Broadview
www.broadview.com.cn

面试宝典

Java程序员面试宝典

第2版

欧立奇 朱梅 段韬 编著

揭开知名IT企业面试、笔试的核心机密
传授程序员岗位求职的关键技巧

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Java 程序员面试宝典

第 2 版

欧立奇 宋梅 段韬 编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内容简介

本书是《Java 程序员面试宝典》的第 2 版。第 2 版在保留第 1 版数据结构、字符串处理、Java 程序设计等主干内容的基础上，大量更新了程序面试题目，内容主要取材于 2009 年到 2011 年各大公司的面试题，以反映第 1 版图书出版近两年来所发生的变化，目的是帮助求职者在面试过程中更好地处理一些新问题，应对新变化。

本书最后着力讲述了如何进行英语面试和电话面试，并对求职中签约、毁约的注意事项及群体面试进行了解析。本书的面试题除了有详细的解析外，对相关知识点也有扩展说明。希望这些内容对读者从求职就业到提升计算机专业知识有显著的帮助。

本书适合（但不限于）将要找工作的程序员、高校计算机类应届毕业生，以及其他计算机爱好者阅读。

图书在版编目（CIP）数据

Java 程序员面试宝典／欧立奇，朱梅，段韬编著. —2 版. —北京：电子工业出版社，2011.7
ISBN 978-7-121-13767-9

Ⅰ. ①J… Ⅱ. ①欧… ②朱… ③段… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2011）第 104835 号

责任编辑：李利健
印　　刷：
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京海淀区万寿路 173 信箱　邮编：100036
开　　本：787×980　1/16　印张：22.5　字数：576 千字
印　　次：2011 年 7 月第 1 次印刷
印　　数：5 000 册　　定价：46.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

本书是 Java 程序员面试宝典的第 2 版，同时也是《程序员面试宝典》的姊妹书。

第 2 版在保留第 1 版数据结构、字符串处理、Java 程序设计等主干内容的基础上，大量更新了程序面试题目，内容主要取材于 2009 年到 2011 年各大公司的面试题，以反映第 1 版图书出版近两年来所发生的变化，目的是帮助求职者在面试过程中更好地处理一些新问题，应对新变化。

本书相对上一版的变化主要有以下 3 点：

（1）针对求职过程这一章，我们在原版内容（笔试、电话面试、面试）的基础上添加了两节（签约、违约），以更好地帮助求职者应对求职过程中出现一些的细节和麻烦。

（2）针对 Java 程序设计这部分内容，我们更新了绝大部分的例题。如近两年随着 IDE 的频繁使用，异常处理问题、JVM 配置、运算符处理的问题却与日俱增，所以我们增加了这些内容。又如递归面试已经很少出现常见的菲波那契问题，而代之考验求职者如何提高循环递归效率问题。

（3）针对近两年面试过程中出现的新题型，本书补充了新的章节，如程序设计中的类型转换问题、泛型问题、白盒测试问题、数据结构中的排序问题、智力测试中的博弈测试等。

就编程而言，虽然 Java 和 C++大相径庭，但在更加有趣的语言后面的东西是设计模式、分析模式、求职解答、算法策略、信息化……也就是说，本书中追求的是程序员求职背后的一些东西，即对于技术的本质理解。所以本书虽命名为 Java 程序员面试宝典，但不仅限于对 Java 技术的单纯讲解。因为只有这样，求职者才能不被语言所羁绊，而对于一个企业而言，除了看中求职者对语言的熟练程度，更看重工作经验、大局观和整体架构等超脱语言的东西。

本书结构是一种问询式的结构。这样不仅言简意赅，平易近人，而且可以容纳更多的题目，真正达到宝典之效用。但本书又不简单作为一个题库出现，对一个类型的问题不简单加以重复。本书采用循序渐进的办法：（1）将重要概念加以复习；（2）完善解题思路，而不是仅仅给出答案；（3）给出完整可靠的答案，如果是可以验证的，要给出验证的结果；（4）综合几种解题方案，给出最优解；（5）触类旁通，给出语言背后的算法本质性解释。本书的解题思路不仅能够让大家知道什么是正确的解决方案，而且让大家明白怎样能获得最佳方案。

本书不同于其他 Java 程序书籍的主要特点如下。

◆ 细

中国软件企业比较小，面试涉及的方面比较多，且比较基础，比如常会考一些编程基础性的题，而原有的面试书籍对此方面鲜有触及。本书把面试中国内公司最易考到的基础考点，放在 Java 基础程序设计里面，希望能切切实实解决实际面试问题。

◆ 深

面试题通过一道题考一个专类方面的能力。说起 Java，人们首先想到的是 Java 编程语言，然而事实上，Java 是一种技术，它由 4 个方面组成：Java 编程语言、Java 类文件格式、Java 虚拟机和 Java 应用程序接口（Java API）。从面试者的角度来讲，一个测试也许能从多方面揭示应试者的素质，至少你能了解应试者所掌握类的水平。市面上流行的面试书籍在此专业性的分类方面做得不够，正因为如此，本书中追求的是程序员求职背后的一些东西：对于技术的本质理解，而不仅限于对 Java 技术的单纯讲解。

◆ 广

本书包括但不仅限于 Java 程序员面试试题。对求职市场面试试题做了妥善分类后，将面试对象分为软件开发人员、网络工程师、测试工程师和系统管理员。实际上，市面上流行的面试书籍仅对软件开发人员比较侧重，而忽略网络工程师和测试工程师，本书就这一方面给出了详细论断，并结合大量考题分析题目特点给出应试方案。本书将在这些方面做出改进，以适应市场需求。

同时本书对外企经常考到的 UML 及设计模式内容也做了深入的分析，从本质上诠释面试的真谛。

◆ 真

第 2 版在保留原书主干内容的基础上，内容非常时新，可以算做面试者求职前的一份全真模拟。同时作者将求职中的细节问题（简历，招聘，签约，违约），以及笔试、面试中的感悟融入书中，给求职者以最真切的人文关怀。真情实感，娓娓道来，指引读者走上理想的工作岗位。

本书不是一本万能书籍，但可以肯定是您工作与求职的好助手、好伙伴！

编著者

目　录

第 1 部分　求职过程

古人云：凡事预则立，不预则废。机会都是垂青有准备的人。为了得到一份满意的工作，大家一定要对整个求职过程有清醒的了解。把能够预见的、必须做的事情早一些做完，这样在大规模招聘开始的时候就可以专心地为面试做准备。求职过程中会发生很多预料不到的事情，当你的计划被这些事情打乱之后，要做的事会越堆越多，一步落后，步步落后。如果能够尽早把能做的事情做完，即便有计划外的事件发生，也不会产生太严重的影响。努力地使事态的发展处在自己能控制的范围之内，这样无论发生任何事都能有应对之策。

第 2 部分　Java 程序设计

本部分主要以 Java 设计语言为基础，通过大量实际的例子分析各大公司 Java 面试题目，从技术上分析面试题的内涵。一般公司的面试题都是两套：C++或 Java，面试者可以选择。

第 3 部分　数据结构和设计模式

本部分主要介绍求职面试过程中出现的第二个重要的板块—数据结构，包括字符串的使用、堆、栈、排序方法等。此外，随着外企研发机构大量迁入我国，外企针对软件工程知识的考核，包括设计模式、UML、敏捷软件开发，以及.NET 技术和完全面向对象语言 C#的面试题目将会有增无减，今后设计模式在面试中的比重会进一步提高。

第 4 部分 UNIX、Oracle、网络

本部分主要介绍求职面试过程中出现的第三个重要的板块—操作系统、数据库、网络知识。作为一个程序员，尤其是系统管理方面的程序员，对这几部分有深刻的理解和领悟是相当重要的。

第 5 部分 Java 开源

EJB 组件曾经被认为是一个重量级的组件。EJB 3.0 规范的重要目标就是简化 EJB 的开发，提供一个相对轻量级的组件方案。Spring 基于轻量内核，然后通过集成第三方的服务器来提供完整的架构。

第 6 部分 综合面试题

本部分主要介绍求职面试过程中出现的第五个重要的板块—英语面试、电话面试和智力测试。其中，英语面试不同于普通的英语面试。就一个程序员而言，最好能够用英文流利地介绍自己的求职经历，这是进外企非常重要的一步。此外，还必须对几个常用的问题有相关的解答，比如你最大的缺点是什么。有些问题即便是用中文，你都很难回答，更何况是用英文去回答。但是求职过程本身就是一个准备的过程，精心地准备，等待机会——机会总是垂青于那些精心准备的人。

如果你是一个具有战略眼光，且期待进入国际性跨国大企业的求职者，本章值得你仔细研读。

第 1 部分

求职过程

The procedure of applying for a job

本部分将详述作为一个计算机专业的应届毕业生或程序员，在求职面试中应该注意的一些问题。

古人云：凡事预则立，不预则废。机会都是垂青有准备的人。为了得到一份满意的工作，大家一定要对整个求职过程有清醒的了解。把能够预见的、必须做的事情早一些做完，这样在大规模招聘开始的时候就可以专心地为面试做准备。求职过程中会发生很多预料不到的事情，当你的计划被这些事情打乱之后，要做的事会越堆越多，一步落后，步步落后。如果能够尽早把能做的事情做完，即便有计划外的事件发生，也不会产生太严重的影响。努力地使事态的发展处在自己能控制的范围之内，这样无论发生任何事都能有应对之策。

第1章

应聘求职

每年的9月到次年的1月，都是应届生求职、在职人员跳槽的高峰期。对于即将成为程序员的应届毕业生们，在求职过程中怎样确定目标公司和目标职位；对于已经是程序员的跳槽大军，是按照技术路线发展自己的职业生涯，还是走向管理岗位继续自己的职业道路，或者是改变自己的发展轨迹；大家在求职过程中要注意哪些细节？这些都是大家所关心的话题。

国内的IT业比国外兴起得晚，而且目前还没有权威的适合中国本土程序员的职业生涯发展规划。因此，国内流行的“35岁退休说”其实是一种误解，只要我们好好规划自己的职业生涯，提高自己的技术水平、沟通技巧和管理能力，就能够获得更高、更好的职位，完全可以像国外的程序员一样工作到60岁再退休。

让我们先从应聘流程中的注意事项，这个轻松却又容易被人忽略的话题开始吧。

1.1 应聘渠道

对于应届生而言，可以选择参加校园宣讲会的形式投递简历。如图1-1所示，这是EMC公司2006年校园宣讲会日程表。我们可以选择就近的城市参加它的宣讲会并投递简历。

招聘会投递的简历是“纸”的简历。尽管现在网上投递电子简历的方式非常流行，但是“纸”的简历仍然有着其无可比拟的优势。HR（人力资源经理）拿到“纸”的简历，相比一份电子简历更有一种亲切感，重视程度也较电子简历高一些。

第二种方式是投递电子简历，可以通过公司的电子信箱和公司网站招聘信息栏（数据库），以及各大招聘门户网站，如中华英才或者智联招聘等，来投递自己的电子简历。

EMC 2006 校园宣讲会行程表

日期	时间	城市	学校	地点
2006.04.03	19:00-21:00	北京	北京大学	英杰交流中心阳光大厅
2006.04.04	19:00-21:00		清华大学	就业指导中心多功能厅
2006.04.05	19:00-21:00	南京	东南大学	群贤楼报告厅
	19:00-21:00	成都	成都电子科技大学	报告厅
2006.04.06	19:00-21:00	西安	西安交通大学	就业指导中心信息发布厅
	19:00-21:00	上海	上海交通大学	光华楼（闵行物区）
2006.04.07	15:00-17:00		上海复旦大学	登辉讲报告厅

图 1-1　EMC 公司 2006 年校园宣讲会日程表

1.2　应聘流程

应聘的一个完整流程如图 1-2 所示。

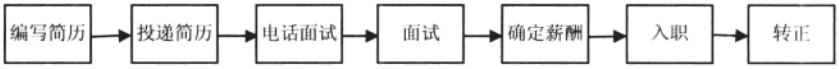
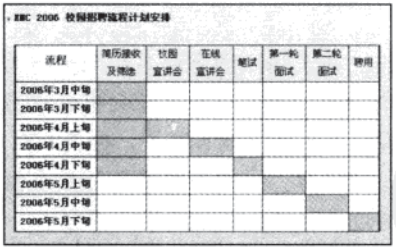


图 1-2　完整的应聘流程

通常，一个外企的应聘流程是一个很长的过程，有时甚至可能达到一两个月。还以 EMC 公司为例，如图 1-3 所示，让我们看一下该公司的应聘流程。



EMC 2006 校园招聘流程计划安排

流程	简历接收及筛选	校园宣讲会	在线宣讲会	笔试	第一轮面试	第二轮面试	聘用
2006年3月中旬							
2006年3月下旬							
2006年4月上旬							
2006年4月中旬							
2006年4月下旬							
2006年5月上旬							
2006年5月中旬							
2006年5月下旬							

图 1-3　EMC 公司的应聘流程

由图 1-3 可知，比较正规的企业的应聘流程一般分为 5 个部分：简历筛选、笔试、第一轮面试（含电话面试或邮件面试）、第二轮面试、发 Offer。在下面的章节中我们会就这 5 个步骤做详细的阐述。

第2章

简历书写

据统计，80%的简历都是不合格的。不少人事管理者抱怨收到的许多简历在格式上很糟糕。简历应该如何做到在格式上简洁明了，重点突出？求职信应该如何有足够的内容推销自己？如何控制长度，言简意赅？相信读了本章，你会对简历的撰写有一个新的认识。

2.1 简历注意事项

1．简历不要太长

一般的简历普遍都太长。其实简历内容过多反而会掩盖一些有价值的闪光点。而且，每到招聘的时候，一个企业，尤其是大企业会收到很多份简历，工作人员不可能都仔细研读，一份简历一般只用1分钟就看完了，再长的简历也超不过3分钟。所以，简历要尽量短。我们做过一个计算，**一份中文简历压缩在两页左右就可以把所有的内容突出了。**一页显得求职者过于轻浮，三四页就太多了。

简历过长的一个重要原因是有的人把中学经历都写了上去，其实这完全没有必要，除非你中学时代有特殊成就，比如在奥林匹克竞赛中获过奖。一般来说，**学习经历应该从大学开始写起。**

很多学生的求职简历都附了厚厚一摞成绩单、荣誉证书的复印件，其实简历上可以不要这些东西，只需要在简历上列出所获得的比较重要的荣誉即可。如果企业对此感兴趣，会要求求职者在面试时把这些带去。

2．简历一定要真实客观

求职简历一定要按照实际情况填写，任何虚假的内容都不要写。即使有的人靠含

有水分的简历得到面试的机会，面试时也会露出马脚的。千万不要为了得到一次面试机会就编写虚假简历。被招聘方发现后，你几乎就再也没有机会进入这家公司了。而且对于应届生来说，出现这种情况后，还有可能影响到同校的其他同学。

北京某高校一位计算机专业本科毕业的女孩子，简历上写的是2009年毕业，但面试中被发现她是2010年毕业的，而且没有任何工作经验。这个女孩儿比较诚实，说是同学教她这样做的。

她这种编制虚假简历的做法应该否定，因为谁都不希望被骗。作为面试官来说，首先希望应聘者是一个诚实的人。我希望她在听到同学那个不明智的建议时，首先不应选择这种做法，其次要尽力阻止其他人这样做。因为，就像面试官代表公司形象一样，她在某种程度上也代表了她所毕业的学校来参加面试！最起码她传达给HR的信息与她同专业应届生的简历所表述出的信息相比，其可信度较差。

3．不要过分谦虚

在简历中不要注水并不等于把自己的一切（包括弱项）都要写进去。有的学生在简历里特别注明自己某项能力不强，这就是过分谦虚了，实际上不写这些并不代表说假话。有的求职学生在简历上写道："我刚刚走入社会，没有工作经验，愿意从事贵公司任何基层工作。"这也是过分谦虚的表现，这会让招聘者认为你什么职位都适合，其实也就是什么职位都不适合。

4．简历要写上求职的职位

在求职简历上一定要注明求职的职位。每份简历都要根据你所申请的职位来设计，突出你在这方面的优点，不能把自己说成是一个全才，任何职位都适合。**不要只准备一份简历，要根据工作性质有侧重地表现自己。**如果你认为一家单位有两个职位都适合你，可以向该单位同时投两份简历。

在我曾看到的一些简历中，经常有如下的错误：简历上描述的多为Windows操作系统下C/C++开发经验，但申请的目标职位为"Linux操作系统下的C/C++开发工程师"。这样当然不容易得到应聘职位的面试机会。还有就是去应聘ERP、CRM方面的职位，而简历里却大肆强调自己在嵌入式编程方面的优势。就算你非常优秀，你对这个企业还是没有用处。

有些简历里面没有详细的项目描述及责任描述，在责任描述栏仅仅填写"软件开发"或者在工作业绩栏仅仅填写"可以"两字。这样的信息传达无疑是不成功的。

作为求职的开始，我们**要编写一份或者几份有针对性的简历，也就是按照对方的要求突出自己相关的经历。**只要你的优势与招聘方的需要吻合，并且比其他应聘者突出的话，你就胜利了。

5．在文字、排版、格式上不要出现错误

用人单位最不能容忍的事是简历上出现错别字或者在格式、排版上有技术性错误，以及简历被折叠得皱皱巴巴、有污点，这会让用人单位认为你连自己求职这样的事都不用心，那工作也不会用心。

6．简历不必做得太花哨

一般来说，**简历不必做得太花哨，**用质量好一些的白纸就可以了，尽量用 A4 规格的纸。我曾看到过一份简历封面上赫然写着 4 个大字“通缉伯乐”，给人的感觉就像是在威胁用人单位。现在学生简历中比较流行做封面的形式，其实没有必要，这会增加简历的厚度，实际上完全可以不用封皮。

7．简历言辞要简洁直白

大学生的求职简历很多言辞过于华丽，形容词、修饰语过多，这样的简历一般不会打动招聘者。简历最好多用动宾结构的句子，简洁直白。

8．不要写上对薪水的要求

在简历上写上对工资的要求要冒很大的风险，最好不写。如果薪水要求太高，会让企业感觉雇不起你；如果要求太低，会让企业感觉你无足轻重。对于刚出校门的大学生来说，第一份工作的薪水不重要，不要在这方面费太多脑筋。

9．不要写太多个人情况

不要把个人资料写得如此详细，姓名、电话是必需的，出生年月可有可无。如果应聘国家机关、事业单位，应该写上政治面貌。如果到外企求职，这一项也可省去，其他都可不写。

10．不要用怪的字体

笔者见过一份简历，用中空字体，还有斜体字，这些都是很忌讳的。试想：一个 HR 挑了一天的简历，很累了，还要歪着头看你的简历。你想你的胜算能有多大？其实用简单的宋体 5 号字就很好了，不用标新立异。

2.2 简历模板

一份合格的求职简历应该包括以下内容：

姓名、电话（或其他联系方式）等个人资料应该放在简历的最上面，这主要是为了方便用人单位与求职者及时取得联系。紧接着是毕业的学校、专业和时间。下面应

该注明应聘的职位和目标。

接下去就是简历上最重要的部分：工作经历。对于初出茅庐的大学生来说，这部分包括勤工助学、课外活动、义务工作、参加各种各样的团体组织、实习经历和实习单位的评价等。这部分内容要写得详细些，指明你在社团中、在活动中做了哪些工作，取得了什么样的成绩。用人单位要通过求职者的这些经历考查其团队精神、组织协调能力等。

兴趣爱好也最好列上两三项，用人单位可就此观察求职者的工作、生活态度。

如果应聘外资企业、大的跨国公司，一定要附上英文简历，而且要把最近的经历放在最前面，简历前面最好附一封推荐信。一定要认真对待英文简历的编写，因为它会泄漏你的实际英文水平。

下面是一份简历模板。

求 职 简 历

姓 名	柯小平	**性 别**	男	**出 生 日 期**	1985/03/19
学 校 及 专 业	西北大学计算机系软件与理论专业				
学　　历	硕士				
移 动 电 话	13096964884	**电 子 邮 件**	jinder24@263.net		
IT&英语技能	1．软件结构设计，需求分析能力。 2．精通 Java、JSP。 3．熟悉 Windows 开发平台，精通 J2EE 体系。 4．熟悉 Eclipse 开发工具，熟悉 UML 统一建模语言。 5．深入理解面向对象的思想，并能熟练地应用于具体的件设计开发工作中。 6．英语水平：国家六级 598 分。				
项目经验（近期）	2009/1—2009/7 搜猪网西安站点运营与技术开发总监助理 1> 深入参与项目的前期准备、需求分析和人员招聘工作，组建最初的 9 人工作团队。 2> 深入参与项目的开发，负责系统分析与架构设计以及部分编码工作，领导团队按时保质完成前台页面、业务逻辑和后台数据库的构建、测试工作。 3> 独立开发的新闻信息自动采编系统解决了 12000 条初始内容填充的手工工作量，完全解决站点最初建设的内容空白问题，加快了网站上线的时间进度。 4> 与营销团队共同制定了“西安商圈”栏目，成功创建了西安地区的商业黄页系统。				

