTextAligned(Element.ALIGN_LEFT, "新添加的内容。", 100, 600, 30);   //设置文本对齐、内容、位置和旋转角度为 30°
```

实例 426　拆分 PDF 文档

光盘位置：光盘\MR\426　　中级　　趣味指数：★★★

实例说明

创建完 PDF 文档后，还可以对 PDF 文档进行拆分。本实例实现了拆分 PDF 文档的功能，运行程序，将原来的文档拆分成多个新的子文档，如图 13.53 和图 13.54 所示分别是原文档拆分后生成的两个新文档。

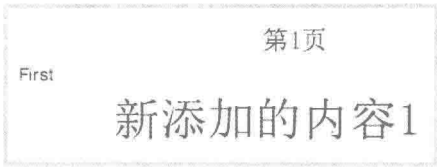


图 13.53　拆分后生成的第 1 个文档

图 13.54　拆分后生成的第 2 个文档

关键技术

本实例主要是使用 PdfReader 类和 PdfCopy 类实现将原文档拆分成多个子文档的功能，其中新文档中的数据通过 PdfReader 对象获得，子文档中的内容通过 PdfCopy 对象的 addPage()方法进行添加。实现拆分 PDF 文档功能的关键代码如下：

```
copy = new PdfCopy(document, new FileOutputStream(fileName));      //创建复本并关联文档与输出流对象
document.open();                                                   //打开文档
copy.addPage(copy.getImportedPage(reader, i + 1));                 //根据获得的页创建新文档
document.close();                                                  //关闭文档
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 SplitPDFDocument 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    String filePathFile = "c:\\原文档.pdf ";                       //需要拆分的原文档
    PdfReader reader = null;                                       //声明 PdfReader 对象
    try {
        reader = new PdfReader(filePathFile);                      //创建 PdfReader 对象
    } catch (IOException e) {
        e.printStackTrace();
    }
    int pageN = reader.getNumberOfPages();                         //获取文件内的页数
    for (int i = 0; i < pageN; i++) {                              //循环向外拆分页
        Document document = new Document(reader
                .getPageSizeWithRotation(i + 1));                  //创建文档，同时获得前面循环的页
        PdfCopy copy = null;
        try {
            int len = filePathFile.length();                       //获得文件完整路径的长度
            String noExt = filePathFile.substring(0, len - 5);     //去除文件扩展名后的路径
            String fileName = noExt + "-" + (i + 1) + ".pdf";      //拆分后生成的文件名称
            copy = new PdfCopy(document, new FileOutputStream(fileName));  //创建复本并关联文档与输出流对象
            document.open();                                       //打开文档
            copy.addPage(copy.getImportedPage(reader, i + 1));     //根据获得的页创建新文档
            document.close();                                      //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
```

```
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

秘笈心法

心法领悟 426：使拆分后生成的子文档中包含多页内容。

本实例生成的新文档中只包含一页内容，如果希望子文档中包含多页内容，可以在关闭文档前多次使用 addPage()方法添加页面内容，每添加一次就有一页内容，如果添加两次就会有两页内容，例如：

```
copy.addPage(copy.getImportedPage(reader, 1));          //根据获得的页创建新文档
copy.addPage(copy.getImportedPage(reader, 2));          //根据获得的页创建新文档
```

实例 427　合并 PDF 文档

光盘位置：光盘\MR\427　　中级　　趣味指数：★★★

实例说明

如果创建了多个 PDF 文档，可以将这些 PDF 文档合并成一个，这样即可通过一个文档查看多个 PDF 文档中的内容。本实例实现了将多个文档合并成一个文档的功能，打开合并后的文档，可以看到第 2 页和第 3 页的内容，分别如图 13.55 和图 13.56 所示。

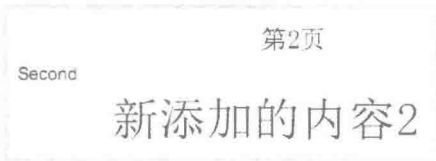


图 13.55　合并后文档第 2 页的内容

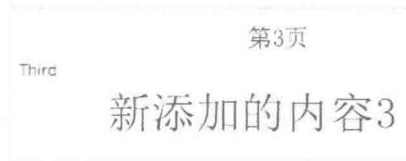


图 13.56　合并后文档第 3 页的内容

关键技术

本实例主要是通过数组对需要合并的 PDF 文档进行存储，然后使用 PdfCopy 类和 for 循环语句实现了将多个 PDF 文档合并成一个大的 PDF 文档。实现合并文档的关键代码如下：

```
for (int i = 0; i < subFiles.length; i++) {                    //循环，获取待合并文件长度
    PdfReader reader = new PdfReader(subFiles[i]);             //读取待合并文件长度
    int totalPages = reader.getNumberOfPages();                //获得每个子文档的总页数
    for (int p = 1; p <= totalPages; p++) {                    //遍历子文档的每一页
        copy.addPage(copy.getImportedPage(reader, p));         //将子文档的每一页添加到新文档中
    }
}
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 UnitePDFDocument 类，向该类中添加 main()方法。主要代码如下：

```
public static void main(String[] args) {
    String[] subFiles = { "c:\\原文档-1.pdf", "c:\\原文档-2.pdf", "c:\\原文档-3.pdf" };    //待合并的 PDF 文档
    String newFile = "c:\\合并结果.pdf";                                                   //合并后的新文档
    Document document = new Document();                                                    //创建文本文档
    try {
        PdfCopy copy = new PdfCopy(document, new FileOutputStream(newFile));               //创建 copy 对象关联文档与输出流
```

```
            document.open();                                          //打开文档
            for (int i = 0; i < subFiles.length; i++) {               //循环，获取待合并文件长度
                PdfReader reader = new PdfReader(subFiles[i]);        //读取待合并文件长度
                int totalPages = reader.getNumberOfPages();           //获得每个子文档的总页数
                for (int p = 1; p <= totalPages; p++) {               //遍历子文档的每一页
                    copy.addPage(copy.getImportedPage(reader, p));    //将子文档的每一页添加到新文档中
                }
            }
            document.close();                                         //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 427：手动选择需要合并的 PDF 文档。

本实例通过程序将指定的多个 PDF 文档合并成一个文档，而在实际应用中，用户并不一定会编写程序代码，为此可以做成通过文件选择对话框选择需要合并的多个 PDF 文档，然后再将所选择的 PDF 文档合并成一个大的 PDF 文档。

实例 428　打印 PDF 文档

光盘位置：光盘\MR\428　　中级　　趣味指数：★★★★

实例说明

在 Java 中，如果希望通过程序打印 PDF 文档，可以使用 PdfBox 工具包实现。本实例实现了打印 PDF 文档的功能，运行程序，将打印如图 13.57 所示的 PDF 文档。

这是一个3行3列的表格

1.1	1.2	1.3
2.1	2.2	2.3
3.1	3.2	3.3

图 13.57　需要打印的 PDF 文档

关键技术

本实例主要是通过 PdfBox 工具包中 PDDocument 类的 load()方法完成装载 PDF 文档并获得 PDDocument 对象，然后通过该对象创建 Printable 对象，并将 Printable 对象与打印作业关联，从而实现了打印 PDF 文档的功能，实现代码如下：

```
PDDocument document = PDDocument.load("创建表格.pdf");    //获取待打印的文档
Printable printable = new PDPageable(document);           //创建 Printable 对象
job.setPrintable(printable);                              //设置打印工作
```

设计过程

（1）创建一个 Java 工程。

（2）向工程中添加一个 PrintPDF 类，向该类中添加 main()方法。主要代码如下：

```
PrinterJob job = PrinterJob.getPrinterJob();                    //创建打印作业
PDDocument document = PDDocument.load("创建表格.pdf");          //获取待打印的文档
Printable printable = new PDPageable(document);                 //创建 Printable 对象
job.setPrintable(printable);                                    //设置打印工作
job.print();                                                    //打印
```

秘笈心法

心法领悟 428：添加应用程序链接注释。

在使用 iText 生成 PDF 文档时，可以根据需要为 PDF 文档添加应用程序链接注释。应用程序链接注释即指为文档添加一个矩形区域，当用户单击该矩形区域时，就会打开应用程序链接注释所链接的 Windows 应用程序。实现应用程序链接注释的代码如下：

```
Annotation annotation = new Annotation(10f, 10f, 500f, 820f,
                "c:/Windows/system32/mspaint.exe", null, null, null);   //创建应用程序链接注释，用于打开 Windows 7 的画图程序
document.add(annotation);                                               //为文档添加应用程序链接注释
```

第14章

绘制 PDF 图形和图像

- 绘制图形
- 绘制图像

14.1 绘 制 图 形

实例 429 在 PDF 文档中绘制直线

光盘位置：光盘\MR\429　　中级　　趣味指数：★★★

实例说明

本实例演示如何通过 Java 应用程序在 PDF 文档中绘制直线。运行程序，将看到在生成的 PDF 文档中绘制有 3 条直线，效果如图 14.1 所示。

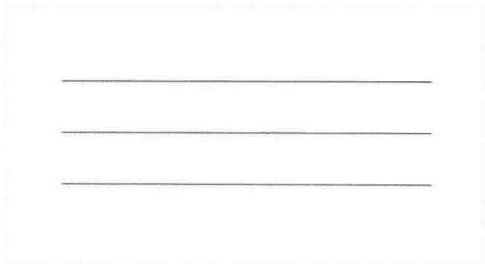


图 14.1 在 PDF 文档中绘制的直线

关键技术

本实例主要是通过 PdfContentByte 类的 moveTo()和 lineTo()方法绘制直线，然后使用 stroke()方法确认直线的绘制。在 PDF 文档中绘制直线的代码如下：

```
cb.moveTo(50, 780);                                    //绘制起点坐标
cb.lineTo(260, 780);                                   //绘制终点坐标
cb.stroke();                                           //确认直线的绘制
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DrawLineDemo 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中绘制直线的功能，主方法的主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                //创建文档对象
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\绘制直线.pdf"));   //关联文档对象与输出流
        document.open();                               //打开文档
        PdfContentByte cb = writer.getDirectContent();  //获取内容
        cb.moveTo(50, 780);                            //绘制起点坐标
        cb.lineTo(260, 780);                           //绘制终点坐标
        cb.stroke();                                   //确认直线的绘制
        //省略了绘制另外两条直线的代码
        document.close();                              //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 429：绘制五子棋的棋盘。

掌握了直线的绘制方法，即可绘制五子棋的棋盘，方法是绘制一些水平距离相等的直线，然后再绘制一些垂直距离相等且与水平直线相交的直线，从而完成五子棋棋盘的绘制。

实例 430　在 PDF 文档中绘制矩形

光盘位置：光盘\MR\430　　中级　　趣味指数：★★★

实例说明

本实例演示如何通过 Java 应用程序在 PDF 文档中绘制矩形。运行程序，将看到在生成的 PDF 文档中绘制了两个矩形，效果如图 14.2 所示。

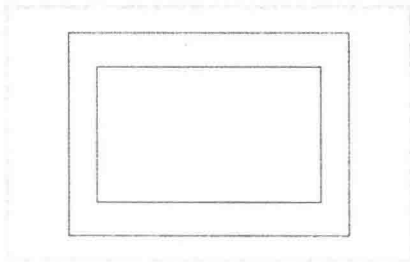


图 14.2　在 PDF 文档中绘制矩形的效果

关键技术

本实例主要是通过 PdfContentByte 类的 rectangle()方法绘制矩形，然后使用 stroke()方法确认矩形的绘制。在 PDF 文档中绘制矩形的代码如下：

```
cb.rectangle(50, 650, 200, 150);                    //绘制矩形
cb.stroke();                                        //确认绘制的矩形
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DrawRectangleDemo 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中绘制矩形的操作，主方法的主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                          //创建文档对象
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\绘制矩形.pdf"));       //关联文档对象与输出流
        document.open();                                         //打开文档
        PdfContentByte cb = writer.getDirectContent();           //获取内容
        cb.rectangle(50, 650, 200, 150);                         //绘制矩形
        cb.stroke();                                             //确认绘制的矩形
        //省略了绘制另一个矩形的代码
        document.close();                                        //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 430：使用 PdfContentByte 类绘制填充矩形。

在使用 PdfContentByte 类的 rectangle()方法绘制矩形后，如果再调用 fill()和 fillStroke()方法，这样绘制的矩形就是有填充色的矩形。例如：

```
cb.rectangle(50, 650, 200, 150);                    //绘制矩形
cb.fill();                                          //使绘制的图形有填充色
cb.fillStroke();                                    //确认绘制的填充矩形
```

实例 431 在 PDF 文档中绘制圆

光盘位置：光盘\MR\431　　中级　　趣味指数：★★★

实例说明

本实例演示如何通过 Java 应用程序在 PDF 文档中绘制圆形。运行程序，将看到在生成的 PDF 文档中绘制有一个大的圆形和一个有填充色的小圆，效果如图 14.3 所示。

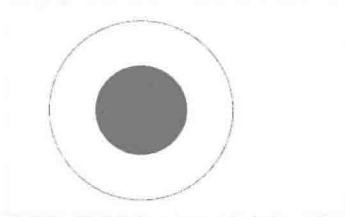


图 14.3　在 PDF 文档中绘制圆形的效果

关键技术

本实例主要是通过 PdfContentByte 类的 circle()方法绘制圆形，然后使用 stroke()方法确认圆形的绘制。在 PDF 文档中绘制圆形的代码如下：

```
cb.circle(120, 720, 80);                            //绘制圆形
cb.stroke();                                        //确认绘制的圆形
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DrawCircleDemo 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中绘制圆形的操作，主方法的主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                 //创建文档对象
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\绘制圆形.pdf"));  //关联文档对象与输出流
        document.open();                                //打开文档
        PdfContentByte cb = writer.getDirectContent();  //获取内容
        cb.circle(120, 720, 80);                        //绘制圆形
        cb.stroke();                                    //确认绘制的圆形
        cb.circle(120, 720, 40);                        //绘制圆形
        cb.fill();                                      //填充圆形
        cb.fillStroke();                                //确认绘制的填充圆形
        document.close();                               //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
```

```
        } catch (DocumentException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 431：改变图形的线条颜色和填充颜色。

在绘制图形时，如果需要改变图形的线条颜色和填充颜色，可以分别使用 setColorStroke()和 setColorFill()方法实现，这样即可根据需要绘制各种填充色和有填充色的图形。

实例 432　使用 Graphics2D 绘制图形

光盘位置：光盘\MR\432　　中级　　趣味指数：★★★

实例说明

在绘制 PDF 文档时，可以通过多种方式在 PDF 文档中绘制图形。本实例将使用 Graphics2D 类在 PDF 文档中绘制图形，效果如图 14.4 所示。

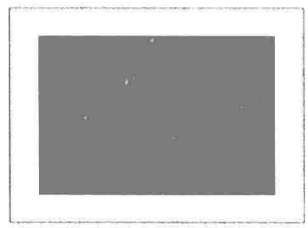


图 14.4　使用 Graphics2D 在 PDF 中绘制的图形

关键技术

本实例主要是通过 PdfContentByte 类的 createGraphics()方法获得指定大小的 Graphics2D 对象，然后使用 Graphics2D 类的方法绘制图形。在 PDF 文档中使用 Graphics2D 类绘制图形的代码如下：

```
Rectangle2D rect1 = new Rectangle2D.Double(50, 50, 200, 150);          //创建矩形对象
g.draw(rect1);                                                         //绘制矩形
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 UseGraphics2D 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中使用 Graphics2D 绘制图形的操作，主方法的代码如下：

```
public static void main(String[] args) throws MalformedURLException {
    Document document = new Document();                                //创建文档对象
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\使用 Graphics2D 绘制图形.pdf"));  //关联文档对象与输出流
        document.open();                                               //打开文档
        PdfContentByte cb = writer.getDirectContent();                 //获取文档内容
        Graphics2D g = cb.createGraphics(850, 850);                    //创建 Graphics 和坐标
        Rectangle2D rect1 = new Rectangle2D.Double(50, 50, 200, 150);  //创建矩形对象
        g.draw(rect1);                                                 //绘制矩形
        Rectangle2D rect2 = new Rectangle2D.Double(70, 70, 160, 110);  //创建矩形对象
        g.fill(rect2);                                                 //绘制填充矩形
```

```
            g.dispose();                                          //部署
            cb.stroke();                                          //确认绘制的图形
            document.close();                                     //关闭文档
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 432：绘制内容更丰富的图形。

使用 Graphics2D 类，可以绘制内容更加丰富的图形，例如调整画笔颜色、粗细等，也可以对图形进行各种运算，例如加运算、减运算、交运算等，从而使 PDF 文档的内容更加丰富多彩。

实例 433　使用 PdfGraphics2D 绘制文本　中级

光盘位置：光盘\MR\433　趣味指数：★★★

实例说明

在绘制 PDF 文档时，除了可以用前面讲的内容绘制图形以外，还可以使用 PdfGraphics2D 类实现文本和图形的绘制。本实例使用 PdfGraphics2D 实现了在 PDF 文档中绘制文本的功能，效果如图 14.5 所示。

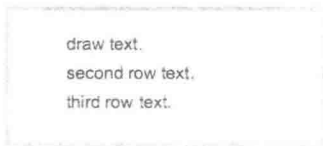


图 14.5　使用 PdfGraphics2D 绘制文本的效果

关键技术

本实例主要是通过 PdfContentByte 类的 createGraphics()方法获得指定大小的 Graphics2D 对象，然后将该 Graphics2D 对象强制转换为 PdfGraphics2D 对象，并通过 PdfGraphics2D 对象的 drawString()方法完成文本的绘制。在 PDF 文档中使用 PdfGraphics2D 类绘制文本的代码如下：

```
PdfGraphics2D g = (PdfGraphics2D) cb.createGraphics(700, 800);    //获得 PdfGraphics2D 对象
g.drawString("draw text. ", 54, 10);                              //绘制文本
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PdfGraphics2DText 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中使用 PdfGraphics2D 绘制文本的操作，主方法的代码如下：

```
public static void main(String[] args) throws MalformedURLException {
    Document document = new Document();                           //创建文档对象
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\使用 PdfGraphics2D 绘制文本.pdf"));  //关联文档对象与输出流
        document.open();                                          //打开文档
        PdfContentByte cb = writer.getDirectContent();            //获取文档内容
        PdfGraphics2D g = (PdfGraphics2D) cb.createGraphics(700, 800);  //获得 PdfGraphics2D 对象
        g.drawString("draw text. ", 54, 10);                      //绘制文本
        g.drawString("second row text. ", 54, 30);                //绘制文本
        g.drawString("third row text. ", 54, 50);                 //绘制文本
```

```
            g.dispose();                                    //部署
            cb.stroke();                                    //确认绘制的内容
            document.close();                               //关闭文档
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 433：影响文本绘制位置的因素。

在使用 PdfGraphics2D 对象绘制文本时，文本的绘制位置会受到 PdfGraphics2D 对象的影响，如果大小不适合，会导致文本绘制位置不正确，这时需要调整一下 PdfGraphics2D 对象的大小。

实例 434　使用 PdfGraphics2D 绘制图形

光盘位置：光盘\MR\434　　中级　　趣味指数：★★★★

实例说明

在绘制 PDF 文档时，还可以使用 PdfGraphics2D 类实现图形的绘制。本实例使用 PdfGraphics2D 实现了在 PDF 文档中绘制图形的功能，效果如图 14.6 所示。

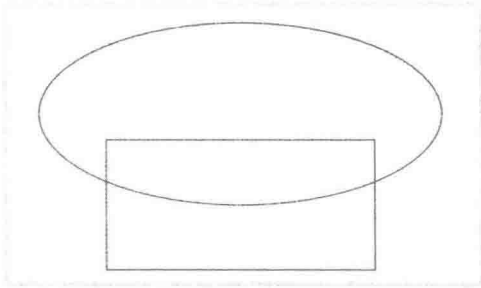


图 14.6　使用 PdfGraphics2D 绘制图形的效果

关键技术

本实例主要是通过 PdfContentByte 类的 createGraphics()方法获得指定大小的 Graphics2D 对象，然后将该 Graphics2D 对象强制转换为 PdfGraphics2D 对象，并通过 PdfGraphics2D 对象的 draw()方法完成图形的绘制。在 PDF 文档中使用 PdfGraphics2D 类绘制图形的代码如下：

```
Rectangle2D rect = new Rectangle2D.Double(120, 100, 200, 100);      //创建矩形对象
Ellipse2D circle = new Ellipse2D.Double();                          //创建椭圆对象
circle.setFrameFromCenter(220, 80, 370, 150);                       //设置椭圆的中心点坐标和角点坐标
g.draw(rect);                                                       //绘制矩形对象
g.draw(circle);                                                     //绘制圆形对象
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PdfGraphics2DShape 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中使用 PdfGraphics2D 绘制图形的操作，主方法的代码如下：

```
public static void main(String[] args) throws MalformedURLException {
    Document document = new Document();                             //创建文档对象
```

```
        try {
            PdfWriter writer = PdfWriter.getInstance(document,
                    new FileOutputStream("c:\\使用 PdfGraphics2D 绘制图形.pdf"));   //关联文档对象与输出流
            document.open();                                                        //打开文档
            PdfContentByte cb = writer.getDirectContent();                          //获取文档内容
            PdfGraphics2D g = (PdfGraphics2D) cb.createGraphics(700, 800);          //创建 Graphics 和坐标
            Rectangle2D rect = new Rectangle2D.Double(120, 100, 200, 100);          //创建矩形对象
            Ellipse2D circle = new Ellipse2D.Double();                              //创建椭圆对象
            circle.setFrameFromCenter(220, 80, 370, 150);                           //设置椭圆的中心点坐标和角点坐标
            g.draw(rect);                                                           //绘制矩形对象
            g.draw(circle);                                                         //绘制圆形对象
            g.dispose();                                                            //部署
            cb.stroke();                                                            //确认绘制的内容
            document.close();                                                       //关闭文档
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 434：使用 PdfGraphics2D 类的方法绘制图形。

本实例通过使用 PdfGraphics2D 类的 draw()方法绘制图形对象，实现了图形的绘制，除此之外，还可以使用 PdfGraphics2D 类的 drawXXX()方法绘制图形。例如，使用 drawOval()方法可以绘制圆、使用 drawRect()方法可以绘制矩形等，从而极大地方便了在 PDF 文档中绘制图形的操作。

实例 435　使用 PdfGraphics2D 绘制有填充色的图形

高级

光盘位置：光盘\MR\435

趣味指数：★★★★

实例说明

在绘制 PDF 文档时，使用 PdfGraphics2D 类除了可以绘制文本和基本图形外，还可以绘制有填充色的图形。本实例演示如何使用 PdfGraphics2D 在 PDF 文档中绘制有填充色的图形，效果如图 14.7 所示。

图 14.7　使用 PdfGraphics2D 绘制有填充色的图形

关键技术

本实例主要是通过 PdfContentByte 类的 createGraphics()方法获得指定大小的 Graphics2D 对象，然后将该 Graphics2D 对象强制转换为 PdfGraphics2D 对象，并通过 PdfGraphics2D 对象的 fill()方法完成有填充色图形的绘制。在 PDF 文档中使用 PdfGraphics2D 类绘制有填充色的图形代码如下：

```
Rectangle2D rect = new Rectangle2D.Double(120, 130, 200, 100);   //创建矩形对象
g.setColor(Color.BLUE);                                          //设置颜色
g.fill(rect);                                                    //绘制填充的矩形
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PdfGraphics2DFillShape 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中使用 PdfGraphics2D 绘制有填充色的图形，主方法的代码如下：

```
public static void main(String[] args) throws MalformedURLException {
    Document document = new Document();                                          //创建文档对象
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\使用 PdfGraphics2D 绘制有填充色的图形.pdf"));   //关联文档对象与输出流
        document.open();                                                         //打开文档
        PdfContentByte cb = writer.getDirectContent();                           //获取文档内容
        PdfGraphics2D g = (PdfGraphics2D) cb.createGraphics(700, 800);           //创建 Graphics 和坐标
        Rectangle2D rect = new Rectangle2D.Double(120, 130, 200, 100);           //创建矩形对象
        Ellipse2D circle = new Ellipse2D.Double();                               //创建椭圆对象
        circle.setFrameFromCenter(220, 80, 370, 120);                            //设置椭圆形的中心点坐标和角点坐标
        g.setColor(Color.BLUE);                                                  //设置颜色
        g.fill(rect);                                                            //绘制填充的矩形
        g.setColor(Color.PINK);                                                  //设置颜色
        g.fill(circle);                                                          //绘制填充的圆形
        g.dispose();                                                             //部署
        cb.stroke();                                                             //确认绘制图形
        document.close();                                                        //关闭文档
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 435：使用 PdfGraphics2D 的方法绘制填充图形。

本实例通过使用 PdfGraphics2D 的 fill()方法绘制图形对象，实现了有填充色的图形的绘制，除此之外，还可以使用 PdfGraphics2D 类的 fillXXX()方法绘制填充图形。例如，使用 fillPolygon()方法可以绘制有填充色的多边形、使用 fillRoundRect()方法可以绘制有填充色的圆角矩形等，从而极大地方便了在 PDF 文档中绘制有填充色的图形的操作。

实例 436　使用 PdfGraphics2D 旋转绘制的图形

高级

光盘位置：光盘\MR\436

趣味指数：★★★★

实例说明

在绘制 PDF 文档时，使用 PdfGraphics2D 类除了可以绘制文本和基本图形外，还可以旋转绘制的图形。本实例演示如何使用 PdfGraphics2D 类在 PDF 文档中旋转绘制的图形，效果如图 14.8 所示。

关键技术

本实例主要是通过 PdfContentByte 类的 createGraphics()方法获得指定大小的 Graphics2D 对象，然后将该 Graphics2D 对象强制转换为 PdfGraphics2D 对象，并通过 PdfGraphics2D 对象的 rotate()方法设置图形的旋转点和旋转角度，从而实现旋转绘制图形的操作。在 PDF 文档中使用 PdfGraphics2D 类旋转绘制图形的代码如下：

```
Rectangle2D rect = new Rectangle2D.Double(200, 200, 150, 200);   //创建矩形对象
g.rotate(20, 380, 150);                                          //旋转图形
g.draw(rect);                                                    //绘制矩形对象
```

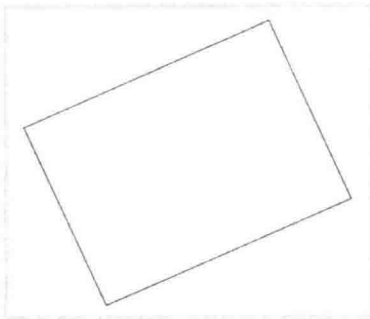


图 14.8　使用 PdfGraphics2D 类旋转图形

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PdfGraphics2DRotateShape 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中使用 PdfGraphics2D 类旋转图形的操作，主方法的代码如下：

```
public static void main(String[] args) throws MalformedURLException {
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\使用 PdfGraphics2D 旋转绘制的图形.pdf"));  //关联文档对象与输出流
        document.open();                                                   //打开文档
        PdfContentByte cb = writer.getDirectContent();                     //获取文档内容
        PdfGraphics2D g = (PdfGraphics2D) cb.createGraphics(700, 800);     //创建 PdfGraphics2D 对象
        Rectangle2D rect = new Rectangle2D.Double(200, 200, 150, 200);     //创建矩形对象
        g.setColor(Color.BLUE);                                            //设置图形颜色
        g.rotate(20, 380, 150);                                            //旋转图形
        g.draw(rect);                                                      //绘制矩形对象
        g.dispose();                                                       //部署
        document.close();                                                  //关闭文档
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 436：解决不能使用 rotate()方法旋转图形的问题。

在使用 PdfGraphics2D 的 rotate()方法旋转绘制的图形时，如果先使用 draw()或 fill()方法绘制图形，然后再使用 rotate()方法设置图形的旋转角度，图形就不会旋转，所以要使图形旋转，必须先使用 rotate()方法，然后再使用 draw()或 fill()方法绘制图形。

实例 437　使用 PdfGraphics2D 缩放绘制的图形

高级

光盘位置：光盘\MR\437

趣味指数：★★★★

实例说明

在绘制 PDF 文档时，使用 PdfGraphics2D 类除了可以绘制文本和基本图形外，还可以缩放绘制的图形。本实例演示如何使用 PdfGraphics2D 在 PDF 文档中缩放绘制的图形，效果如图 14.9 所示。

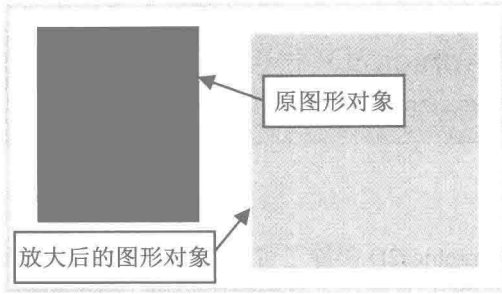


图 14.9　使用 PdfGraphics2D 类缩放图形的效果

关键技术

本实例主要是通过 PdfContentByte 类的 createGraphics()方法获得指定大小的 Graphics2D 对象，然后将该 Graphics2D 对象强制转换为 PdfGraphics2D 对象，并通过 PdfGraphics2D 对象的 scale()方法对图形进行缩放，从而实现缩放绘制图形的操作。在 PDF 文档中使用 PdfGraphics2D 类缩放绘制图形的代码如下：

```
rect = new Rectangle2D.Double(150, 30, 120, 150);                    //创建矩形对象
g.scale(1.4, 1.2f);                                                  //缩放矩形对象
g.setColor(Color.PINK);                                              //设置颜色
g.fill(rect);                                                        //绘制有填充色的图形
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PdfGraphics2DZoomShape 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中使用 PdfGraphics2D 类缩放图形的操作，主方法的代码如下：

```
public static void main(String[] args) throws MalformedURLException {
    Document document = new Document();                              //创建文档对象
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\使用 PdfGraphics2D 缩放绘制的图形.pdf"));   //关联文档对象与输出流
        document.open();                                             //打开文档
        PdfContentByte cb = writer.getDirectContent();               //获取文档内容
        PdfGraphics2D g = (PdfGraphics2D) cb.createGraphics(700, 800);   //创建 Graphics 和坐标
        Rectangle2D rect = new Rectangle2D.Double(50, 30, 120, 150);     //创建原矩形对象
        g.setColor(Color.BLUE);                                      //设置颜色
        g.fill(rect);                                                //绘制有填充色的图形
        rect = new Rectangle2D.Double(150, 30, 120, 150);            //创建与原矩形大小相同的矩形对象
        g.scale(1.4, 1.2f);                                          //缩放矩形对象
        g.setColor(Color.PINK);                                      //设置颜色
        g.fill(rect);                                                //绘制有填充色的图形
        g.dispose();                                                 //部署
        cb.stroke();                                                 //确认绘制内容
        document.close();                                            //关闭文档
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 437：放大和缩小图形时 scale()方法的参数设置。

在使用 PdfGraphics2D 的 scale()方法对图形进行缩放时，其参数大于 1，则放大图形；参数小于 1，则缩小图形，在使用时可以根据需要进行设置。

实例 438 使用 PdfGraphics2D 平移绘制的图形

高级

光盘位置：光盘\MR\438

趣味指数：★★★★

实例说明

在绘制 PDF 文档时，使用 PdfGraphics2D 类除了可以绘制文本和基本图形外，还可以平移绘制的图形。本实例演示如何使用 PdfGraphics2D 在 PDF 文档中平移绘制的图形，效果如图 14.10 所示。

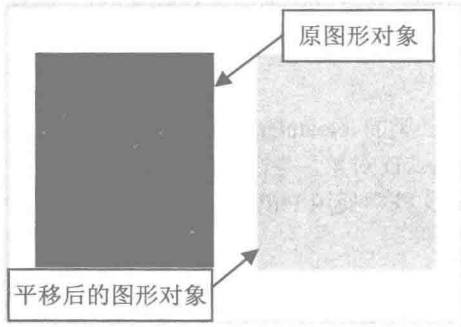


图 14.10 使用 PdfGraphics2D 类平移图形的效果

关键技术

本实例主要是通过 PdfContentByte 类的 createGraphics()方法获得指定大小的 Graphics2D 对象，然后将该 Graphics2D 对象强制转换为 PdfGraphics2D 对象，并通过 PdfGraphics2D 对象的 translate()方法对图形进行平移，从而实现平移绘制图形的操作。在 PDF 文档中使用 PdfGraphics2D 类平移绘制图形的代码如下：

```
Rectangle2D rect = new Rectangle2D.Double(50, 30, 120, 150);          //创建原矩形对象
g.translate(150.0f, 1.0f);                                            //平移矩形对象
g.setColor(Color.PINK);                                               //设置颜色
g.fill(rect);                                                         //绘制有填充色的图形
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PdfGraphics2DTransferShape 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中使用 PdfGraphics2D 类平移绘制的图形，主方法的代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                   //创建文档对象
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\使用 PdfGraphics2D 平移绘制的图形.pdf"));  //关联文档对象与输出流
        document.open();                                                  //打开文档
        PdfContentByte cb = writer.getDirectContent();                    //获取文档内容
        PdfGraphics2D g = (PdfGraphics2D) cb.createGraphics(1200, 800);   //创建 Graphics 和坐标
        Rectangle2D rect = new Rectangle2D.Double(50, 30, 120, 150);      //创建原矩形对象
        g.setColor(Color.BLUE);                                           //设置颜色
        g.fill(rect);                                                     //绘制有填充色的图形
        g.translate(150.0f, 1.0f);                                        //平移矩形对象
        g.setColor(Color.PINK);                                           //设置颜色
        g.fill(rect);                                                     //绘制有填充色的图形
        g.dispose();                                                      //部署
        cb.stroke();                                                      //确认绘制内容
        document.close();                                                 //关闭文档
    } catch (DocumentException e) {
```

```
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 438：控制图形的平移方向。

在使用 PdfGraphics2D 的 translate()方法对图形进行平移时，如果第一个参数大于 0，则向右平移图形，如果第一个参数小于 0，则向左平移图形；同理，如果第二个参数大于 0，则向下平移图形，如果第二个参数小于 0，则向上平移图形。

14.2　绘 制 图 像

实例 439	添加图片	中级
	光盘位置：光盘\MR\439	趣味指数：★★★★

实例说明

在创建 PDF 文档时，除了需要在文档中绘制文本和图形信息外，有时还需要在文档中添加图片。本实例实现了在 PDF 文档中添加图片的功能，效果如图 14.11 所示。

图 14.11　在 PDF 文档中添加图片的效果

关键技术

本实例主要使用 Image 类创建指定图片的图像对象，然后使用 Document 类的 add()方法将图像对象添加到文档中，从而实现了在 PDF 文档中添加图片的功能。在 PDF 文档中添加图片的代码如下：

```
Image image = Image.getInstance("image/picture.jpg");          //创建图像对象
document.add(image);                                           //向文档添加图片
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 AddPicture 类，该类中只有一个 main()主方法，在应用程序启动后执行该方法，完

成在 PDF 文档中添加图片的功能，主方法的代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                          //创建文档对象
    try {
        PdfWriter.getInstance(document,
                new FileOutputStream("c:\\添加图片.pdf"));       //关联文档对象与输出流
        document.open();                                         //打开文档
        Image image = Image.getInstance("image/picture.jpg");    //创建图像对象
        document.add(image);                                     //向文档添加图片
        document.close();                                        //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 439：控制图片在文档中的位置。

当使用 Image 类创建了图像对象以后，可以使用该类的 setAbsolutePosition()方法设置图片在文档中的位置，这样即可根据需要指定图片应该显示的位置。例如：

```
Image image = Image.getInstance("image/picture.jpg");    //创建图像对象
image.setAbsolutePosition(200, 180);                     //设置图片的显示位置
```

实例 440　设置图片对齐方式

光盘位置：光盘\MR\440　　中级　　趣味指数：★★★

实例说明

在 PDF 文档中绘制图片以后，可以根据需要调整图片的对齐方式。本实例实现了在 PDF 文档中调整图片对齐方式的功能，效果如图 14.12 所示。

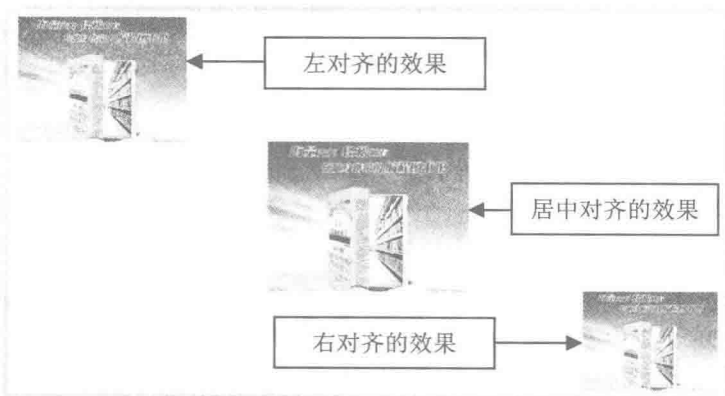


图 14.12　在 PDF 文档中设置图片的对齐方式

关键技术

本实例主要使用 Image 类创建指定图片的图像对象，然后使用 Image 类的 setAlignment()方法设置图片的对

齐方式。例如，下面是在 PDF 文档中设置图片左对齐的代码。

```
Image image = Image.getInstance("image/picture.jpg");          //创建图像对象
image.setAlignment(Image.LEFT);                                //设置图片居左
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PictureAlignment 类，该类中只有一个 main()主方法，在应用程序启动后执行该方法，完成在 PDF 文档中对齐图片的功能，主方法的代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                        //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\设置图片对齐方式.pdf"));                    //关联文档对象与输出流
        document.open();                                       //打开文档
        Image image = Image.getInstance("image/picture.jpg");  //创建图像对象
        image.setAlignment(Image.LEFT);                        //设置图片居左
        image.scalePercent(25);                                //设置原图像的比例
        document.add(image);                                   //向文档添加图片
        image = Image.getInstance("image/picture.jpg");        //创建图像对象
        image.setAlignment(Image.MIDDLE);                      //设置图片居中
        image.scalePercent(30);                                //设置原图像的比例
        document.add(image);                                   //向文档添加图片
        image = Image.getInstance("image/picture.jpg");        //创建图像对象
        image.setAlignment(Image.RIGHT);                       //设置图片居右
        image.scalePercent(20);                                //设置原图像的比例
        document.add(image);                                   //向文档添加图片
        document.close();                                      //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 440：为什么要设置图片对齐方式？

在 PDF 文档中绘制图片以后，图片默认显示在文档的左侧，即默认情况下图片是靠左对齐的，如果需要图片靠右或居中对齐，就需要设置图片的对齐方式。

实例 441　将图片设置为背景

光盘位置：光盘\MR\441　　中级　　趣味指数：★★★★

实例说明

在 PDF 文档中既可以绘制图片，也可以绘制文本，但是有时需要将图片设置为文档的背景。本实例讲解了如何将图片设置为文档的背景，效果如图 14.13 所示。

关键技术

本实例主要使用 Image 类创建指定图片的图像对象，然后使用 Image 类的 setAlignment()方法设置图片的对

齐方式，并将该方法的参数设置为 Image.UNDERLYING，从而实现了将图片设置为文档背景的功能。将图片设置为 PDF 文档背景的代码如下：

```
Image image = Image.getInstance("image/gb.jpg");                    //定义图片对象
image.setAlignment(Image.UNDERLYING);                               //将图片设置为背景
```

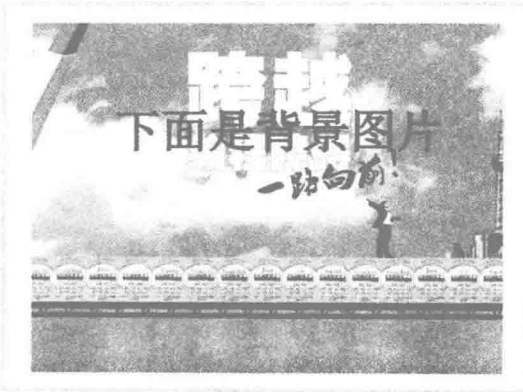


图 14.13　将图片设置为背景的效果

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetPictureBackground 类，该类中只有一个 main()主方法，在应用程序启动后执行该方法，完成在 PDF 文档中添加图片的功能，主方法的代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                             //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\将图片设置为背景.pdf"));                        //关联文档对象与输出流
        document.open();                                            //打开文档
        BaseFont bfChinese = BaseFont.createFont("STSong-Light",
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);             //定义基础字体
        Font FontChinese = new Font(bfChinese, 50, Font.BOLD,
                BaseColor.BLUE);                                    //实例化字体类与设置字体大小属性
        Paragraph p = new Paragraph("下面是背景图片", FontChinese); //创建段落对象
        p.setSpacingBefore(60);                                     //设置段落上边距
        p.setAlignment(com.itextpdf.text.Element.ALIGN_CENTER);     //设置段落居中
        Image image = Image.getInstance("image/gb.jpg");            //定义图片对象
        image.setAlignment(Image.UNDERLYING);                       //将图片设置为背景
        document.add(image);                                        //向文档添加图片
        document.add(p);                                            //添加段落
        document.close();                                           //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 441：设置为背景的同时设置图片的对齐方式。

在 PDF 文档中，可以将图片设置为文档背景，同时还可以设置图片的对齐方式，例如，将图片设置为背景

并居中显示的代码如下：

```
image.setAlignment(Image.UNDERLYING | Image.MIDDLE);          //将图片设置为背景并居中显示
```

实例 442　设置文字环绕

光盘位置：光盘\MR\442　　高级　　趣味指数：★★★★

实例说明

在 PDF 文档中绘制图片时，可以设置图片为文字环绕效果，使文字显示在图片的周围。本实例实现了设置文字环绕的功能，效果如图 14.14 所示。

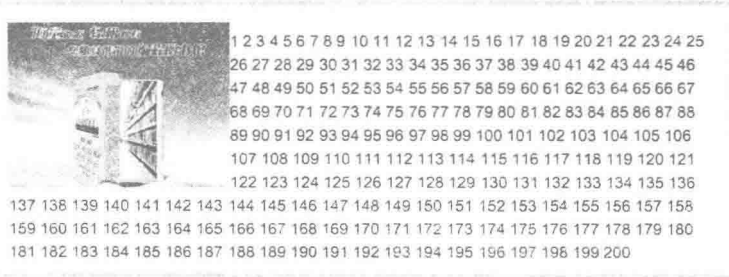


图 14.14　设置文字环绕的效果

关键技术

本实例主要使用 Image 类创建指定图片的图像对象，然后使用 Image 类的 setAlignment()方法设置图片的对齐方式，并将该方法的参数设置为 Image.TEXTWRAP，从而实现了图片的文字环绕功能。在 PDF 文档设置文字环绕的代码如下：

```
Image image = Image.getInstance("image/picture.jpg");          //创建图像对象
image.setAlignment(Image.TEXTWRAP);                            //将图片设置为文字环绕
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetTextWrap 类，该类中只有一个 main()主方法，在应用程序启动后执行该方法，完成在 PDF 文档中设置图片文字环绕的功能，主方法的代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                        //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\设置文字环绕.pdf"));                       //关联文档对象与输出流
        document.open();                                       //打开文档
        Image image = Image.getInstance("image/picture.jpg");  //创建图像对象
        image.scalePercent(33);                                //设置原图像的比例
        image.setAlignment(Image.TEXTWRAP);                    //将图片设置为文字环绕
        document.add(image);                                   //向文档添加图片
        StringBuffer sb = new StringBuffer();                  //创建字符串缓存
        for (int i = 1; i <= 200; i++) {
            sb.append(i + " ");                                //向字符串缓存中添加内容
        }
        Paragraph p = new Paragraph(sb.toString());            //创建段落对象
        document.add(p);                                       //将段落添加到文档中
        document.close();                                      //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
```

```
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 442：设置为文字环绕的同时设置图片的对齐方式。

在 PDF 文档中，可以为图片设置文字环绕效果，同时还可以设置图片的对齐方式。例如，将图片设置为文字环绕并居右显示的代码如下：

```
image.setAlignment(Image.RIGHT | Image.TEXTWRAP);          //设置图片为文字环绕并居右显示
```

实例 443　设置图片大小

光盘位置：光盘\MR\443　　高级　　趣味指数：★★★★

实例说明

在 PDF 文档中绘制图片时，可以对图片的大小进行调整。本实例实现了在 PDF 文档中设置图片大小的功能，将原来较大的图片缩小后的效果如图 14.15 所示。

图 14.15　图片缩小后的效果

关键技术

本实例主要使用 Image 类创建指定图片的图像对象，然后使用 Image 类的 scaleAbsolute()方法设置图片新的宽度和高度，从而实现了设置图片大小的功能。设置 PDF 文档中图片大小的代码如下：

```
Image image = Image.getInstance("image/picture.jpg");          //创建图像对象
image.scaleAbsolute(180, 120);                                 //设置图片新的宽度和高度
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetPictureSize 类，该类中只有一个 main()主方法，在应用程序启动后执行该方法，完成设置 PDF 文档中图片大小的功能，主方法的代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                        //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
```

```
                "c:\\设置图片大小.pdf"));                    //关联文档对象与输出流
            document.open();                                  //打开文档
            Image image = Image.getInstance("image/picture.jpg");  //创建图像对象
            image.setAlignment(Image.MIDDLE);                 //居中显示图片
            image.scaleAbsolute(180, 120);                    //设置图片新的宽度和高度
            document.add(image);                              //向文档添加图片
            document.close();                                 //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 443：设置图片上间距和下间距。

当在 PDF 文档中添加图片后，如果希望图片与上、下内容有一定的间距，可以使用 Image 类的 setSpacingBefore()和 setSpacingAfter()方法进行设置。例如，设置图片与上面内容的间距，实现代码如下：

```
Image image = Image.getInstance("image/picture.jpg");   //创建图像对象
image.setSpacingBefore(10.0f);                          //设置图片与上面内容的间距
```

实例 444　调整图片比例

光盘位置：光盘\MR\444　　高级　　趣味指数：★★★★

实例说明

在 PDF 文档中绘制图片时，可以根据需要调整图片的比例。本实例实现了在 PDF 文档中调整图片比例的功能，将原来较大的图片比例缩小后的效果如图 14.16 所示。

图 14.16　图片比例缩小后的效果

关键技术

本实例主要使用 Image 类创建指定图片的图像对象，然后使用 Image 类中有一个参数的重载方法 scalePercent()调整图片的比例。调整 PDF 文档中图片比例的代码如下：

```
Image image = Image.getInstance("image/picture.jpg");   //创建图像对象
image.scalePercent(30);                                 //调整图片的比例，使其大小为原图片的 30%
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetPictureScale 类，该类中只有一个 main()主方法，在应用程序启动后执行该方法，实现调整 PDF 文档中图片比例的功能，主方法的代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\调整图片比例.pdf"));    //关联文档对象与输出流
        document.open();                                                   //打开文档
        Image image = Image.getInstance("image/picture.jpg");              //创建图像对象
        image.scalePercent(30);                                            //调整图片的比例，使其大小为原图片的 30%
        document.add(image);                                               //向文档添加图片
        document.close();                                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 444：通过调整图片比例放大图片。

本实例通过调整图片比例将图片缩小为原图片的30%，如果需要放大图片，只需要将Image类的scalePercent()方法的参数设置为大于 100 的值即可，因为该方法是以百分比调整图片的缩放比例的。

实例 445　设置高度和宽度的比例

中级

光盘位置：光盘\MR\445　　趣味指数：★★★★

实例说明

在 PDF 文档中绘制图片时，有时需要对图片的高度和宽度比例进行设置。本实例演示如何在 PDF 文档中对图片高度和宽度的比例进行调整，将原来较大的图片调整高度和宽度比例缩小后的效果如图 14.17 所示。

图 14.17　调整图片高度和宽度比例缩小后的效果

关键技术

本实例主要使用 Image 类创建指定图片的图像对象，然后使用 Image 类中有两个参数的重载方法 scalePercent() 调整图片高度和宽度的比例。调整 PDF 文档中图片高度和宽度比例的代码如下：

```
Image image = Image.getInstance("image/picture.jpg");          //创建图像对象
image.scalePercent(50, 40);                                    //设置宽度和高度比例分别为原图片的 50%和 40%
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetHeightWidth 类，该类中只有一个 main()主方法，在应用程序启动后执行该方法，完成设置 PDF 文档中图片高度和宽度比例的功能，主方法的代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                        //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\设置高度和宽度的比例.pdf"));                  //关联文档对象与输出流
        document.open();                                       //打开文档
        Image image = Image.getInstance("image/picture.jpg");  //创建图像对象
        image.scalePercent(50, 40);                            //设置宽度和高度比例分别为原图片的 50%和 40%
        document.add(image);                                   //向文档添加图片
        document.close();                                      //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 445：任意调整图片在某一方向上的缩放比例。

本实例将图片的宽度和高度分别调整为原图片的 50%和 60%，也就是将宽度和高度缩小了，如果需要将图片的宽度变大、高度变小，则可以将 Image 类 scalePercent()方法的第一个参数设置为大于 100 的值，第二个参数设置为小于 100 的值，反之亦然。如果希望宽度不变，可以将第一个参数设置为 100；如果希望高度不变，则可以将第二个参数设置为 100。

实例 446　旋转图片

实例位置：光盘\MR\446　　高级　　趣味指数：★★★★

实例说明

在 PDF 文档中绘制图片时，有时需要对图片进行旋转，使其以一定的旋转角度显示。本实例演示如何在 PDF 文档中调整图片的旋转角度，效果如图 14.18 所示。

关键技术

本实例主要使用 Image 类创建指定图片的图像对象，然后使用 Image 类的 setRotation()方法设置图片的旋转角度。调整图片旋转角度的代码如下：

```
Image image = Image.getInstance("image/picture.jpg");          //创建图像对象
image.setRotation(320);                                        //设置旋转弧度
```

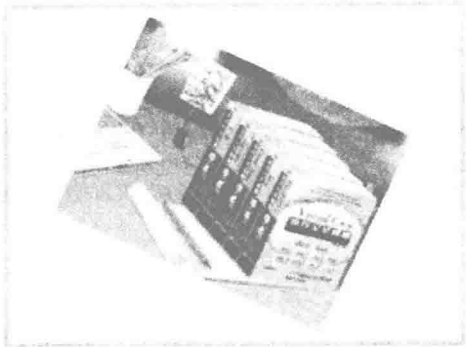


图 14.18　旋转图片的效果

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PictureRotate 类，该类中只有一个 main()主方法，在应用程序启动后执行该方法，完成调整 PDF 文档中图片旋转角度的功能，主方法的代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                          //创建文档对象
    try {
        PdfWriter.getInstance(document,
                new FileOutputStream("c:\\旋转图片.pdf"));       //关联文档对象与输出流
        document.open();                                         //打开文档
        Image image = Image.getInstance("image/picture.jpg");    //创建图像对象
        image.setRotation(320);                                  //设置旋转弧度
        document.add(image);                                     //向文档添加图片
        document.close();                                        //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 446：注意 setRotation()方法的参数。

使用 Image 类的 setRotation()方法，可以设置图片的旋转角度，该方法的参数类型为 float 型，所以在使用时如果为其传递的弧度值是小数，则需要在小数值后加 f 或 F，否则程序会出错，这是因为在 Java 中，小数值默认为 double 型。

实例 447　使用 PdfGraphics2D 绘制图片

高级

光盘位置：光盘\MR\447

趣味指数：★★★★

实例说明

在实例 439 至实例 446 的讲解中，都是使用 Document 类的 add()方法在 PDF 文档中添加图片。本实例演示如何使用 PdfGraphics2D 类向 PDF 文档中添加图片，效果如图 14.19 所示。

图 14.19　使用 PdfGraphics2D 类添加图片

关键技术

本实例主要使用 PdfGraphics2D 类的 drawImage()方法，完成在 PDF 文档中添加图片的功能。使用 PdfGraphics2D 类添加图片的代码如下：

```
PdfGraphics2D g = (PdfGraphics2D) cb.createGraphics(700, 800);                    //创建 PdfGraphics2D 对象
BufferedImage image = ImageIO.read(new File("image/picture.jpg"));                //获取图片
g.drawImage(image, 50, 10, null);                                                 //绘制图片
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PdfGraphics2DPicture 类，该类中只有一个 main()主方法，在应用程序启动后执行该方法，完成使用 PdfGraphics2D 类在 PDF 文档中添加图片的功能，主方法的代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                           //打开文档
    try {
        PdfWriter writer = PdfWriter.getInstance(document,
                new FileOutputStream("c:\\使用 PdfGraphics2D 绘制图片.pdf"));     //关联文档与输出流
        document.open();                                                          //打开文档
        PdfContentByte cb = writer.getDirectContent();                            //获取文档内容
        PdfGraphics2D g = (PdfGraphics2D) cb.createGraphics(700, 800);            //创建 PdfGraphics2D 对象
        BufferedImage image = ImageIO.read(new File("image/picture.jpg"));        //获取图片
        g.drawImage(image, 50, 10, null);                                         //绘制图片
        g.dispose();                                                              //部署
        cb.stroke();                                                              //确认绘制的内容
        document.close();                                                         //关闭文档
    } catch (IOException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 447：将 Image 类创建的图片添加到 PDF 文档中。

本实例使用 BufferedImage 类创建了缓冲图像对象，并将其绘制到 PDF 文档中，实际上，如果有一个 java.awt.Image 类的实例，也可以使用 PdfGraphics2D 类的 drawImage()方法直接将其绘制到 PDF 文档中。例如：

```
PdfGraphics2D g = (PdfGraphics2D) cb.createGraphics(700, 800);                    //创建 PdfGraphics2D 对象
Image image = ImageIO.read(new File("image/picture.jpg"));                        //获取图片
g.drawImage(image, 50, 10, null);                                                 //绘制图片
```

第15章

绘制 PDF 表格

- Table 表格
- PdfPTable 表格

15.1　Table 表格

实例 448	创建具有指定列数的表格 光盘位置：光盘\MR\448	中级 趣味指数：★★★

实例说明

本实例演示如何通过 Java 应用程序，在 PDF 文档中使用 Table 类绘制具有指定列数的表格。运行程序，将创建一个 5 列的表格，效果如图 15.1 所示。

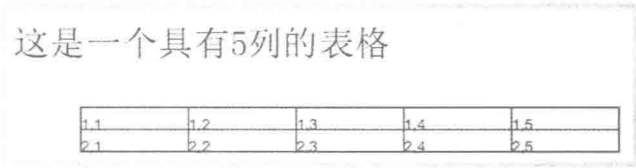


图 15.1　使用 Table 创建具有指定列数的表格

注意：Table 类是 iText 早期版本中创建表格的类，在新版本的 iText 包中已经将 Table 类合并到 PdfPTable 中了，这里之所以对 Table 类进行讲解，完全是为使用 Table 类创建表格的用户提供方便，对于刚刚学习使用 iText 进行 PDF 表格开发的读者，可以跳过本节内容直接学习后面的 PdfPTable 类。

说明：从图 15.1 可以看出，表格中的内容与下边线重叠相交，该问题可以通过添加单元格填距来解决，有关添加单元格填距的内容将在后面的实例中进行讲解。下面的实例与此相同，不再重复说明。

关键技术

本实例主要是通过 Table 类的构造方法 Table(int col)实现了创建具有指定列数的表格。其实现代码如下：

```
Table table = new Table(5);                          //创建一个 5 列的表格
table.addCell("1,1");                                //将单元格顺次加入到表格，当一行充满时自动换行
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 CreateTableColumn 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中绘制具有指定列数表格的操作。主方法的主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                          //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\创建具有指定列数的表格.pdf"));               //关联文档对象与输出流
        document.open();                                         //打开文档
        BaseFont bfChinese = BaseFont.createFont("STSong-Light",
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);          //定义基本字体
        Font fontChinese = new Font(bfChinese, 28, Font.NORMAL); //实例化字体
        document.add(new com.lowagie.text.Paragraph("这是一个具有 5 列的表格",
                fontChinese));                                   //向文档中添加内容
        Table table = new Table(5);                              //创建一个 5 列的表格
        table.addCell("1,1");                                    //将单元格顺次加入到表格，当一行充满时自动换行
        table.addCell("1,2");                                    //将单元格顺次加入到表格，当一行充满时自动换行
        //省略了添加单元格内容的部分代码
        document.add(table);                                     //将表格添加到文档中
        document.close();                                        //关闭文档
```

```
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 448：使表格的单元格显示中文。

在使用 Table 类的 addCell()方法向表格中添加内容时，如果传递的字符串参数是汉字，就不会在单元格中显示，这时可以使用 Phrase 对象对汉字进行中文处理后再添加到表格的单元格中，例如：

```
BaseFont bfChinese = BaseFont.createFont("STSong-Light","UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);    //定义基本字体
Font fontChinese = new Font(bfChinese, 28, Font.NORMAL);                                            //实例化字体
table.addCell(new Phrase("中文",fontChinese));                                                      //添加单元格内容
```

实例 449　创建具有指定行列数的表格

光盘位置：光盘\MR\449　　中级　　趣味指数：★★★

实例说明

本实例演示如何通过 Java 应用程序在 PDF 文档中使用 Table 类绘制具有指定行列数的表格。运行程序，将创建一个 3 行 3 列的表格，效果如图 15.2 所示。

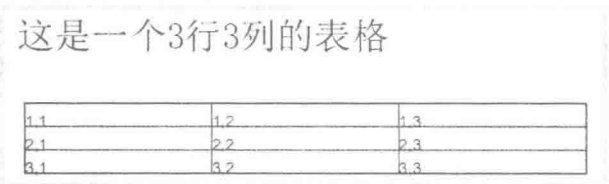


图 15.2　使用 Table 创建具有指定行列数的表格

关键技术

本实例主要是通过 Table 类的构造方法 Table(int col, int row)实现了创建具有指定行列数的表格。其实现代码如下：

```
Table table = new Table(3, 3);                    //创建一个 3 行 3 列的表格
table.addCell("1,1");                             //将单元格顺次加入到表格，当一行充满时自动换行
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 CreateTableRowAndColumn 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中绘制具有指定行列数表格的操作。主方法的主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                          //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\创建具有指定行列数的表格.pdf"));             //关联文档对象与输出流
        document.open();                                         //打开文档
        BaseFont bfChinese = BaseFont.createFont("STSong-Light",
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);          //定义基本字体
        Font fontChinese = new Font(bfChinese, 28, Font.NORMAL); //实例化字体
```

```
            document.add(new Paragraph("          这是一个 3 行 3 列的表格", fontChinese));
            Table table = new Table(3, 3);                    //创建一个 3 行 3 列的表格
            table.addCell("1,1");                             //将单元格顺次加入到表格，当一行充满时自动换行
            //省略了添加单元格内容的其他代码
            document.add(table);                              //将表格添加到文档
            document.close();                                 //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 449：添加单元格内容后表格自动增加行数。

在创建指定行列数的表格后，如果表格中添加的行超出了指定的行数，表格会自动增加行数，而无须再重新设置表格的行数，即表格的行数会随着添加内容的增多自动增加。例如，本实例实现了 3 行 3 列的表格，如果再向表格中添加单元格内容，表格的行数就会增加。

实例 450　设置表格的边框宽度

光盘位置：光盘\MR\450　　中级　　趣味指数：★★★

实例说明

在 PDF 中创建表格时，可能需要调整表格边框的宽度，本实例实现了调整表格边框宽度的功能。运行程序，可以看到调整表格边框宽度的效果，如图 15.3 所示。

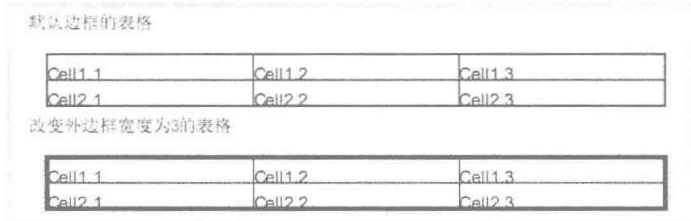


图 15.3　调整表格边框宽度的效果

关键技术

本实例主要是通过 Table 类的 setBorderWidth()方法对表格的外边框宽度进行调整。其实现代码如下：

```
Table table = new Table(3);                           //定义表格
table.setBorderWidth(3);                              //设置表格的外边框宽度
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetTableBoderWidth 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中调整表格外边框宽度的操作。主方法的主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                   //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
```

```
                "c:\\设置表格的边框宽度.pdf"));                                    //关联文档对象与输出流
            document.open();                                                      //打开文档
            BaseFont bfChinese = BaseFont.createFont("STSong-Light",
                    "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                       //定义基本字体
            Font fontChinese = new Font(bfChinese, 12, Font.NORMAL);              //实例化字体
            Table table1 = new Table(3);                                          //定义表格
            document.add(new Paragraph("默认边框的表格", fontChinese));            //创建段落
            table1.addCell("Cell1.1");                                            //将单元格顺次加入到表格，当一行充满时自动换行
            //省略了部分向表格中添加单元格内容的代码
            Table table2 = new Table(3);                                          //定义表格
            table2.setBorderWidth(3);                                             //设置表格的外边框宽度
            table2.addCell("Cell1.1");                                            //将单元格顺次加入到表格，当一行充满时自动换行
            //省略了部分向表格中添加单元格内容的代码
            document.add(table1);                                                 //将表格添加到文档
            document.add(new Paragraph("改变外边框宽度为 3 的表格", fontChinese));  //创建段落
            document.add(table2);                                                 //将表格添加到文档
            document.close();                                                     //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 450：选择设置表格某条边框的宽度。

在实际的 PDF 文档中，还可以根据需要选择设置表格某条边框的宽度，可以使用 Table 类的 setBorderWidthXxx()方法实现。例如，要设置表格下边框的宽度，可以使用如下代码：

```
Table table = new Table(3);                                                       //定义表格
table.setBorderWidthBottom(3);                                                    //设置表格的下边框宽度
```

实例 451　设置表格的边框颜色

光盘位置：光盘\MR\451　　中级　　趣味指数：★★★★

实例说明

在 PDF 中创建表格时，可能需要调整表格边框的颜色，本实例实现了调整表格边框颜色的功能。运行程序，可以看到调整表格边框颜色的效果，如图 15.4 所示。

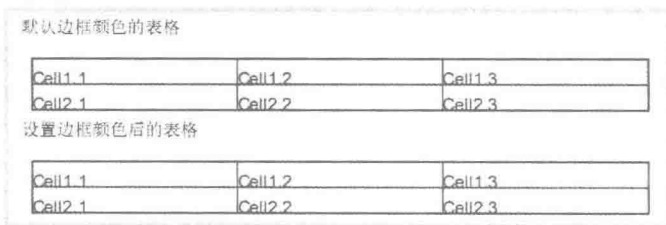


图 15.4　设置表格边框颜色的效果

关键技术

本实例主要是通过 Table 类的 setBorderColor()方法对表格的外边框颜色进行调整。其实现代码如下：

```
Table table = new Table(3);                                    //定义表格
table.setBorderColor(Color.BLUE);                              //设置表格边框颜色
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetTableBorderColor 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中调整表格外边框颜色的操作。主方法的主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                        //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream(
                "c:\\设置表格的边框颜色.pdf"));                  //关联文档对象与输出流
        document.open();                                       //打开文档
        BaseFont bfChinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);  //定义基础字体
        Font fontChinese = new Font(bfChinese, 12, Font.NORMAL);   //实例化字体
        Table table = new Table(3);                            //定义表格
        document.add(new Paragraph("默认边框颜色的表格", fontChinese));  //向文档添加内容
        table.addCell("Cell1.1");                              //将单元格顺次加入到表格，当一行充满时自动换行
        //省略了部分向表格中添加单元格内容的代码
        Table table1 = new Table(3);                           //定义表格
        table1.setBorderColor(Color.BLUE);                     //设置表格边框颜色
        table1.addCell("Cell1.1");                             //将单元格顺次加入到表格，当一行充满时自动换行
        //省略了部分向表格中添加单元格内容的代码
        document.add(table);                                   //将表格添加到文档
        document.add(new Paragraph("设置边框颜色后的表格", fontChinese)); //向文档添加内容
        document.add(table1);                                  //将表格添加到文档
        document.close();                                      //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 451：选择设置表格某条边框的颜色。

在实际的 PDF 文档中，还可以根据需要选择设置表格某条边框的颜色，可以先使用 Table 类的 setBorder()方法指定对指定边框进行设置，然后再使用 Table 类的 setBorderColor()方法设置颜色。例如，要设置表格上边框和下边框的颜色，可以使用如下代码：

```
Table table = new Table(3);                                    //定义表格
table.setBorder(Table.TOP | Table.BOTTOM);                     //对表格上边框和下边框进行设置
table.setBorderColor(Color.BLUE);                              //设置表格边框颜色
```

实例 452　设置单元格间距

光盘位置：光盘\MR\452　　中级　　趣味指数：★★★

实例说明

在 PDF 中创建的表格，默认单元格是没有间距的，如果希望单元格之间有间距，可以对单元格的间距进行设置。本实例实现了设置表格中单元格间距的功能。运行程序，可以看到设置表格单元格间距后的效果，如图 15.5 所示。

图 15.5　设置表格单元格间距的效果

关键技术

本实例主要是通过 Table 类的 setSpacing()方法对表格中单元格的间距进行调整，其实现代码如下：

```
Table table = new Table(3);                                    //定义表格
table.setSpacing(3);                                           //设置表格边框与单元格的间距
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetCellSpace 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中调整表格中单元格间距的操作。主方法的主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                                    //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\设置单元格的间距.pdf"));        //关联文档对象与输出流
        document.open();                                                   //打开文档
        BaseFont bfChinese = BaseFont.createFont("STSong-Light",
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                    //定义基础字体
        Font fontChinese = new Font(bfChinese, 12, Font.NORMAL);           //实例化字体
        Table table = new Table(3);                                        //定义表格
        document.add(new Paragraph("默认的表格", fontChinese));             //向文档添加内容
        table.addCell("Cell1.1");                                          //将单元格顺次加入到表格，当一行充满时自动换行
        //省略了部分向表格中添加单元格内容的代码
        Table table1 = new Table(3);                                       //定义表格
        table1.setSpacing(3);                                              //设置表格边框与单元格的间距
        table1.setBorderColor(Color.GREEN);                                //设置表格边框颜色
        table1.addCell("Cell1.1");                                         //将单元格顺次加入到表格，当一行充满时自动换行
        //省略了部分向表格中添加单元格内容的代码
        document.add(table);                                               //将表格添加到文档
        document.add(new Paragraph("设置表格边框颜色并设置单元格的间距值为 3",fontChinese));    //向文档添加内容
        document.add(table1);                                              //将表格添加到文档
        document.close();                                                  //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 452：设置单元格以页优先方式显示。

在实际的 PDF 文档中，除了可以调整单元格的间距外，还可以设置单元格是否以页优先方式显示。以页优先方式显示即指当单元格内容较多时，使该单元自动到下一页显示，而不与其他单元格在一页显示。设置单元格以页优先方式显示的代码如下：

```
Table table1 = new Table(3);                                   //定义表格
table1.setCellsFitPage(true);                                  //设置单元格以页优先的方式显示
```

实例 453	设置单元格填距 光盘位置：光盘\MR\453	中级 趣味指数：★★★

实例说明

在 PDF 中创建的表格，默认情况下在单元格中添加的内容与边界是没有间距的，因而出现了内容与边框重叠相交的情况。本实例实现了设置表格中单元格填距的功能。运行程序，可以看到设置表格中单元格填距后的效果，如图 15.6 所示，这时单元格中的内容与单元格的边框不再重叠相交。

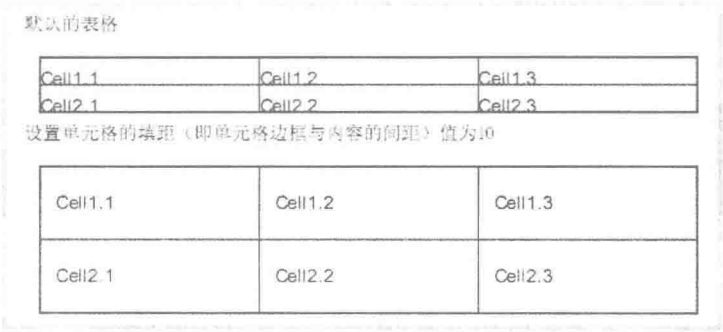


图 15.6　设置表格中单元格填距的效果

关键技术

本实例主要是通过 Table 类的 setPadding()方法对表格中单元格的填距进行调整。其实现代码如下：

```
Table table1 = new Table(3);                          //定义表格
table1.setPadding(10f);                               //设置表格边框与单元格的填距
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetCellPadding 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中调整表格中单元格填距的操作。主方法的主要代码如下：

```
public static void main(String[] args) {
    Document document = new Document();                          //创建文档对象
    try {
        PdfWriter.getInstance(document, new FileOutputStream("c:\\设置单元格填距.pdf"));      //关联文档对象与输出流
        document.open();                                         //打开文档
        BaseFont bfChinese = BaseFont.createFont("STSong-Light",
                "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);          //定义基础字体
        Font fontChinese = new Font(bfChinese, 12, Font.NORMAL); //实例化字体
        Table table = new Table(3);                              //定义表格
        document.add(new Paragraph("默认的表格", fontChinese));   //向文档添加内容
        table.addCell("Cell1.1");                                //将单元格顺次加入到表格，当一行充满时自动换行
        //省略了部分向表格中添加单元格内容的代码
        Table table1 = new Table(3);                             //定义表格
        table1.setPadding(10f);                                  //设置表格边框与单元格的填距
        table1.addCell("Cell1.1");                               //将单元格顺次加入到表格，当一行充满时自动换行
        //省略了部分向表格中添加单元格内容的代码
        document.add(table);                                     //将表格添加到文档
        document.add(new Paragraph("设置单元格的填距值为 10", fontChinese));      //向文档添加内容
        document.add(table1);                                    //将表格添加到文档
        document.close();                                        //关闭文档
```

```
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

秘笈心法

心法领悟 453：设置表格的单元格以灰度显示。

在 PDF 文档中绘制的表格，其单元格默认都是白色底色，也可以将表格的单元格设置为灰度显示，代码如下：

```
Table table = new Table(3);                          //定义表格
table.getDefaultCell().setGrayFill(0.8f);            //设置单元格灰度显示
```

实例 454　设置表格的表头

光盘位置：光盘\MR\454　　中级　　趣味指数：★★★

实例说明

在 PDF 文档中创建表格时，为了区分表头和表格内容，可以为表格设置表头。本实例实现了为表格设置表头的功能。运行程序，将在 PDF 文档中添加一个有表头的表格，效果如图 15.7 所示。

编号	姓名	年龄	电话
95**01	李×辉	30	0431-2222****
95**02	李×辉	30	0431-2222****
95**03	李×辉	30	0431-2222****

图 15.7　为表格设置表头的效果

关键技术

本实例主要是通过 Cell 类创建单元格对象，然后使用 setHeader()方法将单元格设置为表格的表头，代码如下：