	2009/7—2009/9 与实验室人员合作，在基于 J2EE+ SQL Server 2003 的平台下开发西北大学人事管理系统程序。该项目是西北大学基金项目，目的为完成西北大学教职员工信息的统一规范化管理。系统分为教师科、劳资科、人事科、人才交流中心等几部分，实现了各个部门之间的信息统一化协调管理。该系统使用 Eclipse、Power Designer、MS SQL 开发完成。
工作经验	2007/7 至今 西北大学 就读于西北大学计算机系，在学习期间，与实验室小组合作完成网上选课系统（JSP+ SQL Server 2003）和人事管理系统（J2EE+ SQL Server 2000）的研发工作，以及教务管理系统（PowerBuilder 8+ SQL Server 2000）的测试工作。
奖学金	中国石油奖学金优秀学生一等奖。
其他特长	文学和美术功底较好，擅长网页制作，Photoshop 和 Dreamweaver 应用水平较好。擅长表述，能够胜任教学工作。
个人评价	我无法掩饰我对这份工作的渴望——一份有科研挑战性的职位。作为老师，我喜欢传道、授业、解惑，形成一套自己的理论并潜移默化地影响着我的学生；同样，作为科研工作者，我也被 Java 的华贵多彩而吸引，那是真正的逻辑之美。此外，很多时候为了项目的完满，必须具备一种“不破楼兰终不还”的决心和“不积跬步无以至千里”的恒心。 我谦和、谨慎，富有团队精神。希望您能给我这样一个机会展示自己。 谢谢! 柯小平 2010.1.18

Resume

Name	Xiaoping Ke	**Gender**	Male
School & Major	Northwest University Master of Computer Software and Theory		
Mobile phone	13096964884	**E-mail**	jinder24@263.net
Education	Master		
Career Objective	To obtain a challenging position as a software engineer with an emphasis in software design and development.		
Computer Skills &	Languages: Java,JSP, XML, UML		

English Skills	Systems: Windows, UNIX Database: MS SQL Server, Oracle English Skills: CET-6
Recent Experience	2010/1—2010/7 Northwest University Personnel Managing System Northwest University Personnel Managing System is a system of automanage the personnel information. It is designed by PowerDesigner. The project is a C/S architecture system. It is based on a Microsoft SQL database, and the UI is developed by Delphi 7. In this project, I designed the schema of database, programmed database connectivity using Delphi 7. 2009/9—2010/1 Northwest University Network Course-selected System Based on J2EE+SQL 2000 we finished Northwest Network Course-selected System. Everybody in this University can select, cancel, query course in network. The project is a B/S architecture system; the code is developed by java, and runs on the J2EE plat. In this project, I used the JDBC interface which is provided by the database program. And after that, I joined the testing of whole system.
Self Assessment	I am an active, innovative man, a good team-worker, with rich IT knowledge and developing experience. I am fit for a job of programming in an IT company.

第3章

求职五步曲

笔试、电话面试和面试是顺利求职的 3 个过程。三关全过才能顺利签约，只要有一关没能通过，就会被“刷”掉。除此之外，签约本身又何尝不是一个重要的考试？涉及你的未来、人生、行业甚至家庭。当然有签约就有可能会有违约，真希望你们不必走第五步，但是这个世界毕竟不是童话。

3.1 笔试

笔者认为笔试是程序员面试的 3 个过程中最重要的一个环节，也是最难以提升的一个环节。本书中主要叙述的也是程序员的笔试经历。不论你有多么大的才干，多么广博的知识，如果未能通过笔试，则无缘下面的进程。表 3-1 描述了各类 IT 公司笔试所考题目的类型。

表 3-1 各类 IT 公司笔试所考题目的类型

公司名称	公司类型	笔试内容
Trend	网络公司	C++ 或 Java，网络，数据库，设计模式，智力测试，英语阅读
SAP	软件咨询，ERP，CRM	C++，概率问题，设计模式，智力测试
Advantech	硬件，自动化公司	C++（尤其是指针问题），嵌入式编程
Synopsys	电子类公司	C++（尤其是指针问题），数据结构
NEC	综合软件公司	C，数据结构
金山	综合软件公司	C++或 PHP，数据库，数据结构，设计模式
华为	通信公司	C++或 Java，数据结构，数据库
中兴	通信公司	C++或 Java，数据结构，数据库

续表

公 司 名 称	公 司 类 型	笔 试 内 容
VIA	硬件公司	C++（尤其是指针问题），嵌入式编程
华为 3COM	网络公司	C++，网络
SPSS	数据统计软件公司	C++（尤其是继承、多态问题），数据结构
Sybase	数据库公司	C++，Linux，UNIX
Motorola	网络公司	C++，网络
IBM	综合软件公司	C++或 Java
Oracle	数据库公司	Java，数据库
HP	综合软件公司	C++
腾讯	综合软件公司	C++
Yahoo	综合软件公司	C++或 Java 或 C#
微软	综合软件公司	C++，数据结构，智力测试
神州数码	金融软件公司	C++或 Java，数据结构，数据库（SQL）
大唐移动	通信公司	C++
Siemens	数据通信公司	C++，设计模式
Grapecity	软件公司	C++，C#，智力测验

根据上表，对各大 IT 公司的笔试题目和所考的内容，我们可以窥见一斑，并得出以下几个结论。

1．语言的偏向性

综合表 3-1 所示，**IT 公司笔试在编程语言上有一定偏向性，以 C、C++为主，或者以 Java 为主。**语言本身并没有什么高低贵贱之分，但相对来说，考到 Delphi 或者 VB 的可能性很小。作为应届毕业生，如果只学过 VB、VF，却从来没有接触过 C 系或 Java 语言，则在笔试中是比较吃亏的。

2．英语的重要性

笔者所接触过的**外企的笔试试卷基本上都是英语试卷，**无论从出题到解答，都要求面试人员用英文回答，所以必须有很好的英文阅读能力，这也是外企招人对英语非常看重的原因。其实也不需要英语一定要通过六级，但**一定要有相对多的单词量，能够看懂考题的意思。**然后按自己的想法组织语言来描述就可以。

国内企业一般对外语要求不是很看重，题目也是中文的。如果不想进外企的话，也不用刻意准备英语。

3．智力测试

之所以要强调这一点，是和市面上过度强调外企智力测试有关。实际上，在笔者参加过的微软等外企笔试中，智力测试只占很小的比例，约 3%～5%。而华为、神州数码等国内 IT 企业基本上没有智力测试，完全是技术考试。所以，奉劝大家**不要把精力都投在所谓的外企智力测试上面，**还是应该以准备技术方面的笔试为主。

4．有的放矢准备简历

不同的公司会考不同的内容，这就像高中时准备不同科目考试的差别。比如，神州数码不会考嵌入式编程，而 VIA 考设计模式的可能性很小。一般有点儿偏“硬”的 IT 公司会对 C++中指针的用法、数据结构考得比较多。偏“软”的企业会对设计模式、模板着重一些。所以本书分得很细，力求对各种 IT 公司的笔试题目做一个详尽的阐述。

作为求职者，**笔试前，你首先要搞清这个公司的基本情况，**它是做什么的，它有什么产品，你是学什么专业的，有的放矢才能获胜。

5．纸上写程序

搞计算机的肯定不习惯在纸上写程序，然而技术面试的时候，这是面试官最常用的一招。让写的常见程序有：数据结构书上的程序、经典 Java 程序（字符串分解、冒泡排序等）。第一次在面试官眼皮底下在纸上写程序，思路容易乱。建议大家事先多练习，找个同学坐在边上，在他面前写程序，把该同学当成面试官。经过多次考验，在纸上写程序就基本不慌了。

每次面试总会有些问题回答得不好，事后一定要总结，把不懂的问题搞明白。例如，一个求职者就碰到两家公司问了同样的问题，第一次答不出，事后没查，在第二家公司又被问到同样的问题，这当然是很郁闷的事情。

3.2 电话面试

电话面试主要是对简历上一些模糊信息的确认、之前经历的验证、针对应聘职位简单技术问题的提问，以及英文方面的考查。

由于模式的限制，电话面试时间不会很长。在这个环节中，**一定要表现得自信、礼貌、认真、严肃，**这样会在声音上给对方一个良好的印象。如果声音慵懒，语气生硬，除非是技术题目及英文方面表现得足够好，否则很难予以平衡。

在回答电话面试的问题时，不要紧张，要留心对方的问题，这些问题也许在见面的面试中还会再出现。如果对方在电话面试中要求你做英文的自我介绍，或者干脆用英文和你对话，那么在电话面试结束后一定要好好准备英文面试的内容。

笔者曾经参加过 Sybase、SAP、麒麟原创等公司的电话面试。外企一般都会要求你做一个英文自我介绍和用英文回答一些小问题，总的来说，不会涉及太多技术方面的问题，因为用英语来描述技术对国人而言还是有一定困难的。国企会问到技术问题，笔者就曾被问到如何在 C++中调用 C 程序、索引的分类等技术问题，回答基本上要靠平时的积累和对知识的掌控能力。

3.3　面试

一个比较好的面试是能够问出求职者擅长哪方面或哪方面不足的。如果面试官针对求职者的不足之处穷追猛打，或者炫耀自己的才能，这是不足取的。

对于求职者而言，面试是重点环节，**要守时**是当然的。如果不能按时参加面试，最好提前通知对方。着装上不需要过分准备，舒服、干净就好了。一般的 IT 公司对技术人员都不会有很高的着装要求。虽然着装不要求，但精神状态一定要好。饱满的精神状态会显得你很自信。

若有笔试（有时笔试和面试是同时进行的，即面试官会在提问后请你回答并写下详细描述），也无非是与应聘职位相关的技术考查或者英文考查，如英汉互译等。应视你应聘职位的等级进行准备。

初级职位会针对你的编程能力和以往的项目经验进行重点考查。如果面试官针对你做的某个项目反复提问，那么你就需要注意了，要么是面试官在这个方面特别精通，要么就是未来的职位需要用到这方面的技术。我们应该抱着一种诚恳的态度来回答，对熟悉的技术点可以详细阐述，对于不熟悉的部分可以诚实地告诉面试官，**千万不要不懂装懂。**不过，我们同意**可以引导与面试官的谈话，尽量把他引到我们所擅长的领域。**在 SPSS 公司面试时，在回答完面试官的单链表逆置和复制构造函数问题之后，我把话题引入了我所擅长的设计模式方面，这是一种谈话的艺术。

应聘中级职位时，不但会考查代码编写，而且会对软件架构或相关行业知识方面进行考查。代码编写方面，主要以考查某种编程技巧来判断你对代码的驾驭能力。比如，某国际知名软件公司经常会让面试者编写 Java 异常函数。越是简单的函数越能考验应聘者的编码能力。你不但要实现功能，而且还要对可能出现的错误编写防御性代码，这些经验都需要在实际编程过程中积累。

应聘高级职位时，应聘者肯定对技术或某个行业有相当程度的了解，这时主要是看你与职位的契合程度、企业文化的配比性（即将人力资源及成本配比作为服务体系的重要组成部分，将公司企业文化中核心理念及价值观作为客户服务的重要媒介）**及整体感觉。**应聘管理职位的话，考查的更多是管理技巧、沟通技巧和性格因素。应聘架构

师一般会考查行业背景与软件架构方面的知识，比如 UML 或建模工具的使用等。应聘技术专家的职位，则会针对相关技术进行深度考查，而不会再考查一般性的编码能力。

面谈的时候，要与面试官保持目光接触，显示出你的友好、真诚、自信和果断。如果你不与对方保持目光接触，或者习惯性地瞟着左上角或者右上角，会传达给对方你对目前话题表现冷淡、紧张、说谎或者缺乏安全感的感觉。

如果对方问到的某个问题你不是很熟悉，有一段时间保持沉默时，请不要尴尬和紧张。**面试过程中允许保持沉默，你完全可以用这段时间来思考。**可以用呼吸调整自己的状态。如果过于紧张，可以直接告诉对方，表达出自己的紧张情绪，能够起到很好的舒缓作用，而且紧张本来也是正常的表现。

在面试过程中，应聘者也保有自己的权利。比如面试时间过长，从上午一直拖到下午，而你未进午餐就被要求开始下午的面试的话，你完全可以要求进餐后再开始。面试是一个双方信息沟通及达成合作目的的会谈，是一个双方彼此考量和认知的过程。**不要忽略自己应有的权利。**

面谈后，如果对方觉得你在技术、沟通、态度各方面都不错，也许会增加一个素质测评，确认一下对你的判断。

素质测评一般考查性格、能力、职业等方面，以判断你的价值观是否与企业相符。我们不需要去猜测这些题目到底要考查些什么，凭着你的第一感觉填写就可以了。在几十道甚至上百道题目中，都有几道题是从不同角度考查一个方向的，凭猜测答题反而会前后有悖。

当然，要先看清楚题目，搞清楚是选择一个最适合你自己的，还是选择描述得最不恰当的。在通过面试之后，如果有多家公司和职位的 Offer 可以选择的话，我们可以将公司的行业排名、公司性质、人员规模、发展前景、企业文化、培训机制，结合自身的生活水平、职业生涯发展规划来进行排列，选出最适合自己的公司和职位。

建议准备一个日程本，记录每一次宣讲会、笔试和面试的时间，这样一旦公司打电话来预约面试，可以马上查找日程本上的空闲时间，不至于发生时间上的冲突。每投一份简历，记录下公司的职位和要求，如果一段时间以后（1 个月或更长）有面试机会，可以翻出来看看，有所准备。**根据不同的公司，准备不同的简历，**千万不要一概而论，不同的公司在意的东西不一样。每参加完一次笔试或面试，把题目回忆一下，核对一下答案，不会做的题目更要好好弄懂。同学之间信息共享，总有人有你没有的信息。如果投了很多份简历，一点儿回音都没有，你就得好好看看简历是否有问题，以便增加一些吸引 HR 眼球的内容。

3.4 签约

首先向你表示衷心的祝贺！如果到了签约环节，那说明你已经顺利通过了笔试、

面试，拿到了 Offer。一般来说，面试成功后，就会有口头 Offer 或者是电话 Offer 了。正式的 Offer 应该提供以下几项：

1）薪水（税前还是税后）
2）补助（税前还是税后）
3）工作职位
4）工作时间、地点
5）保险公积金等福利

常见格式如下（美国 VVNT 公司发给笔者的邮件 Offer）：

Dear 欧立奇，

Please be informed this offer letter will be taken effect only on fulfillment of all the following condition:
You have passed the medical check up at a hospital in Xian designated by VVNT

OFFER LETTER

We are pleased to offer you the position of Software Developer at Band 4 level in Research & Development department, reporting to the / , on the following terms and conditions:

Working Location:
At one of VVNT's R&D laboratories within Xian City (work location may change due to work requirements)

Office Hours:
Office hours in Xian are 8:30am to 5:30pm, lunch break is 12:00pm - 13:00pm, Monday through Friday.

Probation Period:
Three-month probation starting on _____

Contract Period:
Three-year contract period from _____ to _____

Compensation and Benefits:
1) Your Basic Salary (before tax) will be RMB 5990 yuan per month.
2) Your Gross Allowance (after tax) will be RMB 2010 yuan per month.
3) You will be entitled to social insurance and housing fund according to applicable government regulations.
4) Subject to VVNT's discretion, you will be entitled to participate in VVNT's commercial insurance program, details of which are set out in VVNT's Guideline of Commercial Insurance Benefit (as may be changed from time to time), an updated copy of which can be seen on the VVNT intranet website.
5) You will be entitled to annual leave according to VVNT's leave management policy.
6) You will be entitled to all statutory public holidays.

If the above is agreeable to you, please sign and return the attached duplicate copy of this letter indicating your acceptance of the position and the terms and conditions set out the above.

Looking forward to working with you!

Signed by:

HR -Manager_Xian　　New Hire

Date:　2008-03-19

欢迎您加入 VVNT 研究开发中心，请准备以下资料在拟定的时间到人力资源部报到。

收到 OFFER 后 2 天之内：
以邮件形式确认是否接受 OFFER
提供英文名（为了防止重名，请另外提供至少 3 个备选英文名）

正式入职后的第一周以邮件形式提供以下规格的照片至此电子信箱 eihuang@VVNT.com.cn：
　　数码彩色照片（蓝底，尺寸不限）1 张

入职当日必须提供以下资料，否则此录用信失效并不予办理入职手续：

1）详细简历 1 份。
2）彩色照片（1 寸，底色不限）2 张。
3）毕业证、学位证、英语考级证原件及复印件 1 份。
4）若已参加职称评定，提供职称证书；若已婚，提供结婚证原件及复印件 1 份。
5）离职证明原件及复印件 3 份。
6）身份证原件及复印件 5 份（若身份证为二代证，请同时提供身份证背面复印件）。
7）到 VVNT 报到前所任职的公司的劳动合同原件（个人所保留件）或社会保险记录原件。
8）如户口是西安市户口，请准备户口本（个人姓名页）复印件 1 份。
9）若户口是西安市户口并已经购买社会保险，请提供劳动手册原件或失业证。
10）英文名（请于接受 OFFER 当日的 2 天内，以电子邮件的方式提供）。
11）西安市区任何国内银行存折账号作工资账号使用，报到当天请携带存折原件。
关于复印件：A4 纸复印，　请勿在一张 A4 纸上复印多份证件。
关于英文名：为避免重名，　请多准备几个名字作为候补一同提供。

VVNT 西安研究开发中心

人力资源部

在签约前，一定要向 HR 或其他人打听清楚以下信息：

1）户口。

要问清楚这个单位是“保证解决户口”、“尽力解决户口”、“不保证解决户口”还是“不管户口”。尤其是在进行校园招聘时，对于签约北京、上海单位的同学，这点非常重要，因为北京、上海对于外地人落户非常严格。

一般来讲，大多数国企、事业单位、研究所、公务员都是有能力解决户口的，但是外企和私企解决户口的能力与前面的单位比要差很多，但是不同的单位也有很大的差别，像 IBM、华为每年就能拿到很多名额。所以，对于这些单位，更要问清楚，到底有多大可能性解决户口。如果企业不能解决户口，你就只能办理临时居住证。

如果你想在一个城市长期发展的话，户口的作用是非常大的，以北京为例：如果没有北京户口，当你想跳槽时，会发现能选择的单位很有限，因为很多单位招人时，往往都要求北京生源、北京户口。这是户口带给我们的直接影响，从长远看，还有结婚、出国、子女就学、业务往来等各方面都会受到影响。当然，如果你将来想出国，或不想在北京常住，那么户口可能就不重要了。

所以，对于大多数人来说，要想获得北京、上海户口，基本上只有毕业这一次机会。这点，请一定要想清楚。需要特别说明的是，对于那些“尽力解决户口”、“不保证解决户口”的单位，跟你签了协议，实际上你就要承担一定风险。一旦最后没给你落户，大多数情况下，户口和档案会被打回原籍，因为那时再签约别的单位就会比较麻烦。

在日益激烈的就业形势下，户口和薪水很难两全，既解决户口、薪水又高的单位是很少的。一定要在两者中间权衡轻重，不要作出让自己后悔的决定。

2）待遇。

待遇是签约前必然要谈的部分。这里面的因素非常多，待遇主要包括：工资、奖金、补贴、福利、股票（期权）、保险、公积金。以下具体介绍各部分应注意的细节。

<1>工资：一定要问清楚是税前还是税后，这点不用多说。另外，还要问清楚，发多少个月。例如：税前工资 7000 元，发 13 个月，则年收入 7000 元×13 元=91000 元。很多单位有年底双薪，还有一些单位会发 14～16 个月不等。

<2>奖金：很多单位奖金都占收入很大一部分，例如：联想、百度、中航信都有季度奖、年终奖，另外还有项目奖，华为也有项目奖、年终奖，瞬联就没有奖金。不同的单位情况不同，奖金的数额也不一样，通常几千至数万不等，所以，关于这一点，一定要问清楚，而且要确定能拿到的奖金，取最低数。

<3>补贴：有些单位会有各种补贴，例如：通信补贴、住房补贴、伙食补贴等，例如：华为有 800～1000 元的餐补。有些单位的这些补贴加在一起，收入会非常可观，这也要问清楚。

<4>福利：对于一些国企和事业单位来说，往往会有一些福利，例如：过节费、防暑降温费、取暖费、购物券、电影票、生活用品等。

<5>股票：对于很多公司来说，股票是它们提供的非常有诱惑力的福利，一般来说，已经上市的公司提供股票的可能性不大，反倒是一些即将上市的公司提供股票的可能性很大，对此，一定要看准机遇，不要轻易错过。

<6>保险、公积金：即常说的“五险一金”。“五险”指的是养老保险、医疗保险、失业保险、人身意外伤害保险、生育保险，“一金”指的是住房公积金。这些是国家规定的，企业不得以任何理由拒绝为你缴纳，而且个人和企业出的比例是有规定的（但是也有一些企业拒绝缴纳公积金的例子）。这里要注意的是缴费基数。很多单位在这上面做文章，例如：你的工资是 5000 元，他们以 2000 元为缴费基数，也就是说，用它去乘以固定的比例给你缴纳五险一金，对此，一定要注意问清楚缴费基数。有些单位公积金比例上得非常高，所以你工资扣得也很多，那意味着公司交的钱更多，而一旦买房时，这些钱都是你自己的，所以，这部分收入不能忽视。此外，有些单位还会向你提供补充医疗保险、补充养老保险、补充意外保险、住房无息贷款或经济适用房等，也要问清楚。

把这些收入加起来，得到年收入。然后考虑工作地的工资水平和消费水平。例如：年薪 8 万在西安，无疑是比年薪 10 万在上海要高多了。

<7>年假：即每年除了法定节假日之外可以休息的天数，这个自然是高校最多（寒暑假），研究所、外企可能会少些，比如 Ppform 公司一年是 15～20 天年假，30 天探亲假（不可以同时休）；Nortel 是第一年 12 天年假，然后每年递增，直到 21 天为止；华为没有年假，要靠每月最后一天周六加班来攒假期作为自己的年假。不上班的时候觉得假期无足轻重，上了班就会觉得假期弥足珍贵。

3）工作内容。

要问清楚自己的具体职位，这个职位的工作内容在公司所处的地位。一般来讲，如果是核心业务部门，公司会比较重视，发展前景会更好，如果是其他辅助部门，可能受重视程度会差一些，当然没有绝对的，关键还是看你的工作有没有技术含量，对于你个人能力的提高、职业生涯有没有帮助，对于你跳槽、升职有没有帮助。

4）加班/出差情况。

对于有些公司来说，加班是在所难免的，如：华为、中兴、微软、IBM……基本上绝大多数 IT 企业都要加班；而对于有些职位来说，频繁地出差是在所难免的，例如：现场工程师、市场、销售等。对于这些问题，要提前有所了解，有思想准备，像中兴海外可能会派到非洲若干年，条件很苦。如果自己不能忍受长期的加班、出差，建议不要签。另外，要问清楚加班是否有加班费。现在很多公司加班都是没有加班费的，对于加班，国家有规定：如果周六、周日加班，可以获得正常工资 2 倍的加班费；如果是五一、十一这些法定假日加班，可以获得正常工资 3 倍的加班费。另外就是出差补贴，一般来讲，

出差基本是不需要你花钱的，而且很多公司会有额外的出差补贴，例如：华为非洲区好像是每天补助40～70美金不等。这个也要问清楚，因为都是自己的合法权益。

5）培训。

对于应届毕业生来说，公司的培训体系是一个非常重要的考虑因素，如果一家公司有非常好的培训体系，那么可以让你在几年内迅速成长为一个出色的人才，对你的职业生涯无疑是有巨大帮助的。像宝洁、SAP、infosys，最出名的都是它们完善的培训体系，确实可以让你的个人能力在短时间内得到极大的提高，所以，它们每年能吸引那么多同学去应聘。从某种程度上来讲，良好的培训是比优厚的待遇更有吸引力的。所以，在签约前，一定问清楚单位有哪些培训计划，再看这些培训计划对个人的成长是否有帮助。

6）发展机会。

这也是非常关键的一个因素。如果有一个很好的工作机会，可以让你直接接触最先进、最核心的业务，或者可以接触到公司的高层，或者可以获得一些非常有用的客户资源，或者可以在短期内迅速进入管理层，这就是非常理想的机遇。当然，如果你希望稳定，进入高校研究所这样的单位也是不错的选择。在考虑发展机会这个因素时，应主要考虑三个方面。

<1>行业背景：要综合考虑公司所处行业的背景和发展现状，更重要的是，要对这个行业的发展前景有准确的预测。

<2>公司背景：要考虑这家公司在行业中所处的地位，目前的发展状况、经营业绩，以及未来的发展预期。

<3>个人机会：要看自己所处的部门在公司的地位，自己的职位的升职机会、发展前景。

7）签约年限及违约金。

一般单位签3年，也有签5年的，还有的单位签1年，如华为。此外，很多单位还有保密合同，不同单位的情况不一样。同时，违约金也会有相关规定。一般来讲，违约金特别高的，要慎重签约。

除此之外，签约时还要考虑很多个人因素，比如，双亲在哪，以后回家照顾老人是否方便；配偶或者男（女）朋友的问题，会不会两地分居。笔者曾经开玩笑和女友说："你在我身边相当于我年薪多了6万。"这并非笑谈，因为感情的融洽不是金钱能够衡量的，所以，不要把钱看得太重，毕竟对于一个人来说，生活的和谐还是要放在首位的。

3.5　违约

拒绝别人虽不像被别人拒绝那样痛苦，但同样是一件痛苦的事情。

大部分人准备违约，无外乎有一个主要原因：遇到了更好的单位。于是，违约也

成了非常普遍的现象。决定违约前一定要计算违约成本，想清楚以下问题：

1）新单位是否比原单位高一个档次？即是否值得为了新单位而违约原单位？如果两家单位差不多，建议最好不要违约。

2）新单位给的最晚签约期限是什么时候？如果跟原单位提出违约，能否在新单位的签约期限前办完？如果没有把握，建议不要违约。

3）原单位以前是否有过成功违约的案例？影响如何？如果以前的违约案例大多不顺利，建议不要违约。

这里面最关键的因素就是：原单位对待你违约的态度。毕竟，这算一个不是很好的行为，对原单位造成损失，对个人声誉和学校声誉也会造成不好的影响。这个态度决定了你能否顺利违约、违约需要的时间，以及能否及时与新单位签约。

如果一定要违约，最好能做到以下几点：

1）与新单位坦诚相告，说明自己的情况，询问能否宽限时间。如果新单位不给你放宽时间，你就没必要违约。当然，你也可以不说，但你必须确保在新单位签约期限前，你能顺利与原单位办完违约手续，否则，你极有可能面临“竹篮打水一场空”的危险。

2）与原单位一定要好好协商，态度诚恳一些。首先要感谢对方的知遇之恩，其次说清楚自己为什么违约，并为自己的行为向对方道歉。同时，要尽可能减少你的违约给学校声誉造成的损失，因为那家单位很有可能因为你的违约而改变对你们学校学生的印象，受害的可能是同校的同学。所以，要想办法来弥补。通常，可以向单位推荐几个自己的同学或朋友，希望能给他们机会。当你放弃机会的同时，别忘记了给周围的人争取机会。

对于应届毕业生来说，违约可能会更麻烦，一个基本的违约流程是：

1）与原单位协商，向原单位接收违约，按照三方协议规定，交纳违约金（有些单位不收违约金），从原单位开出退函。

2）从新单位获取接收函。

3）拿着原单位退函和新单位接收函到就业指导中心领新的三方协议（有时也不需要接收函）。

4）拿新的三方协议与新单位签约。

这个过程中的关键在于第一步：如何与原单位协商，拿到退函。具体的情况视不同单位而不一样，有的单位可能会拖很久，例如，华为通常到 3 月份才给开退函。所以，如果新单位的签约时间很紧，而原单位又不会很快给你开退函，那结果很可能是：你两家单位都签不了。

总之，就业时要经过慎重考虑，不要轻易签约，更不要轻易违约，那样无论对谁都是巨大的伤害。对于你的每一个决定，自己都要为此承担相应的后果和代价！

最后，祝愿每个读者都能顺利签约自己满意的单位！

第4章

职业生涯发展规划

在一般情况下，我们工作一年之后，对自己的喜好及擅长都有了更加深刻的了解，这时会有较为明确的职业发展规划。

4.1 缺乏工作经验的应届毕业生

即将毕业的学生们通常对自己的目标职位很模糊，只要是与计算机相关的工作都想试一下。但是现在公司看重的除了学生的基本素质，即沟通能力、团队协作、学习能力、外语水平等之外，也会关注应届毕业生在校及实习经历中与目标职位相关的经验。假设与导师做的课题或者实习中接触到 J2EE 企业级开发，那么在应聘时寻找一份有相关要求的工作就更容易。而靠这样的经历去找一份 C/C++开发的职位可能就略微难一些。

上海某高校的一位学生在课余时间开发了一个基于校园网内部的搜索引擎。比起商用的搜索引擎，其搜索效率、数据量不算出色，但是该生通过编写自己的搜索引擎，详细了解了网络编程、网页爬虫等领域的知识。这个搜索引擎也表现出了他专业技能的水平，从而为他赢得了前往国际某著名网络公司应聘的机会。

所以，在大学期间，我们可以通过参加创新杯比赛、著名软件公司举行的各种编程大赛、各种技术社团的活动来增加编程经验，以获取公司对你专业技能的肯定。各种编程大赛中获得的名次、实践大赛中的作品，都可以作为工作经验的替代。

对于通过校园招聘招人的大公司，一份有分量的简历只是第一步。有分量指的是成绩尚可，有让他们感兴趣的实习经历，有一定的获奖经历，担任过一定的职务，英语能力还行。这仅仅是第一步，它能让你从众多应聘者中被选出来参加初试，接下来就看你的真正功力和造化了。

初试的要点是基本功扎实，自信乐观，英语交流能力不错，够聪明，够机灵。基

本功扎实并且聪明尤为重要。某位毕业生参加 Sybase 公司面试，过了印度技术官的英语技术面试，第二天参加他们的 Aptitude Test（智商测试），误认为是态度测试（Attitude Test），结果没发挥好。智商测试通常让你在很短的时间内做大量的逻辑题和智力题。不要在前面的题目上浪费太多时间，后面的题目往往更加简单。

另一位求职者通过了微软公司的笔试和电话面试，后来去参加了正式的面试。一连 3 轮，面试官全都是微软高级技术经理，面对这么高级别的面试官，求职者难免紧张。3 轮全英文面试，写了 6 个程序，不算难，但是考得很细，注重求职者的逻辑思维能力、反应能力和编程技巧。写完程序之后马上设计测试用例。或许是没有参加过特别正规的项目开发的缘故，他表现一般，有几个程序有疏漏，面试官加以提醒，虽然最后能够改正，但是加重了他的紧张情绪，没能闯入下一轮。

你也没必要因为自己的学校而显得不够自信，只要打好技术功底，多参加正规的实践项目，找工作的时候自然会顺利。**此外在求职过程中，整体形势和个人形势没有必然联系。**整体形势好的，个人形势未必好。往往整体形势好了，个人容易盲目乐观，在准备不充分的情况下，很容易被莫名其妙地淘汰。即使明年的整体形势比今年还要好，招人的公司比今年还要多，还是建议大家脚踏实地做好充分的知识储备和心理准备，找工作绝对是一场硬仗。

有一种说法：80%的 Offer 掌握在 20%的牛人手中。每年 10 月、11 月应届毕业生刚刚开始找工作的时候，正是牛人们发威的时候，笔试、面试都有他们的份，到了发 Offer 的时候，他们手中集中了很多好 Offer。这时候我们得摆正心态，尽自己最大努力，发挥出自己最好的水平就行了，不用太在意结果。晚些时候往往反而会有好 Offer。写论文的同时抽空复习一下基础的课程：数据结构、Java 编程思想、TCP/IP、操作系统、计算机网络、UML、OOA&OOP、自己做过的项目的知识等。不要怕笔试和面试，笔试得多了，感觉就来了。

可能你找到了工作，并且不止一个。但手头的 Offer 再多，也只能跟一家公司签约，面对的诱惑再多，也只能选择一个。不用羡慕那些手头有很多好 Offer 的人，他们其实很痛苦，这是一种甜蜜的烦恼。罗列出你最在意的方面，把几家公司做详细的比较（见表 4-1）。做选择有时候很感性，理性的数据往往不如公司的一名普通员工给你的印象更能影响你的决定。

表 4-1　各公司信息比较

比 较 内 容	权 重 指 数	A 公司	B 公司
发展机会	20%		
公司前景	10%		

续表

比较内容	权重指数	A公司	B公司
技术方向	10%		
培训机制	5%		
公司性质	5%		
人员规模	3%		
企业文化	5%		
行业排名	5%		
薪酬	20%		
……			

4.2 更换工作的程序员们

如果你是跳槽者中的一员，你要明白**频繁跳槽对职业生涯发展是有害无益的，**招聘方也十分关注求职者的稳定性。一般来说，每份工作都要维持一年以上，能够在某家公司工作满 3 年，才会对公司所在行业及这家公司有比较深入的了解。决定更换工作时，我们要先问问自己要在哪个方向继续自己的职业生涯。假设目前你是某家公司的开发人员，要应聘更大规模公司的同等职位，应该注意下面两点。

首先，比起创业型公司，大公司的开发流程要求会更加规范和严格，有的时候我们必须放弃一些编程的习惯。严格的开发流程对文档的依赖性很大，我们必须做到文档优先。这样的一种环境可能是初入大公司的程序员最难接受的一点。

其次，小公司里那种 Superman 型的程序员在大公司里很少见到。我曾经听一个程序员朋友抱怨他们公司的架构师连 ASP 代码都不会写，其实这是很正常的事情。架构师的工作是将业务需求变成计算机软件的模块和类，他们不需要了解具体代码的编写，只需要分析几种软件平台之间的实现难度和效率差异就够了。当然，大公司也有所谓的技术高手，但这种技术高手并不是精通几种开发语言的“万能钥匙”，而是对某种技术有深入理解，能够解决深层次问题的人。

中国的 IT 界，“技而优则仕”的比较多。很多技术出身的人员做到管理岗位后，关注的仍然是技术细节。但实际上，人员的管理也是一门很大的学问。技术主管的个人风格会影响整个团队的氛围。如果主管不善沟通，只关心 Deadline，那么整个团队将会毫无活力，主管的技术再高超也不会得到信服。如果主管善于沟通，关心下属，那么整个团队就会生机勃勃，即使加班也有劲头。

假设你已不想再做开发，想要转向测试或其他相关岗位，如实施、技术支持，甚至培训、售前等，那你一定要认真向目前在做这份工作的人员了解他们的实际职责与相关要求，确认是否可以接受转换岗位后带来的挑战。如果确定，则可以选择具有相同行业背景的目标职位，并且调整好自己的心理状态，给自己一段较长的时间来适应这种改变。刚开始时感觉无从下手或者有较大落差是很正常的，最起码要在半年之后才能证实你和这个岗位的匹配度。

如果你现在已经有了较为明确的职业生涯发展规划，推荐大家使用倒推法使之切合实际并行之有效。以一个普通程序员为例，可以首先为自己的目标设置一个年限，并列出实现这个目标所需要的专业技能，然后使用倒推法确定阶段目标，直至将这个阶段目标倒推至一个月后，那它就会是一个很具体的目标了。只要你坚持去做，就会逐步实现自己的最终目标。

当然，除此之外，你还要时时关注业界动态，尽可能多地参加在职培训，并且补充外语方面的技能。这样才能保持你继续前进的步伐。

当然，**最重要的是我们要把握好自己，把握好自己要走的路。**其实任何一个职位都需要我们努力工作，任何一份工作都无法“钦定”我们的终身。

4.3 快乐地工作

人的一生很漫长，你无法想象你还能够经历什么；人一生也很短暂，在你觉得还没经历些什么的时候就已经老了。别人的经历其实都是故事，别人的成功也不能复制。

现在在中国，大概很少有人是一份职业做到底的，虽然如此，第一份工作还是有些需要注意的地方，有两件事情格外重要，第一件是入行，第二件事情是跟人。第一份工作对人最大的影响就是入行，现代的职业分工已经很细，我们基本上只能在一个行业里成为专家，不可能在多个行业里成为专家。很多案例也证明即使一个人在一个行业非常成功，到另外一个行业往往完全不是那么回事情，“你想改变世界，还是想卖一辈子汽水？”是乔布斯邀请百事可乐总裁约翰•斯考利加盟苹果时所说的话，结果这位在百事非常成功的约翰，到了苹果表现平平。其实没有哪个行业特别好，也没有哪个行业特别差，或许有报道说哪个行业的平均薪资比较高，但是他们没说的是那个行业的平均压力也比较大。看上去很美的行业，一旦进入才发现很多地方其实并不那么完美，只是外人看不见。

说实话，我自己都没有发财，所以我的建议只是让人快乐工作的建议，不是如何发财的建议，我们只讨论一般普通打工者的情况。我认为选择什么行业并没有太大关

系，看问题不能只看眼前。比如，从 2005 年开始，国家开始整顿医疗行业，很多医药公司开不下去，很多医药行业的销售开始转行。其实医药行业的不景气是针对所有公司的，并非针对一家公司，大家的日子都不好过，这个时候撤资是非常不划算的，大多数正规的医药公司即使不做新生意，撑个两三年总是能撑的，光景总归还会好起来的，那个时候别人都跑了而你没跑，现在的日子应该会好过很多。有的时候觉得自己这个行业不行了，问题是再不行的行业，做的人少了也变成了好行业，当大家都觉得不好的时候，往往却是最好的时候。大家都觉得金融行业好，金融行业门槛高不说，有多少人削尖脑袋要钻进去，竞争激励，进去以后还要时时提防，一个疏忽就被后来的人给挤掉了，压力巨大，又如何谈得上快乐？也就未必是“好”工作了。

太阳能这个东西至今还不能进入实际应用的阶段，但是中国已经有 7 家和太阳能有关的公司在纽交所上市了，国美、苏宁、永乐其实是贸易型企业，也能上市，鲁泰纺织连续 10 年利润增长超过 50%，卖茶的一茶一座，卖衣服的海澜之家都能上市……其实选什么行业真的不重要，关键是怎么做。事情都是人做出来的，关键是人。

有一点是需要记住的，这个世界上，有史以来直到我们能够预见得到的未来，成功的人总是少数，有钱的人总是少数，大多数人是一般的。因此，大多数人的做法和看法往往都不是距离成功最近的做法和看法；大多数人说好的东西未必就好，大多数人说不好的东西不见得不好。大多数人都去炒股的时候，说明跌只是时间问题，大家越是热情高涨的时候，跌的日子越近。少数人买房子的时候，房价不会涨，而房价涨得差不多的时候，大多数人才开始买房子。不会有这样一件事情让大家都变成功，发了财，历史上不曾有过，将来也不会发生。有些东西即使一时运气好而得到了，还是会在某个时候某个地方失去的。

年轻人在职业生涯的刚开始尤其要注意的是，要做自己快乐的事情，不要让自己今后几十年的人生总是提心吊胆的，更不值得为了一份工作赔上自己的青春年华。人还是要看长远一点。很多时候，看起来最近的路，其实是最远的路；看起来最远的路，其实是最近的路。要让自己在职业的道路上走得更远，首先要让自己工作得快乐，如果一份工作让你觉得不快乐，甚至很受罪，那么你就是那个坚持不到终点的选手，即使你坚持到终点了，这样痛苦的人生有意思么？其次要对未来做好规划，尽量让自己劳逸结合，要知道那是一个很漫长的过程，不要在一开始就把力气和耐心耗尽了，当力气耐心耗尽而又遭遇挫折，大多数人就会陷入沮丧悲观，跳槽换工作也就变成很自然的事情。对于初入职场还不能很好控制自己心态的人，掌握好自己的节奏，不要跟着别人的脚步乱了自己的节奏，清楚自己在做什么，清楚自己的目标。