```
Table table1 = new Table(4);                         //定义表格
cell3.setHeader(true);                               //将单元格设置为表头
table.addCell(cell0);                                //向表格添加单元格
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetTableHeader 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中添加含有表头的表格的操作，主要代码如下：

```
Document document = new Document();                  //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置表格的表头.pdf"));                   //关联文档对象与输出流
document.open();                                     //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);      //定义基础字体
Font FontChinese = new Font(Chinese, 12, Font.NORMAL);  //实例化字体
Table table = new Table(4);                          //定义表格
```

```
cell0 = new Cell(new Paragraph("编号", FontChinese));                    //创建单元格
cell0.setHorizontalAlignment(Element.ALIGN_CENTER);                      //设置横向居中对齐
cell0.setVerticalAlignment(Element.ALIGN_CENTER);                        //设置垂直居中对齐
cell0.setBackgroundColor(Color.GRAY);                                    //设置背景颜色
cell0.setHeader(true);                                                   //将单元格设置为表头
cell1 = new Cell(new Paragraph("姓名", FontChinese));                    //创建单元格
cell1.setHorizontalAlignment(Element.ALIGN_CENTER);                      //设置横向居中对齐
cell1.setVerticalAlignment(Element.ALIGN_CENTER);                        //设置垂直居中对齐
cell1.setBackgroundColor(Color.GRAY);                                    //设置背景颜色
cell1.setHeader(true);                                                   //将单元格设置为表头
cell2 = new Cell(new Paragraph("年龄", FontChinese));                    //创建单元格
cell2.setHorizontalAlignment(Element.ALIGN_CENTER);                      //设置横向居中对齐
cell2.setVerticalAlignment(Element.ALIGN_CENTER);                        //设置垂直居中对齐
cell2.setBackgroundColor(Color.GRAY);                                    //设置背景颜色
cell2.setHeader(true);                                                   //将单元格设置为表头
cell3 = new Cell(new Paragraph("电话", FontChinese));                    //创建单元格
cell3.setHorizontalAlignment(Element.ALIGN_CENTER);                      //设置横向居中对齐
cell3.setVerticalAlignment(Element.ALIGN_CENTER);                        //设置垂直居中对齐
cell3.setBackgroundColor(Color.GRAY);                                    //设置背景颜色
cell3.setHeader(true);                                                   //将单元格设置为表头
//向表格添加单元格
table.addCell(cell0);
table.addCell(cell1);
table.addCell(cell2);
table.addCell(cell3);
table.setPadding(4);                                                     //设置内容与单元格间距
for (int i = 1; i <= 3; i++) {                                           //向表格的单元格添加内容
    table.addCell(new Paragraph("95**0" + i));
    table.addCell(new Paragraph("李*辉", FontChinese));
    table.addCell(new Paragraph("30"));
    table.addCell(new Paragraph("0431-2222****"));
}
document.add(table);                                                     //将表格添加到文档
document.close();                                                        //关闭文档
```

秘笈心法

心法领悟 454：使表格在每一页都显示表头。

本实例为表格添加了表头，但是当表格的内容较多，无法在一页完全显示时，就会自动显示到下一页，这时下一页是不会显示表头的，为了使表格在每一页都显示表头，可以使用 setHeader()和 endHeaders()方法实现，代码如下：

```
Table table1 = new Table(4);                                             //定义表格
cell3.setHeader(true);                                                   //将单元格设置为表头
table.addCell(cell0);                                                    //向表格添加单元格
//省略了向表格中添加单元格的其他代码
table.endHeaders();                                                      //使每一页显示表头
```

实例 455　设置单元格所占的列数

光盘位置：光盘\MR\455　　中级　　趣味指数：★★★

实例说明

在 PDF 文档中创建表格时，有时需要将一行中的多个单元格合并成一个，这时可以通过设置单元格所占的列数来实现。本实例实现了设置单元格所占列数的功能。运行程序，将在 PDF 文档中显示设置单元格所占列数的效果，如图 15.8 所示。

图 15.8　设置表格中单元格所占列数的效果

关键技术

本实例主要是通过 Cell 类创建单元格对象，然后使用 setColspan()方法设置单元格所占的列数，从而实现合并表格一行中的多个单元格的功能，主要代码如下：

```
Cell cell = new Cell(new Paragraph("我占据 2 列", FontChinese));          //定义一个表格单元格对象
cell.setColspan(2);                                                      //设置表格列跨度（合并两个单元格）
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetCellColumn 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中设置表格的单元格所占列数的操作，主要代码如下：

```
ocument document = new Document();                                       //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置单元格所占的列数.pdf"));                                  //关联文档对象与输出流
document.open();                                                         //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                          //定义基础字体
Font FontChinese = new Font(Chinese, 10, Font.NORMAL, Color.red);        //实例化字体
Table table = new Table(5);                                              //定义表格
table.addCell("1,1");                                                    //将单元格顺次加入到表格，当一行充满时自动换行
table.addCell("2,1");
Cell cell = new Cell(new Paragraph("我占据 2 列", FontChinese));          //定义一个表格单元
cell.setColspan(2);                                                      //设置表格列跨度（合并两个单元格）
table.addCell(cell);                                                     //将单元加入到表格
table.addCell("2,1");                                                    //将单元格顺次加入到表格，当一行充满时自动换行
//省略了部分向表格中添加单元格内容的代码
Cell cell2 = new Cell(new Paragraph("我占据 4 列 ", FontChinese));        //定义一个表格单元
cell2.setColspan(4);                                                     //设置表格列跨度（合并 4 个单元格）
table.addCell(cell2);                                                    //将单元加入到表格
table.addCell("3,1");                                                    //将单元格顺次加入到表格，当一行充满时自动换行
//省略了部分向表格中添加单元格内容的代码
Cell cell3 = new Cell(new Paragraph("我占据 3 列 ", FontChinese));        //定义一个表格单元
cell3.setColspan(3);                                                     //设置表格列跨度（合并 3 个单元格）
table.addCell(cell3);                                                    //将单元加入到表格
table.addCell("2,3");
document.add(table);                                                     //将表格添加到文档
document.close();                                                        //关闭文档
```

秘笈心法

心法领悟 455：合并表格中的整行单元格。

本实例合并了表格中某一行的部分单元格，如果需要合并一整行单元格，可以将单元格的跨度设置为表格的列数，然后将该单元格添加到表格中这一行的第一个单元格处，实现代码如下：

```
Table table = new Table(5);                                              //定义有 5 列的表格
Cell cell = new Cell(new Paragraph("我占据 2 列", FontChinese));          //定义一个表格单元格对象
cell.setColspan(5);                                                      //设置表格列跨度（合并 5 个单元格）
table.addCell(cell);                                                     //将单元格加入到表格
```

实例 456　设置单元格所占的行数

光盘位置：光盘\MR\456　　中级　　趣味指数：★★★

实例说明

在 PDF 文档中创建表格时，有时需要将一列中的多个单元格合并成一个，这可以通过设置单元格所占的行数来实现，本实例实现了设置单元格所占行数的功能。运行程序，将在 PDF 文档中显示设置单元格所占行数的效果，如图 15.9 所示。

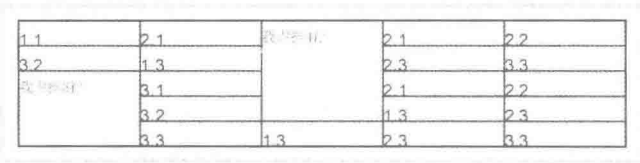


图 15.9　设置表格中单元格所占行数的效果

关键技术

本实例主要是通过 Cell 类创建单元格对象，然后使用 setRowspan()方法设置单元格所占的行数，从而实现合并表格一列中的多个单元格的功能，主要代码如下：

```
Cell cell = new Cell(new Paragraph("我占据 4 行", FontChinese));        //定义一个表格单元
cell.setRowspan(4);                                                    //设置表格行跨度（合并 4 个单元格）
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetCellRowspan 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中设置表格的单元格所占行数的操作，主要代码如下：

```
Document document = new Document();                                    //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置单元格所占的行数.pdf"));                               //关联文档对象与输出流
document.open();                                                       //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                        //定义基础字体
Font FontChinese = new Font(Chinese, 10, Font.NORMAL, Color.red);      //实例化字体
Table table = new Table(5);                                            //定义表格
table.addCell("1.1");                                                  //将单元格顺次加入到表格，当一行充满时自动换行
table.addCell("2.1");
Cell cell = new Cell(new Paragraph("我占据 4 行", FontChinese));        //定义一个表格单元
cell.setRowspan(4);                                                    //设置表格行跨度（合并 4 个单元格）
table.addCell(cell);                                                   //将单元格加入到表格
table.addCell("2.1");                                                  //将单元格顺次加入到表格，当一行充满时自动换行
//省略了部分向表格中添加单元格内容的代码
Cell cell2 = new Cell(new Paragraph("我占据 3 行 ", FontChinese));      //定义一个表格单元
cell2.setRowspan(3);                                                   //设置表格行跨度（合并 3 个单元格）
table.addCell(cell2);                                                  //将单元格加入到表格
table.addCell("3.1");                                                  //将单元格顺次加入到表格，当一行充满时自动换行
//省略了部分向表格中添加单元格内容的代码
document.add(table);                                                   //将表格添加到文档
document.close();                                                      //关闭文档
```

秘笈心法

心法领悟 456：合并表格中的整列单元格。

本实例合并了表格中某一列的部分单元格，如果需要合并一整列单元格，可以将单元格的行跨度设置为表格的行数，然后将该单元格添加到表格中这一列的第一个单元格处，假设表格有 4 行，则实现合并整列单元格的代码如下：

```
Table table = new Table(5);                                    //定义有 5 列的表格
Cell cell = new Cell(new Paragraph("我占据 4 行", FontChinese));  //定义一个表格单元
cell.setRowspan(4);                                            //设置表格行跨度（合并 4 个单元格）
table.addCell(cell);                                           //将单元格加入到表格
```

实例 457　设置单元格的背景色

光盘位置：光盘\MR\457　　中级　　趣味指数：★★★★

实例说明

在 PDF 文档中创建表格时，有时需要设置单元格的背景色，以改变表格中某些单元格的视觉效果。本实例实现了设置单元格背景色的功能。运行程序，将在 PDF 文档中显示设置单元格背景色的效果，如图 15.10 所示。

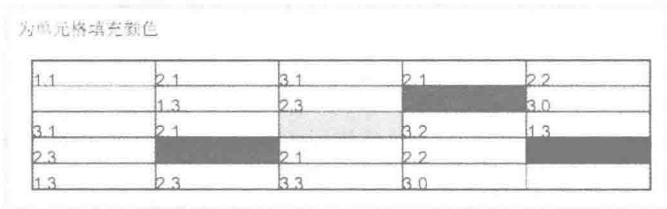


图 15.10　设置表格中单元格背景色的效果

关键技术

本实例主要是通过 Cell 类创建单元格对象，然后使用 setBackgroundColor()方法设置单元格的背景颜色。设置单元格背景颜色的代码如下：

```
Cell cell = new Cell();                                        //创建单元格
cell.setBackgroundColor(Color.yellow);                         //为单元格填充背景色
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetCellBackgroundColor 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中设置表格中单元格背景颜色的操作，主要代码如下：

```
Document document = new Document();                            //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置单元格的背景色.pdf"));                          //关联文档对象与输出流
document.open();                                               //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                //定义基础字体
Font FontChinese = new Font(Chinese, 12, Font.NORMAL);         //实例化字体
document
        .add(new Paragraph("为单元格填充颜色", FontChinese));
Table table = new Table(5);                                    //定义表格
table.addCell("1.1");                                          //将单元格顺次加入到表格，当一行充满时自动换行
table.addCell("2.1");
table.addCell("3.1");
table.addCell("2.1");
table.addCell("2.2");
Cell cell = new Cell();                                        //创建单元格
cell.setBackgroundColor(Color.yellow);                         //为单元格填充背景色
```

```
table.addCell(cell);                                    //将单元格填入到表格
table.addCell("1.3");
table.addCell("2.3");
Cell cell2 = new Cell();                                //创建单元格
cell2.setBackgroundColor(Color.red);                    //为单元格填充背景色
table.addCell(cell2);
table.addCell("3.0");
//省略了部分向表格中添加单元格内容的代码
document.add(table);                                    //将表格添加到文档
document.close();                                       //关闭文档
```

秘笈心法

心法领悟 457：为表格的整行单元格设置背景色。

在表格中，由于某些数据行的内容比较重要，可以将这些内容整行设置背景颜色，方法是使用 Cell 类创建一整行的单元格对象，并设置背景颜色，然后将这些单元格添加到表格中。

实例 458　嵌套表格

光盘位置：光盘\MR\458　　高级　　趣味指数：★★★★

实例说明

在 PDF 文档中创建表格时，有时可能需要在表格中嵌套表格。本实例实现了在表格中嵌套表格的功能。运行程序，可以看到在表格中嵌套了两个表格，效果如图 15.11 所示。

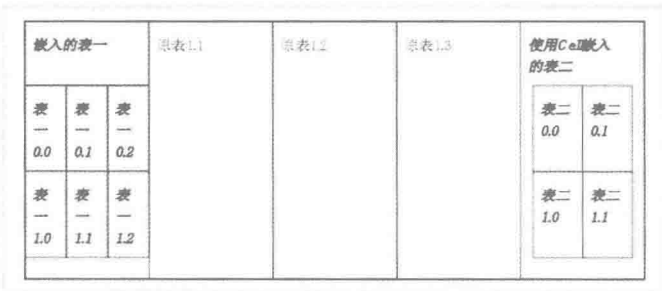


图 15.11　在表格中嵌套表格的效果

关键技术

本实例主要是通过 Table 类的 insertTable()方法插入表格，以及使用 Cell 类的 add()方法添加表格，并将 Cell 类的实例作为单元格添加到表格中，从而实现了在表格中嵌套表格的功能。在表格中嵌套表格的代码如下：

```
Table table1 = new Table(3);                                          //创建表格
Table table2 = new Table(2);                                          //创建表格
Cell tableCell = new Cell(new Paragraph("使用 Cell 嵌入的表二", FontChinese));  //创建一个单元格
tableCell.add(table2);                                                //将表格添加到单元格
Table table3 = new Table(5, 5);                                       //创建 5 行 5 列的原表
table3.insertTable(table1);                                           //将第 1 个表格嵌入到原表中第 1 列
table3.addCell(tableCell);                                            //添加单元格，实现第 2 个表格的嵌入
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 InsertTable 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档表格中嵌套表格的操作，主要代码如下：

```
Document document = new Document();                                    //创建文档对象
PdfWriter.getInstance(document,
        new FileOutputStream("c:\\嵌套表格.pdf"));                      //关联文档对象与输出流
document.open();                                                       //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                        //定义基础字体
Font FontChinese = new Font(Chinese, 10, Font.BOLDITALIC, Color.BLUE); //实例化字体
Font font = new Font(Chinese, 10, Font.NORMAL);
Table table1 = new Table(3);                                           //创建表格
Cell cell = new Cell(new Paragraph("嵌入的表一", FontChinese));          //创建单元格
cell.setColspan(3);                                                    //设置列跨度
//单元格添入到表格，行满自动换行
table1.addCell(cell);
table1.addCell(new Paragraph("表一 0.0", FontChinese));
table1.addCell(new Paragraph("表一 0.1", FontChinese));
table1.addCell(new Paragraph("表一 0.2", FontChinese));
table1.addCell(new Paragraph("表一 1.0", FontChinese));
table1.addCell(new Paragraph("表一 1.1", FontChinese));
table1.addCell(new Paragraph("表一 1.2", FontChinese));
Table table2 = new Table(2);                                           //创建表格
//单元格添入到表格，行满自动换行
table2.addCell(new Paragraph("表二 0.0", FontChinese));
table2.addCell(new Paragraph("表二 0.1", FontChinese));
table2.addCell(new Paragraph("表二 1.0", FontChinese));
table2.addCell(new Paragraph("表二 1.1", FontChinese));
Cell tableCell = new Cell(new Paragraph("使用 Cell 嵌入的表二", FontChinese)); //创建一个单元格
tableCell.add(table2);                                                 //将表格添加到单元格
Table table3 = new Table(5, 5);                                        //创建 5 行 5 列的原表
table3.insertTable(table1);                                            //将第 1 个表格嵌入到原表中第 1 列
//单元格添入到表格，行满自动换行
table3.addCell(new Paragraph("原表 1.1", font));
table3.addCell(new Paragraph("原表 1.2", font));
table3.addCell(new Paragraph("原表 1.3", font));
table3.setPadding(5);                                                  //设置填充值为 5
table3.addCell(tableCell);                                             //添加单元格，实现第 2 个表格的嵌入
document.add(table3);                                                  //向文档中添加原表
document.close();                                                      //关闭文档
```

秘笈心法

心法领悟 458：两种嵌套表格的区别。

本实例通过 Table 类的 insertTable()方法插入表格，以及使用 Cell 类的 add()方法添加表格，并将 Cell 类的实例作为单元格添加到表格中，从而实现了在表格中嵌套表格的功能，但是这两种方法是有区别的，其中使用 Table 类的 insertTable()方法嵌套表格时不能设置填距，而使用 Cell 类的方法时是可以设置填距的。

实例 459 偏移表格

光盘位置：光盘\MR\459　　高级　　趣味指数：★★★★

实例说明

在 PDF 文档中创建表格时，有时可能需要在一页文档中添加多个表格，这时可以对表格之间的间距进行设置，即设置表格的偏移。本实例实现了设置表格偏移的功能。运行程序，可以看到表格偏移的效果，如图 15.12 所示。

关键技术

本实例主要使用 Table 类的 setOffset()方法设置表格的偏移，代码如下：

```
Table table = new Table(3);                    //创建表格
table.setOffset(0);                            //设置表格偏移值为 0
```

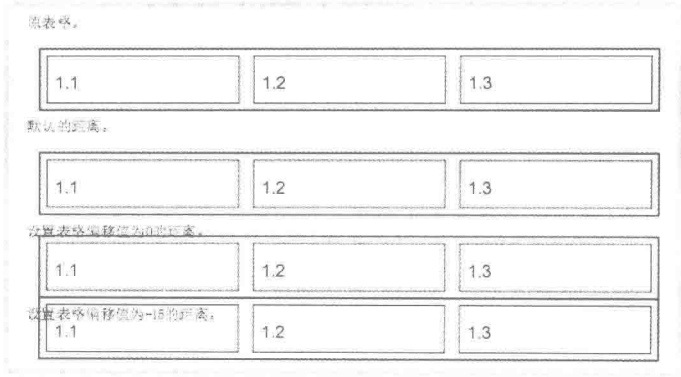


图 15.12　设置表格偏移的效果

设计过程

（1）新建一个项目。

（2）在项目中创建一个 ExcursionTable 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，完成在 PDF 文档中偏移表格的操作，主要代码如下：

```
Document document = new Document();                          //创建文档对象
PdfWriter.getInstance(document,
        new FileOutputStream("c:\\偏移表格.pdf"));           //关联文档对象与输出流
document.open();                                             //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);              //定义基础字体
Font FontChinese = new Font(Chinese, 10, Font.NORMAL);       //实例化字体
Table table = new Table(3);                                  //创建表格
table.setBorderWidth(1);                                     //设置表格边框宽度
table.setBorderColor(Color.blue);                            //设置表格边框颜色
table.setSpacing(5);                                         //设置表格与单元格的间距
table.setPadding(5);                                         //设置单元格与内容的间距
table.addCell("1.1");                                        //添加单元格
table.addCell("1.2");
table.addCell("1.3");
document
        .add(new Paragraph("原表格。", FontChinese));        //向文档添加内容
document.add(table);                                         //向文档添加表格
document.add(new Paragraph("默认的距离。",
        FontChinese));
document.add(table);
document.add(new Paragraph("设置表格偏移值为 0 的距离。", FontChinese));
table.setOffset(0);                                          //设置表格偏移数值
document.add(table);
document.add(new Paragraph("设置表格偏移值为-15 的距离。", FontChinese));
table.setOffset(-15);                                        //设置表格偏移数值
document.add(table);
document.close();                                            //关闭文档
```

秘笈心法

心法领悟 459：将多个表格连接成一个大表格。

如果已经创建了多个表格，当需要将这些表格的内容连接成一个大表格时，可以通过表格偏移来实现，但是需要试探着调整表格的偏移值，直到多个表格连接成一个大表格为止。

15.2 PdfPTable 表格

实例 460	创建表格	中级
	光光盘位置：光盘\MR\460	趣味指数：★★★★

实例说明

15.1 节讲解了使用 iText 的早期版本中的 Table 类创建表格，本实例使用 iText 的最新版本中的 PdfPTable 创建表格。运行程序，效果如图 15.13 所示。

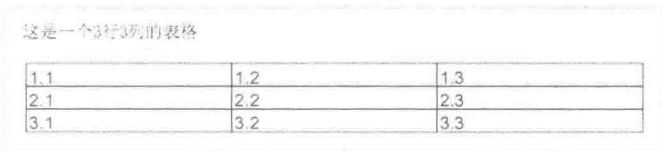


图 15.13 使用 PdfPTable 类创建的表格

注意： 在 iText 的新版本中，已经将早期版本的 Table 类合并到 PdfPTable 中，因此在新版本的 iText 包中已经找不到 Table 类了。

关键技术

本实例主要是使用 PdfPTable 类的构造方法 PdfPTable(int cols)创建具有指定列数的表格。其实现代码如下：

```
PdfPTable table = new PdfPTable(3);                    //创建表格
table.addCell("1.1");                                  //将单元格顺次加入到表格，当一行充满时自动换行
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 CreateTable 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，实现使用 PdfPTable 类在 PDF 文档中创建表格的功能，主要代码如下：

```
Document document = new Document();                    //创建文档对象
PdfWriter.getInstance(document,
        new FileOutputStream("c:\\创建表格.pdf"));     //关联文档对象与输出流
document.open();                                       //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);        //定义基础字体
Font FontChinese = new Font(Chinese, 12, Font.NORMAL); //实例化字体
PdfPTable table = new PdfPTable(3);                    //创建表格
table.addCell("1.1");                                  //将单元格顺次加入到表格，当一行充满时自动换行
//省略了部分向表格中添加单元格内容的代码
document.add(new Paragraph("这是一个 3 行 3 列的表格\n.",
        FontChinese));                                 //向文档添加内容
document.add(table);                                   //将表格添加到文档
document.close();                                      //关闭文档
```

秘笈心法

心法领悟 460：使用已有的 PdfPTable 类创建表格。

在创建表格时，如果已经创建了一个 PdfPTable 对象，则可以将该对象作为 PdfPTable 类构造方法的参数创建一个新的 PdfPTable 对象，新对象与已有对象具有相同的内容，同时还可以在新的表格对象中添加单元格，

例如：

```
PdfPTable table1 = new PdfPTable(table);          //使用已有的表格对象 table 创建新的表格对象 table1
table1.addCell("12.1");                           //在新的表格对象中添加单元格
```

实例 461　设置表格宽度

光盘位置：光盘\MR\461　　中级　　趣味指数：★★★

实例说明

在 PDF 文档中添加的表格，默认宽度就是页面的宽度，如果需要，可以对表格的宽度进行设置。本实例演示如何设置表格的宽度。运行程序，可以看到设置表格宽度后的效果，如图 15.14 所示。

5%	10%	30%	55%
w	1	2	3
h	1	2	3
s	1	2	3

图 15.14　设置表格宽度的效果

关键技术

本实例主要使用 PdfPTable 类的 setWidthPercentage()方法，通过百分比设置表格宽度，代码如下：

```
float[] widths = { 0.05f, 0.10f, 0.30f, 0.55f };  //设置列宽相关比率为 5%、10%、30%、55%
PdfPTable table = new PdfPTable(widths);          //创建表格关联列宽
table.setWidthPercentage(60);                     //为表格设置百分比宽度
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetTableWidth 类，该类中只有一个 main()主方法，用于实现在 PDF 文档中设置表格宽度的功能，主要代码如下：

```
Document document = new Document();                          //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置绝对宽度.pdf"));                            //关联文档对象与输出流
document.open();                                             //打开文档
float[] widths = { 0.05f, 0.10f, 0.30f, 0.55f };             //设置列宽相关比率为 5%、10%、30%、55%
PdfPTable table = new PdfPTable(widths);                     //创建表格关联列宽
table.setWidthPercentage(60);                                //为表格设置百分比宽度
//将单元格顺次加入到表格，当一行充满时自动换行
table.addCell("5%");
table.addCell("10%");
table.addCell("30%");
table.addCell("55%");
//省略了部分向表格中添加单元格内容的代码
document.add(table);                                         //将表格添加到文档
document.close();                                            //关闭文档
```

秘笈心法

心法领悟 461：合并多行多列的单元格。

在表格中可以合并多行多列的单元格，使其变成一个较大的单元格，方法是同时设置单元格行跨度和列跨度，要合并两行两列的单元格，可以使用如下代码实现：

```
PdfPTable table = new PdfPTable(3);               //定义表格
PdfPCell cell = new PdfPCell();                   //创建单元格
cell.setRowspan(2);                               //设置单元格的行跨度
```

```
cell.setColspan(2);                                    //设置单元格的列跨度
table.addCell(cell);                                   //将单元格添加到表格
```

实例 462 设置表格对齐方式

光盘位置：光盘\MR\462　　中级　　趣味指数：★★★

实例说明

在 PDF 文档中添加的表格，除了可以设置表格的宽度以外，还可以设置表格的对齐方式。本实例演示了如何设置表格的对齐方式。运行程序，可以看到表格左对齐、居中对齐和右对齐的效果，如图 15.15 所示。

图 15.15　设置表格对齐方式的效果

关键技术

本实例主要使用 PdfPTable 类的 setHorizontalAlignment()方法设置表格的对齐方式。在 PDF 文档中设置表格在水平方向上左对齐的代码如下：

```
PdfPTable table = new PdfPTable(3);                    //定义表格
table.setHorizontalAlignment(Element.ALIGN_LEFT);      //设置水平左对齐
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetTableAllignment 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，实现在 PDF 文档中设置表格对齐方式的功能，主要代码如下：

```
Document document = new Document();                    //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置表格对齐方式.pdf"));                   //关联文档对象与输出流
document.open();                                       //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);        //定义基础字体
Font FontChinese = new Font(Chinese, 10, Font.BOLD);   //实例化字体
PdfPTable table = new PdfPTable(3);                    //定义表格
table.setTotalWidth(200);                              //设置表格宽度为 200
table.setLockedWidth(true);
PdfPCell cell = new PdfPCell(new Paragraph("new table colspan 3"));  //定义一个表格单元
cell.setColspan(3);                                    //设置表格跨度
table.addCell(cell);                                   //将单元格加入到表格
//将单元格顺次加入到表格，当一行充满时自动换行
table.addCell("1.1");
table.addCell("2.1");
table.addCell("3.1");
table.setHorizontalAlignment(Element.ALIGN_LEFT);      //设置水平左对齐
PdfPTable table1 = new PdfPTable(3);                   //定义表格
table1.setTotalWidth(200);                             //设置表格宽度为 200
table1.setLockedWidth(true);
```

```
PdfPCell cell2 = new PdfPCell(new Paragraph("new table colspan 3"));   //定义一个表格单元
cell2.setColspan(3);                                                  //设置表格跨度
table1.addCell(cell2);                                                //将单元格加入到表格
//将单元格顺次加入到表格，当一行充满时自动换行
table1.addCell("1.1");
table1.addCell("2.1");
table1.addCell("3.1");
table1.setHorizontalAlignment(Element.ALIGN_CENTER);                  //设置水平居中对齐
PdfPTable table2 = new PdfPTable(3);                                  //定义表格
table2.setTotalWidth(200);                                            //设置表格宽度为 200
table2.setLockedWidth(true);
PdfPCell cell3 = new PdfPCell(new Paragraph("new table colspan 3"));  //定义一个表格单元
cell3.setColspan(3);                                                  //设置表格跨度
table2.addCell(cell2);                                                //将单元格加入到表格
//将单元格顺次加入到表格，当一行充满时自动换行
table2.addCell("1.1");
table2.addCell("2.1");
table2.addCell("3.1");
table2.setHorizontalAlignment(Element.ALIGN_RIGHT);                   //设置水平右对齐
Paragraph p = new Paragraph("表格居左对齐\n\n", FontChinese);
p.setAlignment(Element.ALIGN_LEFT);
document.add(p);                                                      //向文档添加内容
document.add(table);                                                  //将表格添加到文档
Paragraph p2 = new Paragraph("表格居中对齐\n\n", FontChinese);
p2.setAlignment(Element.ALIGN_CENTER);
document.add(p2);                                                     //向文档添加内容
document.add(table1);                                                 //将表格添加到文档
Paragraph p3 = new Paragraph("表格居右对齐\n\n", FontChinese);
p3.setAlignment(Element.ALIGN_RIGHT);
document.add(p3);                                                     //向文档添加内容
document.add(table2);                                                 //将表格添加到文档
document.close();                                                     //关闭文档
```

秘笈心法

心法领悟 462：使用 PdfPCell 类设置单元格的列跨度。

在使用 PdfPTable 创建表格时，如果希望指定单元格的列跨度，可以使用 PdfPCell 类创建单元格对象，然后使用 setColspan()方法设置单元格的列跨度。例如：

```
PdfPTable table = new PdfPTable(3);                                   //定义表格
PdfPCell cell = new PdfPCell(new Paragraph("new table colspan 3"));   //定义一个表格单元格对象
cell.setColspan(3);                                                   //设置单元格的列跨度
table.addCell(cell);                                                  //将单元格添加到表格
```

实例 463　设置表格的列宽

光盘位置：光盘\MR\463　　中级　　趣味指数：★★★

实例说明

在 PDF 文档中添加表格时，默认情况下所有列的宽度都是相同的，如果用户想根据需要设置表格的列宽，可以通过本实例讲解的方法实现。运行程序，可以看到对表格列宽进行设置后的效果，如图 15.16 所示。

w	1	2	3
h	1	2	3
s	1	2	3

图 15.16　设置表格列宽的效果

关键技术

本实例主要使用 PdfPTable 类的构造方法 PdfPTable(float[] widths)创建表格对象，其中，参数 widths 是一个表示表格列宽相对比例的 float 数组。在 PDF 文档中设置表格列宽的代码如下：

```
float[] widths = { 0.05f, 0.10f, 0.30f, 0.55f };          //设置列宽相对比例为 5%、10%、30%、55%
PdfPTable table = new PdfPTable(widths);                  //创建表格关联列宽
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetTableColWidths 类，该类中只有一个 main()主方法，用于实现在 PDF 文档中设置表格列宽的功能，主要代码如下：

```
Document document = new Document();                       //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置表格的列宽.pdf"));                        //关联文档对象与输出流
document.open();                                          //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);           //定义基础字体
Font FontChinese = new Font(Chinese, 10, Font.NORMAL,
        new BaseColor(90, 90, 90));                       //实例化字体
float[] widths = { 0.05f, 0.10f, 0.30f, 0.55f };          //设置列宽相对比例为 5%、10%、30%、55%
PdfPTable table = new PdfPTable(widths);                  //创建表格关联列宽
table.addCell(new Paragraph("列宽为 5%", FontChinese));    //将单元格顺次加入到表格，当一行充满时自动换行
table.addCell(new Paragraph("列宽为 10%", FontChinese));
table.addCell(new Paragraph("列宽为 30%", FontChinese));
table.addCell(new Paragraph("列宽为 55%", FontChinese));
//省略了部分向表格中添加单元格内容的代码
document.add(table);                                      //将表格添加到文档
document.close();                                         //关闭文档
```

秘笈心法

心法领悟 463：另一种设置表格列宽的方法。

除了使用 PdfPTable 类的构造方法确定表格的列宽外，还可以使用该类的 setTotalWidth()方法设置表格的列宽，实现代码如下：

```
float[] widths = { 0.05f, 0.10f, 0.30f, 0.55f };          //设置列宽相对比例为 5%、10%、30%、55%
PdfPTable table = new PdfPTable(4);                       //创建表格
table.setTotalWidth(widths);                              //设置表格的列宽
```

实例 464 设置绝对宽度

光盘位置：光盘\MR\464　　中级　　趣味指数：★★★★

实例说明

在 PDF 文档中添加表格时，可以根据需要设置表格的绝对宽度。本实例实现了设置表格绝对宽度的功能。运行程序，可以看到默认表格宽度和设置表格绝对宽度的效果，如图 15.17 所示。

关键技术

本实例主要使用 PdfPTable 类的 setTotalWidth()方法设置表格的绝对宽度，其实现代码如下：

```
PdfPTable table = new PdfPTable(3);                       //定义表格
table.setTotalWidth(200);                                 //设置表格宽度为 200
```

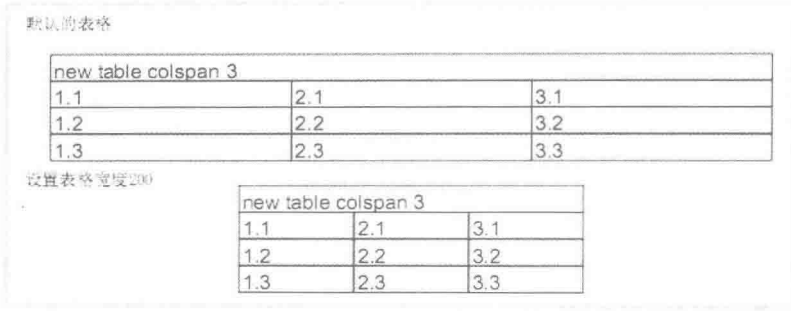


图 15.17　默认表格与设置表格绝对宽度后的效果

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetAbsoluteWidth 类，该类中只有一个 main()主方法，在应用程序启动后将执行该方法的代码，实现使用 PdfPTable 类在 PDF 文档中设置表格宽度的功能，主要代码如下：

```
Document document = new Document();                                    //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置绝对宽度.pdf"));                                       //关联文档对象与输出流
document.open();                                                       //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light",
        "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                        //定义基础字体
Font FontChinese = new Font(Chinese, 10, Font.NORMAL);                 //实例化字体
PdfPTable table1 = new PdfPTable(3);                                   //定义表格
PdfPCell cell1 = new PdfPCell(new Paragraph("new table colspan 3"));   //定义一个表格单元
cell1.setColspan(3);                                                   //设置表格跨度
table1.addCell(cell1);                                                 //将单元格加入到表格
table1.addCell("1.1");                                                 //将单元格顺次加入到表格，当一行充满时自动换行
//省略了部分向表格中添加单元格内容的代码
PdfPTable table2 = new PdfPTable(3);                                   //创建表格
PdfPCell cell2 = new PdfPCell(new Paragraph("new table colspan 3"));   //定义一个表格单元
cell2.setColspan(3);                                                   //设置表格跨度
table2.addCell(cell2);                                                 //将单元格加入到表格
table2.addCell("1.1");                                                 //将单元格顺次加入到表格，当一行充满时自动换行
//省略了部分向表格中添加单元格内容的代码
table2.setTotalWidth(200);                                             //设置表格宽度为 200
table2.setLockedWidth(true);                                           //锁定宽度
document.add(new Paragraph("默认的表格", FontChinese));
document.add(table1);                                                  //将表格添加到文档
document.add(new Paragraph("设置表格宽度 200", FontChinese));
document.add(table2);                                                  //将表格添加到文档
document.close();                                                      //关闭文档
```

秘笈心法

心法领悟 464：解决设置表格的绝对宽度的问题。

在 PDF 文档中创建表格时，如果对表格的宽度进行了设置，但是表格的宽度却没有改变，这时需要使用 PdfPTable 类的 setLockedWidth()方法进行锁定，这样表格宽度才能发生变化，例如：

```
PdfPTable table = new PdfPTable(3);                                    //定义表格
table.setTotalWidth(200);                                              //设置表格宽度为 200
table.setLockedWidth(true);                                            //锁定宽度，使所设置的宽度有效
```

实例 465 嵌套表格

光盘位置：光盘\MR\465　　高级　　趣味指数：★★★★

实例说明

在实例 458 中讲解了使用 Table 类嵌套表格，本实例使用 PdfPTable 类实现了嵌套表格的功能。运行程序，可以看到嵌套表格的效果，如图 15.18 所示。

table1		table2	text	text
1.1	1.2	2.1 2.2	cell	cell

图 15.18　使用 PdfPTable 类嵌套表格的效果

关键技术

本实例主要通过 PdfPTable 类的 addCell()方法实现了表格的嵌套功能，代码如下：

```
table.addCell(table1);                    //添加嵌套的表格
table.addCell(table2);                    //添加嵌套的表格
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PdfInsertTable 类，该类中只有一个 main()主方法，用于实现在 PDF 文档的表格中嵌套表格的功能，主要代码如下：

```
Document document = new Document();
PdfWriter.getInstance(document, new FileOutputStream("c:\\Pdf 嵌套表格.pdf"));
document.open();                          //打开文档
PdfPTable table = new PdfPTable(4);       //创建表格对象
PdfPTable table1 = new PdfPTable(2);      //创建表格对象
table1.addCell("1.1");                    //添加单元格内容
table1.addCell("1.2");                    //添加单元格内容
PdfPTable table2 = new PdfPTable(1);      //创建表格对象
table2.addCell("2.1");                    //添加单元格内容
table2.addCell("2.2");                    //添加单元格内容
table.addCell("table1");                  //添加单元格内容
table.addCell("table2");                  //添加单元格内容
table.addCell("text");                    //添加单元格内容
table.addCell("text");                    //添加单元格内容
table.addCell(table1);                    //添加嵌套的表格
table.addCell(table2);                    //添加嵌套的表格
table.addCell("cell");                    //添加单元格内容
table.addCell("cell");                    //添加单元格内容
document.add(table);                      //将表格添加到文档中
document.close();                         //关闭文档
```

秘笈心法

心法领悟 465：调整嵌套的表格与单元格边框的间距。

本实例嵌套的表格与单元格的边框有一定的间距，如果希望控制该间距，可以使用如下代码实现：

```
table.getDefaultCell().setPadding(0);     //设置单元格填距
table.addCell(table1);                    //添加嵌套的表格
table.addCell(table2);                    //添加嵌套的表格
```

实例 466　在表格中添加图片

光盘位置：光盘\MR\466　　高级　　趣味指数：★★★★

实例说明

在表格中，除了可以添加普通的文本信息、嵌套表格之外，还可以添加图片。本实例实现了在表格中添加图片的功能。运行程序，可以看到在表格中添加图片的效果，如图 15.19 所示。

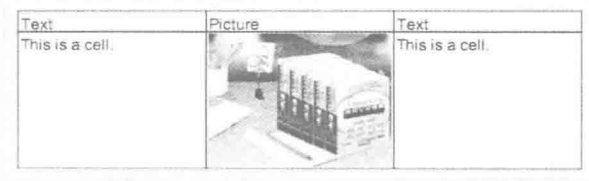


图 15.19　在表格中添加图片的效果

关键技术

本实例主要是通过 Image 类创建一个图像对象，然后使用 PdfPTable 类的 addCell()方法将图像对象添加到表格的单元格中，从而实现了在表格中添加图片的功能，实现代码如下：

```
Image image = Image.getInstance("image/picture.jpg");          //创建图像对象
PdfPTable table = new PdfPTable(3);                            //定义表格
table.addCell(image);                                          //向单元格中添加图像对象
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 AddPictureInTable 类，该类中只有一个 main()主方法，用于实现在 PDF 文档的表格中嵌套表格的功能，主要代码如下：

```
Document document = new Document();                            //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\在表格中添加图片.pdf"));                          //关联文档对象与输出流
document.open();                                               //打开文档
Image image = Image.getInstance("image/picture.jpg");          //创建图像对象
PdfPTable table = new PdfPTable(3);                            //定义表格
table.addCell("Text");                                         //添加单元格内容
table.addCell("Picture");                                      //添加单元格内容
table.addCell("Text");                                         //添加单元格内容
table.addCell("This is a cell.");                              //添加单元格内容
table.addCell(image);                                          //向单元格中添加图像对象
table.addCell("This is a cell.");                              //添加单元格内容
document.add(table);                                           //将表格添加到文档
document.close();                                              //关闭文档
```

秘笈心法

心法领悟 466：使图片跨越多行多列的单元格。

本实例在表格中添加的图片单独占据一个单元格，如果需要，可以使图片跨越多行多列的单元格，方法是将图片对象添加到跨越多行多列的单元格对象中，然后将该单元格添加到表格中，实现代码如下：

```
cell.setRowspan(2);                                            //设置单元格的行跨度
cell.setColspan(2);                                            //设置单元格的列跨度
cell.addElement(image);                                        //在单元格中添加图片
table.addCell(cell);                                           //将单元格添加到表格
```

实例 467 设置单元格的高度

光盘位置：光盘\MR\467

中级

趣味指数：★★★

实例说明

在 PDF 文档中添加的表格都是以默认的高度显示的，如果需要调整表格的高度，可以通过本实例讲解的方法实现。运行程序，可以看到设置表格中单元格高度后的效果，如图 15.20 所示。

default height	AAA
set height	ABC
minimum height	A cat may look at a king.

图 15.20 设置单元格高度的显示效果

关键技术

本实例主要是通过 PdfPCell 类的 setFixedHeight()和 setMinimumHeight()方法设置单元格高度的。其实现代码如下：

```
PdfPCell cell1 = new PdfPCell(new Paragraph("FixedHeight"));          //定义单元格
cell1.setFixedHeight(60);                                             //设置单元格高度为 60
PdfPCell cell2 = new PdfPCell(new Paragraph("MinimumHeight"));        //定义单元格
cell2.setMinimumHeight(40);                                           //设置单元格高度为 40
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetCellHeight 类，该类中只有一个 main()主方法，实现设置 PDF 表格中单元格高度的功能，主要代码如下：

```
Font font = FontFactory.getFont("COURIER", 10, Font.BOLD);            //定义一个字体
Font xfont = FontFactory.getFont("HELVETICA", 10, Font.BOLD);         //定义一个字体
Document document = new Document();                                   //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置单元格的高度.pdf"));                                  //关联文档对象与输出流
document.open();                                                      //打开文档
PdfPTable table = new PdfPTable(2);                                   //定义表格
table.getDefaultCell().setBackgroundColor(BaseColor.ORANGE);
table.addCell(new Paragraph("default height", xfont));                //向表格添加单元格
PdfPCell cell = new PdfPCell(new Paragraph("AAA", font));             //定义一个表格单元
table.addCell(cell);                                                  //向表格添加单元格
table.addCell(new Paragraph("set height", xfont));
PdfPCell cell2 = new PdfPCell(new Paragraph("ABC", font));            //定义单元格
cell2.setFixedHeight(60);                                             //设置单元格高度为 60
table.addCell(cell2);                                                 //将单元格加入到表格
table.addCell(new Paragraph("minimum height", xfont));
PdfPCell cell3 = new PdfPCell(new Paragraph(
        "A cat may look at a king. ", font));                         //定义单元格
cell3.setMinimumHeight(40);                                           //设置单元格高度为 40
table.addCell(cell3);                                                 //将单元格加入到表格
document.add(table);                                                  //将表格添加到文档
document.close();                                                     //关闭文档
```

秘笈心法

心法领悟 467：PdfPCell 类的 setFixedHeight()和 setMinimumHeight()方法的区别。

使用 PdfPCell 类的 setFixedHeight()方法设置单元格的高度后，单元格的高度是固定的，当单元格中的内容较多时，单元格的高度也不会自动增加；而 PdfPCell 类的 setMinimumHeight()方法则是设置单元格的最小高度，当单元格中的内容较多时，会根据内容自动增加单元格的高度。

实例 468　设置单元格的对齐方式

光盘位置：光盘\MR\468　　中级　　趣味指数：★★★

实例说明

在 PDF 文档中添加的表格，单元格中的内容默认是靠左对齐的，本实例讲解如何设置单元格中内容的对齐方式。运行程序，可以看到设置表格中单元格对齐方式后的效果，如图 15.21 所示。

alignment left	I think Bale will win
alignment right	I think Bale will win
alignment justified	I think Bale will win
alignment center	I think Bale will win

图 15.21　设置单元格对齐方式的效果

关键技术

本实例主要是通过 PdfPCell 类的 setHorizontalAlignment()方法实现了设置单元格对齐方式的功能。设置单元格内容右对齐的实现代码如下：

```
Paragraph p = new Paragraph("I think Bale will win");          //定义段落对象
PdfPCell cell = new PdfPCell(p);                               //定义单元格
cell.setHorizontalAlignment(Element.ALIGN_RIGHT);              //设置单元格水平向右对齐
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetCellAlignment 类，该类中只有一个 main()主方法，实现设置 PDF 表格中单元格对齐方式的功能，主要代码如下：

```
Document document = new Document();                            //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置单元格的对齐方式.pdf"));                       //关联文档对象与输出流
document.open();                                               //打开文档
PdfPTable table = new PdfPTable(2);                            //定义表格
Paragraph p = new Paragraph("I think Bale will win");          //定义段落和内容
table.addCell("alignment left");                               //向单元格添加内容
PdfPCell cell = new PdfPCell(p);                               //定义单元格
cell.setHorizontalAlignment(Element.ALIGN_LEFT);               //设置单元格水平向左对齐
table.addCell(cell);
table.addCell("alignment right");                              //向单元格添加内容
PdfPCell cell1 = new PdfPCell(p);                              //定义单元格
cell1.setHorizontalAlignment(Element.ALIGN_RIGHT);             //设置单元格水平向右对齐
table.addCell(cell1);
table.addCell("alignment justified");                          //向单元格添加内容
PdfPCell cell2 = new PdfPCell(p);                              //定义单元格
cell2.setHorizontalAlignment(Element.ALIGN_JUSTIFIED);         //设置单元格为合理的对齐方式
```

```
table.addCell(cell2);
table.addCell("alignment center");                    //向单元格添加内容
PdfPCell cell3 = new PdfPCell(p);                     //定义单元格
cell3.setHorizontalAlignment(Element.ALIGN_CENTER);   //设置单元格水平居中对齐
table.addCell(cell3);
table.addCell("Bale\nblah\nblah\nblah\nblah");         //向单元格添加内容
document.add(table);                                  //将表格添加到文档
document.close();                                     //关闭文档
```

秘笈心法

心法领悟 468：设置单元格的垂直对齐方式。

本实例使用 PdfPCell 类的 setHorizontalAlignment()方法，设置了单元格中内容的水平对齐方式，如果需要设置单元格的垂直对齐方式，可以使用 setVerticalAlignment()方法实现。设置单元格内容垂直居中对齐可以使用如下代码实现：

```
Paragraph p = new Paragraph("I think Bale will win");   //定义段落对象
PdfPCell cell = new PdfPCell(p);                        //定义单元格
cell.setVerticalAlignment(Element.ALIGN_MIDDLE);        //设置单元格垂直居中对齐
```

实例 469　设置单元格的填充和行间距

中级

光盘位置：光盘\MR\469

趣味指数：★★★

实例说明

在 PDF 文档的表格中添加的内容，通常情况下与单元格的边框间使用默认的距离（即填充），而且同一单元格中的多行文本的行间距也是默认的。本实例实现了设置单元格的填充和行间距的功能。运行程序，可以看到设置表格中单元格的填充和行间距的效果，如图 15.22 所示。

图 15.22　设置单元格填充和行间距的效果

关键技术

本实例主要是通过 PdfPTable 类的 getDefaultCell()方法获得 PdfPCell 对象，然后使用 PdfPCell 类的 setPadding()方法实现设置单元格填充的功能，使用 setLeading()方法实现设置单元格内容行间距的功能。设置单元格填充和行间距的实现代码如下：

```
PdfPTable table = new PdfPTable(2);                   //定义表格
table.getDefaultCell().setPadding(24);                //设置单元格填充为 24
PdfPTable table1 = new PdfPTable(2);                  //定义表格
table1.getDefaultCell().setLeading(12, 1);            //添加行间距
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 SetCellPadSpacing 类，该类中只有一个 main()主方法，实现设置 PDF 表格中单元格的填充和行间距的功能，主要代码如下：

```
Document document = new Document();                                          //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\设置单元格的填充和行间距.pdf"));                                  //关联文档对象与输出流
document.open();                                                             //打开文档
PdfPTable table = new PdfPTable(2);                                          //定义表格
table.addCell("no Padding");                                                 //向单元格添加内容
table.addCell("MingRiCompany MingRi MingRiCompany");                         //单元格填充前内容
table.addCell("Set Padding");                                                //向单元格添加内容
table.getDefaultCell().setPadding(24);                                       //设置单元格填充为 24
table.addCell("MingRi MingRiCompany");                                       //单元格填充后内容
document.add(table);                                                         //将表格添加到文档
PdfPTable table1 = new PdfPTable(2);                                         //定义表格
table1.addCell("no Leading");                                                //向单元格添加内容
table1.addCell("MingRi MingRi MingRiCompanyMingRiCompany MingRiCompany");    //添加行间距前内容
table1.getDefaultCell().setLeading(12, 1);                                   //添加行间距
table1.addCell("Set Leading");                                               //向单元格添加内容
table1.addCell("MingRi MingRi MingRiCompanyMingRiCompany MingRiCompany");    //添加行间距后内容
document.add(table1);                                                        //将表格添加到文档
document.close();                                                            //关闭文档
```

秘笈心法

心法领悟 469：PdfPCell 创建的单元格对象与 PdfPTable 的 getDefaultCell()方法获得的单元格对象的区别。

使用 PdfPCell 类创建的单元格对象，使用 setPadding()方法设置的填充只对一个单元格起作用，同样使用 setLeading()方法设置的行间距也只对一个单元格起作用；而使用 PdfPTable 类的 getDefaultCell()方法获得的单元格对象，则是对使用 setPadding()和 setLeading()方法之后，再通过 addCell(String cell)方法添加的单元格内容都起作用。

实例 470　行优先分页

光盘位置：光盘\MR\470　　高级　　趣味指数：★★★

实例说明

如果 PdfPTable 表格中某一行的内容较多，而无法在当前页完全显示一整行内容，将自动按照行优先的方式进行分页，即自动将该行放到下一页显示，并且只有在文档的一整页都无法显示表格完整的一行时，才会将该行的内容拆开显示在两页或多页中。运行程序，可以看到表格按行优先分页的效果，如图 15.23 所示。

关键技术

本实例主要是通过 iText 工具包中 PdfPTable 类创建表格的固有特性实现的，即使用 PdfPTable 类创建的表格，默认是按照行优先分页的方式进行分页的，并不需要软件开发人员对程序进行任何处理。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 RowFirstPagination 类，该类中只有一个 main()主方法，实现 PDF 表格中单元格的行优先分页功能，主要代码如下：

```
Document document = new Document();                                    //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\行优先分页.pdf"));                                          //关联文档对象与输出流
document.open();                                                       //打开文档
String[] data = { "C033010", "MX", "980", "350", "800", "999",
        "655", "800", "23", "860" };                                   //定义数据信息
PdfPTable table = new PdfPTable(10);                                   //定义表格
int columnwidths[] = { 8, 3, 11, 10, 8, 6, 8, 12, 3, 6 };              //定义列宽
table.setWidths(columnwidths);                                         //向表格添加列宽
table.setWidthPercentage(100);                                         //向表格添加绝对宽度
table.getDefaultCell().setPadding(3);                                  //设置单元格填充为 3
table.getDefaultCell().setHorizontalAlignment(Element.ALIGN_CENTER);   //设置单元格居中对齐
table.addCell("Number");                                               //将单元格内容顺次加入到表格，当一行充满时自动换行
//省略了添加表格标题的部分代码
table.setHeaderRows(1);                                                //为表格每一页设置表头
for (int i = 1; i < 100; i++) {                                        //循环向表格中添加 100 条记录
    if (i % 2 == 1) {
        table.getDefaultCell().setBackgroundColor(
                BaseColor.LIGHT_GRAY);                                 //填充颜色
    } else {
        table.getDefaultCell().setBackgroundColor(BaseColor.WHITE);    //填充颜色
    }
    for (int x = 0; x < 10; x++) {
        String var = data[x];                                          //获得数组中的数据
        for (int y = 0; y < i; y++) {
            var += "\n" + y;                                           //连接字符串生成单元格内容
        }
        table.addCell(var);                                            //为单元格添加内容
    }
}
document.add(table);                                                   //向文档添加表格
document.close();                                                      //关闭文档
```

Number	Name	aggression	defend	reaction	shoot	header	bodybalance	age	speed
C033010	M	980	350	800	999	655	800	23	860
0	X	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1
2	1	2	2	2	2	2	2	2	2
3	2	3	3	3	3	3	3	3	3
4	3	4	4	4	4	4	4	4	4
5	4	5	5	5	5	5	5	5	5
6	5	6	6	6	6	6	6	6	6
7	6	7	7	7	7	7	7	7	7
8	7	8	8	8	8	8	8	8	8
9	8	9	9	9	9	9	9	9	9
10	9	10	10	10	10	10	10	10	10
11	10	11	11	11	11	11	11	11	11
12	11	12	12	12	12	12	12	12	12
13	12	13	13	13	13	13	13	13	13
14	13	14	14	14	14	14	14	14	14
15	14	15	15	15	15	15	15	15	15
16	15	16	16	16	16	16	16	16	16
17	16	17	17	17	17	17	17	17	17
18	17	18	18	18	18	18	18	18	18
19	18	19	19	19	19	19	19	19	19
20	19	20	20	20	20	20	20	20	20
21	20	21	21	21	21	21	21	21	21
22	21	22	22	22	22	22	22	22	22
23	22	23	23	23	23	23	23	23	23
24	23	24	24	24	24	24	24	24	24
25	24	25	25	25	25	25	25	25	25
26	25	26	26	26	26	26	26	26	26
	26								

由于下一行无法在当前页显示，所以自动移到下一页，因此这里空白较大

图 15.23　表格按行优先分页的显示效果

秘笈心法

心法领悟 470：手动设置行优先分页。

除了 PDF 表格默认的行优先分页以外，还可以手动设置表格的行优先分页功能，方法是使用 PdfPTable 类的 setSplitLate()方法，并将入口参数设置为 true，实现代码如下：

```
PdfPTable table = new PdfPTable(10);                    //定义表格
table.setSplitLate(true);                               //设置表格行优先分页
```

实例 471　页优先分页

光盘位置：光盘\MR\471　　高级　　趣味指数：★★★

实例说明

如果 PdfPTable 表格中某一行的内容较多，而无法在当前页完全显示一整行内容，可以采用页优先的方式，使当前页显示不完的内容到下一页显示。本实例实现了页优先分页的功能。运行程序，可以看到表格按页优先分页的效果，如图 15.24 所示。

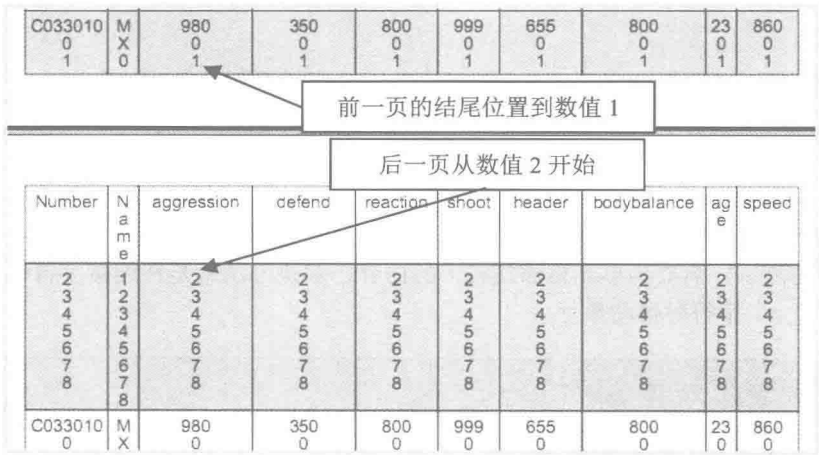


图 15.24　表格按页优先分页的显示效果

关键技术

本实例主要是通过 PdfPTable 类的 setSplitLate()方法，并将入口参数设置为 false，从而实现了表格按页优先分页的功能。其实现代码如下：

```
PdfPTable table = new PdfPTable(10);                    //定义表格
table.setSplitLate(false);                              //设置表格页优先分页方式
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PageFirstPagination 类，该类中只有一个 main()主方法，实现 PDF 表格中单元格的页优先分页功能，主要代码如下：

```
Document document = new Document();                     //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\页优先分页.pdf"));                          //关联文档对象与输出流
document.open();                                        //打开文档
String[] data = { "C033010", "MX", "980", "350", "800", "999",
        "655", "800", "23", "860" };                    //定义数据信息
PdfPTable table = new PdfPTable(10);                    //定义表格
int columnwidths[] = { 8, 3, 11, 10, 8, 6, 8, 12, 3, 6 };//定义列宽
table.setWidths(columnwidths);                          //向表格添加列宽
```

```
table.setWidthPercentage(100);                                  //向表格添加绝对宽度
table.getDefaultCell().setPadding(3);                           //设置单元格填充为 3
table.getDefaultCell().setHorizontalAlignment(Element.ALIGN_CENTER);  //设置单元格居中对齐
table.addCell("Number");                                        //将单元格内容顺次加入到表格，当一行充满时自动换行
//省略了添加表格标题的部分代码
table.setHeaderRows(1);                                         //为表格每一页设置表头
for (int i = 1; i < 100; i++) {                                 //循环向表格中添加 100 条记录
    if (i % 2 == 1) {
        table.getDefaultCell().setBackgroundColor(
                BaseColor.LIGHT_GRAY);                          //填充颜色
    } else {
        table.getDefaultCell().setBackgroundColor(BaseColor.WHITE);   //填充颜色
    }
    for (int x = 0; x < 10; x++) {
        String var = data[x];                                   //获得数组中的数据
        for (int y = 0; y < i; y++) {
            var += "\n" + y;                                    //连接字符串生成单元格内容
        }
        table.addCell(var);                                     //为单元格添加内容
    }
}
table.setSplitLate(false);                                      //设置表格页优先分页方式
document.add(table);                                            //向文档添加表格
document.close();                                               //关闭文档
```

秘笈心法

心法领悟 471：在单元格中添加多行内容。

为了方便本实例的测试，需要为单元格添加较多的内容，解决办法就是在连接字符串时使用“\n”字符，这样程序就会在遇到“\n”字符时自动换行。

实例 472　强行在一页显示

光盘位置：光盘\MR\472　　高级　　趣味指数：★★★

实例说明

如果 PdfPTable 表格中某一行的内容较多，而无法在当前页完全显示一整行内容，但是又不想进行分页，可以设置表格的单元格强行在一页显示，但是这种情况可能会丢失数据。本实例实现了使表格的单元格强行在一页显示的功能。运行程序，可以看到表格单元格强行在一页显示的效果，如图 15.25 所示。

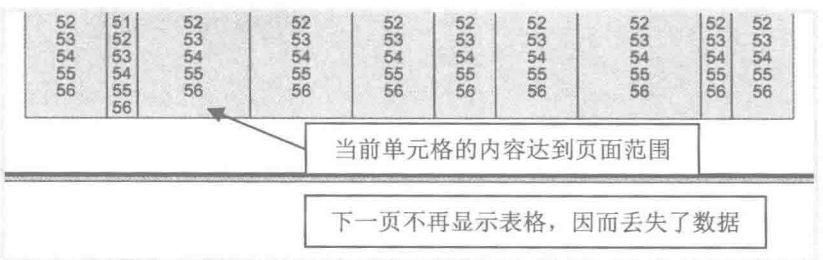


图 15.25　强行在一页显示表格单元格的效果

关键技术

本实例主要是通过 PdfPTable 类的 setSplitLate()方法，并将入口参数设置为 false，从而实现了表格的单元格

强行在一页显示的功能。其实现代码如下：

```
PdfPTable table = new PdfPTable(10);                    //定义表格
table.setSplitRows(false);                              //使行强行在一页显示，但是可能会丢失数据
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 ForceShowPage 类，该类中只有一个 main()主方法，实现了 PDF 表格中单元格强行在一页显示的功能，主要代码如下：

```
Document document = new Document();                     //创建文档对象
PdfWriter.getInstance(document, new FileOutputStream(
        "c:\\页优先分页.pdf"));                          //关联文档对象与输出流
document.open();                                        //打开文档
String[] data = { "C033010", "MX", "980", "350", "800", "999",
        "655", "800", "23", "860" };                    //定义数据信息
PdfPTable table = new PdfPTable(10);                    //定义表格
int columnwidths[] = { 8, 3, 11, 10, 8, 6, 8, 12, 3, 6 };  //定义列宽
table.setWidths(columnwidths);                          //向表格添加列宽
table.setWidthPercentage(100);                          //向表格添加绝对宽度
table.getDefaultCell().setPadding(3);                   //设置单元格填充为 3
table.getDefaultCell().setHorizontalAlignment(Element.ALIGN_CENTER);  //设置单元格居中对齐
table.addCell("Number");                                //将单元格内容顺次加入到表格，当一行充满时自动换行
//省略了添加表格标题的部分代码
table.setHeaderRows(1);                                 //为表格每一页设置表头
for (int i = 1; i < 100; i++) {                         //循环向表格中添加 100 条记录
    if (i % 2 == 1) {
        table.getDefaultCell().setBackgroundColor(
                BaseColor.LIGHT_GRAY);                  //填充颜色
    } else {
        table.getDefaultCell().setBackgroundColor(BaseColor.WHITE);  //填充颜色
    }
    for (int x = 0; x < 10; x++) {
        String var = data[x];                           //获得数组中的数据
        for (int y = 0; y < i; y++) {
            var += "\n" + y;                            //连接字符串生成单元格内容
        }
        table.addCell(var);                             //为单元格添加内容
    }
}
table.setSplitRows(false);                              //使行强行在一页显示，但是可能会丢失数据
document.add(table);                                    //向文档添加表格
document.close();                                       //关闭文档
```

秘笈心法

心法领悟 472：应避免使用强行在一页显示单元格功能。

由于强行在一页显示单元格可能会导致数据的丢失，所以在实际开发 PDF 文档时，如果没有特殊要求，应尽量避免这种设置，以防止造成不必要的数据丢失。

实例 473　绝对定位表格

光盘位置：光盘\MR\473　　高级　　趣味指数：★★★★

实例说明

可以根据需要，设置在 PDF 文档中添加的表格的位置。本实例实现了绝对定位表格的功能。运行程序，可以看到默认表格位置和绝对定位表格后的效果，如图 15.26 所示。

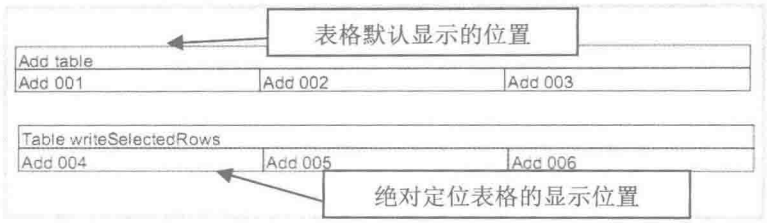


图 15.26　默认表格位置和绝对定位表格后的效果

关键技术

本实例主要是通过 PdfPTable 类的 writeSelectedRows()方法实现了绝对定位表格的功能。其实现代码如下：

```
table = new PdfPTable(3);                                    //定义新表格
table.writeSelectedRows(0, 2, 50, 750, writer.getDirectContent());   //在指定位置添加表格内容
```

说明：上面代码中，writeSelectedRows()方法的第 1 个参数是表格起始行的索引，第 2 个参数是结束行的索引（不包含），第 3 个参数和第 4 个参数是表格的位置坐标，最后一个参数是 PdfContentByte 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个 AbsoluteOrientationTable 类，该类中只有一个 main()主方法，实现了 PDF 表格的绝对定位功能，主要代码如下：

```
Document document = new Document();                          //创建文档对象
try {
    PdfWriter writer = PdfWriter.getInstance(document,
            new FileOutputStream("c:\\绝对定位表格.pdf"));       //关联文档对象与输出流
    document.open();                                         //打开文档
    float[] columnSize = { 21F, 21F, 21F };                  //设置列宽
    PdfPTable table = null;
    PdfPCell cell = null;
    table = new PdfPTable(columnSize);                       //定义新表格
    table.getDefaultCell().setBorder(1);                     //设置表格边框宽度
    table.setHorizontalAlignment(Element.ALIGN_CENTER);      //设置居中对齐
    table.setTotalWidth(500);                                //设置总宽度 500
    table.setLockedWidth(true);                              //锁定宽度
    cell = new PdfPCell(new Phrase("Add table"));            //定义单元格
    cell.setColspan(3);                                      //设置单元格跨度 3
    table.addCell(cell);                                     //向表格添加单元格
    table.addCell(new PdfPCell(new Phrase("Add 001")));      //向表格添加内容
    table.addCell(new PdfPCell(new Phrase("Add 002")));      //向表格添加内容
    table.addCell(new PdfPCell(new Phrase("Add 003")));      //向表格添加内容
    document.add(table);                                     //向文档添加表格
    table = new PdfPTable(columnSize);                       //定义新表格
    table.getDefaultCell().setBorder(1);                     //设置表格边框宽度
    table.setHorizontalAlignment(Element.ALIGN_CENTER);      //设置居中对齐
    table.setTotalWidth(500);                                //设置总宽度 500
    table.setLockedWidth(true);                              //锁定宽度
    cell = new PdfPCell(new Phrase("Table writeSelectedRows"));  //定义单元格
    cell.setColspan(columnSize.length);                      //设置单元格跨度 3
    table.addCell(cell);                                     //向表格添加单元格
    table.addCell(new PdfPCell(new Phrase("Add 004")));      //向表格添加内容
    table.addCell(new PdfPCell(new Phrase("Add 005")));      //向表格添加内容
    table.addCell(new PdfPCell(new Phrase("Add 006")));      //向表格添加内容
    table.writeSelectedRows(0, 2, 50, 750, writer.getDirectContent());  //在指定位置添加表格内容
    document.close();                                        //关闭文档
```

```
} catch (DocumentException de) {
}
```

秘笈心法

心法领悟 473：使用 PdfPTable 类的 setTotalWidth()方法解决程序出错问题。

在使用本实例绝对定位表格时，如果没有使用 setTotalWidth()方法设置表格的宽度，程序会出错，解决办法就是使用 setTotalWidth()方法设置表格的宽度，再指定表格的位置。

实例 474　大表格的内存处理

光盘位置：光盘\MR\474　　高级　　趣味指数：★★★★

实例说明

在 PDF 文档中添加的表格，如果表格的内容非常多，就会占用很多的系统资源，为此可以对大表格进行内存处理，从而减少不必要的资源浪费。本实例实现了对大表格的内存处理。运行程序，对有 7 页内容的大表格进行内存处理的效果如图 15.27 所示。

10%	20%	10%	20%	10%	30%
1	明日科技	1	明日科技	2	明日科技
448	明日科技	448	明日科技	449	明日科技
449	明日科技	450	明日科技	450	明日科技
451	明日科技	451	明日科技	452	明日科技
452	明日科技	453	明日科技	453	明日科技
454	明日科技	454	明日科技	455	明日科技
455	明日科技	456	明日科技	456	明日科技
457	明日科技	457	明日科技	458	明日科技
458	明日科技	459	明日科技	459	明日科技
460	明日科技	460	明日科技	461	明日科技
461	明日科技	462	明日科技	462	明日科技
463	明日科技	463	明日科技	464	明日科技
464	明日科技	465	明日科技	465	明日科技
466	明日科技	466	明日科技	467	明日科技
467	明日科技	468	明日科技	468	明日科技
469	明日科技	469	明日科技	470	明日科技
470	明日科技	471	明日科技	471	明日科技
472	明日科技	472	明日科技	473	明日科技
473	明日科技	474	明日科技	474	明日科技
475	明日科技	475	明日科技	476	明日科技
476	明日科技	477	明日科技	477	明日科技
478	明日科技	478	明日科技	479	明日科技
479	明日科技	480	明日科技	480	明日科技
481	明日科技	481	明日科技	482	明日科技
482	明日科技	483	明日科技	483	明日科技
484	明日科技	484	明日科技	485	明日科技
485	明日科技	486	明日科技	486	明日科技
487	明日科技	487	明日科技	488	明日科技
488	明日科技	489	明日科技	489	明日科技
490	明日科技	490	明日科技	491	明日科技
491	明日科技	492	明日科技	492	明日科技
493	明日科技	493	明日科技	494	明日科技
494	明日科技	495	明日科技	495	明日科技
496	明日科技	496	明日科技	497	明日科技
497	明日科技	498	明日科技	498	明日科技

图 15.27　对大表格进行内存处理的效果

关键技术

本实例主要是通过 PdfPTable 类的 deleteBodyRows()方法删除表格的多余行，然后再向表格中添加单元格，

并重新将表格添加到文档中，相当于在一个文档中添加了多个表格，从而实现了对大表格的内存处理功能，其实现代码如下：

```
for (int i = 1; i <= 500; i++) {                                    //循环向表格中添加 500 条记录
    if (i % bigtablesize == 4) {                                    //求余
        document.add(table);                                        //向文档添加表格
        table.deleteBodyRows();                                     //删除多余行
        table.setSkipFirstHeader(true);                             //使表头始终保持在首行
    }
    PdfPCell cell0 = new PdfPCell(new Paragraph(String.valueOf(i), fontChinese1));  //向单元格添加内容
    table.addCell(cell0);                                           //向表格添加单元格
    PdfPCell cell1 = new PdfPCell(new Paragraph("明日科技", fontChinese1));         //向单元格添加内容
    table.addCell(cell1);                                           //向表格添加单元格
    //省略了部分向表格中添加其他单元格的代码
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 BigTableEMSHandle 类，该类中只有一个 main()主方法，实现了 PDF 表格的大表格内存处理功能，主要代码如下：

```
int bigtablesize = 5;
Document document = new Document();                                 //创建文档对象
try {
    PdfWriter.getInstance(document, new FileOutputStream(
            "c:\\大表格的内存处理.pdf"));                            //关联文档对象与输出流
    document.open();                                                //打开文档

    BaseFont chinese = BaseFont.createFont("STSong-Light",
            "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);                 //定义基础字体
    Font fontChinese1 = new Font(chinese, 10, Font.NORMAL,
            new BaseColor(90, 90, 90));                             //实例化字体类与设置字体大小属性
    Font fontChinese2 = new Font(chinese, 15, Font.NORMAL,
            BaseColor.BLUE);                                        //实例化字体类与设置字体大小属性
    document.add(new Paragraph("大表格的内存管理\n\n", fontChinese2));
    float[] hw = { 0.1f, 0.2f, 0.1f, 0.2f, 0.1f, 0.3f };           //设置列宽
    PdfPTable table = new PdfPTable(hw);                            //创建表格
    table.setHeaderRows(2);                                         //设置头排
    table.addCell("10%");                                           //将单元格内容顺次加入到表格，当一行充满时自动换行
    table.addCell("20%");
    table.addCell("10%");
    table.addCell("20%");
    table.addCell("10%");
    table.addCell("30%");
    for (int i = 1; i <= 500; i++) {                                //循环向表格中添加 500 条记录
        if (i % bigtablesize == 4) {                                //求余
            document.add(table);                                    //向文档添加表格
            table.deleteBodyRows();                                 //删除多余行
            table.setSkipFirstHeader(true);                         //使表头始终保持在首行
        }
        PdfPCell cell0 = new PdfPCell(new Paragraph(String.valueOf(i), fontChinese1)); //向单元格添加内容
        table.addCell(cell0);                                       //向表格添加单元格
        PdfPCell cell1 = new PdfPCell(new Paragraph("明日科技", fontChinese1));         //向单元格添加内容
        table.addCell(cell1);                                       //向表格添加单元格
        PdfPCell cell2 = new PdfPCell(new Paragraph(String.valueOf(i), fontChinese1)); //向单元格添加内容
        table.addCell(cell2);                                       //向表格添加单元格
        PdfPCell cell3 = new PdfPCell(new Paragraph("明日科技", fontChinese1));         //向单元格添加内容
        table.addCell(cell3);                                       //向表格添加单元格
    }
```

```
        document.close();                                    //关闭文档
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
```

秘笈心法

心法领悟 474：避免进行大表格的内存处理时，在一页中出现多次表头。

在进行大表格的内存处理时，如果不使用 PdfPTable 类的 setSkipFirstHeader()方法使表头始终保持在首行，在 PDF 文档中就会出现在一页中出现多次表头的情况，可以通过如下代码解决：

```
table.setSkipFirstHeader(true);                              //使表头始终保持在首行
```

第16章

设置阅读器参数

- 设置页面参数
- 设置工具栏和全屏模式参数

16.1　设置页面参数

实例 475	只显示一个页面 光盘位置：光盘\MR\475	初级 趣味指数：★★★

实例说明

在查看 PDF 文档时，如果只希望看到一整页信息，而不显示其他页的内容，可以通过设置阅读器来实现，本实例实现了该功能。运行程序，将看到在阅读器中只显示了一页内容，而不会出现其他页面的部分内容，假设文档有两页内容，则效果如图 16.1 和图 16.2 所示，如果用户通过拖动滚动条查看页面内容，则在页面中只能显示图 16.1 和图 16.2 中的内容，而不会每一页都显示一部分内容。