对一个初入职场的人来说，入行后一定要跟一个好领导、好老师。刚进社会的人做事情往往没有经验，需要有人言传身教。对于一个人的发展来说，一个好领导是非

常重要的。所谓“好”的标准，不是他让你少干活多拿钱，而是具有以下三个特点。

首先，好领导要有宽广的心胸，如果一个领导每天都会发脾气，那几乎可以肯定他不是一个心胸宽广的人，能发脾气的时候却不发脾气的领导，多半是非常厉害的领导。有些领导最大的毛病是容忍不了能力比自己强的人，所以常常可以看到的一个现象是，领导很有能力，手下一群庸才或者一群闲人。如果看到这样的环境，还是不要去的好。

其次，领导要愿意从下属的角度来思考问题，这一点其实是从面试的时候就能发现的，如果这位领导总是从自己的角度来考虑问题，几乎不听你说什么，这就危险了。从下属的角度来考虑问题并不代表同意下属的说法，但他必须了解下属的立场，下属为什么要这么想，然后他才有办法说服你，只关心自己怎么想的领导往往难以获得下属的信服。

最后，领导敢于承担责任，如果出了问题就把责任往下推，有了功劳就往自己身上揽，这样的领导不跟也罢。选择领导，要选择关键时刻能扛得住的领导，能够为下属的错误埋单的领导，因为这是他作为领导的责任。

有可能你碰不到好领导，因为他坐领导的位置，所以他的话就比较有道理，这是传统观念官本位的误区，可能有大量的这种无知无能的领导，只是这对于你其实是好事，如果将来有一天你要超过他，你希望他比较聪明还是比较笨？相对来说，这样的领导其实不难搞定，只是你要把自己的身段放下来而已。多认识一些人，多和比自己强的人打交道，同样能找到好的老师，不要和一群同样郁闷的人一起控诉社会，控诉老板，这帮不上你，只会让你更消极，职场上最忌讳的是你还在这家公司却又不停地抱怨公司本身。正确的做法是和那些比你强的人打交道，看他们是怎么想的，怎么做的，学习他们，然后提升自己的能力才是最重要的。

希望所有的读者都能快乐地工作，不断地提升自己。

第 2 部分

Java 程序设计

Java program design

本部分主要以 Java 设计语言为基础，通过大量实际的例子分析各大公司 Java 面试题目，从技术上分析面试题的内涵。一般公司的面试题都是两套：C++或 Java，面试者可以选择。

许多面试题看似简单，却需要深厚的基本功才能给出完美的解答。企业要求面试者写一个最简单的 final 方法都可看出面试者在技术上究竟达到了怎样的程度，我们能真正写好一个 final 方法吗？我们都觉得自己能，可是我们写出的 final 很可能只能拿到 10 分中的 2 分。读者可从本部分中关于 Java 的几个常见考点，如 reflection 问题、I/O 问题、面向对象及接口等方面看看自己属于什么样的层次。此外，还有一些面试题考查面试者敏捷的思维能力。

分析这些面试题，本身包含很强的趣味性。而作为一名研发人员，通过对这些面试题的深入剖析则可进一步增强自身的内功。

第5章

Java 程序设计基本概念

对于一个求职者或者应届毕业生来说，公司除了对项目经验有所问询之外，最好的考量办法就是检查基本功，包括编程风格，以及对赋值语句、递增语句、类型转换、数据交换等程序设计基本概念的理解。当然，在考试之前最好对自己所掌握的程序概念知识有所了解，尤其是对各种细致的考点要加以重视。以下考题来自真实的笔试资料，希望读者先不要看答案，自我解答后再与答案加以比对，找出自己的不足。

5.1 JVM

面试例题 1：下面给出的 Java 中 ClassLoader 中的描述，哪些描述是正确的？（　　）

A．ClassLoader 没有层次关系

B．所有类中的 ClassLoader 都是 AppClassLoader

C．通过 Class.forName(String className)，能够动态加载一个类

D．不同的 ClassLoader 加载同一个 Class 文件，所得的类是相同的

解析：

A 选项错误，ClassLoader 具备层次关系。

B 选项错误，ClassLoader 不止一种。

D 选项错误，不同的类装载器分别创建的同一个类的字节码数据属于完全不同的对象，没有任何关联。

答案：C

扩展知识：ClassLoader 知识。

（1）ClassLoader 基本概念

与 C 或 C++编写的程序不同，Java 程序并不是一个可执行文件，而是由许多独立的类文件组成的，每一个文件对应一个 Java 类。此外，这些类文件并非全部都装入内存，而是根据程序需要逐渐载入。ClassLoader 是 JVM 实现的一部分，ClassLoader 包括 bootstrap classloader（启动类加载器），ClassLoader 在 JVM 运行的时候加载 Java 核心的 API，以满足 Java 程序最基本的需求，其中就包括用户定义的 ClassLoader，这里所谓的用户定义，是指通过 Java 程序实现的两个 ClassLoader：一个是 ExtClassLoader，它的作用是用来加载 Java 的扩展 API，也就是/lib/ext 中的类；第二个是 AppClassLoader，它是用来加载用户机器上 CLASSPATH 设置目录中的 Class 的，通常在没有指定 ClassLoader 的情况下，程序员自定义的类就由该 ClassLoader 进行加载。

（2）ClassLoader 加载流程

当运行一个程序的时候，JVM 启动，运行 bootstrap classloader，该 ClassLoader 加载 Java 核心 API（ExtClassLoader 和 AppClassLoader 也在此时被加载），然后调用 ExtClassLoader 加载扩展 API，最后 AppClassLoader 加载 CLASSPATH 目录下定义的 Class，这就是一个程序最基本的加载流程。

下面来看一下 ClassLoader 中的一段代码：

```
protected synchronized Class loadClass(String name, boolean resolve)
throws ClassNotFoundException
{
// 首先检查该 name 指定的 class 是否有被加载
Class c = findLoadedClass(name);
if (c == null) {
try {
if (parent != null) {
//如果 parent 不为 null，则调用 parent 的 loadClass 进行加载
c = parent.loadClass(name, false);
} else {
//parent 为 null，则调用 BootstrapClassLoader 进行加载
c = findBootstrapClass0(name);
}
} catch (ClassNotFoundException e) {
//如果仍然无法加载成功，则调用自身的 findClass 进行加载
c = findClass(name);
}
}
if (resolve) {
```

```
resolveClass(c);
}
return c;
}
```

从上面一段代码中可以看出，一个类加载的过程使用了一种父类委托模式。为什么要使用这种父类委托模式呢？

第 1 个原因就是这样可以避免重复加载，当父类已经加载了该类的时候，就没有必要子 ClassLoader 再加载一次。

第 2 个原因就是考虑到安全因素，如果不使用这种委托模式，那么可以随时使用自定义的 String 来动态替代 Java 核心 API 中定义的类型，这样会存在非常大的安全隐患，而父类委托的方式就可以避免这种情况，因为 String 已经在启动时被加载，所以，用户自定义类是无法加载一个自定义的 ClassLoader。

（3）一些重要的方法

1）loadClass 方法。

ClassLoader.loadClass() 是 ClassLoader 的入口点。该方法的定义如下：

```
Class loadClass( String name, boolean resolve );
```

name 是指 JVM 需要的类的名称，如 Foo 或 java.lang.Object。resolve 参数告诉方法是否需要解析类。在准备执行类之前，应考虑类解析。注意：并不总是需要解析，如果 JVM 只需要知道该类是否存在或找出该类的超类，那么就不需要解析。

2）defineClass 方法。

defineClass 方法接受由原始字节组成的数组，并把它转换成 Class 对象。原始数组包含如从文件系统或网络装入的数据。defineClass 管理 JVM 的许多复杂的实现层面——它把字节码分析成运行时数据结构、校验有效性等。因为 defineClass 方法被标记成 final 的，所以也不能覆盖它。

3）findSystemClass 方法。

findSystemClass 方法从本地文件系统装入文件。它在本地文件系统中寻找类文件，如果存在，就使用 defineClass 将原始字节转换成 Class 对象，以将该文件转换成类。当运行 Java 应用程序时，这是 JVM 正常装入类的默认机制。对于定制的 ClassLoader，只有在尝试其他方法装入类之后，再使用 findSystemClass。这是因为 ClassLoader 是负责执行装入类的相关步骤，不负责所有类的所有信息。例如，即使 ClassLoader 从远程的 Web 站点装入了某些类，仍然需要在本地机器上装入大量的基本 Java 库。而这些类库不是我们所关心的，所以要 JVM 以默认方式从本地文件系统装入它们，这就是 findSystemClass 的用途。

4）resolveClass 方法。

正如前面所提到的，可以不完全地（不带解析）装入类，也可以完全地（带解析）装入类。当编写我们自己的 loadClass 时，可以调用 resolveClass，这取决于 loadClass 的 resolve 参数的值。

5）findLoadedClass 方法。

findLoadedClass 充当一个缓存：当请求 loadClass 装入类时，它调用该方法来查看 ClassLoader 是否已装入这个类，这样可以避免重新装入已存在类所造成的麻烦。

6）findClass 方法。

loadClass 默认实现调用这个新方法。findClass 的用途包含 ClassLoader 的所有特殊代码，而无须要复制其他代码（例如，当专门的方法失败时，调用系统 ClassLoader）。

目的是从本地文件系统使用实现的类装载器装载一个类。为了创建自己的类装载器，应该扩展 ClassLoader 类，这是一个抽象类。可以创建一个 FileClassLoaderextends ClassLoader，然后覆盖 ClassLoader 中的 findClass（String name）方法，这个方法通过类的名字而得到一个 Class 对象。

```
public Class findClass(String name)
    {
        byte [] data = loadClassData(name);
        return defineClass(name, data, 0 , data.length);
    }
```

7）getSystemClassLoader 方法。

如果覆盖 findClass 或 loadClass，getSystemClassLoader 能以实际的 ClassLoader 对象来访问系统 ClassLoader（而不是固定地从 findSystemClass 调用它）。为了将类请求委托给父类 ClassLoader，这个新方法允许 ClassLoader 获取它的父类 ClassLoader。当使用特殊方法，定制的 ClassLoader 不能找到类时，可以使用这种方法。

父类 ClassLoader 被定义成创建该 ClassLoader 所包含代码的对象的 ClassLoader。

8）forName 方法。

Class 类中有一个静态方法 forName，这个方法和 ClassLoader 中的 loadClass 方法的目的一样，都是用来加载 class 的，但是两者在作用上却有所区别。

```
Class clazz = Class.forName("something");
```

或者

```
ClassLoadercl = Thread.currentThread().getContextClassLoader();
Class clazz = cl.loadClass("something");
```

Class.forName()调用 Class.forName(name, initialize, loader); 也就是 Class.forName("something"); 等同于 Class.forName ("something", true, CALLCLASS.class.getClassLoader());

第二个参数“true”是用于设置加载类的时候是否连接该类，true 就连接，否则就不连接。关于连接，在此解释一下，在 JVM 加载类的时候，需要经过三个步骤：装载、连接、初始化。装载就是找到相应的 class 文件，读入 JVM；初始化就是 class 文件初始化。这里详述一下连接，连接分三步：

第一步是验证 class 是否符合规格。

第二步是准备，就是为类变量分配内存的同时设置默认初始值。

第三步就是解释，而这步就是可选的，根据上面 loadClass 方法的第二个参数来判定是否需要解释，这里的解释是指根据类中的符号引用查找相应的实体，再把符号引用替换成一个直接引用的过程。

在 Java API 文档中，loadClass 方法的定义是 protected，也就是说，该方法是被保护的，而用户使用的方法是一个参数，一个参数的 loadclass 方法实际上就是调用了两个参数，第二个参数默认为 false。因此，在这里可以看出通过 loadClass 加载类实际上就是加载的时候并不对该类进行解释，因此不会初始化该类。而 Class 类的 forName 方法则相反，使用 forName 加载的时候就会将 Class 进行解释和初始化。

面试例题 2：Which characters does JVM use（JVM 使用哪种字符表示）？（　　）

A．ASCII characters　　B．Unicode characters

C．Cp1252　　D．UTF-8

解析：JVM 的设计者当初决定 JVM 中所有字符的表示形式时，是不允许使用各种编码方式的字符并存的。这是因为如果在内存中的 Java 字符可以以 GB2312、UTF-16、BIG5 等各种编码形式存在，那么对开发者来说，连进行最基本的字符串打印、连接等操作都会寸步难行。例如，一个 GB2312 的字符串后面连接一个 UTF-8 的字符串，那么连接后的最终结果应该是什么编码的呢？选哪一个都没有道理。

Java 开发者必须牢记：在 Java 中字符只以一种形式存在，那就是 Unicode（不选择任何特定的编码，直接使用它们在字符集中的编号，这是统一的唯一方法）。

但“在 Java 中”到底是指在哪里呢？是指在 JVM 中、在内存中、在你的代码里声明的每一个 char、String 类型的变量中。例如，你在程序中这样写：

```
char han='永';
```

在内存的相应区域，这个字符就表示为 0x6c38，可以用下面的代码证明：

```
char han='永';
System.out.format("%x",(short)han);
```

输出是：6c38。反过来用 Unicode 编号来指定一个字符也可以，像这样：

```
char han=0x6c38;
System.out.println(han);
```

输出是：永

这其实也是说，只要你正确地读入了“永”字，那么它在内存中的表示形式一定是 0x6c38，没有任何其他的值能代表这个字。

JVM 的这种约定使得一个字符分为两部分：JVM 内部和 OS 的文件系统。在 JVM 内部，统一使用 Unicode 表示，当这个字符被从 JVM 内部移到外部（即保存为文件系统中的一个文件的内容时），就进行了编码转换，使用了具体的编码方案。因此可以说，所有的编码转换只发生在边界的地方，JVM 和 OS 的交界处，也就是各种输入/输出流（或者 Reader，Writer 类）起作用的地方。

所有的 I/O 基本上可以分为两大阵营：面向字符的输入/输出流；面向字节的输入/输出流。面向字符或者说面向字节中的所谓“面向”，是指这些类在处理输入/输出的时候，在哪个意义上保持一致。

如果面向字节，那么这类工作要保证系统中的文件二进制内容和读入 JVM 内部的二进制内容一致，不能变换任何 0 和 1 的顺序。这种输入/输出方式很适合读入视频文件或者音频文件，或者任何不需要做变换的文件内容。

而面向字符的 I/O 是指希望系统中的文件的字符和读入内存的“字符”（注意和字节的区别）要一致。例如，我们的中文版 WindowsXP 系统上有一个 GBK 的文本文件，其中有一个“永”字，这个字的 GBK 编码什么不用管，当我们使用面向字符的 I/O 把它读入内存并保存在一个 char 型变量中时，我希望 I/O 系统不要直接把“永”字的 GBK 编码放到这个字符（char）型变量中，我不关心这个 char 型变量具体的二进制内容到底是多少，我只希望这个字符读进来之后仍然是“永”字。

从这个意义上也可以看出，面向字符的 I/O 类，也就是 Reader 和 Writer 类，实际上隐式地做了编码转换，在输出时，将内存中的 Unicode 字符使用系统默认的编码方式进行了编码，而在输入时，将文件系统中已经编码过的字符使用默认编码方案进行了还原。这里要注意：Reader 和 Writer 只会使用这个默认的编码来做转换，而不能为一个 Reader 或者 Writer 指定转换时使用的编码。这也意味着，如果使用中文版 Windows

XP 系统，其中存放了一个 UTF-8 编码的文件，当采用 Reader 类来读入的时候，它还会使用 GBK 来做转换，转换后的内容当然不对。这其实是一种傻瓜式的功能提供方式，对大多数初级用户（以及不需要跨平台的高级用户）来说反而是一件好事。

如果用到 GBK 编码以外的文件，就必须采用编码转换：一个字符与字节之间的转换。因此，Java 的 I/O 系统中能够指定转换编码的地方，也就在字符与字节转换的地方，那就是 InputStreamReader 和 OutputStreamWriter。这两个类是字节流和字符流之间的适配器类，它们承担编码转换的任务。

答案：B

5.2 i++

面试例题 1：下列程序的输出结果是多少？

```
public class Test {
 static {
      int x = 5;
 }
 static int x, y;
 public static void main(String[] args) {
      x--;
      myMethod();
      System.out.println(x + y++ + x);
 }
 public static void myMethod() {
      y = x++ + ++x;
 }
}
```

解析：不论是 C++还是 Java，对 i++类的面试问题总是很多。这里要注意运算符的优先级问题。

对上面代码的解释如下：

```
public class Test
{
   static{int x =5;}//在第一次被载入 JVM 时运行，但由于是局部变量，x=5 不影响后面的值
   static int x,y;//初始化时 x=0;y=0;
   public static void main(String[] args)
   {
   x--;
   System.out.println(x);//步骤 1：在运行 myMethod();之前，x 是-1，开始调用 myMethod()函数
   myMethod();//步骤 4：在运行 myMethod();之后 x 是 1，y 是 0
   System.out.println(x+ y++ +x); //步骤 5：运行 x+(y++)+x=1+0+1=2。
```

```
    }

    public static void myMethod()
    {
    y = x++ + ++x;
    System.out.println(y);          //步骤 2：进入 myMethod()运行 y = (x++) + (++x)后 y=0
    System.out.println(x);          //步骤 3：此时 x=1
    }
}
```

答案：2

面试例题 2：Given the following class:（给定下面的类：）

```
public class ZeroPrint{
public static void main(String argv[]){
      int i =0;
      //Here
      }
}
```

Which of the following lines if placed after the comment //Here will print out is not 0?（哪个选项替换掉类中的//Here 不会输出结果 0？）[Sun 公司 2006 年 10 月面试题]

A．System.out.println(i++);　　C．System.out.println(i);

B．System.out.println(i+'0');　　D．System.out.println(i--);

解析：这道题主要考查后置操作符，选项 A 是自加运算，选项 D 是自减运算（但选项 A 和选项 D 都是先打印 0，再自加或自减）。选项 B 中'0'是一个 char 类型，而不是一个数字类型。System.out.println(i+'0');的结果会得到 48。

答案：B

面试例题 3：下列程序的输出结果是（　　）。

```
import java.util.*;

public  class  Test  {
 public  static  void  main(String[]  args)  {
      int  j = 0;
      for  (int  i = 0; i < 100; i++)
      {
          j = j++;
      }
   System.out.println(j);
   }
}
```

A．0　　B．99　　C．100　　D．101

解析：因为 Java 用了中间缓存变量的机制，所以，j=j++；可换成如下写法：

```
temp=j;
j=j+1;
j=temp;
```

所以结果为 0。

答案：A

面试例题 4：If there are " int a=5,　b=3;",　the values of a and b are __ and __ after execute　" if(!(a==b)&&(a==1+b++)){};". （假如“int　a=5，b=3；”，则执行" if(!(a==b)&&(a==1+b++)){}; "后 a 和 b 的值分别为__和__。)[金山公司 2005 年面试题]

A．5,3　　B．0,1　　C．0,3　　D．5,4

解析：表达式运算面试例题。因为“(!(a==b))”运算结束后，仍然继续执行(a==1+b++))，所以答案是 5，4。

答案：D

面试例题 5：以下代码的执行结果是多少？[金山公司 2005 年面试题]

```
import java.util.*;
public class  Test3
{


 public static void main(String[] args) {
       int i=0;
       i=i++ + ++i;
       int j=0;
       j=++j + j++ + j++ + j++;
       int k=0;
       k=k++ + k++ + k++ + ++k;
        int h=0;
       h=++h + ++h;
       int p1=0,p2=0; int q1=0,q2=0;
       q1=++p1;
       q2=p2++;

       System.out.println("i "+i);
       System.out.println("j "+j);
       System.out.println("k "+k);
       System.out.println("h "+h);
       System.out.println("p1 "+p1);
```

```
        System.out.println("p2 "+p2);
        System.out.println("q1 "+q1);
        System.out.println("q2 "+q2);

    }
}
```

解析：i++和++i 使用的不同点在于一个是程序完毕后自增，一个是程序开始前自增。

“i = i++ + ++i;”执行的过程是先执行 i++，但是 i 自增 1 操作是稍后才执行，所以此时的 i 还是 0，然后执行++i，++i 后 i 的值是 1，执行完++i 后要补增 i++，所以此时 i 的值实际上是 2，0+2=2，然后赋值给 i，最终 i 的值是 2。

“j = ++j + j++ + j++ + j++;”执行的过程是先++j，所以 j 的值是 1，然后执行 j++，j++后 j 的值仍然是 1，然后再执行 j++，执行后的结果仍然是 1，但要补增刚才的 j++，所以此时 j 的值实际上是 2，然后执行最后一个 j++，执行后的结果仍然是 2，但要补增刚才的 j++，所以此时 j 的值实际上是 3，所以 1+1+2+3=7，然后赋值给 j，最终 j 的值是 7。