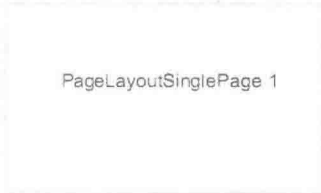


图 16.1　文档中第 1 页的内容

图 16.2　文档中第 2 页的内容

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageLayoutSinglePage 来实现只显示一个页面的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\只显示一个页面.pdf"));      //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageLayoutSinglePage);                                            //设置阅读器只显示一个页面
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 OnlyShowOnePage 类，该类中只有一个 main()主方法，实现在 PDF 阅读器中只显示一个页面的功能。主方法的主要代码如下：

```
Document document = new Document();                                          //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\只显示一个页面.pdf"));                      //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageLayoutSinglePage);                 //设置阅读器只显示一个页面
document.open();                                                             //打开文档
document.add(new Paragraph("PageLayoutSinglePage 1"));                       //向文档中添加内容
document.newPage();                                                          //添加新页
document.add(new Paragraph("PageLayoutSinglePage 2"));                       //向文档中添加内容
document.close();                                                            //关闭文档
```

秘笈心法

心法领悟 475：为文档添加多页内容的两种常用方法。

为了测试在阅读器中只显示一个页面的功能，文档中至少要有两页内容，因此要为文档添加页码，最简单的方法是使用 Document 类的 newPage()方法；另一种方法是向文档中添加的内容较多时，如果一页无法完全显示，则剩下的内容会自动到下页显示，从而也可以实现为文档添加多页内容。

实例 476	单列显示 光盘位置：光盘\MR\476	初级 趣味指数：★★

实例说明

在使用阅读器查看 PDF 文档时，可以根据需要使阅读器页面中只显示一列文档，也可以显示两列文档。本实例实现了使阅读器页面显示一列文档的功能，即实现了单列显示功能，效果如图 16.3 所示。

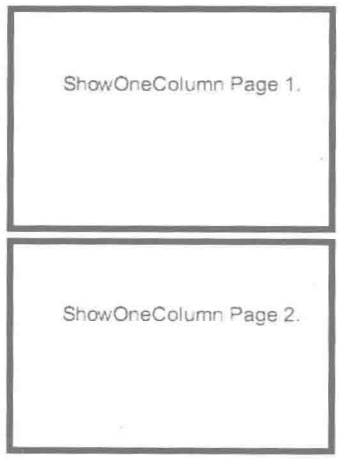


图 16.3　单列显示效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageLayout OneColumn 实现单列显示的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\单列显示.pdf"));      //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageLayoutOneColumn);                                        //设置阅读器单列显示
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 ShowPageLayoutOneColumn 类，该类中只有一个 main()主方法，实现 PDF 文档在阅读器中单列显示的功能。主方法的主要代码如下：

```
Document document = new Document();                                          //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\单列显示.pdf"));                           //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageLayoutOneColumn);                  //设置阅读器单列显示
document.open();                                                             //打开文档
document.add(new Paragraph("ShowOneColumn Page 1."));                        //向文档中添加内容
document.newPage();                                                          //增加新页
document.add(new Paragraph("ShowOneColumn Page 2."));                        //向文档中添加内容
document.close();                                                            //关闭文档
```

秘笈心法

心法领悟 476：双页显示并且单页在右。

使用阅读器查看 PDF 文档时，可以使用双页显示并且单页在右的方式查看文档内容，方法是使用 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageLayoutTwoRight。

实例 477	双列显示奇页在左 光盘位置：光盘\MR\477	初级 趣味指数：★★

实例说明

在使用阅读器查看 PDF 文档时，可以根据需要使阅读器页面显示两列文档，并使奇数页在左侧显示，本实例实现了该功能，效果如图 16.4 所示。

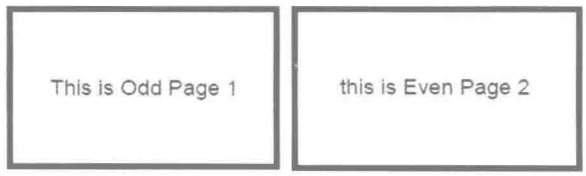


图 16.4　双列显示奇页在左的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageLayoutTwoColumnLeft 实现双列显示奇页在左的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\双列显示奇页在左.pdf"));   //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageLayoutTwoColumnLeft);                                          //设置阅读器双列显示奇页在左
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 ShowPageLayoutTwoColumnLeft 类，该类中只有一个 main()主方法，实现在 PDF 阅读器中双列显示奇页在左的功能，主方法的主要代码如下：

```
Document document = new Document();                                          //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\双列显示奇页在左.pdf"));                    //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageLayoutTwoColumnLeft);              //设置阅读器双列显示奇页在左
document.open();                                                             //打开文档
document.add(new Paragraph("This is Odd Page 1"));                           //向文档中添加内容
document.newPage();                                                          //新增第 2 页
document.add(new Paragraph("this is Even Page 2"));                          //向文档中添加内容
document.newPage();                                                          //新增第 3 页
document.add(new Paragraph("This is Odd Page 3"));                           //向文档中添加内容
document.newPage();                                                          //新增第 4 页
document.add(new Paragraph("This is Even Page 4"));                          //向文档中添加内容
document.close();                                                            //关闭文档
```

秘笈心法

心法领悟 477：双页显示并且单页在左。

使用阅读器查看 PDF 文档时，可以使用双页显示单页在左的方式查看文档内容，方法是使用 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageLayoutTwoLeft。

实例 478 双列显示奇页在右

光盘位置：光盘\MR\478　　初级　　趣味指数：★★★

实例说明

在使用阅读器查看 PDF 文档时，可以根据需要使阅读器页面显示两列文档，并使奇数页在右侧显示。本实例实现了该功能，效果如图 16.5 所示。

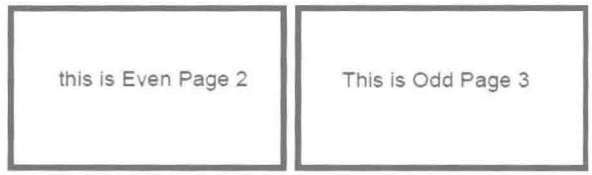


图 16.5　双列显示奇页在右的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageLayoutTwoColumnRight 实现双列显示奇页在右的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\双列显示奇页在右.pdf")); //关联文档与输出流
writer.setViewerPreferences(PdfWriter.PageLayoutTwoColumnRight);                                    //设置阅读器双列显示奇页在右
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 ShowPageLayoutTwoColumnRight 类，该类中只有一个 main()主方法，实现在 PDF 阅读器中双列显示奇页在右的功能。主方法的主要代码如下：

```
Document document = new Document();                                          //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\双列显示奇页在右.pdf"));                    //关联文档与输出流
writer.setViewerPreferences(PdfWriter.PageLayoutTwoColumnRight);             //设置阅读器双列显示奇页在右
document.open();                                                             //打开文档
document.add(new Paragraph("This is Odd Page 1"));                           //向文档中添加内容
document.newPage();                                                          //新增第 2 页
document.add(new Paragraph("this is Even Page 2"));                          //向文档中添加内容
document.newPage();                                                          //新增第 3 页
document.add(new Paragraph("This is Odd Page 3"));                           //向文档中添加内容
document.newPage();                                                          //新增第 4 页
document.add(new Paragraph("This is Even Page 4"));                          //向文档中添加内容
document.close();
```

秘笈心法

心法领悟 478：实现本地跳转。

本地跳转是指单击文档中的本地链接，可以跳转到文档中的目标链接。通过本地跳转，可以方便地在相关页面之间查看，实现本地跳转的代码如下：

```
Chunk local = new Chunk("goto destination.").setLocalGoto("link");           //创建本地语句块，用于跳转到目标语句块
Chunk destination = new Chunk("destination.").setLocalDestination("link");   //创建目标语句块
document.add(local);                                                         //在文档中添加语句块
document.newPage();                                                          //创建新页
document.add(destination);                                                   //在文档的新页中添加语句块
```

实例 479	显示大纲 实例位置：光盘\MR\479	高级 趣味指数：★★★★★

实例说明

在使用阅读器查看 PDF 文档时，可以根据需要使阅读器以大纲模式显示页面内容。本实例实现了以大纲模式显示页面内容的功能，效果如图 16.6 所示。

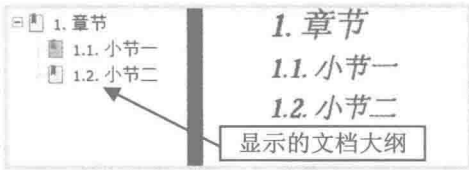


图 16.6　以大纲模式显示页面内容的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageModeUseOutlines 实现使阅读器以大纲模式显示文档内容的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\显示大纲.pdf"));      //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeUseOutlines);                                        //设置阅读器显示大纲
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 ShowPageModeUseOutlines 类，该类中只有一个 main()主方法，实现在 PDF 阅读器中以大纲模式显示文档内容的功能。主方法的主要代码如下：

```
Document document = new Document();                                                                //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\显示大纲.pdf"));                                                  //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeUseOutlines);                                        //设置阅读器显示大纲
document.open();                                                                                   //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);     //定义基础字体
Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,BaseColor.RED);                          //实例化字体类、设置字体大小和颜色
Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,BaseColor.BLUE);                         //实例化字体类、设置字体大小和颜色
Paragraph paragraph = new Paragraph("章节",fontChinese1);                                          //创建段落对象
Chapter chapter = new Chapter(paragraph,1);                                                        //创建章节对象
paragraph = new Paragraph("小节一",fontChinese2);                                                  //创建段落对象
chapter.addSection(paragraph);                                                                     //添加小节
paragraph = new Paragraph("小节二",fontChinese2);                                                  //创建段落对象
chapter.addSection(paragraph);                                                                     //添加小节
document.add(chapter);                                                                             //向文档添加章节
document.close();                                                                                  //关闭文档
```

秘笈心法

心法领悟 479：实现在不同的 PDF 文档中异地跳转。

有时需要在不同的 PDF 文档中通过链接进行相互跳转，这可以使用异地跳转功能实现。实现从当前文档异地跳转到 c:/aaa.pdf 文档第 5 页的代码如下：

```
Chunk chunk = new Chunk("to c:/aaa.pdf   page 5.").setRemoteGoto("c:/aaa.pdf", 5);     //创建实现异地跳转到 c:/aaa.pdf 文档第 5 页的语句块
document.add(chunk);                                                                   //在文档中添加语句块
```

实例 480 显示缩略图

光盘位置：光盘\MR\480

中级

趣味指数：★★★

实例说明

在使用阅读器查看 PDF 文档时，可以根据需要使阅读器以缩略图模式显示页面内容。本实例实现了以缩略图模式显示页面内容的功能，效果如图 16.7 所示。

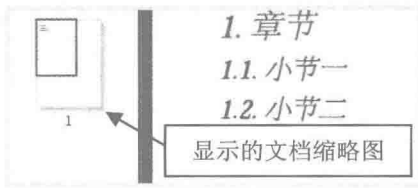


图 16.7 显示缩略图的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageModeUseThumbs 实现使阅读器显示缩略图的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\显示缩略图.pdf"));     //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeUseThumbs);                                          //设置阅读器显示缩略图
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 ShowPageModeUseThumbs 类，该类中只有一个 main()主方法，实现在 PDF 阅读器中显示缩略图的功能。主方法的主要代码如下：

```
Document document = new Document();                                                                //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\显示缩略图.pdf"));                                               //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeUseThumbs);                                          //设置阅读器显示缩略图
document.open();                                                                                   //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);     //定义基础字体
Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,BaseColor.RED);                          //实例化字体类、设置字体大小和颜色
Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,BaseColor.BLUE);                         //实例化字体类、设置字体大小和颜色
Paragraph paragraph = new Paragraph("章节",fontChinese1);                                          //创建段落对象
Chapter chapter = new Chapter(paragraph,1);                                                        //创建章节对象
paragraph = new Paragraph("小节一",fontChinese2);                                                  //创建段落对象
chapter.addSection(paragraph);                                                                     //添加小节
paragraph = new Paragraph("小节二",fontChinese2);                                                  //创建段落对象
chapter.addSection(paragraph);                                                                     //添加小节
document.add(chapter);                                                                             //向文档添加章节
document.close();                                                                                  //关闭文档
```

秘笈心法

心法领悟 480：运行 Windows 应用程序。

如果需要在 PDF 文档中运行 Windows 应用程序，例如运行“计算器”程序，可以先用 PdfAction 类有 4 个参数的构造方法创建计算器的 PdfAction 对象，然后创建语句块对象并链接到该 PdfAction 对象，从而实现运行 Windows 应用程序的功能。运行 Windows 7“计算器”的代码如下：

```
PdfAction pdfAction = new PdfAction("c:/Windows/system32/calc.exe", null, null, null);//创建 PdfAction 对象，用于启动 Windows 7 自带的“计算器”
```

```
Chunk chunk = new Chunk("Run calc.exe file.").setAction(pdfAction);    //创建运行应用程序的语句块
document.add(chunk);                                                   //在文档中添加语句块
```

实例 481　不显示大纲和缩略图

光盘位置：光盘\MR\481　　初级　　趣味指数：★★★

实例说明

在使用阅读器查看 PDF 文档时，可以根据需要使阅读器不显示大纲和缩略图。本实例实现了使阅读器不显示大纲和缩略图的功能，效果如图 16.8 所示。

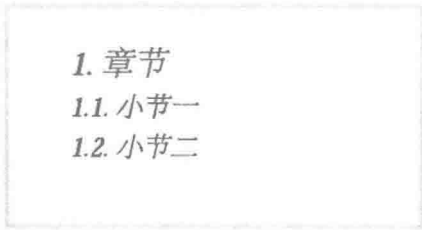


图 16.8　不显示大纲和缩略图的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageMode UseNone 实现使阅读器不显示大纲和缩略图的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\不显示大纲和缩略图.pdf")); //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeUseNone);                                                //设置阅读器不显示大纲和缩略图
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 PageModeUseNone 类，该类中只有一个 main()主方法，实现在 PDF 阅读器中不显示大纲和缩略图的功能。主方法的主要代码如下：

```
Document document = new Document();                                                                    //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\不显示大纲和缩略图.pdf"));                                            //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeUseNone);                                                //设置阅读器不显示大纲和缩略图
document.open();                                                                                       //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);         //定义基础字体
Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,BaseColor.RED);                              //实例化字体类、设置字体大小和颜色
Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,BaseColor.BLUE);                             //实例化字体类、设置字体大小和颜色
Paragraph paragraph = new Paragraph("章节",fontChinese1);                                              //创建段落对象
Chapter chapter = new Chapter(paragraph,1);                                                            //创建章节对象
paragraph = new Paragraph("小节一",fontChinese2);                                                      //创建段落对象
chapter.addSection(paragraph);                                                                         //添加小节
paragraph = new Paragraph("小节二",fontChinese2);                                                      //创建段落对象
chapter.addSection(paragraph);                                                                         //添加小节
document.add(chapter);                                                                                 //向文档添加章节
document.close();                                                                                      //关闭文档
```

秘笈心法

心法领悟 481：通过 PDF 文档运行指定的文件。

如果希望通过 PDF 文档来运行指定的文件，可以使用 PdfAction 类有一个参数的构造方法，创建一个到指

定文件的 PdfAction 对象，然后通过语句块链接到该 PdfAction 对象，这样即可通过 PDF 文档运行指定的文件，例如，运行 c:/User.log 文件的代码如下：

```
PdfAction pdfAction = new PdfAction("c:/User.log");                    //创建 PdfAction 对象，用于运行 c:/User.log 文件
Chunk chunk = new Chunk("Run c:/User.log file.").setAction(pdfAction); //创建运行指定文件的语句块
document.add(chunk);                                                   //在文档中添加语句块
```

实例 482　全屏显示

光盘位置：光盘\MR\482　　中级　　趣味指数：★★★

实例说明

在使用阅读器查看 PDF 文档时，可以在阅读器中以全屏模式显示 PDF 文档，本实例实现了该功能。运行程序后，使用阅读器查看 PDF 文档时，文档将占据整个屏幕，效果如图 16.9 所示。

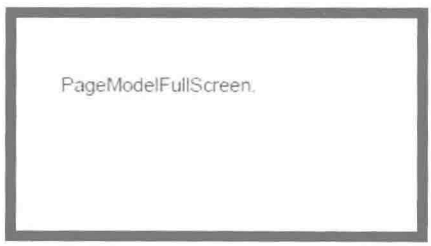


图 16.9　以全屏模式显示 PDF 文档的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageModeFullScreen 实现使阅读器以全屏模式显示 PDF 文档的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\全屏显示.pdf"));  //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeFullScreen);                                     //设置阅读器以全屏模式显示
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 FullScreen 类，该类中只有一个 main()主方法，实现在 PDF 阅读器中以全屏模式显示 PDF 文档的功能。主方法的主要代码如下：

```
Document document = new Document();                                    //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\全屏显示.pdf"));                     //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeFullScreen);             //设置阅读器以全屏模式显示
document.open();                                                       //打开文档
document.add(new Paragraph("PageModelFullScreen."));                   //向文档中添加内容
document.close();                                                      //关闭文档
```

秘笈心法

心法领悟 482：通过锚点实现内部链接。

在 Java 中，可以使用 iText 工具包的 Anchor 类创建锚点对象，然后为其指定 name 属性，作为链接的目的地，接着再创建一个锚点对象，然后指定该锚点对象的 reference 属性为前一个锚点的 name 属性，从而可以实现通过锚点完成内容链接，代码如下：

```
Anchor anchor1 = new Anchor("internal link.");  //创建 Anchor 对象
anchor1.setName("link");                        //指定 Anchor 对象的 name 属性为 link
```

```
Anchor anchor2 = new Anchor("click goto internal link.");        //创建 Anchor 对象
anchor2.setReference("#link");                                   //指定 Anchor 对象的引用为 link
document.add(anchor2);                                           //在文档中添加 Anchor 对象
document.newPage();                                              //创建新页
document.add(anchor1);                                           //在文档中添加 Anchor 对象
```

16.2　设置工具栏和全屏模式参数

实例 483　显示和隐藏工具栏

光盘位置：光盘\MR\483　　初级　　趣味指数：★★

实例说明

在使用阅读器查看 PDF 文档时，为了节省页面空间，可以在查看文档时隐藏阅读器的工具。本实例实现了隐藏 PDF 阅读器工具栏的功能。运行程序，效果如图 16.10 所示。

图 16.10　隐藏 PDF 阅读器工具栏的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.HideToolbar 实现隐藏 PDF 阅读器工具栏的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\显示和隐藏工具栏.pdf"));    //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.HideToolbar);                                                    //设置阅读器隐藏工具栏
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 ShowAndHideToolbar 类，该类中只有一个 main()主方法，实现在查看 PDF 文档时隐藏 PDF 阅读器工具栏的功能。主方法的主要代码如下：

```
Document document = new Document();                              //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\显示和隐藏工具栏.pdf"));        //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.HideToolbar);              //设置阅读器隐藏工具栏
document.open();                                                 //打开文档
document.add(new Paragraph("HideToolbar."));                     //向文档中添加内容
document.close();                                                //关闭文档
```

秘笈心法

心法领悟 483：通过锚点实现外部链接。

在 Java 中，可以使用 iText 工具包的 Anchor 类创建锚点对象，然后通过锚点对象的 reference 属性指定外部链接，实现通过锚点访问外部链接的功能，实现代码如下：

```
Anchor anchor = new Anchor("www.hao123.com");                    //创建 Anchor 对象
anchor.setReference("http://www.hao123.com");                    //指定外部链接
document.add(anchor);                                            //在文档中添加 Anchor 对象
```

实例 484 显示和隐藏菜单

光盘位置：光盘\MR\484

初级

趣味指数：★★★

实例说明

在使用阅读器查看 PDF 文档时，为了节省页面空间，可以在查看文档时隐藏阅读器的菜单。本实例实现了隐藏 PDF 阅读器菜单的功能。运行程序，效果如图 16.11 所示。

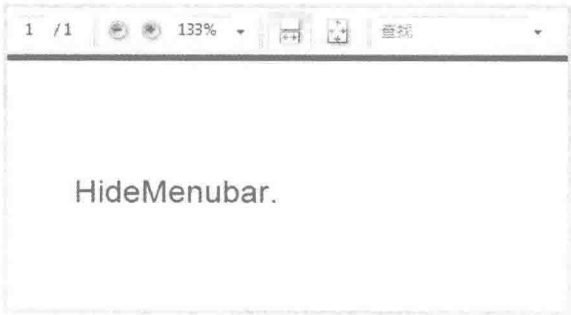


图 16.11　隐藏 PDF 阅读器菜单的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.HideMenubar 实现隐藏 PDF 阅读器菜单的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\显示和隐藏菜单.pdf"));    //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.HideMenubar);                                                  //设置阅读器隐藏菜单
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 ShowAndHideMenubar 类，该类中只有一个 main()主方法，实现在查看 PDF 文档时隐藏 PDF 阅读器菜单的功能。主方法的主要代码如下：

```
Document document = new Document();                                  //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\显示和隐藏菜单.pdf"));              //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.HideMenubar);                  //设置阅读器隐藏菜单
document.open();                                                     //打开文档
document.add(new Paragraph("HideMenubar."));                         //向文档中添加内容
document.close();                                                    //关闭文档
```

秘笈心法

心法领悟 484：为 PDF 文档添加文本注释。

在使用 iText 生成 PDF 文档时，可以根据需要生成相应的文本注释，用于对文档中的内容进行说明。实现文本注释可以使用 Annotation 类，实现代码如下：

```
Annotation annotation = new Annotation("Text Annotation","This is a simple text annotation.");    //创建注释对象
document.add(annotation);                                                                          //将注释添加到文档中
```

实例 485	显示和隐藏页面元素 光盘指引：光盘\MR\485	中级 趣味指数：★★★

实例说明

在使用阅读器查看 PDF 文档时，如果文档的内容较多，无法在一页显示，就会显示滚动条等页面元素，本实例实现了隐藏 PDF 阅读器页面元素的功能。运行程序，可以看到文档中内容较多时，隐藏了 PDF 阅读器中滚动条等页面元素的效果，如图 16.12 所示。

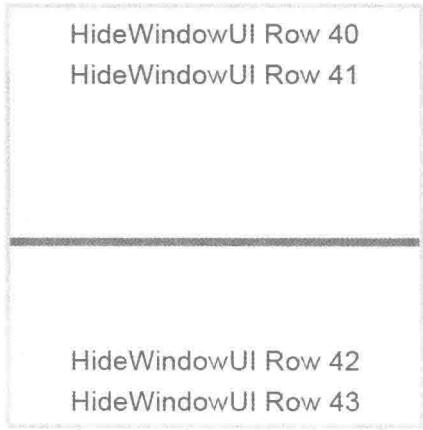


图 16.12　隐藏 PDF 阅读器页面元素的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.HideWindowUI 实现隐藏 PDF 阅读器页面元素的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\显示和隐藏页面元素.pdf"));      //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.HideWindowUI);                                                       //设置阅读器隐藏界面元素
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 ShowAndHideWindowUI 类，该类中只有一个 main()主方法，实现在查看 PDF 文档时隐藏 PDF 阅读器中页面元素的功能。主方法的主要代码如下：

```
Document document = new Document();                                                    //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\显示和隐藏页面元素.pdf"));                            //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.HideWindowUI);                                   //设置阅读器隐藏界面元素
document.open();                                                                       //打开文档
for (int i = 1; i <= 100; i++) {
    document.add(new Paragraph("HideWindowUI Row " + i));                              //向文档中添加段落内容
}
document.close();                                                                      //关闭文档
```

秘笈心法

心法领悟 485：为 PDF 文档添加外部注释。

在使用 iText 生成 PDF 文档时，可以根据需要为 PDF 文档添加外部注释。外部注释即指为文档添加一个矩

形区域，当用户单击该矩形区域时，就会打开外部注释所链接的网站。实现外部注释的代码如下：

```
Annotation annotation = new Annotation(10f, 10f, 500f, 820f, "http://www.baidu.com");   //创建外部注释
document.add(annotation);                                                               //为文档添加外部注释
```

实例 486　使文档窗口适合显示第一页

初级

光盘位置：光盘\MR\486

趣味指数：★★★

实例说明

在使用阅读器查看 PDF 文档时，如果文档的纸张较大，则无法在一页完全显示，这时可以将 PDF 阅读器设置为适合显示第一页，本实例实现了该功能。运行程序，效果如图 16.13 所示。

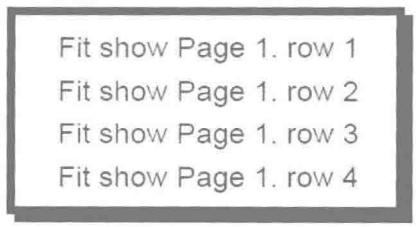


图 16.13　设置 PDF 阅读器适合显示第一页的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.FitWindow 实现使 PDF 阅读器适合显示第一页的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\使文档窗口适合显示第一页.pdf")); //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.FitWindow);                                                          //设置阅读器适合显示第一页
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 FitWindowFirstPage 类，该类中只有一个 main()主方法，实现使 PDF 阅读器适合显示第一页的功能，主方法的主要代码如下：

```
Document document = new Document();                                    //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\使文档窗口适合显示第一页.pdf"));      //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.FitWindow);                      //设置阅读器适合显示第一页
document.open();                                                       //打开文档
for (int i = 1; i <= 100; i++) {
    document.add(new Paragraph("Fit show Page 1. row " + i));          //向文档中添加段落内容
}
document.close();                                                      //关闭文档
```

秘笈心法

心法领悟 486：添加外部 PDF 文档链接注释。

在使用 iText 生成 PDF 文档时，可以根据需要为 PDF 文档添加外部 PDF 文档链接注释。外部 PDF 文档链接注释即指为文档添加一个矩形区域，当用户单击该矩形区域时，就会打开外部 PDF 文档链接注释所链接的 PDF 文档的指定页。实现外部 PDF 文档链接注释的代码如下：

```
Annotation annotation = new Annotation(10f, 10f, 500f, 820f, "c:/aaa.pdf",3);   //创建外部 PDF 文档链接注释，用于链接到 c:/aaa.pdf 的第 3 页
document.add(annotation);                                                       //为文档添加外部注释
```

实例 487　在屏幕中央显示文档窗口

光盘位置：光盘\MR\487　　中级　　趣味指数：★★

实例说明

在使用阅读器查看 PDF 文档时，阅读器窗口将显示在上次关闭的位置，如果需要，可以通过程序设置阅读器在每次打开 PDF 文档时都显示在屏幕中央。本实例实现了使 PDF 阅读器显示在屏幕中央的功能，运行程序，效果如图 16.14 所示。

图 16.14　在屏幕中央显示阅读器

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.CenterWindow 实现使 PDF 阅读器显示在屏幕中央的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\在屏幕中央显示文档窗口.pdf"));   //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.CenterWindow);                                                          //设置阅读器在屏幕中央显示
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 CenterWindow 类，该类中只有一个 main()主方法，实现使 PDF 阅读器在屏幕中央显示的功能。主方法的主要代码如下：

```
Document document = new Document();                                          //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\在屏幕中央显示文档窗口.pdf"));              //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.CenterWindow);                         //设置阅读器在屏幕中央显示
document.open();                                                             //打开文档
document.add(new Paragraph("ShowCenterWindow"));                             //向文档中添加内容
document.close();                                                            //关闭文档
```

秘笈心法

心法领悟 487：添加指定行为链接注释，以进入文档的下一页。

在使用 iText 生成 PDF 文档时，可以根据需要为 PDF 文档添加指定行为链接注释。指定行为链接注释即指为文档添加一个矩形区域，当用户单击该矩形区域时，就会按注释所指定的行为执行操作。例如，实现通过指

定行为链接注释从当前页链接到下一页的代码如下：

```
Annotation annotation = new Annotation(10f, 10f, 500f, 820f, PdfAction.NEXTPAGE);   //创建指定行为链接注释，用于链接到当前页的下一页
document.add(annotation);                                                           //为文档添加指定行为链接注释
```

实例 488　全屏模式下显示大纲

光盘位置：光盘\MR\488　　初级　　趣味指数：★★★

实例说明

在使用阅读器查看 PDF 文档时，如果希望在全屏模式下显示大纲，可以通过本实例讲解的方法设置。运行程序，可以看到 PDF 阅读器在全屏模式下显示大纲的效果，如图 16.15 所示。

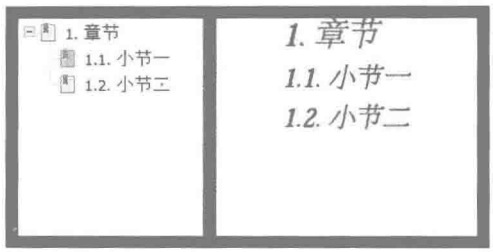


图 16.15　在全屏模式下显示大纲的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageModeFullScreen，即设置为全屏模式，然后再使用该方法将参数设置为 PdfWriter.NonFullScreenPageModeUseOutlines 实现使 PDF 阅读器在全屏模式下显示大纲的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\全屏模式下显示大纲.pdf")); //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeFullScreen);                                            //设置为全屏模式
writer.setViewerPreferences(PdfWriter.NonFullScreenPageModeUseOutlines);                              //设置阅读器在全屏模式下显示大纲
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 NonFullScreenPageModeUseOutlines 类，该类中只有一个 main()主方法，实现使 PDF 阅读器在全屏模式下显示大纲的功能。主方法的主要代码如下：

```
Document document = new Document();                                                                   //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\全屏模式下显示大纲.pdf"));                                          //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeFullScreen);                                            //设置为全屏模式
writer
        .setViewerPreferences(PdfWriter.NonFullScreenPageModeUseOutlines);                            //设置阅读器在全屏模式下显示大纲
document.open();                                                                                      //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);        //定义基础字体
Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,BaseColor.RED);                             //实例化字体类、设置字体大小和颜色
Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,BaseColor.BLUE);                            //实例化字体类、设置字体大小和颜色
Paragraph paragraph = new Paragraph("章节",fontChinese1);                                             //创建段落对象
Chapter chapter = new Chapter(paragraph,1);                                                           //创建章节对象
paragraph = new Paragraph("小节一",fontChinese2);                                                     //创建段落对象
chapter.addSection(paragraph);                                                                        //添加小节
paragraph = new Paragraph("小节二",fontChinese2);                                                     //创建段落对象
chapter.addSection(paragraph);                                                                        //添加小节
document.add(chapter);                                                                                //向文档添加章节
document.close();                                                                                     //关闭文档
```

秘笈心法

心法领悟 488：添加指定行为链接注释，以进入文档的前一页。

实现通过指定行为链接注释从当前页链接到前一页的代码如下：

```
Annotation annotation = new Annotation(10f, 10f, 500f, 820f, PdfAction.PREVPAGE);   //创建指定行为链接注释，用于链接到当前页的前一页
document.add(annotation);                                                           //为文档添加指定行为链接注释
```

实例 489　全屏模式下显示缩略图

光盘位置：光盘\MR\489　　高级　　趣味指数：★★★★★

实例说明

在使用阅读器查看 PDF 文档时，如果希望在全屏模式下显示缩略图，可以通过本实例讲解的方法实现。运行程序，可以看到 PDF 阅读器在全屏模式下显示缩略图的效果，如图 16.16 所示。

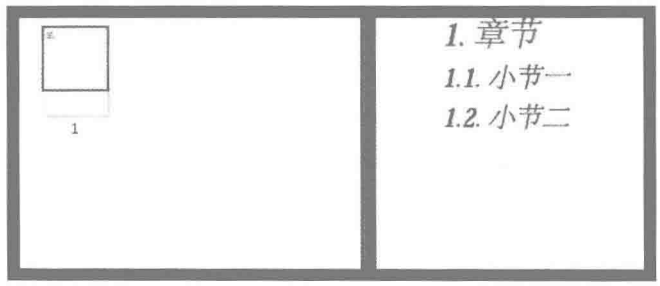


图 16.16　在全屏模式下显示缩略图的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageModeFullScreen，即设置为全屏模式，然后再使用该方法将入口参数设置为 PdfWriter.NonFullScreenPageModeUseThumbs 实现使 PDF 阅读器在全屏模式下显示缩略图的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\全屏模式下显示缩略图.pdf"));  //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeFullScreen);                                                  //设置为全屏模式
writer.setViewerPreferences(PdfWriter.NonFullScreenPageModeUseThumbs);                                      //设置阅读器在全屏模式下显示缩略图
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 NonFullScreenPageModeUseThumbs 类，该类中只有一个 main()主方法，实现使 PDF 阅读器在全屏模式下显示缩略图的功能。主方法的主要代码如下：

```
Document document = new Document();                                                                          //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\全屏模式下显示缩略图.pdf"));                                                  //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeFullScreen);                                                   //设置为全屏模式
writer
        .setViewerPreferences(PdfWriter.NonFullScreenPageModeUseThumbs);                                     //设置阅读器在全屏模式下显示缩略图
document.open();                                                                                             //打开文档
BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);               //定义基础字体
Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,BaseColor.RED);                                    //实例化字体类、设置字体大小和颜色
Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,BaseColor.BLUE);                                   //实例化字体类、设置字体大小和颜色
Paragraph paragraph = new Paragraph("章节",fontChinese1);                                                    //创建段落对象
Chapter chapter = new Chapter(paragraph,1);                                                                  //创建章节对象
```

```
paragraph = new Paragraph("小节一",fontChinese2);                    //创建段落对象
chapter.addSection(paragraph);                                       //添加小节
paragraph = new Paragraph("小节二",fontChinese2);                    //创建段落对象
chapter.addSection(paragraph);                                       //添加小节
document.add(chapter);                                               //向文档添加章节
document.close();                                                    //关闭文档
```

秘笈心法

心法领悟 489：添加指定行为链接注释，以进入文档的第一页。

实现通过指定行为链接注释从当前页链接到第一页的代码如下：

```
Annotation annotation = new Annotation(10f, 10f, 500f, 820f, PdfAction.FIRSTPAGE);   //创建指定行为链接注释，用于链接到文档的第一页
document.add(annotation);                                                            //为文档添加指定行为链接注释
```

实例 490　全屏模式下不显示大纲和缩略图　初级

光盘位置：光盘\MR\490　　趣味指数：★★★★

实例说明

在使用阅读器查看 PDF 文档时，如果不希望在全屏模式下显示大纲和缩略图，可以通过本实例讲解的方法实现。运行程序，可以看到 PDF 阅读器在全屏模式下不显示大纲和缩略图的效果，如图 16.17 所示。

图 16.17　在全屏模式下不显示大纲和缩略图的效果

关键技术

本实例主要是通过 PdfWriter 类的 setViewerPreferences()方法，并将入口参数设置为 PdfWriter.PageModeFullScreen，即设置为全屏模式，然后再使用该方法将入口参数设置为 PdfWriter.NonFullScreenPageModeUseNone 实现使 PDF 阅读器在全屏模式下不显示大纲和缩略图的功能，实现代码如下：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("c:\\全屏模式下不显示大纲和缩略图.pdf"));   //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeFullScreen);                                                        //设置为全屏模式
writer.setViewerPreferences(PdfWriter.NonFullScreenPageModeUseNone);        //设置全屏模式下阅读器不显示大纲和缩略图
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 NonFullScreenPageModeUseNone 类，该类中只有一个 main()主方法，实现使 PDF 阅读器在全屏模式下不显示大纲和缩略图的功能。主方法的主要代码如下：

```
Document document = new Document();                                  //创建文档对象
PdfWriter writer = PdfWriter.getInstance(document,
        new FileOutputStream("c:\\全屏模式下不显示大纲和缩略图.pdf"));   //关联文档对象与输出流
writer.setViewerPreferences(PdfWriter.PageModeFullScreen);           //设置为全屏模式
writer.setViewerPreferences(PdfWriter.NonFullScreenPageModeUseNone); //设置全屏模式下阅读器不显示大纲和缩略图
document.open();                                                     //打开文档
```

```
BaseFont Chinese = BaseFont.createFont("STSong-Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);        //定义基础字体
Font fontChinese1 = new Font(Chinese, 18, Font.BOLDITALIC,BaseColor.RED);          //实例化字体类、设置字体大小和颜色
Font fontChinese2 = new Font(Chinese, 15, Font.BOLDITALIC,BaseColor.BLUE);         //实例化字体类、设置字体大小和颜色
Paragraph paragraph = new Paragraph("章节",fontChinese1);                          //创建段落对象
Chapter chapter = new Chapter(paragraph,1);                                        //创建章节对象
paragraph = new Paragraph("小节一",fontChinese2);                                  //创建段落对象
chapter.addSection(paragraph);                                                     //添加小节
paragraph = new Paragraph("小节二",fontChinese2);                                  //创建段落对象
chapter.addSection(paragraph);                                                     //添加小节
document.add(chapter);                                                             //向文档添加章节
document.close();                                                                  //关闭文档
```

秘笈心法

心法领悟 490：添加指定行为链接注释，以进入文档的最后一页。

实现通过指定行为链接注释从当前页链接到最后一页的代码如下：

```
Annotation annotation = new Annotation(10f, 10f, 500f, 820f, PdfAction.LASTPAGE);  //创建指定行为链接注释，用于链接到文档的最后一页
document.add(annotation);                                                          //为文档添加指定行为链接注释
```

网络技术篇

第17章

网络应用基础

- 网络地址解析
- 网络资源管理

17.1　网络地址解析

实例 491　获取本地主机的 IP 地址

初级

光盘位置：光盘\MR\491

趣味指数：★★★

实例说明

在进行网络编程时，为了满足网络通信的需要，经常需要获取本地主机的 IP 地址。本实例将演示如何通过 Java 程序获得本地主机的 IP 地址。运行程序，单击窗体上的“获取 IP 地址”按钮，将在文本框中显示本地主机的 IP 地址，效果如图 17.1 所示。

图 17.1　获取本地主机的 IP 地址

关键技术

本实例主要是通过 InetAddress 类的 getLocalHost()方法获得本地主机的 InetAddress 对象，然后调用该对象的 getHostAddress()方法获得本地主机的 IP 地址。

（1）通过 InetAddress 类的 getLocalHost()方法可以获得本地主机的 InetAddress 对象，该方法的定义如下：

```
public static InetAddress getLocalHost() throws UnknownHostException
```

参数说明

❶ 返回值：返回本地主机的 InetAddress 对象。

❷ 异常：如果找不到任何主机，则抛出 UnknowHostException 异常。

（2）通过 InetAddress 类的 getHostAddress()方法可以获得本地主机的 IP 地址，该方法的定义如下：

```
public String getHostAddress()
```

参数说明

返回值：以字符串的形式获得 IP 地址。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 GetLocalHostIpFrame。

（3）将 GetLocalHostIpFrame 窗体的内容面板设置为绝对布局，然后在面板上放置两个 JLabel 标签和一个 JTextField 文本框，其中，文本框的名称为 tf_ip；再添加两个 JButton 命令按钮，分别用于获得本地主机的 IP 地址和退出系统。

（4）在“获取 IP 地址”按钮的事件中，实现获得本地主机的 IP 地址的功能，并显示在文本框中，该按钮的代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        try {
            InetAddress inetAddr = InetAddress.getLocalHost();  //创建本地主机的 InetAddress 对象
```

```
                String ip = inetAddr.getHostAddress();          //获得本地主机的 IP 地址
                tf_ip.setText(ip);                              //在文本框中显示 IP 地址
            } catch (UnknownHostException e1) {
                e1.printStackTrace();
            }
        }
    });
```

秘笈心法

心法领悟 491：在网络中使用本机。

在网络中使用本机，可以通过本机的 IP 地址访问，也可以通过 127.0.0.1 或 localhost 访问，但是只有 IP 地址能够唯一标识网络中的一台计算机。

实例 492　获取本地主机的域名和主机名　初级

光盘位置：光盘\MR\492　趣味指数：★★★

实例说明

在进行网络编程时，除了使用本地主机的 IP 地址外，也经常需要使用本地主机的域名和主机名。本实例将演示如何通过 Java 程序获得本地主机的域名和主机名。运行程序，单击窗体上的“获取域名和主机名”按钮，将在相应的文本框中显示本地主机的域名和主机名，效果如图 17.2 所示。

图 17.2　获取本地主机的域名和主机名

关键技术

本实例主要是通过 InetAddress 类的 getLocalHost()方法获得本地主机的 InetAddress 对象，然后调用该对象的 getHostName()方法获得本地主机的名称，以及使用 getCanonicaHostName()方法获得本地主机的域名。

（1）通过 InetAddress 类的 getHostName()方法可以获得本地主机的主机名，该方法的定义如下：

```
public String getHostName()
```

参数说明

返回值：本地主机的主机名。

（2）通过 InetAddress 类的 getCanonicalHostName()方法可以获得本地主机的域名，该方法的定义如下：

```
public String getCanonicalHostName()
```

参数说明

返回值：本地主机的域名。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 GetLocalHostNameFrame。

（3）将 GetLocalHostNameFrame 窗体的内容面板设置为绝对布局，然后在面板上放置 3 个 JLabel 标签和

两个 JTextField 文本框，其中，文本框的名称分别为 tf_canonical 和 tf_host；再添加两个 JButton 命令按钮，分别用于获得本地主机的域名和主机名、退出系统。

（4）在“获取域名和主机名”按钮的事件中，实现获得本地主机的域名和主机名的功能，并显示在相应的文本框中，该按钮的代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        try {
            InetAddress inetAddr = InetAddress.getLocalHost();          //创建本地主机的 InetAddress 对象
            String canonical = inetAddr.getCanonicalHostName();         //获取本地主机的域名
            String host = inetAddr.getHostName();                       //获取本地主机的主机名
            tf_canonical.setText(canonical);                            //在文本框中显示本地主机的域名
            tf_host.setText(host);                                      //在文本框中显示本地主机的主机名
        } catch (UnknownHostException e1) {
            e1.printStackTrace();
        }
    }
});
```

秘笈心法

心法领悟 492：为什么域名和主机名重名？

当获得本地主机的域名和主机名时，如果本地主机没有域名，则显示的域名和主机名重名；如果本地主机有域名，就会显示对应的域名。

实例 493　通过域名获得 IP 地址

光盘位置：光盘\MR\493　　中级　　趣味指数：★★★

实例说明

在进行网络编程时，有时需要通过域名获得 IP 地址。本实例将演示如何在 Java 应用程序中通过域名获得 IP 地址。运行程序，在“输入域名”文本框中输入域名 ZZK-PC，然后单击窗体上的“获取 IP 地址”按钮，将在“IP 地址”文本框中显示对应的 IP 地址，效果如图 17.3 所示。

图 17.3　通过域名获得 IP 地址

关键技术

本实例主要是通过 InetAddress 类的 getByName()方法获得网络中指定域名的 InetAddress 对象，然后调用该对象的 getHostAddress()方法获得具有该域名的主机 IP 地址。

（1）通过 InetAddress 类的 getByName()方法可以获得网络中指定域名的 InetAddress 对象，该方法的定义如下：

```
public static InetAddress getByName(String host) throws UnknownHostException
```

参数说明

❶ host：指定的主机域名或主机名。

❷ 返回值：返回指定域名主机的 InetAddress 对象。

❸ 异常：如果找不到任何主机，则抛出 UnknowHostException 异常。

（2）通过 InetAddress 类的 getHostAddress()方法可以获得指定主机的 IP 地址，用法与本实例相同，不再重复讲解。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 ByDomainGainIpFrame。

（3）将 ByDomainGainIpFrame 窗体的内容面板设置为绝对布局，然后在内容面板的合适位置放置标签、文本框和命令按钮控件，完成窗体界面的设置。

（4）在“获取 IP 地址”按钮的事件中，实现根据输入的域名获得 IP 地址的功能，并显示在相应的文本框中，该按钮的事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        try {
            String domain = tf_domain.getText();                    //获得输入的域名
            InetAddress inetAddr = InetAddress.getByName(domain);   //创建 InetAddress 对象
            String ip = inetAddr.getHostAddress();                  //获得 IP 地址
            tf_ip.setText(ip);                                      //在文本框中显示 IP 地址
        } catch (UnknownHostException e1) {
            e1.printStackTrace();
        }
    }
});
```

秘笈心法

心法领悟 493：IP 地址的功能。

在计算机网络中，是通过 IP 地址来标识不同的计算机的，但是 IP 地址并不容易记忆，所以为了方便用户使用而推出了域名。Java 语言从方便用户和程序员的角度，提供了将域名转换成 IP 地址的方法。可以根据 IP 地址判断域名对应网站的位置。

实例 494　通过 IP 地址获得域名和主机名

光盘位置：光盘\MR\494　　中级　　趣味指数：★★★★

实例说明

在进行网络编程时，可以通过指定主机的 IP 地址获得对应主机的域名和主机名。本实例演示如何在 Java 应用程序中通过 IP 地址获得域名和主机名。运行程序，在“输入 IP 地址”文本框中输入 IP 地址 192.168.1.247，然后单击窗体上的“获取域名和主机名”按钮，将在“域名”和“主机名”文本框中显示对应的域名和主机名，效果如图 17.4 所示。

图 17.4　通过 IP 地址获得域名和主机名

关键技术

本实例主要是将 IP 地址转换为字节数组，然后通过 InetAddress 类的 getByAddress()方法获得网络主机中具有指定 IP 地址的 InetAddress 对象，然后调用该对象的 getCanonicalHostName()方法获得对应的域名，通过 getHostName()方法获得对应的主机名。

（1）将 IP 地址转换为字节数组，可以通过如下代码实现：

```
String ip = tf_ip.getText();                          //IP 地址
String[] ipStr = ip.split("[.]");                     //IP 地址转换为字符串数组
byte[] ipBytes = new byte[4];                         //声明存储转换后 IP 地址的字节数组
for (int i = 0; i < 4; i++) {
    int m = Integer.parseInt(ipStr[i]);               //转换为整数
    byte b = (byte) (m & 0xff);                       //转换为字节
    ipBytes[i] = b;                                   //赋值给字节数组
}
```

说明：上面代码中的 tf_ip 指窗体上用于输入 IP 地址的文本框控件。

（2）通过 InetAddress 类的 getByAddress()方法可以获得网络主机中具有指定 IP 地址的 InetAddress 对象，然后调用该 InetAddress 对象的 getCanonicalHostName()方法获得对应的域名，通过 getHostName()方法获得对应的主机名，实现代码如下：

```
InetAddress inetAddr = InetAddress.getByAddress(ipBytes);   //创建 InetAddress 对象
String canonical = inetAddr.getCanonicalHostName();         //获取域名
String host = inetAddr.getHostName();                       //获取主机名
```

说明：上面代码中的 ipBytes 是指定 IP 地址的字节数组表示形式。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 ByIpGainDomainFrame。

（3）将 ByIpGainDomainFrame 窗体的内容面板设置为绝对布局，然后在内容面板的合适位置放置标签、文本框和命令按钮控件，完成窗体界面的设置。

（4）在“获取域名和主机名”按钮的事件中实现根据输入的 IP 地址获得指定主机的域名和主机名的功能，并显示在相应的文本框中，该按钮的事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        try {
            String ip = tf_ip.getText();                          //IP 地址
            String[] ipStr = ip.split("[.]");                     //IP 地址转换为字符串数组
            byte[] ipBytes = new byte[4];                         //声明存储转换后 IP 地址的字节数组
            for (int i = 0; i < 4; i++) {
                int m = Integer.parseInt(ipStr[i]);               //转换为整数
                byte b = (byte) (m & 0xff);                       //转换为字节
                ipBytes[i] = b;                                   //赋值给字节数组
            }
            InetAddress inetAddr = InetAddress.getByAddress(ipBytes); //创建 InetAddress 对象
            String canonical = inetAddr.getCanonicalHostName();   //获取域名
            String host = inetAddr.getHostName();                 //获取主机名
            tf_canonical.setText(canonical);                      //在文本框中显示域名
            tf_host.setText(host);                                //在文本框中显示主机名
        } catch (UnknownHostException e1) {
            e1.printStackTrace();
        }
    }
});
```

秘笈心法

心法领悟 494：域名的用途。

通常情况下，域名中的不同字母可以表示不同的含义，例如，com 表示商业组织、edu 表示教育组织等，可以根据由 IP 地址获得的域名大致判断网站的用途。

实例 495 获得内网的所有 IP 地址

光盘位置：光盘\MR\495　　中级　　趣味指数：★★★

实例说明

在进行网络编程时，有时需要对局域网内的所有主机进行遍历，为此需要获得内网的所有 IP 地址。本实例演示如何在 Java 应用程序中获得内网的所有 IP 地址。运行程序，单击窗体上的“显示所有 IP”按钮，将在文本域中显示内网中所有主机的 IP 地址，效果如图 17.5 所示。

图 17.5　获得内网的所有 IP 地址

关键技术

本实例首先获得本机 IP 地址所属的网段，然后 ping 网络中的 IP 地址，通过输入流对象读取所 ping 结果，并判断是否为内网的 IP 地址，从而实现了本实例的功能。

（1）获得本机 IP 地址所属的网段，可以通过如下代码实现：

```
InetAddress host = InetAddress.getLocalHost();                    //获得本机的 InetAddress 对象
String hostAddress = host.getHostAddress();                       //获得本机的 IP 地址
int pos = hostAddress.lastIndexOf(".");                           //获得 IP 地址中最后一个点的位置
String wd = hostAddress.substring(0, pos + 1);                    //对本机的 IP 地址进行截取，获得网段
```

（2）ping 网络中的 IP 地址，通过输入流对象读取所 ping 结果，并判断是否为内网的 IP 地址，实现代码如下：

```
//获得所 ping 的 IP 进程，-w 280 是等待每次回复的超时时间，-n 1 是要发送的回显请求数
Process process = Runtime.getRuntime().exec("ping " + ip + " -w 280 -n 1");
InputStream is = process.getInputStream();                        //获得进程的输入流对象
InputStreamReader isr = new InputStreamReader(is);                //创建 InputStreamReader 对象
BufferedReader in = new BufferedReader(isr);                      //创建缓冲字符流对象
String line = in.readLine();                                      //读取信息
while (line != null) {
    if (line != null && !line.equals("")) {
        if (line.substring(0, 2).equals("来自") || (line.length() > 10 && line.substring(0, 10) .equals("Reply from"))) {      //是 ping 通的 IP 地址
            pingMap.put(ip, "true");                              //向集合中添加 IP
```

```
            }
        }
        line = in.readLine();                                    //再读取信息
    }
```

说明：上面代码中的 ip 是一个具体的 IP 地址值，如果是内网中的 IP 地址值，则将其添加到集合中。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 GainAllIpFrame。

（3）在 GainAllIpFrame 窗体的内容面板顶部位置放置一个面板控件，并在该面板控件上添加两个命令按钮，然后在内容面板的中部位置添加一个滚动面板控件，并在滚动面板上放置一个文本域控件，完成窗体界面的设置。

（4）在 GainAllIpFrame 窗体类中定义一个 gainAllIp()方法，用于获得内容中的所有 IP 地址，并追加到文本域中显示，该方法的代码如下：

```
InetAddress host = InetAddress.getLocalHost();                   //获得本机的 InetAddress 对象
String hostAddress = host.getHostAddress();                      //获得本机的 IP 地址
int pos = hostAddress.lastIndexOf(".");                          //获得 IP 地址中最后一个点的位置
String wd = hostAddress.substring(0, pos + 1);                   //对本机的 IP 地址进行截取，获得网段
for (int i = 1; i <= 255; i++) {                                 //对局域网的 IP 地址进行遍历
    String ip = wd + i;                                          //生成 IP 地址
    PingIpThread thread = new PingIpThread(ip);                  //创建线程对象
    thread.start();                                              //启动线程对象
}
Set<String> set = pingMap.keySet();                              //获得集合中键的 Set 视图
Iterator<String> it = set.iterator();                            //获得迭代器对象
while (it.hasNext()) {                                           //迭代器中有元素，则执行循环体
    String key = it.next();                                      //获得下一个键的名称
    String value = pingMap.get(key);                             //获得指定键的值
    if (value.equals("true")) {
        ta_allIp.append(key + "\n");                             //追加显示 IP 地址
    }
}
```

（5）在 GainAllIpFrame 窗体类中定义一个 PingIpThread 线程类，用于 ping 指定的 IP 地址，并判断其是否为有效的内网 IP 地址，如果是，则添加到集合对象中，该类的代码如下：

```
public String ip;                                                //表示 IP 地址的成员变量
public PingIpThread(String ip) {                                 //参数为需要判断的 IP 地址
    this.ip = ip;
}
public void run() {
    try {
        //获得所 ping 的 IP 进程，-w 280 是等待每次回复的超时时间，-n 1 是要发送的回显请求数
        Process process = Runtime.getRuntime().exec(
                "ping " + ip + " -w 280 -n 1");
        InputStream is = process.getInputStream();               //获得进程的输入流对象
        InputStreamReader isr = new InputStreamReader(is);       //创建 InputStreamReader 对象
        BufferedReader in = new BufferedReader(isr);             //创建缓冲字符流对象
        String line = in.readLine();                             //读取信息
        while (line != null) {
            if (line != null && !line.equals("")) {
                if (line.substring(0, 2).equals("来自")
                        || (line.length() > 10 && line.substring(0, 10)
                                .equals("Reply from"))) {        //判断是 ping 通过的 IP 地址
                    pingMap.put(ip, "true");                     //向集合中添加 IP 地址
                }
            }
            line = in.readLine();                                //再读取信息
        }
    } catch (IOException e) {
    }
}
```

秘笈心法

心法领悟 495：避免设置的 IP 地址重复。

在内网中增加计算机时，需要设置其 IP 地址，为了避免 IP 地址重复，可以先获得内网的所有 IP 地址，然后再进行设置。

17.2 网络资源管理

实例 496	获取网络资源的大小	中级
	光盘位置：光盘\MR\496	趣味指数：★★★

实例说明

本实例演示了如何获得网络资源的大小。运行程序，在“输入网址”文本框中输入要访问的网站网址，单击“获得大小”按钮，将获得该网络资源的大小，效果如图 17.6 所示。

图 17.6 获取网络资源的大小

关键技术

本实例主要是通过 URLConnection 类的 getContentLength()方法获得网络资源的大小，并将其显示在文本框中。getContentLength()方法的代码如下：

```
URLConnection urlConn = url.openConnection();                    //获得网络连接对象
long size = urlConn.getContentLength();                          //以字节为单位获得网络资源的大小
```

说明：上面代码中的 url 是一个有效的 URL 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 InternetSizeFrame 窗体类。

（3）在 InternetSizeFrame 窗体类中将内部面板设置为绝对布局，然后在窗体上添加标签、文本框和命令按钮控件，完成窗体界面的设计。

（4）在 InternetSizeFrame 窗体类中定义获得网络资源大小的 netSourceSize()方法，该方法的代码如下：

```
public long netSourceSize(String sUrl) throws Exception{
    URL url = new URL(sUrl);                                     //创建 URL 对象
    URLConnection urlConn = url.openConnection();                //获得网络链接对象
    urlConn.connect();                                           //打开到 url 引用资源的通信链接
    return urlConn.getContentLength();                           //以字节为单位返回资源的大小
}
```

（5）在窗体类 InternetSizeFrame 的“获得大小”按钮事件中，添加获得网络资源大小，并将网络资源大小显示在文本框中的事件代码：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        String address = tf_address.getText().trim();                    //获得输入的网址
        try{
            long len = netSourceSize(address);                           //调用方法获取网络资源的大小
            tf_size.setText(String.valueOf(len)+" 字节");                //在文本框中显示网络资源的大小
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
});
```

秘笈心法

心法领悟 496：获取网络资源大小的意义。

网络已经走进千家万户，用户经常需要从网络上下载大量的资源，这时可以先获得网络资源的大小，然后合理安排下载时间和顺序，从而可以有效地利用网络资源，尤其对于上网时间有限的用户来说，可以根据网络资源的大小确定在较短的时间内是否能完成下载。

实例 497　解析网页中的内容

光盘位置：光盘\MR\497　　中级　　趣味指数：★★★

实例说明

本实例演示了如何在 Java 应用程序中解析网页的内容。运行程序，在“输入网址”文本框中输入网址，单击“解析网页”按钮，将在文本域中显示解析的网页内容，效果如图 17.7 所示。

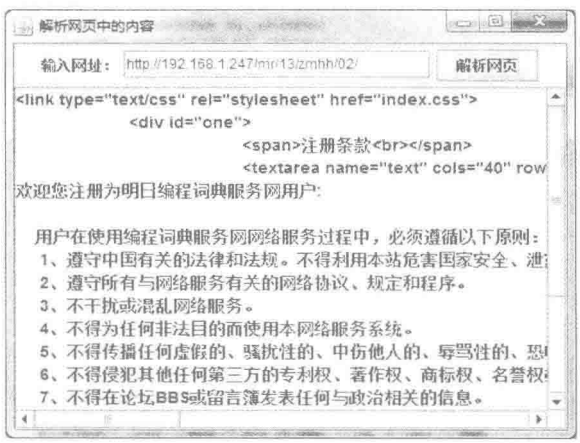


图 17.7　解析网页中的内容

关键技术

本实例主要是通过 URLConnection 类的 getInputStream()方法获得网页资源的输入流对象，然后从该输入流中读取信息，完成解析网页内容的操作。

使用 URLConnection 类的 getInputStream()方法获得网页资源的输入流对象，可以通过如下代码实现：

```
URLConnection conn = url.openConnection();                           //获得网页资源的连接对象
InputStream is = conn.getInputStream();                              //获得网页资源的输入流对象
```

说明：上面代码中的 url 是一个有效的网页资源的 URL 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 InternetContentFrame 窗体类。

（3）在 InternetContentFrame 窗体类中创建用于解析网页内容的 getURLConnection()方法，该方法的代码如下：

```
public Collection<String> getURLCollection(String urlString){
    URL url = null;                                          //声明 URL
    URLConnection conn = null;                               //声明 URLConnection
    Collection<String> urlCollection = new ArrayList<String>();  //创建集合对象
    try{
        url = new URL(urlString);                            //创建 URL 对象
        conn = url.openConnection();                         //获得链接对象
        conn.connect();                                      //打开到 url 引用资源的通信链接
        InputStream is = conn.getInputStream();              //获取流对象
        InputStreamReader in = new InputStreamReader(is,"UTF-8");  //转换为字符流
        BufferedReader br = new BufferedReader(in);          //创建缓冲流对象
        String nextLine = br.readLine();                     //读取信息，解析网页
        while (nextLine !=null){
            urlCollection.add(nextLine);                     //解析网页的全部内容，添加到集合中
            nextLine = br.readLine();                        //读取信息，解析网页
        }
    }catch(Exception ex){
        ex.printStackTrace();
    }
    return urlCollection;
}
```

（4）在 InternetContentFrame 窗体类中为“解析网页”按钮添加事件代码，实现调用 getURLConnection()方法解析网页内容并显示在文本域中的功能，该按钮的事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        String address = tf_address.getText().trim();            //获得输入的网址
        Collection urlCollection = getURLCollection(address);    //调用方法，获得网页内容的集合对象
        Iterator it = urlCollection.iterator();                  //获得集合的迭代器对象
        while(it.hasNext()){
            ta_content.append((String)it.next()+"\n");           //在文本域中显示解析的内容
        }
    }
});
```

秘笈心法

心法领悟 497：获得网页的源码。

当用户浏览网页时，若觉得某个网页的内容较好，但是该网页却对用户进行了限制，使得用户无法保存网页，这时就可以通过解析网页的内容，获得网页的源码，用户只需要将源码保存为网页，即可通过浏览器查看网页内容。

实例 498 网络资源的单线程下载

光盘位置：光盘\MR\498　　高级　　趣味指数：★★★★

实例说明

本实例实现了网络资源的单线程下载，即在一个线程中完成网络资源的下载。运行程序，输入下载资源的网

址，单击窗体上的“单击开始下载”按钮，将下载网络资源，并在下载完毕后进行提示，效果如图 17.8 和图 17.9 所示。

图 17.8　网络资源的单线程下载界面

图 17.9　提示下载完毕的消息框

关键技术

本实例主要是在主线程中利用 URLConnection 对象的 getInputStream()方法获得网络资源的输入流对象，并将读取的信息通过输出流保存到用户主机上，从而实现了网络资源的单线程下载。

（1）通过 URLConnection 对象的 getInputStream()方法获得网络资源的输入流对象，用于读取网络资源的信息。

（2）使用 FileOutputStream 类创建输出流对象，然后使用该类的 write()方法将从输入流获得的网络资源保存到磁盘上，实现网络资源的单线程下载，代码如下：

```
FileOutputStream out = new FileOutputStream("c:/"+fileName);          //创建输出流对象
byte[] bytes = new byte[1024];                                        //声明存放下载内容的字节数组
int len = in.read(bytes);                                             //从输入流中读取内容
while (len != -1){
    out.write(bytes,0,len);                                           //将读取的内容写到输出流
    len = in.read(bytes);                                             //继续从输入流中读取内容
}
```

说明：上面代码中的 fileName 是需要下载的网络资源的名称。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 SingleThreadDownloadFrame 窗体类。

（3）在 SingleThreadDownloadFrame 窗体类的内容面板上添加相应控件，完成窗体界面的设计。

（4）在 SingleThreadDownloadFrame 窗体类中定义 download()方法，用于根据参数 urlAddr 指定的地址完成网络资源的单线程下载，该方法的代码如下：

```
public void download(String urlAddr){                                 //从指定网址下载文件
    try {
        URL url = new URL(urlAddr);                                   //创建 URL 对象
        URLConnection urlConn = url.openConnection();                 //获得链接对象
        urlConn.connect();                                            //打开到 url 引用资源的通信链接
        InputStream in = urlConn.getInputStream() ;                   //获得输入流对象
        String filePath = url.getFile();                              //获得完整路径
        int pos = filePath.lastIndexOf("/");                          //获得路径中最后一个斜杠的位置
        String fileName = filePath.substring(pos+1);                  //截取文件名
        FileOutputStream out = new FileOutputStream("c:/"+fileName);  //创建输出流对象
        byte[] bytes = new byte[1024];                                //声明存放下载内容的字节数组
        int len = in.read(bytes);                                     //从输入流中读取内容
        while (len != -1){
            out.write(bytes,0,len);                                   //将读取的内容写到输出流
            len = in.read(bytes);                                     //继续从输入流中读取内容
        }
```

```
            out.close();                                    //关闭输出流
            in.close();                                     //关闭输入流
            JOptionPane.showMessageDialog(null, "下载完毕");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

（5）为 SingleThreadDownloadFrame 窗体中“单击开始下载”按钮添加事件代码，用于调用 download()方法，该按钮的事件代码如下：

```
button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        String address = tf_address.getText().trim();       //获得网址
        download(address);                                  //下载文件
    }
});
```

秘笈心法

心法领悟 498：及时关闭流的必要性。

在下载网络资源时，需要及时关闭 IO 流，因为每个 IO 流都会占用较多的系统资源，并且 IO 流并不能被垃圾回收机制回收，当下载网络资源的用户较多时，就会造成不必要的资源浪费，甚至会使系统崩溃。

实例 499 网络资源的多线程下载

高级

光盘位置：光盘\MR\499

趣味指数：★★★★

实例说明

本实例实现了网络资源的多线程下载，即主线程启动后，再通过单独的线程完成网络资源的下载。运行程序，输入下载资源的网址，单击窗体上的“下载”按钮，将完成网络资源下载，并在下载完毕后进行提示，效果如图 17.10 和图 17.11 所示。

图 17.10 网络资源的多线程下载界面

图 17.11 提示完成下载消息框

关键技术

本实例主要是通过创建线程类，然后在下载时指定线程数，并通过 RandomAccessFile 类的 seek()方法定位下一个写入点，同时通过 write()方法写入文件，从而完成网络资源的多线程下载。

（1）通过实现 Runnable 接口创建 DownMultiThread 类，在该类的成员声明区定义网络资源地址、需要写入的目标文件对象、写入的开始位置和结束位置 4 个成员变量，并通过构造方法为其赋值，代码如下：

```
public class DownMultiThread implements Runnable{
    private String sUrl = "";                               //网络资源地址
    private File desFile;                                   //需要写入的目标文件对象
    private long startPos;                                  //写入的开始位置
    private long endPos;                                    //写入的结束位置
    /**
     * @param sUrl 网络资源地址
     * @param file 需要写入的目标文件对象
```

```
     * @param startPos 写入的开始位置
     * @param endPos 写入的结束位置
     */
    public DownMultiThread(String sUrl,File desFile,long startPos,long endPos) {
        this.sUrl = sUrl;
        this.desFile = desFile;
        this.startPos = startPos;
        this.endPos = endPos;
    }
    //省略了其他代码
}
```

（2）使用 RandomAccessFile 类的 seek()方法定位下一个写入点，并通过 write()方法将网络资源写入文件，实现代码如下：

```
RandomAccessFile out = new RandomAccessFile(desFile, "rw");          //创建可读写的流对象
out.seek(startPos);                                                  //指定读写的开始标记
out.write(buff,0,len);                                               //写入磁盘文件
```

说明：上面代码中的 buff 是存储网络资源的字节数组，len 是数组中所存储内容的实际长度。

设计过程

（1）新建一个项目。

（2）在项目中创建一个实现 Runnable 接口的线程类 DownMultiThread，用于实现网络资源的下载，该线程类中 run()方法的关键代码如下：

```
RandomAccessFile out = new RandomAccessFile(desFile, "rw");          //创建可读写的流对象
out.seek(startPos);                                                  //指定读写的开始标记
InputStream in = conn.getInputStream();                              //获得网络资源的输入流对象
BufferedInputStream bin = new BufferedInputStream(in);               //创建输入缓冲流对象
byte[] buff = new byte[2048];                                        //创建字节数组
int len = -1;                                                        //声明存放读取字节数的变量
len=bin.read(buff);                                                  //读取到内容并添加到字节数组
while (len!=-1){
    out.write(buff,0,len);                                           //写入磁盘文件
    len=bin.read(buff);                                              //读取到内容并添加到字节数组
}
```

（3）在项目中创建一个继承 JFrame 类的 MultiThreadDownFrame 窗体类，在该类中定义一个 download()方法，用于实现通过线程类 DownMultiThread 下载网络资源，该方法的代码如下：

```
public void download(String url, String dest, int threadNum)
        throws Exception {
    URL downURL = new URL(url);                                      //创建网络资源的 URL
    HttpURLConnection conn = (HttpURLConnection) downURL.openConnection();  //打开网络链接
    long fileLength = -1;                                            //用于存储文件长度的变量
    int stateFlagCode = conn.getResponseCode();                      //获得链接状态标记代码
    if (stateFlagCode == 200) {                                      //网络链接正常
        fileLength = conn.getContentLength();                        //获得文件的长度
        conn.disconnect();                                           //取消网络链接
    }
    if (fileLength > 0) {
        long byteCounts = fileLength / threadNum + 1;                //计算每个线程的字节数
        File file = new File(dest);                                  //创建目标文件的 File 对象
        int i = 0;
        while (i < threadNum) {
            long startPosition = byteCounts * i;                     //定义开始位置
            long endPosition = byteCounts * (i + 1);                 //定义结束位置
            if (i == threadNum - 1) {
                DownMultiThread fileThread = new DownMultiThread(url, file,
                        startPosition, 0);                           //创建 DownMultiThread 线程的实例
                new Thread(fileThread).start();                      //启动线程对象
            } else {
                DownMultiThread fileThread = new DownMultiThread(url, file,
                        startPosition, endPosition);                 //创建 DownMultiThread 线程的实例
                new Thread(fileThread).start();                      //启动线程对象
```

```
                }
                i++;
            }
            JOptionPane.showMessageDialog(null, "完成网络资源的下载。");
        }
    }
```

（4）在 MultiThreadDownFrame 窗体类的“下载”按钮事件中，添加调用 download()方法的代码，完成网络资源的多线程下载，该按钮的事件代码如下：

```
String address = tf_address.getText();                    //获得网络资源地址
download(address, "c:\\01.flv", 2);                       //调用 download()方法，将下载的网络资源保存到磁盘
```

秘笈心法

心法领悟 499：提高多线程下载的系统性能。

使用多线程进行网络资源下载时，由于创建线程是非常耗费资源的，如果创建的线程较多，将极大地影响系统性能，这时可以使用线程池来提高系统性能，当需要创建线程时，可以从线程池中取出空闲线程，从而提高多线程下载的系统性能。

实例 500　下载网络资源的断点续传

光盘位置：光盘\MR\500　　高级　　趣味指数：★★★★★☆

实例说明

本实例实现了网络资源的断点续传功能。运行程序，输入下载资源的网址，然后按 Enter 键，将显示网络资源的大小、上次读取到的字节位置以及未读取的字节数；输入下载的起始位置和结束位置，单击窗体上的“开始下载”按钮，开始下载网络资源，如果没有下载完成，可以接着上次的下载位置继续下载，直到全部资源下载完成后弹出消息框并退出系统，效果如图 17.12 和图 17.13 所示。

图 17.12　网络资源的多线程下载界面

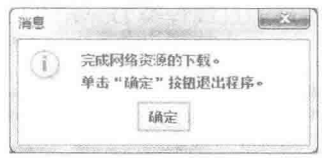


图 17.13　显示完成下载消息框

关键技术

本实例主要是通过设置请求参数 RANGE 实现的，通过该参数，可以指定下载网络资源的字节区间，从而实现每次下载部分网络资源的功能。

例如，将该参数设置为 RANGE: bytes = 0-1024，就表示将网络资源中 0~1024 之间的内容下载到客户机；将该参数设置为 RANGE: bytes =1024-，就表示将网络资源中从 1024 到结束位置的内容全部下载到客户机。要设置 RANGE 属性可以通过如下代码实现：

```
connection.setRequestProperty("User-Agent", "NetFox");          //设置请求属性
String rangeProperty = "bytes=" + startPosition + "-";          //定义请求范围属性
if (endPosition > 0) {
    rangeProperty += endPosition;                               //调整请求范围属性
}
connection.setRequestProperty("RANGE", rangeProperty);          //设置请求范围属性
```

说明：上面代码中的 connection 是连接到网络资源的 HttpURLConnection 对象，startPosition 是下载的起始位置，endPosition 是下载的结束位置。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 BreakPointSuperveneFrame 窗体类。

（3）在 BreakPointSuperveneFrame 窗体类的内容面板上添加相应控件，完成窗体界面的设计。

（4）在 BreakPointSuperveneFrame 窗体类中定义 download()方法，用于实现网络资源的断点续传，该方法的代码如下：

```
public void download(long startPosition, long endPosition) {
    try {
        URL url = new URL(urlAddress);                                  //获得网络资源的 URL
        HttpURLConnection connection = (HttpURLConnection) url
                .openConnection();                                      //获得连接对象
        connection.setRequestProperty("User-Agent", "NetFox");          //设置请求属性
        String rangeProperty = "bytes=" + startPosition + "-";          //定义请求范围属性
        if (endPosition > 0) {
            rangeProperty += endPosition;                               //调整请求范围属性
        }
        connection.setRequestProperty("RANGE", rangeProperty);          //设置请求范围属性
        connection.connect();                                           //连接网络资源
        InputStream in = connection.getInputStream();                   //获得输入流对象
        String file = url.getFile();                                    //获得文件对象
        String name = file.substring(file.lastIndexOf('/') + 1);        //获得文件名
        FileOutputStream out = new FileOutputStream("c:/" + name, true);//创建输出流对象，保存下载的资源
        byte[] buff = new byte[2048];                                   //创建字节数组
        int len = 0;                                                    //定义存储读取内容长度的变量
        len = in.read(buff);                                            //读取内容
        while (len != -1) {
            out.write(buff, 0, len);                                    //写入磁盘
            len = in.read(buff);                                        //读取内容
        }
        out.close();                                                    //关闭流
        in.close();                                                     //关闭流
        connection.disconnect();                                        //断开连接
        if (readToPos > 0 && readToPos == totalLength) {
            JOptionPane.showMessageDialog(null, "完成网络资源的下载。\n 单击“确定”按钮退出程序。");
            System.exit(0);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

（5）在 BreakPointSuperveneFrame 窗体上，为“网络资源的地址”文本框添加动作事件，用于获得文件大小和初始化窗体上的控件，该事件的代码如下：

```
tf_address.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        try {
            urlAddress = tf_address.getText().trim();
            URL url = new URL(urlAddress);                              //获得网络资源的 URL
            HttpURLConnection connection = (HttpURLConnection) url
                    .openConnection();                                  //获得连接对象
            connection.connect();                                       //连接网络资源
            totalLength = connection.getContentLength();                //获得网络资源的长度
            connection.disconnect();                                    //断开连接
            tf_totalLength.setText(String.valueOf(totalLength));        //显示总长度
            tf_readToPos.setText("0");                                  //显示上次读取到的位置
            residuaryLength = totalLength;                              //未读内容为文件总长度
            tf_residuaryLength.setText(String.valueOf(residuaryLength));//显示未读内容
```

```
            } catch (MalformedURLException e1) {
                e1.printStackTrace();
            } catch (IOException e2) {
                e2.printStackTrace();
            }

        }
    });
```

（6）在 BreakPointSuperveneFrame 窗体上为“开始下载”按钮添加动作事件，用于根据输入的起始位置和结束位置完成网络资源的断点续传，该事件的代码如下：

```
    button.addActionListener(new ActionListener() {
        public void actionPerformed(final ActionEvent e) {
            if (totalLength == 0) {
                JOptionPane.showMessageDialog(null,
                        "没有网络资源。\n\n 请输入正确的网址，然后回车。");
                return;
            }
            long startPos = 0;                                          //起始位置
            long endPos = 0;                                            //结束位置
            try {
                startPos = Long.parseLong(tf_startPos.getText().trim());    //起始位置
                endPos = Long.parseLong(tf_endPos.getText().trim());        //结束位置
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(null, "输入的起始位置或结束位置不正确。");
                return;
            }
            readToPos = endPos;                                         //记录读取到的位置
            residuaryLength = totalLength - readToPos;                  //记录未读内容的大小
            tf_readToPos.setText(String.valueOf(readToPos));            //显示读取到的位置
            tf_residuaryLength.setText(String.valueOf(residuaryLength));//显示未读字节数
            tf_startPos.setText(String.valueOf(readToPos));             //设置下一个读取点的开始位置
            tf_endPos.setText(String.valueOf(totalLength));             //设置下一个读取点的结束位置
            tf_endPos.requestFocus();                                   //使结束位置文本框获得焦点
            tf_endPos.selectAll();                                      //选择结束位置文本框中的全部内容，方便输入结束位置值
            download(startPos, endPos);                                 //调用方法进行下载
        }
    });
```

秘笈心法

心法领悟 500：改进下载网络资源的断点续传功能。

为了使本实例更加实用，可以对该实例进行修改，使其能在程序关闭后记录上次的下载位置，当再次打开程序时，可以从上次下载的位置继续下载，直到全部下载结束为止，方法是在程序关闭时，将下载状态保存到文件中，然后在程序打开时通过该文件进行判断，这样即可改进本实例。

第18章

TCP 套接字

- Socket 基础
- TCP 网络通信
- TCP 实用程序

18.1 Socket 基础

实例 501 建立服务器套接字

光盘位置：光盘\MR\501

初级

趣味指数：★★★

实例说明

在进行网络编程时，需要先运行服务器程序，然后等待客户程序连接，因此需要在服务器程序中建立服务器套接字，并等待客户程序的连接。本实例演示如何创建服务器套接字对象。运行程序，效果如图 18.1 所示。

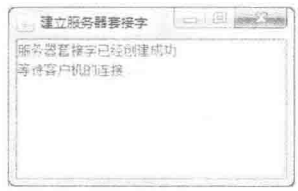


图 18.1 建立服务器套接字

关键技术

本实例主要是通过 ServerSocket 类创建绑定到指定端口的服务器套接字对象，然后调用 ServerSocket 类的 accept()方法监听客户端的连接。

（1）通过 ServerSocket 类的构造方法，可以创建绑定到指定端口的服务器套接字对象，其定义如下：

```
public ServerSocket(int port) throws IOException
```

参数说明

❶ port：服务器套接字绑定的端口号。

❷ 使用该构造方法需要处理 IO 异常。

（2）通过 ServerSocket 类的 accept()方法可以监听客户端的连接，并返回一个套接字对象，该方法的定义如下：

```
public Socket accept() throws IOException
```

参数说明

❶ 返回值：与客户端通信的套接字对象。

❷ 使用该方法需要处理 IO 异常。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 ServerSocketFrame。

（3）在 ServerSocketFrame 窗体的内容面板上添加一个 JscrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，命名为 ta_info，用于显示服务器套接字的创建信息以及与客户端的连接信息。

（4）实现创建服务器套接字和等待客户端连接的代码定义在 getServer()方法中，该方法的代码如下：

```
public void getServer() {
    try {
        server = new ServerSocket(1978);                    //实例化 Socket 对象
        ta_info.append("服务器套接字已经创建成功\n");  //输出信息
        while (true) {                                      //如果套接字是连接状态
            ta_info.append("等待客户机的连接.....\n");  //输出信息
            socket = server.accept();                       //等客户端连接来实例化 Socket 对象
```

```
                ta_info.append("连接成功.....\n");          //输出信息
            }
        } catch (Exception e) {
            e.printStackTrace();                          //输出异常信息
        }
    }
```

秘笈心法

心法领悟 501：建立服务器套接字时可以使用的端口号。

在进行套接字编程时，应该使用大于 1024 的端口号，这样可以避免端口冲突，因为 0~1024 之间的端口已经被一些特殊的协议和服务所占用，如 80 端口被 HTTP 协议所占用、21 端口被 FTP 服务所占用等。

实例 502　建立客户端套接字

光盘位置：光盘\MR\502　　中级　　趣味指数：★★★★

实例说明

在进行网络编程时，为了与服务器端进行通信，需要创建客户端套接字。本实例演示如何创建客户端套接字对象。运行程序，效果如图 18.2 所示。

图 18.2　建立客户端套接字

注意：本实例单独运行时会发生异常，因此需要先运行实例 501，然后再运行本实例，这样程序就不会发生异常了。

关键技术

本实例主要是通过 Socket 类创建到指定服务器和端口的套接字对象，从而实现与服务器的连接。

通过 Socket 类的构造方法可以创建到指定服务器和端口的套接字对象，其定义如下：

```
public Socket(String host, int port) throws UnknownHostException, IOException
```

参数说明

❶ host：要连接服务器的主机名或 IP 地址。

❷ port：连接到服务器的端口号。

❸ 使用该构造方法需要处理 IO 异常和未知主机异常。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 ClientSocketFrame。

（3）在 ClientSocketFrame 窗体的内容面板上添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示客户端与服务器的连接信息。

（4）实现创建客户端套接字并与服务器连接的代码定义在 connect()方法中，该方法的代码如下：

```
private void connect() {                                  //连接套接字方法
    ta_info.append("尝试连接.....\n");                     //在文本域中显示提示信息
```

```
        try {                                                    //捕捉异常
            socket = new Socket("192.168.1.122", 1978);          //实例化 Socket 对象
            ta_info.append("完成连接。\n");                       //在文本域中显示提示信息
        } catch (Exception e) {
            e.printStackTrace();                                 //输出异常信息
        }
    }
```

秘笈心法

心法领悟 502：无法连接服务器。

在使用 Socket 类创建客户端套接字时，所指定的 IP 地址和端口号一定要与服务器相同，否则将无法连接到服务器。

实例 503　设置等待连接的超时时间

中级

光盘位置：光盘\MR\503

趣味指数：★★★

实例说明

在进行网络编程时，进行网络连接是比较消耗资源的，因此，可以对连接的等待时间进行设置，如果在规定的时间没有进行连接，则进行其他处理。运行程序，等待 10 秒钟后，将弹出消息框提示连接超时，效果如图 18.3 所示。

图 18.3　设置等待连接的超时时间

关键技术

本实例主要是通过 ServerSocket 类的实例调用 setSoTimeout()方法设置连接的等待超时时间，从而实现设置等待连接的超时时间的，ServerSocket 类的 setSoTimeout()方法的定义如下：

```
public void setSoTimeout(int timeout) throws SocketException
```

参数说明

❶ timeout：以毫秒为单位指定等待连接的超时时间。

❷ 使用该方法需要处理 SocketException 异常。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 ConnectionTimeoutSetFrame。

（3）在 ConnectionTimeoutSetFrame 窗体的内容面板上添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示服务器等待客户端连接的相关信息，并显示等待连接超时的信息。

（4）实现创建服务器套接字和设置等待连接超时时间的代码定义在 getServer()方法中，该方法的代码如下：

```
public void getserver() {
    try {
        server = new ServerSocket(1978);                    //实例化 Socket 对象
        server.setSoTimeout(10000);                         //设置连接超时时间为 10 秒
        ta_info.append("服务器套接字已经创建成功\n");          //输出信息
        while (true) {                                      //如果套接字是连接状态
            ta_info.append("等待客户机的连接.....\n");         //输出信息
            server.accept();                                //等待客户机连接
        }
    } catch (SocketTimeoutException e) {
        ta_info.append("连接超时.....");
        JOptionPane.showMessageDialog(null, "连接超时.....");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

秘笈心法

心法领悟 503：禁用和启用数据延迟。

可以使用 Socket 类的 setTcpNoDelay()方法，将参数设置为 true，禁用网络传输中的数据延迟；设置为 false，启用数据延迟。对于一般的网络通信，是允许数据延迟的，而对于一些特殊的网络应用，如果允许数据延迟，将会极大地影响数据传输的速度，例如需要实时监控的网络程序以及游戏，就必须禁用数据延迟。

实例 504　获取 Socket 信息

光盘位置：光盘\MR\504　　中级　　趣味指数：★★★★

实例说明

本实例演示在网络通信程序中如何获取远程服务器和客户机的 IP 地址与端口号。运行程序，效果如图 18.4 和图 18.5 所示。其中，图 18.4 是服务器端程序，用于等待客户端的连接；图 18.5 是客户端程序，用于获得远程服务器和客户机的 IP 地址与端口号。

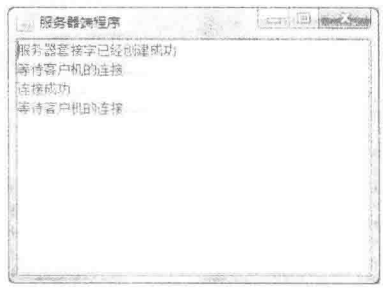


图 18.4　服务器端程序

图 18.5　获取 Socket 信息

关键技术

本实例主要是通过 Socket 类的 getInetAddress()方法获得远程服务器的地址，使用 getLocalAddress()方法获得客户端的地址实现的。

（1）通过 Socket 类的 getInetAddress()方法可以获得远程服务器的地址，进而可以获得远程服务器的 IP 地址和端口号，该方法的定义如下：

```
public InetAddress getInetAddress()
```

参数说明

返回值：此套接字连接到的远程 IP 地址；如果套接字是未连接的，则返回 null。

（2）通过 Socket 类的 getLocalAddress()方法可以获得客户端的地址，进而可以获得客户端的 IP 地址和端口号，该方法的定义如下：

```
public InetAddress getLocalAddress()
```

参数说明

返回值：将套接字绑定到的本地地址；如果尚未绑定套接字，则返回 InetAddress.anyLocalAddress()。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ServerSocketFrame 和一个客户端窗体类 ClientSocketFrame。

（3）在服务器窗体类 ServerSocketFrame 的内容面板上添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示客户端与服务器的连接信息。

（4）在客户端窗体类 ClientSocketFrame 的内容面板上添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示远程服务器和客户机的 IP 地址与端口号。

（5）客户端窗体类 ClientSocketFrame 中，用于显示远程服务器和客户机的 IP 地址与端口号的代码定义在 connect()方法中，该方法的代码如下：

```
private void connect() {                                    //连接套接字方法
    ta.append("尝试连接.....\n");                            //文本域中信息
    try {                                                   //捕捉异常
        socket = new Socket("192.168.1.122", 1978);         //实例化 Socket 对象
        ta.append("完成连接。\n");                           //文本域中提示信息
        InetAddress netAddress = socket.getInetAddress();   //获得远程服务器的地址
        String netIp = netAddress.getHostAddress();         //获得远程服务器的 IP 地址
        int netPort = socket.getPort();                     //获得远程服务器的端口号
        InetAddress localAddress = socket.getLocalAddress();//获得客户端的地址
        String localIp = localAddress.getHostAddress();     //获得客户端的 IP 地址
        int localPort = socket.getLocalPort();              //获得客户端的端口号
        ta.append("远程服务器的 IP 地址：" + netIp + "\n");
        ta.append("远程服务器的端口号：" + netPort + "\n");
        ta.append("客户机本地的 IP 地址：" + localIp + "\n");
        ta.append("客户机本地的端口号：" + localPort + "\n");
    } catch (Exception e) {
        e.printStackTrace();                                //输出异常信息
    }
}
```

秘笈心法

心法领悟 504：另一种获得远程和本地端口的方式。

如果已经建立了套接字对象，并且只需要获得远程和本地计算机的端口号，可以使用 Socket 类的 getPort()方法获得远程计算机的端口号，使用 Socket 类的 getLocalPort()方法获得本地计算机的端口号。

实例 505 接收和发送 Socket 信息

高级

光盘位置：光盘\MR\505

趣味指数：★★★

实例说明

本实例演示在网络通信程序中如何实现信息的发送与接收。运行程序，服务器端程序效果如图 18.6 所示，客户端程序效果如图 18.7 所示。在图 18.7 所示客户端程序的文本框中输入信息，然后按 Enter 键确认，即可将

所输入的信息发送到服务器端，并在图 18.6 所示服务器端显示接收到的信息。

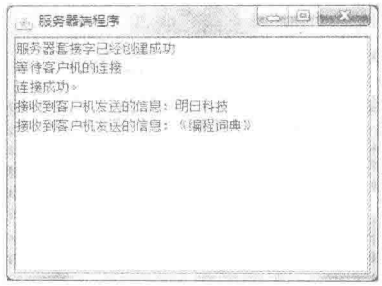


图 18.6　服务器端程序

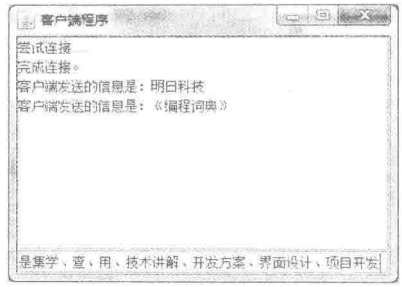


图 18.7　客户端程序

关键技术

本实例主要是通过 Socket 类的 getInputStream()方法获得输入流对象，使用 getOutputStream()方法获得输出流对象实现的。

（1）通过 Socket 类的 getInputStream()方法可以获得输入流对象，该输入流对象用于接收对方发送的信息，该方法的定义如下：

```
public InputStream getInputStream() throws IOException
```

参数说明

❶ 返回值：从此套接字读取字节的输入流，用于接收对方发送的信息。

❷ 使用该方法需要处理 IO 异常。

（2）通过 Socket 类的 getOutputStream()方法可以获得输出流对象，该输出流对象用于向对方发送信息，该方法的定义如下：

```
public OutputStream getOutputStream() throws IOException
```

参数说明

❶ 返回值：将字节写入此套接字的输出流，用于向对方发送信息。

❷ 使用该方法需要处理 IO 异常。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ServerSocketFrame 和一个客户端窗体类 ClientSocketFrame。

（3）在服务器窗体类 ServerSocketFrame 的内容面板上添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示客户端与服务器的连接信息以及接收到的客户端发送的信息。

（4）在客户端窗体类 ClientSocketFrame 的内容面板中部位置添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示连接信息以及客户端本身发送的信息；再向客户端窗体的内容面板底部位置添加一个 JTextField 文本框，命名为 tf_send，用于向服务器端发送信息。

（5）在服务器窗体类 ServerSocketFrame 的成员声明区声明输入流、服务器套接字等成员变量，以便在其他方法中使用，代码如下：

```
private BufferedReader reader;                    //声明 BufferedReader 对象
private ServerSocket server;                      //声明 ServerSocket 对象
private Socket socket;                            //声明 Socket 对象
```

（6）在服务器窗体类 ServerSocketFrame 中定义 getServer()方法，用于创建服务器端套接字、监听客户端程

序，以及创建输入流对象用于接收客户端发送的信息，该方法的代码如下：

```
public void getServer() {
    try {
        server = new ServerSocket(1978);                    //实例化 ServerSocket 对象
        ta_info.append("服务器套接字已经创建成功\n");          //输出信息
        while (true) {                                      //如果套接字是连接状态
            ta_info.append("等待客户机的连接.....\n");        //输出信息
            socket = server.accept();                       //实例化 Socket 对象
            ta_info.append("连接成功。\n");                  //输出信息
            reader = new BufferedReader(new InputStreamReader(socket
                    .getInputStream()));                    //实例化 BufferedReader 对象
            getClientInfo();                                //调用 getClientInfo()方法，该方法也是一个自定义的成员方法
        }
    } catch (Exception e) {
        e.printStackTrace();                                //输出异常信息
    }
}
```

（7）在服务器窗体类 ServerSocketFrame 中再定义一个 getClientInfo()方法，用于接收客户端发送的信息以及关闭输入流对象和套接字对象，该方法的代码如下：

```
private void getClientInfo() {
    try {
        while (true) {                                                          //如果套接字是连接状态
            ta_info.append("接收到客户机发送的信息：" + reader.readLine() + "\n");  //获得客户端信息
        }
    } catch (Exception e) {
        ta_info.append("客户端已退出。\n");                                      //输出异常信息
    } finally {
        try {
            if (reader != null) {
                reader.close();                                                 //关闭流
            }
            if (socket != null) {
                socket.close();                                                 //关闭套接字
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

（8）在客户端窗体类 ClientSocketFrame 的成员声明区声明输出流和套接字对象，以便在其他方法中使用，代码如下：

```
private PrintWriter writer;                                 //声明 PrintWriter 类对象
private Socket socket;                                      //声明 Socket 对象
```

（9）在客户端窗体类 ClientSocketFrame 中定义 connect()方法，用于创建套接字对象和输出流对象以及在文本域中显示连接信息，该方法的代码如下：

```
private void connect() {                                    //连接套接字方法
    ta_info.append("尝试连接.....\n");                       //文本域中信息
    try {                                                   //捕捉异常
        socket = new Socket("192.168.1.122", 1978);         //实例化 Socket 对象
        writer = new PrintWriter(socket.getOutputStream(), true);  //创建输出流对象
        ta_info.append("完成连接。\n");                      //文本域中提示信息
    } catch (Exception e) {
        e.printStackTrace();                                //输出异常信息
    }
}
```

（10）在客户端窗体类 ClientSocketFrame 中为文本框 tf_send 添加事件代码，实现向服务器端发送信息，该方法的代码如下：

```
tf_send.addActionListener(new ActionListener() {            //绑定事件
    public void actionPerformed(ActionEvent e) {
        writer.println(tf_send.getText());                  //文本框中信息写入流
        ta_info.append("客户端发送的信息是：" + tf_send.getText() + "\n");  //将文本框中信息显示在文本域中
        tf_send.setText("");                                //将文本框清空
```

```
        }
    });
```

注意：要使本程序正常运行，必须先运行服务器端程序，然后再运行客户端程序，否则程序会发生异常。

秘笈心法

心法领悟 505：实现服务器端与客户端相互通信。

在本实例的服务器端再添加一个文本框，即可在文本框事件中通过服务器套接字的输出流向客户端发送信息，同样，在客户端通过套接字的输入流就可以接收到服务器发送的信息，从而实现服务器端与客户端相互通信。

实例 506　关闭 Socket 缓冲

光盘位置：光盘\MR\506　　中级　　趣味指数：★★★

实例说明

在网络应用程序中，一般的网络通信是允许数据延迟的，而对于一些特殊的网络应用，例如需要实时监控的网络程序以及游戏，就必须禁用数据延迟，如果允许数据延迟，将会极大地影响数据传输的速度。本实例演示如何关闭 Socket 缓冲。运行程序，效果如图 18.8 所示。

图 18.8　关闭 Socket 缓冲

关键技术

本实例主要是通过 Socket 类的实例调用 setTcpNoDelay()方法将参数设置为 true 来关闭 Socket 缓冲的，该方法的定义如下：

```
public void setTcpNoDelay(boolean on) throws SocketException
```

参数说明

❶ on：为 true 表示关闭 Socket 缓冲；为 false 表示启用 Socket 缓冲。

❷ 使用该方法需要处理 SocketException 异常。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 ServerSocketFrame。

（3）在 ServerSocketFrame 窗体的内容面板上添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示服务器等待客户端连接的相关信息及如果连接成功则关闭 Socket 缓冲的信息。

（4）实现创建服务器套接字和设置关闭 Socket 缓冲的代码定义在 getServer()方法中，该方法的代码如下：

```
public void getserver() {
    try {
        server = new ServerSocket(1978);                    //实例化 Socket 对象
```

```
            ta_info.append("服务器套接字已经创建成功\n");                    //输出信息
            while (true) {                                                  //如果套接字是连接状态
                ta_info.append("等待客户机的连接.....\n");                   //输出信息
                ta_info.append("如果连接成功就会关闭 Socket 缓冲.....\n");   //输出信息
                socket = server.accept();                                   //实例化 Socket 对象
                socket.setTcpNoDelay(true);                                 //关闭 Socket 缓冲，提高数据传输速度
                ta_info.append("已经关闭 Socket 缓冲.....\n");               //输出信息
            }
        } catch (Exception e) {
            e.printStackTrace();                                            //输出异常信息
        }
    }
```

说明：本实例是在服务器端设置关闭套接字缓冲的，关闭客户端套接字缓冲的方法与服务器端相同，所以可以在创建客户端套接字后，调用 setTcpNoDelay()方法关闭客户端 Socket 缓冲。

秘笈心法

心法领悟 506：何时需要关闭 Socket 缓冲？

对于网络速度要求较高的应用程序，例如网络游戏、一些实时应用程序等，需要能够快速得到网络的反馈信息，这时需要关闭 Socket 缓冲，以达到更快的速度。

18.2 TCP 网络通信

实例 507	使用 Socket 通信	高级
	光盘位置：光盘\MR\507	趣味指数：★★★★

实例说明

本实例使用套接字实现服务器端与客户端的通信。运行程序，在服务器端的文本框中输入信息，然后按 Enter 键确认，客户端就会收到服务器端发送的信息；在客户端的文本框中输入信息，然后按 Enter 键确认，服务器端就会收到客户端发送的信息，发送信息后的效果如图 18.9 和图 18.10 所示。

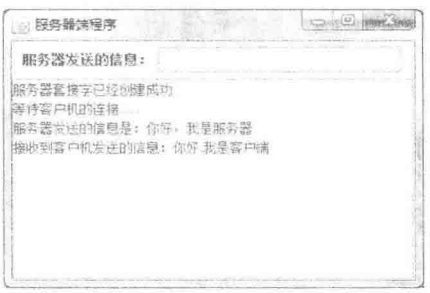


图 18.9 服务器端程序

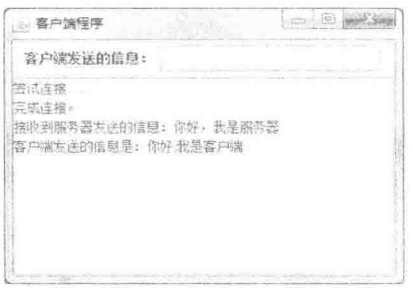


图 18.10 客户端程序

关键技术

本实例主要是通过 Socket 类的 getInputStream()方法获得输入流对象，并借助 InputStreamReader 类将输入流对象转换为 BufferedReader 对象读取接收到的信息，使用 getOutputStream()方法获得输出流对象，并创建了 PrintWriter 对象发送信息。

（1）InputStreamReader 类是字节流通向字符流的桥梁，该类的构造方法定义如下：

public InputStreamReader(InputStream in)

参数说明

in：字节输入流对象。

（2）BufferedReader 类是缓冲字符输入流，可以通过 InputStreamReader 类的实例作为 BufferedReader 类构造方法的参数创建 BufferedReader 对象，BufferedReader 类的构造方法定义如下：

public BufferedReader(Reader in)

参数说明

in：字符输入流对象。

（3）通过 PrintWriter 类的构造方法，以通过 Socket 类的 getOutputStream()方法获得的输出流对象作为参数创建 PrintWriter 对象，PrintWriter 类的构造方法定义如下：

public PrintWriter(OutputStream out, boolean autoFlush)

参数说明

❶ out：用通过 Socket 类的 getOutputStream()方法获得的输出流对象。

❷ autoFlush：如果为 true，则 println()、printf()或 format()方法将刷新输出缓冲区，反之则不刷新输出缓冲区。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ServerSocketFrame 和一个客户端窗体类 ClientSocketFrame。

（3）在服务器窗体类 ServerSocketFrame 的内容面板中央添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示客户端与服务器的连接信息以及接收到的客户端发送的信息；再在服务器窗体的内容面板上部位置添加一个 JPanel 面板控件，并在面板上添加一个 JLabel 标签和一个 JTextField 文本框，文本框的名称为 tf_send，用于向客户端发送信息。

（4）在客户端窗体类 ClientSocketFrame 的内容面板中部位置添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示连接信息以及客户端本身发送的信息；再在客户端窗体的内容面板上部位置添加一个 JPanel 面板控件，并在面板上添加一个 JLabel 标签和一个 JTextField 文本框，文本框的名称为 tf_send，用于向服务器端发送信息。

（5）在服务器窗体类 ServerSocketFrame 的成员声明区声明输出流、输入流、服务器套接字等成员变量，以便在其他方法中使用，代码如下：

```
private PrintWriter writer;                          //声明 PrintWriter 类对象
private BufferedReader reader;                       //声明 BufferedReader 对象
private ServerSocket server;                         //声明 ServerSocket 对象
private Socket socket;                               //声明 Socket 对象
```

（6）在服务器窗体类 ServerSocketFrame 中定义 getServer()方法，用于创建服务器端套接字、监听客户端程序，以及创建向客户端发送信息的输出流对象、用于接收客户端发送信息的输入流对象，该方法的代码如下：

```
public void getserver() {
    try {
        server = new ServerSocket(1978);                 //实例化 Socket 对象
        ta_info.append("服务器套接字已经创建成功\n");    //输出信息
        while (true) {                                   //如果套接字是连接状态
            ta_info.append("等待客户机的连接.....\n");   //输出信息
            socket = server.accept();                    //实例化 Socket 对象
            reader = new BufferedReader(new InputStreamReader(socket
                    .getInputStream()));                 //实例化 BufferedReader 对象
            writer = new PrintWriter(socket.getOutputStream(), true);
            getClientInfo();                             //调用 getClientInfo()方法
        }
    } catch (Exception e) {
        e.printStackTrace();                             //输出异常信息
    }
}
```

（7）在服务器窗体类 ServerSocketFrame 中再定义一个 getClientInfo()方法，用于接收客户端发送的信息以及关闭输入流对象和套接字对象，该方法的代码如下：

```
private void getClientInfo() {
    try {
        while (true) {                                          //如果套接字是连接状态
            String line = reader.readLine();                    //读取客户端发送的信息
            if (line != null)
                ta_info.append("接收到客户机发送的信息：" + line + "\n"); //获得客户端信息
        }
    } catch (Exception e) {
        ta_info.append("客户端已退出。\n");                      //输出异常信息
    } finally {
        try {
            if (reader != null) {
                reader.close();                                 //关闭流
            }
            if (socket != null) {
                socket.close();                                 //关闭套接字
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

（8）在服务器端窗体类 ServerSocketFrame 中为文本框 tf_send 添加事件代码，实现向客户端发送信息，其事件代码如下：

```
tf_send.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
        writer.println(tf_send.getText());                      //文本框中信息写入流
        ta_info.append("服务器发送的信息是：" + tf_send.getText() + "\n"); //将文本框中的信息显示在文本域中
        tf_send.setText(" ");                                   //将文本框清空
    }
});
```

（9）在客户端窗体类 ClientSocketFrame 的成员声明区声明输出流、输入流和套接字对象，以便在其他方法中使用，代码如下：

```
private PrintWriter writer;                                     //声明 PrintWriter 类对象
private BufferedReader reader;                                  //声明 BufferedReader 对象
private Socket socket;                                          //声明 Socket 对象
```

（10）在客户端窗体类 ClientSocketFrame 中定义 connect()方法，用于创建套接字对象输入流和输出流对象以及在文本域中显示连接信息和接收服务器端发送的信息，该方法的代码如下：

```
private void connect() {                                        //连接套接字方法
    ta_info.append("尝试连接.....\n");                           //文本域中信息
    try {                                                       //捕捉异常
        socket = new Socket("192.168.1.122", 1978);             //实例化 Socket 对象
        while (true) {
            writer = new PrintWriter(socket.getOutputStream(), true);  //创建输出流对象
            reader = new BufferedReader(new InputStreamReader(socket
                    .getInputStream()));                        //实例化 BufferedReader 对象
            ta_info.append("完成连接。\n");                      //文本域中提示信息
            getClientInfo();
        }
    } catch (Exception e) {
        e.printStackTrace();                                    //输出异常信息
    }
}
```

（11）在客户端窗体类 ClientSocketFrame 中再定义一个 getClientInfo()方法，用于接收服务器端发送的信息以及关闭输入流对象和套接字对象，该方法的代码如下：

```
private void getClientInfo() {
    try {
        while (true) {                                          //如果套接字是连接状态
            if (reader != null) {
```

```
                String line = reader.readLine();                          //读取服务器发送的信息
                if (line != null)
                    ta_info.append("接收到服务器发送的信息：" + line + "\n");    //获得客户端信息
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (reader != null) {
                reader.close();                                           //关闭流
            }
            if (socket != null) {
                socket.close();                                           //关闭套接字
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

（12）在客户端窗体类 ClientSocketFrame 中为文本框 tf_send 添加事件代码，实现向服务器端发送信息，该方法的代码如下：

```
tf_send.addActionListener(new ActionListener() {                          //绑定事件
    public void actionPerformed(ActionEvent e) {
        writer.println(tf_send.getText());                                //文本框中信息写入流
        ta_info.append("客户端发送的信息是：" + tf_send.getText()
                + "\n");                                                  //将文本框中的信息显示在文本域中
        tf_send.setText(" ");                                             //将文本框清空
    }
});
```

秘笈心法

心法领悟 507：实现 Socket 通信的原理。

要实现 Socket 通信，需要有一个服务器端程序和一个客户端程序，服务器端程序负责对指定的端口进行监听，并创建套接字对象；客户端则通过套接字与网络中的这个端口连接，如果连接成功就可以与服务器端通信，并且要先运行服务器端程序，再运行客户端程序。

实例 508　防止 Socket 传递汉字乱码

光盘位置：光盘\MR\508　　高级　　趣味指数：★★★

实例说明

在使用套接字进行网络编程时，使用 Socket 传递汉字有时会出现乱码，本实例将讲解如何防止出现汉字乱码。运行程序，效果如图 18.11 和图 18.12 所示。

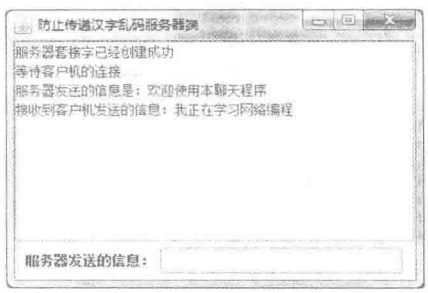


图 18.11　防止传递汉字乱码服务器端

图 18.12　防止传递汉字乱码客户端

关键技术

之所以会在传递中出现汉字乱码，是因为发送数据和接收数据所用的编码不同，因此，要解决传递汉字乱码问题，只需要在创建输入流和输出流对象时使用相同的编码，在使用 OutputStreamWriter 类和 InputStreamReader 类创建对象时，在构造方法中指定相同的编码。

（1）OutputStreamWriter 类是字符流通向字节流的桥梁。它使用指定的字符集将要写入流中的字符编码成字节。该类的构造方法定义如下：

```
public OutputStreamWriter(OutputStream out, String charsetName) throws UnsupportedEncodingException
```

参数说明

❶ out：字节输出流对象。

❷ charsetName：受支持的字符集名称。

❸ 使用该方法需要处理 UnsupportedEncodingException 异常。

（2）InputStreamReader 类是字节流通向字符流的桥梁。它使用指定的字符集读取字节并将其解码为字符。该类的构造方法定义如下：

```
public InputStreamReader(InputStream in, String charsetName) throws UnsupportedEncodingException
```

参数说明

❶ in：字节输入流对象。

❷ charsetName：受支持的字符集名称。

❸ 使用该方法需要处理 UnsupportedEncodingException 异常。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ServerSocketFrame 和一个客户端窗体类 ClientSocketFrame。

（3）在服务器窗体类 ServerSocketFrame 的内容面板中央添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示客户端与服务器的连接信息以及接收到的客户端发送的信息；再在服务器窗体的内容面板上部位置添加一个 JPanel 面板控件，并在面板上添加一个 JLabel 标签和一个 JTextField 文本框，文本框的名称为 tf_send，用于向客户端发送信息。

（4）在客户端窗体类 ClientSocketFrame 的内容面板中部位置添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示连接信息以及客户端本身发送的信息；再在客户端窗体的内容面板上部位置添加一个 JPanel 面板控件，并在面板上添加一个 JLabel 标签和一个 JTextField 文本框，文本框的名称为 tf_send，用于向服务器端发送信息。

（5）在服务器窗体类 ServerSocketFrame 中定义 getServer()方法，用于创建服务器端套接字、监听客户端程序，以及创建向客户端发送信息的输出流对象和用于接收客户端发送信息的输入流对象，该方法的代码如下：

```
public void getserver() {
    try {
        server = new ServerSocket(1978);                                    //实例化 Socket 对象
        ta_info.append("服务器套接字已经创建成功\n");                          //输出信息
        while (true) {                                                      //如果套接字是连接状态
            ta_info.append("等待客户机的连接.....\n");                        //输出信息
            socket = server.accept();                                       //实例化 Socket 对象
            reader = new BufferedReader(new InputStreamReader(socket
                    .getInputStream(), "UTF-8"));                           //实例化 BufferedReader 对象
            out = new OutputStreamWriter(socket.getOutputStream(), "UTF-8");//实例化 OutputStreamWriter 对象
            writer = new PrintWriter(out, true);                            //实例化 PrintWriter 对象
            getClientInfo();                                                //调用 getClientInfo()方法
        }
    } catch (Exception e) {
        e.printStackTrace();                                                //输出异常信息
    }
}
```

（6）在客户端窗体类 ClientSocketFrame 中定义 connect()方法，用于创建套接字对象输入流和输出流对象以及在文本域中显示连接信息和接收服务器端发送的信息，该方法的代码如下：

```
private void connect() {                                          //连接套接字方法
    ta_info.append("尝试连接.....\n");                              //文本域中信息
    try {                                                         //捕捉异常
        socket = new Socket("192.168.1.122", 1978);               //实例化 Socket 对象
        while (true) {
            out = new OutputStreamWriter(socket.getOutputStream(), "UTF-8");   //实例化 OutputStreamWriter 对象
            writer = new PrintWriter(out, true);                  //实例化 PrintWriter 对象
            reader = new BufferedReader(new InputStreamReader(socket
                    .getInputStream(), "UTF-8"));                 //实例化 BufferedReader 对象
            ta_info.append("完成连接。\n");                         //文本域中提示信息
            getClientInfo();                                      //调用方法
        }
    } catch (Exception e) {
        e.printStackTrace();                                      //输出异常信息
    }
}
```

秘笈心法

心法领悟 508：使用 Socket 传递汉字的原则。

由于使用 Socket 通信是通过 IO 流实现的，因此为了避免传递汉字乱码，只需记住一个原则，那就是通过什么字符集发送的数据，就使用什么字符集接收数据，这样就不会发生 Socket 传递汉字乱码的问题。

实例 509　使用 Socket 传递对象

光盘位置：光盘\MR\509　　高级　　趣味指数：★★★

实例说明

在使用套接字进行网络编程时，有时需要通过 Socket 传递对象，本实例讲解如何通过套接字传递对象。运行程序，效果如图 18.13 和图 18.14 所示。

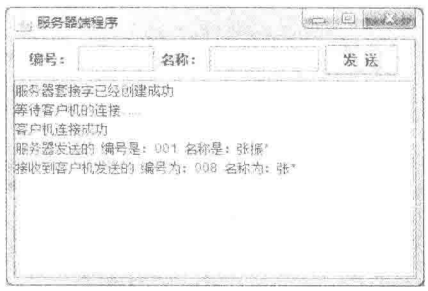


图 18.13　使用 Socket 传递对象服务器端

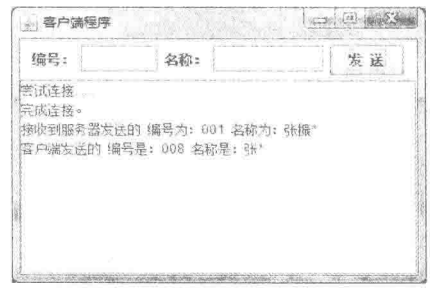


图 18.14　使用 Socket 传递对象客户端

关键技术

在 Java 中使用 Socket 传递对象时，该对象必须是序列化的。在 Java 中实现 Serializable 接口的类所创建的对象就是序列化对象，可以通过 Socket 进行传递，从而实现了使用 Socket 传递对象的功能。

Serializable 接口在 java.io 包中，该接口没有方法，只是用于标识对象是可序列化的，该接口的定义如下：

```
public interface Serializable
```

使用 Serializable 接口可以创建序列化类，例如，创建序列化类 Student，可以用如下代码实现：

```
public class Student implements Serializable {                   //序列化对象类
    //省略了类的成员
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个实现 Serializable 接口的序列化类 Student，该类的代码如下：

```
public class Student implements Serializable {                    //序列化对象类
    private String id;                                             //类的成员变量
    private String name;                                           //类的成员变量
    public String getId() {                                        //成员变量的 getter 方法
        return id;
    }
    public void setId(String id) {                                 //成员变量的 setter 方法
        this.id = id;
    }
    public String getName() {                                      //成员变量的 getter 方法
        return name;
    }
    public void setName(String name) {                             //成员变量的 setter 方法
        this.name = name;
    }
}
```

（3）在项目中创建一个继承自 JFrame 类的服务器窗体类 ServerSocketFrame 和一个客户端窗体类 ClientSocketFrame。

（4）在服务器窗体类 ServerSocketFrame 的内容面板中央添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示客户端与服务器的连接信息以及接收到的客户端发送的信息；再在服务器窗体的内容面板上部位置添加一个 JPanel 面板控件，并在面板上添加两个 JLabel 标签和两个 JTextField 文本框，文本框的名称分为 tf_id 和 tf_name，用于为 Student 类创建的对象传递参数值，并将 Student 对象发送到客户端。

（5）在客户端窗体类 ClientSocketFrame 的内容面板中部位置添加一个 JScrollPane 滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示客户端与服务器的连接是否成功以及接收到的服务器端发送的信息；再在客户端窗体的内容面板上部位置添加一个 JPanel 面板控件，并在面板上添加两个 JLabel 标签和两个 JTextField 文本框，文本框的名称分别为 tf_id 和 tf_name，用于为 Student 类创建的对象传递参数值，并将 Student 对象发送到服务器端。

（6）在服务器窗体类 ServerSocketFrame 中定义 getServer()方法，用于创建服务器端套接字、监听客户端程序，以及创建向客户端发送信息的输出流对象和用于接收客户端发送信息的输入流对象，该方法的关键代码如下：

```
server = new ServerSocket(1978);                                   //实例化 Socket 对象
ta_info.append("服务器套接字已经创建成功\n");                        //输出信息
while (true) {                                                     //如果套接字是连接状态
    ta_info.append("等待客户机的连接.....\n");                       //输出信息
    socket = server.accept();                                      //实例化 Socket 对象
    ta_info.append("客户机连接成功\n");                             //输出信息
    out = new ObjectOutputStream(socket.getOutputStream());        //创建输出流对象
    in = new ObjectInputStream(socket.getInputStream());           //创建输入流对象
    getClientInfo();                                               //调用 getClientInfo()方法
}
```

（7）在服务器窗体类 ServerSocketFrame 中定义 getClientInfo()方法，用于接收客户端发送的信息，该方法的关键代码如下：

```
while (true) {                                                     //如果套接字是连接状态
    Student stud = (Student)in.readObject();                       //将接收到的内容转换为 Student 类型
    if (stud!=null)
    ta_info.append("接收到客户机发送的  编号为：" + stud.getId() + "  名称为：" +stud.getName() + "\n");    //获得客户端信息
}
```

（8）在客户端窗体类 ClientSocketFrame 中，“发送”按钮的事件用于向服务器传递对象，其事件代码如下：

```
Student stud = new Student();                                      //创建 Student 对象
```

```
stud.setId(tf_id.getText());                    //将文本框中的内容设置为 Student 对象的 id 值
stud.setName(tf_name.getText());                //将文本框中的内容设置为 Student 对象的 name 值
try {
    out.writeObject(stud);                      //将 Student 对象发送到服务器
} catch (IOException e1) {
    e1.printStackTrace();                       //打印异常信息
}
ta_info.append("客户端发送的　编号是："+tf_id.getText()+"　名称是："+tf_name.getText()+"\n");      //将文本框中信息显示在文本域中
tf_id.setText(null);                            //将文本框清空
tf_name.setText(null);                          //将文本框清空
```

说明：上面代码使用套接字获得的输出流对象，将 Student 类创建的序列化对象发送到服务器端。

秘笈心法

心法领悟 509：Socket 可以传递的对象。

在使用 Java 语言编写网络程序时，有时需要使用 Socket 传递对象，但并不是所有的对象都能用 Socket 传递，只有序列化的对象才可以使用 Socket 传递。

实例 510　使用 Socket 传输图片

光盘位置：光盘\MR\510　　高级　　趣味指数：★★★★

实例说明

在使用套接字进行网络编程时，有时需要通过 Socket 传输图片，本实例讲解如何通过套接字传输图片。运行程序，在服务器端选择图片，单击“发送”按钮，就会将图片发送到客户端，效果如图 18.15 和图 18.16 所示，也可以在客户端选择图片，单击“发送”按钮，向服务器端发送图片。

图 18.15　服务器端选择图片并发送

图 18.16　客户端显示接收到的图片

关键技术

本实例通过使用 DataInputStream 类的 read()方法将图片文件读取到字节数组，然后使用 DataOutputStream 类从 DataOutput 类继承的 write()方法输出字节数组，从而实现使用 Socket 传输图片的功能。

（1）使用 DataInputStream 类的 read()方法可以将文件中的信息读取到字节数组，该方法的定义如下：

```
public final int read(byte[] b) throws IOException
```

参数说明

❶ b：存储读取信息的字节数组。

❷ 返回值：读取到的字节总数，如果已经是文件尾，则返回-1。

❸ 异常处理：使用该方法需要处理 IOException 异常。

（2）使用 DataOutputStream 类从 DataOutput 类继承的 write()方法，可以输出字节数组，该方法的定义

如下：

```
void write(byte[] b) throws IOException
```

参数说明

❶ b：需要写入输出流中的字节数组。

❷ 使用该方法需要处理 IOException 异常。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ServerSocketFrame 和一个客户端窗体类 ClientSocketFrame，并完成服务器和客户端窗体界面的设计。

（3）在服务器窗体类 ServerSocketFrame 中定义 getServer()方法，用于创建服务器端套接字、监听客户端程序，以及创建向客户端发送信息的输出流对象和用于接收客户端发送信息的输入流对象，该方法的关键代码如下：

```
server = new ServerSocket(1978);                          //实例化 Socket 对象
while (true) {                                            //如果套接字是连接状态
    socket = server.accept();                             //实例化 Socket 对象
    out = new DataOutputStream(socket.getOutputStream()); //获得输出流对象
    in = new DataInputStream(socket.getInputStream());    //获得输入流对象
    getClientInfo();                                      //调用 getClientInfo()方法
}
```

（4）在服务器窗体类 ServerSocketFrame 中定义 getClientInfo()方法，用于接收客户端发送的图片，该方法的关键代码如下：

```
long lengths = in.readLong();                             //读取图片文件的长度
byte[] bt = new byte[(int) lengths];                      //创建字节数组
for (int i = 0; i < bt.length; i++) {
    bt[i] = in.readByte();                                //读取字节信息并存储到字节数组
}
receiveImg = new ImageIcon(bt).getImage();                //创建图像对象
receiveImagePanel.repaint();                              //重新绘制图像
```

（5）在客户端窗体类 ClientSocketFrame 中，“发送”按钮的事件用于向服务器传输图片，其事件代码如下：

```
DataInputStream inStream = null;                          //定义数据输入流对象
if (imgFile != null) {
    lengths = imgFile.length();                           //获得选择图片的大小
    inStream = new DataInputStream(new FileInputStream(imgFile));  //创建输入流对象
} else {
    JOptionPane.showMessageDialog(null, "还没有选择图片文件。");
    return;
}
out.writeLong(lengths);                                   //将文件的大小写入输出流
byte[] bt = new byte[(int) lengths];                      //创建字节数组
int len = -1;
while ((len = inStream.read(bt)) != -1) {                 //将图片文件读取到字节数组
    out.write(bt);                                        //将字节数组写入输出流
}
```

秘笈心法

心法领悟 510：不要被假象迷惑。

说到使用 Socket 传输图片，很多程序员都会想到创建一个图片的 Image 对象，然后使用 Socket 对象传递该 Image 对象，但是由于 Image 不是序列化对象，因此不能使用 Socket 传递，所以这时必须考虑使用其他方法，例如可以使用本实例中讲解的方法，而不要被假象迷惑。

实例 511　使用 Socket 传输音频

光盘位置：光盘\MR\511　　高级　　趣味指数：★★★

实例说明

在使用套接字进行网络编程时，有时需要通过 Socket 传输音频文件，本实例讲解如何通过套接字传输音频文件。运行程序，在服务器端选择音频文件，单击“发送”按钮，即可将音频文件发送到客户端，在客户端弹出的“保存”对话框中指定文件的保存位置和文件名，单击“保存”按钮，即可完成音频文件的传输，效果如图 18.17 和图 18.18 所示，也可以在客户端选择音频文件，单击“发送”按钮，向服务器端发送音频文件。

图 18.17　服务器端选择音频并发送

图 18.18　客户端显示文件接收完毕

关键技术

本实例也是使用 DataInputStream 类的 read()方法和 DataOutputStream 类从 DataOutput 类继承的 write()方法实现了对音频文件的读写操作，与实例 510 不同的是，本实例使用保存对话框，将接收到的音频文件保存到接收方的主机上。

（1）有关 DataInputStream 类的 read()方法，以及 DataOutputStream 类从 DataOutput 类继承的 write()方法，请参考实例 510。

（2）本实例使用文件保存对话框，在接收方保存所接收到的音频文件。在 Java 中，将 FileDialog 对象的模式设置为 FileDailog.SAVE，这样该对话框就是文件保存对话框，创建文件保存对话框的代码如下：

```
FileDialog dialog = new FileDialog(ServerSocketFrame.this, "保存");          //创建对话框
dialog.setMode(FileDialog.SAVE);                                             //设置对话框模式为保存对话框
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ServerSocketFrame 和一个客户端窗体类 ClientSocketFrame，并完成服务器和客户端窗体界面的设计。

（3）在服务器窗体类 ServerSocketFrame 中定义 getServer()方法，用于创建服务器端套接字、监听客户端程序，以及创建向客户端发送信息的输出流对象和用于接收客户端发送信息的输入流对象，并显示相应的信息，该方法的关键代码如下：

```
server = new ServerSocket(1978);                                             //实例化 Socket 对象
ta_info.append("服务器套接字已经创建成功\n");                                  //输出信息
ta_info.append("等待客户机的连接.....\n");                                     //输出信息
socket = server.accept();                                                    //实例化 Socket 对象
ta_info.append("客户机连接成功.....\n");                                       //输出信息
while (true) {                                                               //如果套接字是连接状态
    if (socket != null && !socket.isClosed()) {
```

```
            out = new DataOutputStream(socket.getOutputStream());          //获得输出流对象
            in = new DataInputStream(socket.getInputStream());             //获得输入流对象
            getClientInfo();                                               //调用 getClientInfo()方法
        } else {
            socket = server.accept();                                      //实例化 Socket 对象
        }
    }
```

（4）在服务器窗体类 ServerSocketFrame 中定义 getClientInfo()方法，用于接收客户端发送的音频文件，并弹出“保存”对话框，保存接收到的音频文件，该方法的关键代码如下：

```
String name = in.readUTF();                                                //读取文件名
long lengths = in.readLong();                                              //读取文件的长度
byte[] bt = new byte[(int) lengths];                                       //创建字节数组
for (int i = 0; i < bt.length; i++) {
    bt[i] = in.readByte();                                                 //读取字节信息并存储到字节数组
}
FileDialog dialog = new FileDialog(ServerSocketFrame.this, "保存");        //创建对话框
dialog.setMode(FileDialog.SAVE);                                           //设置对话框为保存对话框
dialog.setFile(name);                                                      //设置保存对话框显示的文件名
dialog.setVisible(true);                                                   //显示保存对话框
String path = dialog.getDirectory();                                       //获得文件的保存路径
String newFileName = dialog.getFile();                                     //获得保存的文件名
if (path == null || newFileName == null) {
    return;
}
String pathAndName = path + "\\" + newFileName;                            //文件的完整路径
FileOutputStream fOut = new FileOutputStream(pathAndName);                 //创建输出流对象
fOut.write(bt);                                                            //向输出流写信息
fOut.flush();                                                              //更新缓冲区
fOut.close();                                                              //关闭输出流对象
ta_info.append("文件接收完毕。\n");
```

（5）在客户端窗体类 ClientSocketFrame 中，“发送”按钮的事件用于向服务器传输音频文件，其事件代码如下：

```
DataInputStream inStream = null;                                           //定义数据输入流对象
if (file != null) {
    lengths = file.length();                                               //获得所选择音频文件的大小
    inStream = new DataInputStream(new FileInputStream(file));             //创建输入流对象
} else {
    JOptionPane.showMessageDialog(null, "还没有选择音频文件。");
    return;
}
out.writeUTF(fileName);                                                    //写入音频文件名
out.writeLong(lengths);                                                    //将文件的大小写入输出流
byte[] bt = new byte[(int) lengths];                                       //创建字节数组
int len = -1;                                                              //用于存储读取到的字节数
while ((len = inStream.read(bt)) != -1) {                                  //将音频文件读取到字节数组
    out.write(bt);                                                         //将字节数组写入输出流
}
out.flush();                                                               //更新输出流对象
out.close();                                                               //关闭输出流对象
ta_info.append("文件发送完毕。\n");
```

秘笈心法

心法领悟 511：关于 TCP 协议的话题。

TCP 协议是一种以固接连线为基础的协议，提供两台计算机间可靠的数据传送。TCP 可以保证数据从一端传送至连接的另一端时能够确实送达，而且抵达的数据的排列顺序和送出时的顺序相同。因此 TCP 协议适合可靠性要求比较高的场合。就像拨打电话一样，必须先拨号给对方，等两端确定连接后，能听到对方说话，也知道对方回应的是什么。

实例 512　使用 Socket 传输视频

光盘位置：光盘\MR\512　　高级　　趣味指数：★★★★★

实例说明

在使用套接字进行网络编程时，有时需要通过 Socket 传输视频文件，本实例讲解如何通过套接字传输视频文件。运行程序，在服务器端或客户端选择视频文件，单击“发送”按钮，即可将视频文件发送给对方，并弹出“保存”对话框，然后由用户指定文件的保存位置和文件名，单击对话框中的“保存”按钮，完成视频文件的传输，如图 18.19 所示。

图 18.19　用于保存接收到的视频文件的“保存”对话框

关键技术

本实例与实例 511 基本相同，只是在文件选择对话框中指定的用于发送的文件类型不同，为了方便用户操作，在发送不同类型的文件时，为文件选择对话框指定相应的文件类型，下面是为文件选择对话框指定音频和视频的代码。

（1）为文件选择对话框指定音频类型的文件，代码如下：

```
JFileChooser fileChooser = new JFileChooser();                                          //创建文件选择器
FileFilter filter = new FileNameExtensionFilter("音频文件（WAV/MIDI/MP3/AU)", "WAV", "MID", "MP3", "AU");  //创建音频过滤器
fileChooser.setFileFilter(filter);                                                      //设置过滤器
int flag = fileChooser.showOpenDialog(null);                                            //显示打开对话框
```

说明：上面代码打开的文件选择对话框中，只显示文件类型是 WAV、MID、MP3 和 AU 类型的音频文件，这样可以方便用户对音频文件进行选择。

（2）为文件选择对话框指定视频类型的文件，代码如下：

```
JFileChooser fileChooser = new JFileChooser();                                          //创建文件选择器
FileFilter filter = new FileNameExtensionFilter("视频文件（AVI/MPG/DAT/RM)", "AVI", "MPG", "DAT", "RM");  //创建视频过滤器
fileChooser.setFileFilter(filter);                                                      //设置过滤器
int flag = fileChooser.showOpenDialog(null);                                            //显示打开对话框
```

说明：上面代码打开的文件选择对话框中，只显示文件类型是 AVI、MPG、DAT 和 RM 类型的视频文件，这样可以方便用户对视频文件进行选择。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ServerSocketFrame 和一个客户端窗体类 ClientSocketFrame，并完成服务器和客户端窗体界面的设计。

（3）在服务器窗体类 ServerSocketFrame 中定义 getServer()方法，用于创建服务器端套接字、监听客户端程序，以及创建向客户端发送信息的输出流对象和用于接收客户端发送信息的输入流对象，并显示相应的信息。

（4）在服务器窗体类 ServerSocketFrame 中定义 getClientInfo()方法，用于接收客户端发送的视频文件，并弹出“保存”对话框，保存接收到的视频文件。

（5）在客户端窗体类 ClientSocketFrame 中定义连接服务器的 connect()方法，接收服务器信息的 getServerInfo()方法，以及在“发送”按钮的事件中添加向服务器传输视频文件的代码，完成客户端程序的功能。

说明：由于本实例的代码与实例基本相同，只是在选择要发送的文件时所显示的文件类型不同，这里不再给出，如果需要可以查看源程序代码。

秘笈心法

心法领悟 512：使用 UDP 提高数据传输速度。

用户数据报（UDP）是网络信息传输的另一种形式。基于 UDP 的通信和基于 TCP 的通信不同，基于 UDP 的信息传递更快，但不提供可靠性保证。虽然 UDP 是一种不可靠的协议，但在需要较快地传输信息、可以容忍小的错误时，可以考虑使用。

实例 513　一个服务器与一个客户端通信　中级

光盘位置：光盘\MR\513　趣味指数：★★★★

实例说明

在使用套接字进行网络编程时，需要在服务器和客户端之间进行通信，本实例讲解如何实现一个服务器与一个客户端进行通信。运行程序，效果如图 18.20 和图 18.21 所示。

图 18.20　服务器端发送和接收信息的效果

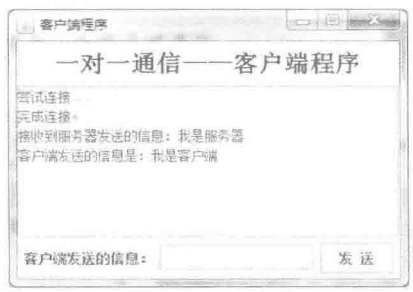


图 18.21　客户端接收和发送信息的效果

关键技术

本实例在服务器端和客户端没有使用线程来处理接收到的信息，所以当有多个客户连接到服务器时，只有第一个客户能够与服务器进行通信，而其他客户必须等待，只有第一个客户退出，服务器才能与下一个客户进行通信，依此类推。

（1）服务器端通过 getClientInfo()方法接收客户端发送的信息，该方法的关键代码如下：

```
private void getClientInfo() {
    try {
        while (true) {
            String line = reader.readLine();                          //读取客户端发送的信息
            if (line != null)
                ta_info.append("接收到客户机发送的信息：" + line + "\n");   //显示客户端发送的信息
        }
    } catch (Exception e) {
        ta_info.append("客户端已退出。\n");                              //输出异常信息
    }
}
```

（2）客户端通过 getServerInfo()方法接收服务器端发送的信息，该方法的关键代码如下：

```
private void getServerInfo() {
    try {
        while (true) {
            if (reader != null) {
                String line = reader.readLine();                      //读取服务器端发送的信息
                if (line != null)
                    ta_info.append("接收到服务器发送的信息：" + line + "\n");  //显示服务器端发送的信息
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ServerSocketFrame 和一个客户端窗体类 ClientSocketFrame。

（3）在服务器窗体类 ServerSocketFrame 中定义 getServer()方法，用于创建服务器端套接字、监听客户端程序、创建向客户端发送信息的输出流对象和用于接收客户端发送信息的输入流对象，该方法的关键代码如下：

```
server = new ServerSocket(1978);                                  //实例化 Socket 对象
ta_info.append("服务器套接字已经创建成功\n");                        //输出信息
while (true) {                                                    //如果套接字是连接状态
    ta_info.append("等待客户机的连接.....\n");                       //输出信息
    socket = server.accept();                                     //实例化 Socket 对象
    reader = new BufferedReader(new InputStreamReader(socket
            .getInputStream()));                                  //实例化 BufferedReader 对象
    writer = new PrintWriter(socket.getOutputStream(), true);     //实例化 PrintWriter 对象
    getClientInfo();                                              //调用 getClientInfo()方法
}
```

（4）在服务器窗体类 ServerSocketFrame 中，“发送”按钮用于向客户端发送信息，其事件代码如下：

```
writer.println(tf_send.getText());                                //将文本框中信息写入流
ta_info.append("服务器发送的信息是：" + tf_send.getText() + "\n");   //将文本框中信息显示在文本域中
tf_send.setText("");                                              //将文本框清空
```

（5）在客户端窗体类 ClientSocketFrame 中定义 connect()方法，用于创建与服务器连接的套接字对象、输入流对象和输出流对象，以及在文本域中显示与服务器的连接信息和接收到服务器端发送的信息，该方法的关键代码如下：

```
socket = new Socket("192.168.1.122", 1978);                       //实例化 Socket 对象
while (true) {
    writer = new PrintWriter(socket.getOutputStream(), true);     //实例化 PrintWriter 对象
    reader = new BufferedReader(new InputStreamReader(socket
            .getInputStream()));                                  //实例化 BufferedReader 对象
    ta_info.append("完成连接。\n");                                  //文本域中提示信息
    getServerInfo();
}
```

（6）在客户端窗体类 ServerSocketFrame 中，“发送”按钮用于向服务器端发送信息，其事件代码如下：

```
writer.println(tf_send.getText());                          //将文本框中信息写入流
ta_info.append("客户端发送的信息是：" + tf_send.getText() + "\n");  //将文本框中信息显示在文本域中
tf_send.setText("");                                        //将文本框清空
```

秘笈心法

心法领悟 513：服务器与客户端一对一通信的原理。

服务器与客户端一对一通信的原理是，服务器端启动后对客户端进行监听，如果有客户端连接到服务器，即可与客户端通信，并且在客户端没有退出时，再打开其他客户端程序，则只有第一个连接到服务器的客户端程序能接收到服务器的信息，同样服务器也只能接收到第一个客户端发送的信息。如果第一个客户端退出了，则第二个连接到服务器的客户端可以与服务器通信，依此类推。这主要是由于服务器端的所有操作都是在一个主线程中完成的。

实例 514　一个服务器与多个客户端通信　高级

光盘位置：光盘\MR\514　趣味指数：★★★★

实例说明

在使用套接字进行网络编程时，需要在服务器和客户端之间进行通信，本实例讲解如何通过一个服务器与多个客户端进行通信。运行程序，服务器启动后，启动两个客户端程序，然后通过服务器向客户端发送信息，两个客户端都会收到服务器发送的信息，效果如图 18.22 和图 18.23 所示。

图 18.22　第一个客户端接收到的服务器信息

图 18.23　第二个客户端接收到的服务器信息

关键技术

本实例在服务器端通过线程来处理不同客户发送的信息，所以当有多个客户连接到服务器时，服务器会为每个客户建立一个线程来处理接收到的信息，而不会产生阻塞，从而实现了一个服务器与多个客户端通信，并且在关闭客户端窗体时，会向服务器端发送退出客户端的索引。

（1）在服务器端创建线程类 ServerThread，用于接收客户端发送的信息以及处理客户端的退出信息，该线程类中 run()方法的关键代码如下：

```
public void run() {
    try {
        if (socket != null) {
            reader = new BufferedReader(new InputStreamReader(socket.getInputStream())); //实例化 BufferedReader 对象
            int index = -1;                                                               //存储退出的客户端索引值
            try {
                while (true) {                                                            //如果套接字是连接状态
                    String line = reader.readLine();                                      //读取客户端信息
                    try {
                        index = Integer.parseInt(line);                                   //获得退出的客户端索引值
                    } catch (Exception ex) {
                        index = -1;
```

```
                    }
                    if (line != null) {
                        ta_info.append("接收到客户机发送的信息：" + line + "\n");        //获得客户端信息
                    }
                }
            } catch (Exception e) {
                if (index != -1) {
                    vector.set(index, null);                                          //将退出的客户端套接字设置为 null
                    ta_info.append("第" + (index + 1) + "个客户端已经退出。\n");        //输出异常信息
                }
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

（2）客户端窗体的关闭事件，用于向服务器发送退出客户的索引值，客户端窗体的关闭事件代码如下：

```
addWindowListener(new WindowAdapter() {
    public void windowClosing(final WindowEvent e) {                                 //窗体关闭事件
        writer.println(String.valueOf(index));                                        //向服务器端发送退出客户的索引值
    }
});
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ServerSocketFrame 和一个客户端窗体类 ClientSocketFrame。

（3）在服务器窗体类 ServerSocketFrame 中定义 getServer()方法，用于创建服务器端套接字、监听客户端程序、创建向客户端发送信息的输出流对象，并向客户端发送连接用户的套接字索引以及创建并启动线程对象，用于接收客户端发送的信息，该方法的关键代码如下：

```
server = new ServerSocket(1978);                                                      //实例化 Socket 对象
ta_info.append("服务器套接字已经创建成功\n");                                          //输出信息
while (true) {                                                                        //如果套接字是连接状态
    socket = server.accept();                                                         //实例化 Socket 对象
    counts++;                                                                         //计算连接客户总数
    ta_info.append("第" + counts + "个客户连接成功\n");                                //输出信息
    PrintWriter out = new PrintWriter(socket.getOutputStream(), true);                //创建输出流对象
    out.println(String.valueOf(counts - 1));                                          //向客户端发送套接字索引
    vector.add(socket);                                                               //存储客户端套接字对象
    new ServerThread(socket).start();                                                 //创建并启动线程对象
}
```

（4）在客户端窗体类 ClientSocketFrame 中定义 connect()方法，用于创建与服务器连接的套接字对象、输入流对象和输出流对象，从服务器读取客户端的索引以及在文本域中显示是第几个与服务器连接的用户和接收服务器端发送的信息，该方法的关键代码如下：

```
socket = new Socket("192.168.1.122", 1978);                                           //实例化 Socket 对象
while (true) {
    writer = new PrintWriter(socket.getOutputStream(), true);                         //创建输出流对象
    reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));      //实例化 BufferedReader 对象
    index = Integer.parseInt(reader.readLine());                                      //获得客户登录服务器的索引值
    ta_info.append("你是第"+(index+1)+"个完成连接的用户。\n");                          //文本域中提示信息
    getServerInfo();                                                                  //调用接收服务器信息的方法
}
```

秘笈心法

心法领悟 514：服务器与客户端一对多通信的原理。

由于服务器与客户端一对一通信存在不足，可以使用服务器与客户端一对多进行通信，其原理是在服务器端为每个客户建立一个线程，这样服务器就可以向多个客户端发送信息了，同样，每个客户端也都可以向服务

器发送信息，而与客户端程序连接到服务器的顺序无关。

实例 515 客户端一对多通信

光盘位置：光盘\MR\515　　高级　　趣味指数：★★★★★

实例说明

在使用套接字进行网络编程时，有时需要在不同的客户端之间进行通信，其中有一种通信方式就是一个客户端与其他的多个客户端进行通信。本实例讲解如何实现一个客户端与其他多个客户端进行通信。运行程序，服务器启动后，启动 3 个客户端程序，然后通过第一个客户端向另外两个客户端发送信息，则另外两个客户端都会收到服务器发送的信息，效果如图 18.24 和图 18.25 所示。

图 18.24　第二个客户端接收到第一个客户端的信息

图 18.25　第三个客户端接收到第一个客户端的信息

关键技术

本实例主要是在服务器端通过线程对客户端发送的信息进行监听，当有客户端发送信息时，就会将该信息发送给其他已经登录到服务器的客户端，但是不会向发送方发送该信息，在客户端也通过线程来监听服务器转发的信息。

（1）在服务器端创建线程类 ServerThread，用于接收客户端发送的信息，并转发给其他已经连接到服务器的客户端，该线程类中 run()方法的关键代码如下：

```
while (true) {
    String info = in.readLine();                                    //读取信息
    for (Socket s : vector) {                                       //遍历所有客户端套接字对象
        if (s != socket) {                                          //如果不是发送信息的套接字对象
            PrintWriter out = new PrintWriter(s .getOutputStream(), true);  //创建输出流对象
            out.println(info);                                      //发送信息
            out.flush();                                            //刷新输出缓冲区
        }
    }
}
```

（2）在客户端创建线程类 ClientThread，用于接收服务器端转发的客户端信息，并在客户端的文本域中显示接收到的信息，该线程类中 run()方法的关键代码如下：

```
while (true) {
    String info = in.readLine();                                    //读取信息
    ta_info.append(info + "\n");                                    //在文本域中显示信息
    if (info.equals("88")) {                                        //如果接收到的信息是 88，则结束线程的执行
        break;                                                      //结束线程
    }
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ClientOneToMany_ServerFrame 和一个客户端窗体类 ClientOneToMany_ClientFrame。

（3）在服务器窗体类 ClientOneToMany_ServerFrame 中定义 createSocket()方法，用于创建服务器端套接字、监听客户端程序，以及创建并启动线程对象将接收到客户端发送的信息转发给其他客户端，该方法的关键代码如下：

```
server = new ServerSocket(1978);
while (true) {
    ta_info.append("等待新客户连接.....\n");
    socket = server.accept();                          //创建套接字对象
    vector.add(socket);                                //将套接字对象添加到向量对象中
    ta_info.append("客户端连接成功。" + socket + "\n");
    new ServerThread(socket).start();                  //创建并启动线程对象
}
```

（4）在客户端窗体类 ClientOneToMany_ClientFrame 中定义 createClientSocket()方法，用于创建与服务器连接的套接字对象、创建输出流对象以及启动线程对象接收服务器端转发的信息，该方法的关键代码如下：

```
Socket socket = new Socket("192.168.1.122", 1978);          //创建套接字对象
out = new PrintWriter(socket.getOutputStream(), true);      //创建输出流对象
new ClientThread(socket).start();                           //创建并启动线程对象
```

秘笈心法

心法领悟 515：允许客户端接收自己发送的信息。

本实例运行后，客户端发送的信息能够被其他客户端接收，但是本身并不能接收到自己发送的信息，如果允许客户端接收自己发送的信息，可以将服务器端线程类 run()中的 if 条件语句去掉。

实例 516　客户端一对一通信

光盘位置：光盘\MR\516　　中级　　趣味指数：★★★★

实例说明

在使用套接字进行网络编程时，有时需要在不同的客户端之间进行通信，其中有一种通信方式就是一个客户端与另一个指定的客户端进行通信。本实例讲解如何实现一个客户端与另一个指定的客户端进行通信。运行程序，服务器启动后，启动 3 个客户端程序，并分别以 aaa、bbb 和 ccc 进行登录，然后在左侧的用户列表中选择接收信息的用户，输入聊天信息即可发送到目标用户，效果如图 18.26 和图 18.27 所示。

图 18.26　客户端 aaa 接收到 bbb 的信息

图 18.27　客户端 bbb 接收到 aaa 的信息

关键技术

本实例主要是在服务器端通过线程对客户端发送的信息进行监听，并对登录用户和消息分别进行处理，如

果是登录用户，就将所有用户添加到客户端的用户列表中，如果是消息，就转发给指定的用户；客户端则通过线程对接收到的信息进行处理，如果是登录用户，就添加到用户列表中，如果是消息，就追加到文本域中。

（1）在服务器端创建线程类 ServerThread，用于对登录用户和消息分别进行处理，如果是登录用户，就将所有用户添加到客户端的用户列表中；如果是消息，就转发给指定的用户，该线程类 run()方法中用于判断是登录用户还是消息的关键代码如下：

```
if (info.startsWith("用户：")) {                    //添加登录用户到客户端列表
    //省略了向客户端用户列表添加登录用户的代码
} else {                                            //转发接收的消息
    //省略了向客户端指定用户发送信息的代码
}
```

（2）在客户端创建线程类 ClientThread，用于对接收到的信息进行处理，如果是登录用户，就添加到用户列表中；如果是消息，就追加到文本域中，该线程类 run()方法中用于判断接收到的是登录用户还是消息的关键代码如下：

```
if (!info.startsWith("MSG:")) {                     //如果接收到的不是消息，即是登录用户
    //省略了向用户列表添加登录用户的代码
} else {                                            //如果接收到的是消息
    ta_info.append(info + "\n");                    //在文本域中显示信息
    if (info.equals("88")) {
        break;                                      //结束线程
    }
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ClientOneToOne_ServerFrame 和一个客户端窗体类 ClientOneToOne_ClientFrame。

（3）在服务器窗体类 ClientOneToMany_ServerFrame 中定义 ServerThread 线程类，在该线程类的 run()方法中有一部分代码用于为客户端添加用户列表，其关键代码如下：

```
if (info.startsWith("用户：")) {                    //添加登录用户到客户端列表
    key = info.substring(3, info.length());         //获得用户名并作为键使用
    map.put(key, socket);                           //添加键值对
    Set<String> set = map.keySet();                 //获得集合中所有键的 Set 视图
    Iterator<String> keyIt = set.iterator();        //获得所有键的迭代器
    while (keyIt.hasNext()) {
        String receiveKey = keyIt.next();           //获得表示接收信息的键
        Socket s = map.get(receiveKey);             //获得与该键对应的套接字对象
        PrintWriter out = new PrintWriter(s
                .getOutputStream(), true);          //创建输出流对象
        Iterator<String> keyIt1 = set.iterator();   //获得所有键的迭代器
        while (keyIt1.hasNext()) {
            String receiveKey1 = keyIt1.next();     //获得键，用于向客户端添加用户列表
            out.println(receiveKey1);               //发送信息
            out.flush();                            //刷新输出缓冲区
        }
    }

} else {                                            //转发接收到的消息
    //省略了转发接收消息的代码
}
```

（4）在服务器窗体类 ClientOneToMany_ServerFrame 的线程类 ServerThread 中，run()方法中有一部分代码用于转发客户端发送的消息，其关键代码如下：

```
if (info.startsWith("用户：")) {                    //添加登录用户到客户端列表
    //省略了向客户端添加用户列表的代码
} else {                                            //转发接收的消息
    key = info.substring(info.indexOf("：发送给：") + 5, info
            .indexOf("：的信息是："));              //获得接收方的 key 值，即接收方的用户名
    String sendUser = info.substring(0, info
```

```
                .indexOf(":  发送给:  "));                    //获得发送方的 key 值，即发送方的用户名
        Set<String> set = map.keySet();                       //获得集合中所有键的 Set 视图
        Iterator<String> keyIt = set.iterator();              //获得所有键的迭代器
        while (keyIt.hasNext()) {
            String receiveKey = keyIt.next();                 //获得表示接收信息的键
            if (key.equals(receiveKey)
                    && !sendUser.equals(receiveKey)) {        //如果是发送方，但不是用户本身
                Socket s = map.get(receiveKey);               //获得与该键对应的套接字对象
                PrintWriter out = new PrintWriter(s
                        .getOutputStream(), true);            //创建输出流对象

                out.println("MSG:"+info);                     //发送信息
                out.flush();                                  //刷新输出缓冲区
            }
        }
    }
```

（5）在客户端窗体类 ClientOneToOne_ClientFrame 中定义 ClientThread 线程类，用于对接收到的服务器的信息进行处理，如果是登录用户，就添加到用户列表中；如果是消息，就追加到文本域中，该线程类的 run()方法实现该功能的关键代码如下：