“k = k++ + k++ + k++ + ++k;”执行的过程是先 k++，所以 k 的值是 0，然后执行 k++，k++后 k 的值仍然是 0，但要补增刚才的 k++，所以此时 k 的值实际上是 1，然后再执行最后一个 k++，执行后的结果仍然是 1，但要补增刚才的 k++，所以此时 k 的值实际上是 2，最后执行++k，执行结果为 3，再补增刚才的 k++，k 的实际结果是 4。所以 0+1+2+4=7，然后赋值给 k，最终 k 的值是 7。

“h = ++h + ++h;”是先自增 h，h 值为 1，再自增 h，h 值为 2。所以 1+2=3，然后赋值给 h，最终 h 的值是 3。

“q1=++p1;”先自增 p1，p1 的值是 1，再赋值给 q1，所以 q1 的值是 1。

“q2=p2++;”先赋值 p2 给 q2，q2 的值是 0，然后自增 p2，所以 p2 的值是 1。

答案：

i=2，j=7，k=7，h=3，p1=1，p2=1，q1=1，q2=0。

5.3　类型转换

面试例题 1：Which of the following will compile correctly? [中国杭州著名 B2B 软件公司 2009 年 10 月笔试题]

A．Short myshort = 99S;　　C．float z = 1.0;

B．int t = "abc".length();　　D．char c =17c;

解析：Short myshort = 99S;这句要执行自动装箱，调用 shortValue 方法，显然 99S 无法得到值。

将 float z =1.0;改为 float z =1.0f; 就行了，系统默认的浮点数是 double 型。

在 Java 中，length 是属性，一般用来说明数组的长度；length()是方法，用来求数组中某个元素的字符串长度。例如：

```
String[] s = {"adfasf","sdfgs"};
 s.length;
s[1].length();
```

length()是字符串的方法，返回字符串的长度。

s[1].length()就是"sdfgs".length()值为 5；

length 是数组的属性，s. length 的值为 2。

答案：D

扩展知识：Java 的数据类型转换。

Java 的数据类型分为三大类，即布尔型、字符型和数值型，其中，数值型又分为整型和浮点型。相对于数据类型，Java 的变量类型为布尔型 boolean；字符型 char；整型 byte、short、int、long；浮点型 float、double。其中四种整型变量和两种浮点型变量分别对应于不同的精度和范围。此外，编程时还经常用到两种类变量，即 String 和 Date。

（1）数据类型转换的种类

Java 数据类型的转换一般分三种，分别是：简单数据类型之间的转换、字符串与其他数据类型的转换、其他实用数据类型的转换。

（2）简单数据类型之间的转换

在 Java 中，整型、实型、字符型被视为简单数据类型，这些类型由低级到高级分别为(byte，short，char)—int—long—float—double。

简单数据类型之间的转换又可以分为：低级到高级的自动类型转换、高级到低级的强制类型转换、包装类过渡类型能够转换。

1）自动类型转换。

低级变量可以直接转换为高级变量，这叫自动类型转换。例如，下面的语句可以在 Java 中直接通过：

```
byte b;int i=b;long l=b;float f=b;double d=b;
```

如果低级类型为 char 型，向高级类型(整型)转换时，会转换为对应的 ASCII 码值，例如

```
char c='c';  int i=c;  System.out.println("output:"+i);
```

输出：output:99;

对于 byte、short、char 三种类型而言，它们是相同级别的，因此，不能相互自动转换，可以使用下述的强制类型转换。

```
short i=99;char c=(char)i;System.out.println("output:"+c);
```

输出：output:c;

2）强制类型转换。

将高级变量转换为低级变量时，情况会复杂一些，你可以使用强制类型转换。如:

```
int i=99;byte b=(byte)i;char c=(char)i;
```

这种转换可能会导致溢出或精度的下降。

3）包装类过渡类型转换。

Java 的包装类就是可以直接将简单类型的变量表示为一个类。Java 共有六个包装类，分别是 Boolean、Character、Integer、Long、Float 和 Double，从字面上可以看出它们分别对应于 boolean、char、int、long、float 和 double。而 String 和 Date 本身就是类，不存在包装类的概念。

在进行简单数据类型之间的转换（自动转换或强制转换）时，可以利用包装类进行中间过渡。一般情况下，首先声明一个变量，然后生成一个对应的包装类，就可以利用包装类的各种方法进行类型转换了。例如，

例 1：当希望把 float 型转换为 double 型时:

```
float f1=100.00f;
Float F1=new float(f1);
Double d1=F1.doubleValue();
```

例 2：当希望把 double 型转换为 int 型时:

```
double d1=100.00;   Double D1=new Double(d1);   int i1=D1.intValue();
```

例 3：当希望把 int 型转换为 double 型时，自动转换:

```
int i1=200;   double d1=i1;;
```

例 4：简单类型的变量转换为相应的包装类，可以利用包装类的构造函数。即

```
Boolean(boolean value)
Character(char value)
Integer(int value)
Long(long value)
```

```
Float(float value)
Double(double value)
```

利用这种方法也可以实现不同数值型变量间的转换，例如，对于一个双精度实型类，int Value()可以得到其对应的整型变量，而 double Value()可以得到其对应的双精度实型变量。

（3）字符串型与其他数据类型的转换

通过查阅类库中各个类提供的成员方法可以看到，几乎从 java.lang.Object 类派生的所有类都提供了 toString()方法，即将该类转换为字符串。例如：Characrer、Integer、Float、Double、Boolean、Short 等类的 toString()方法用于将字符、整数、浮点数、双精度数、逻辑数、短整型等类转换为字符串，如下所示：

```
class Test {
 public static void main(String args[]) {
       int i1 = 10;
       float f1 = 3.14f;
       double d1 = 3.1415926;
       Integer I1 = new Integer(i1);
       Float F1 = new Float(f1);
       Double D1 = new Double(d1);
       String si1=I1.toString();
       String sf1=F1.toString();
       String sd1=D1.toString();
       System.out.println("si1" + si1);
       System.out.println("sf1" + sf1);
       System.out.println("sd1" + sd1);
 }
}
```

（4）将字符型直接作为数值转换为其他数据类型

将字符型变量转换为数值型变量实际上有两种对应关系：一种是将其转换成对应的 ASCII 码；另一种是转换关系，例如，'1'就是指数值 1，而不是其 ASCII 码，对于这种转换，可以使用 Character 的 getNumericValue(char ch)方法。

面试例题 2：下面代码的输出结果是（　　）。[中国杭州著名 B2B 软件公司 T2009 年 10 月笔试题]

```
int i = 012;
int j = 034;
int k = (int)056L;
int l = 078;
System.out.println(i);
```

```
System.out.println(j);
System.out.println(k);
```

A．输出 12,34,56　　C．int k = (int)056L;行编译错误

B．输出 10,28,46　　D．int l = 078;行编译错误

解析：int l = 078;行编译错误，因为 078 是八进制，只能选 0～7 的数字，不应该有 8。

答案：D

面试例题 3：以下程序错误的是_____。

A．short s=1;s=s+1;　　B．short s=1;s+=1;

解析：s+1 为 int，不能直接赋值给 short。

答案：A

5.4　程序结构

面试例题 1：什么时候用 assert？（API 级的技术人员有可能会问这个问题。）[中国台湾著名 CPU 生产厂商 V2007 年 11 月笔试题]

解析：assert 是断言。断言是一个包含布尔表达式的语句，在执行这个语句时，假定该表达式为 true，如果表达式计算为 false，那么系统会报告一个 Assertionerror。它用于调试目的。

```
assert(a > 0); // throws an Assertionerror if a <= 0
```

断言可以有以下两种形式：

```
assert Expression1;
assert Expression1 : Expression2;
```

Expression1 应该总是产生一个布尔值。Expression2 可以是一个值的任意表达式，这个值用于生成显示更多调试信息的 String 消息。

断言在默认情况下是禁用的。要在编译时启用断言，需要使用 source 1.4 标记：javac-source 1.4Test.java

要在运行时启用断言，可使用-enableassertions 或者-ea 标记。

要在运行时选择禁用断言，可使用-da 或者-disableassertions 标记。

要在系统类中启用断言，可使用-esa 或者-dsa 标记，还可以在包的基础上启用或者禁用断言。

答案：可以在预计正常情况下不会到达的任何位置上放置断言。断言可以用于验证传递给私有方法的参数。不过，断言不应该用于验证传递给公有方法的参数，因为不管是否启用了断言，公有方法都必须检查其参数。不过，既可以在公有方法中，也可以在非公有方法中利用断言测试后置条件。另外，断言不应该以任何方式改变程序的状态。

面试例题 2：Which declaration for the main() method in a stand-alone program are NOT valid?（哪一个 main 函数的声明是不合法的？）[法国著名通讯公司 2005 年和 2009 年面试题]

A．public static void main()

B．public static void main(String[] string)

C．public static voidmain(String[] exp) throws FileNot FoundException

D．static void main(String[] args)

解析：A、B 选项显然是合法的。C 选项抛出一个文件异常，但也是合法的，并可以通过，代码如下。

```
import java.io.FileNotFoundException;
public class zero
{
   public static void main(String[] exp)throws FileNotFoundException
   {


   }
}
```

至于选项 D，因为 main 方法必须是 public 的，默认的代表是 protect，所以是不合法的。

答案：D

5.5 运算符

面试例题 1：以下代码的输出结果是（　　）。[中国台湾著名杀毒软件公司 Q2009 年 11 月笔试题]

```
public class Test {
 public static void main(String[] args) {
 int i = 42;
     String s = (i<40) ? "life":(i>50)?"universe":"everything";
     System.out.println(s);
```

```
    }
}
```

A．life　　B．universe　　C．everything　　D．以上答案都不对

解析：语言中运算符分为 3 类：单目运算符、二目运算符、三目运算符。

单目运算就是只对一个参数进行运算（如++、--等）；双目运算就是对两个参数进行运算（如+、-、>、<等）；三目运算就是对三个参数进行运算（如？、:）。

```
s=a?b:c;相当于(if(a) s =b;else s =c;)。
```

本题中的三目运算符有点特殊，是一个嵌套三目运算符。

```
String s = (i<40) ? "life":(i>50)?"universe":"everything";
```

相当于

```
String s2;
if(i<40)
{
     s2 = "life";
}
else if(i>50)
{
     s2 = "universe";
}else
{
     s2 = "everything";
}
```

所以答案是 everything。

答案：C

面试例题 2：下列程序的输出结果是（　　）。[中国东北著名软件公司 D2009 年 3 月笔试题]

```
import java.util.*;
public  class  Test  {
  public  static  void  main(String[]  args)  {
  boolean  b  =  true?false:true  ==  true?false:true;
  System.out.println(b);
  }
}
```

A．true　　B．false　　C．null　　D．以上答案都不对

解析：三目运算符是右结合性的，所以应该理解为：

```
boolean  b =true?false:((true == true)?false:true);
```

本题考查的是对运算符的优先级别，基本的顺序是（1 级优先级最高，16 级最小）：

```
1 级  —  . ()
2 级  —  ++  --
3 级  —  new
4 级  —  *  /  %
5 级  —  +  -
6 级  —  >>  <<  >> >
7 级  —  >  <  > =  <=
8 级  —  ==  !=
9 级  —  &
10 级  —  ^
11 级  —  !
12 级  —  &&
13 级  —  ||
14 级  —  ?:
15 级  —  =  +=  -=  *=  /=  %=  ^=
16 级  —  &=  <<=  >>=
```

答案：B

面试例题 3：以下代码输出结果是（　　）。[中国东北著名软件公司 D2009 年 3 月笔试题]

```
public class Test {
     public static void main(String[] args) {
    int a = 5;
    System.out.println("value is  " + ((a < 5) ? 10.9 : 9));
      }
}
```

A．编译错误　　B．10.9　　C．9　　D．以上答案都不对

解析：如果你不假思索地直接选 C，就恰恰中了题目设置的陷阱。注意到（(a < 5) ? 10.9 : 9）里面有一个 10.9，而后面直接跟了一个 9 。这时 Java 就会根据运算符的精度类型进行自动类型转换。由于前面有一个 10.9，因此，后面的 9 也会自动变成 9.0。因此，答案选 D。

答案：D

面试例题 4：以下代码的输出结果是（　　）。[中国著名网络软件公司 X2009 年 10 月笔试题]

```
import java.util.*;
public class Test {
```

```
    public static void main(String[] args) {
    char x = 'x';
    int i = 10;
    System.out.println(false ? i : x);
    System.out.println(false ? 10 : x);
    }
}
```

A．120 x　　B．120 120　　C．x 120　　D．以上答案都不对

解析：int i = 10;中的 i 是一个变量，因此，第一个输出 x 被提升为 int 型了，因为 i 是 int 类型，x 的 int 值为 120，所以第一个输出为 120。

至于第 2 个输出，Java 编程规范中提到：当后两个表达式有一个是常量表达式（本题中是 10）时，另外一个是类型是 T（本题中是 char）时，而常量表达式可以被 T 表示时（representable in type T），输出结果是 T 类型。所以，因为 10 是常量，可以被 char 表示。输出结果是 char 型的。

答案：A

面试例题 5：What does the following program print ?（下面程序的运行结果是多少？）[中国著名 ERP 软件公司 Y2010 年 1 月笔试题]

```
import java.util.*;
public class Test {
    public static void main(String[] args) {
        int m = 5, n = 5;
        if((m != 5) && (n++ == 5)){}
        System.out.println("a." + n);

        m = n = 5;
        if((m != 5) & (n++ == 6)){}
        System.out.println("b." + n);

        m = n = 5;
        if((m == 5) || (n++ == 5)){}
        System.out.println("c." + n);

        m = n = 5;
        if((m == 5) | (n++ == 6)){}
        System.out.println("d." + n);

        int a = 1, b = 2;
        int c = a & b;
        System.out.println("a&b" + c);
```

```
    }
}
```

解析：

“&”、“|”、“^”这三个是什么运算符？相信多数面试者基本上都会回答“位运算符”，但这样的回答并不完整。其实它们还可以充当布尔逻辑运算符（前提是两边的数据类型为布尔类型）。

在位运算中，int c = a & b; 的意思是首先使 a 和 b 按位与，a 是 1，b 是 2，a 的二进制数位是 0001，c 的二进制数位是 0010。“与”的结果如下表:

a	0	0	0	1
b	0	0	1	0
a&b	0	0	0	0

在布尔逻辑运算符中，这三个运算符充当着“布尔逻辑与”、“布尔逻辑或”和“布尔逻辑异或”的角色。布尔逻辑与（&）和布尔逻辑或（|）运算符的工作方式同逻辑与（&&）和逻辑或（||）的工作方式相同，布尔逻辑运算符的优先级别要高于逻辑运算符。

&、|逻辑运算与&&、||逻辑运算的重要区别是：前者是非短路运算，后者是短路运算。

编译器对于&&和||已经优化过，凡&&前面的是 false，那么&&后面的表达式就不用再做了。||前面的是 true，||后面的也就不做了，这就是所谓的“短路”，而布尔逻辑运算符就没有这个特点，无论运算符&、|前面的是 true 或 false，运算符后面的表达式都得继续进行运算。这就是所谓的“非短路”。

下面分析题目中的 4 段代码：

```
代码 1：
int m = 5, n = 5;
if((m != 5) && (n++ == 5)){}
System.out.println("a." + n);
```

上面这段代码中，(m != 5) 为 false，由于中间的逻辑符是&& (短路)，(n++ == 5) 就不用做了，所以 n 还是 5。

```
代码 2：
m = n = 5;
if((m != 5) & (n++ == 6)){}
System.out.println("b." + n);
```

上面这段代码中，(m != 5) 为 false，由于中间的逻辑符是& (非短路)，(n++ == 5) 仍然需要计算，所以 n 是 6。

代码 3：

```
m = n = 5;
if((m == 5) || (n++ == 5)){}
System.out.println("c." + n);
```

上面这段代码中，(m == 5) 为 true，由于中间的逻辑符是||（短路），(n++ == 5) 就不用做了，所以 n 还是 5。

代码 4：

```
m = n = 5;
if((m == 5) | (n++ == 6)){}
System.out.println("d." + n);
```

上面这段代码中，(m == 5) 为 true，由于中间的逻辑符是|（非短路），(n++ == 5) 仍然需要计算，所以 n 是 6。

答案：5，6，5，6，0

面试例题 6：下列程序的输出结果是（　　）。[中国著名 ERP 软件公司 Y2010 年 1 月笔试题]

```
import java.util.*;
public  class  Test  {
public  static  void  main(String[]  args)  {
     int  num  =  32;
     System.out.println(num  >>  32);
     }
}
```

A．32　　B．16　　C．1　　D．0

解析：移位操作符右边的参数要先进行模的 32 运算，并且移位是对二进制的操作，而二进制中 8 位是一个循环。所以，num>>32 等于 num>>0，而 num>>33 等于 num>>1。

答案：A

5.6　异常

面试例题 1：Which of the following is NOT true regarding to RuntimeException?（关于运行时异常说法不正确的是）[中国台湾著名 CPU 生产厂商 V2009 年 9 月笔试题]

A．RuntimeException is the superclass of those exceptions that must be thrown during the normal operation of the Java Virtul Machine.（运行时异常是一个超类，当 Java 虚拟机正常时一定抛出）

B．A method is not required to declare in its throws clause any subclasses of RuntimeException that might be thrown during the execution of the method but not caught.（运行时异常可以不去捕捉）

C．An RuntimeException is a subclass of Throwable that indicates serious problems that a reasonable application should not try to catch.（运行时异常是一个子类，当出现严重运行问题时也不会抛出）

D．NullPointerException is one kind of RuntimeException.（空异常是一种运行时异常）

解析：一个合理的应用程序不应该（没必要）捕捉运行时异常。

答案：C

面试例题 2：定义了如下类和测试方法，请问测试时期待要捕获下面哪个选项的异常？[英国某通讯软件公司 S2009 年 11 月笔试题]

```
class MyException extends Exception{
 MyException(){}
 }

 class A{
 public int format(String str)throws MyException{
 int i = Integer.valueOf(str);
 return i;
 }
 }

 public void testTester(){
 new A().format("1");
}
```

A．Exception　　B．MyException

C．MyException 和 NumberFormatException　　D．RuntimeException

解析：本题问的是期待捕捉哪个异常，因为函数 format 显式地声明抛出 MyException 异常，故期待捕捉 MyException，但实际上捕捉的异常是 runtime 异常 NumberFormatException。

答案：B

扩展知识：关于 Java 异常

1．什么是异常

在 Java 程序运行时，常常会出现一些非正常的现象，这种情况称为运行错

误。根据其性质可以分为错误和异常。Java 程序中（无论是谁写的代码），所有抛出（throw）的异常都必须从 Throwable 派生而来。类 Throwable 有两个直接子类：Error 和 Exception。

一般来说，最常见的错误有程序进入死循环、内存泄漏等。这种情况下，程序运行时本身无法解决，只能通过其他程序干预。Java 对应的类为 Error 类。Error 类对象由 Java 虚拟机生成并抛弃（通常 Java 程序不对这类异常进行处理）。

异常是程序执行时遇到的非正常情况或意外行为。以下这些情况一般都可以引发异常：代码或调用的代码（如共享库）中有错误，操作系统资源不可用，公共语言运行库遇到意外情况（如无法验证代码）等。常见的有数组下标越界、算法溢出（超出数值表达范围）、除数为零、无效参数、内存溢出等。这种情况不像错误类那样，程序运行时本身可以解决，由异常代码调整程序运行方向，使程序仍可继续运行，直至正常结束。

Java 异常对应的类为 Exception 类。Exception 类对象是 Java 程序处理或抛弃的对象，它有各种不同的子类分别对应于不同类型的异常。 Java 编译器要求程序必须捕获或声明所有的非运行时异常，但对运行时异常可以不做处理。其中类 RuntimeException 代表运行时由 Java 虚拟机生成的异常，原因是编程错误。其他则为非运行时异常，原因是程序碰到了意外情况，如输入/输出异常 IOException 等。

2. 异常关键字

Java 异常处理的关键语句有五个：try、catch、throw、throws、finally。其中，try、catch、finally 三个语句块应注意的问题：

1）try、catch、finally 三个语句块均不能单独使用，三者可以组成 try...catch...finally、try...catch、try...finally 三种结构，catch 语句可以有一个或多个，finally 语句最多一个。