```
    if (!info.startsWith("MSG:")) {
        boolean itemFlag = false;                             //标记是否为列表框添加列表项，为 true 不添加，为 false 添加
        for (int i = 0; i < model.getSize(); i++) {
            if (info.equals((String) model.getElementAt(i))) {
                itemFlag = true;                              //标记为 true，表示是用户
            }
        }
        if (!itemFlag) {
            model.addElement(info);                           //添加列表项
        } else {
            itemFlag = false;                                 //标记设置为 false
        }
    } else {
        ta_info.append(info + "\n");                          //在文本域中显示信息
        if (info.equals("88")) {
            break;                                            //结束线程
        }
    }
```

秘笈心法

心法领悟 516：在客户端显示自己发送的信息。

本实例中，客户端发送的信息能够被某个指定的客户端接收，但是客户端本身并没有显示自己发送的信息，如果希望在客户端显示自己发送的信息，可以在客户端发送信息时，向文本域追加发送者、接收者和发送的信息。

实例 517　基于 Socket 的数据库编程

中级

光盘位置：光盘\MR\517　　趣味指数：★★★

实例说明

本实例实现了基于 Socket 的数据库编程。运行程序，在客户端输入信息，单击“保存”按钮，将通过 Socket 将信息发送到服务器端，服务器端接收到信息后，将数据信息保存到数据库中，并反馈给客户端程序，效果如图 18.28 和图 18.29 所示。

关键技术

本实例通过 Socket 将客户端的信息发送到服务器端，然后服务器端对接收到的信息进行处理，并通过 JDBC-ODBC 桥的方式连接到 Access 数据库，然后通过 JDBC 技术将数据信息保存到数据库中。

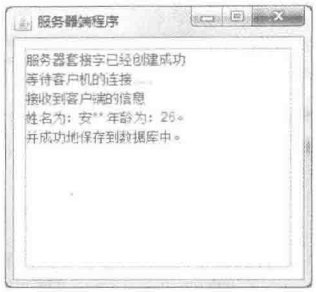


图 18.28　将客户端发送的信息保存到数据库中

图 18.29　获得服务器端保存成功的反馈信息

（1）使用 JDBC-ODBC 桥加载数据库驱动，其实现代码如下：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");                    //加载数据库驱动
```

（2）连接到数据库 c:/database/db_picture.mdb 的代码如下：

```
Connection conn = null;                                            //定义数据库连接
String url = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ= src/database/db_picture.mdb ";   //数据库 db_picture.mdb 的 URL
String username = "";                                              //数据库的用户名
String password = "";                                              //数据库的密码
conn = DriverManager.getConnection(url, username, password);       //建立连接
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个 DAO 类，用于加载数据库驱动及建立到数据库的连接，该类的关键代码如下：

```
public DAO() {
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");                 //加载数据库驱动
    } catch (ClassNotFoundException e) {
        JOptionPane.showMessageDialog(null, "数据库驱动加载失败。\n" + e.getMessage());
    }
}
public static Connection getConn() {
    try {
        Connection conn = null;                                        //定义数据库连接
        String url = " jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=src/database/db_picture.mdb";   //数据库 db_picture.mdb 的 URL
        String username = "";                                          //数据库的用户名
        String password = "";                                          //数据库的密码
        conn = DriverManager.getConnection(url, username, password);   //建立连接
        return conn;                                                   //返回连接
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "数据库连接失败。\n" + e.getMessage());
        return null;
    }
}
```

（3）在项目中创建一个继承自 JFrame 类的服务器窗体类 DatabaseServerFrame 和一个客户端窗体类 DatabaseClientFrame。

（4）在服务器窗体类 DatabaseServerFrame 中定义一个 getClientInfo()方法，用于接收客户端发送的信息，并将接收到的信息保存到数据库中，然后反馈给客户端程序，该方法的关键代码如下：

```
BufferedReader reader;                                             //声明 BufferedReader 对象
while (true) {                                                     //如果套接字是连接状态
    reader = new BufferedReader(new InputStreamReader(socket
            .getInputStream()));                                   //实例化 BufferedReader 对象
    String line = reader.readLine();                               //读取客户端信息
    if (line != null) {
        String[] value = new String[2];                            //创建数组
        value[0] = line.substring(0, line.indexOf(":data:"));      //获得姓名
        value[1] = line.substring(line.indexOf(":data:") + 6);     //获得年龄
        ta_info.append("接收到客户端的信息\n 姓名为："+value[0]+" 年龄为："+value[1]+"。\n");
        try {
```

```
                Connection conn = DAO.getConn();                          //获得数据库连接
                String sql = "insert into tb_employee (name,age) values(?,?)";   //定义 SQL 语句
                PreparedStatement ps = conn.prepareStatement(sql);        //创建 PreparedStatement 对象，并传递 SQL 语句
                ps.setString(1, value[0]);                                //为第 1 个参数赋值
                ps.setInt(2, Integer.parseInt(value[1]));                 //为第 2 个参数赋值
                int flag = ps.executeUpdate();                            //执行 SQL 语句，获得更新记录数
                ps.close();                                               //关闭 PreparedStatement 对象
                conn.close();                                             //关闭连接
                if (flag > 0) {
                    ta_info.append("并成功地保存到数据库中。\n");
                    writer.println("保存成功。");                          //向客户端输出保存成功的信息
                } else {
                    writer.println("保存失败。\n");                        //向客户端输出保存失败的信息
                }
            } catch (SQLException ee) {
                writer.println("保存失败。\n" + ee.getMessage());          //向客户端输出保存失败的信息
            }
        }
    }
```

说明：由于本实例的其他代码与前几个实例相似，这里不再给出，如果需要请查看源程序代码。

秘笈心法

心法领悟 517：胖服务器端的 Socket 数据库编程。

本实例使用的是胖服务器端的 Socket 数据库编程。胖服务器端的数据库编程是指客户端只是简单地将数据发送到服务器，而服务器端则负责数据库的连接、保存数据等操作，并将保存结果发送给客户端。

实例 518　使用 Proxy 创建代理服务器

光盘位置：光盘\MR\518　　高级　　趣味指数：★★★★

实例说明

本实例实现了使用 Proxy 创建的代理服务器访问网络资源。运行程序，输入代理服务器的地址和端口号及要访问的网站网址，就会在“访问结果”标签下面的文本域中显示相应的信息，效果如图 18.30 所示。

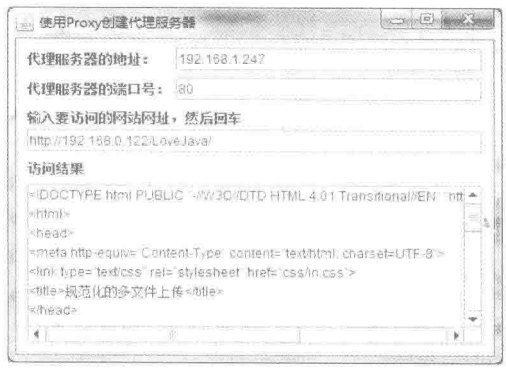


图 18.30　使用 Proxy 创建代理服务器

关键技术

本实例使用 Proxy 类创建代理，并在使用 URL 类的 openConnection()方法打开连接时，将该代理传递给 openConnection()方法，从而实现了使用代理服务器的功能。

（1）使用 Proxy 类可以创建代理，代码如下：

```
proxy = new Proxy(Proxy.Type.HTTP, new InetSocketAddress(proxyAddress, proxyPort));          //创建代理
```

说明：上面代码中的 Proxy.Type.HTTP 用于指定代理类型为 HTTP 代理；InetSocketAddress 是代理套接字地址，用于指定代理服务器和端口号。

（2）通过代理服务器打开连接，代码如下：

```
URLConnection urlConn = url.openConnection(proxy);                                          //通过代理服务器打开连接
```

说明：上面代码中的 url 是要访问网站的 URL 对象；proxy 是所创建的代理；urlConn 是所要打开连接的 URLConnection 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 UserProxyFrame，并完成窗体界面的设计。

（3）在窗体类 UserProxyFrame 中定义 accessUrl()方法，用于实现通过代理访问网络资源，该方法的代码如下：

```
url = new URL(accessAddress);                                                   //创建 URL 对象
proxy = new Proxy(Proxy.Type.HTTP, new InetSocketAddress(proxyAddress, proxyPort));  //创建代理
urlConn = url.openConnection(proxy);                                            //通过代理打开连接
scanner = new Scanner(urlConn.getInputStream(), "UTF8");                        //通过流创建扫描器
ta_show.setText(null);                                                          //清空文本域的内容
StringBuffer sb = new StringBuffer();                                           //创建字符串缓存
while (scanner.hasNextLine()) {                                                 //判断扫描器是否有数据
    String line = scanner.nextLine();                                           //从扫描器获得一行数据
    sb.append(line + "\n");                                                     //向字符串缓存添加信息
}
if (sb != null) {
    ta_show.append(sb.toString());                                              //在文本域中显示信息
}
```

（4）在输入网址文本框的动作事件中添加用于调用 accessUrl()方法的代理，将读取的网络资源显示到文本域中，其事件代码如下：

```
try {
    String proxyAddress = tf_proxyAddress.getText().trim();                     //代理服务器的地址
    int proxyPort = Integer.parseInt(tf_proxyPort.getText().trim());            //代理服务器的端口
    String accessAddress = tf_accessAddress.getText().trim();                   //需要打开的网站网址
    accessUrl(proxyAddress, proxyPort, accessAddress);                          //调用方法，使用代理访问网站
} catch (Exception ex) {
    JOptionPane.showMessageDialog(null, "输入的信息有误。");
}
```

秘笈心法

心法领悟 518：代理服务器的原理。

使用代理服务器是为了保证网络资源的安全，当客户端访问网站时，使用代理服务器访问服务器，然后代理服务器将信息返回给客户端浏览器，但是使用代理服务器是有时限的，有时需要多次尝试才能成功。

实例 519 使用 ProxySelector 选择代理服务器

高级

光盘位置：光盘\MR\519

趣味指数：★★★

实例说明

本实例实现了使用 ProxySelector 选择代理服务器的功能。运行程序，输入代理服务器的地址，然后输入要

访问的网站网址，就会在“访问结果”标签下面的文本域中显示相应的信息，效果如图 18.31 所示。

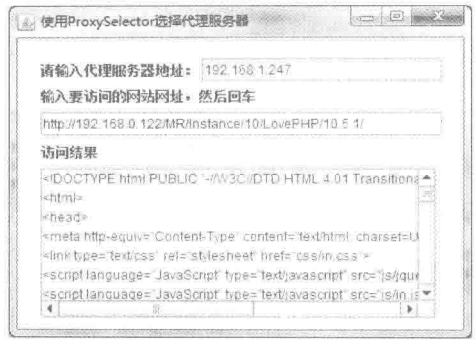


图 18.31　使用 ProxySelector 选择代理服务器

关键技术

在 Java 中，可以通过实现 ProxySelector 类，指定连接到 URL 引用的网络资源时，选择要使用的代理服务器，但是如果为本地系统设置了多个代理服务器，则可以不实现 ProxySelector 类，系统会使用默认的 ProxySelector 选择代理服务器。本实例就是使用默认的 ProxySelector 选择代理服务器，代码如下：

```
Properties propperties = System.getProperties();                    //获得系统属性对象
propperties.setProperty("http.proxyHost", proxyAddress);            //设置 HTTP 服务使用的代理服务器地址
propperties.setProperty("http.proxyPort", "80");                    //设置 HTTP 服务使用的代理服务器端口
propperties.setProperty("https.proxyHost", proxyAddress);           //设置安全 HTTP 服务使用的代理服务器地址
propperties.setProperty("https.proxyPort", "443");                  //设置安全 HTTP 服务使用的代理服务器端口
propperties.setProperty("ftp.proxyHost", proxyAddress);             //设置 FTP 访问的代理服务器地址
propperties.setProperty("ftp.proxyPort", "21");                     //设置 FTP 访问的代理服务器端口
propperties.setProperty("socks.ProxyHost", proxyAddress);           //设置 SOCKS 代理服务器的地址
propperties.setProperty("socks.ProxyPort", "1080");                 //设置 SOCKS 代理服务器的端口
```

说明：上面代码指定了 4 种类型的代理，并使用了各服务器的默认端口，其中所使用的代理包括 HTTP、安全 HTTP、FTP 以及 SOCKS 服务使用的代理，这样设置完代理，程序就会根据访问的内容选择相应的代理。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 ProxySelectorFrame，并完成窗体界面的设计。

（3）在窗体类 ProxySelectorFrame 中定义 setProxyInfo()方法，用于为本地系统设置可以选择的代理，该方法的代码如下：

```
public void setProxyInfo(String proxyAddress) {
    Properties propperties = System.getProperties();                //获得系统属性对象
    propperties.setProperty("http.proxyHost", proxyAddress);        //设置 HTTP 服务使用的代理服务器地址
    propperties.setProperty("http.proxyPort", "80");                //设置 HTTP 服务使用的代理服务器端口
    propperties.setProperty("https.proxyHost", proxyAddress);       //设置安全 HTTP 服务使用的代理服务器地址
    propperties.setProperty("https.proxyPort", "443");              //设置安全 HTTP 服务使用的代理服务器端口
    propperties.setProperty("ftp.proxyHost", proxyAddress);         //设置 FTP 访问的代理服务器地址
    propperties.setProperty("ftp.proxyPort", "21");                 //设置 FTP 访问的代理服务器端口
    propperties.setProperty("socks.ProxyHost", proxyAddress);       //设置 SOCKS 代理服务器的地址
    propperties.setProperty("socks.ProxyPort", "1080");             //设置 SOCKS 代理服务器的端口
}
```

（4）在窗体类 ProxySelectorFrame 中定义 removeProxyInfo()方法，用于从本地系统中移除所设置的代理，该方法的代码如下：

```
public void removeProxyInfo() {
    Properties propperties = System.getProperties();                //获得系统属性对象
```

```
        propperties.remove("http.proxyHost");                    //清除 HTTP 服务使用的代理服务器地址
        propperties.remove("http.proxyPort");                    //清除 HTTP 服务使用的代理服务器端口
        propperties.remove("https.proxyHost");                   //清除安全 HTTP 服务使用的代理服务器地址
        propperties.remove("https.proxyPort");                   //清除安全 HTTP 服务使用的代理服务器端口
        propperties.remove("ftp.proxyHost");                     //清除 FTP 访问的代理服务器地址
        propperties.remove("ftp.proxyPort");                     //清除 FTP 访问的代理服务器端口
        propperties.remove("socksProxyHost");                    //清除 SOCKS 代理服务器的地址
        propperties.remove("socksProxyPort");                    //清除 SOCKS 代理服务器的端口
    }
```

（5）在窗体类 ProxySelectorFrame 中定义 useHttpAccess()方法，用于测试使用 HTTP 服务时所选择的代理服务器，该方法的代码如下：

```
    public void useHttpAccess(String accessAddress) throws Exception{
        URL url = new URL(accessAddress);                        //创建 URL 对象
        URLConnection urlConn = url.openConnection();            //自动调用系统设置的 HTTP 代理服务器，并打开连接
        Scanner scanner = new Scanner(urlConn.getInputStream(),"utf8");  //通过流创建扫描对象
        StringBuffer sb = new StringBuffer();                    //创建字符串缓存
        while (scanner.hasNextLine()) {                          //如果扫描器中有信息
            sb.append(scanner.nextLine()+"\n");                  //读取代理服务器上的网页内容，并添加到字符串缓存中
        }
        if (sb!=null) {
            ta_info.append(sb.toString());                       //显示网页内容
        }
    }
```

（6）在输入网址文本框的动作事件中，实现了用于设置代理、进行测试和移除代理的功能，其事件代码如下：

```
    try {
        String proxyAddress = tf_proxyAddress.getText().trim();  //代理服务器地址
        setProxyInfo(proxyAddress);                              //设置本地代理
        String accessAddress = tf_accessAddress.getText().trim();//获得需要访问的网址
        useHttpAccess(accessAddress);                            //调用方法，进行 HTTP 访问
        removeProxyInfo();                                       //清除本地代理
    } catch (Exception ex) {
    }
```

秘笈心法

心法领悟 519：选择代理服务器的功能。

当服务器有多种服务时，可以使用选择代理服务器来增加服务器的安全性，当客户端访问服务器时，会根据所使用的服务，通过相应的代理服务器与服务器连接，然后将信息通过选择的代理服务器反馈给客户端。

18.3 TCP 实用程序

实例 520 聊天室服务器端　　高级

光盘位置：光盘\MR\520　　趣味指数：★★★★

实例说明

本实例实现了聊天室服务器端的功能。运行程序，服务器端等待客户端的连接，并显示客户端的连接信息，如图 18.32 所示为有 3 个客户端连接到服务器，然后有一个客户端退出的效果。

关键技术

本实例使用 Hashtable 类来存储连接到服务器的用户名和套接字对象，并使用 String 类的 startWith()方法判断客户端发送信息的类型，从而实现了向服务器端添加登录用户、发送退出信息、通过服务器转发客户端发送的信息等功能，最终完成了聊天室服务器端程序的开发。

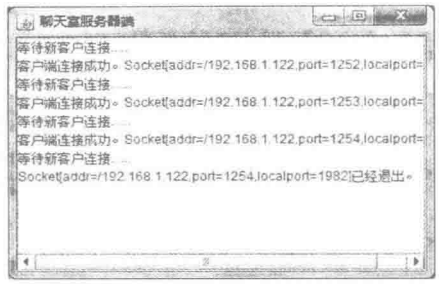


图 18.32　聊天室服务器端

（1）Hashtable 类实现了一个哈希表，用于存储键值映射关系，任何非 null 对象都可以用作键和值使用，在本实例中用到该类的构造方法定义如下：

public Hashtable()

参数说明

使用默认的初始容量 11 和加载因子 0.75 创建一个新的空哈希表。

（2）String 类的 startWith()方法用于判断当前字符串是否以参数指定的字符串为前缀，该方法的定义如下：

public boolean startsWith(String prefix)

参数说明

❶ prefix：指定的前缀字符串。

❷ 返回值：如果该方法以指定的前缀开始，则返回 true；否则返回 false。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的服务器窗体类 ChatServerFrame，然后在窗体上添加一个滚动面板，并在滚动面板上添加一个文本域控件，完成窗体界面的设计。

（3）在服务器窗体类 ChatServerFrame 的成员声明区定义一个 Hashtable 对象，用于存储登录用户的用户名和套接字对象，代码如下：

```
private Hashtable<String, Socket> map = new Hashtable<String, Socket>();          //用于存储连接到服务器的用户和客户端套接字对象
```

（4）在服务器窗体类 ChatServerFrame 中定义 createSocket()方法，用于创建服务器套接字对象、获得连接到服务器的客户端套接字对象以及启动线程对象对客户端发送的信息进行处理，该方法的代码如下：

```
public void createSocket() {
    try {
        server = new ServerSocket(1982);                                    //创建服务器套接字对象
        while (true) {
            ta_info.append("等待新客户连接.....\n");
            socket = server.accept();                                       //获得套接字对象
            ta_info.append("客户端连接成功。" + socket + "\n");
            new ServerThread(socket).start();                               //创建并启动线程对象
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（5）在服务器窗体类 ChatServerFrame 中定义内部线程类 ServerThread，用于对客户端的连接信息以及发送的信息进行处理和转发，该类的定义如下：

```
class ServerThread extends Thread {
    //省略了部分代码
    public void run() {
        try {
            ObjectInputStream ins = new ObjectInputStream(socket.getInputStream()); //获得输入流对象
            while (true) {
                //省略了部分代码
                if (v != null && v.size() > 0) {
```

```
                for (int i = 0; i < v.size(); i++) {
                    String info = (String) v.get(i);                          //读取信息
                    String key = "";
                    if (info.startsWith("用户：")) {                           //添加登录用户到客户端列表
                        //省略了部分代码
                    } else if (info.startsWith("退出：")) {
                        //省略了部分代码
                    } else {                                                  //转发接收的消息
                        key = info.substring(info.indexOf("：发送给：") + 5,
                                info.indexOf("：的信息是："));                 //获得接收方的 key 值，即接收方的用户名
                        String sendUser = info.substring(0, info
                                .indexOf("：发送给："));                       //获得发送方的 key 值，即发送方的用户名
                        Set<String> set = map.keySet();                       //获得集合中所有键的 Set 视图
                        Iterator<String> keyIt = set.iterator();              //获得所有键的迭代器
                        while (keyIt.hasNext()) {
                            String receiveKey = keyIt.next();                 //获得表示接收信息的键
                            if (key.equals(receiveKey) && !sendUser.equals(receiveKey)) { //与接收用户相同，但不是发送用户
                                Socket s = map.get(receiveKey);               //获得与该键对应的套接字对象
                                PrintWriter out = new PrintWriter(s.getOutputStream(), true); //创建输出流对象
                                out.println("MSG:" + info);                   //发送信息
                                out.flush();                                  //刷新输出缓冲区
                            }
                        }
                    }
                }
            }
        } catch (IOException e) {
            ta_info.append(socket + "已经退出。\n");
        }
    }
}
```

秘笈心法

心法领悟 520：使聊天室服务器端可以发送信息。

本实例实现的聊天室服务器端只是简单地实现监听客户端功能，并转发客户端之间发送的信息，为完善本实例，使其可以向客户端群发信息或选择客户端发送信息，实现该功能可以参考实例 513 和实例 514。

实例 521　聊天室客户端

光盘位置：光盘\MR\521　　高级　　趣味指数：★★★★

实例说明

本实例实现了聊天室客户端。运行程序，用户登录服务器后，可以从用户列表中选择单个用户进行聊天，也可以选择多个用户进行聊天，其中选择单个用户进行聊天的效果如图 18.33 和图 18.34 所示。

图 18.33　阳光**与开心**的聊天记录

图 18.34　开心**与阳光**的聊天记录

关键技术

通过线程对接收到的信息进行处理分为 3 种情况，第 1 种为接收到的是登录用户；第 2 种为接收到的是退出提示；第 3 种为接收到的是消息，实现这 3 种功能的关键代码如下：

```
String info = in.readLine().trim();                                    //读取信息
if (!info.startsWith("MSG:")) {                                        //接收到的不是消息
    if (info.startsWith("退出：")) {                                    //接收到的是退出消息
        model.removeElement(info.substring(3));                        //从用户列表中移除用户
    } else {                                                           //接收到的是登录用户
        boolean itemFlag = false;                                      //标记是否为列表框添加列表项，为 true 则不添加，为 false 则添加
        for (int i = 0; i < model.getSize(); i++) {                    //对用户列表进行遍历
            if (info.equals((String) model.getElementAt(i))) {         //如果用户列表中存在该用户名
                itemFlag = true;                                       //设置为 true，表示不添加到用户列表
                break;                                                 //结束 for 循环
            }
        }
        if (!itemFlag) {
                model.addElement(info);                                //将登录用户添加到用户列表
        }
    }
} else {                                                               //如果获得的是消息，则在文本域中显示接收到的消息
    DateFormat df = DateFormat.getDateInstance();                      //获得 DateFormat 实例
    String dateString = df.format(new Date());                         //格式化为日期
    df = DateFormat.getTimeInstance(DateFormat.MEDIUM);                //获得 DateFormat 实例
    String timeString = df.format(new Date());                         //格式化为时间
    String sendUser = info.substring(4,info.indexOf("：发送给："));     //获得发送信息的用户
    String receiveInfo = info.substring(info.indexOf("：的信息是：")+6); //获得接收到的信息
    ta_info.append("   "+sendUser + "      " +dateString+"   "+timeString+"\n  "+receiveInfo+"\n");        //在文本域中显示信息
    ta_info.setSelectionStart(ta_info.getText().length()-1);           //设置选择的起始位置
    ta_info.setSelectionEnd(ta_info.getText().length());               //设置选择的结束位置
    tf_send.requestFocus();                                            //使发送信息文本框获得焦点
}
```

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的客户端窗体类 ChatClientFrame，用于进行用户登录、发送聊天信息和显示聊天信息，在该类中完成窗体界面的设计。

（3）在客户端窗体类 ChatClientFrame 中定义 createClientSocket()方法，用于创建套接字对象、输出流对象以及启动线程对象对服务器转发的信息进行处理，该方法的代码如下：

```
public void createClientSocket() {
    try {
        Socket socket = new Socket("192.168.1.122", 1982);             //创建套接字对象
        out = new ObjectOutputStream(socket.getOutputStream());        //创建输出流对象
        new ClientThread(socket).start();                              //创建并启动线程对象
    } catch (UnknownHostException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（4）在客户端窗体类 ChatClientFrame 中定义内部线程类 ClientThread，用于对服务器端转发的信息进行处理，并显示在相应的控件中，该类的关键代码已在关键技术中给出。

（5）在客户端窗体类 ChatClientFrame 中为“登录”按钮添加实现用户登录功能的代码，“登录”按钮的事件代码如下：

```
if (loginFlag) {                                                       //已登录标记为 true
    JOptionPane.showMessageDialog(null, "在同一窗口只能登录一次。");
    return;
}
```

```
String userName = tf_newUser.getText().trim();                                    //获得登录用户名
Vector v = new Vector();                                                          //定义向量，用于存储登录用户
v.add("用户：" + userName);                                                       //添加登录用户
try {
    out.writeObject(v);                                                           //将用户向量发送到服务器
    out.flush();                                                                  //刷新输出缓冲区
} catch (IOException ex) {
    ex.printStackTrace();
}
tf_newUser.setEnabled(false);                                                     //禁用用户文本框
loginFlag = true;                                                                 //将已登录标记设置为 true
```

（6）在客户端窗体类 ChatClientFrame 中定义发送聊天信息的 send()方法，该方法的代码如下：

```
private void send() {
    if (!loginFlag) {
        JOptionPane.showMessageDialog(null, "请先登录。");
        return;                                                                   //如果用户没登录则返回
    }
    String sendUserName = tf_newUser.getText().trim();                            //获得登录用户名
    String info = tf_send.getText();                                              //获得输入的发送信息
    if (info.equals("")) {
        return;                                                                   //如果没输入信息则返回，即不发送
    }
    Vector<String> v = new Vector<String>();                                      //创建向量对象，用于存储发送的消息
    Object[] receiveUserNames = user_list.getSelectedValues();                    //获得选择的用户数组
    if (receiveUserNames.length <= 0) {
        return;                                                                   //如果没选择用户则返回
    }
    for (int i = 0; i < receiveUserNames.length; i++) {
        String msg = sendUserName + "：发送给：" + (String) receiveUserNames[i]
                + "：的信息是：  " + info;                                        //定义发送的信息
        v.add(msg);                                                               //将信息添加到向量
    }
    try {
        out.writeObject(v);                                                       //将向量写入输出流，完成信息的发送
        out.flush();                                                              //刷新输出缓冲区
    } catch (IOException e) {
        e.printStackTrace();
    }
    DateFormat df = DateFormat.getDateInstance();                                 //获得 DateFormat 实例
    String dateString = df.format(new Date());                                    //格式化为日期
    df = DateFormat.getTimeInstance(DateFormat.MEDIUM);                           //获得 DateFormat 实例
    String timeString = df.format(new Date());                                    //格式化为时间
    String sendUser = tf_newUser.getText().trim();                                //获得发送信息的用户
    String receiveInfo = tf_send.getText().trim();                                //显示发送的信息
    ta_info.append("   "+sendUser + "      " +dateString+"   "+timeString+"\n   "+receiveInfo+"\n");  //在文本域中显示信息
    tf_send.setText(null);                                                        //清空文本框
    ta_info.setSelectionStart(ta_info.getText().length()-1);                      //设置选择的起始位置
    ta_info.setSelectionEnd(ta_info.getText().length());                          //设置选择的结束位置
    tf_send.requestFocus();                                                       //使发送信息文本框获得焦点
}
```

秘笈心法

心法领悟 521：使用文件记录聊天信息。

在使用本实例时，如果客户端退出，再打开客户端重新连接服务器或清空接收信息文本域中的信息，就无法找回聊天记录了，为此可以将聊天记录追加到文件中，需要时可以通过文件找回聊天信息。

第19章

邮件收发

- 简单邮件
- 复杂邮件

19.1 简 单 邮 件

实例 522	配置邮件服务	初级
	光盘位置：光盘\MR\522	趣味指数：★★★★

实例说明

为了能够收发邮件，首先需要安装和配置邮件服务器。本实例使用的是 Merak 邮件服务器，该服务器是一个收费的商业软件，可以免费试用 30 天，该服务器配置后的效果如图 19.1 所示。

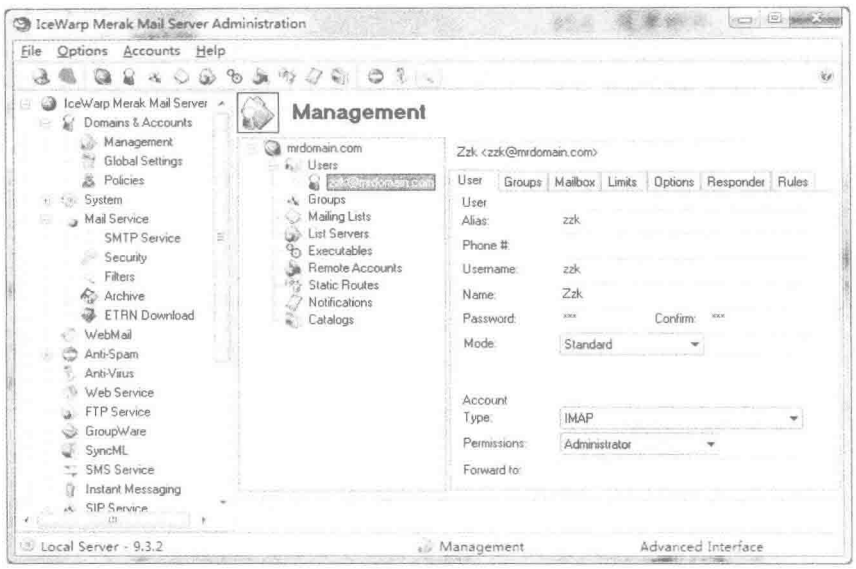


图 19.1 邮件服务器配置后的效果

关键技术

本实例要实现安装和配置邮件服务器，为此可以安装一个 Merak 邮件服务器，该服务器支持 STMP、POP3 和 IMAP 协议，目前的最新版本是 Mefak 9.3.2，用户可以从网上免费下载该服务器的试用版（可以免费试用 30 天）。从网上下载的是一个压缩包，解压后将看到一个名为 merak-9.3.2.exe 的程序，即为该服务器的安装程序，双击该程序即可安装邮件服务器。

说明：下载时，可以到百度网站搜索关键字“Merak 服务器”，然后找到相关的下载链接，然后根据提示完成下载。

设计过程

（1）双击安装程序 merak-9.3.2.exe，将打开如图 19.2 所示的对话框，选择语言后，单击 OK 按钮，将打开如图 19.3 所示的对话框，询问是否继续安装。

（2）单击如图 19.3 所示的对话框中的“是”按钮，将打开安装向导窗口，如图 19.4 所示。

（3）单击如图 19.4 所示的安装向导窗口中的 Next 按钮，将打开如图 19.5 所示的安装向导窗口。

（4）在图 19.5 所示的安装向导窗口中选中 I agree with the license agreement 复选框，接受许可协议，然后

单击 Yes 按钮，将打开如图 19.6 所示的安装向导窗口。

（5）在图 19.6 所示的安装向导窗口中输入用户的详细信息后，单击 Next 按钮，将打开如图 19.7 所示的安装向导窗口。

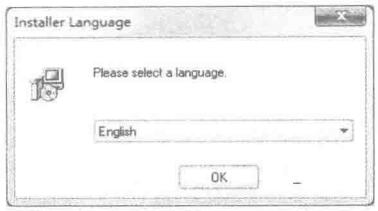


图 19.2　选择语言对话框

图 19.3　询问是否继续安装对话框

图 19.4　安装向导窗口 1

图 19.5　安装向导窗口 2

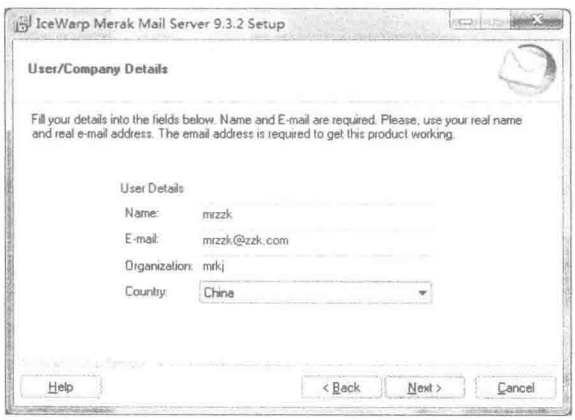


图 19.6　安装向导窗口 3

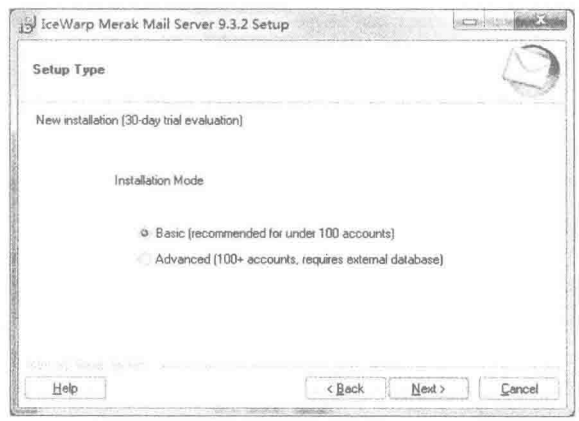


图 19.7　安装向导窗口 4

（6）选中图 19.7 所示的安装向导窗口中的 Basic 单选按钮，然后单击 Next 按钮，将打开如图 19.8 所示的安装向导窗口。

（7）在图 19.8 所示的安装向导窗口中采用默认设置，然后单击 Next 按钮，将打开如图 19.9 所示的安装向导窗口。

（8）在图 19.9 所示的安装向导窗口中指定安装路径，然后单击 Next 按钮，等待安装完成将出现如图 19.10 所示的安装向导窗口。

（9）在图 19.10 所示的安装向导窗口中采用默认设置，然后单击 Finish 按钮，完成邮件服务的安装，并打开如图 19.11 所示的 Wizard 对话框。

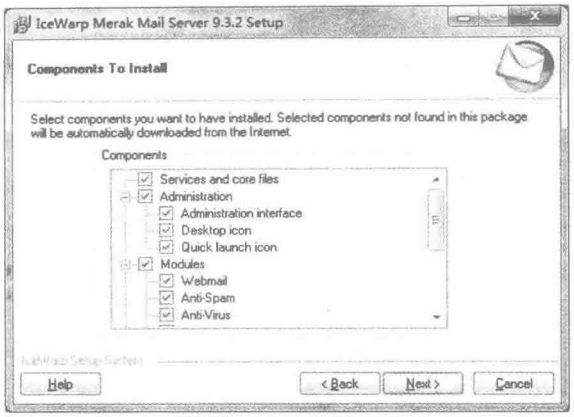


图 19.8　安装向导窗口 5

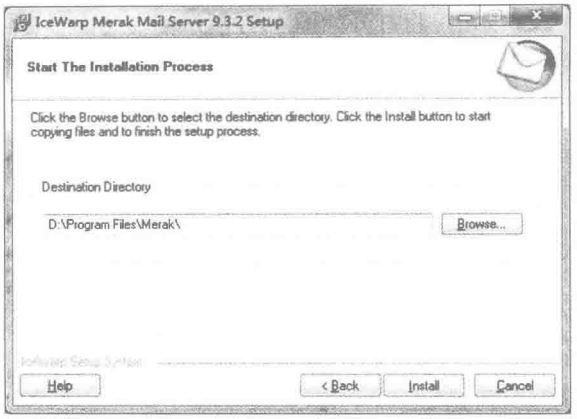


图 19.9　安装向导窗口 6

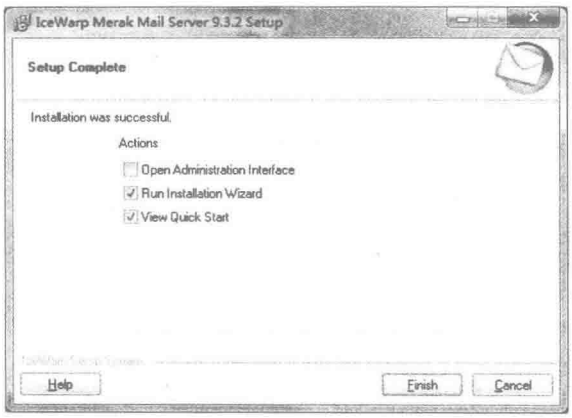


图 19.10　安装向导窗口 7

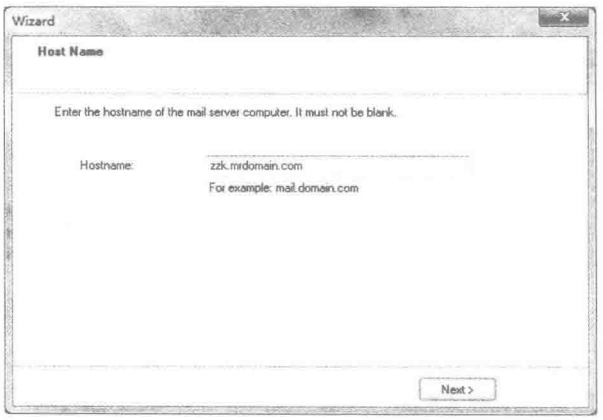


图 19.11　Wizard 对话框 1

（10）在图 19.11 所示的 Wizard 对话框中输入邮件服务的名称，这里输入 zzk.mrdomain.com，然后单击 Next 按钮，将打开如图 19.12 所示的 Wizard 对话框。

（11）在图 19.12 所示的 Wizard 对话框中输入域名，这里输入 mrdomain.com，然后单击 Next 按钮，将打开如图 19.13 所示的 Wizard 对话框。

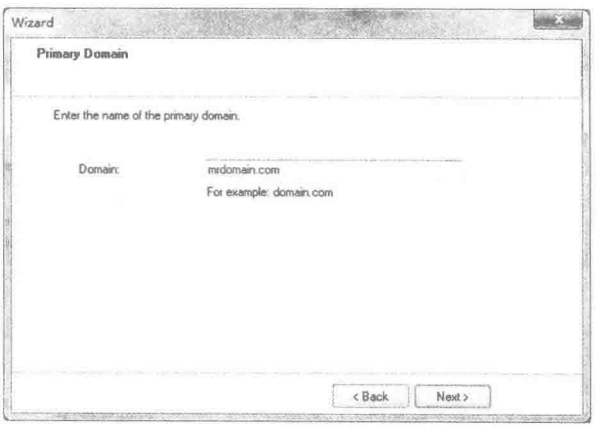


图 19.12　Wizard 对话框 2

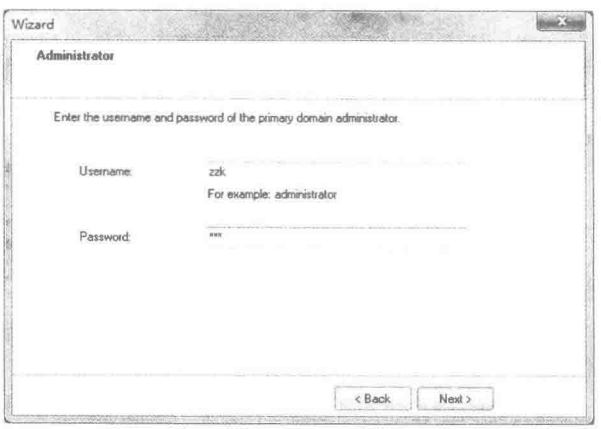


图 19.13　Wizard 对话框 3

（12）在图 19.13 所示的 Wizard 对话框中输入用户名 zzk 和密码 zzk，然后单击 Next 按钮，将打开如

图 19.14 所示的 Wizard 对话框。

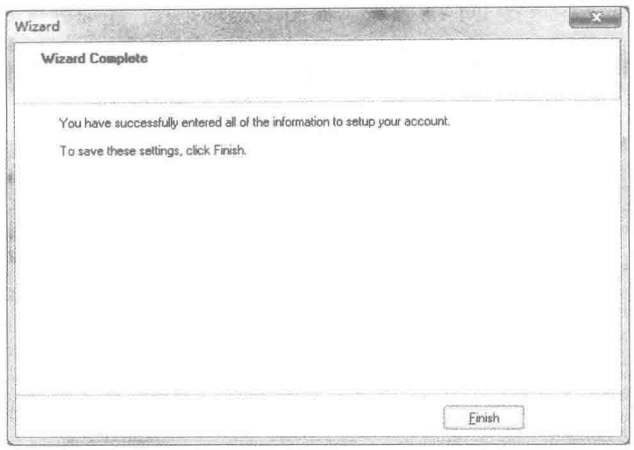


图 19.14　Wizard 对话框 4

（13）在图 19.14 所示的 Wizard 对话框中单击 Finish 按钮，完成邮件账户的设置。

（14）选择“开始”/“所有程序”/IceWarp Merak Mail Server/Administration 命令，打开邮件服务器窗口，如图 19.15 所示，同时弹出如图 19.16 所示的 Information 对话框，提示“可以试用 30 天，这是第一天的信息”。

图 19.15　邮件服务器窗口

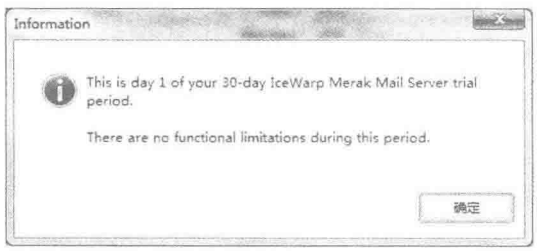


图 19.16　Information 对话框

（15）单击 Information 对话框中的“确定”按钮，返回如图 19.15 所示的窗口，展开 IceWarp Merak Mail Server/

Domains & Accounts/Management，在中间的列表框中选择 mrdomain.com/Users/zzk@mrdomain.com，然后在右侧的 Account 栏的 Type 下拉列表中选择 IMAP 服务，如图 19.17 所示，完成邮件服务的配置。

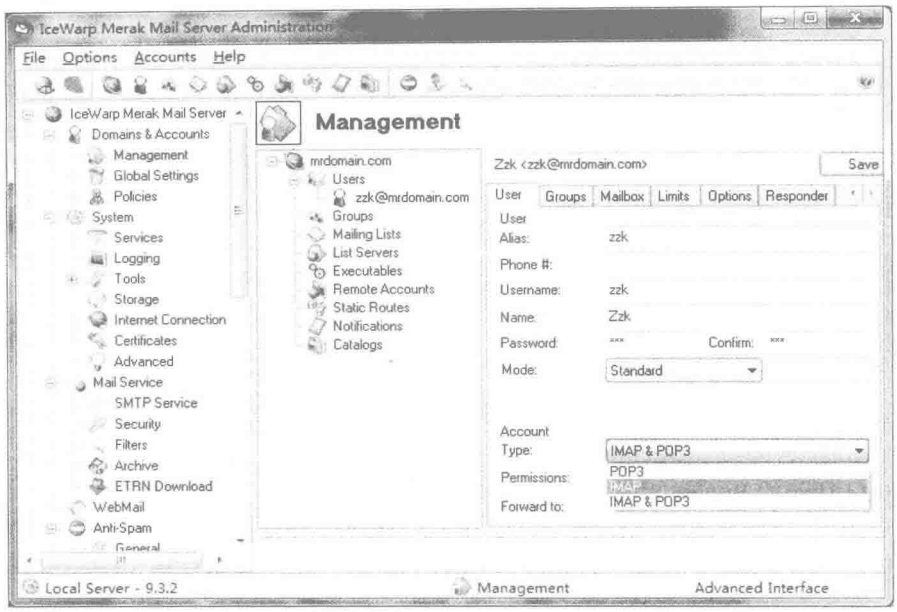


图 19.17　邮件服务器配置后的效果

秘笈心法

心法领悟 522：安装和配置邮件服务器的意义。

之所以要安装和配置邮件服务器，是为了以后能够收发邮件，如果没有安装和配置邮件服务器，将无法进行后继的邮件收发操作，所以安装和配置邮件服务器是进行邮件开发的关键。

实例 523　发送邮件

光盘位置：光盘\MR\523　　中级　　趣味指数：★★★★

实例说明

本实例实现了邮件的发送。运行程序，输入收件人地址、发件人地址、主题和正文，单击“发送”按钮，如果发送成功，将显示消息框进行提示，效果如图 19.18 和图 19.19 所示。

图 19.18　发送邮件窗体

图 19.19　提示邮件发送成功的消息框

关键技术

本实例的实现主要是通过 Message 类将接收方地址、发送方地址和要发送的信息封装起来，然后使用 Transport 类的 send()方法完成信息的发送。

（1）使用 Message 类将接收方地址、发送方地址和要发送的信息封装起来，实现代码如下：

```
Message msg = new MimeMessage(session);                          //创建 Message 对象
InternetAddress[] toAddrs = InternetAddress.parse(toAddr,false); //创建接收方的 InternetAddress 对象
msg.setRecipients(Message.RecipientType.TO, toAddrs);            //指定接收方
msg.setSentDate(new Date());                                     //指定发送日期
msg.setSubject(title);                                           //设置主题
msg.setFrom(new InternetAddress(fromAddr));                      //指定发送者
msg.setText(text);                                               //指定发送内容
```

（2）使用 Transport 类的 send()方法可以将 Message 对象封装的信息发送给接收方，实现代码如下：

```
Transport.send(msg);                                             //发送邮件
```

说明：上面代码中的 msg 是封装有接收方地址、发送方地址和要发送信息的 Message 对象。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 SendMailFrame。

（3）将 SendMailFrame 窗体的内容面板设置为绝对布局，然后放置相应的控件完成窗体界面的设计。

（4）在 SendMailFrame 窗体类中定义一个 init()方法，用于完成属性设置和 Session 对象的创建，该方法的代码如下：

```
public void init() throws Exception {
    Properties props = new Properties();                                  //创建属性对象
    props.put("mail.transport.protocol", sendProtocol);                   //指定邮件传输协议
    props.put("mail.smtp.class", "com.sun.mail.smtp.SMTPTransport");      //指定传输协议使用的类
    props.put("mail.smtp.host", sendHost);                                //定义发送邮件的主机
    session = Session.getDefaultInstance(props);                          //创建 Session 对象
}
```

（5）在 SendMailFrame 窗体类中定义一个 sendMessage()方法，用于完成邮件的发送，该方法的代码如下：

```
/**
 * @param fromAddr 发送者
 * @param toAddr 接收者
 * @param title 主题
 * @param text 内容
 * @throws Exception 异常
 */
public void sendMessage(String fromAddr,String toAddr,String title,String text) throws Exception {
    Message msg = new MimeMessage(session);                               //创建 Message 对象
    InternetAddress[] toAddrs = InternetAddress.parse(toAddr,false);      //创建接收方的 InternetAddress 对象
    msg.setRecipients(Message.RecipientType.TO, toAddrs);                 //指定接收方
    msg.setSentDate(new Date());                                          //指定发送日期
    msg.setSubject(title);                                                //设置主题
    msg.setFrom(new InternetAddress(fromAddr));                           //指定发送者
    msg.setText(text);                                                    //指定发送内容
    Transport.send(msg);                                                  //发送邮件
    JOptionPane.showMessageDialog(null, "邮件发送成功。");
}
```

秘笈心法

心法领悟 523：发送邮件必须指定的协议。

要实现邮件的发送，必须指定 SMTP 协议，方法是通过 Properties 创建一个属性对象，然后使用 Properties 对象的 put()方法指定使用的协议和发送邮件的主机等信息，最后将 Properties 对象作为获得 Session 对象的参数，

从而实现指定 SMTP 协议的功能。

实例 524　接收邮件

光盘位置：光盘\MR\524　　高级　　趣味指数：★★★

实例说明

本实例实现了简单邮件的接收，即只能接收文本信息的邮件。运行程序，单击窗体上的“接收邮件”按钮，可以接收到已经发送的邮件，效果如图 19.20 所示。

图 19.20　接收邮件窗体

关键技术

本实例主要是通过为 Store 对象指定接收邮件协议和连接接收服务器，然后通过 Store 对象的 getFolder()方法获得 inbox 邮件夹的 Folder 对象，进而获得接收到的邮件信息。

（1）使用 Store 类指定接收邮件协议和连接服务器，可以通过如下代码实现：

```
Properties props = new Properties();                          //声明 Properties 对象
props.put("mail.store.protocol", receiveProtocol);            //指定接收协议
props.put("mail.imap.class", "com.sun.mail.imap.IMAPStore");  //指定使用 Store 进行接收
session = Session.getDefaultInstance(props);                  //获得 Session 对象
store = session.getStore(receiveProtocol);                    //获得 Store 对象
store.connect(receiveHost,username,password);                 //连接接收服务器
```

（2）使用 Store 对象的 getFolder()方法获得 inbox 邮件夹的 Folder 对象，可以通过如下代码实现：

```
Folder folder = store.getFolder("inbox");                     //获得 inbox 邮件夹的 Folder 对象
```

说明：inbox 邮件夹是邮件服务器默认提供的邮件夹，用于存储接收到的邮件，因此可以通过该邮件夹获得接收到的邮件信息。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 ReceiveMailFrame。

（3）在 ReceiveMailFrame 窗体的内容面板中部添加一个滚动面板，再在滚动面板上添加一个文本域控件，然后在窗体的底部添加一个面板，并在面板上添加两个命令按钮，完成窗体界面的设计。

（4）在 ReceiveMailFrame 窗体类中定义一个 init()方法，用于完成属性设置和 Store 对象的创建，该方法的代码如下：

```
public void init() throws Exception {
    Properties props = new Properties();                      //声明 Properties 对象
```

```
        props.put("mail.store.protocol", receiveProtocol);                    //指定接收协议
        props.put("mail.imap.class", "com.sun.mail.imap.IMAPStore");          //指定使用 Store 进行接收
        session = Session.getDefaultInstance(props);                          //获得 Session 对象
        store = session.getStore(receiveProtocol);                            //获得 Store 对象
        store.connect(receiveHost,username,password);                         //连接接收服务器
    }
```

（5）在 ReceiveMailFrame 窗体类中定义一个 receiveMessage()方法，用于完成邮件的接收，该方法的代码如下：

```
    public void receiveMessage() throws Exception {
        Folder folder = store.getFolder("inbox");                             //获得 inbox 邮件夹的 Folder 对象
        if (folder == null) {
            throw new Exception("不存在 inbox 邮件夹。");
        }
        folder.open(Folder.READ_ONLY);                                        //以只读方式打开邮件夹
        ta_receive.append("您共收到"+folder.getMessageCount()+"个电子邮件。\n\n");
        Message[] messages = folder.getMessages();                            //获得邮件夹中的所有邮件
        for (int i = 0;i<messages.length;i++){
            ta_receive.append("----第"+(i+1)+"个邮件----\n");
            ta_receive.append("主题："+folder.getMessage(i+1).getSubject()+"\n");      //主题
            ta_receive.append("正文："+folder.getMessage(i+1).getContent()+"\n");      //正文
            ta_receive.append("发送日期："+folder.getMessage(i+1).getSentDate()+"\n"); //发送日期
            Address[] ias = folder.getMessage(i+1).getFrom();                 //发件人地址
            ta_receive.append("发件人："+ias[0]+"\n");
            Address[] iasTo = folder.getMessage(i+1).getAllRecipients();      //收件人地址
            ta_receive.append("收件人："+iasTo[0]+"\n\n");
        }
        folder.close(false);                                                  //关闭邮件夹，但不删除邮件
        store.close();                                                        //关闭 Store 对象
    }
```

秘笈心法

心法领悟 524：接收邮件的关键。

在进行邮件接收时，关键是指定接收协议，可以使用 POP3 协议和 IMAP 协议进行邮件接收。指定了邮件接收协议后，即可通过 Store 对象的 getFolder()方法获得 inbox 邮件夹的 Folder 对象，该对象提供了一系列访问邮件夹的方法，可以获得所有已接收邮件的 Message 数组，并从中读取邮件信息。

19.2　复 杂 邮 件

实例 525　发送带附件的邮件　　高级

光盘位置：光盘\MR\525　　趣味指数：★★★★★

实例说明

在发送邮件时，除了发送简单的文本信息外，还可以发送文件，即常说的附件，本实例实现了带附件的邮件的发送。运行程序，输入收件人地址、发件人地址、主题和正文后，如果需要发送附件，可以单击“添加附件”按钮，然后从打开的对话框中选择需要发送的附件，将其添加到附件列表中，然后单击“发送”按钮，如果邮件发送成功将显示消息框进行提示，效果如图 19.21 和图 19.22 所示。

关键技术

本实例主要是通过向 Multipart 对象添加多个 MimeBodyPart 对象（每个 MimeBodyPart 对象代表一部分内容，可以是文本内容，也可以是附件），再将 Multipart 对象作为 Message 对象的 setContent()方法的参数，从而实现

了带附件邮件的发送。

图 19.21　发送带附件的邮件

图 19.22　提示附件发送成功

发送带文本和附件的邮件，可以通过如下代码实现：

```
Multipart multipart = new MimeMultipart();                                    //可以添加复杂内容的 Multipart 对象
MimeBodyPart mimeBodyPartText = new MimeBodyPart();                           //添加正文的 MimeBodyPart 对象
mimeBodyPartText.setText(text);                                               //指定正文
multipart.addBodyPart(mimeBodyPartText);                                      //添加到 Multipart 对象上
if (filePathAndName!=null && !filePathAndName.equals("")){
    MimeBodyPart mimeBodyPartAdjunct = new MimeBodyPart();                    //添加附件的 MimeBodyPart 对象
    FileDataSource fileDataSource = new FileDataSource(filePathAndName);     //创建附件的 FileDataSource 对象
    mimeBodyPartAdjunct.setDataHandler(new DataHandler(fileDataSource));      //指定数据
    mimeBodyPartAdjunct.setDisposition(Part.ATTACHMENT);                      //指定添加的内容是附件
    String name = fileDataSource.getName();
    mimeBodyPartAdjunct.setFileName(MimeUtility.encodeText(name, "GBK", null));  //指定附件文件的名称
    multipart.addBodyPart(mimeBodyPartAdjunct);                               //添加到 Multipart 对象上
}
msg.setContent(multipart);                                                    //设置邮件内容
Transport.send(msg);                                                          //发送邮件
```

说明：上面代码中的 msg 是有效的 Message 对象，该对象封装了接收者、发送者、发送日期以及发送协议等信息。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 SendAttachmentMailFrame。

（3）将 SendAttachmentMailFrame 窗体的内容面板设置为绝对布局，然后在内容面板的合适位置放置标签、文本框、文本域和命令按钮等控件，完成窗体界面的设置。

（4）在 SendAttachmentMailFrame 窗体类中定义 sendMessage()方法用于完成带附件邮件的发送，该方法的代码如下：

```
/**
 * @param fromAddr 发送方地址
 * @param toAddr 接收方地址
 * @param title 主题
 * @param text 文本内容
 * @throws Exception 异常
 */
public void sendMessage(String fromAddr,String toAddr,String title,String text) throws Exception {
    Message msg = new MimeMessage(session);                                   //创建 Message 对象
    InternetAddress[] toAddrs = InternetAddress.parse(toAddr,false);          //接收方地址
    msg.setRecipients(Message.RecipientType.TO, toAddrs);                     //指定接收方
    msg.setSentDate(new Date());                                              //设置发送日期
```

```
        msg.setSubject(title);                                          //设置主题
        msg.setFrom(new InternetAddress(fromAddr));                     //设置发送地址
        //这里省略了关键技术中给出的代码
        filePathAndName = null;
        JOptionPane.showMessageDialog(null, "邮件发送成功。");
    }
```

秘笈心法

心法领悟 525：实现多个附件的发送。

本实例只能发送一个附件，为此可以对该程序进行扩展，使其能够发送多个附件，方法是对每个附件都创建一个 MimeBodyPart 对象，并添加到 Multipart 对象上。

实例 526　接收带附件的邮件

光盘位置：光盘\MR\526　　高级　　趣味指数：★★★★★

实例说明

接收邮件时，除了可以接收只有文本信息的邮件，还可以接收带有附件的邮件，本实例实现了带附件邮件的接收。运行程序，单击窗体上的“接收邮件并下载附件”按钮，将在文本域中显示所有已接收邮件的内容，如果有附件还会显示“保存”对话框实现附件文件的保存，效果如图 19.23 和图 19.24 所示。

图 19.23　接收邮件窗体

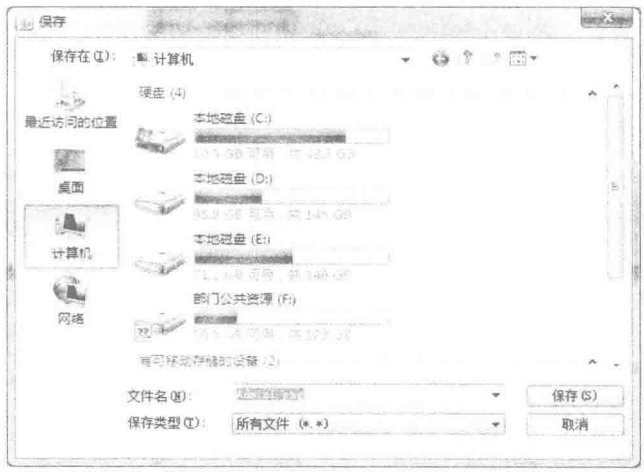


图 19.24　用于保存附件的对话框

关键技术

本实例的实现主要是通过对接收内容进行分类处理，文本内容直接显示在文本域中；对于有附件的内容，除了在文本域中显示文本，还显示附件的文件名称，同时显示“保存”对话框保存接收到的附件文件。

对接收到的内容进行分类处理，可以通过如下代码实现：

```
Message[] messages = folder.getMessages();                  //获得邮件夹中的所有邮件
for (int i = 0; i < messages.length; i++) {                 //遍历所有邮件
    Object content = messages[i].getContent();              //获得邮件内容
    if (content instanceof Multipart) {                     //邮件内容是 Multipart 的实例，说明有附件，否则说明没有附件
        //有附件执行的代码
    } else {
        //没有附件执行的代码
    }
}
```

说明：上面代码中的 folder 是收件夹 inbox 的 Folder 对象。上述代码省略了读取邮件内容的代码。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的窗体类 ReceiveMailFrame。

（3）在 ReceiveMailFrame 窗体的内容面板中部添加一个滚动面板，再在滚动面板上添加一个文本域控件，然后在窗体的底部添加一个面板，并在面板上添加两个命令按钮，完成窗体界面的设计。

（4）在 ReceiveMailFrame 窗体类中定义一个 receiveMessage()方法，实现对有附件邮件和无附件邮件的接收，该方法的代码如下：

```
public void receiveMessage() throws Exception {
    Folder folder = store.getFolder("inbox");                                    //获得 inbox 邮件夹的 Folder 对象
    if (folder == null) {
        throw new Exception("不存在 inbox 邮件夹。");
    }
    folder.open(Folder.READ_ONLY);                                               //以只读方式打开邮件夹
    ta_receive.append("您共收到" + folder.getMessageCount() + "个电子邮件。\n\n");
    Message[] messages = folder.getMessages();                                   //获得邮件夹中的所有邮件
    for (int i = 0; i < messages.length; i++) {
        Object content = messages[i].getContent();                               //获得邮件内容
        if (content instanceof Multipart) {                                      //有附件执行的代码
            ta_receive.append("----第" + (i + 1) + "个邮件----\n");
            ta_receive.append("主题：" + folder.getMessage(i + 1).getSubject() + "\n");   //主题
            Multipart mPart = (Multipart) content;                               //转换为 Multipart 对象
            ta_receive.append("正文：" + mPart.getBodyPart(0).getContent() + "\n");       //正文内容
            ta_receive.append("发送日期：" + folder.getMessage(i + 1).getSentDate() + "\n");  //发送日期
            Address[] ias = folder.getMessage(i + 1).getFrom();                  //发件人地址
            ta_receive.append("发件人：" + ias[0] + "\n");
            Address[] iasTo = folder.getMessage(i + 1).getAllRecipients();       //收件人地址
            ta_receive.append("收件人：" + iasTo[0] + "\n\n");
            try {
                String fileName = mPart.getBodyPart(1).getFileName();            //获得文件名
                ta_receive.append("接收到一个名为“" + MimeUtility.decodeText(fileName) + "”的附件\n");
                InputStream in = mPart.getBodyPart(1).getInputStream();
                FileDialog dialog = new FileDialog(ReceiveMailFrame.this, "保存");   //创建对话框
                dialog.setMode(FileDialog.SAVE);                                 //设置对话框为“保存”对话框
                dialog.setFile(MimeUtility.decodeText(fileName));
                dialog.setVisible(true);                                         //显示“保存”对话框
                String path = dialog.getDirectory();                             //获得文件的保存路径
                String saveFileName = dialog.getFile();                          //获得保存的文件名
                if (path == null || saveFileName == null) {
                    return;
                }
                OutputStream out = new BufferedOutputStream(new FileOutputStream(path + "/" + saveFileName));   //保存附件文件的输出流
                int len = -1;
                while ((len = in.read()) != -1) {
                    out.write(len);                                              //向输出流写入数据
                }
                out.close();                                                     //关闭输出流对象
                in.close();                                                      //关闭输入流对象
            } catch (Exception ex) {
            }
        } else {            //没有附件执行的代码
            ta_receive.append("----第" + (i + 1) + "个邮件----\n");
            ta_receive.append("主题：" + folder.getMessage(i + 1).getSubject() + "\n");   //主题
            ta_receive.append("正文：" + folder.getMessage(i + 1).getContent() + "\n");   //正文
            ta_receive.append("发送日期：" + folder.getMessage(i + 1).getSentDate() + "\n");  //发送日期
            Address[] ias = folder.getMessage(i + 1).getFrom();                  //发件人地址
            ta_receive.append("发件人：" + ias[0] + "\n");
            Address[] iasTo = folder.getMessage(i + 1).getAllRecipients();       //收件人地址
```

```
                ta_receive.append("收件人：" + iasTo[0] + "\n\n");
            }
        }
        folder.close(false);                                    //关闭邮件夹，但不删除邮件
        store.close();                                          //关闭 Store 对象
    }
```

秘笈心法

心法领悟 526：选择接收需要的附件文件。

本实例在接收邮件时，只要邮件有附件，就会显示“保存”对话框提示用户保存附件文件，而在实际使用邮件时，并不需要一次性地将所有邮件的附件都保存起来，为此可以将该实例改为根据需要选择下载附件，实现这一功能可以将所有附件都添加到一个列表框中，这样用户就可以从列表框中选择需要的附件，然后通过命令按钮的事件确认下载选择的附件。

实例 527　发送邮件时进行身份验证

实例位置：光盘\MR\527　　高级　　趣味指数：★★★★

实例说明

本实例演示如何在发送邮件时，对发送者的身份进行验证。运行程序，输入收件人地址、发件人地址、主题和正文后，如果需要发送附件，可以单击“添加附件”按钮，并从打开的对话框中选择需要发送的附件，将其添加到附件列表中，然后单击“发送”按钮，将弹出“身份验证对话框”，要求输入用户名和密码，效果如图 19.25 和图 19.26 所示。

图 19.25　发送邮件窗体

图 19.26　对发送者进行身份验证

关键技术

本实例将 SMTP 邮件服务器的 mail.smtp.auth 属性设置为 true，这样当用户使用 SMTP 邮件服务器发送邮件时，就需要对其身份进行验证，然后创建一个 Authenticator 类的子类 CheckAuthenticator，在该子类中定义一个对话框类和一个返回值类型是 PasswordAuthentication 的 getPasswordAuthentication()方法，再创建一个 CheckAuthenticator 对象，并将该对象传递给 Session 对象的 getDefaultInstance()方法，这样在使用 Transport 类的 send()方法发送信息时，就会显示进行身份验证的对话框。

设置 SMTP 邮件服务器的 mail.smtp.auth 属性、将 Authenticator 类的子类 CheckAuthenticator 的对象传递给 Session 对象的 getDefaultInstance()方法，可以通过如下代码实现：

```
Properties props = new Properties();                            //创建属性对象
```

```
props.put("mail.smtp.auth", "true");                    //设置发邮件时需要身份验证
Authenticator check = new CheckAuthenticator();         //创建 Authenticator 实例
session = Session.getDefaultInstance(props,check);      //创建 Session 对象
```

说明：通过上面代码的设置，在使用 Transport 类的 send()方法发送信息时，会弹出身份验证对话框，如果用户输入的用户名和密码正确，就能够成功发送邮件，否则将无法成功发送邮件。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 Authenticator 类的 CheckAuthenticator 类，该类用于弹出身份验证对话框，对发送邮件的用户进行身份验证，该类的关键代码如下：

```
public class CheckAuthenticator extends Authenticator {
    class UserPassDialog extends JDialog {                  //用于显示对话框的内部类
        private JPasswordField pf_pwd;                      //密码文本框
        private JTextField tf_user;                         //用户名文本框
        public UserPassDialog() {
            super();
            //省略了初始化对话框的代码
        }
    }
    public PasswordAuthentication getPasswordAuthentication() {
        UserPassDialog dialog = new UserPassDialog();       //创建对话框
        dialog.setModal(true);                              //设置为模式对话框
        dialog.setVisible(true);                            //显示对话框
        String user = dialog.tf_user.getText().trim();      //获得用户名
        String pwd = new String(dialog.pf_pwd.getPassword()); //获得密码
        dialog.tf_user.setText(null);                       //清空用户名文本框
        dialog.pf_pwd.setText(null);                        //清空密码文本框
        return new PasswordAuthentication(user, pwd);       //返回 PasswordAuthentication 对象
    }
}
```

（3）在项目中创建一个继承 JFrame 类的 SendAttachmentMailFrame 窗体类，将该窗体的内容面板设置为绝对布局，然后放置相应的控件完成窗体界面的设计。

（4）在 SendAttachmentMailFrame 窗体类中定义一个 init()方法，用于设置 SMTP 邮件服务器的 mail.smtp.auth 属性、将 Authenticator 类的子类 CheckAuthenticator 的对象传递给 Session 对象的 getDefaultInstance()方法，这样当用户发送邮件时就需要进行身份验证，该方法的代码如下：

```
public void init() throws Exception {
    Properties props = new Properties();                    //创建属性对象
    props.put("mail.transport.protocol", sendProtocol);     //指定邮件传输协议
    props.put("mail.smtp.class", "com.sun.mail.smtp.SMTPTransport"); //指定传输协议使用的类
    props.put("mail.smtp.host", sendHost);                  //定义发送邮件的主机
    props.put("mail.smtp.auth", "true");                    //设置发邮件时需要身份验证
    Authenticator check = new CheckAuthenticator();         //创建 Authenticator 实例
    session = Session.getDefaultInstance(props,check);      //创建 Session 对象
}
```

（5）在 SendAttachmentMailFrame 窗体类中定义一个 sendMessage()方法，用于完成邮件的发送，并且在发送邮件时会显示身份验证对话框，该方法的代码与实例 525 中的 sendMessage()方法相同，这里不再重复给出，如果需要请查看源程序代码。

秘笈心法

心法领悟 527：为什么要对发送者的身份进行验证。

对发送者的身份进行验证，是为了防止其他人非法使用用户的邮箱发送邮件，例如邮箱打开时，若用户需要离开一小段时间，其他用户要想用该邮箱发送邮件，就必须输入密码，否则不能发送邮件。

实例 528　接收邮件时进行身份验证

光盘位置：光盘\MR\528　　高级　　趣味指数：★★★★★

实例说明

本实例演示了如何在接收邮件时，对用户的身份进行验证。运行程序，将弹出“身份验证对话框”，要求用户输入用户名和密码，输入正确，单击对话框中的“确定”按钮，将显示接收邮件窗口，单击该窗口中的“接收邮件并下载附件”按钮，将显示接收到的邮件信息，效果如图 19.27 和图 19.28 所示。

图 19.27　对收件者进行身份验证的对话框

图 19.28　接收邮件窗体

关键技术

由于在使用 IMAP 服务器接收邮件时，需要使用 Store 对象的 connect()方法，而该方法需要指定收件者的用户名和密码，如果将该方法的用户名和密码设置为 null，并创建一个 Authenticator 类的子类 CheckAuthenticator，在该子类中定义一个对话框类和一个返回值类型是 PasswordAuthentication 的 getPasswordAuthentication()方法，然后创建一个 CheckAuthenticator 对象，并将该对象传递给 Session 对象的 getDefaultInstance()方法，这样在使用 Store 对象的 connect()方法连接邮件服务器时，就会显示进行身份验证的对话框。

对接收邮件者进行身份验证，可以通过如下代码实现：

```
Properties props = new Properties();                    //声明 Properties 对象
Authenticator check = new CheckAuthenticator();         //创建 Authenticator 实例
session = Session.getDefaultInstance(props,check);      //获得 Session 对象
store = session.getStore(receiveProtocol);              //获得 Store 对象
store.connect(receiveHost, null, null);                 //连接接收服务器，并进行身份验证
```

说明：通过上面代码的设置，在使用 Store 对象的 connect()方法连接邮件服务器时，就会显示进行身份验证的对话框，如果输入的用户名和密码正确，就可以接收邮件，否则将无法接收邮件。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 Authenticator 类的 CheckAuthenticator 类，该类用于弹出身份验证对话框，对接收邮件的用户进行身份验证，该类的实现代码与实例 527 中的代码相同，这里不再重复给出，如果需要请查看源程序代码。

（3）在项目中创建一个继承 JFrame 类的 ReceiveMailFrame 窗体类，在该窗体内容面板的中部添加一个滚动面板，再在滚动面板上添加一个文本域控件，然后在窗体的底部添加一个面板，并在面板上添加两个命令按钮，完成窗体界面的设计。

（4）在 ReceiveMailFrame 窗体类中定义一个 init()方法，用于设置接收邮件使用的协议，以及将 Authenticator 类的子类 CheckAuthenticator 的对象传递给 Session 对象的 getDefaultInstance()方法，获得 Store 对象并连接到接收服务器，这样当用户接收邮件时就需要进行身份验证，该方法的代码如下：

```
public void init() throws Exception {
    Properties props = new Properties();                          //声明 Properties 对象
    props.put("mail.store.protocol", receiveProtocol);            //指定接收协议
    props.put("mail.imap.class", "com.sun.mail.imap.IMAPStore");  //指定使用 Store 进行接收
    Authenticator check = new CheckAuthenticator();               //创建 Authenticator 实例
    session = Session.getDefaultInstance(props,check);            //获得 Session 对象
    store = session.getStore(receiveProtocol);                    //获得 Store 对象
    store.connect(receiveHost, null, null);                       //连接接收服务器，并进行身份验证
}
```

说明：上面代码执行后，将显示对收件者进行身份验证的对话框，用户只有输入正确的用户名和密码，才可以查看接收到的邮件，否则将无法连接到邮件接收服务器。

（5）在 ReceiveMailFrame 窗体类中定义一个 receiveMessage()方法，用于完成邮件的接收，该方法的代码与实例 526 中的 receiveMessage()方法相同，这里不再重复给出，如果需要请查看源程序代码。

秘笈心法

心法领悟 528：验证接收者的身份意义重大。

很多用户会通过邮件传递信息，如公司的文件、个人信息等，其中有些内容可能是公司机密或是个人隐私，如果被他人窃取会造成严重后果，因此可以通过对接收者的身份进行验证，来有效地保护个人隐私和公司机密等重要信息。

实例 529　显示未读邮件

光盘\MR\529　　中级　　趣味指数：★★★★

实例说明

本实例实现了显示未读邮件的功能。随着邮件数量的不断增加，查看邮件会比较耗时费力，为此可以直接将未读邮件读取出来，以减少查看邮件的时间。运行程序，单击窗体上的“接收邮件并下载附件”按钮，可以将未读邮件的内容显示在文本域中，效果如图 19.29 所示。

图 19.29　显示未读邮件的效果

关键技术

本实例主要是通过获得邮件的 UID 来实现显示未读邮件的功能，方法是将已读邮件的 UID 保存到文件中，

下次读取时，判断读取的 UID 是否在文件中，如果不存在，则表示是未读邮件，并读取邮件的内容。

获得邮件的 UID 可以通过 IMAPFolder 类的 getUID()方法实现，该方法需要一个 Message 类型的参数，获得邮件 UID 的代码如下：