2）try、catch、finally 三个代码块中变量的作用域为代码块内部，分别独立而不能相互访问。如果要在三个块中都可以访问，则需要将变量定义到这些块的外面。

3）若有多个 catch 块，只会匹配其中一个异常类并执行 catch 块代码，而不会再执行别的 catch 块，并且匹配 catch 语句的顺序是由上到下的。

throw、throws 关键字的区别如下：

throw 关键字用于方法体内部，用来抛出一个 Throwable 类型的异常。如果抛出了检查异常，则还应该在头部声明方法可能抛出的异常类型。该方法的调用者也必须检查处理抛出的异常。如果所有的方法都层层上抛获取的异常，最终 JVM 会进行处理，处理也很简单，就是打印异常消息和堆栈信息。如果抛出

的是 Error 或 RuntimeException，则该方法的调用者可选择处理该异常。

throws 关键字用于方法体外部的方法声明部分，用来声明方法可能会抛出某些异常。仅当抛出了检查异常，该方法的调用者才必须处理或者重新抛出该异常。当方法的调用者无力处理该异常的时候，应该继续抛出，而不是囫囵吞枣般地在 catch 块中打印堆栈信息来做处理。

3．Java 异常和 C++异常的区别

在 C++异常处理模型中，它给予程序员最大的自由度和发挥空间，允许程序员抛出任何想要的异常对象，它可以是语言系统中原本所提供的各种简单数据类型（如 int 、float 、double 等），也可以是用户自定义的抽象数据对象（如 class 的 object 实例）。但是 C++ 语言规范中并无此约束，况且由于各个子系统（基础运行库）不是一个厂商（某一个人）统一设计的，所以导致每个子系统设计出的异常对象系统彼此相差甚远。这给最终使用（重用）这些库的程序员带来了很大的不一致性，甚至是很大的麻烦，需要花费很多时间来学习和熟悉这些不同的异常对象子系统。更大的问题是，这些不同子系统之间语义上的不一致，从而造成程序员在最终处理这些异常时，将很难把它们统一起来，例如，MFC 库系统中，采用 CMemoryException 来表示一个与内存操作相关的异常；而其他的库系统中很可能就会采用另外一个 class 来表示内存操作的异常错误。本质上说，缺乏规范和统一所造成的恶劣后果。所以，设计 Java 语言的时候，就要充分考虑到这些问题，把它们纳入语言的统一规范中，这对广大的程序员来说，无疑是一件好事。

实际的 Java 编程中，由于 JDK 平台已经为我们设计好了非常丰富和完整的异常对象分类模型。因此，Java 程序员一般不需要重新定义自己的异常对象，而且即便是需要扩展自定义的异常对象，也往往会从 Exception 派生而来。所以，对于 Java 程序员而言，它一般只需要在它的顶级函数中用 catch（Exception ex）就可以捕获出所有的异常对象，而不必像 C++中采用 catch（…） 那样的语法。

4．异常处理中常见的问题

（1）过于庞大的 try 块

某些程序员把大量的代码放入单个 try 块，试图用一个 catch 语句捕获所有的异常和处理所有可能出现的异常，实际上这是一个坏习惯。原因就在于为了图省事，不愿花时间分析一大块代码中哪几行代码会抛出异常、异常的具体类型是什么。把大量的语句装入单个巨大的 try 块就像是出门旅游时把全部家当塞入集装箱带走，虽然东西是带上了，但要找出来可不容易。

对于这种问题，可以设置多个异常抛出点来解决。异常对象从产生点产生后，到被捕捉后终止生命的全过程中，实际上是一个传值过程，所以，应根据实际来合理控制检测异常个数。catch 语句表示会出现某种异常，而且希望能够处理该异常。所以在 catch 语句中就应该尽量指定具体异常类型，也可使用多个 catch，用于分别处理不同的异常。例如，要捕获一个最明显的异常是 SQLException，这是 JDBC 操作中常见的异常。另一个可能的异常是 IOException，因为它要操作 OutputStreamWriter。显然，在同一个 catch 块中处理这两种截然不同的异常是不合适的。如果用两个 catch 块分别捕获 SQLException 和 IOException 就要好多了。

（2）异常的完整性

在 Java 语言中，如果一个函数运行时可能会向上层调用者函数抛出一个异常，那么，它就必须在该函数的声明中显式地注明（采用 throws 关键字）。否则编译器会报出错误信息："must be caught or declared to be thrown"。其中"must be caught"指在 Java 的异常处理模型中，要求所有被抛出的异常都必须有对应的"异常处理模块"。如果你在程序中利用 throw 出现一个异常，那么在你的程序（函数中）就必须要用 catch 处理这个异常。例如下面的例子中，抛出了一个 Exception 类型的异常，所以在该函数中，就必须有一个 catch，并处理此异常。如果没有这个 catch，Java 语言在编译时就直接拦住这种可能出现错误的情况，不让程序通过。

```
try {
          ......
 // throw Exception
}
catch(Exception ex)
{
   // find Exception
     // hand of it
}
```

"declared to be thrown" 指的是"必须显式地声明某个函数可能会向外部抛出一个异常"，也即是说，如果一个函数内部，它可能抛出了一种类型的异常，但该函数内部又不想用 catch 处理这种类型的异常，此时，它就必须（强制性的）使用 throws 关键字来显式地声明该函数可能会向外部抛出一个异常，以便该函数的调用者知晓并能够及时处理这种类型的异常。如下列代码：

```
class MyException extends Exception {
 MyException() {
 }
}
```

```
class My1Exception extends Exception {
 My1Exception() {
 }
}
class A {
 public int format(String str) {
       int i = Integer.valueOf(str);
       // throw new MyException();
       return i;
 }
 public static void testTester() throws MyException, My1Exception {
 new A().format("S");
 }
}
public class Test
{
 public static void main(String[] args) throws MyException, My1Exception {
       A.testTester();
 }
}
```

5．RuntimeException 异常

在 Java 异常处理中，一般有两类异常：其一，就是通过 throw 语句，程序员在代码中人为抛出的异常（由于运行时动态地监测到了一个错误）；另外一个是系统运行时异常，例如，“被 0 除”、“空字符串”、“无效句柄”等，对于这类异常，程序员实际上完全可以避免它，只要我们写代码时足够小心严谨。因此，为了彻底解决这种隐患，提高程序整体可靠性（不至于因为编码时考虑不周或一个小疏忽导致系统运行时崩溃。），使用 RuntimeException 异常就是为了实现这样的功能。

Java 语言中的这两种异常中，前者叫 checked exception，它是从 java.lang.Exception 类衍生出来的；后者叫 runtime exception，它是从 java.lang.RuntimeException 类衍生出来的。

下面就是一个被零除的例子：

```
public class Test {
 public static void main(String[] args) {
        test();
 }
 static void test() {
        int i = 4;
        int j = 0;
        //运行时，这里将触发了一个 ArithmeticException
        //ArithmeticException 从 RuntimeException 派生而来
```

```
        System.out.println("i / j = " + i / j);
    }
}
```

运行结果如下：

```
java.lang.ArithmeticException: / by zero
at Test.test(Test.java:16)
at Test.main(Test.java:8)
Exception in thread "main"
```

下面是一个空 string 的例子：

```
import java.io.*;
public class Test
{
 public static void main(String[] args)
 {
      test();
 }
 static void test()
 {
      String str = null;
      str.compareTo("abc");
      // 运行时，这里将触发了一个 NullPointerException
      // NullPointerException 从 RuntimeException 派生而来
 }
}
```

针对 RuntimeException 类型的异常，javac 是无法通过编译时的静态语法检测来判断到底哪些函数（或哪些区域的代码）可能抛出这类异常（这完全取决于运行时状态，或者说运行态所决定的），也正因为如此，Java 异常处理模型中的“must be caught or declared to be thrown”规则也不适用于 RuntimeException。但是 Java 虚拟机却需要有效地捕获并处理此类异常。当然，RuntimeException 也可以被程序员显式地抛出，而且为了程序的可靠性，对一些可能出现“运行时异常（RuntimeException）”的代码区域，程序员最好能够及时地处理这些意外的异常，也即通过 catch（RuntimeExcetion）或 catch（Exception）来捕获它们。如下面的示例程序，代码如下：

```
import java.io.*;
public class Test
{
 public static void main(String[] args)
 {
```

```
        try
        {
                test();
        }
        catch (Exception e)
        {
                System.out.println("A Exception!");
                e.printStackTrace();
        }
    }
    static void test() throws RuntimeException
    {
        String str = null;
        str.compareTo("abc");
    }
}
```

面试例题 3：谈谈 final、finally、finalize 的区别。

解析：

1．final

final 可以用于控制成员、方法，或者是一个类是否可被覆写或继承等功能，这些特点使 final 在 Java 中拥有了一个不可或缺的地位，也是学习 Java 时必须要知道和掌握的关键字之一。

（1）final 成员

当在类中定义变量时，若在其前面加上 final 关键字，那就是说，这个变量一旦被初始化，便不可改变，这里不可改变的意思对基本类型来说是其值不可变，而对于对象变量来说是其引用不可变。其初始化可以在两个地方，一是其定义处，二是在构造函数中，两者只能选其一。

还有一种用法是定义方法中的参数为 final。对于基本类型的变量，这样做并没有什么实际意义，因为基本类型的变量在调用方法时是传值的，也就是说，你可以在方法中更改这个参数变量而不会影响到调用语句，然而对于对象变量，却显得很实用，因为对象变量在传递时是传递其引用的，这样，你在方法中对对象变量的修改也会影响到调用语句中的对象变量。当你在方法中不需要改变作为参数的对象变量时，明确使用 final 进行声明，会防止你无意的修改而影响到调用方法。

（2）final 方法

将方法声明为 final 有两个原因，第一就是说明已经知道这个方法提供的功能满足要求，不需要进行扩展，并且也不允许任何从此类继承的类来覆写这个方法，但是仍然可以继承这个方法，也就是说，可以直接使用。第二就是允许编译器将所有对此方

法的调用转化为 inline（行内）调用的机制，它会在调用 final 方法时，直接将方法主体插入到调用处，而不是进行例行的方法调用，例如，保存断点、压栈等，这样可能会使程序效率有所提高。然而当方法主体非常庞大时，或在多处调用此方法时，调用主体代码便会迅速膨胀，可能反而会影响效率，所以要慎用 final 进行方法定义。

（3）final 类

当将 final 用于类时，就需要仔细考虑，因为一个 final 类是无法被任何人继承的，那也就意味着此类在一个继承树中是一个叶子类，并且此类的设计已被认为很完美，不需要进行修改或扩展。对于 final 类中的成员，可以定义其为 final，也可以不是 final。而对于方法，由于所属类为 final 的关系，自然也就成了 final 型的。也可以明确地给 final 类中的方法加上一个 final，但这显然没有意义。

2．finally

finally 关键字是对 Java 异常处理模型的最佳补充。finally 结构使代码总会执行，而不管有无异常发生。使用 finally 可以维护对象的内部状态，并可以清理非内存资源。如果没有 finally，你的代码就会很费解。

3．finalize

根据 Java 语言规范，JVM 保证调用 finalize 函数之前，这个对象是不可达的，但是 JVM 不保证这个函数一定会被调用。另外，规范还保证 finalize 函数最多运行一次。

通常，finalize 用于一些不容易控制，并且非常重要的资源的释放，例如，一些 I/O 的操作、数据的连接。这些资源的释放对整个应用程序是非常关键的。在这种情况下，程序员应该以通过程序本身管理（包括释放）这些资源为主，以 finalize 函数释放资源方式为辅，形成一种双保险的管理机制，而不应该仅仅依靠 finalize 来释放资源。

答案：

1．final 修饰符（关键字）

如果一个类被声明为 final，意味着它不能再派生出新的子类，不能作为父类被继承。因此，一个类不能既被声明为 abstract，又被声明为 final。将变量或方法声明为 final，可以保证它们在使用中不被改变。其初始化可以在两个地方：一是其定义处，也就是说，在 final 变量定义时直接给其赋值；二是在构造函数中。这两个地方只能选其一，要么在定义时给值，要么在构造函数中给值，不能同时既在定义时给了值，又在构造函数中给另外的值，而在以后的引用中只能读取，不可修改。被声明为 final 的方法也同样只能使用，不能重写（override）。

2．finally

在异常处理时提供 finally 块来执行任何清除操作。如果抛出一个异常，那么相匹

配的 catch 子句就会执行，然后控制就会进入 finally 块（如果有的话）。

3．finalize

finalize 是方法名。Java 技术允许使用 finalize()方法在垃圾收集器将对象从内存中清除出去之前做必要的清理工作。这个方法是由垃圾收集器在确定这个对象没有被引用时对这个对象调用的。它是在 Object 类中定义的，因此，所有的类都继承了它。子类覆盖 finalize()方法以整理系统资源或者执行其他清理工作。finalize()方法是在垃圾收集器删除对象之前对这个对象调用的。

面试例题 4：try{}里有一个 return 语句，那么紧跟在这个 try 后的 finally{}里的 code 会不会被执行，什么时候被执行?在 return 前还是后？[中国大陆著名杀毒软件公司 JS2009 年 10 月笔试题]

A．会执行，在 return 前执行　　C．不会执行

B．会执行，在 return 后执行　　D．会抛出异常

解析：finally 结构在 Java、C++、C#中，代码总会执行，而 Java、C++、C#里很多回收机制代码就在 finally 里，而一个函数 return 后就会销毁，因此，要在 return 前执行。

答案：A

面试例题 5：下面四段 Java 程序中哪些不能被编译通过？[加拿大著名通讯软件公司 N2009 年 12 月笔试题]

程序 1：

```
import java.io.*;
public class Test {
 public static void main(String[] args) {
      try {
            test();
      } catch (Exception ex) {
            ex.printStackTrace();
      }
 }
 static void test() {
      try {
            throw new Exception("test");
      } catch (Exception ex) {
            ex.printStackTrace();
      }
 }
}
```

程序 2：

```
import java.io.*;
public class Test {
 public static void main(String[] args) {
      try {
            test();
      }

      catch (Exception ex) {
            ex.printStackTrace();
      }
 }
 static void test() {
      throw new Exception("test");
 }
}
```

程序 3：

```
import java.io.*;
public class Test {
 public static void main(String[] args) {
      try {
            test();
      } catch (Exception ex) {
            ex.printStackTrace();
      }

 }
 static void test() throws Exception {
      throw new Exception("test");
 }
}
```

程序 4：

```
import java.io.*;
public class Test {
 public static void main(String[] args) {
      try {

            test();
      } catch (IOException ex) {
            ex.printStackTrace();
      }
 }
 static void test() throws Exception {
 }
}
```

A．程序 1 和程序 2　　C．程序 1 和程序 3

B．程序 3 和程序 4　　D．程序 2 和程序 4

解析：程序 1 是很经典的异常捕捉方法，可以通过编译；程序 2 虽然在这里能够捕获到 Exception 类型的异常，但是 test 函数中单独 throw（抛）一个异常却不加捕获是错误的，所以不能通过编译；程序 3 由于函数显式地声明了可能抛出 Exception 类型的异常，所以这种写法又能够被编译通过；程序 4 中，虽然 test() 函数并没有真正抛出一个 Exception 类型的异常，但是由于函数在声明时，表示它可能抛出一个 Exception 类型的异常。所以这里必须 catch 一个 Exception 类型的异常，否则不能被编译通过。

答案：D

面试例题 6：下列输出结果是什么？[中国杭州著名 B2B 软件公司 T2009 年 10 月笔试题]

```
import java.io.*;
public class Test
{
 public static void main(String[] args)
 {
      try{
           test();
              System.out.println("condition 1");
           }catch (ArrayIndexOutOfBoundsException e){
              System.out.println("condition 2");
           }catch (Exception e){
              System.out.println("condition 3");
           }finally{
              System.out.println("finally");
           }
      //test();
 }
 static void test()
 {
      String str = "cc";
      str.compareTo("abc");

 }
}
```

解析：因为 test()方法运行是正常的，所以不会抛出异常。

答案：

condition 1

finally

5.7　反射

面试例题：什么是 Reflection（反射）？其他语言有这种特点吗？

解析：反射主要是指程序可以访问、检测和修改它本身的状态或行为的一种能力。这一概念的提出很快引发了计算机科学领域关于应用反射性的研究。它首先被程序语言的设计领域所采用，并在 LISP 和面向对象方面取得了成绩，其中 LEAD/LEAD++、OpenC++、MetaXa 和 OpenJava 等就是基于反射机制的语言。最近，反射机制也被应用到了视窗系统、操作系统和文件系统中。

反射本身并不是一个新概念，它可能会被联想到光学中的反射概念，尽管计算机科学赋予了反射概念新的含义，但是，从现象上来说，它们确实有某些相通之处。在计算机科学领域，反射是指一类应用，它们能够自描述和自控制。也就是说，这类应用通过采用某种机制来实现对自己行为的描述（self-representation）和监测（examination），并能根据自身行为的状态和结果，调整或修改应用所描述行为的状态和相关的语义。可以看出，与一般的反射概念相比，计算机科学领域的反射不单单指反射本身，还包括对反射结果所采取的措施。所有采用反射机制的系统（即反射系统）都希望使系统的实现更开放。可以说，实现了反射机制的系统都具有开放性，但具有开放性的系统并不一定采用了反射机制，开放性是反射系统的必要条件。一般来说，反射系统除了满足开放性条件外，还必须满足原因连接（Causally-connected）。所谓原因连接，是指对反射系统自描述的改变能够立即反映到系统底层的实际状态和行为上的情况，反之亦然。开放性和原因连接是反射系统的两大基本要素。

答案：Java 中的反射是一种强大的工具，它能够创建灵活的代码，这些代码可以在运行时装配，无须在组件之间进行链接。反射允许在编写与执行时，使程序代码能够接入装载到 JVM 中的类的内部信息，而不是源代码中选定的类协作的代码。这使反射成为构建灵活应用的主要工具。需注意的是，如果使用不当，反射的成本会很高。

Java 中的类反射 Reflection 是 Java 程序开发语言的特征之一，它允许运行中的 Java 程序对自身进行检查，或者说“自审”，并能直接操作程序的内部属性。Java 的这一能力在实际应用中也许用得不是很多，但是在其他的程序设计语言中根本就不存在这一特性。例如，Pascal、C 或者 C++中就没有办法在程序中获得与函数定义相关的信息。

第6章

传递与引用

Java 语言明确说明取消了指针，因为指针往往是在带来方便的同时导致代码不安全的根源，而且还会使程序变得非常复杂和难以理解，滥用指针写成的代码不亚于使用早已臭名昭著的 GOTO 语句。Java 放弃指针的概念绝对是极其明智的。但这只是在 Java 语言中没有明确的指针定义，实质上，每一个 new 语句返回的都是一个指针的引用，只不过在大多数时候 Java 不用关心如何操作这个“指针”，更不用像在操作 C++的指针那样胆战心惊，唯一要多注意的是在给函数传递对象的时候。

传值与引用问题中的静态变量、私有变量、clone 等问题也是各大公司的常备考点。本章不对传值与引用基本知识做回顾和分析（请参考其他经典著作），只是通过对各公司面试题目进行全面仔细的解析，帮读者解决其中的难点。

以下的考题来自真实的笔试资料，希望读者先不要看答案，自我解答后再与答案加以比对，找出自己的不足。

6.1 传值与传引用

面试例题 1：Explain call by value and call by reference. Which of these two does Java support?（解释：Java 中是传值还是传引用）[中国大陆某著名网络公司 B2009 年 9 月面试题]

解析：就像光到底是波还是粒子的问题一样众说纷纭，对于 Java 参数是传值还是传引用的问题，也有很多错误的理解和认识。我们首先要搞清楚一点就是：不管 Java 参数的类型是什么，一律传递参数的副本。对此，*thinking in Java* 一书给出的经典解释是 When you're passing primitives into a method, you get a distinct copy of the primitive. When you're passing a reference into a method, you get a copy of the reference.（如果 Java 是传值，那么传递的是值的副本；如果 Java 是传引用，那么传递的是引用的副本。）

在 Java 中，变量分为以下两类：

① 对于基本类型变量（int、long、double、float、byte、boolean、char），Java 是传值的副本。（这里 Java 和 C++相同）

② 对于一切对象型变量，Java 都是传引用的副本。其实传引用副本的实质就是复制指向地址的指针，只不过 Java 不像 C++中有显著的*和&符号。（这里 Java 和 C++不同，在 C++中，当参数是引用类型时，传递的是真实引用而不是引用副本）