```
Message[] messages = folder.getMessages();                    //获得邮件夹中的所有邮件
for (int i = 0; i < messages.length; i++) {
    long uid = folder.getUID(messages[i]);                    //获得邮件的 UID
}
```

说明：上面代码中的 folder 是有效的 IMAPFolder 对象的实例；messages 是邮件夹中邮件的 Message 数组。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ReceiveMailFrame 窗体类。

（3）在 ReceiveMailFrame 窗体的内容面板中部添加一个滚动面板，再在滚动面板上添加一个文本域控件，然后在窗体的底部添加一个面板，并在面板上添加两个命令按钮，完成窗体界面的设计。

（4）在 ReceiveMailFrame 窗体类中定义一个 receiveMessage()方法，实现通过邮件的 UID 判断未读邮件并读取未读邮件内容的功能，该方法的代码如下：

```
public void receiveMessage() throws Exception {
    IMAPFolder folder = (IMAPFolder) store.getFolder("inbox");          //获得 inbox 邮件夹的 IMAPFolder 对象
    if (folder == null) {
        throw new Exception("不存在 inbox 邮件夹。");
    }
    folder.open(Folder.READ_ONLY);                                      //以只读方式打开邮件夹
    Message[] messages = folder.getMessages();                          //获得邮件夹中的所有邮件
    for (int i = 0; i < messages.length; i++) {
        long uid = folder.getUID(messages[i]);                          //获得邮件的 UID
        try {
            objectIn = new ObjectInputStream(new FileInputStream("c:/readedEmail.txt"));        //创建输入流对象
            Object readObj = objectIn.readObject();                     //读取信息
            if (readObj == null) {                                      //文件中未存储已读文件的 UID，说明是未读邮件
                readedEmailUIDVector.add(String.valueOf(uid));          //将读取邮件的 UID 添加到向量中
                objectOut = new ObjectOutputStream(new FileOutputStream("c:/readedEmail.txt"));  //创建输出流对象
                objectOut.writeObject(readedEmailUIDVector);            //将已读邮件 UID 的向量写入磁盘
                //读取邮件信息
                noReadNums++;
                readMessage(messages[i], folder, i);                    //调用 readMessage()方法，读取邮件信息
            } else {
                boolean readedFlag = false;                             //标记为 false，表示未读
                readedEmailUIDVector = (Vector<String>) readObj;
                for (int j = 0; j < readedEmailUIDVector.size(); j++) {
                    if (String.valueOf(uid).equals(readedEmailUIDVector.get(j))) {
                        readedFlag = true;                              //标记为 true，表示已读
                        break;
                    }
                }
                if (readedFlag) {
                    continue;
                } else {
                    readedEmailUIDVector.add(String.valueOf(uid));      //将读取邮件的 UID 添加到向量中
                    objectOut = new ObjectOutputStream(new FileOutputStream("c:/readedEmail.txt"));  //创建输出流对象
                    objectOut.writeObject(readedEmailUIDVector);        //将已读邮件 UID 的向量写入磁盘
                    //读取邮件信息
                    noReadNums++;
                    readMessage(messages[i], folder, i);                //调用 readMessage()方法，读取邮件信息
                }
            }
        } catch (Exception ex) {
            readedEmailUIDVector.add(String.valueOf(uid));              //将读取邮件的 UID 添加到向量中
            objectOut = new ObjectOutputStream(new FileOutputStream("c:/readedEmail.txt"));  //创建输出流对象
```

```
                objectOut.writeObject(readedEmailUIDVector);                    //将已读邮件 UID 的向量写入磁盘
                //读取邮件信息
                noReadNums++;
                readMessage(messages[i], folder, i);                            //调用 readMessage()方法，读取邮件信息
            }
        }
        ta_receive.append("您共收到" + folder.getMessageCount() + "个邮件。\n");
        if (noReadNums > 0) {
            ta_receive.append("其中上述" + noReadNums + "个是新邮件。\n\n");
        } else {
            ta_receive.append("没有未读邮件。\n\n");
        }
        folder.close(false);                                                    //关闭邮件夹，但不删除邮件
        store.close();                                                          //关闭 Store 对象
        if (objectOut != null)
            objectOut.close();
        if (objectIn != null)
            objectIn.close();
    }
```

（5）在 ReceiveMailFrame 窗体类中定义 readMessage()方法，实现读取邮件内容的功能，该方法的代码如下：

```
public void readMessage(Message message, IMAPFolder folder, int i) throws Exception {
    Object content = message.getContent();                                      //获得邮件内容
    if (content instanceof Multipart) {                                         //有附件执行的代码
        ta_receive.append("----第" + (i + 1) + "个邮件----\n");
        ta_receive.append("主题：" + folder.getMessage(i + 1).getSubject() + "\n");   //主题
        Multipart mPart = (Multipart) content;                                  //创建 Multipart 对象
        ta_receive.append("正文：" + mPart.getBodyPart(0).getContent() + "\n");   //正文
        ta_receive.append("发送日期：" + folder.getMessage(i + 1).getSentDate() + "\n");   //发送日期
        Address[] ias = folder.getMessage(i + 1).getFrom();                     //发件人地址
        ta_receive.append("发件人：" + ias[0] + "\n");
        Address[] iasTo = folder.getMessage(i + 1).getAllRecipients();          //收件人地址
        ta_receive.append("收件人：" + iasTo[0] + "\n\n");
        try {
            String fileName = mPart.getBodyPart(1).getFileName();               //获得文件名
            ta_receive.append("接收到一个名为“" + MimeUtility.decodeText(fileName) + "”的附件\n\n");   //获得附件文件名
            InputStream in = mPart.getBodyPart(1).getInputStream();             //获得输入流对象
            FileDialog dialog = new FileDialog(ReceiveMailFrame.this, "保存");  //创建对话框
            dialog.setMode(FileDialog.SAVE);                                    //设置对话框为保存对话框
            dialog.setFile(MimeUtility.decodeText(fileName));                   //设置对话框显示的文件名
            dialog.setVisible(true);                                            //显示保存对话框
            String path = dialog.getDirectory();                                //获得文件的保存路径
            String saveFileName = dialog.getFile();                             //获得保存的文件名
            if (path == null || saveFileName == null) {
                return;
            }
            OutputStream out = new BufferedOutputStream(new FileOutputStream(path + "/" + saveFileName));   //创建输出流对象
            int len = -1;
            while ((len = in.read()) != -1) {                                   //读取内容，如果没到文件尾则执行循环体
                out.write(len);                                                 //写入文件
            }
            out.close();
            in.close();
        } catch (Exception ex) {
        }
    } else {                                                                    //没有附件执行的代码
        ta_receive.append("----第" + (i + 1) + "个邮件----\n");
        ta_receive.append("主题：" + folder.getMessage(i + 1).getSubject() + "\n");   //主题
        ta_receive.append("正文：" + folder.getMessage(i + 1).getContent() + "\n");   //正文
        ta_receive.append("发送日期：" + folder.getMessage(i + 1).getSentDate() + "\n");   //发送日期
        Address[] ias = folder.getMessage(i + 1).getFrom();                     //发件人地址
        ta_receive.append("发件人：" + ias[0] + "\n");
        Address[] iasTo = folder.getMessage(i + 1).getAllRecipients();          //收件人地址
        ta_receive.append("收件人：" + iasTo[0] + "\n\n");
    }
}
```

秘笈心法

心法领悟 529：通过窗体界面选择需要的未读邮件。

本实例是通过文本域一次性将所有未读邮件显示出来，可能有些邮件并不需要进行查看，为此可以通过窗体界面将所有未读邮件的名称和发件人显示出来，方法是使用 JTable 表格，然后让用户从表格中选择要查看的邮件，并将所选择邮件的内容显示在文本域中。

实例 530　显示已读邮件　　中级

光盘位置：光盘\MR\530　　趣味指数：★★★

实例说明

本实例实现了显示已读邮件的功能。在邮件读取完之后，由于某种原因可能还需要对已读邮件进行查看，为此可以通过查看已读邮件快速找到需要的邮件，而不必查看所有邮件。运行程序，单击窗体上的“接收邮件并下载附件”按钮，可以将已读邮件的内容显示在文本域中，效果如图 19.30 所示。

图 19.30　读取已读邮件的效果

关键技术

本实例主要是通过获得邮件的 UID，从而实现显示已读邮件的功能，方法是在读取邮件时，首先获得邮件的 UID，然后判断该邮件的 UID 是否与文件中存储的已读邮件的 UID 相同，如果相同，则表示是已读邮件。

判断邮件为已读邮件可以通过如下代码实现：

```
Message[] messages = folder.getMessages();                          //获得邮件夹中的所有邮件
for (int i = 0; i < messages.length; i++) {
    long uid = folder.getUID(messages[i]);                          //获得邮件的 UID
    try {
        objectIn = new ObjectInputStream(new FileInputStream("c:/readedEmail.txt"));   //创建输入流对象
        Object readObj = objectIn.readObject();                     //从输入流读取信息
        if (readObj == null) {                                      //文件中未存储已读文件的 UID
            JOptionPane.showMessageDialog(null, "没有已读邮件。");
            return;
        } else {
            readedEmailUIDVector = (Vector<String>) readObj;        //将从文件中读取的内容转换为 Vector 对象
            for (int j = 0; j < readedEmailUIDVector.size(); j++) { //遍历向量对象
                if (String.valueOf(uid).equals(readedEmailUIDVector.get(j))) {   //是已读邮件
                    ReadedNums++;                                   //已读邮件数加 1
                    readMessage(messages[i], folder, i);            //调用 readMessage()方法读取已读邮件内容
                }
```

```
                }
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
```

说明：上面代码中的 folder 是有效的 IMAPFolder 对象的实例；文件 c:/readedEmail.txt 是用于存储已读邮件 UID 的文本文件；readMessage()方法用于读取邮件信息。

设计过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 ReceiveMailFrame 窗体类。

（3）在 ReceiveMailFrame 窗体的内容面板中部添加一个滚动面板，再在滚动面板上添加一个文本域控件，然后在窗体的底部添加一个面板，并在面板上添加两个命令按钮，完成窗体界面的设计。

（4）在 ReceiveMailFrame 窗体类中定义一个 receiveMessage()方法，实现通过邮件的 UID 判断已读邮件并读取已读邮件内容的功能，该方法的代码如下：

```
public void receiveMessage() throws Exception {
    IMAPFolder folder = (IMAPFolder) store.getFolder("inbox");                    //获得 inbox 邮件夹
    if (folder == null) {
        throw new Exception("不存在 inbox 邮件夹。");
    }
    folder.open(Folder.READ_ONLY);                                                //以只读方式打开邮件夹
    Message[] messages = folder.getMessages();                                    //获得邮件夹中的所有邮件
    for (int i = 0; i < messages.length; i++) {
        long uid = folder.getUID(messages[i]);                                    //获得邮件的 UID
        try {
            objectIn = new ObjectInputStream(new FileInputStream("c:/readedEmail.txt"));  //创建输入流对象
            Object readObj = objectIn.readObject();                               //从输入流读取信息
            if (readObj == null) {                                                //文件中未存储已读文件的 UID
                JOptionPane.showMessageDialog(null, "没有已读邮件。");
                return;
            } else {
                readedEmailUIDVector = (Vector<String>) readObj;                  //将从文件中读取的内容转换为 Vector 对象
                for (int j = 0; j < readedEmailUIDVector.size(); j++) {           //遍历向量对象
                    if (String.valueOf(uid).equals(readedEmailUIDVector.get(j))) {  //是已读邮件
                        ReadedNums++;                                             //已读邮件数加 1
                        readMessage(messages[i], folder, i);                      //调用 readMessage()方法读取已读邮件内容
                    }
                }
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }

    }
    ta_receive.append("您共收到" + folder.getMessageCount() + "个邮件。\n");        //显示收到的邮件总数
    if (ReadedNums > 0) {
        if (ReadedNums == folder.getMessageCount()) {
            ta_receive.append("全部已读。\n\n");
        } else {
            ta_receive.append("其中上述" + ReadedNums + "个是已读邮件。\n\n");
        }
    } else {
        ta_receive.append("没有已读邮件。\n\n");
    }
    folder.close(false);                                                          //关闭邮件夹，但不删除邮件
```

```
        store.close();                                          //关闭 Store 对象
        if (objectOut != null)
            objectOut.close();
        if (objectIn != null)
            objectIn.close();
    }
```

（5）在 ReceiveMailFrame 窗体类中定义一个 readMessage()方法，实现读取邮件的内容，该方法的代码与实例 529 中相同，这里不再重复给出。

秘笈心法

心法领悟 530：通过窗体界面选择已读邮件和未读邮件。

在进行邮件程序开发时，可以通过窗体界面将已读邮件和未读邮件分别进行处理，当用户单击“已读邮件”按钮时，只显示已读邮件的列表；当用户单击“未读邮件”按钮时，只显示未读邮件的列表，这样就可以方便用户查看邮件，只要从邮件列表中选择需要查看的邮件，就会将邮件内容显示在文本域中。

Java 安全与 Applet 应用篇

第20章

Java 安全

- Java 对称加密
- Java 非对称加密
- Java 单项加密

20.1　Java 对称加密

实例 531	使用 BASE64 加密 光盘\MR\531	中级 趣味指数：★★★

实例说明

在 Java 加密技术中，BASE64 是一种最简单、最基本的加密技术。本实例使用 BASE64 对字符串“明日科技”进行加密，使其变成一个不可以直接识别的字符串，运行效果如图 20.1 所示。

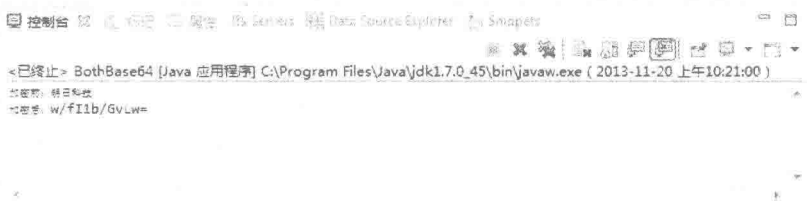


图 20.1　使用 BASE64 加密

说明：BASE64 把原始的数据转换成另一种编码格式保存起来，以至于很多用户都认为 BASE64 不能算作一种加密技术，而只是一种格式转换。

关键技术

使用 BASE64 加密时，使用 BASE64Encoder 类的 encodeBuffer()方法，该方法会产生一个 String 类型的返回值，即加密以后的字符。语法如下：

```
public static String encodeBuffer(byte[] arg0)
```

参数说明

arg0：表示要加密的数据。

设计过程

（1）新建一个 Java 文件。

（2）创建 encryptBASE64()方法，在方法内部创建一个 BASE64Encoder 的对象，使用 BASE64Encoder 的 encodeBuffer()方法加密数据，代码如下：

```
public static String encryptBASE64(byte[] data) {
    //加密数据
    return (new BASE64Encoder()).encodeBuffer(data);
}
```

（3）对“明日科技”字符串进行加密，先将其转成 byte[]类型，再传到 encryptBASE64()方法中，即可得到加密后的数据，代码如下：

```
String data = "明日科技";
System.out.println("加密前：" + data);
String data1 = BothBase64.encryptBASE64(data.getBytes());
System.out.println("加密后：" + data1);
```

秘笈心法

心法领悟 531：使用 BASE64 加密以后的字符串的字节处理。

使用 BASE64 加密以后的字符串个数都是 4 的倍数，一个字符占两个字节，即 BASE64 加密以后的字符串必须是 8 个字节的倍数，如果不足，需在后面补“=”号，所以很多时候能看到使用 BASE64 加密以后的字符以“=”结束。

实例 532 使用 BASE64 解密

光盘位置：光盘\MR\532　　中级　　趣味指数：★★★★

实例说明

BASE64 的加密算法是固定的，对一个明文加密，无论加密多少次，每次产生的密文都是一样的，例如，使用 BASE64 对“明日科技”加密以后的密文是 w/fI1b/GvLw=。

BASE64 的加密方式是可逆的，本实例直接对密文 w/fI1b/GvLw=进行解密，即可得到明文。运行效果如图 20.2 所示。

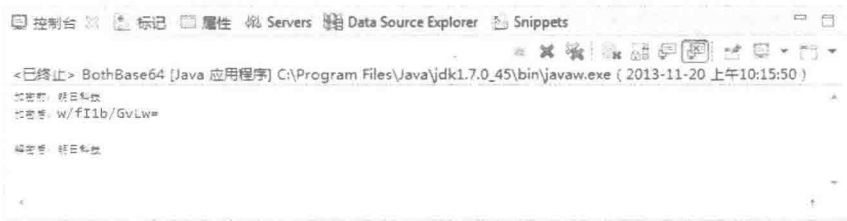


图 20.2　使用 BASE64 解密

关键技术

使用 BASE64 解密时，会用到 BASE64Decoder 类的 decodeBuffer()方法，该方法返回 byte[]类型的数据，即解密以后的数据。语法如下：

```
public static byte[] decodeBuffer(String data)
```

参数说明

data：表示要解密的数据。

设计过程

（1）新建一个 Java 文件。

（2）创建 encryptBASE64()方法，在方法内部创建一个 BASE64Encoder 的对象，使用 BASE64Encoder 的 encodeBuffer()方法加密数据。代码如下：

```
public static String encryptBASE64(byte[] data) {
    //加密数据
    return (new BASE64Encoder()).encodeBuffer(data);
}
```

（3）创建 decryptBASE64()方法，在方法内部创建一个 BASE64Encoder 的对象，使用 BASE64Encoder 的 decodeBuffer()方法解密数据。代码如下：

```
public static byte[] decryptBASE64(String data) throws IOException {
    //解密数据
    return (new BASE64Decoder()).decodeBuffer(data);
}
```

（4）对“明日科技”字符串进行加密，先转成 byte[]类型再传到 encryptBASE64()方法中，即可得到加密以后的数据。再使用 decryptBASE64()方法对加密的数据进行解密，该方法的返回值是 byte[]类型，把 byte[]转换成 String 类型即可得到“明日科技”。代码如下：

```
public static void main(String[] avg) throws IOException {
```

```
        String data = "明日科技";
        System.out.println("加密前：" + data);
        String data1 = BothBase64.encryptBASE64(data.getBytes());
        System.out.println("加密后：" + data1);
        byte[] data2 = BothBase64.decryptBASE64(data1);
        System.out.println("解密后：" + new String(data2));
    }
```

秘笈心法

心法领悟 532：使用 BASE64 加密与解密的回车处理。

使用 BASE64 加密以后的字符串每隔 76 个字符都会插入一个回车符，所以即使明文很长，打印出来的密文也不会在一行显示。但是在解密时，这些回车符则显得无足轻重，即使密文不含回车符也可以正常解密。

实例 533　生成 DES 的密钥

光盘位置：光盘\MR\533

高级

趣味指数：★★★

实例说明

DES 加密是比较早的加密技术，加密时需要先生成一个密钥，加密端用该密钥对数据进行加密，解密时，解密的一端也需要使用该密钥对数据解密。密钥生成以后是 byte[]类型的数据，这种加密和解密使用同一个密钥的技术称为对称加密。本实例中将密钥储存在名为 keyData.dat 的文件中。生成文件如图 20.3 所示。

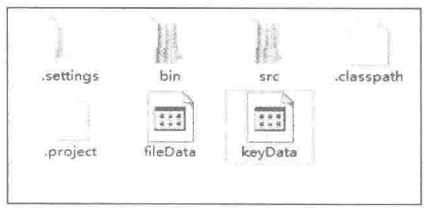


图 20.3　生成密钥文件

关键技术

使用 KeyGenerator 类的 getInstance()方法可以创建密钥生成器，语法如下：

```
public static final KeyAgreement getInstance(String algorithm) throws NoSuchAlgorithmException
```

参数说明

algorithm：表示要使用的加密、解密算法。

设计过程

（1）新建一个 Java 文件。

（2）创建 writeFile()方法，把 byte[]类型的数据保存到文件中，使用这个方法把密钥保存在 keyData.dat 文件中，其中 data 是密钥数据，fileName 是保存的文件路径和名称。使用 FileOutputStream 的构造方法先创建一个 fileOutputStream，再使用 fileOutputStream 把 data 数据写到文件中。代码如下：

```
public void writeFile(byte[] data, String fileName) {
    try {
        //打开文件流
        FileOutputStream fileOutputStream = new FileOutputStream(fileName);
        //写文件
        fileOutputStream.write(data);
        //关闭文件流
        fileOutputStream.close();
    } catch (FileNotFoundException e2) {
```

```
            e2.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

（3）首先通过 KeyGenerator 的 getInstance()方法创建一个密钥生成器，然后通过 SecureRandom 类初始化密钥生成器，接着使用 KeyGenerator 的 generateKey()方法生成 SecretKey，用 SecretKey 类的 getEncoded()方法把密钥转成 byte[]的类型，再使用 writeFile()方法保存密钥。

```
private void initKey() throws NoSuchAlgorithmException {
    //产生一个随机数源
    SecureRandom secureRandom = new SecureRandom();
    //为 DES 算法生成一个 KeyGenerator
    KeyGenerator generator = KeyGenerator.getInstance(algorithm);
    generator.init(secureRandom);
    SecretKey key = generator.generateKey();
    //生成密钥数据，保存到文件中
    writeFile(key.getEncoded(), keyFile);
}
```

秘笈心法

心法领悟 533：KeyGenerator 的初始化。

SecureRandom 用于初始化 KeyGenerator，如果在生成密钥时不使用 SecureRandom 对 KeyGenerator 初始化，KeyGenerator 会使用系统默认的方式自动初始化。

实例 534　使用 DES 加密

光盘位置：光盘\MR\534　　高级　　趣味指数：★★★

实例说明

DES 的密钥生成后，即可对“明日科技”字符串加密，由于加密前密钥保存在 keyData.dat 文件中，先使用 readFile()方法读取密钥文件，再把文件内容转换成 SecretKey 对象，然后对字符串进行加密，加密后会产生一个 byte[]类型的数据，将其保存在 fileData.dat 文件中。生成文件如图 20.4 所示。

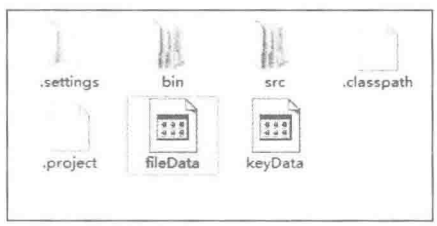


图 20.4　生成加密文件

关键技术

使用 Cipher 类的 getInstance()方法可以返回实现指定转换的 Cipher 对象，语法如下：

```
public static final Cipher getInstance(String transformation) throws NoSuchAlgorithmException, NoSuchPaddingException
```

参数说明

transformation：表示要转换的加密、解密算法。

设计过程

（1）新建一个 Java 文件。

（2）使用 readFile()方法读取密钥数据或者密文数据，代码如下：

```
public byte[] readFile(String fileName) {
    try {
        //创建文件
        File file = new File(fileName);
        //创建文件输入流
        FileInputStream fileInputStream = new FileInputStream(file);
        byte[] data = new byte[(int) file.length()];
        //读取文件输入流
        fileInputStream.read(data);
        fileInputStream.close();
        return data;
    } catch (FileNotFoundException e1) {
        e1.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}
```

（3）创建 toKey()方法，使用 readFile()方法读取密钥文件，使用 DESKeySpec 和 SecretKeyFactory 类把密钥数据转换成密钥对象，代码如下：

```
private Key toKey() throws InvalidKeyException, NoSuchAlgorithmException,
        InvalidKeySpecException {
    byte[] key = readFile(keyFile);
    DESKeySpec keySpec = new DESKeySpec(key);
    SecretKeyFactory factory = SecretKeyFactory.getInstance(algorithm);
    //转化密钥成 Key 进行加密解密
    SecretKey secretKey = factory.generateSecret(keySpec);
    return secretKey;
}
```

（4）创建 encrypt()方法，在方法中先调用 toKey()方法获取密钥，再使用 Cipher.ENCRYPT_MODE 常量和 Key 初始化 Cipher，并对数据进行加密，然后把加密数据保存到文件中，代码如下：

```
public void encrypt(byte[] data) throws InvalidKeyException,
        NoSuchAlgorithmException, InvalidKeySpecException,
        NoSuchPaddingException, IllegalBlockSizeException,
        BadPaddingException {
    Key key = toKey();
    //使用 Cipher 实际完成加密操作
    Cipher cipher = Cipher.getInstance(algorithm);
    //使用密钥初始化 Cipher
    cipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] f = cipher.doFinal(data);
    writeFile(f, dataFile);
}
```

秘笈心法

心法领悟 534：使用 DES 加密的相关方法。

在使用 DES 加密时，使用 writeFile()方法把密文保存在.dat 文件中，把密钥保存到文件中也使用该方法，类似这种可以在多处使用的功能，尽量写成一个方法，以减少代码的重复开发。

实例 535　使用 DES 解密

高级

光盘位置：光盘\MR\535　　趣味指数：★★★

实例说明

密钥保存在 keyData.dat 文件中，密文保存在 fileData.dat 文件中，只要解密方得到加密方传来的这两个文件，即可对其进行解密，看到最后的明文，运行效果如图 20.5 所示。

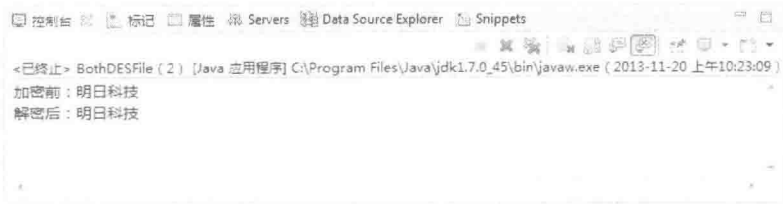


图 20.5　使用 DES 解密

这种应用是很常见的，如某软件公司把自己的软件发布到网上供大家下载，该软件就类似于实例中的 fileData.dat 文件，但是软件的下载者如果没有 keyData.dat 文件，就无法使用该软件，所以需要向软件公司支付一定的费用来获取 keyData.dat 文件。

关键技术

使用 Cipher 类的 init()方法可以用密钥初始化该 Cipher，语法如下：

```
public final void init(int opmode, Key key) throws InvalidKeyException
```

参数说明

- ☑ opmode：表示 Cipher 的操作模式，其值可以是 ENCRYPT_MODE、DECRYPT_MODE、WRAP_MODE 和 UNWRAP_MODE。
- ☑ key：表示密钥。

设计过程

（1）新建一个 Java 文件。

（2）使用 readFile()方法读取密钥数据或者密文数据，代码见实例 534。

（3）创建 toKey()方法，使用 readFile()方法读取密钥文件或者加密文件，使用 DESKeySpec 和 SecretKeyFactory 类把密钥数据转换成密钥对象，代码见实例 534。

（4）创建 decrypt()方法，在方法中调用 toKey()方法获取密钥对象，然后创建一个 DES 的 Cipher 实例，使用 Cipher 类的 init()方法进行初始化，使用常量 Cipher.DECRYPT_MODE 指定 Cipher 为解密模式。接着通过 readFile()方法从 fileData.dat 文件中获取密文，再使用 Cipher 类的 doFinal()方法把密文数据转化成明文，代码如下：

```
public String decrypt() throws InvalidKeyException,
        NoSuchAlgorithmException, InvalidKeySpecException,
        NoSuchPaddingException, IllegalBlockSizeException,
        BadPaddingException {
    Key key = toKey();
    Cipher cipher = Cipher.getInstance(algorithm);
    //指定解密
    cipher.init(Cipher.DECRYPT_MODE, key);
    byte[] f = readFile(dataFile);
    return new String(cipher.doFinal(f));
}
```

秘笈心法

心法领悟 535：使用 DES 加密与解密的开关。

通过学习 DES 加密、解密，细心的读者可能已经注意到，在 DES 的加密或者解密的过程中操作明文或者密文时都是调用 Cipher 中的 doFinal()方法，那么加密和解密的区别在哪里呢？就是在初始化 Cipher 时使用的 init()方法中的常量。该常量就像是一个开关，使用 Cipher.ENCRYPT_MODE 常量时表示要加密；使用 Cipher.DECRYPT_MODE 时表示要解密。

实例 536　PBE 的盐值

光盘位置：光盘\MR\536　　高级　　趣味指数：★★★★

实例说明

PBE 也是一种对称加密算法，该算法需要一个密码用来生成 Key，还需要一个随机数，在加密或解密时，同时使用 Key 和这个随机数才能完成加密和解密工作。这个随机数叫做“盐”，使用盐和 Key 加密完数据后，还要把盐传给解密方对密文解密，所以把盐保存在一个名为 saltData.dat 的文件中。生成文件如图 20.6 所示。

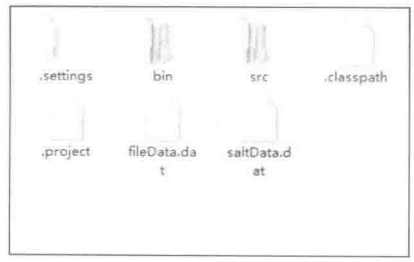


图 20.6　生成 saltData.dat 文件

关键技术

使用 Random 类的 nextBytes()方法可以生成一个随机字节，并将其置于用户提供的 byte 数组中，语法如下：

```
public void nextBytes(byte[] bytes)
```

参数说明

bytes：表示要用随机字节填充的 byte 数组，生成 PBE 盐时字节数组长度必须为 8。

设计过程

（1）新建一个 Java 文件。

（2）创建 writeFile()方法把 byte[]类型的数据保存到文件中，代码如下：

```
public void writeFile(byte[] data, String fileName) {
    try {
        //打开文件流
        FileOutputStream fileOutputStream = new FileOutputStream(fileName);
        //写文件
        fileOutputStream.write(data);
        //关闭文件流
        fileOutputStream.close();
    } catch (FileNotFoundException e2) {
        e2.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（3）定义一个 8 位的 byte[]类型变量 salt，使用 Random 类的 nextBytes 类生成盐，然后把盐值写到 saltData.dat 文件中，代码如下：

```
public void initSalt() {
    byte[] salt = new byte[8];
    Random random = new Random();
    //生成随机数
    random.nextBytes(salt);
    //保存盐值
    writeFile(salt, saltFile);
}
```

秘笈心法

心法领悟 536：PBE 加密或者解密中盐值的使用。

使用 PBE 加密或者解密时，都必须使用相同的盐值，而盐值必须是随机的，同样地，明文使用不同的盐值加密产生的密文是不一样的。

实例 537 生成 PBE 的密钥

光盘位置：光盘\MR\537　　高级　　趣味指数：★★★★

实例说明

生成 PBE 密钥时，首先需要设置一个密码，通过密码生成密钥，然后再使用密钥和盐对明文进行加密；解密时，解密方必须知道密码才能对密文进行解密。本实例演示如何生成 PBE 的密钥。

关键技术

使用 SecretKeyFactory 类的 generateSecret()方法可以根据提供的密钥材料生成 SecretKey 对象，语法如下：

```
public final void SecretKey generateSecret(KeySpec arg0) throws InvalidKeySpecException
```

参数说明

arg0：表示密钥材料的实例。

设计过程

（1）新建一个 Java 文件。

（2）使用密码生成 PBEKeySpec 的实例，再使用加密工厂生成 PBE 的 SecretKeyFactory 实例，然后使用 SecretKeyFactory 的 generateSecret()方法根据 PBEKeySpec 的实例生成密钥，代码如下：

```
private Key toKey(String password) throws NoSuchAlgorithmException,
        InvalidKeySpecException {
    //生成 PBEKeySpec 实例
    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray());
    //生成 SecretKeyFactory 实例
    SecretKeyFactory keyFactory = SecretKeyFactory.getInstance(algorithm);
    SecretKey secretKey = keyFactory.generateSecret(keySpec);
    return secretKey;
}
```

秘笈心法

心法领悟 537：PBE 中密钥的使用方法。

PBE 的密钥不需要加密方生成以后再传输给解密方，而是由解密方自己生成密钥，再通过生成的密钥与盐对密文解密，但是解密方生成密钥时，必须使用加密方提供的密码才能生成正确的密钥。

实例 538 使用 PBE 加密

光盘位置：光盘\MR\538　　高级　　趣味指数：★★★

实例说明

PBE 的加密需要盐值和密码，盐是随机生成的 8 位的 byte[]类型，密码是事先设置好用来生成密钥的。本实例把加密后的数据保存在 fileData.dat 文件中，如图 20.7 所示。

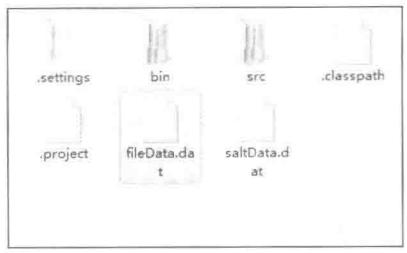


图 20.7　生成密钥文件

关键技术

使用 PBEParameterSpec 的构造方法可以创建一个 PBEParameterSpec 实例，语法如下：

```
public PBEParameterSpec(byte[] salt, int iterationCount)
```

参数说明

❶ salt：表示 PBE 加密时使用的盐值。

❷ iterationCount：表示 PBE 加密时迭代的次数。

设计过程

（1）新建一个 Java 文件。

（2）使用 readFile()方法从 saltData.dat 文件中获取盐值。

（3）创建 encrypt()方法使用盐值和数值 100 生成 PBEParameterSpec 的对象。创建 PBE 的 Cipher 的实例，使用 Cipher.ENCRYPT_MODE、key 和 paramSpec 初始化 cipher。然后使用 doFinal()方法把明文加密成密文，再使用 writeFile()方法把密文数据保存到 fileData.dat 文件中。代码如下：

```
public void encrypt(byte[] data, String password)
        throws NoSuchAlgorithmException, InvalidKeySpecException,
        NoSuchPaddingException, InvalidKeyException,
        InvalidAlgorithmParameterException, IllegalBlockSizeException,
        BadPaddingException {
    //生成密钥
    Key key = toKey(password);
    //获取盐值
    byte[] salt = readFile(saltFile);
    PBEParameterSpec paramSpec = new PBEParameterSpec(salt, 100);
    Cipher cipher = Cipher.getInstance(algorithm);
    //加密数据
    cipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
    writeFile(cipher.doFinal(data), dataFile);
}
```

秘笈心法

心法领悟 538：加密时的迭代数值与加密时间成正比。

在使用 PBEParameterSpec 的构造方法生成 PBE 参数集合时，使用了 100 这个数值作为加密时的迭代数，迭代数值越大，加密时所用的时间就越长；反之，迭代数值越小，加密时间越短。

实例 539　使用 PBE 解密

高级

光盘位置：光盘\MR\539

趣味指数：★★★★

实例说明

加密方确定密码以后即可把密码告诉解密方，待加密完以后，再把盐和加密数据传给解密方。解密方通过

密码生成密钥，再使用密钥和盐对密文进行解密。本实例中，盐值和密文分别保存在 saltData.dat 和 fileData.dat 文件中。解密后即可看见明文数据，运行效果如图 20.8 所示。

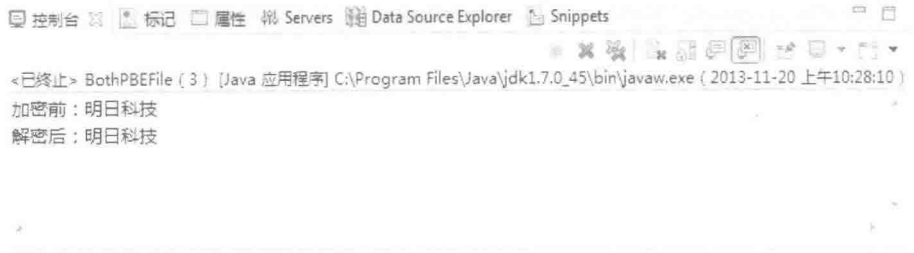


图 20.8　使用 PBE 解密

关键技术

使用 Cipher 类的 doFinal()方法可以按单部分操作加密或解密数据，或者结束一个多部分操作，数据将被加密或解密，语法如下：

```
public final byte[] doFinal(byte[] input) throws IllegalBlockSizeException,BadPaddingException
```

参数说明

input：表示 PBE 加密的输入缓冲区。

设计过程

（1）新建一个 Java 文件。

（2）使用 readFile()方法从 saltData.dat 文件中获取盐值，从 fileData.dat 文件中获取密文。

（3）使用盐值和数值 100 生成 PBEParameterSpec 的对象，然后创建 PBE 的 Cipher 的实例，并使用 Cipher.DECRYPT_MODE、key 和 paramSpec 初始化 Cipher，最后使用 doFinal()方法将密文解密成明文，完成解密工作。代码如下：

```
public String decrypt(String password) throws NoSuchAlgorithmException,
            InvalidKeySpecException, NoSuchPaddingException,
            InvalidKeyException, InvalidAlgorithmParameterException,
            IllegalBlockSizeException, BadPaddingException {
    //获取密钥
    Key key = toKey(password);
    //读取盐值
    byte[] salt = readFile(saltFile);
    //读取加密数据
    byte[] data = readFile(dataFile);
    PBEParameterSpec paramSpec = new PBEParameterSpec(salt, 100);
    Cipher cipher = Cipher.getInstance(algorithm);
    cipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
    //进行解密
    return new String(cipher.doFinal(data));
}
```

秘笈心法

心法领悟 539：PBE 解密关键步骤。

使用 PBE 解密时，先获取密钥和盐值，然后在初始化 Cipher 时设置参数 Cipher.DECRYPT_MODE，表示当前是一个解密操作，最后执行 doFinal()方法完成解密。

20.2　Java 非对称加密

实例 540	生成 RSA 密钥对	高级
	光盘位置：光盘\MR\540	趣味指数：★★★★★

实例说明

RSA 加密必须使用密钥对，即一个公钥、一个私钥。密钥对由服务端生成，生成以后，公钥发送给客户端，供客户端在接收和发送数据时对数据加密、解密；私钥留在服务端，供服务端在发送和接收数据时对数据加密、解密。这种加密和解密使用不同密钥的加密方法称为非对称加密。

本实例中使用 RSA 生成一对密钥，把公钥保存在 keyPublicData.dat 文件中，把私钥保存在 keyPrivateData.dat 文件中，如图 20.9 所示。

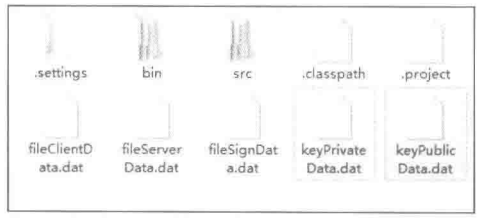


图 20.9　生成密钥文件

关键技术

（1）使用 KeyPairGenerator 类的 generateKeyPair()方法可以生成 RSA 的密钥对，语法如下：

```
public KeyPair generateKeyPair()
```

（2）使用 KeyPair 的 getPublic()和 getPrivate()方法可以分别生成公钥和私钥，语法如下：

```
public PublicKey getPublic()
public PrivateKey getPrivate()
```

设计过程

（1）新建一个 Java 文件。

（2）使用 generateKeyFile()方法可以生成一对密钥。公钥通过 writeFile()方法保存在 keyPublicData.dat 文件中；私钥通过 writeFile()方法保存在 keyPrivateData.dat 文件中。代码如下：

```
public void generateKeyFile() {
        KeyPairGenerator keyPairGen = null;
        try {
                keyPairGen = KeyPairGenerator.getInstance(keyAlgorithm);
        } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
        }
        KeyPair keyPair = keyPairGen.generateKeyPair();

        //公钥
        PublicKey publicKey = keyPair.getPublic();
        writeFile(publicKey.getEncoded(), publickeyFile);

        //私钥
        PrivateKey privateKey = keyPair.getPrivate();
        writeFile(privateKey.getEncoded(), privatekeyFile);
}
```

秘笈心法

心法领悟 540：RSA 的密钥对必须保持一致。

RSA 的密钥对的生成与使用必须配对，通过私钥加密的数据只能使用与其一同生成的公钥才能解密，通过公钥加密的数据也只有使用与其一同生成的私钥才能解密。

实例 541 使用 RSA 的签名

光盘位置：光盘\MR\541　　高级　　趣味指数：★★★★★

实例说明

RSA 的签名也是由服务端生成的，是一个长度为 128 的 byte 类型，当服务端生成签名以后发送给客户端，客户端对该签名进行验证，通过验证以后对数据进行解密。本实例把签名保存在 fileSignData.dat 文件中，运行效果如图 20.10 所示。

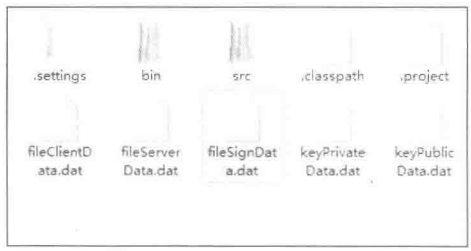


图 20.10　生成数字签名

关键技术

RSA 生成签名时，需要使用私钥和服务端的密文，假设私钥和密文现在已经生成并分别保存在 keyPrivateData.dat 和 keyPublicData.dat 文件中，使用 Signature 类的 sign()方法可以生成数字签名，语法如下：

```
public final byte[] sign() throws SignatureException
```

设计过程

（1）新建两个 Java 文件，一个用于服务端生成签名，一个用于客户端验证签名。

（2）服务端生成密钥对，然后使用私钥和密文生成数字签名，代码如下：

```
public void generateSign() {
    byte[] privateKey = readFile(privatekeyFile);
    byte[] serverData = readFile(serverdataFile);
    //构造 PKCS8EncodedKeySpec 对象
    PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(privateKey);
    //KEY_ALGORITHM 指定的加密算法
    KeyFactory keyFactory = null;
    PrivateKey priKey = null;
    try {
        //生成私钥
        keyFactory = KeyFactory.getInstance(keyAlgorithm);
        priKey = keyFactory.generatePrivate(pkcs8KeySpec);
    } catch (InvalidKeySpecException e) {
        e.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    try {
        //生成数字签名
```

```
            Signature signature = Signature.getInstance(singAlgorithm);
            signature.initSign(priKey);
            signature.update(serverData);
            writeFile(signature.sign(), signdataFile);
        } catch (InvalidKeyException e) {
            e.printStackTrace();
        } catch (SignatureException e) {
            e.printStackTrace();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
    }
```

（3）服务端把数字签名和密文通过某种方式传给客户端。客户端根据公钥和密文使用 verifySign()方法验证数字签名的有效性，只有验证有效时，才可以进行后面的操作，否则中断当前的操作，代码如下：

```
    public boolean verifySign() {
        byte[] data = readFile(serverdataFile);
        byte[] publicKey = readFile(publickeyFile);
        byte[] sign = readFile(signdataFile);
        X509EncodedKeySpec keySpec = new X509EncodedKeySpec(publicKey);
        KeyFactory keyFactory = null;
        PublicKey pubKey = null;
        try {
            keyFactory = KeyFactory.getInstance(keyAlgorithm);
            pubKey = keyFactory.generatePublic(keySpec);
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        try {
            //验证签名
            Signature signature = Signature.getInstance(singAlgorithm);
            signature.initVerify(pubKey);
            signature.update(data);
            //校验成功返回 true，失败返回 false
            return signature.verify(sign);
        } catch (SignatureException e) {
            e.printStackTrace();
        } catch (InvalidKeyException e) {
            e.printStackTrace();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return false;
    }
```

秘笈心法

心法领悟 541：RSA 加密时的密钥算法与证书算法。

使用 RSA 加密，生成密钥时使用 KeyFactory 类的 getInstance()方法，该方法的参数必须指定密钥算法为 RSA；在使用密钥和密文生成数字证书时，则调用 Signature 类的 getInstance()方法，其方法参数要指定证书算法为 MD5withRSA。在加密技术中，不要把密钥算法和证书算法混为一谈。

实例 542　RSA 服务端加密　　高级

光盘位置：光盘\MR\542　　趣味指数：★★★★

实例说明

本实例使用 RSA 算法在服务端把“客户端你好，我是服务端”字符串加密以后传输给客户端，服务端要分

别生成公钥、密文和数字签名，并把这 3 种数据分别传输给客户端。本实例把公钥保存在 keyPublicData.dat 文件中，把密文保存在 fileServerData.dat 文件中，把数字签名保存在 fileSignData.dat 文件中，当客户端得到这 3 个文件以后才能正常对密文解密，解密以后得到的明文就是字符串“客户端你好，我是服务端”。运行效果如图 20.11 所示。

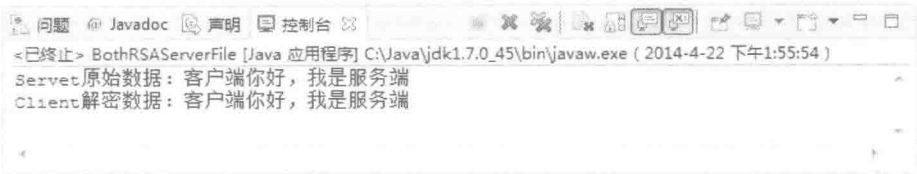


图 20.11　服务端加密数据以后客户端解密

关键技术

使用 KeyFactory 类的 generatePrivate()方法可以获取私钥对象，语法如下：

```
public final PrivateKey generatePrivate(KeySpec keySpec) throws InvalidKeySpecException
```

参数说明

keySpec：表示根据提供的密钥材料生成私钥对象。

设计过程

（1）新建两个 Java 文件，一个用于服务端加密数据，另一个用于客户端解密数据。

（2）在服务端生成公钥和私钥，公钥保存在 keyPublicData.dat 文件中，并传送给客户端使用；私钥保存在 keyPrivateData.dat 文件中，留在本地使用。

（3）服务端使用私钥数据对“客户端你好，我是服务端”加密，并且生成密文保存在 fileServerData.dat 文件中。然后使用密文和 keyPrivateData.dat 文件中的私钥生成签名，把签名保存在 fileSignData.dat 文件中。

（4）服务端将密文数据 fileServerData.dat 文件和签名数据 fileSignData.dat 文件发送给客户端，此时服务端的操作就完成了。

（5）客户端分两次得到服务端传送来的数据，第一次接收到的是公钥文件 keyPublicData.dat；第二次接收到的是密文数据 fileServerData.dat 文件和签名数据 fileSignData.dat 文件。当客户端接收到文件以后，首先进行签名验证。

（6）客户端签名验证通过后，使用公钥和密文进行解密，最后得到明文“客户端你好，我是服务端”。代码如下：

```
String data = "客户端你好，我是服务端";
//服务端操作
BothRSAServerFile bothRSAServerFile = new BothRSAServerFile();
//生成密钥对
bothRSAServerFile.generateKeyFile();
//加密明文
bothRSAServerFile.encryptByPrivateKey(data.getBytes());
//生成签名
bothRSAServerFile.generateSign();
//客户端操作
BothRSAClientFile bothRSAClientFile = new BothRSAClientFile();

byte[]  data1 =null;
if(bothRSAClientFile.verifySign()){
        data1= bothRSAClientFile.decryptByPublicKey();
}
System.out.println("Servet 原始数据："+data);
System.out.println("Client 解密数据："+new String (data1));
```

秘笈心法

心法领悟 542：RSA 加密时服务端向客户端传输数据的过程。

本实例讲述的是 RSA 加密时服务端向客户端传输数据的过程，在该过程中，服务端承担的任务比较多，要生成和保存密钥、密文、签名等。由于数据都是文件的形式，所以服务端与客户端之间的数据传输可以是多样的，但是目前最常用的还是编写应用程序，把文件以网络作为媒体进行传输。

说明：在本实例中，由于服务端与客户端都应用在一台机器上，并且所操作的目录文件名称都是一致的，所以省去了传输的功能。

实例 543　RSA 客户端加密

光盘位置：光盘\MR\543　　高级　　趣味指数：★★★★

实例说明

RSA 服务端加密以后再把数据传送给客户端解密是一个比较繁琐的过程，需要生成数字签名、验证数字签名等，客户端如果回传服务端的数据，则较简单。本实例演示如何由客户端加密数据、服务端解密数据，运行效果如图 20.12 所示。

图 20.12　客户端加密服务端解密

关键技术

使用 KeyFactory 类的 generatePublic()方法可以获取公钥对象，语法如下：

```
public final PublicKey generatePublic(KeySpec keySpec) throws InvalidKeySpecException
```

参数说明

keySpec：表示根据提供的密钥材料生成公钥对象。

设计过程

（1）新建两个 Java 文件，一个用于客户端加密数据，另一个用于服务端解密数据。

（2）客户端用服务端传来的公钥对“服务端你好，这里是客户端”进行加密，并把密文保存在 fileClientData.dat 文件中。服务端接收到 fileClientData.dat 文件后，根据私钥解密，最后得到明文“服务端你好，这里是客户端”。代码如下：

```
//客户端操作
BothRSAClientFile bothRSAClientFile = new BothRSAClientFile();
BothRSAServerFile bothRSAServerFile = new BothRSAServerFile();
String cdata = "服务端你好，这里是客户端";
//获取密文数据
bothRSAClientFile.encryptByPublicKey(cdata.getBytes());
//客户端解密数据
byte [] cdata1 = bothRSAServerFile.decryptByPrivateKey();
```

```
System.out.println("Client 原始数据："+cdata);
System.out.println("Servet 解密数据："+new String (cdata1));
```

秘笈心法

心法领悟 543：RSA 加密中客户端向服务端数据传输的过程。

RSA 加密中，由客户端加密后把密文传给服务端的过程相对来说较简单，客户端不生成签名，服务端也不需要验证签名。

实例 544 DH 服务端加密

光盘位置：光盘\MR\544　　高级　　趣味指数：★★★★

实例说明

DH 加密要先由服务端生成密钥对，并把公钥传给客户端，客户端根据服务端的公钥再生成一对密钥，然后把客户端的公钥传给服务端。这时服务端根据客户端的公钥和自己的私钥生成另一个密钥，这里称为机密密钥。服务端根据机密密钥对明文进行加密，然后把密文传给客户端。客户端根据服务端的密钥和自己的私钥生成客户端的机密密钥，然后再根据自己的机密密钥进行解密。本实例使用 DH 加密算法，在服务端把字符串“客户端你好，我是服务端”加密以后保存在 fileServerData.dat 文件中发送给客户端，然后由客户端解密，在控制台打印出来。运行效果如图 20.13 所示。

图 20.13　服务端加密客户端解密

关键技术

使用 KeyAgreement 类的 init()方法可以用给定的密钥初始化密钥协定，语法如下：

```
public final void init(Key key) throws InvalidKeyException
```

参数说明

key：表示参与者的私有信息。

设计过程

（1）新建两个 Java 文件，一个用于服务端加密数据，另一个用于客户端解密数据。

（2）在服务端创建 generateServerKeyFile()方法生成密钥对，公钥保存在 keyServerPublicData.dat 文件中，私钥保存在 keyServerPrivateData.dat 文件中，代码如下：

```
public void generateServerKeyFile() {
    KeyPairGenerator keyPairGen = null;
    try {
        //生成服务端密钥对
        keyPairGen = KeyPairGenerator.getInstance(keyAlgorithm);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    KeyPair keyPair = keyPairGen.generateKeyPair();
    //公钥
    PublicKey publicKey = keyPair.getPublic();
    writeFile(publicKey.getEncoded(), publicServerkeyFile);
```

```
        //私钥
        PrivateKey privateKey = keyPair.getPrivate();
        writeFile(privateKey.getEncoded(), privateServerkeyFile);
    }
```

（3）服务端把 keyServerPublicData.dat 文件传给客户端，客户端生成公钥后保存在 keyClientPublicData.dat 文件中，再把客户端公钥传给服务端，代码如下：

```
public void generateClientKeyFile() {
        KeyPairGenerator keyPairGen = null;
        try {
                //获取服务端公钥
                byte[] publicServerkey = readFile(publicServerkeyFile);
                X509EncodedKeySpec x509EncodedKeySpec = new X509EncodedKeySpec(publicServerkey);
                KeyFactory keyFactory = KeyFactory.getInstance(keyAlgorithm);
                PublicKey publicKey = keyFactory.generatePublic(x509EncodedKeySpec);
                DHParameterSpec dhParameterSpec = ((DHPublicKey) publicKey).getParams();
                //生成客户端密钥对
                keyPairGen = KeyPairGenerator.getInstance(keyFactory.getAlgorithm());
                keyPairGen.initialize(dhParameterSpec);
        } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
        } catch (InvalidKeySpecException e) {
                e.printStackTrace();
        } catch (InvalidAlgorithmParameterException e) {
                e.printStackTrace();
        }
        KeyPair keyPair = keyPairGen.generateKeyPair();
        //公钥
        PublicKey publicKey = keyPair.getPublic();
        writeFile(publicKey.getEncoded(), publicClientkeyFile);
        //私钥
        PrivateKey privateKey = keyPair.getPrivate();
        writeFile(privateKey.getEncoded(), privateClientkeyFile);
}
```

（4）在服务端创建 getServerSecretKey()方法。因为 DH 服务端的加密并不只是使用自己的私钥，而是根据自己的私钥和客户端的公钥生成一个机密密钥，再使用该机密密钥进行加密，所以 DH 服务端获取到自己的私钥和客户端公钥的二进制文件以后，使用客户端公钥构造一个 X509EncodedKeySpec 对象，KeyFactory 密钥工场可以根据该对象生成一个公钥对象，然后使用自己的私钥构造一个 PKCS8EncodedKeySpec 对象，KeyFactory 密钥工场可以根据该对象生成一个私钥对象。

生成私钥对象和公钥对象后，就要根据这两个密钥生成一个机密密钥。生成机密密钥前，先使用 DH 算法创建一个 KeyAgreement 密钥协定，然后使用私钥初始化该密钥协定，再把公钥传递给密钥协定的 doPhase()方法，最后调用 KeyAgreement 的 generateSecret()方法生成机密密钥，代码如下：

```
private SecretKey getServerSecretKey() {
        byte[] privateServerKey = readFile(privateServerkeyFile);
        byte[] publicClientKey = readFile(publicClientkeyFile);
        X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(publicClientKey);
        PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(privateServerKey);
        Key publicKey = null;
        KeyFactory keyFactory = null;
        Key privateKey = null;
        KeyAgreement keyAgree = null;
        try {
                keyFactory = KeyFactory.getInstance(keyAlgorithm);
                publicKey = keyFactory.generatePublic(x509KeySpec);
                privateKey = keyFactory.generatePrivate(pkcs8KeySpec);
                //创建密钥协议
                keyAgree = KeyAgreement.getInstance(keyFactory.getAlgorithm());
                //初始化密钥
                keyAgree.init(privateKey);
                keyAgree.doPhase(publicKey, true);
                //生成服务端机密密钥
                return keyAgree.generateSecret(secretAlgorithm);
        } catch (InvalidKeyException e) {
                e.printStackTrace();
```

```
        } catch (IllegalStateException e) {
            e.printStackTrace();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
        return null;
    }
```

（5）在服务端创建 encryptForServer()方法，根据机密密钥把数据加密，生成密文文件 fileServerData.dat 并传给客户端，代码如下：

```
    public void encryptForServer(byte[] data) {
        //获取机密密钥
        SecretKey secretKey = getServerSecretKey();
        try {
            //加密数据，然后保存到文件中
            Cipher cipher = Cipher.getInstance(secretKey.getAlgorithm());
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            writeFile(cipher.doFinal(data), serverdataFile);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (NoSuchPaddingException e) {
            e.printStackTrace();
        } catch (InvalidKeyException e) {
            e.printStackTrace();
        } catch (IllegalBlockSizeException e) {
            e.printStackTrace();
        } catch (BadPaddingException e) {
            e.printStackTrace();
        }
    }
```

（6）在客户端创建 getClientSecretKey()方法，使用服务端的公钥和自己的私钥生成客户端的机密密钥，并使用该机密密钥进行解密，代码如下：

```
    public byte[] decryptForClient() {
        //获取机密密钥
        SecretKey secretKey = getClientSecretKey();
        try {
            //读取文件，然后解密数据
            byte[] data = readFile(serverdataFile);
            Cipher cipher = Cipher.getInstance(secretKey.getAlgorithm());
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return cipher.doFinal(data);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (NoSuchPaddingException e) {
            e.printStackTrace();
        } catch (InvalidKeyException e) {
            e.printStackTrace();
        } catch (IllegalBlockSizeException e) {
            e.printStackTrace();
        } catch (BadPaddingException e) {
            e.printStackTrace();
        }
        return null;
    }
```

秘笈心法

心法领悟 544：密钥中公钥与私钥的规范不可混淆。

在生成机密密钥时，把二进制文件分别转换成公钥与私钥的对象，在这个过程中使用到两个类：X509EncodedKeySpec 与 PKCS8EncodedKeySpec。X509EncodedKeySpec 中定义了公钥的规范，所以密钥工场可以使用 X509EncodedKeySpec 生成公钥；而 PKCS8EncodedKeySpec 中定义了私钥的规范，所以密钥工场可以使用 PKCS8EncodedKeySpec 生成私钥，两个密钥规范不可以颠倒使用。

实例 545　DH 客户端加密

光盘位置：光盘\MR\545　　高级　　趣味指数：★★★★

实例说明

DH 的加密算法中，客户端也要生成密钥对，然后通过服务端传来的公钥与自己的私钥共同生成客户端的机密密钥，再使用机密密钥对明文进行加密。本实例中，客户端把字符串“服务端你好，我是客户端”加密以后保存在 fileClientData.dat 文件中发送给服务端，然后由服务端解密以后在控制台打印出来。运行效果如图 20.14 所示。

图 20.14　客户端加密服务端解密

关键技术

使用 KeyAgreement 类的 doPhase()方法可以给定密钥执行此密钥协定的下一个阶段，语法如下：

```
public final Key doPhase(Key key, boolean lastPhase) throws InvalidKeyException,
```

参数说明

❶ key：表示此阶段的密钥。

❷ lastPhase：表示是否是此密钥协定最后阶段的标志。

设计过程

（1）新建两个 Java 文件，一个用于客户端加密数据，一个用于服务端解密数据。

（2）DH 算法中，客户端的机密密钥、数据加密、数据解密都与服务端原理相同，唯一不同的是 DH 的客户端生成密钥比较特别，需要先获取服务端的公钥，用服务端的公钥生成客户端的密钥对。

创建 generateClientKeyFile()方法从服务端处获取公钥文件 keyServerPublicData.dat，然后通过服务端的公钥生成客户端的密钥对。生成客户端密钥对时，获取服务端公钥以后使用 X509EncodedKeySpec 规范和 KeyFactory 生成 PublicKey，再把 PublicKey 转换成 DHPublicKey 对象，然后通过 DHPublicKey 的 getParams()方法得到 DHParameterSpec 实例。密钥对使用 KeyPairGenerator 生成，客户端的密钥对在生成前使用 DHParameterSpec 实例初始化密钥生成器，再分别使用密钥生成器的 getPublic()和 getPrivate()方法得到密钥对。最后把公钥保存在 keyClientPublicData.dat 文件中，把私钥保存在 keyClientPrivateData.dat 文件中。代码如下：

```
public void generateClientKeyFile() {
    KeyPairGenerator keyPairGen = null;
    try {
        //读取服务端公钥
        byte[] publicServerkey = readFile(publicServerkeyFile);
        X509EncodedKeySpec x509EncodedKeySpec = new X509EncodedKeySpec(publicServerkey);
        KeyFactory keyFactory = KeyFactory.getInstance(keyAlgorithm);
        //生成公钥对象
        PublicKey publicKey = keyFactory.generatePublic(x509EncodedKeySpec);
        DHParameterSpec dhParameterSpec = ((DHPublicKey) publicKey).getParams();
        keyPairGen = KeyPairGenerator.getInstance(keyFactory.getAlgorithm());
        keyPairGen.initialize(dhParameterSpec);
    } catch (NoSuchAlgorithmException e) {
```

```
            e.printStackTrace();
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        } catch (InvalidAlgorithmParameterException e) {
            e.printStackTrace();
        }
        //生成客户端密钥对
        KeyPair keyPair = keyPairGen.generateKeyPair();
        //公钥
        PublicKey publicKey = keyPair.getPublic();
        writeFile(publicKey.getEncoded(), publicClientkeyFile);
        //私钥
        PrivateKey privateKey = keyPair.getPrivate();
        writeFile(privateKey.getEncoded(), privateClientkeyFile);
}
```

（3）创建 getClientSecretKey()方法，使用服务端的公钥与自己的私钥生成客户端的机密密钥。代码如下：

```
private SecretKey getClientSecretKey() {
        byte[] privateClientKey = readFile(privateClientkeyFile);
        byte[] publicServerKey = readFile(publicServerkeyFile);
        //创建 X509EncodedKeySpec 实例
        X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(publicServerKey);
        //创建 PKCS8EncodedKeySpec 实例
        PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(privateClientKey);

        PublicKey publicKey = null;
        KeyFactory keyFactory = null;
        Key privateKey = null;
        KeyAgreement keyAgree = null;

        try {
            keyFactory = KeyFactory.getInstance(keyAlgorithm);
            publicKey = keyFactory.generatePublic(x509KeySpec);
            privateKey = keyFactory.generatePrivate(pkcs8KeySpec);
            //创建密钥协议，生成客户端机密密钥
            keyAgree = KeyAgreement.getInstance(keyFactory.getAlgorithm());
            keyAgree.init(privateKey);
            keyAgree.doPhase(publicKey, true);
            return keyAgree.generateSecret(secretAlgorithm);
        } catch (InvalidKeyException e) {
            e.printStackTrace();
        } catch (IllegalStateException e) {
            e.printStackTrace();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (InvalidKeySpecException e) {
            e.printStackTrace();
        }
        return null;
}
```

（4）客户端使用本地的机密密钥进行加密，并把密文保存在 fileClientData.dat 文件中传给服务端。代码如下：

```
public void encryptForClient(byte[] data) {
        //获取机密密钥
        SecretKey secretKey = getClientSecretKey();
        try {
            //客户端数据加密
            Cipher cipher = Cipher.getInstance(secretKey.getAlgorithm());
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            writeFile(cipher.doFinal(data), clientdataFile);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (NoSuchPaddingException e) {
            e.printStackTrace();
        } catch (InvalidKeyException e) {
            e.printStackTrace();
        } catch (IllegalBlockSizeException e) {
            e.printStackTrace();
        } catch (BadPaddingException e) {
            e.printStackTrace();
        }
}
```

（5）服务端接收到 fileClientData.dat 文件后，使用自己的机密密钥进行解密，并把解密内容打印在控制台上。代码如下：

```
public byte[] decryptForServer() {
        //获取机密密钥
        SecretKey secretKey = getServerSecretKey();
        try {
                //读取密文
                byte[] data = readFile(clientdataFile);
                //服务端数据解密
                Cipher cipher = Cipher.getInstance(secretKey.getAlgorithm());
                cipher.init(Cipher.DECRYPT_MODE, secretKey);
                return cipher.doFinal(data);
        } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
        } catch (NoSuchPaddingException e) {
                e.printStackTrace();
        } catch (InvalidKeyException e) {
                e.printStackTrace();
        } catch (IllegalBlockSizeException e) {
                e.printStackTrace();
        } catch (BadPaddingException e) {
                e.printStackTrace();
        }
        return null;
}
```

秘笈心法

心法领悟 545：DH 加密算法的特点。

DH 加密算法的特点在于服务端与客户端都要生成密钥对，其中，客户端的密钥对要使用服务端的公钥进行初始化，然后服务端与客户端使用各自的私钥与对方的公钥生成自己的机密密钥，无论是服务端还是客户端，在对数据的加密与解密时都使用自己的机密密钥完成。

20.3　Java 单项加密

实例 546　使用 MD5 加密　　低级

光盘位置：光盘\MR\546　　趣味指数：★★★★

实例说明

MD5 加密是一种单项加密技术，加密以后生成固定的 byte[]类型的数据，这种数据不能使用某一种方法恢复，可以用于用户的密码加密，例如，一般软件的登录用户的密码都是使用 MD5 加密的，用户注册时，把用户密码进行加密后保存起来，每次用户登录时输入的密码都会被加密一次，再把加密后的密码与保存在数据库中的密码进行比较验证，通过验证才允许登录，这样把 byte[]类型的数据转换成十六进制的字符串更方便操作，如使用 MD5 对“明日科技”加密以后，密文为 ff1b82e719afc4f874b9f47a0b52b666，运行效果如图 20.15 所示。

图 20.15　使用 MD5 对“明日科技”加密

说明：MD5 加密有一个特点，即同样的明文每次使用 MD5 加密生成的密文是不变的。没有随机性，而且生成的密文只是一个信息摘要。并不是对全文的加密，所以每次生成以后都是一个固定位数字节数组。

关键技术

使用 MessageDigest 类的 getInstance()方法可以创建一个 MessageDigest 实例，语法如下：

```
public static MessageDigest getInstance(String algorithm) throws NoSuchAlgorithmException
```

参数说明

algorithm：表示指定算法名称的信息摘要。

设计过程

（1）新建一个 Java 文件。

（2）创建 encryptMD5()方法，指定加密算法为 MD5 加密，对明文数据进行加密，代码如下：

```
public static byte[] encryptMD5(byte[] data) throws NoSuchAlgorithmException {
    //指定加密算法
    MessageDigest digest = MessageDigest.getInstance(algorithm);
    //进行加密
    digest.update(data);
    return digest.digest();
}
```

（3）创建 encryptMD5toString()方法把 byte[]类型的数据转换成十六进制的字符串，具体代码如下：

```
public static String encryptMD5toString(byte[] data) throws NoSuchAlgorithmException {
    String str = "";
    String str16;
    for (int i = 0; i < data.length; i++) {
        //转换为十六进制数据
        str16 = Integer.toHexString(0xFF & data[i]);
        //如果长度为 1，前位补 0
        if (str16.length() == 1) {
            str = str + "0" + str16;
        } else {
            str = str + str16;
        }
    }
    return str;
}
```

秘笈心法

心法领悟 546：使用 toHexString()方法转换字符的数据长度一致。

十六进制的数据最多有两位数字，在使用 Integer 类的 toHexString()方法把一个数据转换成十六进制字符时，有可能是一位数字或两位数字，如“e”和“14”都是十六进制数据，在存储数据时，为了能让加密以后的数据长度一致，把长度是一位的数字前面加“0”补齐，所以十六进制中的“e”表示成“0e”。

实例 547 使用 Hmac 加密

光盘位置：光盘\MR\547

低级

趣味指数：★★★★★

实例说明

Hmac 加密可以用于数据传输的确认，如用户需要进行一次投票，在投票之前服务端先生成一个密钥，然后把密钥发送给客户端，与此同时，服务端使用该密钥对投票的题目进行加密生成一个信息摘要。当客户端得到密钥以后也使用此密钥对投票的题目进行加密生成信息摘要，并把这个信息摘要和投票结果发送给服务端。服务端收到后，把客户端的信息摘要与服务端的信息摘要进行比较，如果两者一致则认为是有效的投票。

本实例中，使用两个类模拟这种客户端与服务端的情况，都对“明日科技”进行加密，如果经比较一致，则输出“验证通过”；否则输出“验证不通过”，运行效果如图 20.16 所示。

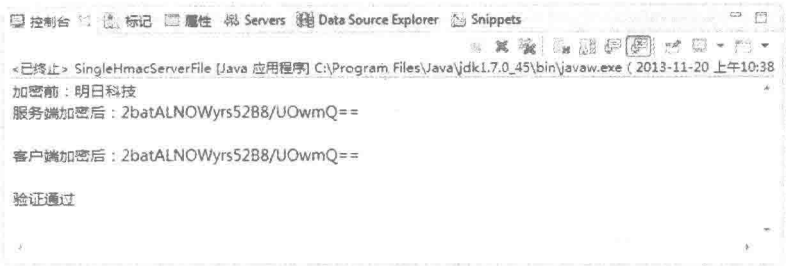


图 20.16　比较 Hmac 摘要信息

关键技术

使用 SecretKeySpec 类的构造函数可以根据密钥文件生成实例，语法如下：

```
public SecretKeySpec(byte[] key, String algorithm)
```

参数说明

❶ key：密钥的内容，复制该数组的内容以防止后续修改。

❷ algorithm：表示密钥内容相关联的密钥算法的名称。

设计过程

（1）新建两个 Java 文件，一个文件用于服务器端加密，另一个文件用于客户端加密。

（2）在服务端创建 initMacKey()方法生成随机密钥，然后把密钥保存在 keyData.bat 文件中。代码如下：

```
public void initMacKey() throws NoSuchAlgorithmException {
        //生成密钥生成器
        KeyGenerator generator = KeyGenerator.getInstance(algorithm);
        //生成密钥
        SecretKey key = generator.generateKey();
        writeFile(key.getEncoded(), keyFile);
}
```

（3）分别在服务端和客户端创建 encryptHMAC()方法加密原文，然后生成 byte[]类型的数据摘要。代码如下：

```
public byte[] encryptHMAC(byte[] data) throws NoSuchAlgorithmException,InvalidKeyException {
        //读取密钥
        byte key[]  文件= readFile(keyFile);
        SecretKey secretKey = new SecretKeySpec(key, algorithm);
        //进行加密操作
        Mac mac = Mac.getInstance(secretKey.getAlgorithm());
        mac.init(secretKey);
        return mac.doFinal();
}
```

（4）对服务端和客户端产生的数据摘要使用 BASE64 进行加密转成 String 类型，然后对二者进行比较，如果相同，则打印“验证通过”；否则打印“验证不通过”。代码如下：

```
public static void main(String[] avg) throws NoSuchAlgorithmException, InvalidKeyException {
        SingleHmacServerFile singleHmacServerFile = new SingleHmacServerFile();
        SingleHmacClientFile singleHmacClientFile = new SingleHmacClientFile();
        String data = "明日科技";
        System.out.println("加密前：" + data);
        String strData = null;
        String strDataClient = null;
        //生成密钥
        singleHmacServerFile.initMacKey();
        //服务器端加密
        strData = BothBase64.encryptBASE64(singleHmacServerFile.encryptHMAC(data.getBytes()));
        //客户端加密
        strDataClient = BothBase64.encryptBASE64(singleHmacClientFile.encryptHMAC(data.getBytes()));
```

```
        System.out.println("服务端加密后：" + strData);
        System.out.println("客户端加密后：" + strDataClient);
        //比较加密结果
        if (strData.equals(strDataClient)) {
            System.out.println("验证通过");
        } else {
            System.out.println("验证不通过");
        }
    }
```

秘笈心法

心法领悟 547：Hmac 加密时密钥的特点。

Hmac 加密时密钥是随机生成的，每个客户端获取的密钥都是唯一的。服务端与客户端密钥与生成的摘要信息的往来，增强了信息在传递过程中的安全性。

实例 548　使用 DSA 加密　低级

光盘位置：光盘\MR\548　趣味指数：★★★★★

实例说明

DSA 的加密适合验证在从服务端向客户端传递数据的过程中，数据是否被第三者修改过。如果把服务端比作老师，把客户端比作家长，老师要把学生成绩单拿给家长看，那么成绩单就可以看作是服务端向客户端传递的数据。老师一般不会直接把成绩单交给家长，而是要通过学生转交，成绩单是公开的，任何人都可以看但不能修改，当家长看到成绩单时，不知道学生传递过程中有没有修改过，这就需要使用一种办法来验证，这就是 DSA 加密。本实例对字符串“明日科技”进行加密，然后验证其有效性，运行效果如图 20.17 所示。

图 20.17　DAS 验证数据传输

关键技术

使用 Signature 类的 verify()方法可以验证签名字节及传输数据是否经过修改，如果没有经过修改，方法返回 true，表示验证通过；如果经过修改，则返回 false，表示验证不通过。语法如下：

```
public final boolean verify(byte[] signature) throws SignatureException
```

参数说明

signature：表示要验证的签名字节。

设计过程

（1）新建两个 Java 文件，一个用于服务端加密，一个用于客户端验证。

（2）DSA 加密需要生成密钥对，其中，公钥传给客户端，私钥在服务端用来生成数字签名。在服务端生成密钥对后，把公钥保存在 keyPublicData.dat 文件中传给客户端，私钥保存在 keyPrivateData.dat 文件中自己使用。

（3）在服务端创建 generatorSign()方法，使用公钥对要传送的数据生成签名。生成数字签名需要使用私钥数据和要传输的数据。使用 PKCS8EncodedKeySpec 类和 KeyFactory 类把二进制私钥数据转化成私钥对象，然后使用私钥对象对 Signature 进行初始化，使用 sign()方法得到数字签名，并把数字签名保存在 fileSignData.dat 文件中，最后把签名文件和字符串“明日科技”直接发送给客户端，代码如下：