需要注意的是：String 类型也是对象型变量，所以它必然是传引用副本。不要因为 String 在 Java 里面非常易于使用，而且不需要 new，就被蒙蔽而把 String 当做基本变量类型。只不过 String 是一个非可变类，使得其传值还是传引用显得没什么区别。

对基本类型而言，传值就是把自己复制一份传递，即使自己的副本变了，自己也不变。而对于对象类型而言，它传的引用副本（类似于 C++中的指针）指向自己的地址，而不是自己实际值的副本。为什么要这么做呢？因为对象类型是放在堆里的，一方面，速度相对于基本类型比较慢，另一方面，对象类型本身比较大，如果采用重新复制对象值的办法，浪费内存且速度又慢。就像你要张三（张三相当于函数）打开仓库并检查库里面的货物（仓库相当于地址），有必要新建一座仓库（并放入相同货物）给张三么？没有必要，你只需要把钥匙（引用）复制一把寄给张三就可以了，张三会拿备用钥匙（引用副本，但是有时效性，函数结束，钥匙销毁）打开仓库。

在这里提一下，很多经典书籍包括 *thinking in Java* 都是这样解释的："不管是基本类型还是对象类型，都是传值。"这种说法也不能算错，因为它们把引用副本也当做是一种"值"。但是笔者认为：传值和传引用本来就是两个不同的内容，没必要把两者弄在一起，弄在一起反而更不易理解。

下面看几个例子。

例 1：

```
public class Test {
    public static void test(boolean test) {
        test = ! test;
        System.out.println("In test(boolean) : test = " + test);
    }
    public static void main(String[] args) {
        boolean test = true;
        System.out.println("Before test(boolean) : test = " + test);
        test(test);
        System.out.println("After test(boolean) : test = " + test);
    }
}
```

运行结果：

```
Before test(boolean) : test = true
In test(boolean) : test = false
After test(boolean) : test = true
```

不难看出，虽然在 test(boolean) 方法中改变了传进来的参数值，但对这个参数源变量本身并没有影响，即对 main (String[]) 方法中的 test 变量没有影响，说明参数类型是简单类型的时候，是按值传递的。以参数形式传递简单类型的变量时，实际上是将参数的值作为一个副本传进方法函数的，那么在方法函数中不管怎么改变其值，其结果都是只改变了副本的值，而不是源值。

例 2：

```
public class Test {
    public static void test(StringBuffer str) {
        str.append(", World!");
    }
        public static void main(String[] args) {
        StringBuffer string = new StringBuffer("Hello");
        test(string);
        System.out.println(string);
    }
}
```

运行结果如下：

```
Hello, World!
```

test(string)调用了 test(StringBuffer)方法，并将 string 作为参数传递了进去。这里 string 是一个引用，Java 对于引用形式传递对象类型的变量时，实际上是将引用作为一个副本传进方法函数的。那么这个函数里面的引用副本所指向的是什么呢？是对象的地址。通过引用副本（复制的钥匙）找到地址（仓库）并修改地址中的值，也就修改了对象。

例 3：

```
public class Test {
    public static void test(String str) {
        str = "World";
    }
    public static void main(String[] args) {
        String string = "Hello";
        test(string);
        System.out.println(string);
    }
}
```

运行结果如下：

```
Hello
```

为什么会这样呢？这是因为当执行 str = "World";时，其过程为：首先系统会自动生成一个新 String 对象，并把这个新对象的值设为"World!"，然后把这个对象的引用赋给 str（可以理解为 str 这把钥匙原来是指向"Hello"这个仓库，但是现在要求 str 这把钥匙重新指向"World"这个仓库）。我们必须清楚的一点是 String 类是 final 类型的，因此，你不可以继承和修改这个类。str = "World";其实是隐含的让 Java 生成一个新的 String 对象。既然对象都是新的，那就与原来的"Hello"没有任何关系。当函数结束，str 作用消失，原来的内存地址上的内容未加改变，所以打印结果仍然是 Hello。而例 2 中的 str.append(", World!");就不同了，StringBuffer 是产生一块内存空间，进行相关的增、删、改操作都在其中进行，所以为其添加一句",World!"仍然是在同一段内存地址上进行，str 所指向的引用并没有改变。

答案：对于基本类型变量，Java 是传值的副本；对于一切对象型变量，Java 都是传引用的副本。

面试例题 2：下列代码的输出结果是多少，为什么？[美国著名软件公司 I2009 年 11 月面试题]

```
class Value {
 public int i = 15;
}

public class Test {
 public static void main(String[] args) {
       Test t = new Test();
       t.first();
 }

 public void first() {
       int i = 5;
       Value v = new Value();
       v.i = 25;
       second(v, i);
       System.out.println(v.i);
 }

 public void second(Value v, int i) {
       i = 0;
       v.i = 20;
       Value val = new Value();
       v = val;
       System.out.println(v.i + " " + i);
 }
}
```

解析：方法参数有基本类型，如 int 等，另外一种类型是 Object 对象。Java 方法参数传对象，传的是对这个对象引用的一份副本，即地址值，跟原来的引用都是指向

同一个对象。下面我们看一段程序：

```
public class Test
 {
       public static void main(String[] args)
            {
                char ch[] = {'H','e','l','l','o'};
               change(ch);
                 System.out.println(ch);
            }

       public static void change(char ch[])
       {
               ch[0] = 'C';
       }
 }
```

上面程序中，数组传值的本质是传地址值的副本。好比说一个仓库有一把钥匙 A，传值（传副本）相当于现在重新配了一把一模一样的钥匙 B，但是还是指向这个仓库。当 ch[0] = 'C'时，相当于通过这把备用钥匙 B 改变了仓库里的物资（比如原来装着韭菜，现在换成萝卜）。等备用钥匙的寿命结束（函数结束），再用主钥匙打开仓库时就会发现仓库已经改变（Hello 变成了 Cello）。

对于数组，函数里“ch[0]= 'C';”语句的含义是将 ch 所指向的内存中偏移是 0 的内容由"H"换成"C"，也就是说，ch 所指的对象的内容被改变了，但 ch 并没有变。

在本题中

```
public void second(Value v, int i){
 i=0;
 v.i=20;  //通过引用的副本改变原对象的值为 20
 Value val = new Value(); //new 出了一个新的对象
 v=val; //引用的副本指向了一个新的 Object
 System.out.println(v.i+" "+i); //打印新对象的值
}
```

对象 v 也是传一份引用的副本，v.i=20; 通过引用的副本改变原对象的值为 20。但在语句 Value val = new Value();中，new 引出了一个新的对象，然后执行 v=val; 相当于引用的副本指向了一个新的 Object。所以 v.i=15 是改变新的 Object 的值，而不改变原对象的值。

答案：15 0 20

6.2 静态变量与私有变量

面试例题 1：关于静态变量的创建，哪一个选项是正确的？（　　）

A．一旦一个静态变量被分配，它就不允许改变

B．一个静态变量在一个方法中创建，它在被调用的时候值保持不变

C．在任意多个类的实例中，一个静态变量的实例只存在一个

D．一个静态的标识符只能被应用于 primitive value

解析：选项 A 是对常量的描述。选项 B 是为了迷惑那些习惯使用 VB 的人。选项 D 所说的静态标识符可以被应用到类（但只是一个内部类）、方法和变量中。

答案：C

面试例题 2：当编译和运行下列代码时会出现什么情况？

```
public class Sandys{
private int court;
public static void main(String argv[]){
       Sandys s = new Sandys(99);
       System.out.println(s.court);
       }
Sandys(int ballcount){
       court=ballcount;
       }
}
```

A．编译时错误，court 变量定义的是私有变量

B．编译时错误，当 System.out.println 方法被调用时 s 没有被初始化

C．编译和执行时没有输出结果

D．编译运行时输出结果是 99

解析：事实上，变量 court 被定义为私有变量，并不能阻止构造器对它进行初始化，所以 court 被初始化成 99。

答案：D

面试例题 3：当你编译和运行下面的代码时，会出现下面选项中的哪种情况？（　　）

```
public class Pvf{

static boolean Paddy;
public static void main(String argv[]){
       System.out.println(Paddy);
       }

}
```

A．编译时错误　　B．编译通过并输出结果 false

C．编译通过并输出结果 true　　D．编译通过并输出结果 null

解析：定义在类里面的变量会被赋予一个默认的值，而布尔类型的默认初始值是 false，所以此题的输出结果是 false。

答案：B

面试例题 4：给定下面的程序。

```
public class Sytch
{
int x=2000;
public static void main(String argv[])
{
System.out.println("Ms "+argv[1]+"Please pay $"+x);
 }

}
```

如果用命令行参数 Java Sytch Jones Diggle 编译和运行上面的程序，会出现下面哪个结果？（　　）

A．编译通过并输出 Ms Diggle Please pay $2000 结果

B．编译时错误

C．编译通过并输出 Ms Jones Please pay $2000

D．编译通过但是运行时错误

解析：因为 main 方法是静态的，不能访问一个非静态的变量 x。

答案：B

6.3 输入/输出流

面试例题 1：假设任何异常处理已经被创建，下列哪个选项是创建 RandomAccessFile 类实例的正确选项？（　　）

A．RandomAccessFile raf=new RandomAccessFile("myfile. txt","rw");

B．RandomAccessFile raf=new RandomAccessFile(new Data InputStr-eam());

C．RandomAccessFile raf=new RandomAccessFile("myfile. txt");

D. RandomAccessFile raf=new RandomAccessFile(new File ("myfile.txt"));

解析：在 Java 的 I/O 结构中，RandomAccessFile 是比较不寻常的类，它直接继承于 Object，它并不属于 Streams 结构的一部分。

答案：A

面试例题 2：需要读一个比较大的文本文件，这个文件里有很多字节的数据，那么下列最合适读这类文件的选项是哪个？（　　）

A．new FileInputStream("file.name");

B．new InputStreamReader(new FileInputStream("file.name"));

C．new BufferedReader(new InputStreamReader(new FileInputStream("file.name")));

D．new RandomAccessFile raf=new RandomAccessFile("myfile.txt","+rw");

解析：这道题最主要的是具有很多字节数的文本文件，那么就要考虑到 BufferedReader。

答案：C

面试例题 3：请给出一段代码描述字符串写入文件。

答案：代码如下。

```
import java.io.*;

public class Test
{
 public static void main(String args[])
 {
      try
      {
           FileOutputStream out = new FileOutputStream
              ("FileName.txt");
           out.write("字符串写入文件".getBytes());
           out.close();
      }
      catch (IOException ioe)
      {
      }
 }//main
}
```

面试例题 4：写一个 Java 应用程序，从键盘输入两个整数，然后输出它们的平方值及立方值。

解析：在 Java 中没有像 C 语言那样有一个专供接受从键盘输入值的 scanf 函数，所以一般的做法是从键盘输入一行字符，保存到字符串 s 中，再将字符组成的字符串 s 转换为整型数据后返回。

答案：代码如下。

```
//package cuboid;
```

```
import java.io.*;

class InputData {                          //定义从键盘输入数据的类
  static private String s = "";
  static public void input() {             //从键盘输入一行字符,保存到字符串 s 中
    BufferedReader bu = new BufferedReader(
       new InputStreamReader(System.in)
       );
    try {
      s = bu.readLine();
    }
    catch (IOException e) {}               //捕获输入/输出异常
  }

  static public int getInt() {             //静态方法可直接用类名调用
    input();
    return Integer.parseInt(s);            //将字符组成的字符串 s 转换为整型数据后返回
  }
}

class Result {
  void print(int d) {
    System.out.println(d + "的平方:" + d * d);
    System.out.println(d + "的立方:" + d * d * d);
  }
}

public class PrintResult {
  public static void main(String[] args) {
    Result result = new Result();
    System.out.println("请输入一个整数:");
    int a = InputData.getInt();
    result.print(a);
  }
}
```

面试例题 5：8、64、256 都是 2 的阶次方数（例如，8 是 2 的 3 次方），用 Java 编写程序来判断一个整数是不是 2 的阶次方数，并说明哪个方法更好。

解析：如果一个数是 2 的阶次方数，那么它的二进制数的首位一般是 1，后面接若干个 0。比如 8 就是 1000，64 是 1000000。如果将这个数减 1 后，再与该数做和（&）运算，则应该全为 0（例如，8 与 7，一个二进制数是 1000，一个二进制数是 111，它们做和运算后全为 0）。所以((d-1)&(d)) == 0。

答案：代码如下。

```
//package cuboid;
```

```
import java.io.*;

class InputData {                    //定义从键盘输入数据的类
  static private String s = "";
  static public void input() {       //从键盘输入一行字符,保存到字符串 s 中
    BufferedReader bu = new BufferedReader(
        new InputStreamReader(System.in)
        );
    try {
      s = bu.readLine();
    }
    catch (IOException e) {}         //捕获输入/输出异常
  }

  static public int getInt() {       //静态方法可直接用类名调用
    input();
    return Integer.parseInt(s);      //将字符组成的字符串 s 转换为整型数据后返回
  }
}

class Result {
  void print(int d) {

    if(((d-1)&(d)) == 0 && (d!=0))

      System.out.println("是 2 的阶次");
    else
      System.out.println("不是 2 的阶次");
  }
}

public class PrintResult {
  public static void main(String[] args) {
    Result result = new Result();
    System.out.println("请输入一个整数:");
    int a = InputData.getInt();
    result.print(a);
  }
}
```

6.4 序列化

面试例题：如何实现 Java 的序列化？[英国著名图形软件公司面试题]

解析：Java 的“对象序列化”能将一个实现了 Serializable 接口的对象转换成一组

byte，这样日后要用这个对象的时候，就能把这些 byte 数据恢复出来，并据此重新构建那个对象。这一点甚至在跨网络的环境下也是如此，这就意味着序列化机制能自动补偿操作系统方面的差异。也就是说，可以在 Windows 机器上创建一个对象，序列化之后，再通过网络传到 UNIX 机器上，最后在那里进行重建。不用担心在不同的平台上数据是怎样表示的，以及 byte 顺序怎样，或者别的什么细节。

对象序列化能实现“轻量级的 persistence（lightweight persistence）”。所谓 persistence，是指对象的生命周期不是由程序是否运行决定的，在程序的两次调用之间对象仍然还活着。通过“将做过序列化处理的对象写入磁盘，等到程序再次运行的时候再把它读出来”，可以达到 persistence 的效果。之所以说“轻量级”，是因为不能用像“persistent”这样的关键词来直接定义一个对象，然后让系统去处理所有的细节（虽然将来有可能会这样）。相反，必须明确地进行序列化（serialize）和解序列化（deserialize）。

之所以要在语言里加入对象序列化，是因为要用它来实现两个重要的功能。Java 的远程方法调用（Remote Method Invocation，简称 RMI）能像调用自己机器上的对象那样去调用其他机器上的对象。当向远程对象传递消息的时候，就需要通过对象序列化来传送参数和返回值。对 JavaBean 来说，对象序列化也是必不可少的。Bean 的状态信息通常是在设计时配置的，这些状态信息必须保存起来，供程序启动的时候用，对象序列化就负责这个工作。

答案：序列化一个对象还是比较简单的，只要让它实现 Serializable 接口就行了（这是一个“标记接口（tagging interface）”，没有任何方法）。但是，当语言引入序列化概念之后，它的很多标准类库的类，包括 primitive 的 wrapper 类、所有的容器类，以及别的很多类，都会相应地发生改变，甚至连 Class 对象都会被序列化。

要想序列化对象，必须先创建一个 OutputStream，然后把它嵌进 ObjectOutputStream。这时就能用 writeObject( )方法把对象写入 OutputStream。读的时候需要把 InputStream 嵌到 ObjectInputStream 中，然后再调用 readObject( )方法。不过这样读出来的只是一个 Object 的 reference，因此，在用之前，还得先下传。

对象序列化不仅能保存对象的副本，而且还会跟着对象中的 reference 把它所引用的对象也保存起来，然后再继续跟踪那些对象的 reference，以此类推。这种情形常被称为“单个对象所联结的‘对象网’”。这个机制所涵盖的范围不仅包括对象的成员数据，而且还包含数组里面的 reference。如果要自己实现对象序列化，那么编写跟踪这些链接的程序将会是一件非常痛苦的任务。但是，Java 的对象序列化就能精确无误地做到这一点，毫无疑问，它的遍历算法是做过优化的。

第7章

循环、条件、概率

递归过程的执行总是一个过程体未执行完，就带着本次执行的结果又进入另一轮过程体的执行……如此反复，不断深入，直到某次过程的执行遇到终止递归调用的条件成立时，则不再深入，而执行本次的过程体余下的部分，然后又返回到上一次调用的过程体中，执行其余下的部分……如此反复，直到回到起始位置上，才最终结束整个递归过程的执行，得到相应的执行结果。递归过程的程序设计的核心就是参照这种执行流程，设计出一种适合“逐步深入，而后又逐步返回”的递归调用模型，以解决实际的面试例题。

在程序设计面试中，一个能够完成任务的解决方案是最重要的，解决方案的执行效率要放在第二位考虑。因此，除非试题另有要求，你应该从最先想到的解决方案入手。如果它是一个递归性的方案，你不妨向面试官说明一下递归算法天生的低效率问题，表示你知道这些事情。有时候你同时想到两个解决方案：递归的和循环的，并且实现方式差不多，你可以把两个都向考官介绍一下。

7.1 典型递归问题

面试例题 1：把一个数组里的数组合全部列出，比如 1 和 2 列出来为 1,2,12,21。[日本著名软件公司 S2007 年面试题]

解析：本题考查的是循环递归算法。

答案：代码如下。

```
import java.util.*;
import java.io.*;
public class Test {
```

```
    public static void main(String[] args) throws Exception {
     String[] array = new String[] {
         "1", "2", "3", "4"
     };
     listAll(Arrays.asList(array), "");
    }

    public static void listAll(List candidate, String prefix) {
//      if (candidate.isEmpty()) {
        System.out.println(prefix);
//      }

     for (int i = 0; i < candidate.size(); i++) {
      List temp = new LinkedList(candidate);
      listAll(temp, prefix + temp.remove(i));
     }
    }
}
```

面试例题 2：试用递归的方法编程计算菲波那契数列的通项 f(n)，已知 f1=1，f2=1，以后每项都是前两项的和。[中国大陆著名杀毒软件公司 J2005，2009 年面试题]

解析：本题的考点是递归问题。

答案：代码如下。

```
import java.util.*;
import java.io.*;
public class fibonacci
{ public static int k=0;
  public static void main(String[] args) throws Exception
  {
    Scanner cin=new Scanner(System.in);
    long a=cin.nextLong();
    System.out.println(fibonacci(a));
    System.out.println("共递归调用了"+k+"次");
   }
   public static long fibonacci(long m){
   if(m==0||m==1)
   { k++;
    return m;
   }
    else return fibonacci(m-1)+fibonacci(m-2);
   }

}
```

面试例题 3：一个字符串中可能包含 a～z 中的多个字符，如有重复，如 String data＝"aavzcadfdsfsdhshgWasdfasdf"，求出现次数最多的那个字母及次数，如有多个重复的则都求出。[中国大陆著名杀毒软件公司 J2007 年面试题]

解析：主要思路如下。

（1）引入 TreeSet：通过集合快速找到所有出现的字符串。

（2）引入 ArrayList：为了快速排序，再通过 StringBuffer 生成排序后的字符串。

（3）通过 String api 中的基本方法 indexOf lastIndexOf 来计算 TreeSet 中每个字符串的最大值。

（4）如果出现相同的，则把相同的都记录在一个列表中。

（5）记录第一个出现次数最多的字符（为了计算多个字符串相同情况）。

（6）计算最大字符串列表中哪些才是真正出现次数最多的。

答案：代码如下。

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.TreeSet;

public class Test {

 public static void main(String[] args) {
      // TODO Auto-generated method stub
      String input = "aavzcadfdsfsdhshgWasdfasdfdddaaa";
      new Test().doString(input);
 }

 public void doString(String input) {
      char[] chars = input.toCharArray();
      ArrayList lists = new ArrayList();
      TreeSet set = new TreeSet();
      for (int i = 0; i < chars.length; i++) {
           lists.add(String.valueOf(chars[i]));
           set.add(String.valueOf(chars[i]));
      }
      System.out.println(set);
      Collections.sort(lists);
      System.out.println(lists);

      StringBuffer sb = new StringBuffer();
      for (int i = 0; i < lists.size(); i++) {
           sb.append(lists.get(i));
      }

      input = sb.toString();
```

```
        System.out.println(input);
        int max = 0;
        String maxString = "";
        ArrayList maxlist = new ArrayList();