```
public void generatorSign(byte[] data) throws NoSuchAlgorithmException, InvalidKeySpecException, InvalidKeyException, SignatureException {
    //读取私钥
    byte[] privateKey = readFile(privatekeyFile);
    PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(privateKey);
    //algorithm 指定的加密算法
    KeyFactory keyFactory = null;
    PrivateKey priKey = null;
    keyFactory = KeyFactory.getInstance(algorithm);
    priKey = keyFactory.generatePrivate(pkcs8KeySpec);
    //生成数字签名
    Signature signature = Signature.getInstance(keyFactory.getAlgorithm());
    signature.initSign(priKey);
    signature.update(data);
    writeFile(signature.sign(), signdataFile);
}
```

（4）把要传输的数据和数字签名发送给客户端，在客户端创建 verifySign()方法，先使用 X509EncodedKeySpec 和 KeyFactory 把公钥的二进制数据转换成公钥对象，再使用公钥对象初始化 Signature 的验证，然后对接收到的字符串“明日科技”进行验证，如果 Signature 类的 verify()方法返回 true，表示数据没有被修改过；否则数据被修改过，代码如下：

```
public boolean verifySign(byte[] data) {
    //获取公钥数据
    byte[] key = readFile(publickeyFile);
    //获取签名
    byte[] sign = readFile(signdataFile);
    X509EncodedKeySpec keySpec = new X509EncodedKeySpec(key);
    KeyFactory keyFactory = null;
    PublicKey publicKey = null;
    try {
        //获取公钥
        keyFactory = KeyFactory.getInstance(algorithm);
        publicKey = keyFactory.generatePublic(keySpec);
    } catch (InvalidKeySpecException e) {
        e.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    try {
        //验证数字签名
        Signature signature = Signature.getInstance(keyFactory.getAlgorithm());
        signature.initVerify(publicKey);
        signature.update(data);
        return signature.verify(sign);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    } catch (InvalidKeyException e) {
        e.printStackTrace();
    } catch (SignatureException e) {
        e.printStackTrace();
    }
    return false;
}
```

秘笈心法

心法领悟 548：DSA 加密算法的特点。

DSA 加密算法的重点不是对原文加密，而是验证原文在传输过程中是否做了修改，所以从服务端到客户端的数据传输是明文，不需要对其进行加密。

第21章

Applet 的应用

- Applet 在 html 中的使用
- Applet 的方法
- Applet 中的文字处理
- Applet 中的图形处理
- Applet 中的图像处理
- Applet 中的文字动画

21.1　Applet 在 html 中的使用

实例 549	在 html 中显示 Applet 光盘位置：光盘\MR\549	初级 趣味指数：★★★

实例说明

Applet 是用 Java 语言开发的、嵌套到 html 中能够在 Internet 中传输的小应用程序。本实例讲解如何在 html 中显示 Applet。运行程序，可以看到在 html 中显示 Applet 的效果，如图 21.1 所示。

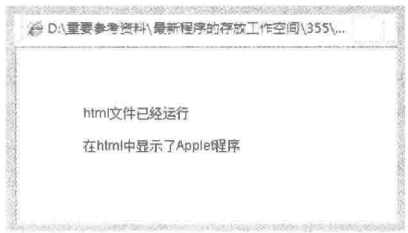


图 21.1　在 html 中显示 Applet 的效果

关键技术

本实例主要是通过继承 Applet 创建小应用程序，然后使用 html 的<applet>标记实现在 html 中显示 Applet，实现代码如下：

```
<applet code = "com.zzk.HtmlShowApplet.class" width ="260" height = "300">
</applet>
```

说明：在<applet>标记中，code 用于指定需要在 html 中显示的 Applet 小应用程序；width 用于指定小应用程序的宽度；height 用于指定小应用程序的高度。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 HtmlShowApplet 小应用程序，用于在其中绘制文本，该类的代码如下：

```
public class HtmlShowApplet extends Applet {
    public void paint(Graphics g){
        g.drawString("html 文件已经运行", 50, 50);                    //绘制文本
        g.drawString("在 html 中显示了 Applet 程序", 50, 80);          //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 HtmlShowApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<applet code = "com.zzk.HtmlShowApplet.class" width ="260" height = "300">
</applet>
```

注意：在运行小应用程序时，一定要双击编译后生成的 bin 文件夹中的 HtmlShowApplet.html 文件，而不是双击 src 文件夹中的 HtmlShowApplet.html 文件，否则程序将会发生错误而无法在浏览器中显示小应用程序，以下各实例与此相同。

秘笈心法

心法领悟 549：为小应用程序指定绝对路径。

在使用<applet>标记运行小应用程序时，如果需要指定绝对路径，可以通过在<applet>标记中添加 codebase 属性实现，例如要运行 c:/com.zzk 文件夹中的 HtmlShowApplet.class 文件，可以使用如下代码：

```
<applet codebase = "c:/" code = "com.zzk.HtmlShowApplet.class" width ="260" height = "300">
</applet>
```

实例 550　设置 Applet 的显示位置

光盘位置：光盘\MR\550　　初级　　趣味指数：★★★

实例说明

通过 html 可以在浏览器中显示 Applet 小应用程序，同时用户还可以根据需要设置 Applet 在浏览器中的显示位置。运行程序，可以看到 Applet 在 html 中居右显示的效果，如图 21.2 所示。

图 21.2　Applet 在 html 中居右显示的效果

关键技术

本实例主要通过在 html 的<applet>标记中使用 Align 属性指定 Applet 的显示位置，在 html 中设置 Applet 显示位置的实现代码如下：

```
<html>
原有内容 1
<applet code = "com.zzk.ShowPositionApplet.class" width ="180" height = "120" Align = "right">
</applet>
原有内容 2
</html>
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 ShowPositionApplet 小应用程序，用于在其中绘制文本，该类的代码如下：

```
public class ShowPositionApplet extends Applet {
    String author;                                  //声明成员变量
    public void init() {
        author = "ZhenKun Zhang";                   //初始化成员变量
    }
    public void paint(Graphics g){
        g.setColor(Color.blue);                     //设置颜色
        g.drawString(author, 50, 30);               //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 ShowPositionApplet.html 的文件，用于设置 Applet 小应用程序在

浏览器中的显示位置，该.html 文件的代码如下：

```
<html>
原有内容 1
<applet code = "com.zzk.ShowPositionApplet.class" width ="180" height = "120" Align = "right">
</applet>
原有内容 2
</html>
```

秘笈心法

心法领悟 550：使用<applet>标记中的 code 属性必须注意的事项。

在使用<applet>标记运行小应用程序时，code 属性一定要使用对源程序编译后生成的字节码文件，并且如果有包，还必须指定完整的包名和字节码文件名，否则将无法运行小应用程序。

实例 551　Applet 获取页面传递的参数

初级

光盘位置：光盘\MR\551

趣味指数：★★

实例说明

通过 html 可以在浏览器中显示 Applet 小应用程序，同时用户还可以根据需要通过 html 页面向 Applet 传递参数。运行程序，可以看到在 Applet 中显示了从 html 页面中获得的参数值，效果如图 21.3 所示。

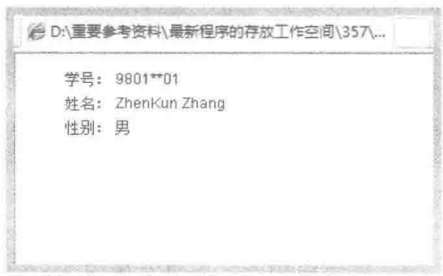


图 21.3　Applet 获取到 html 传递的参数

关键技术

本实例主要是通过在 html 的<applet>标记中添加子标记<param>，并在该标记中通过 name 属性指定参数的名称，通过 value 属性指定参数的值，在 html 中向 Applet 传递参数的代码如下：

```
<applet   code = "com.zzk.ShowParamApplet.class" width ="260" height = "180">
<param name = "name" value = "ZhenKun Zhang" />
</applet>
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 ShowParamApplet 小应用程序，用于获得从 html 页面传递的参数值，并在小应用程序中显示，该类的代码如下：

```
public class ShowParamApplet extends Applet {
    String id;                                  //声明成员变量
    String name;                                //声明成员变量
    String sex;                                 //声明成员变量
    public void init() {
        id = getParameter("id");                //获得参数 id 的值
        name = getParameter("name");            //获得参数 name 的值
        sex = getParameter("sex");              //获得参数 sex 的值
```

```
    }
    public void paint(Graphics g) {
        g.setColor(Color.blue);                    //设置颜色
        g.drawString("学号：", 30, 10);             //绘制文本
        g.drawString(id, 70, 10);                  //绘制获得的 id 值
        g.drawString("姓名：", 30, 30);             //绘制文本
        g.drawString(name, 70, 30);                //绘制获得的 name 值
        g.drawString("性别：", 30, 50);             //绘制文本
        g.drawString(sex, 70, 50);                 //绘制获得的 sex 值
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 ShowParamApplet.html 的 html 文件，用于向 Applet 小应用程序传递参数和显示小应用程序，该 html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.ShowParamApplet.class" width ="260" height = "180">
<param name = "id" value = "9801**01" />
<param name = "name" value = "ZhenKun Zhang" />
<param name = "sex" value = "男" />
</applet>
</html>
```

秘笈心法

心法领悟 551：解决浏览器无法完全显示 Applet 的问题。

在使用浏览器打开 Applet 时，有时会出现无法完全显示 Applet 小应用程序的情况，这时可以将.html 文件中<applet>标记中的 width 属性和 height 属性的值调大些，即可完全显示 Applet 小应用程序。

实例 552 使用<applet>标记中的 archive 属性

光盘位置：光盘\MR\552　　初级　　趣味指数：★★★★

实例说明

通过 html 可以在浏览器中显示 Applet 小应用程序，如果在使用 Applet 时已经存在一个.jar 文件，则可以通过 html 将该.jar 文件与 Applet 关联，从而可以直接使用该.jar 文件所包含的内容，而不必重新创建与该文件相同的内容。运行程序，可以看到将图 21.4 所示的.jar 文件所包含的添加有 Swing 控件的面板，添加到如图 21.5 所示的在浏览器中显示的 Applet 小应用程序中的效果。

图 21.4　在.jar 文件中包含的面板

图 21.5　在 Applet 中显示关联的.jar 文件包含的内容

关键技术

本实例主要是在 html 的<applet>标记中，通过 archive 属性指定关联的.jar 文件，以实现通过 Applet 小应用

程序调用该.jar 文件中包含的内容。在 html 中使用<applet>标记中的 archive 属性的实现代码如下：

```
<applet code = "com.zzk.UseArchivePropertyApplet.class" width ="300" height = "220" archive = "NewPanelApp.jar">
</applet>
```

设计过程

（1）新建一个名为 552 的项目，在该项目中创建一个名为 NewPanelApp 的项目，然后在 NewPanelApp 项目的 src 文件夹中创建一个包 npanel，并在该包中创建一个名为 NewPanel 的 JPanel 面板类，该面板类的代码如下：

```
package npanel;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JTextField;
public class NewPanel extends JPanel {
    private JTextField textField;                          //声明文本框
    public NewPanel() {
        super();                                           //调用超类的构造方法
        setLayout(null);                                   //设置面板为绝对布局
        setBounds(100, 100, 254, 167);                     //设置面板的位置和大小
        final JButton button = new JButton();              //创建按钮
        button.setText("按钮一");                          //指定按钮标题
        button.setBounds(38, 86, 73, 28);                  //指定按钮在面板中的显示位置和大小
        add(button);                                       //将按钮添加到面板上
        final JButton button_1 = new JButton();            //创建按钮
        button_1.setText("按钮二");                        //指定按钮标题
        button_1.setBounds(140, 86, 73, 28);               //指定按钮在面板中的显示位置和大小
        add(button_1);                                     //将按钮添加到面板上
        textField = new JTextField();                      //创建文本框
        textField.setText("这是一个文本框控件");           //指定文本框的标题
        textField.setBounds(38, 35, 175, 22);              //设置文本框的显示位置和大小
        add(textField);                                    //将文本框添加到面板上
    }
}
```

（2）使用 Eclipse 的导出功能将 NewPanelApp 项目导出为 NewPanelApp.jar 文件，并将该文件复制到名为 552 的项目 src 文件夹中。

（3）在名为 552 的项目 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 JApplet 类的 UseArchivePropertyApplet 小应用程序，在该小应用程序中使用与 html 关联的 NewPanelApp.jar 文件，该类的代码如下：

```
public class UseArchivePropertyApplet extends JApplet {
    public void init() {
        //使用与.html 文件关联的 NewPanelApp.jar 文件中的 NewPanel 类创建对象
        npanel.NewPanel npanel = new npanel.NewPanel();
        setLayout(null);                                   //设置容器为绝对布局
        npanel.setBounds(10, 10, 254, 167);                //设置 NewPanel 对象的显示位置和大小
        add(npanel);                                       //将 NewPanel 对象添加到容器中
    }
}
```

（4）在名为 552 的项目 src 文件夹中创建一个名为 UseArchivePropertyApplet.html 的文件，用于通过.html 文件关联 Applet 与 NewPanelApp.jar 文件，并在浏览器中显示 Applet 文件，该.html 文件的代码如下：

```
<applet code = "com.zzk.UseArchivePropertyApplet.class" width ="300" height = "220" archive = "NewPanelApp.jar">
</applet>
```

秘笈心法

心法领悟 552：避免 Applet 使用.jar 文件时编译错误。

为了避免本实例在编译时出错，可以将 NewPanelApp.jar 文件复制到名为 552 的项目文件夹中，然后将其配置到该项目的类库路径中，这样程序在编译时就不会出错了。

21.2 Applet 的方法

实例 553 使用 paint()方法绘制页面内容

初级

光盘位置：光盘\MR\553

趣味指数：★★★

实例说明

使用 Applet 在小应用程序中绘制的内容通常都是使用 paint()方法实现的。本实例实现了使用 paint()方法在小应用程序中绘制内容。运行程序，可以看到在小应用程序中绘制的内容，效果如图 21.6 所示。

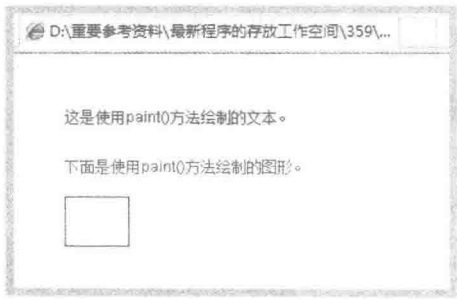


图 21.6 使用 paint()方法在小应用程序中绘制内容

关键技术

本实例主要是通过 paint()方法的形参 g 实现了在小应用程序中绘制内容的功能，该方法的定义如下：

```
public void paint(Graphics g)
```

参数说明

g：Graphics 类的实例，可以通过该类提供的方法在小应用程序中绘制文本、图形和图像等内容。

说明：Applet 的 paint()方法是从 Container 类继承的方法，在使用时需要重写该方法。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 PaintMethodApplet 小应用程序，用于在其中使用 paint()方法绘制内容，该类的代码如下：

```
public class PaintMethodApplet extends Applet {
    public void paint(Graphics g) {
        g.setColor(Color.BLUE);                                  //设置颜色
        g.drawString("这是使用 paint()方法绘制的文本。", 30, 40);    //绘制文本内容
        g.setColor(Color.RED);                                   //设置颜色
        g.drawString("下面是使用 paint()方法绘制的图形。", 30, 80);  //绘制文本内容
        g.drawRect(30, 100, 50, 40);                             //绘制图形
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 PaintMethodApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.PaintMethodApplet.class" width ="260" height = "180">
</applet>
</html>
```

秘笈心法

心法领悟 553：使用 paint()方法绘制对象。

使用 paint()方法可以在 Applet 页面中绘制文本、图形和图像等内容，如果希望绘制对象，将 paint()方法的 Graphics 类型的参数强制转换为 Graphics2D 类型，即可在 Applet 页面中绘制对象。

实例 554　使用 update()方法更新页面内容

光盘位置：光盘\MR\554　　初级　　趣味指数：★★★

实例说明

在 Applet 小应用程序中，除了可以使用 paint()方法绘制内容外，还可以使用 update()方法更新页面内容。本实例实现了使用 update()方法更新页面内容的功能。运行程序，每次使浏览器窗口由最小化状态转换为非最小化状态，或从非最小化状态转换为最小化状态时，就会看到浏览器页面中内容的变化，效果分别如图 21.7 和图 21.8 所示。

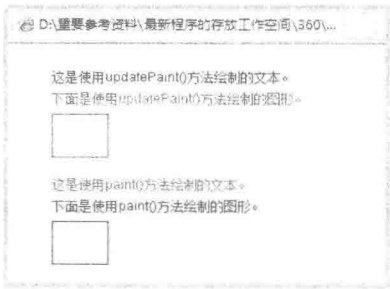


图 21.7　使用 update()方法更新页面 1

图 21.8　使用 update()方法更新页面 2

关键技术

本实例主要是通过在 update()方法中标记变量对页面的变化进行控制，从而实现了使用 update()方法更新页面的功能，该方法的定义如下：

```
public void update(Graphics g)
```

参数说明

g：Graphics 类的实例，可以通过该类提供的方法更新小应用程序页面的内容。

说明：Applet 的 update()方法也是从 Container 类继承的方法，在使用时可以根据需要确定是否重写。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 UpdateMethodApplet 小应用程序，用于在其中使用 update()方法更新内容，该类的代码如下：

```
public class UpdateMethodApplet extends Applet {
    boolean flag = false;                                    //定义标记变量
    public void start() {
        repaint();                                           //重新调用 paint()方法
    }
    public void paint(Graphics g) {
        g.setColor(Color.RED);                               //设置颜色
```

```
        g.drawString("这是使用 paint()方法绘制的文本。", 30, 120);          //绘制文本
        g.setColor(Color.BLUE);                                          //设置颜色
        g.drawString("下面是使用 paint()方法绘制的图形。", 30, 140);        //绘制文本
        g.drawRect(30, 150, 50, 40);                                     //绘制矩形
        update(g);                                                       //调用 update()方法
    }
    public void update(Graphics g) {
        if (flag) {
            g.clearRect(0, 0, 300, 220);                                 //标记变量为 true 时，清除内容
            flag = false;                                                //设置标记变量为 false
        } else {
            flag = true;                                                 //设置标记变量为 true
        }
        g.setColor(Color.BLUE);                                          //设置颜色
        g.drawString("这是使用 updatePaint()方法绘制的文本。", 30, 20);     //绘制文本
        g.setColor(Color.RED);                                           //设置颜色
        g.drawString("下面是使用 updatePaint()方法绘制的图形。", 30, 40);   //绘制文本
        g.drawRect(30, 50, 50, 40);                                      //绘制矩形
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 UpdateMethodApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.UpdateMethodApplet.class" width ="300" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 554：使用 update()方法解决画图时屏幕闪烁问题。

在使用 Graphics 画图时，有时会出现屏幕闪烁，这时可以重写 update()方法，并在该方法中调用 paint()方法，即可解决画图时屏幕闪烁的问题。

实例 555　使用 repaint()方法重新绘制页面

初级

光盘位置：光盘\MR\555

趣味指数：★★★

实例说明

在 Applet 小应用程序中绘制内容之后，如果希望重新绘制页面内容，可以通过 repaint()方法实现。本实例讲解如何使用 repaint()方法重新绘制页面内容。运行程序，可以在浏览器窗口中看到如图 21.9 所示的内容，然后刷新浏览器页面，可以看到重新绘制的内容，效果如图 21.10 所示。

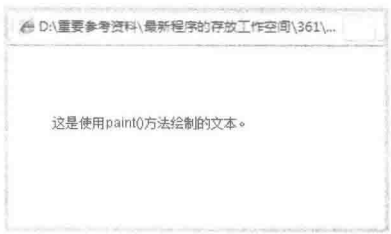


图 21.9　初次运行时绘制的内容

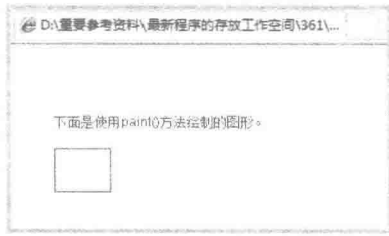


图 21.10　刷新页面后绘制的内容

关键技术

本实例主要是通过定义静态成员变量 iFlag 并在 Applet 的 start()方法中调整 iFlag 的值，然后调用 repaint()

方法，从而实现了重新绘制页面内容的功能。定义静态成员变量和 start()方法的代码如下：

```
static int iFlag = 0;                                    //定义标记变量
public void start() {
    iFlag++;                                             //调整标记变量的值
    repaint();                                           //重新调用 paint()方法
}
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 RepaintMethodApplet 小应用程序，用于在其中使用 repaint()方法重新绘制内容，该类的代码如下：

```
public class RepaintMethodApplet extends Applet {
    static int iFlag = 0;                                //定义标记变量
    public void start() {
        iFlag++;                                         //调整标记变量的值
        repaint();                                       //重新调用 paint()方法
    }
    public void paint(Graphics g) {
        if (iFlag == 1) {
            g.drawString("这是使用 paint()方法绘制的文本。", 30, 60);    //绘制文本
        } else if (iFlag == 2) {
            g.setColor(Color.RED);                       //设置颜色
            g.drawString("下面是使用 paint()方法绘制的图形。", 30, 60);  //绘制文本
            g.drawRect(30, 80, 50, 40);                  //绘制矩形
        } else if (iFlag == 3) {
            g.setColor(Color.BLUE);                      //设置颜色
            g.drawString("下面是使用 paint()方法绘制的图形。", 30, 60);  //绘制文本
            g.drawRect(30, 80, 50, 40);                  //绘制矩形
        } else {
            g.setColor(Color.GREEN);                     //设置颜色
            g.drawString("下面是使用 paint()方法绘制的图形。", 30, 60);  //绘制文本
            g.drawRect(30, 80, 50, 40);                  //绘制矩形
        }
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 RepaintMethodApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.RepaintMethodApplet.class" width ="300" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 555：repaint()方法调用 paint()和 update()方法的顺序。

如果当前组件是轻量级组件，repaint()方法会尽快调用此组件的 paint()方法；否则，repaint()方法会尽快调用此组件的 update()方法。

实例 556　Applet 显示地址栏上的路径

初级

光盘位置：光盘\MR\556

趣味指数：★★★★

实例说明

在浏览器中运行 Applet 小应用程序时，如果需要获得地址栏上的路径，可以通过本实例讲解的方法实现。

运行程序，将在浏览器窗口中显示地址栏上的路径，效果如图 21.11 所示。

图 21.11　Applet 显示地址栏上路径的效果

关键技术

本实例主要是通过 Applet 类的 getDocumentBase()方法获得地址栏上路径的 URL，然后使用 URL 对象的 getFile()方法获得地址栏上的路径。

（1）通过 Applet 类的 getDocumentBase()方法可以获得地址栏上路径的 URL，该方法的定义如下：

```
public URL getDocumentBase()
```

参数说明

返回值：该方法执行成功，则获得地址栏上路径的 URL 对象。

（2）通过 URL 类的 getFile()方法可以获得当前 URL 对象的完整路径，该方法的定义如下：

```
public String getFile()
```

参数说明

返回值：该方法执行成功，则获得当前 URL 对象的完整路径。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 GainAddressBarPathApplet 小应用程序，用于在其中显示浏览器地址栏上的路径，该类的代码如下：

```
public class GainAddressBarPathApplet extends Applet {
    URL url;                                    //声明 URL 对象
    public void start() {
        url = this.getDocumentBase();           //获得地址栏上路径的 URL 对象
    }
    public void paint(Graphics g) {
        g.setColor(Color.blue);                 //设置颜色
        g.drawString(url.getFile(), 30, 20);    //绘制地址栏上的路径
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 GainAddressBarPathApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.GainAddressBarPathApplet.class" width ="1200" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 556：获得地址栏路径的另一种方法。

本实例是使用 URL 类的 getFile()方法获得地址栏上的路径的，实际开发中，还可以使用 URL 类的 toString()方法，区别在于使用 toString()方法获得的路径前有“file:”标记，而使用 getFile()方法则没有。

实例 557　Applet 显示 class 存放的路径

光盘位置：光盘\MR\557　　初级　　趣味指数：★★★★

实例说明

在使用 Applet 小应用程序时，如果希望获得 class 的存放路径，可以通过本实例讲解的方法实现。运行程序，将在浏览器窗口中显示 class 的存放路径，效果如图 21.12 所示。

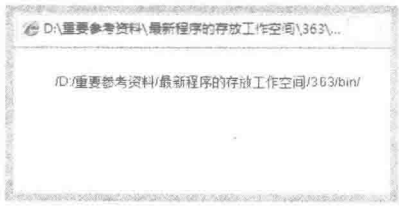


图 21.12　Applet 显示 class 的存放路径

说明：由于本实例的字节码文件存放到 bin 文件夹中，因此在浏览器中显示了图 21.12 所示的路径。

关键技术

本实例主要是通过 Applet 类的 getCodeBase()方法获得 class 存放路径的 URL 对象，然后使用 URL 对象的 getFile()方法获得 class 存放路径的字符串。

通过 Applet 类的 getCodeBase()方法可以获得 class 存放路径的 URL 对象，该方法的定义如下：

```
public URL getCodeBase()
```

参数说明

返回值：该方法执行成功，则获得 class 存放路径的 URL 对象。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 ClassSavePathApplet 小应用程序，用于在其中获得 class 的存放路径，该类的代码如下：

```
public class ClassSavePathApplet extends Applet {
    URL url;                                    //声明 URL 对象
    public void start() {
        url = this.getCodeBase();               //获得 class 存放路径的 URL 对象
    }
    public void paint(Graphics g) {
        g.setColor(Color.blue);                 //设置颜色
        g.drawString(url.getFile(), 30, 20);    //绘制 class 的存放路径
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 ClassSavePathApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.ClassSavePathApplet.class" width ="1200" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 557：在 Applet 中获得需要的资源。

由于 Applet 的安全限制，当需要使用 Applet 获得某些资源时，如需要图片文件或其他文件时，将这些需要的资源放到 class 的存放路径中，即可获得 class 的存放路径，进而获得 class 路径下的资源。

21.3 Applet 中的文字处理

实例 558	Applet 字体大小 资源\MR\558	初级 趣味指数：★★★★

实例说明

在 Applet 小应用程序绘制文本时，字体的大小是固定的，如果希望改变字体的大小，可以通过本实例讲解的方法实现。运行程序，将在浏览器窗口中看到改变字体大小后的效果，如图 21.13 所示。

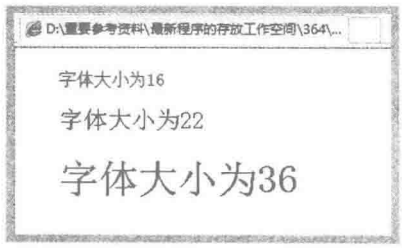


图 21.13 设置不同字体大小的显示效果

关键技术

本实例主要是通过 Font 类创建指定大小的字体对象，然后使用 Graphics 类的 setFont()方法设置绘图上下文的字体，从而实现了改变字体大小的功能，实现代码如下：

```
Font font = new Font("宋体", Font.PLAIN, 16);          //创建字体对象，字体大小为 16
g.setFont(font);                                       //设置字体
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 SetFontSizeApplet 小应用程序，用于在其中设置文本的字体，该类的代码如下：

```
public class SetFontSizeApplet extends Applet {
    public void paint(Graphics g) {
        Font font = new Font("宋体", Font.PLAIN, 16);          //创建字体对象，字体大小为 16
        g.setFont(font);                                       //设置字体
        g.drawString("字体大小为 16", 30, 20);                  //绘制文本
        font = new Font("宋体", Font.PLAIN, 22);               //创建字体对象，字体大小为 22
        g.setFont(font);                                       //设置字体
        g.drawString("字体大小为 22", 30, 60);                  //绘制文本
        font = new Font("宋体", Font.PLAIN, 36);               //创建字体对象，字体大小为 36
        g.setFont(font);                                       //设置字体
        g.drawString("字体大小为 36", 30, 120);                 //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 SetFontSizeApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.SetFontSizeApplet.class" width ="300" height = "220">
```

```
</applet>
</html>
```

秘笈心法

心法领悟 558：设置字体时需要注意的事项。

在为 Applet 设置字体时，为了使设置的字体有效，必须先设置字体，然后再绘制文本，这样才能按设置的字体显示文本，如果在程序中又重新设置了字体，则以后绘制的文本将使用新设置的字体。

实例 559　在 Applet 文字位置　初级

趣味指数：★★★

实例说明

在 Applet 小应用程序绘制文字内容时，可以对文字的显示位置进行控制。本实例实现了控制文字位置的功能。运行程序，将在浏览器窗口中看到设置文字位置后的效果，如图 21.14 所示。

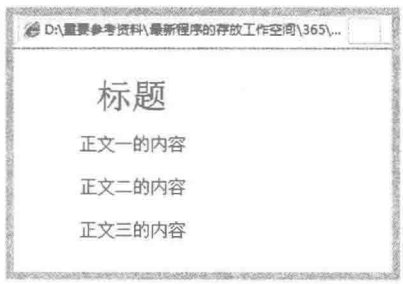


图 21.14　设置文字位置的显示效果

关键技术

本实例主要是通过 Graphics 类的 drawString()方法实现了设置文字显示位置的功能，实现代码如下：

```
g.drawString("标题", 65, 40);                    //绘制文本，其位置坐标为(65, 40)
```

说明：drawString()方法的第一个参数是绘制的文字内容，后两个参数分别表示绘制文字的横坐标值和纵坐标值，因此上面代码将在坐标（65, 50）处绘制文字内容“标题”。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 SetTextPositionApplet 小应用程序，用于在其中实现控制文字位置的功能，该类的代码如下：

```
public class SetTextPositionApplet extends Applet {
    public void paint(Graphics g) {
        Font font = new Font("宋体", Font.PLAIN, 32);          //创建字体对象
        g.setFont(font);                                       //设置字体
        g.drawString("标题", 65, 40);                          //绘制文本，其位置坐标为(65, 40)
        font = new Font("宋体", Font.PLAIN, 16);               //创建字体对象
        g.setFont(font);                                       //设置字体
        g.drawString("正文一的内容", 50, 80);                  //绘制文本，其位置坐标为(50, 80)
        g.drawString("正文二的内容", 50, 120);                 //绘制文本，其位置坐标为(50, 120)
        g.drawString("正文三的内容", 50, 160);                 //绘制文本，其位置坐标为(50, 160)
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 SetTextPositionApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.SetTextPositionApplet.class" width ="300" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 559：试探法控制文字的显示位置。

在 Applet 中控制文字显示位置时，不可能一次就将文字设置到合适的位置，这时可以试探着改变文字绘制点的坐标，然后再看文字位置是否合适，并重复上述过程直到满意为止。

实例 560　控制 Applet 字体样式

初级

光盘位置：光盘\MR\560

趣味指数：★★

实例说明

在 Applet 小应用程序中绘制的文本，除了可以设置字体的大小，还可以设置字体的样式。本实例实现了设置字体样式的功能。运行程序，将在浏览器窗口中看到设置不同字体样式的效果，如图 21.15 所示。

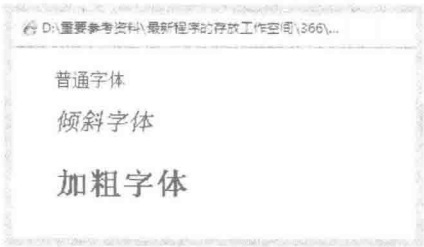


图 21.15　设置不同字体样式的效果

关键技术

本实例主要是通过 Font 类创建具有指定样式的字体对象，然后使用 Graphics 类的 setFont()方法设置绘图上下文的字体，从而实现了设置字体样式的功能，其中 Font 类的构造方法的定义如下：

```
public Font(String name, int style, int size)
```

参数说明

❶ name：需要指定的字体名称。

❷ style：需要指定的字体样式。

❸ size：需要设置的字体大小。

说明：该构造方法的第二个参数 style 可以取值为 Font.BOLD（粗体样式）、Font.ITALIC（斜体样式）和 Font.PLAIN（普通样式）。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 SetFontStyleApplet 小应用程序，用于在其中设置字体的样式，该类的代码如下：

```
public class SetFontStyleApplet extends Applet {
```

```
    public void paint(Graphics g) {
        Font font = new Font("宋体", Font.PLAIN, 16);          //创建字体对象，字体样式为普通字体
        g.setFont(font);                                       //设置字体
        g.drawString("普通字体", 30, 20);                      //绘制文本
        font = new Font("宋体", Font.ITALIC, 22);              //创建字体对象，字体样式为倾斜字体
        g.setFont(font);                                       //设置字体
        g.drawString("倾斜字体", 30, 60);                      //绘制文本
        font = new Font("宋体", Font.BOLD, 28);                //创建字体对象，字体样式为加粗字体
        g.setFont(font);                                       //设置字体
        g.drawString("加粗字体", 30, 120);                     //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 SetFontStyleApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.SetFontStyleApplet.class" width ="300" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 560：混合使用多种字体样式。

在 Applet 中设置字体样式时，除了可以指定粗体、斜体等单一的字体样式，还可以混合指定多种字体样式，方法是在设置字体样式时，使用“|”进行分隔，例如，粗斜体可以使用如下代码实现：

```
Font font = new Font( "宋体" , Font.BOLD | Font.ITALIC , 16);          //创建字体对象，字体样式为粗斜体
```

实例 561　Applet 中绘制立体效果的文字

初级

光盘位置：光盘\MR\561

趣味指数：★★★

实例说明

在 Applet 中绘制的文字，除了可以指定字体大小和字体样式，还可以设置为立体效果。本实例实现了在 Applet 中绘制立体效果文字的功能。运行程序，将在浏览器窗口中看到立体效果的文字，如图 21.16 所示。

图 21.16　在 Applet 中绘制的立体效果文字

关键技术

本实例主要是使用循环语句在 Applet 中绘制文字，并对绘制的文字进行偏移，再使底层的文字以灰色显示，最上面的文字以黑色显示，从而实现了立体效果文字的绘制，实现代码如下：

```
g.setColor(Color.GRAY);                                //设置颜色为灰色
int i = 0;                                             //循环变量
while (i < 8) {
    g.drawString(value, x, y);                         //绘制文本
    x += 1;                                            //调整绘制点的横坐标值
```

```
        y += 1;                                    //调整绘制点的纵坐标值
        i++;                                       //调整循环变量的值
    }
    g.setColor(Color.BLACK);                       //设置颜色为黑色
    g.drawString(value, x, y);                     //绘制文本
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 SolidTextApplet 小应用程序，用于在其中绘制立体效果的文字，该类的代码如下：

```
public class SolidTextApplet extends Applet {
    public void paint(Graphics g) {                //重写 paint()方法
        String value = "立体文字效果";
        int x = 10;                                //文本位置的横坐标
        int y = 120;                               //文本位置的纵坐标
        Font font = new Font("宋体", Font.BOLD, 72);  //创建字体对象
        g.setFont(font);                           //设置字体
        //这里省略了关键技术中给出的代码
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 SolidTextApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.SolidTextApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 561：改变立体文字的绘制方向。

本实例采用从左上向右下的方向绘制立体效果文字，如果需要改变立体文字的绘制方向，只需要在循环语句中调整改变 x 值和 y 值的两条语句即可，例如，要采用从右下向左上的方向绘制立体效果的文字，可以使用如下代码：

```
x -= 1;                                            //调整绘制点的横坐标值
y -= 1;                                            //调整绘制点的纵坐标值
```

实例 562　Applet 中绘制阴影效果的文字

初级

趣味指数：★★★★

光盘\MR\562

实例说明

本实例实现了在 Applet 中绘制阴影效果的文字。运行程序，将在浏览器窗口中显示阴影效果的文字，如图 21.17 所示。

图 21.17　在 Applet 中绘制的阴影效果文字

关键技术

本实例主要是通过在 Applet 中绘制两组相同的文字，并对绘制的文字进行偏移，使下层的文字以灰色显示，上层的文字以黑色显示，从而实现了阴影效果文字的绘制，实现代码如下：

```
g.setColor(Color.GRAY);                    //设置颜色为灰色
g.drawString(value, x, y);                 //绘制文本
x += 3;                                    //调整绘制点的横坐标值
y += 3;                                    //调整绘制点的纵坐标值
g.setColor(Color.BLACK);                   //设置颜色为黑色
g.drawString(value, x, y);                 //绘制文本
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 ShadowTextApplet 小应用程序，用于在其中绘制阴影效果的文字，该类的代码如下：

```
public class ShadowTextApplet extends Applet {
    public void paint(Graphics g) {                          //重写 paint()方法
        String value = "阴影文字";
        int x = 10;                                          //文本位置的横坐标
        int y = 120;                                         //文本位置的纵坐标
        Font font = new Font("华文行楷", Font.BOLD, 80);      //创建字体对象
        g.setFont(font);                                     //设置字体
        //省略了关键技术中给出的代码
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 ShadowTextApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.ShadowTextApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 562：改变阴影文字的阴影效果。

本实例绘制的阴影文字，阴影与文字之间的偏移值在 x 和 y 方向上都是 3，阴影效果可能不是特别明显，为此，可以通过调整 x 和 y 的值使阴影更加明显，例如，将阴影与文字之间的偏移值在 x 和 y 方向上都改为 5，可以使用如下代码：

```
x += 5;                                    //调整绘制点的横坐标值
y += 5;                                    //调整绘制点的纵坐标值
```

实例 563　Applet 中绘制倾斜效果的文字

初级

实例位置：光盘\MR\563

趣味指数：★★★★

实例说明

在 Applet 中绘制特效文字时，可以将文字绘制成倾斜效果。本实例实现了绘制倾斜效果的文字。运行程序，将在浏览器窗口中显示倾斜效果的文字，如图 21.18 所示。

关键技术

本实例主要是通过 Graphics2D 类的 shear()方法实现了在 Applet 中绘制倾斜效果的文字，实现代码如下：

```
g2.shear(0.1, -0.4);                                    //倾斜画布
g2.drawString(value, x, y);                             //绘制文本
```

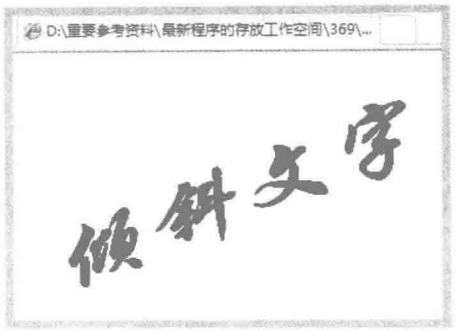


图 21.18　在 Applet 中绘制的倾斜效果文字

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 ShearTextApplet 小应用程序，用于在其中绘制倾斜效果的文字，该类的代码如下：

```
public class ShearTextApplet extends Applet {
    public void paint(Graphics g) {                     //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                 //转换为 Graphics2D 类型
        String value = "倾斜文字";                       //绘制的文本
        int x = 10;                                     //文本位置的横坐标
        int y = 190;                                    //文本位置的纵坐标
        Font font = new Font("华文行楷", Font.BOLD, 72); //创建字体对象
        g2.setFont(font);                               //设置字体
        g2.shear(0.1, -0.4);                            //倾斜画布
        g2.drawString(value, x, y);                     //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 ShearTextApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.ShearTextApplet.class" width ="380" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 563：倾斜文字与斜体字是完全不同的。

在绘制倾斜文字时，不要将倾斜文字与斜体字混为一谈，倾斜文字是对文字扭曲使其达到倾斜的效果；而斜体字则是一种字体样式，两者是完全不同的。

实例 564　Applet 中绘制渐变效果的文字

初级

光盘位置：光盘\MR\564

趣味指数：★★★★★

实例说明

在 Applet 中，除了可以绘制单一颜色的文字，还可以绘制渐变效果的文字。本实例实现了绘制渐变效果文

字的功能。运行程序，将在浏览器窗口中显示渐变效果的文字，如图 21.19 所示。

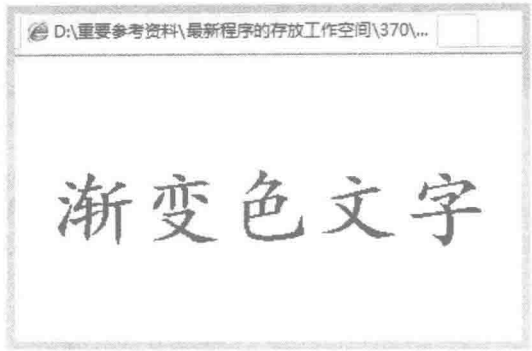


图 21.19　在 Applet 中绘制的渐变效果文字

关键技术

本实例主要是通过 GradientPaint 类创建渐变色对象，并使用 Graphics2D 类的 setPaint()方法为绘图上下文设置渐变颜色，从而实现了在 Applet 中绘制渐变效果的文字，实现代码如下：

```
GradientPaint paint = new GradientPaint(20, 20, Color.BLUE, 100, 120, Color.RED, true);        //创建循环渐变的 GradientPaint 对象
g2.setPaint(paint);                                                                            //设置渐变
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 GradientTextApplet 小应用程序，用于在其中绘制渐变效果的文字，该类的代码如下：

```
public class GradientTextApplet extends Applet {
    public void paint(Graphics g) {                                                            //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                                                        //转换为 Graphics2D 类型
        String value = "渐变色文字";                                                            //绘制的文本
        int x = 15;                                                                            //文本位置的横坐标
        int y = 120;                                                                           //文本位置的纵坐标
        Font font = new Font("楷体", Font.BOLD, 60);                                           //创建字体对象
        g2.setFont(font);                                                                      //设置字体
        GradientPaint paint = new GradientPaint(20, 20, Color.BLUE, 100, 120, Color.RED, true);//创建循环渐变的 GradientPaint 对象
        g2.setPaint(paint);                                                                    //设置渐变
        g2.drawString(value, x, y);                                                            //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 GradientTextApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.GradientTextApplet.class" width ="380" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 564：实现水平方向的渐变。

在使用 GradientPaint 类创建对象时，如果希望颜色在水平方向上渐变，可以将用户空间中第一个点和第二个点的纵坐标设置为相同的值，而横坐标设置为不同的值，例如：

```
GradientPaint paint = new GradientPaint(20, 20, Color.BLUE, 100, 20, Color.RED, true);        //创建在水平方向上循环渐变的 GradientPaint 对象
```

实例 565 Applet 中绘制会变色的文字

光盘位置：光盘\MR\565

初级

趣味指数：★★★★

实例说明

在前面的几个实例中讲解的 Applet，绘制的文本都是静态的，本实例实现了在 Applet 中绘制会变色的文字动画。运行程序，将在浏览器窗口中显示颜色随机变化的文字，效果如图 21.20 所示。

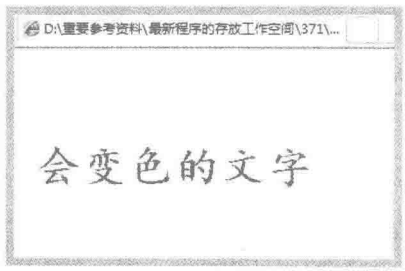


图 21.20　在 Applet 中绘制会变色文字的效果

关键技术

本实例主要是通过 Random 类创建随机数对象，然后通过该对象调用 nextInt()方法获得指定范围的随机整数，并通过线程实现了会变色的文字特效，其中线程的 run()方法的代码如下：

```
public void run() {
    Random random = new Random();                    //创建随机数对象
    while (true) {
        int R = random.nextInt(256);                 //随机产生颜色的 R 值
        int G = random.nextInt(256);                 //随机产生颜色的 G 值
        int B = random.nextInt(256);                 //随机产生颜色的 B 值
        color = new Color(R, G, B);                  //创建颜色对象
        repaint();                                   //调用 paint()方法
        try {
            Thread.sleep(300);                       //休眠 300 毫秒
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 ChangeTextColorApplet 小应用程序，用于在其中绘制颜色随机变化的文字，该类的代码如下：

```
public class ChangeTextColorApplet extends Applet implements Runnable {
    Color color = new Color(0, 0, 255);              //创建颜色对象
    public void init() {
        Thread thread = new Thread(this);            //创建线程对象
        thread.start();                              //启动线程对象
    }
    public void paint(Graphics g) {                  //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;              //转换为 Graphics2D 类型
        String value = "会变色的文字";                 //绘制的文本
        int x = 10;                                  //文本位置的横坐标
        int y = 110;                                 //文本位置的纵坐标
```

```
            Font font = new Font("楷体", Font.BOLD, 40);      //创建字体对象
            g2.setFont(font);                                 //设置字体
            g2.setColor(color);                               //设置颜色
            g2.drawString(value, x, y);                       //绘制文本
        }
        public void run() {
            //省略的代码请参阅本实例的关键技术部分
        }
    }
```

（3）在项目的 src 文件夹中创建一个名为 ChangeTextColorApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.ChangeTextColorApplet.class" width ="380" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 565：调整文字颜色变化的速度。

在使用本实例时，如果对颜色变化的速度不满意，可以调整 run()方法中的语句 Thread.sleep(300);的实参，如果觉得颜色变化的速度快，可以将实参改为大于 300 的值；如果觉得颜色变化的速度慢，可以将实参改为小于 300 的值。

实例 566　Applet 中绘制顺时针旋转的文字

初级

光盘位置：光盘\MR\566

趣味指数：★★★★

实例说明

本实例将用户在文本框中输入的内容以顺时针方向在 Applet 中进行旋转绘制。运行程序，将在浏览器窗口中看到顺时针旋转的文字，效果如图 21.21 所示。

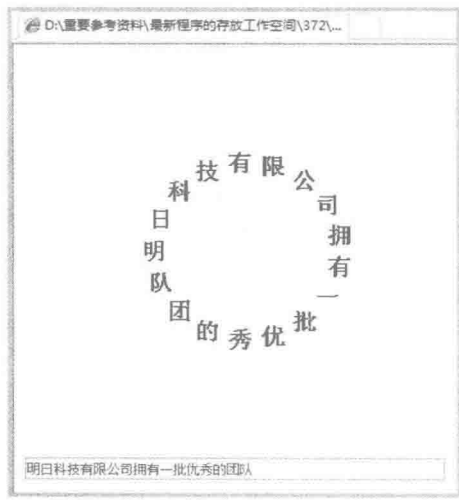


图 21.21　在 Applet 中绘制顺时针旋转的文字

关键技术

本实例主要是通过确定圆的半径，然后用半径乘以角度的正弦值和余弦值，计算出文字绘制点的横、纵坐标，从而实现顺时针旋转文字的功能。代码如下：

```
char[] array = text.toCharArray();                                  //将文本转换为字符数组
int len = array.length * 5;                                         //定义圆的半径，同时可以调整文字的距离
Font font = new Font("宋体", Font.BOLD, 22);                        //创建字体
g2.setFont(font);                                                   //设置字体
double angle = 0;                                                   //定义初始角度
for (int i = 0; i < array.length; i++) {                            //遍历字符串中的字符
    if (i == 0) {
        g2.setColor(Color.BLUE);                                    //第一个字符用蓝色
    } else {
        g2.setColor(Color.BLACK);                                   //其他字符用黑色
    }
    int x = (int) (len * Math.sin(Math.toRadians(angle + 270)));    //计算每个文字的横坐标位置
    int y = (int) (len * Math.cos(Math.toRadians(angle + 270)));    //计算每个文字的纵坐标位置
    g2.drawString(array[i] + "", width / 2 + x, height / 2 - y);    //绘制字符
    angle = angle + 360d / array.length;                            //改变角度
}
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 JApplet 类的 ClockwiseTextApplet 小应用程序，该类的代码如下：

```
public class ClockwiseTextApplet extends JApplet {
    private JTextField textField;
    ClockwiseTextPanel clockwiseTextPanel = new ClockwiseTextPanel();   //创建面板类的实例
    public void init() {
        setLayout(new BorderLayout());
        add(clockwiseTextPanel);                                        //将面板类的实例添加到窗体容器中
        textField = new JTextField();
        textField.addCaretListener(new CaretListener() {
            public void caretUpdate(CaretEvent arg0) {
                String text = textField.getText();                      //获取文本框字符串
                clockwiseTextPanel.setText(text);                       //为面板中的 text 变量赋值
            }
        });
        getContentPane().add(textField, BorderLayout.SOUTH);
    }
    //省略了内容类 ClockwiseTextPanel 的代码
}
```

（3）创建 ClockwiseTextApplet 类的内部面板类 ClockwiseTextPanel，用于绘制顺时针旋转的文字，该内部类的代码如下：

```
class ClockwiseTextPanel extends JPanel {                           //创建内部面板类
    private String text;
    public ClockwiseTextPanel() {
        setOpaque(false);                                           //设置面板为透明
        setLayout(null);                                            //设置为绝对布局
    }
    public String getText() {
        return text;                                                //获得成员变量的值
    }
    public void setText(String text) {
        this.text = text;                                           //为成员变量赋值
        repaint();                                                  //调整 paint()方法
    }
    public void paint(Graphics g) {                                 //重写 paint()方法
        Graphics2D g2 = (Graphics2D) g;                             //获得 Graphics2D 的实例
        int width = getWidth();                                     //获得面板的宽度
        int height = getHeight();                                   //获得面板的高度
        if (text != null) {
            //省略了关键技术中给出的代码
        }
    }
}
```

（4）在项目的 src 文件夹中创建一个名为 ClockwiseTextApplet.html 的文件，用于在浏览器中运行 Applet

小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.ClockwiseTextApplet.class" width ="380" height = "400">
</applet>
</html>
```

秘笈心法

心法领悟 566：改为逆时针旋转文字。

本实例实现了顺时针旋转文字的功能，如果希望逆时针旋转文字，可以对内部类中的如下代码进行修改：

```
int y = (int) (len * Math.cos(Math.toRadians(angle + 270)));    //计算每个文字的纵坐标位置
```

更改为以下代码可实现逆时针旋转文字：

```
int y = (int) (len * Math.cos(Math.toRadians(angle - 270)));    //计算每个文字的纵坐标位置
```

实例 567　Applet 中动态绘制文本

光盘位置：光盘\MR\567　　初级　　趣味指数：★★★★

实例说明

本实例实现了在小应用程序中动态绘制文本的功能，即在程序运行后，将字符串中的字符逐个顺次绘制到小应用程序中。运行程序，将在浏览器窗口中看到动态绘制的文本，效果如图 21.22 所示。

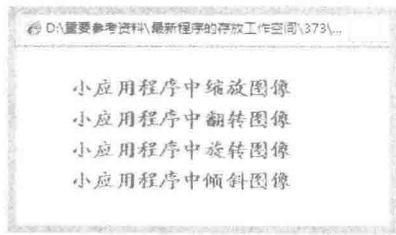


图 21.22　在 Applet 中动态绘制的文本

关键技术

本实例主要是通过 String 类的 substring()方法截取字符串中的单个字符，然后判断是否为换行符，如果不是，就调整字符的横坐标值，纵坐标不变；如果是，就同时调整 x 和 y 坐标值，使其到下一行合适的位置继续绘制字符，实现代码如下：

```
for (int i = 0; i < textStrings.length(); i++) {
    Thread.sleep(400);                          //当前线程休眠 400 毫秒
    value = textStrings.substring(i, i + 1);    //截取字符串中的一个字符
    if (value.equals("\n")) {                   //是换行符
        x = 20;                                 //下一行起始点的 x 坐标
        y += 30;                                //下一行文本的 y 坐标
    } else {                                    //不是回车或换行符
        x += 20;                                //当前行下一个字的 x 坐标
    }
    repaint();                                  //调用 repaint()方法
}
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DynamicDrawTextApplet 小应用程序，用于在其中动态绘制文本，该类的代码如下：

```
public class DynamicDrawTextApplet extends Applet implements Runnable {
    int x = 20;                                          //起始点的 x 坐标
    int y = 30;                                          //起始点的 y 坐标
    //需要动态绘制的字符串
    String textStrings = "小应用程序中缩放图像\n 小应用程序中翻转图像\n 小应用程序中旋转图像\n 小应用程序中倾斜图像";
    String value = "";                                   //存储字符串中的单个字符
    public void init() {
        Thread thread = new Thread(this);                //创建线程对象
        thread.start();                                  //启动线程对象
    }
    public void paint(Graphics g) {
        Font font = new Font("华文楷体", Font.BOLD, 20);  //创建字体对象
        g.setFont(font);                                 //指定字体
        g.setColor(Color.RED);                           //指定颜色
        g.drawString(value, x, y);                       //绘制文本
    }
    public void update(Graphics g) {                     //重写 update()方法，防止无法显示绘制的所有内容
        paint(g);                                        //调用 paint()方法
    }
    public void run() {
        try {
            for (int i = 0; i < textStrings.length(); i++) {
                Thread.sleep(400);                       //当前线程休眠 400 毫秒
                value = textStrings.substring(i, i + 1); //截取字符串中的一个字符
                if (value.equals("\n")) {                //是换行符
                    x = 20;                              //下一行起始点的 x 坐标
                    y += 30;                             //下一行文本的 y 坐标
                } else {                                 //不是回车或换行符
                    x += 20;                             //当前行下一个字的 x 坐标
                }
                repaint();                               //调用 repaint()方法
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DynamicDrawTextApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet code = "com.zzk.DynamicDrawTextApplet.class" width ="380" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 567：防止无法显示所有动态绘制的文本（即只显示当前绘制的字符，不显示其他字符）。

在小应用程序中动态绘制文本时，需要重写 update()方法，并在该方法中调用 paint()方法，否则会无法显示所有动态绘制的文本，而只显示当前绘制的字符。

21.4 Applet 中的图形处理

实例 568 Applet 绘制直线　　初级

光盘位置：光盘\MR\568　　趣味指数：★★

实例说明

前面实例讲解了如何在 Applet 中绘制文本，从本实例开始讲解如何在 Applet 中绘制图形，如直线、矩形、

圆形等，本实例讲解如何在 Applet 中绘制直线。运行程序，将在浏览器窗口中看到绘制的直线，效果如图 21.23 所示。

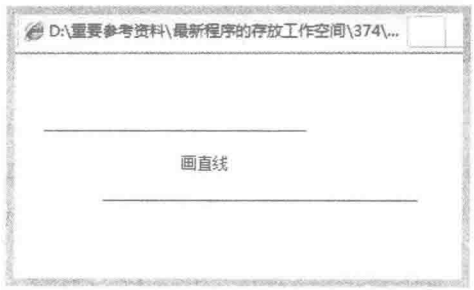


图 21.23　在 Applet 中绘制的直线

关键技术

本实例主要是通过 Graphics 类的 drawLine()方法实现了在 Applet 中绘制直线的功能，在 Applet 中绘制直线需要在 paint()方法中进行，实现代码如下：

```
g.drawLine(10, 100, 80, 100);                    //绘制直线
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawLineApplet 小应用程序，用于在其中绘制直线，该类的代码如下：

```
public class DrawLineApplet extends Applet {
    public void paint(Graphics g) {
        String value = "画直线";
        int x = 215;                             //直线的横坐标（右）
        int y = 45;                              //直线的纵坐标（右）
        int x1 = 15;                             //直线的横坐标（左）
        int y1 = 45;                             //直线的纵坐标（左）
        int x2 = 300;                            //直线的横坐标（右）
        int y2 = 100;                            //直线的纵坐标（右）
        int x3 = 60;                             //直线的横坐标（左）
        int y3 = 100;                            //直线的纵坐标（左）
        g.setColor(Color.blue);                  //设置颜色为蓝色
        g.drawLine(x, y, x1, y1);                //绘制直线
        g.drawLine(x2, y2, x3, y3);              //绘制直线
        g.drawString(value, 120, 75);            //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawLineApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawLineApplet.class" width ="800" height = "500">
</applet>
</html>
```

秘笈心法

心法领悟 568：绘制竖线和斜线。

在绘制直线时，如果第一个点的 x 坐标值与第二个点的 x 坐标值相同，则绘制的是竖线；如果第一个点的横、纵坐标值与第二个点的横、纵坐标值均不相同，则绘制的是斜线。

实例 569 Applet 绘制矩形

光盘位置：光盘\MR\569

中级

趣味指数：★★★

实例说明

在日常生产和生活中经常使用矩形，例如教学用的白板、房屋的窗户等，本实例讲解如何在 Applet 中绘制矩形。运行程序，将在浏览器窗口中看到所绘制的矩形，效果如图 21.24 所示。

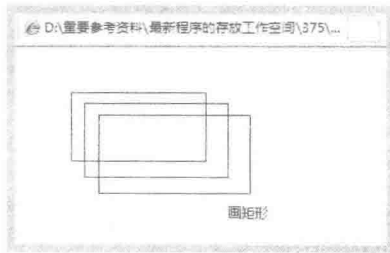


图 21.24 在 Applet 中绘制的矩形

关键技术

本实例主要是通过 Graphics 类的 drawRect()方法实现了在 Applet 中绘制矩形的功能，在 Applet 中绘制矩形也需要在 paint()方法中进行，实现代码如下：

```
g.drawRect(50, 60, 200, 160);                    //绘制矩形
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawRectApplet 小应用程序，用于在其中绘制矩形，该类的代码如下：

```
public class DrawRectApplet extends Applet {
    public void paint(Graphics g) {
        String value = "画矩形";
        int x = 42;                              //矩形的横坐标
        int y = 27;                              //矩形的纵坐标
        int width = 122;                         //矩形的宽度
        int height = 64;                         //矩形的高度
        g.setColor(Color.BLUE);                  //设置颜色为蓝色
        g.drawRect(x, y, width, height);         //绘制矩形
        //省略了绘制其他矩形的代码
        g.drawString(value, 185, 143);           //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawRectApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawRectApplet.class" width="500" height="220">
</applet>
</html>
```

秘笈心法

心法领悟 569：使用 Graphics2D 绘制矩形对象。

除了可以使用 Graphics 类的 drawRect()方法绘制矩形外，还可以使用 Graphics2D 类的 draw()方法绘制矩形对象，方法是先使用 Rectangle2D.Float 类创建矩形对象，然后将其作为参数传递给 draw()方法，从而实现使用 Graphics2D 类绘制矩形对象的功能。

实例 570　Applet 绘制圆角矩形

初级

趣味指数：★★★★

实例说明

圆角矩形在生活中也是随处可见的，例如桌面、餐盘表面等，它们的 4 个角就是圆角的，可以避免刮伤。本实例讲解如何在 Applet 中绘制圆角矩形。运行程序，将在浏览器窗口中看到所绘制的圆角矩形，效果如图 21.25 所示。

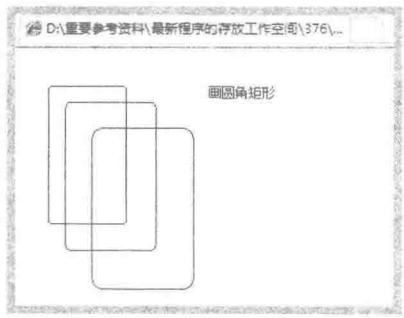


图 21.25　在 Applet 中绘制的圆角矩形

关键技术

本实例主要是通过 Graphics 类的 drawRoundRect()方法实现了在 Applet 中绘制圆角矩形的功能。在 Applet 中绘制圆角矩形的代码如下：

```
g.drawRoundRect(20, 40, 200, 180, 20,15);      //绘制图形
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawRoundRectApplet 小应用程序，用于在其中绘制圆角矩形，该类的代码如下：

```
public class DrawRoundRectApplet extends Applet {
    public void paint(Graphics g) {
        String value = "画圆角矩形";
        int x = 20;                                    //圆角矩形位置的横坐标
        int y = 20;                                    //圆角矩形位置的纵坐标
        int width = 70;                                //圆角矩形宽度
        int height = 129;                              //圆角矩形高度
        int xr = 5;                                    //圆角矩形圆角的水平弧度
        int yr = 7;                                    //圆角矩形圆角的垂直弧度
        g.setColor(Color.blue);                        //设置颜色
        g.drawRoundRect(x, y, width, height, xr, yr);  //绘制图形
        //省略了绘制其他图形的代码
        g.drawString(value, 165, 30);                  //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawRoundRectApplet.html 的文件，用于在浏览器中运行 Applet

小应用程序，该.html 文件的代码如下：

```
<html>
<applet  code = "com.zzk.DrawRoundRectApplet.class" width ="500" height = "320">
</applet>
</html>
```

秘笈心法

心法领悟 570：绘制有填充色的圆角矩形。

本实例使用 Graphics 类的 drawRoundRect()方法绘制了圆角矩形，如果需要绘制有填充色的圆角矩形，可以使用 Graphics 类的 fillRoundRect()方法实现，将本实例的 drawRoundRect()方法改为 fillRoundRect()方法即可。

实例 571　Applet 绘制椭圆

初级

实例位置：光盘\MR\571

趣味指数：★★★★

实例说明

在生活中经常看到椭圆，例如某些盘子、梳妆镜的形状、鸡蛋等。本实例讲解如何在 Applet 中绘制椭圆。运行程序，将在浏览器窗口中看到所绘制的椭圆，效果如图 21.26 所示。

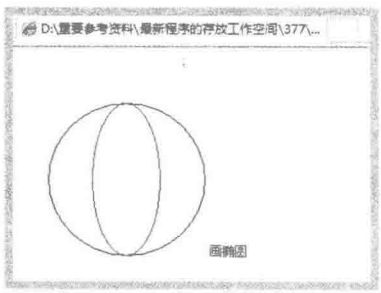


图 21.26　在 Applet 中绘制的椭圆

关键技术

本实例主要是通过 Graphics 类的 drawOval()方法实现了在 Applet 中绘制椭圆的功能。绘制椭圆的实现代码如下：

```
g.drawOval(50, 80, 150, 120);                    //绘制椭圆
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawOvalApplet 小应用程序，用于在其中绘制椭圆，该类的代码如下：

```
public class DrawOvalApplet extends Applet {
    public void paint(Graphics g) {
        String value = "画椭圆";
        int x = 25;                                  //椭圆位置横坐标
        int y = 40;                                  //椭圆位置纵坐标
        int xr = 150;                                //椭圆的横坐标半径
        int yr = 150;                                //椭圆的纵坐标半径
        g.drawOval(x, y, xr, yr);                    //绘制椭圆
        //省略了绘制另一个椭圆的代码
        g.drawString(value, 180, 190);               //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawOvalApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawOvalApplet.class" width ="500" height = "320">
</applet>
</html>
```

秘笈心法

心法领悟 571：使用 Graphics2D 绘制椭圆对象。

除了可以使用 Graphics 类的 drawOval()方法绘制椭圆外，还可以使用 Graphics2D 类的 draw()方法绘制椭圆对象，方法是先使用 Ellipse2D.Float 类创建椭圆对象，然后将其作为参数传递给 draw()方法，从而实现使用 Graphics2D 类绘制椭圆对象的功能。

实例 572　Applet 绘制圆弧

初级

趣味指数：★★★★★

实例说明

本实例讲解如何在 Applet 中绘制圆弧。运行程序，可以在浏览器窗口中看到所绘制的圆弧，效果如图 21.27 所示。

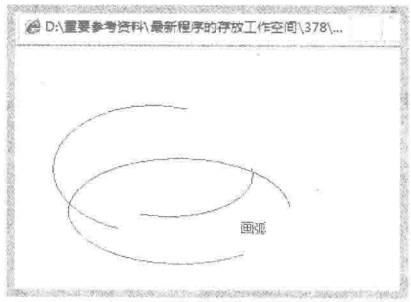


图 21.27　在 Applet 中绘制的圆弧

关键技术

本实例主要是通过 Graphics 类的 drawArc()方法实现了在 Applet 中绘制圆弧的功能。绘制圆弧的实现代码如下：

```
g.drawArc(20, 50, 180, 120, 30, 120);                    //绘制圆弧
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawArcApplet 小应用程序，用于在其中绘制圆弧，该类的代码如下：

```
public class DrawArcApplet extends Applet {
    public void paint(Graphics g) {
        String value = "画弧";
        int x = 35;                                      //弧位置的横坐标
        int y = 65;                                      //弧位置的纵坐标
        int l = 150;                                     //弧的长度
        int width = 80;                                  //弧的宽度
        int startAngle = 10;                             //弧的起始角度
```

```
            int endAngle = -120;                                    //终止画弧前扫过的角度
            g.setColor(Color.red);                                  //设置颜色
            g.drawArc(x + 20, y, l, width, startAngle, endAngle);   //绘制弧
            //省略了其他绘制圆弧的代码
            g.drawString(value, 195, 160);
        }
    }
```

（3）在项目的 src 文件夹中创建一个名为 DrawArcApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawArcApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 572：使用 Graphics2D 类绘制有填充色的圆弧对象。

在绘制圆弧时，可以使用 Graphics 类的 fillArc()方法绘制有填充色的圆弧，如果需要绘制有填充色的圆弧对象，可以使用 Arc2D.Float 类创建圆弧对象，然后将其作为 Graphics2D 类 fill()方法的参数，即可实现有填充色圆弧对象的绘制。

实例 573　Applet 绘制折线

实例位置：光盘\MR\573　　初级　　趣味指数：★★★★

实例说明

本实例讲解如何在 Applet 中绘制折线。运行程序，可以在浏览器窗口中看到所绘制的折线，效果如图 21.28 所示。

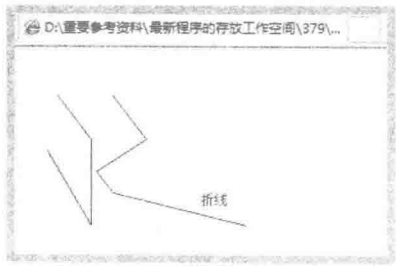


图 21.28　在 Applet 中绘制的折线

关键技术

本实例主要是通过 Graphics 类的 drawPolyline()方法实现了在 Applet 中绘制折线的功能。该方法的定义如下：

```
public abstract void drawPolyline(int[] xPoints, int[] yPoints, int nPoints)
```

参数说明

❶ xPoints：折线绘制点的 x 坐标数组。

❷ yPoints：折线绘制点的 y 坐标数组。

❸ nPoints：折线中点的总数。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawPolylineApplet

小应用程序，用于在其中绘制折线，该类的代码如下：

```
public class DrawPolylineApplet extends Applet {
    public void paint(Graphics g) {
        String value = "折线";
        int[] x = { 30, 60, 60, 20 };                    //声明表示折线横坐标的数组
        int[] y = { 30, 70, 150, 80 };                   //声明表示折线纵坐标的数组
        int num1 = x.length;                             //提取 x，y 坐标对数组的长度
        g.setColor(Color.blue);                          //设置颜色
        g.drawPolyline(x, y, num1);                      //绘制折线
        int[] x0 = { 80, 110, 65, 80, 200 };             //声明表示折线横坐标的数组
        int[] y0 = { 30, 70, 100, 120, 150 };            //声明表示折线纵坐标的数组
        int num2 = x0.length;                            //提取 x0，y0 坐标对数组的长度
        g.drawPolyline(x0, y0, num2);                    //绘制折线
        g.drawString(value, 160, 130);                   //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawPolylineApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawPolylineApplet.class" width ="500" height = "320">
</applet>
</html>
```

秘笈心法

心法领悟 573：防止绘制的折线在一条直线上。

在绘制折线时，折线绘制点的 x 坐标数组或 y 坐标数组中，如果有 3 个或 3 个以上连续的坐标值相同，就会出现折线绘制到一条直线上的情况，为了防止此情况，应使 3 个或 3 个以上连续的坐标值不同。

实例 574　Applet 绘制多角形

光盘位置：光盘\MR\574　　初级　　趣味指数：★★★

实例说明

在前面的实例中讲解了如何在 Applet 中绘制直线、矩形、椭圆等图形，接下来讲解如何在 Applet 中绘制多角形。运行程序，可以在浏览器窗口中看到所绘制的多角形，效果如图 21.29 所示。

图 21.29　在 Applet 中绘制的多角形

关键技术

本实例主要是通过 Graphics 类的 drawPolypon()方法实现了在 Applet 中绘制多角形的功能，例如可以使用如下代码绘制五角形：

```
int x[] = { 60, 103, 170, 150, 120 };          //声明多角形的横坐标数组
int y[] = { 80, 180, 140, 80, 120 };           //声明多角形的纵坐标数组
g.drawPolygon(x, y, 5);                        //绘制五角形
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawPolygonApplet 小应用程序，用于在其中绘制多角形，该类的代码如下：

```
public class DrawPolygonApplet extends Applet {
    public void paint(Graphics g) {
        String value = "绘制多角形";
        int x[] = { 60, 103, 170, 150, 120 };          //声明多角形的横坐标数组
        int y[] = { 80, 180, 140, 80, 120 };           //声明多角形的纵坐标数组
        int num = x.length;                            //取得多角形 x，y 坐标对数组的长度
        g.setColor(Color.blue);                        //设置颜色
        g.drawPolygon(x, y, num);                      //绘制多角形
        g.drawString(value, 120, 70);                  //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawPolygonApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawPolygonApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 574：绘制多角形时需要注意坐标数组和顶点个数。

绘制多角形时，x 与 y 坐标数组的元素个数必须相同，或者顶点个数小于等于元素个数最少的那个数组元素个数，否则绘制多角形时程序会出错。

实例 575　Applet 绘制图片

光盘位置：光盘\MR\575　　初级　　趣味指数：★★★★★

实例说明

在前面的实例中讲解了如何在 Applet 中绘制文本和图形，接下来讲解如何在 Applet 中绘制图片。运行程序，可以在浏览器窗口中看到所绘制的图片，效果如图 21.30 所示。

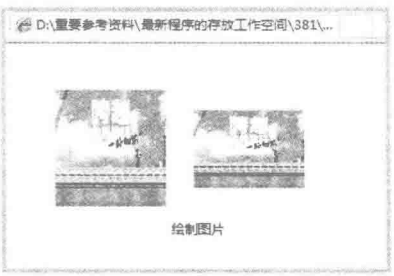


图 21.30　在 Applet 中绘制的图片

关键技术

本实例主要是通过 Graphics 类的 drawImage()方法实现了在 Applet 中绘制图片的功能。在 Applet 中绘制图

片的实现代码如下：

```
Image image = getImage(getCodeBase(), "com/zzk/PD2.jpg");                    //获得图片信息
g.drawImage(image, 50, 30, 200, 160, this);                                  //绘制图像
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawImageApplet 小应用程序，用于在其中绘制图片，该类的代码如下：

```
public class DrawImageApplet extends Applet {
    public void paint(Graphics g) {                                          //初始化方法
        String value = "绘制图片";
        Image image = null;                                                  //声明图像对象
        image = getImage(getCodeBase(), "com/zzk/PD2.jpg");                  //获得图片信息
        int x = 10;                                                          //图像位置的横坐标
        int y = 20;                                                          //图像位置的纵坐标
        int width = image.getWidth(this);                                    //获得图像的宽度
        int height = image.getHeight(this);                                  //获取图像的高度
        g.drawImage(image, x + 150, y + 30, width / 5, height / 5, this);    //绘制图像
        g.drawImage(image, x + 25, y + 10, (int) (width * 0.2), (int) (height * 0.3), this);  //绘制图像
        g.drawString(value, 140, 170);                                       //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawImageApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawImageApplet.class" width ="1200" height = "1000">
</applet>
</html>
```

秘笈心法

心法领悟 575：通过相对路径获得图片对象。

在 Applet 小应用程序中，为了使用相对路径获得图片对象，通常将图片放到项目的 class 路径或 class 的子路径中，这样可以通过 Applet 类的 getCodeBase()方法获得相对路径，从而可以通过 Applet 类的 getImage()方法以相对路径获得图片对象，有助于程序的移植。

实例 576　Applet 中的图形加运算

光盘位置：光盘\MR\576　　中级　　趣味指数：★★★★★

实例说明

在前面的实例中讲解了图形的绘制，接下来讲解如何在 Applet 中对图形进行加运算。运行程序，可以在浏览器窗口中看到图形加运算后的效果，如图 21.31 所示。

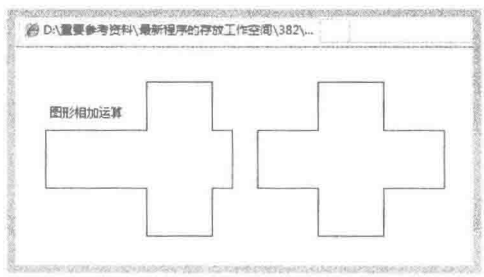


图 21.31　Applet 中图形加运算的效果

关键技术

本实例主要是通过 Area 区域封装图形对象，然后使用 Area 类的 add()方法对两个 Area 区域图形进行加运算。进行图形加运算的实现代码如下：

```
Area area1 = new Area(rect1);                    //创建区域矩形
Area area2 = new Area(rect2);                    //创建区域矩形
area1.add(area2);                                //两个区域进行加运算
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 AddOperationApplet 小应用程序，用于在其中绘制进行加运算后得到的图形，该类的代码如下：

```
public class AddOperationApplet extends Applet {
    public void paint(Graphics g) {
        String value = "图形相加运算";
        Graphics2D g2d = (Graphics2D) g;                                    //转换为可用的 Graphics2D 对象
        Rectangle2D.Float rect1 = new Rectangle2D.Float(20, 70, 185, 60);   //创建矩形对象
        Rectangle2D.Float rect2 = new Rectangle2D.Float(120, 20, 65, 160);  //创建矩形对象
        Area area1 = new Area(rect1);                                       //创建区域矩形
        Area area2 = new Area(rect2);                                       //创建区域矩形
        area1.add(area2);                                                   //两个区域进行相加
        g2d.draw(area1);                                                    //绘制相加后的区域矩形
        Rectangle2D.Float rect3 = new Rectangle2D.Float(230, 70, 185, 60);  //创建矩形对象
        Rectangle2D.Float rect4 = new Rectangle2D.Float(290, 20, 65, 160);  //创建矩形对象
        Area area3 = new Area(rect3);                                       //创建区域矩形
        Area area4 = new Area(rect4);                                       //创建区域矩形
        area3.add(area4);                                                   //两个区域进行加运算
        g2d.draw(area3);                                                    //绘制相加后的区域矩形
        g2d.drawString(value, 25, 56);                                      //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 AddOperationApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.AddOperationApplet.class" width ="1200" height = "1000">
</applet>
</html>
```

秘笈心法

心法领悟 576：如何使图形加运算的效果更明显。

在进行图形加运算时，为了使加运算的效果更明显，应该使用没有填充色的图形，如果使用有填充色的图形，会让用户产生是两个图形重叠到一起，还是进行了加运算的疑惑。

实例 577　Applet 中的图形减运算

初级

光盘位置：光盘\MR\577

趣味指数：★★★

实例说明

本实例讲解如何在 Applet 中对图形进行减运算。运行程序，可以在浏览器窗口中看到对图形进行减运算后的效果，如图 21.32 所示。

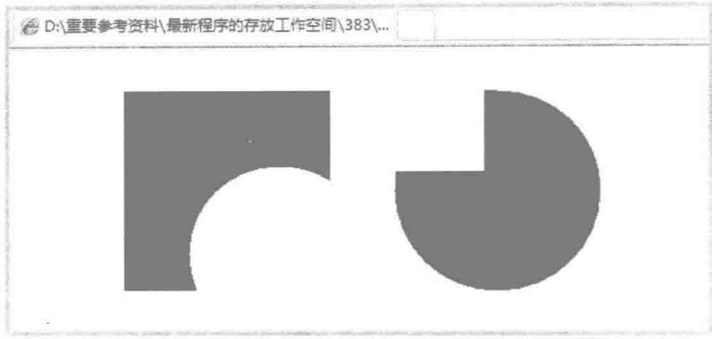


图 21.32　Applet 中图形减运算的效果

关键技术

本实例主要是通过 Area 区域封装图形对象，然后使用 Area 类的 subtract()方法对两个 Area 区域图形进行减运算。进行图形减运算的实现代码如下：

```
Area area1 = new Area(rect1);                    //创建区域矩形
Area area2 = new Area(ellipse1);                 //创建区域椭圆
area1.subtract(area2);                           //两个区域图形相减
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 SubtractOperationApplet 小应用程序，用于在其中绘制进行减运算后得到的图形，该类的代码如下：

```
public class SubtractOperationApplet extends Applet {
    public void paint(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;                                    //强制转换为 Graphics2D 对象
        Rectangle2D.Float rect1 = new Rectangle2D.Float(80, 20, 160, 160);  //创建矩形对象
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(130, 80, 140, 140);  //创建椭圆对象
        Area area1 = new Area(rect1);                                       //创建区域矩形
        Area area2 = new Area(ellipse1);                                    //创建区域椭圆
        area1.subtract(area2);                                              //两个区域图形相减
        g2d.fill(area1);                                                    //绘制相减后的区域图形
        Rectangle2D.Float rect2 = new Rectangle2D.Float(240, -35, 120, 120); //创建矩形对象
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(290, 20, 160, 160);  //创建椭圆对象
        Area area3 = new Area(rect2);                                       //创建矩形区域
        Area area4 = new Area(ellipse2);                                    //创建椭圆对象的区域图形
        area4.subtract(area3);                                              //两个区域图形相减
        g2d.fill(area4);                                                    //绘制相减后的区域图形
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 SubtractOperationApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.SubtractOperationApplet.class" width ="1200" height = "1000">
</applet>
</html>
```

秘笈心法

心法领悟 577：在填充矩形中抠一个圆形的洞。

为了在填充矩形的中心抠一个圆形的洞，可以分别创建适合的矩形区域对象和圆形区域对象，并使圆形区域在矩形区域的范围内，然后通过图形的减运算实现在填充矩形区域中抠一个圆形的洞。

实例 578 Applet 中的图形交运算

光盘位置：光盘\MR\578

初级

趣味指数：★★★

实例说明

本实例讲解如何在 Applet 中对图形进行交运算。运行程序，可以在浏览器窗口中看到对图形进行交运算后的效果，如图 21.33 所示。

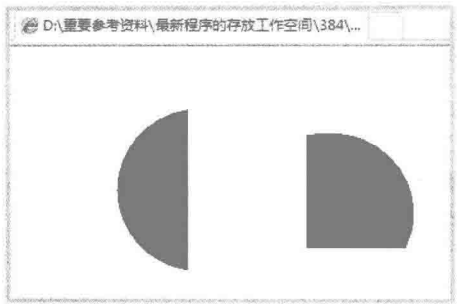


图 21.33 Applet 中图形交运算的效果

关键技术

本实例主要是通过 Area 区域封装图形对象，然后使用 Area 类的 intersect()方法对两个 Area 区域图形进行交运算。进行图形交运算的实现代码如下：

```
Area area1 = new Area(rect1);                    //创建矩形区域
Area area2 = new Area(ellipse1);                 //创建椭圆区域
area1.intersect(area2);                          //两个区域相交
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 IntersectOperationApplet 小应用程序，用于在其中绘制进行交运算后得到的图形，该类的代码如下：

```
public class IntersectOperationApplet extends Applet {
    public void paint(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;                                    //强制转换为 Graphics2D 对象
        Rectangle2D.Float rect1 = new Rectangle2D.Float(0, 40, 140, 140);   //创建矩形对象
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(80, 40, 140, 140);   //创建椭圆对象
        Area area1 = new Area(rect1);                                       //创建矩形区域
        Area area2 = new Area(ellipse1);                                    //创建椭圆区域
        area1.intersect(area2);                                             //两个区域相交
        g2d.fill(area1);                                                    //绘制相交后的区域图形
        Rectangle2D.Float rect2 = new Rectangle2D.Float(240, 0, 160, 160);  //创建矩形对象
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(190, 60, 140, 140);  //创建椭圆对象
        Area area3 = new Area(rect2);                                       //创建矩形区域
        Area area4 = new Area(ellipse2);                                    //创建椭圆区域
        area3.intersect(area4);                                             //两个区域相交
        g2d.fill(area3);                                                    //绘制相交后的区域图形
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 IntersectOperationApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.IntersectOperationApplet.class" width ="1200" height = "1000">
```

```
</applet>
</html>
```

秘笈心法

心法领悟 578：通过交运算绘制圆锥形。

要通过交运算绘制圆锥形，可以先创建一个矩形区域，然后再创建一个适合的圆形区域，使矩形区域的一角与圆形区域相交，这样对两个图形区域进行交运算后，就可以得到一个圆锥形。

实例 579　Applet 中的图形异或运算

光盘位置：光盘\MR\579　　初级　　趣味指数：★★★★

实例说明

本实例讲解如何在 Applet 中对图形进行异或运算。运行程序，可以在浏览器窗口中看到对图形进行异或运算后的效果，如图 21.34 所示。

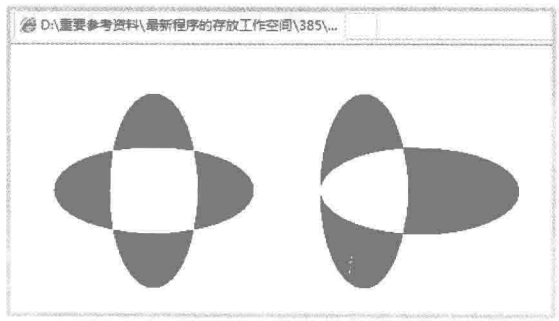


图 21.34　Applet 中图形异或运算的效果

关键技术

本实例主要是通过 Area 区域封装图形对象，然后使用 Area 类的 exclusiveOr()方法对两个 Area 区域图形进行异或运算。进行图形异或运算的实现代码如下：

```
Area area1 = new Area(ellipse1);                    //创建椭圆区域
Area area2 = new Area(ellipse2);                    //创建椭圆区域
area1.exclusiveOr(area2);                           //两个区域图形进行异或运算
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 ExclusiveOrOperationApplet 小应用程序，用于在其中绘制进行异或运算后得到的图形，该类的代码如下：

```
public class ExclusiveOrOperationApplet extends Applet {
    public void paint(Graphics g) {                 //重写 paint()方法
        Graphics2D g2d = (Graphics2D) g;            //强制转换为 Graphics2D 对象
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(30, 80, 180, 80);    //创建椭圆对象
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(80, 30, 80, 180);    //创建椭圆对象
        Area area1 = new Area(ellipse1);            //创建椭圆区域
        Area area2 = new Area(ellipse2);            //创建椭圆区域
        area1.exclusiveOr(area2);                   //两个区域图形进行异或运算
        g2d.fill(area1);                            //绘制异或运算后的区域图形
        Ellipse2D.Float ellipse3 = new Ellipse2D.Float(270, 80, 180, 80);   //创建椭圆对象
        Ellipse2D.Float ellipse4 = new Ellipse2D.Float(270, 30, 80, 180);   //创建椭圆对象
```

```
        Area area3 = new Area(ellipse3);                    //创建椭圆区域
        Area area4 = new Area(ellipse4);                    //创建椭圆区域
        area3.exclusiveOr(area4);                           //绘制异或运算后的区域图形
        g2d.fill(area3);                                    //绘制异或运算后的区域图形
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 ExclusiveOrOperationApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.ExclusiveOrOperationApplet.class" width ="1200" height = "1000">
</applet>
</html>
```

秘笈心法

心法领悟 579：灵活地应用加、减、交和异或等运算。

在实际应用中，可能需要绘制各种图形，如果实现起来比较困难，可以考虑是否能够通过加、减、交和异或等运算得到，如果能够灵活地应用这些运算，可以绘制出各种奇异的图形。

实例 580　Applet 中绘制纹理填充图形

初级

光盘位置：光盘\MR\580

趣味指数：★★★★

实例说明

在前面讲解的实例中，对文本和图形填充的都是单一的颜色或渐变颜色，本实例将在 Applet 中实现纹理填充图形的绘制。运行程序，可以在浏览器窗口中看到所绘制的纹理填充图形，效果如图 21.35 所示。

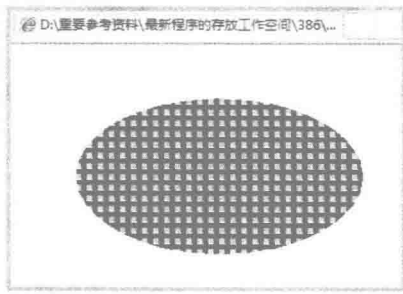


图 21.35　Applet 中绘制的纹理填充图形

关键技术

本实例主要是使用 BufferedImage 创建缓冲图像对象，并在其上绘制作为填充纹理的图形，然后使用 TexturePaint 类创建纹理填充对象，并将 TexturePaint 对象设置为 Graphics2D 对象 setPaint()方法的参数，再绘制图形就会以该纹理进行填充，实现代码如下：

```
BufferedImage img = new BufferedImage(50, 50, BufferedImage.TYPE_INT_RGB);   //创建缓冲对象
Graphics2D g = img.createGraphics();                                         //创建 Graphics2D 对象
g.setPaint(Color.yellow);                                                    //指定颜色
g.draw(new Rectangle(0, 0, 25, 25));                                         //绘制矩形
g.setPaint(new TexturePaint(img, new Rectangle(0, 0, 10, 10)));              //创建纹理并添加到 Graphics2D 中
g.fill(new Ellipse2D.Double(10, 10, 260, 145));                              //填充图形
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawGrainApplet 小应用程序，用于在其中绘制具有填充纹理的图形，该类的代码如下：

```
public class DrawGrainApplet extends Applet {
    private BufferedImage img;                                          //声明图像对象
    public void init() {                                                //初始化方法
        img = new BufferedImage(50, 50, BufferedImage.TYPE_INT_RGB);    //创建缓冲对象
        Graphics2D g = img.createGraphics();                            //创建 Graphics2D 对象
        g.setPaint(Color.yellow);                                       //指定颜色
        g.draw(new Rectangle(0, 0, 25, 25));                            //绘制矩形
        g.setPaint(Color.red);                                          //指定颜色
        g.fill(new Rectangle(25, 0, 25, 25));                           //填充矩形
        g.setPaint(Color.green);                                        //指定颜色
        g.fill(new Rectangle(0, 0, 25, 25));                            //填充矩形
    }
    public void paint(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;                                //强制转换为 Graphics2D 对象
        g2d.setPaint(new TexturePaint(img, new Rectangle(0, 0, 10, 10)));//创建纹理并把它加到 Graphics 中
        g2d.fill(new Ellipse2D.Double(10, 10, 260, 145));               //填充图形
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawGrainApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawGrainApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 580：为文本填充纹理。

为文本填充纹理，也需要先为绘图上下文设置填充纹理，这样当在绘图上下文中绘制文本时，文本的线条就会以该纹理进行填充，从而实现了为文本填充纹理的功能。

21.5　Applet 中的图像处理

实例 581　Applet 中缩放图像　　初级

光盘位置：光盘\MR\581　　趣味指数：★★★★

实例说明

在前面讲解的实例中，只是简单地实现了文本、图形和图像等内容的绘制，本实例将讲解如何在 Applet 中缩放图像。运行程序，可以在浏览器窗口中看到对图像进行缩放后的效果，如图 21.36 所示。

图 21.36　在 Applet 中缩放图像

关键技术

本实例主要是通过 AffineTransform 类的 getScaleInstance()方法创建缩放的 AffineTransform 对象，然后使用 Graphics2D 类的 drawImage()方法绘制图像时，将 AffineTransform 对象传递给 drawImage()方法，实现了缩放图像的功能，实现代码如下：

```
AffineTransform tr = AffineTransform.getScaleInstance(0.25, 0.25);          //创建变形以获得缩放对象
Graphics2D g2d = (Graphics2D) g;                                            //将 g 转换成一个可用的 Graphics2D 对象
g2d.drawImage(img, tr, this);                                               //绘制并缩放图像
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 ZoomImageApplet 小应用程序，用于在其中绘制缩放的图像，该类的代码如下：

```
public class ZoomImageApplet extends Applet {
    public void paint(Graphics g) {
        String value = "缩放图像";
        Image img = null;                                                   //声明图像对象
        img = getImage(getCodeBase(), "com/zzk/PD2.jpg");                   //获得图片信息
        AffineTransform tr = AffineTransform.getScaleInstance(0.25, 0.25);  //创建变形以获得缩放对象
        tr.translate(120, 100);                                             //设置对象平移
        AffineTransform tr2 = AffineTransform.getScaleInstance(0.15, 0.15); //创建变形以获得缩放对象
        tr2.translate(900, 950);                                            //设置对象平移
        Graphics2D g2d = (Graphics2D) g;                                    //将 g 转换成一个可用的 Graphics2D 对象
        g2d.drawImage(img, tr, this);                                       //绘制图像
        g2d.drawImage(img, tr2, this);                                      //绘制图像
        g2d.drawString(value, 60, 150);                                     //绘制文本
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 ZoomImageApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet  code = "com.zzk.ZoomImageApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 581：使用 AffineTransform 类的 getScaleInstance()方法。

在使用 AffineTransform 类的 getScaleInstance()方法获得变换对象时，如果传递给该方法的参数是小于 1 的数，则对原图像进行缩小；如果传递给该方法的参数是大于 1 的数，则对原图像进行放大，并且可以使一个参数大于 1，另一个参数小于 1，这样即可在一个方向放大图像，而在另一个方向缩小图像。

实例 582 Applet 中翻转图像

光盘位置：光盘\MR\582　　中级　　趣味指数：★★★

实例说明

本实例实现了在 Applet 中翻转图像的功能。运行程序，可以在浏览器窗口中看到图像翻转后的效果，如图 21.37 所示。

图 21.37　在 Applet 中翻转图像的效果

关键技术

本实例主要是通过 AffineTransform 类中有 6 个参数的构造方法实现了对图像进行翻转的功能，该构造方法的定义如下：

```
public AffineTransform(double m00, double m10, double m01, double m11, double m02, double m12)
```

参数说明

❶ m00：3×3 矩阵缩放元素的 x 坐标。
❷ m10：3×3 矩阵剪切元素的 y 坐标。
❸ m01：3×3 矩阵剪切元素的 x 坐标。
❹ m11：3×3 矩阵缩放元素的 y 坐标。
❺ m02：3×3 矩阵平移元素的 x 坐标。
❻ m12：3×3 矩阵平移元素的 y 坐标。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 TurnImageApplet 小应用程序，用于在其中绘制翻转的图像，该类的代码如下：

```
public class TurnImageApplet extends Applet {
    public void paint(Graphics g) {
        String value = "翻转图像";
        Image img = null;                                          //声明图像对象
        img = getImage(getCodeBase(), "com/zzk/PD4.jpg");          //获得图片信息
        int w = img.getWidth(this);                                //设置图像的宽度
        int h = img.getHeight(this);                               //设置图像的高度
        Graphics2D g2d = (Graphics2D) g;                           //将 g 转换为可以利用的 Graphics2D
        g2d.drawString(value, 100, 130);                           //绘制文本
        AffineTransform tr = new AffineTransform(-1, 0, 0, 1, w, 0);   //创建变换对象并横向翻转
        AffineTransform tr2 = new AffineTransform(1, 0, 0, -1, 0, h);  //创建变换对象并垂直翻转
        tr.translate(-20, 40);                                     //图像位置的平移
        tr2.translate(120, -40);                                   //图像位置的平移
        g2d.drawImage(img, tr, this);                              //绘制图像
        g2d.drawImage(img, tr2, this);                             //绘制图像
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 TurnImageApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.TurnImageApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 582：使用 AffineTransform 类水平与垂直翻转图像。

在使用 AffineTransform 类的构造方法 AffineTransform()翻转图像时，如果需要水平翻转，将 m00 设置为-1，并将 m11 设置为 1，m02 设置为水平偏移的距离，再将其他参数都设置为 0 即可；如果需要垂直翻转，将 m00 设置为 1，并将 m11 设置为-1，然后将 m12 设置为垂直偏移的距离，再将其他参数都设置为 0 即可。

实例 583　Applet 中旋转图像

光盘位置：光盘\MR\583　　初级　　趣味指数：★★★★

实例说明

本实例实现了在 Applet 中旋转图像的功能。运行程序，可以在浏览器窗口中看到图像旋转后的效果，如图 21.38 所示。

图 21.38　在 Applet 中旋转图像的效果

关键技术

本实例主要是通过 AffineTransform 类的 rotate()方法旋转绘图上下文，然后在绘图上下文绘制图像，实现了旋转图像的功能，该方法的定义如下：

```
public void rotate(double vecx, double vecy, double anchorx, double anchory)
```

参数说明

❶ vecx：旋转向量的 x 坐标。

❷ vecy：旋转向量的 y 坐标。

❸ anchorx：旋转锚点的 x 坐标。

❹ anchory：旋转锚点的 y 坐标。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 ImageRotateApplet 小应用程序，用于在其中绘制旋转的图像，该类的代码如下：

```
public class ImageRotateApplet extends Applet {
    public void paint(Graphics g) {
        String value = "旋转图像";
        Image img = null;                                    //声明图像对象
        img = getImage(getCodeBase(), "com/zzk/PD2.jpg");    //获得图片信息
        Graphics2D g2d = (Graphics2D) g;                     //强制转换为 Graphics2D 对象
        g2d.drawString(value, 180, 150);                     //绘制文本
        int x = 50;                                          //图像位置的横坐标
        int y = -10;                                         //图像位置的纵坐标
        int w = img.getWidth(this);                          //获得图片的宽度
        int h = img.getHeight(this);                         //获得图片的高度
        g2d.drawImage(img, x, y + 50, w / 5, h / 5, this);   //绘制图形
        g2d.drawImage(img, x + 150, y + 50, w / 5, h / 5, this);  //绘制图形
        AffineTransform tr = new AffineTransform();          //创建变形对象
```

```
            tr.rotate(90, 15, 15, 65);                          //设置旋转角度
            g2d.setTransform(tr);                               //执行旋转
            g2d.drawImage(img, x + 150, y + 20, w / 5, h / 5, this);   //绘制图形
            tr.rotate(35, 15, 30, 65);                          //设置旋转角度
            g2d.setTransform(tr);                               //执行旋转
            g2d.drawImage(img, x + 120, y - 60, w / 5, h / 5, this);   //绘制图形
        }
    }
```

（3）在项目的 src 文件夹中创建一个名为 ImageRotateApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.ImageRotateApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 583：旋转图像的意义。

当使用相机拍照时，有时会以一定的角度进行拍摄，当把这样的图像放到文档或程序中时，就可以将图像旋转一定的角度，使其正常显示。

实例 584　Applet 中倾斜图像

光盘位置：光盘\MR\584　　中级　　趣味指数：★★★★

实例说明

本实例实现了在 Applet 中倾斜图像的功能。运行程序，可以在浏览器窗口中看到图像倾斜后的效果，如图 21.39 所示。

图 21.39　在 Applet 中倾斜图像的效果

关键技术

本实例主要是通过 AffineTransform 类的 shear()方法倾斜绘图上下文，然后在绘图上下文绘制图像，实现了倾斜图像的功能，该方法的定义如下：

```
public void shear(double shx, double shy)
```

参数说明

❶ shx：坐标在正 X 轴方向上进行位移的乘数，与其 Y 坐标的因子一样。

❷ shy：坐标在正 Y 轴方向上进行位移的乘数，与其 X 坐标的因子一样。

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 SlantImageApplet 小

应用程序，用于在其中绘制倾斜的图像，该类的代码如下：

```
public class SlantImageApplet extends Applet {
    public void paint(Graphics g) {
        String value = "倾斜图像";
        Image img = null;                                          //声明图像对象
        img = getImage(getCodeBase(), "com/zzk/PD5.jpg");          //获得图片信息
        Graphics2D g2d = (Graphics2D) g;                           //强制转换为 Graphics2D 对象
        g2d.drawString(value, 209, 170);                           //绘制文本
        AffineTransform tr = new AffineTransform();                //创建 AffineTransform 对象
        tr.translate(210, 32);                                     //图像位置的平移
        tr.shear(3, 3);                                            //倾斜图像
        g2d.drawImage(img, tr, this);                              //绘制图像
        AffineTransform tr1 = AffineTransform.getScaleInstance(3.5, 3.5);  //获得 AffineTransform 对象
        tr1.translate(15, 13);                                     //图像位置的平移
        g2d.drawImage(img, tr1, this);                             //绘制图像
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 SlantImageApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.SlantImageApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 584：在倾斜的图像上添加文字。

为了在倾斜的图像上添加文字，可以先创建 AffineTransform 对象，并使用 shear()方法将该对象设置为倾斜，然后将该对象设置为绘图上下文的 transform()方法的参数，这时再分别绘制图像和文字，即可实现在倾斜的图像上添加文字的功能。

实例 585　Applet 中调整图片的亮度

初级

光盘位置：光盘\MR\585

趣味指数：★★★★

实例说明

本实例实现了在 Applet 中调整图片亮度的功能。运行程序，可以在浏览器窗口中看到调整图片亮度前后的效果，如图 21.40 所示，其中，左侧是源图像的效果，右侧是调整亮度后的图片效果。

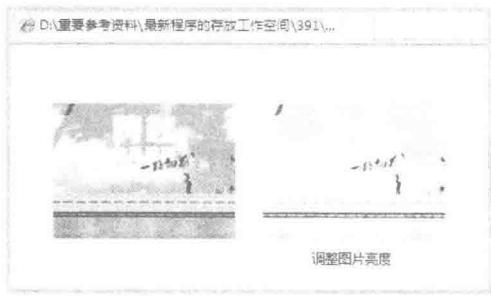


图 21.40　在 Applet 中调整图片亮度前后的效果

关键技术

本实例主要是通过为 RescaleOp 类的构造方法传递两个浮点数创建 RescaleOp 对象，然后使用该对象的 filter()

方法过滤缓冲图像对象，实现了调整图像亮度的功能，实现代码如下：

```
float fa = 2.0f;                                    //声明表示像素分量
float fb = -30.0f;                                  //声明表示像素分量
RescaleOp op = new RescaleOp(fa, fb, null);         //创建 RescaleOp 对象
image = op.filter(image, null);                     //过滤缓冲图像对象，实现调整图像亮度的功能
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 SetImageLightenessApplet 小应用程序，用于在其中绘制源图像和调整亮度后的图像，该类的代码如下：

```
public class SetImageLightenessApplet extends Applet {
    private BufferedImage image;                                    //声明缓冲图像对象
    public void paint(Graphics g) {
        Image img = null;                                           //声明图像对象
        String value = "调整图片亮度";
        img = getImage(getCodeBase(), "com/zzk/PPD.jpg");           //获得图像对象
        int a = img.getWidth(this);                                 //获得图片宽度赋给变量 a
        int b = img.getHeight(this);                                //获得图片高度赋给变量 b
        if (a >= 0 || b >= 0) {
            image = new BufferedImage(img.getWidth(this), img.getHeight(this),
                    BufferedImage.TYPE_INT_RGB);                    //创建缓冲图像对象
            image.getGraphics().drawImage(img, 0, 0, null);         //在缓冲图像对象上绘制图像
            float fa = 2.0f;                                        //声明像素分量
            float fb = -30.0f;                                      //声明像素分量
            RescaleOp op = new RescaleOp(fa, fb, null);             //创建 RescaleOp 对象
            image = op.filter(image, null);                         //过滤缓冲图像对象，实现调整图像亮度的功能
            g.drawImage(img, 30, 40, this);                         //绘制源图像对象
            g.drawImage(image, 220, 40, this);                      //绘制调整亮度后的缓冲图像对象
            g.drawString(value, 265, 188);                          //绘制文本
        }
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 SetImageLightenessApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.SetImageLightenessApplet.class" width ="1200" height = "1000">
</applet>
</html>
```

秘笈心法

心法领悟 585：使图片更具有观赏性。

如果已有图片比较暗，就很难分辨出图像的内容，这时可以通过调整图片亮度，使图片的内容看起来比较清晰，这样的图片更能被用户接受，更具有观赏性。

实例 586　Applet 中绘制中文验证码

高级

光盘位置：光盘\MR\586

趣味指数：★★★★★

实例说明

在网页程序中，使用验证码可以增强程序的安全性。本实例实现了在 Applet 中绘制中文验证码的功能。运行程序，可以在浏览器窗口中看到所绘制的中文验证码，效果如图 21.41 所示。

图 21.41　在 Applet 中绘制的中文验证码

关键技术

本实例主要是通过字符串变量存储作为验证码的中文，然后从字符串中随机获得 4 个中文字符，并对这 4 个中文字符进行旋转和缩放，从而实现了在小应用程序中绘制中文验证码的功能，其中实现旋转和缩放字符的代码如下：

```
Graphics2D gs2d = (Graphics2D) gs;                                    //将文字旋转指定角度
AffineTransform trans = new AffineTransform();                        //实例化 AffineTransform
trans.rotate(random.nextInt(45) * 3.14 / 180, 22 * i + 8, 7);
float scaleSize = random.nextFloat() + 0.8f;                          //缩放文字
if (scaleSize > 1f)
    scaleSize = 1f;                                                   //如果 scaleSize 大于 1，则让其等于 1
trans.scale(scaleSize, scaleSize);                                    //进行缩放
gs2d.setTransform(trans);                                             //设置 AffineTransform 对象
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawChinesePasswordApplet 小应用程序，用于在其中绘制中文验证码，该类的代码如下：

```
public class DrawChinesePasswordApplet extends Applet {
    int WIDTH = 120;                                                  //宽度
    int HEIGHT = 35;                                                  //高度
    private String num = "";                                          //验证码
    Random random = new Random();                                     //实例化 Random
    public void paint(Graphics g) {
        String hanZi = "本实例通过从字符串中随机获得 4 个字符实现了中文验证码的功能";   //定义验证码使用的汉字
        BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
                BufferedImage.TYPE_INT_RGB);                          //实例化 BufferedImage
        Graphics gs = image.getGraphics();                            //获取 Graphics 类的对象
        if (!num.isEmpty()) {
            num = "";                                                 //清空验证码
        }
        Font font = new Font("黑体", Font.BOLD, 20);                  //通过 Font 构造字体
        gs.setFont(font);                                             //设置字体
        gs.fillRect(0, 0, WIDTH, HEIGHT);                             //填充一个矩形
        //输出随机的验证文字
        for (int i = 0; i < 4; i++) {
            int index = random.nextInt(hanZi.length());               //随机获得汉字的索引值
            String ctmp = hanZi.substring(index, index + 1);          //获得指定索引处的一个汉字
            num += ctmp;                                              //更新验证码
            Color color = new Color(20 + random.nextInt(120), 20 + random
                    .nextInt(120), 20 + random.nextInt(120));         //生成随机颜色
            gs.setColor(color);                                       //设置颜色
            //这里省略了关键技术中给出的代码
            gs2d.drawString(ctmp, WIDTH / 6 * i + 28, HEIGHT / 2);    //绘制验证码
        }
        g.drawImage(image, 85, 80, null);                             //在小应用程序中绘制验证码
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawChinesePasswordApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawChinesePasswordApplet.class" width ="1200" height = "1000">
</applet>
</html>
```

秘笈心法

心法领悟 586：使用字符数组存储作为验证码的汉字。

本实例使用字符串变量存储作为验证码的汉字，另外，也可以使用字符数组作为验证码的汉字，然后在字符数组中随机取出 4 个字符，并将其绘制到小应用程序中，也可以实现绘制中文验证码的功能。

实例 587　Applet 中绘制图片验证码

光盘位置：光盘\MR\587　　高级　　趣味指数：★★★★★

实例说明

本实例实现了在 Applet 中绘制图片验证码的功能。运行程序，可以在浏览器窗口中看到所绘制的图片验证码，效果如图 21.42 所示。

图 21.42　在 Applet 中绘制的图片验证码

关键技术

本实例主要是通过在缓冲图像上绘制图片，接着随机获得 4 个 A~Z 的大写字母，并对这 4 个大写字母进行旋转和缩放，然后将其绘制到含有图片的缓冲图像上，最后再将该缓冲图像绘制到小应用程序中，从而实现了在小应用程序中绘制图片验证码的功能，其中在缓冲图像上绘制图片的代码如下：

```
BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
        BufferedImage.TYPE_INT_RGB);                    //实例化 BufferedImage
Graphics gs = image.getGraphics();                      //获取 Graphics 类的对象
if (!num.isEmpty()) {
    num = "";                                           //清空验证码
}
Font font = new Font("黑体", Font.BOLD, 20);            //通过 Font 构造字体
gs.setFont(font);                                       //设置字体
gs.fillRect(0, 0, WIDTH, HEIGHT);                       //填充一个矩形
Image img = null;                                       //声明图像对象
img = getImage(getCodeBase(), "com/zzk/PPD.jpg");       //创建图像对象
gs.drawImage(img, 0, 0, WIDTH, HEIGHT, this);           //在缓冲图像对象上绘制图像
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawPasswordWithImageApplet

小应用程序，用于在其中绘制图片验证码，该类的代码如下：

```
public class DrawPasswordWithImageApplet extends Applet {
    int WIDTH = 120;                                                    //宽度
    int HEIGHT = 35;                                                    //高度
    private String num = "";                                            //验证码
    Random random = new Random();                                       //实例化 Random
    public void paint(Graphics g) {
        //这里省略了关键技术中给出的代码
        for (int i = 0; i < 4; i++) {                                   //输出随机的验证字符
            char ctmp = (char) (random.nextInt(26) + 65);               //生成 A~Z 的字母
            num += ctmp;                                                //更新验证码
            Color color = new Color(20 + random.nextInt(120), 20 + random
                    .nextInt(120), 20 + random.nextInt(120));           //生成随机颜色
            gs.setColor(color);                                         //设置颜色
            Graphics2D gs2d = (Graphics2D) gs;                          //将文字旋转指定角度
            AffineTransform trans = new AffineTransform();              //实例化 AffineTransform
            trans.rotate(random.nextInt(45) * 3.14 / 180, 22 * i + 8, 7);
            float scaleSize = random.nextFloat() + 0.8f;                //缩放文字
            if (scaleSize > 1f)
                scaleSize = 1f;                                         //如果 scaleSize 大于 1，则让其等于 1
            trans.scale(scaleSize, scaleSize);                          //进行缩放
            gs2d.setTransform(trans);                                   //设置 AffineTransform 对象
            gs2d.drawString(String.valueOf(ctmp), WIDTH / 6 * i + 28, HEIGHT / 2);  //绘制验证码
        }
        g.drawImage(image, 85, 80, null);                               //在小应用程序中绘制含有图片验证码的缓冲图像
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawPasswordWithImageApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawPasswordWithImageApplet.class" width ="1200" height = "1000">
</applet>
</html>
```

秘笈心法

心法领悟 587：混合使用字母和数字的验证码。

本实例实现了图片验证码的功能，验证字符是大写字母，而在实际应用中，可以混合使用字母和数字作为验证码，实现方法是将 A~Z 的大写字母和 0~9 的数字通过字符串变量或字符数组存储起来，然后随机取出其中的 4 个字符作为验证字符。

实例 588　Applet 中绘制带干扰线的验证码

光盘位置：光盘\MR\588　　高级　　趣味指数：★★★★★

实例说明

本实例实现了在 Applet 中绘制带干扰线的验证码的功能。运行程序，可以在浏览器窗口中看到所绘制的带干扰线的验证码，效果如图 21.43 所示。

图 21.43　在 Applet 中绘制带干扰线的验证码

关键技术

本实例主要是通过在缓冲图像上绘制图片和两条相互连接的直线，再随机获得 4 个 A~Z 的大写字母，并对这 4 个大写字母进行旋转和缩放，然后将其绘制到含有图片和干扰线的缓冲图像上，最后将该缓冲图像绘制到小应用程序中，从而实现了在小应用程序中绘制带干扰线的验证码的功能，其中在缓冲图像上绘制干扰线的代码如下：

```
BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
        BufferedImage.TYPE_INT_RGB);                    //实例化 BufferedImage
Graphics gs = image.getGraphics();                      //获取 Graphics 类的对象
int startX1 = random.nextInt(20);                       //随机获取第一条干扰线起点的 x 坐标
int startY1 = random.nextInt(20);                       //随机获取第一条干扰线起点的 y 坐标
int startX2 = random.nextInt(30) + 35;                  //随机获取第一条干扰线终点的 x 坐标，也是第二条干扰线起点的 x 坐标
int startY2 = random.nextInt(10) + 20;                  //随机获取第一条干扰线终点的 y 坐标，也是第二条干扰线起点的 y 坐标
int startX3 = random.nextInt(30) + 90;                  //随机获取第二条干扰线终点的 x 坐标
int startY3 = random.nextInt(10) + 5;                   //随机获取第二条干扰线终点的 y 坐标
gs.setColor(Color.RED);
gs.drawLine(startX1, startY1, startX2, startY2);         //绘制第一条干扰线
gs.setColor(Color.BLUE);
gs.drawLine(startX2, startY2, startX3, startY3);         //绘制第二条干扰线
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 DrawPasswordWithDisturbApplet 小应用程序，用于在其中绘制带干扰线的验证码，该类的代码如下：

```
public class DrawPasswordWithDisturbApplet extends Applet {
    int WIDTH = 120;                                    //设置宽度
    int HEIGHT = 35;                                    //设置高度
    private String num = "";                            //验证码
    Random random = new Random();                       //实例化 Random
    public void paint(Graphics g) {
        BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
                BufferedImage.TYPE_INT_RGB);            //实例化 BufferedImage
        Graphics gs = image.getGraphics();              //获取 Graphics 类的对象
        //省略了在缓冲图像上绘制图片的代码
        int startX1 = random.nextInt(20);               //随机获取第一条干扰线起点的 x 坐标
        int startY1 = random.nextInt(20);               //随机获取第一条干扰线起点的 y 坐标
        int startX2 = random.nextInt(30) + 35;          //随机获取第一条干扰线终点的 x 坐标，也是第二条干扰线起点的 x 坐标
        int startY2 = random.nextInt(10) + 20;          //随机获取第一条干扰线终点的 y 坐标，也是第二条干扰线起点的 y 坐标
        int startX3 = random.nextInt(30) + 90;          //随机获取第二条干扰线终点的 x 坐标
        int startY3 = random.nextInt(10) + 5;           //随机获取第二条干扰线终点的 y 坐标
        gs.setColor(Color.RED);
        gs.drawLine(startX1, startY1, startX2, startY2); //绘制第一条干扰线
        gs.setColor(Color.BLUE);
        gs.drawLine(startX2, startY2, startX3, startY3); //绘制第二条干扰线
        for (int i = 0; i < 4; i++) {                   //输出随机的验证文字
            //省略了在缓冲图像上绘制验证码的代码
        }
        g.drawImage(image, 85, 80, null);               //在小应用程序中绘制含有图片和干扰线验证码的缓冲图像
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 DrawPasswordWithDisturbApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.DrawPasswordWithDisturbApplet.class" width ="1200" height = "1000">
</applet>
</html>
```

秘笈心法

心法领悟 588：看不清楚时更换验证码。

在一些程序中，为了避免看不清楚验证码，可以在程序界面中添加一个标签，标明单击更换验证码，这样当看不清楚验证码时，可以通过单击该标签更换验证码。

实例 589 Applet 中模糊图像

光盘位置：光盘\MR\589　　初级　　趣味指数：★★★★★

实例说明

本实例实现了在 Applet 中模糊图像的功能。运行程序，可以在浏览器窗口中看到原图像和模糊后的图像，效果如图 21.44 所示。

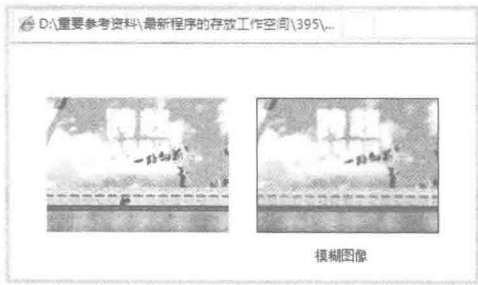


图 21.44　在 Applet 中模糊图像前后的效果

关键技术

本实例主要是通过为 Kernel 类传递两个整数和一个表示像素分量的数组创建 Kernel 对象，然后将该对象传递给 RescaleOp 类的构造方法创建 RescaleOp 对象，并通过 RescaleOp 对象的 filter()方法过滤缓冲图像对象，从而实现了模糊图像的功能，实现代码如下：

```
float[] data = { 0.0532f, 0.132f, 0.0532f, 0.132f, 0.19f, 0.138f,
        0.0532f, 0.132f, 0.0532f };                          //声明表示像素分量的数组
Kernel kernel = new Kernel(3, 3, data);                      //创建 Kernel 对象
ConvolveOp op = new ConvolveOp(kernel);                      //创建 ConvolveOp 对象
image = op.filter(image, null);                              //过滤缓冲图像对象
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 BlurImageApplet 小应用程序，用于在其中绘制原图像和模糊后的图像，该类的代码如下：

```
public class BlurImageApplet extends Applet {
    private BufferedImage image;                             //声明缓冲图像对象
    public void paint(Graphics g) {
        Image img = null;                                    //声明创建图像对象
        String value = "模糊图像";
        img = getImage(getCodeBase(), "com/zzk/PPD.jpg");    //获得图片信息
        int a = img.getWidth(this);                          //获得图片宽度赋给变量 a
        int b = img.getHeight(this);                         //获得图片高度赋给变量 b
        if (a >= 0 || b >= 0) {
            image = new BufferedImage(img.getWidth(this), img.getHeight(this),
                    BufferedImage.TYPE_INT_RGB);             //创建缓冲图像对象
```

```
            image.getGraphics().drawImage(img, 0, 0, null);          //在缓冲图像对象上绘制图像
            float[] data = { 0.0532f, 0.132f, 0.0532f, 0.132f, 0.19f, 0.138f,
                    0.0532f, 0.132f, 0.0532f };                      //声明表示像素分量的数组
            Kernel kernel = new Kernel(3, 3, data);                  //创建 Kernel 对象
            ConvolveOp op = new ConvolveOp(kernel);                  //创建 ConvolveOp 对象
            image = op.filter(image, null);                          //过滤缓冲图像对象
            g.drawImage(img, 25, 35, this);                          //绘制缓冲图像对象
            g.drawImage(image, 215, 35, this);                       //绘制缓冲图像对象
            g.drawString(value, 268, 186);                           //绘制文本
        }
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 BlurImageApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.BlurImageApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 589：使图片具有朦胧的效果。

在使用图片时，如果希望图片有朦胧的效果，可以使用模糊图像的功能对图片进行模糊处理，处理后的图片就具有朦胧的效果。

实例 590　Applet 中锐化图像

初级

趣味指数：★★★★

实例说明

本实例实现了在 Applet 中锐化图像的功能。运行程序，可以在浏览器窗口中看到原图像和锐化后的图像，效果如图 21.45 所示。

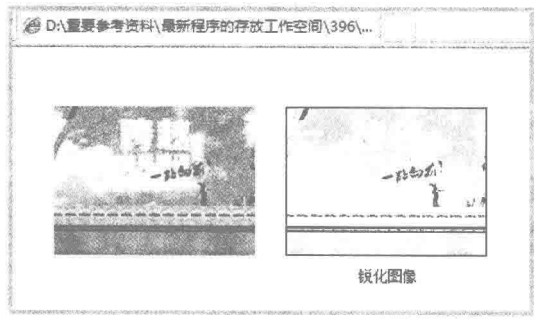


图 21.45　在 Applet 中图像锐化前后的效果

关键技术

本实例也是通过 Kernel 和 RescaleOp 类实现了锐化图像的功能，除了表示像素分量的数组元素值与模糊图像不同外，其他内容都是相同的。实现锐化图像的代码如下：

```
float[] data = { 0.0f, -1.0f, 0.0f, -1.0f, 6.0f, -1.0f, 0.0f,
        -1.0f, 0.0f };                                   //声明表示像素分量的数组
Kernel kernel = new Kernel(3, 3, data);                  //创建 Kernel 对象
ConvolveOp op = new ConvolveOp(kernel);                  //创建 ConvolveOp 对象
image = op.filter(image, null);                          //过滤缓冲图像对象
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 SharperImageApplet 小应用程序，用于在其中绘制原图像和锐化后的图像，该类的代码如下：

```
public class SharperImageApplet extends Applet {
    private BufferedImage image;                                    //声明缓冲图像对象
    public void paint(Graphics g) {
        Image img = null;                                           //声明创建图像对象
        String value = "锐化图像";
        img = getImage(getCodeBase(), "com/zzk/PPD.jpg");           //获得图片信息
        int a = img.getWidth(this);                                 //获得图片宽度赋给变量 a
        int b = img.getHeight(this);                                //获得图片高度赋给变量 b
        if (a >= 0 || b >= 0) {
            image = new BufferedImage(img.getWidth(this), img.getHeight(this),
                    BufferedImage.TYPE_INT_RGB);                    //创建缓冲图像对象
            image.getGraphics().drawImage(img, 0, 0, null);         //在缓冲图像对象上绘制图像
            float[] data = { 0.0f, -1.0f, 0.0f, -1.0f, 6.0f, -1.0f, 0.0f,
                    -1.0f, 0.0f };                                  //声明表示像素分量的数组
            Kernel kernel = new Kernel(3, 3, data);                 //创建 Kernel 对象
            ConvolveOp op = new ConvolveOp(kernel);                 //创建 ConvolveOp 对象
            image = op.filter(image, null);                         //过滤缓冲图像对象
            g.drawImage(img, 25, 35, this);                         //绘制缓冲图像对象
            g.drawImage(image, 215, 35, this);                      //绘制缓冲图像对象
            g.drawString(value, 275, 182);                          //绘制文本
        }
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 SharperImageApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.SharperImageApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 590：锐化图像的作用。

在使用图片时，如果图片比较模糊，可以使用锐化图像功能对图片进行处理，从而可以使模糊的图像更加清晰，在作图软件中，经常使用锐化功能对模糊的图片进行处理。

实例 591　Applet 中照亮图像边缘

初级

光盘位置：光盘\MR\591

趣味指数：★★★★

实例说明

本实例实现了在 Applet 中照亮图像边缘的功能。运行程序，可以在浏览器窗口中看到原图像和照亮边缘后的图像，效果如图 21.46 所示。

关键技术

本实例也是通过 Kernel 和 RescaleOp 类实现了照亮图像边缘的功能，除了表示像素分量的数组元素值与模糊图像和锐化图像不同，其他内容都是相同的。实现照亮图像边缘的代码如下：

```
float[] f = { 0.0f, -1.5f, 0.0f, -1.5f, 6.0f, -1.5f, 0.0f, -1.5f,
        0.0f };                                                     //声明表示像素分量的数组
```

```
Kernel kernel = new Kernel(3, 3, f);                    //创建 Kernel 对象
ConvolveOp op = new ConvolveOp(kernel);                 //创建 RescaleOp 对象
image = op.filter(image, null);                         //过滤缓冲图像对象
```

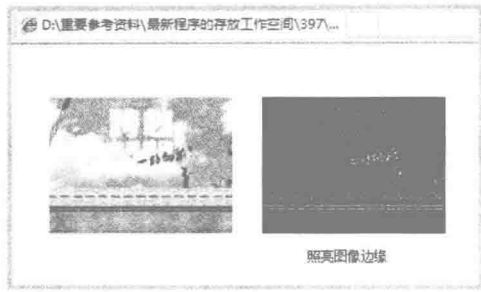


图 21.46　在 Applet 中照亮图像边缘前后的效果

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 lightenImageEdgeApplet 小应用程序，用于在其中绘制原图像和照亮边缘后的图像，该类的代码如下：

```
public class lightenImageEdgeApplet extends Applet {
    private BufferedImage image;                                        //声明缓冲图像对象
    public void paint(Graphics g) {
        Image img = null;                                               //声明创建图像对象
        String value = "照亮图像边缘";
        img = getImage(getCodeBase(), "com/zzk/PPD.jpg");               //获得图片信息
        int a = img.getWidth(this);                                     //获得图片宽度赋给变量 a
        int b = img.getHeight(this);                                    //获得图片高度赋给变量 b
        if (a >= 0 || b >= 0) {
            image = new BufferedImage(a, b, BufferedImage.TYPE_INT_RGB);//创建缓冲图像对象
            image.getGraphics().drawImage(img, 0, 0, this);             //在缓冲图像对象上绘制图像
            float[] f = { 0.0f, -1.5f, 0.0f, -1.5f, 6.0f, -1.5f, 0.0f, -1.5f,
                    0.0f };                                             //声明表示像素分量的数组
            Kernel kernel = new Kernel(3, 3, f);                        //创建 Kernel 对象
            ConvolveOp op = new ConvolveOp(kernel);                     //创建 ConvolveOp 对象
            image = op.filter(image, null);                             //过滤缓冲图像对象
            g.drawImage(img, 25, 35, this);                             //绘制缓冲图像对象
            g.drawImage(image, 217, 35, this);                          //绘制缓冲图像对象
            g.drawString(value, 258, 186);                              //绘制文本
        }
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 lightenImageEdgeApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.lightenImageEdgeApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 591：照亮图像边缘的作用。

通过对图像进行照亮边缘处理，可以识别图像的边缘，并对其添加类似霓虹灯的光亮效果，在作图软件中，也经常使用照亮图像边缘功能对图像进行处理。

实例 592　Applet 中反向图像

初级

趣味指数：★★★

实例说明

本实例实现了在 Applet 中反向图像的功能。运行程序，可以在浏览器窗口中看到原图像和反向后的图像，效果如图 21.47 所示。

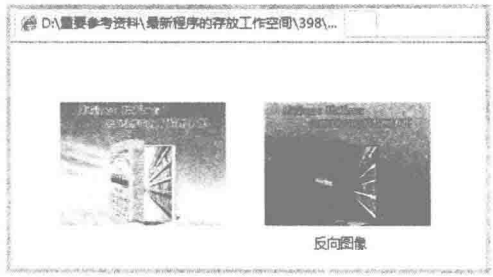


图 21.47　在 Applet 中图像反向前后的效果

关键技术

本实例主要是使用 ShortLookupTable 类通过表示颜色反向分量的数组创建查找表对象，然后将查找表对象作为 LookupOp 类构造方法的参数，创建实现从源到目标查找操作的 LookupOp 对象，最后使用 LookupOp 对象的 filter()方法过滤缓冲图像，从而实现了反向图像的操作，实现代码如下：

```
short[] negative = new short[256 * 1];                          //创建表示颜色反向的分量数组
for (int i = 0; i < 256; i++) {
    negative[i] = (short) (255 - i);                            //为数组赋值
}
ShortLookupTable table = new ShortLookupTable(0, negative);     //创建查找表对象
LookupOp op = new LookupOp(table, null);                        //创建实现从源到目标查找操作的 LookupOp 对象
image = op.filter(image, null);                                 //调用 LookupOp 对象的 filter()方法，实现图像反向功能
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 NegativeImageApplet 小应用程序，用于在其中绘制原图像和反向后的图像，该类的代码如下：

```
public class NegativeImageApplet extends Applet {
    private BufferedImage image;                                //声明缓冲图像对象
    public void paint(Graphics g) {
        String value = "反向图像";
        Image img = null;                                       //声明图像对象
        img = getImage(getCodeBase(), "com/zzk/PD3.jpg");       //获得图片信息
        int a = img.getWidth(this);                             //获得图片宽度赋给变量 a
        int b = img.getHeight(this);                            //获得图片高度赋给变量 b
        if (a >= 0 || b >= 0) {
            image = new BufferedImage(img.getWidth(this), img.getHeight(this),
                    BufferedImage.TYPE_INT_RGB);                //创建缓冲图像对象
            image.getGraphics().drawImage(img, 0, 0, null);     //在缓冲图像对象上绘制图像
            short[] negative = new short[256 * 1];              //创建表示颜色反向的分量数组
            for (int i = 0; i < 256; i++) {
                negative[i] = (short) (255 - i);                //为数组赋值
            }
            ShortLookupTable table = new ShortLookupTable(0, negative);   //创建查找表对象
```

```
                LookupOp op = new LookupOp(table, null);                    //创建实现从源到目标查找操作的 LookupOp 对象
                image = op.filter(image, null);                             //调用 LookupOp 对象的 filter()方法，实现图像反向功能
                if (image != null) {
                    g.drawImage(img, 35, 40, null);                         //绘制缓冲图像对象
                    g.drawImage(image, 220, 40, null);                      //绘制缓冲图像对象
                }
                g.drawString(value, 265, 175);                              //绘制文本
            }
        }
    }
```

（3）在项目的 src 文件夹中创建一个名为 NegativeImageApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.NegativeImageApplet.class" width ="1200" height = "1000">
</applet>
</html>
```

秘笈心法

心法领悟 592：反向的作用。

对图像进行反向处理，是为了调整并反转图像的颜色，使其与原图像有明显的对比，并且该操作是可逆的，即一旦对图像进行了反向操作，如果再对反向后得到的图像进行反向处理，又可以恢复为原来的图像。

实例 593　Applet 中图像动态拉伸

初级

实例位置：光盘\MR\593

趣味指数：★★★

实例说明

本实例实现了在 Applet 中动态拉伸图片的功能。运行程序，可以在浏览器窗口中看到动态拉伸图片的动画效果，即图片由小变大，然后又由大变小，并重复这个过程，如图 21.48 所示是原图片和图片拉伸变大后的效果。

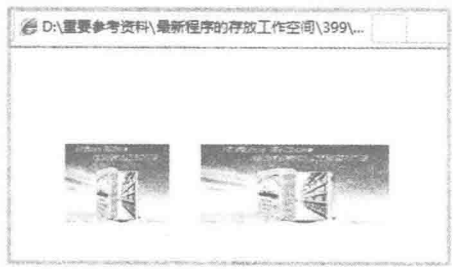


图 21.48　Applet 中动态拉伸图片并使图片变大的效果

关键技术

本实例主要是通过定义标记变量和调整图片宽度的变量，实现了动态拉伸图片的功能，当标记变量为 true 时，使图片的宽度变大；当标记变量为 false 时，使图片的宽度变小。实现改变图片宽度的代码如下：

```
if (flag) {
    xw += 0.1f;                                                             //标记变量为 true 时执行
    if (xw > 2.0f) {                                                        //使宽度变大
        flag = false;                                                       //宽度大于 2.0 时
    }                                                                       //标记变量为 false
} else {
    xw -= 0.1f;                                                             //标记变量为 false 时执行
    if (xw < 0.5f) {                                                        //使宽度变小
                                                                            //宽度小于 2.0 时
```

```
            flag = true;                                    //标记变量为 true
        }
    }
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 ImageElongateApplet 小应用程序，用于在其中绘制原图像和动态拉伸后的图片，该类的代码如下：

```
public class ImageElongateApplet extends Applet implements Runnable {
    private boolean flag = true;                            //声明标记变量
    private static float xw;                                //定义调整图像宽度的变量
    public void init(){
        xw = 0.5f;                                          //初始化图像宽度
    }
    public void start(){
        Thread th = new Thread(this);                       //创建线程对象
        th.start();                                         //启动线程对象
    }
    public void paint(Graphics g) {
        Image img = null;                                   //声明图像对象
        img = getImage(getCodeBase(), "com/zzk/PD4.jpg");   //获得图片信息
        int w = img.getWidth(this);                         //设置图像的宽度
        int h = img.getHeight(this);                        //设置图像的高度
        Graphics2D g2d = (Graphics2D) g;                    //将 g 转换为可以利用的 Graphics2D
        g2d.drawImage(img, w - 50, h, this);                //绘制图像
        AffineTransform tr = new AffineTransform(xw, 0, 0, 1, 150, h);  //创建仿射变换对象进行设置变换（第一个参数）
        g2d.drawImage(img, tr, this);                       //绘制图像

    }
    @Override
    public void run() {
        while (true) {
            if (flag) {                                     //标记变量为 true 时执行
                xw += 0.1f;                                 //使宽度变大
                if (xw > 2.0f) {                            //宽度大于 2.0 时
                    flag = false;                           //标记变量为 false
                }
            } else {                                        //标记变量为 false 时执行
                xw -= 0.1f;                                 //使宽度变小
                if (xw < 0.5f) {                            //宽度小于 2.0 时
                    flag = true;                            //标记变量为 true
                }
            }
            try {
                Thread.sleep(200);                          //休眠 200 毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            repaint();                                      //重新调用 paint()方法
        }
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 ImageElongateApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.ImageElongateApplet.class" width ="500" height = "220">
</applet>
</html>
```

秘笈心法

心法领悟 593：使图片在宽度和高度上同时改变。

本实例使图片在宽度上变大或变小，如果希望图片在宽度和高度上同时改变，可以再增加一个表示调整图片高度的变量，使其随图片的宽度相应地增大或减小，即可实现使图片在宽度和高度上同时改变的功能。

21.6　Applet 中的文字动画

实例 594	Applet 中文字缩放动画	高级
	光盘位置：光盘\MR\594	趣味指数：★★★

实例说明

本实例实现了在 Applet 中进行文字缩放的功能。运行程序，可以在浏览器窗口中看到文字的缩放动画，如图 21.49 所示为文字缩放动画的一种效果。

图 21.49　Applet 中文字缩放动画的一种效果

关键技术

本实例主要是通过定义布尔型标记变量 flag，然后在线程的 run()方法中通过该变量来控制表示字体大小的变量 x 的值，从而实现了文字缩放动画。线程中 run()方法的部分代码如下：

```
if (flag) {                                          //flag 为 true 时
    //省略了使 x 值变小的代码
} else {                                             //flag 为 false 时
    //省略了使 x 值变大的代码
}
font = new Font("华文楷体", Font.BOLD, x);           //重新创建字体对象
repaint();                                           //调用 repaint()方法
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 TextScaleApplet 小应用程序，用于在其中显示文字缩放动画，该类的代码如下：

```
public class TextScaleApplet extends Applet implements Runnable {
    public void start() {
        Thread thread = new Thread(this);                //创建线程对象
        thread.start();                                  //启动线程对象
    }
    private Image img = null;                            //声明图像对象
    boolean flag = false;                                //定义标记变量，用于控制 x 的值
    int x = 12;                                          //定义表示字体大小的变量 x
    Font font = new Font("华文楷体", Font.BOLD, 42);     //创建字体对象
    public void paint(Graphics g) {
        img = getImage(getCodeBase(), "com/zzk/PD3.jpg");//获取图片资源的路径
        Graphics2D g2 = (Graphics2D) g;                  //获得 Graphics2D 对象
```

```
            g2.drawImage(img, 0, 0, getWidth(), getHeight(), this);    //绘制图像
            g2.setFont(font);                                          //指定字体
            g2.setColor(Color.red);                                    //指定颜色
            g2.drawString("学无止境", 30, 120);                         //绘制文本
        }
        public void run() {
            while (true) {
                if (flag) {                                            //flag 为 true 时
                    x -= 2;                                            //x 进行减 1 计算
                    if (x <= 12) {                                     //x 小于等于 12 时
                        x = 12;                                        //x 等于 12
                        flag = false;                                  //为 flag 赋值为 false
                    }
                } else {                                               //flag 为 false 时
                    x += 2;                                            //x 进行加 1 计算
                    if (x >= 72) {                                     //x 大于等于 72 时
                        x = 72;                                        //x 等于 72
                        flag = true;                                   //为 flag 赋值为 true
                    }
                }
                font = new Font("华文楷体", Font.BOLD, x);             //重新创建字体对象
                repaint();                                             //调用 repaint()方法
                try {
                    Thread.sleep(50);                                  //休眠 50 毫秒
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
```

（3）在项目的 src 文件夹中创建一个名为 TextScaleApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.TextScaleApplet.class" width ="350" height = "200">
</applet>
</html>
```

秘笈心法

心法领悟 594：避免在 Applet 中出现屏幕闪烁。

当在 Applet 中实现文字缩放动画时，有时会出现屏幕闪烁现象，这时可以重写 update()方法，并在该方法中调用 paint()方法，从而可以解决屏幕闪烁的问题，代码如下：

```
public void update(Graphics g) {
    paint(g);                          //调用 paint()方法
}
```

实例 595　Applet 中文字跑马灯动画

光盘位置：光盘\MR\595　　高级　　趣味指数：★★★

实例说明

本实例在 Applet 中实现了文字跑马灯动画。运行程序，可以在浏览器窗口中看到文字的跑马灯动画效果，如图 21.50 所示为文字跑马灯动画的一个效果。

关键技术

本实例主要是通过定义布尔型标记变量 flag，然后在线程的 run()方法中通过该变量来控制跑马灯动画文字中每个字符的坐标（该坐标通过数组 X 来存储），从而实现了文字跑马灯动画，线程中 run()方法的代码如下：

```
public void run() {
    boolean flag = false;                          //为 false 时表示第一次执行，x 坐标进行等比递增，否则进行等差递减
    while (true) {                                                    //读取内容
        try {
            for (int i = drawChar.length - 1; i >= 0; i--) {
                if (!flag) {
                    x[i] = x[i] + 20 * i;                             //x 坐标进行等比递增
                } else {
                    x[i] = x[i] + 20;                                 //x 坐标进行等比递减
                }
                if (x[i] >= 360 - 20) {                               //大于窗体宽度减 20 的值时
                    x[i] = 0;                                         //x 坐标为 0
                }
            }
            repaint();                                                //调用 paint()方法
            if (!flag) {
                flag = true;                                          //赋值为 true
            }
            Thread.sleep(1000);                                       //当前线程休眠 1000 毫秒
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

图 21.50　Applet 中文字跑马灯动画的一个效果

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 HorseRaceLightTextApplet 小应用程序，用于在其中显示文字跑马灯动画，该类的代码如下：

```
public class HorseRaceLightTextApplet extends Applet implements Runnable {
    public void start() {
        Thread thread = new Thread(this);                             //创建线程
        thread.start();                                               //启动线程对象
    }
    String value = "拥有编程词典，学习编程真轻松。";                    //存储绘制的内容
    char[] drawChar = value.toCharArray();                            //将绘制内容转换为字符数组
    int[] x = new int[drawChar.length];                               //存储每个字符绘制点 x 坐标的数组
    int y = 100;                                                      //存储绘制点 y 坐标

    public void paint(Graphics g) {
        Image img = null;
        img = getImage(getCodeBase(), "com/zzk/PD3.jpg");             //创建图像对象
        g.clearRect(0, 0, getWidth(), getHeight());                   //清除绘图上下文内容
        g.drawImage(img, 0, 0, getWidth(), getHeight(), this);        //绘制图像
        Font font = new Font("华文楷体", Font.BOLD, 20);              //创建字体对象
        g.setFont(font);                                              //指定字体
        g.setColor(Color.RED);                                        //指定颜色
        for (int j = drawChar.length - 1; j >= 0; j--) {
            g.drawString(drawChar[drawChar.length - 1 - j] + "", x[j], y);   //绘制文本
        }
```

```
    }
    //这里省略了关键技术中给出的 run()方法的代码
    public void update(Graphics g) {                    //重写 update()方法，防止闪烁
        paint(g);
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 HorseRaceLightTextApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.HorseRaceLightTextApplet.class" width ="350" height = "200">
</applet>
</html>
```

秘笈心法

心法领悟 595：绘制边框 3D 高亮显示的矩形。

在 Java 中，可以使用 Graphics 类提供的 drawRect()方法绘制矩形，但是该矩形是平面效果的，这时可以使用 Graphics2D 类提供的 draw3DRect()方法绘制边框为 3D 效果的矩形。

实例 596　Applet 中字幕显示动画

初级

趣味指数：★★

实例说明

本实例在 Applet 中实现了字幕显示动画。运行程序，可以在浏览器窗口中看到字幕显示动画的效果，如图 21.51 所示为字幕显示动画的一个效果。

图 21.51　Applet 中字幕显示动画的一个效果

关键技术

本实例主要是通过判断上次绘制的内容来改变下次将要绘制的文字，从而实现了字幕显示动画。判断绘制内容的实现代码如下：

```
if (value.equals("明日图书网的网址")) {                    //判断上次绘制的内容
    value = "http://www.mingribook.com";                  //改变绘制的内容
} else {
    value = "明日图书网的网址";                            //改变绘制的内容
}
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 CaptionSpecificApplet 小应用程序，用于在其中实现字幕显示动画，该类的代码如下：

```
public class CaptionSpecificApplet extends Applet implements Runnable {
    int x = 50;                                                    //存储绘制点的 x 坐标
    int y = 216;                                                   //存储绘制点的 y 坐标
    String value = "明日图书网的网址";                              //存储绘制的内容
    public void start() {
        Thread thread = new Thread(this);                          //创建线程对象
        thread.start();                                            //启动线程对象
    }
    public void paint(Graphics g) {
        g.clearRect(0, 0, 316, 237);                               //清除绘图上下文内容
        Image img = null;                                          //声明图片对象
        img = getImage(getCodeBase(), "com/zzk/PD3.jpg");
        g.drawImage(img, 0, 0, getWidth(), getHeight(), this);     //绘制图像
        Font font = new Font("华文楷体", Font.BOLD, 20);           //创建字体对象
        g.setFont(font);                                           //指定字体
        g.setColor(Color.RED);                                     //指定颜色
        g.drawString(value, x, y);                                 //绘制文本
    }
    public void run() {
        try {
            while (true) {                                         //读取内容
                Thread.sleep(100);                                 //当前线程休眠 100 毫秒
                if (y <= 216 - 50) {                               //如果已经向上移动 50 像素
                    y = 216;                                       //y 坐标定位到最下方
                    if (value.equals("明日图书网的网址")) {
                        value = "http://www.mingribook.com";       //改变绘制的内容
                    } else {
                        value = "明日图书网的网址";                //改变绘制的内容
                    }
                } else {                                           //如果还没向上移动到 50 像素
                    y -= 2;                                        //y 坐标上移
                }
                repaint();                                         //调用 repaint()方法
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void update(Graphics g) {                               //重写 update()方法，防止闪烁
        paint(g);
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 CaptionSpecificApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.CaptionSpecificApplet.class" width ="350" height = "200">
</applet>
</html>
```

秘笈心法

心法领悟 596：绘制有填充色的 3D 高亮显示的矩形。

在 Java 中，可以使用 Graphics 类提供的 fillRect()方法绘制填充矩形，但是该填充矩形是平面效果的，这时可以使用 Graphics2D 类提供的 fill3DRect()方法绘制有填充色的 3D 高亮显示的矩形。

实例 597　Applet 中文字闪现动画

高级

光盘位置：光盘\MR\597

趣味指数：★★★

实例说明

本实例在 Applet 中实现了文字闪现的动画效果。运行程序，可以在浏览器窗口中看到文字闪现动画的效果，

即在小应用程序中文字时隐时现，并重复这个过程，如图 21.52 所示为文字闪现动画中显示文字的效果。

图 21.52　Applet 中文字闪现动画中显示文字的效果

关键技术

本实例主要是通过定义布尔型标记变量 flag，并通过该变量的值来决定在小应用程序中绘制的内容，如果 flag 的值为 true，则显示绘制的内容；否则，将绘制的内容设置为空字符串，即什么都不显示。实现该功能的代码如下：

```
if (flag) {                                          //flag 为 true
    flag = false;                                    //赋值为 false
    value = "JAVA 编程词典";                          //为 value 赋值
} else {
    flag = true;                                     //赋值为 true
    value = "";                                      //赋值为空字符串
}
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 TextFlashApplet 小应用程序，用于在其中实现文字闪现动画，该类的代码如下：

```
public class TextFlashApplet extends Applet implements Runnable {
    boolean flag = true;                             //标记变量
    String value = "";                               //存放绘制内容的变量
    public void start() {
        Thread thread = new Thread(this);            //创建线程对象
        thread.start();                              //启动线程对象
    }
    public void paint(Graphics g) {
        Image img = null;                            //声明图像对象
        img = getImage(getCodeBase(), "com/zzk/PD3.jpg");
        g.clearRect(0, 0, 310, 230);                 //清除绘图上下文的内容
        g.drawImage(img, 0, 0, getWidth(), getHeight(), this);
        Font font = new Font("华文楷体", Font.BOLD, 42);  //创建字体对象
        g.setFont(font);                             //指定字体
        g.setColor(Color.red);                       //指定颜色
        g.drawString(value, 30, 110);                //绘制文本
    }
    public void run() {
        try {
            while (true) {                           //读取内容
                Thread.sleep(150);                   //当前线程休眠 150 毫秒
                if (flag) {                          //flag 为 true
                    flag = false;                    //赋值为 false
                    value = "JAVA 编程词典";          //为 value 赋值
                } else {
                    flag = true;                     //赋值为 true
                    value = "";                      //赋值为空字符串
```

```
                }
                repaint();                                    //调用 repaint()方法
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 TextFlashApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.TextFlashApplet.class" width ="350" height = "200">
</applet>
</html>
```

秘笈心法

心法领悟 597：绘制两组文字交替显示的动画。

本实例实现了文字闪现动画特效，如果希望绘制两组文字交替显示的动画，可以将本实例中标记变量为 false 时，为变量 value 赋值为空，改为赋值为另一组字符串，即可实现两组文字交替显示的动画特效。

实例 598　Applet 中滚动广告字幕动画

光盘位置：光盘\MR\598　　高级　　趣味指数：★★★

实例说明

本实例在 Applet 中实现了滚动广告字幕的动画效果。运行程序，可以在浏览器窗口中看到滚动广告字幕的动画效果，即文字从小应用程序的右侧向左侧滚动，并重复这个过程，如图 21.53 所示为滚动广告字幕动画的一个效果。

图 21.53　Applet 中滚动广告字幕动画的一个效果

关键技术

本实例主要是通过调整绘制文本的横坐标值来实现滚动广告字幕动画效果的，调整横坐标值的实现代码如下：

```
if (x <= -440) {                                    //该条件可以根据需要自行调整
    x = 316;                                        //x 坐标定位到最右侧
} else {
    x -= 2;                                         //x 坐标左移
}
```

设计过程

（1）新建一个项目。

（2）在项目的 src 文件夹中创建包 com.zzk，然后在该包中创建一个继承 Applet 类的 RollTextApplet 小应

用程序，用于在其中实现滚动广告字幕动画，该类的代码如下：

```
public class RollTextApplet extends Applet implements Runnable {
    String value = "明日图书网的网址：http://www.mingribook.com";   //存放绘制的内容
    int x;                                                          //设置横坐标
    int y;                                                          //设置纵坐标
    public void init() {                                            //初始化方法
        x = 316;                                                    //存储绘制点的 x 坐标
        y = 190;                                                    //存储绘制点的 y 坐标
    }
    public void start() {
        Thread thread = new Thread(this);                           //创建线程对象
        thread.start();                                             //启动线程对象
    }
    public void paint(Graphics g) {
        Image img = null;                                           //声明图像对象
        img = getImage(getCodeBase(), "com/zzk/PD3.jpg");           //获取图片资源路径
        g.drawImage(img, 0, 0, getWidth(), getHeight(), this);      //绘制图像
        g.clearRect(0, 0, 0, 230);                                  //清除绘图上下文的内容
        g.setColor(Color.BLACK);                                    //定义颜色
        Font font = new Font("华文楷体", Font.BOLD, 20);            //创建字体对象
        g.setFont(font);                                            //定义字体
        g.drawString(value, x, y);                                  //绘制文本
    }
    public void run() {
        try {
            while (true) {                                          //读取内容
                Thread.sleep(50);                                   //当前线程休眠 50 毫秒
                if (x <= -440) {                                    //该条件可以根据需要自行调整
                    x = 316;                                        //x 坐标定位到最右侧
                } else {
                    x -= 2;                                         //x 坐标左移
                }
                repaint();                                          //调用 repaint()方法
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void update(Graphics g) {                                //重写 update()方法，防止闪烁
        paint(g);
    }
}
```

（3）在项目的 src 文件夹中创建一个名为 RollTextApplet.html 的文件，用于在浏览器中运行 Applet 小应用程序，该.html 文件的代码如下：

```
<html>
<applet   code = "com.zzk.RollTextApplet.class" width ="316" height = "200">
</applet>
</html>
```

秘笈心法

心法领悟 598：改变文字需要更改相应的代码。

在实际应用中使用本实例时，如果改变了其中的文字，例如改变了文字个数或字体的大小，这时就需要对 run()方法中 if 语句的条件表达式进行相应的修改，否则可能无法达到理想的效果。