        Iterator its = set.iterator();
        while (its.hasNext()) {
            String os = (String) its.next();
            int begin = input.indexOf(os);
            int end = input.lastIndexOf(os);
            int value = end - begin + 1;
            if (value > max) {
                max = value;
                maxString = os;
                maxlist.add(os);
            } else if (value == max) {
                maxlist.add(os);
            }
        }

        int index = 0;
        for (int i = 0; i < maxlist.size(); i++) {
            if (maxlist.get(i).equals(maxString)) {
                index = i;
                break;
            }
        }
        System.out.print("max data");
        for (int i = index; i < maxlist.size(); i++) {
            System.out.print(maxlist.get(i) + " ");
        }
        System.out.println();
        System.out.println("max" + max);
    }

}
```

面试例题 4：利用 1、2、2、3、4 这 5 个数字，用 Java 写一个 main 函数，打印出所有不同的排列，如 12234、21234 等，要求打印出来的不能有重复。[中国大陆著名杀毒软件公司 J2007 年面试题]

解析：这道题的难点在于如何消减重复的 2。对于任意一个串利用递归进行排列时，循环串中的每个字符到第一个字符进行递归。如果串中字符出现重复，则重复的字符只可以利用递归算法一次，即只要与前面相同的字符循环到第一个字符时，不调用递归就可以避免重复。

答案：代码如下。

```
import java.util.*;
public class test123 {
static int count=0;
   public static void main(String[] arg) {
//Scanner r=new Scanner(System.in);
       String s="1223";
       String s2="1232";
       int t = 1&0;
       System.out.println(t);
        /*
       int cc=s2.indexOf(s2.charAt(1));
       int cc2=s2.indexOf(s2.charAt(2));
       System.out.println(s2.charAt(0));
       System.out.println(s2.charAt(1));
       System.out.println(s2.charAt(2));
       System.out.println(s2.charAt(3));
       System.out.println(s2.indexOf(s2.charAt(0)));
       System.out.println(s2.indexOf(s2.charAt(1)));
       System.out.println(s2.indexOf(s2.charAt(2)));
       System.out.println(s2.indexOf(s2.charAt(3)));
       */
       int index[]=new int[s.length()];
       for(int i=0; i<s2.length(); i++)      //该循环将所有的字符第一次出现的位置记录在数组
                                             //index 中
       {index[i]=s2.indexOf(s2.charAt(i));
       //System.out.println(s2.indexOf(s2.charAt(i)));
       }
       Pailie(s,"");
       //System.out.println("Total:"+count);
   }
   static void Pailie(String s,String p) {
       if(s.length()<1) {
       //System.out.println(p);          //字符串长度小于 1，换行
       count++;
       }
       else {
       int index[]=new int[s.length()];
       for(int i=0; i<s.length(); i++) //该循环将所有的字符第一次出现的位置记录在数组 index 中
       {index[i]=s.indexOf(s.charAt(i));
       //System.out.println(s.charAt(i));
       }
       for(int i=0; i<s.length(); i++) {
       if(i==index[i])                 //只有当循环数与第一次记录数相等时才递归，保证相同字符
                                       //中的第一个调用
       {
           //System.out.println("ppp"+index[i]);
           Pailie(s.substring(1),p+s.substring(0,1));  //递归，打印其他字符
```

```
        }
            s=s.substring(1)+s.substring(0,1);          //循环移位
           }
        }
     }
}
```

7.2 循环与条件

面试例题 1：下列程序的输出结果是什么？[中国大陆某软件公司 Y2009 年 9 月笔试题]

```
import java.util.*;
public  class  Test
{
  public static void main(String[]  args)
  {
     for(int i = 0; i <= 10; i++)
           Integer k= new Integer(i);
           System.out.println( "Hello world");
  }
}
```

A．Hello world　　B．Hello　　C．编译错误　　D．以上答案都不对

解析：局部变量声明的作用范围是在一个块内，也可以理解为在“{}”内。for 循环可以不使用“{}”的，但仅限于执行语句（其中并不包括变量声明语句），由于这段代码中 Integer k 的作用范围在整个 main 方法中，这样就造成了变量重复定义的错误。所以，在编译时会出错。若要改正，只要加上一对花括号，让变量声明在块内就可以，代码如下所示：

```
import java.util.*;
public  class  Test
{
  public  static  void  main(String[]  args)
  {
     for(int  i  =  0;  i  <=  10;  i++)
           {Integer  k  =  new  Integer(i);}          //添加花括号
           System.out.println( "Hello world");
  }
}
```

这道题目告诉我们在编写代码时尽量使用标准的语句，哪怕循环体内只有一行，也最好用花括号括起来，以免产生歧义。

答案：C

面试例题 2：Consider a function which, for a given whole number n, returns the number of ones required when writing out all numbers between 0 and n.

For example, f(13)=6. Notice that f(1)=1. What is the next largest n such that f(n)=n? (解释：有一个整数 n，写一个函数 f(n)，返回 0～n 之间出现的“1”的个数。比如 f(1)=1；f(13)=6（1、10、11、12、13 一共 6 个 1），问一个最大的 f(n)=n 中的 n 是什么？）[美国著名搜索引擎公司 G2009 年 12 月笔试题目]

解析：这个题目的关键在效率上，在没有发现很科学、快速地计算出个数的情况下，可以采用缓存的机制。因为就 200000 来说，计算时间已经无法忍受了，因此，可以把前面的计算结果缓存下来，把每次的结果保存好，就不用每次都重新计算，从而可提高效率。例如，计算 101，只要把之前 1～100 的结果与 101 相加就行了。

答案：代码如下：

```
public class Test {
    public static void main(String[] args) {
        int n = 2;
        int res=1;
        while((getOnly(n)+res) != n) {
            res+=getOnly(n);
            ++n;
        }
        System.out.println(n);
    }
    static int getOnly(int num){
        int number=0;
        String s=num+"";
        int len=s.length();
        if(len!=0){
              for(int i=0;i<len;i++){
                   char a=s.charAt(i);
                   if(a=='1'){
                        number++;
                   }
              }
        }return number;
    }
}
```

面试例题 3：Which of the choices below correctly describes the amount of time used by the following code? （下面哪个选项正确地描述了代码运行的调度次数？）

```
n=10;
for(i=1; i<n; i++)
    for(j=1; j<n; j+=n/2)
```

```
for(k=1; k<n; k=2*k)
  x = x +1;
```

A．Θ(n^3)　　B．Θ(n2logn)　　C．Θ(n(log n)*2)　　D．Θ(n log n)

解析：本题考查面试者对时间复杂度的理解。本题涉及如下概念。

1．时间频度

一个算法执行所耗费的时间从理论上是不能算出来的，必须上机运行测试才知道。但不可能也没有必要对每个算法都上机测试，只需知道哪个算法花费的时间多，哪个算法花费的时间少就可以了，并且一个算法花费的时间与算法中语句的执行次数成正比例，哪个算法中语句执行次数多，它花费的时间就多。一个算法中的语句执行次数称为语句频度或时间频度。记为 T(n)。

2．时间复杂度

在刚才提到的时间频度中，n 称为问题的规模，当 n 不断变化时，时间频度 T(n)也会不断变化。但有时我们想知道它变化时呈现什么规律。为此，引入了时间复杂度的概念。

一般情况下，算法中基本操作重复执行的次数是问题规模 n 的某个函数，用 T(n)表示，若有某个辅助函数 f(n)，使得当 n 趋近于无穷大时，T（n）/f(n)的极限值为不等于零的常数，则称 f(n)是 T(n)的同数量级函数。记为 T(n)=O(f(n))，称 O(f(n)) 为算法的渐进时间复杂度，简称时间复杂度。

在各种不同的算法中，若语句的执行次数为一个常数，则时间复杂度为 O(1)，另外，在时间频度不相同时，时间复杂度有可能相同，如 T(n)=n^2+3n+4 与 T(n)=4n^2+2n+1，它们的频度不同，但时间复杂度相同，都为 O(n^2)。

按数量级递增排列，常见的时间复杂度有：

```
常数阶 O(1)
对数阶 O(log(2)n)
线性阶 O(n)
线性对数阶 O(nlog(2)n)
平方阶 O(n^2)
立方阶 O(n^3)
...
k 次方阶 O(n^k)
指数阶 O(2^n)
```

随着规模 n 的不断增大，上述时间复杂度不断增大，算法的执行效率越低。

3．算法的时间复杂度

若要比较不同的算法的时间效率，受限要确定一个度量标准，最直接的办法就是将计算法转化为程序，在计算机上运行，通过计算机内部的计时功能获得精确的时间，

然后进行比较。但该方法受计算机的硬件、软件等因素的影响，会掩盖算法本身的优劣，所以，一般采用事先分析估算的算法，即撇开计算机软硬件等因素，只考虑问题的规模（一般用自然数 n 表示），认为一个特定的算法的时间复杂度只采取于问题的规模，或者说它是问题的规模函数。

为了方便比较，通常的做法是，从算法中选取一种对于所研究的问题（或算法模型）来说是基本运算的操作，以其重复执行的次数作为评价算法时间复杂度的标准。该基本操作多数情况下是由算法最深层环内的语句表示的，基本操作的执行次数实际上就是相应语句的执行次数。

一般来说

```
T(n)=O(f(n))
O(1)<O(log2n)<O(n)<O(n log2 n)<O(n^2)<O(n^3)<O(2^n)
```

所以，要选择时间复杂度量级低的算法。

至于本题，从代码可知，x=x+1;是循环最内侧代码，其时间复杂度最高，所以只求这句代码的复杂度即可。从内到外看，k 循环从 1==2^0 开始每次变成原来的 2 倍，一直到大于 n-1，所以，循环体运行次数是llog(n)l，时间复杂度为 O(log(n))（计算机中 log 默认底数是 2）；j 循环从 1 开始每次递增 n/2，一直到 n-1，第一次递增之后，j 变成(n+2)/2,第二次递增 j，则是 n+1。所以应该是循环了 2 次，但是时间复杂度还是 O(1)，因为常数次数的时间复杂度都是 O(1)的，i 循环从 1 开始，每次增 1，一直到 n-1，所以循环体运行 n-1 次，时间复杂度 O(n)。最后相乘得到总的时间复杂度就是 O(n*1*log(n))=O(n*log(n))。这里要强调一下，时间复杂度都不带常数项或者常数系数的，所以不存在所谓 O(2n)这样的时间复杂度。

答案：D

面试例题 4：如何利用筛选法查找 100 以内的素数？[英国著名图形软件公司面试题]

解析：首先定义 a[i]=1，初始化整个数组，全部初始化为 1，第二步双重循环，从 2 开始，所有 2 的倍数都标记为 0，所有 3 的倍数也标记为 0；然后是 4，但因为 4 已经被标记为零了，跳过；接着是 5，直到所有的数都循环过一遍。

答案：代码如下。

```
public class Test{
public static void main(String argv[]){
{
  int a[101],i,j;
  for (i=1;i<101;i++)
      a[i]=1;
```

/*根据筛选法求出 100 以内的所有素数，所谓筛选法，是指从小到大筛去一个已知素数的所有倍数，例如，根据 2，我们可筛去 4,6,8,···,98,100 等数，然后根据 3 可筛去 9,15,···,99 等数(注意此时 6、12 等数早就被筛去了)，由于 4 被筛去了，下一个用于筛选的素数是 5，以此类推，最后剩余的就是 100 以内的素数*/

```
for ( i=2;i<101;i++)
    {
      if (a[i]!=0)
      for (j=i+i;j<101;)
         {

            if (j%i==0)
               a[j]=0;
               j=j+i;
         }
    }

  for(i=2;i<101;i++)
     if (a[i]!=0)
       System.out.println(i);

}
```

扩展知识（变量的内存分配情况）

求素数的方法不止一种，下面是用开根号的办法求值，也可以达到目的，代码如下。

```
public class Test{
public static void main(String argv[])
{
  int a[101],i,j,k;
  for(i=1;i<100;i++)
    {
       k=(int)sqrt(i);

       for(j=2;j<=k;j++)
       {
         if(i%j==0)
           break;
       }
       if(j>k)
         System.out.println("   "+i);

    }

}
}
```

7.3　概率

面试例题 1：一个鱼塘养鱼若干，请用一个办法尽量准确地估算其中有多少条鱼?[中国台湾著名 CPU 硬件公司 2009 年 11 月笔试题]

解析：本题考的是概率论中的放回抽样。

以从一个口袋中取球为例，每次随机地取一个，每次取一个球后放回袋中，搅匀后再取一球，这种取球方式为放回取样。每次取一个球后不放回袋中，下一次从剩余的球中再取一球，这种取球方式为不放回取样。

答案：先从鱼塘里捞 100 条鱼（可以根据鱼塘中鱼的大概数量级适当变化），然后为每条鱼做一个标记（例如绑上一条丝带），再放回去。过一天时间以后，再捞上 100 条，数一数有多少是做了标记的。可以假设这时做了标记的鱼是已经均匀分布在鱼塘里的，据此，大概就能估计出一共有多少条鱼了。

比如，发现 20 条鱼有标记，那么总鱼数大概为 100*100/20=500 条。

面试例题 2：概率问题：[中国大陆著名网络公司 B2009 年 11 月笔试题]

（1）一个普通的骰子连抛 2 次都是 1 点。问：抛第 3 次是 1 点的概率是小于 1/6、大于 1/6，还是等于 1/6?

（2）一个普通的骰子连抛 10 次都是 1 点。问：抛第 11 次是 1 点的概率是小于 1/6、大于 1/6，还是等于 1/6?

解析：在概率论中，有一个概念叫独立事件。事件 A(或 B)是否发生对事件 B(或 A)发生的概率没有影响，则称 A 与 B 是相互独立事件。贝努里(瑞士数学家和物理学家)有一个独立重复试验：同样的条件下重复地、各次之间相互独立地进行的一种试验。贝努里实验有个结论，n 次独立重复试验里，某事件 A 恰好发生 k(k=0，1，…，n)次的概率为 Pn(k)，组成离散型随机变量的二项分布：

Pn(k) = C(n,k)p^k(1-p)^(n-k)

掷骰子符合独立事件。投掷 1 次为 1 点的概率 p=1/6，不是 1 点的概率为 5/6。

如果本问题换一种问法：

（1）一个普通的骰子连抛 3 次都是 1 点的概率是多少？（此时 n=k=3，p=1/6，Pn(k)=1/216）

（2）一个普通的骰子连抛 11 次都是 1 点的概率是多少？（此时 n=k=11，p=1/6，Pn(k)=1/6^11）

对比本题：

（1）一个普通的骰子连抛 2 次都是 1 点。问：抛第 3 次是 1 点的概率是小于 1/6、大于 1/6，还是等于 1/6?

（2）一个普通的骰子连抛 10 次都是 1 点。问：抛第 11 次是 1 点的概率是小于 1/6、大于 1/6，还是等于 1/6?

这个问法不一样了，对于第（1）题，它假设了前 2 次都出现了 1 点，但第 3 次仍为独立事件，与前 2 次无关，第 3 次出现的概率仍然为 1/6。第（2）题也一样是 1/6。

从概率论来说，都等于 1/6，因为事件都是独立的。但从人的感觉来说，既然出了 10 次 1 点，再出 1 点的可能性较小。这种错误的感觉迷惑很多人。我们往往会被实际经验左右自己的结论，但我们要坚信理论。有人会问：本题中，连抛 10 次都是 1 点这种事情太怪了，简直不可能！概率论里，不可能事件的发生概率是 0，但 0 概率事件可能发生，比如在宇宙中抽一个人，抽到你的概率。这就是一个 0 概率事件可能发生的例子。

随机变量分连续和离散两种，它们各自的分布描述是不同的。对于连续性随机变量，单个具体点的概率密度值为一个有界常数，这个值可以是任意的（包括 0 和 1），但因为点是没有长度的，所以该点的概率密度积分为 0（因为该点概率密度值有界），即该点所对应的事件发生的概率为 0，但这个事件仍然是可能发生的，因为这个事件在事件域内。也就是说，概率为 0 的事件有可能发生。同理，某个点的概率密度值为 1，但该点的概率密度积分仍为 0，所以概率为 1 的事件也不一定必然发生。总之，对于连续性随机变量，讨论单个点的概率是没有意义的（都为 0），我们讨论的是这个随机变量落在一个区间内的概率。

对于离散随机变量，如果它的事件域是有限个事件，则可以认为概率为 0 的事件一定不会发生，概率为 1 的事件必然发生。但若事件是无限的，则还要具体分析，既然 0 概率事件都是有可能发生的，那么概率趋近于零的事件果然有可能发生，只不过我们平时在处理问题的时候，把概率趋近于零的事件作为 0 概率事件，但这也不是绝对的。

本题有三个前提：

（1）既然题目说是“一个普通的骰子”，按照常理就是有 6 种数字的骰子。

（2）普通骰子投一次投出 1 的概率是 1/6。

（3）概率本身就是一个估计值，零概率事件都可能发生，更不要说小概率事件了。

最后的结论是：投 10 次连续出现 10 次 1 的概率是 1/6 的 10 次方，但这个小概率事件的发生和下一次投没有关系。如果承认“普通骰子投一次投出 1 的概率是 1/6”，那么下次出现 1 的概率还是 1/6。

答案：1/6，1/6。

第8章

Java 内存管理

内存管理太重要了，花多少口舌介绍它都不过分。笔者曾经见到这样一句话："C++程序员觉得内存管理太重要了，所以一定要自己进行管理；Java/C#程序员觉得内存管理太重要了，所以一定不能自己去管理。"从某种意义上说，两者都是对的。面试中内存管理涉及堆、栈、哈希表、内存泄漏等诸方面。

8.1 垃圾收集

面试例题 1：Java 中的垃圾收集器相对于以前的语言的优势是什么？[SAP 公司 2005 年 10 月面试题]

答案：过去的语言（如 C 语言）要求程序员显式地分配内存、释放内存。程序在需要时分配内存，不需要时释放内存。但是这种做法常常引起"内存泄漏"，即由于某种原因使分配的内存始终没有得到释放。如果该任务不断地重复，程序最终会耗尽内存并异常终止，至少无法继续运行。相比之下，Java 不要求程序员显式地分配内存和释放内存，避免了很多潜在问题。Java 在创建对象时会自动分配内存，并当该对象的引用不存在时释放这块内存。

Java 中使用被称为垃圾收集器的技术来监视 Java 程序的运行，当对象不再使用时，就自动释放对象所使用的内存。Java 使用一系列软指针来跟踪对象的各个引用，并用一个对象表将这些软指针映射为对象的引用。之所以称为软指针，是因为这些指针并不直接指向对象，而是指向对象的引用。使用软指针，Java 的垃圾收集器能够以单独的线程在后台运行，并依次检查每个对象。通过更改对象表项，垃圾收集器可以标记对象、移除对象、移动对象或检查对象。

垃圾收集器是自动运行的，一般情况下，无须显式地请求垃圾收集器。程序运行时，垃圾收集器会不时检查对象的各个引用，并回收无引用对象所占用的内存。调用 System 类中的静态 gc()方法可以运行垃圾收集器，但这样并不能保证立即回收指定对象。

面试例题 2：下面说法中哪项是正确的？（多选）

A．Java 虚拟机中的自动垃圾回收机阻止程序运行溢出内存

B．一段程序可以建议垃圾回收执行，但是不能强迫它执行

C．垃圾回收是一个独立的平台

D．当一个对象的所有引用都被置为空时，这个对象就可以变为能被垃圾回收

解析：如果一个程序保持创建一个引用对象时，其他的任何引用没有被抛弃，也就会造成内存耗尽的结果。垃圾回收并不是一个独立的平台，它具有平台依赖。

Java 垃圾回收机制：gc 即垃圾收集机制，是指 JVM 用于释放那些不再使用的对象所占用的内存。Java 语言并不要求 JVM 有 gc，也没有规定 gc 如何工作。不过常用的 JVM 都有 gc，而且大多数 gc 都使用类似的算法管理内存和执行收集操作。Java 的垃圾回收机制是为所有的 Java 应用进程服务的，而不是为某个特定的进程服务的。因此，任何一个进程都不能命令垃圾回收机制做什么、怎么做或做多少。在 JVM 垃圾收集器收集一个对象之前，一般要求程序调用适当的方法释放资源，但在没有明确释放资源的情况下，Java 提供了默认机制终止化该对象来释放资源，这个方法就是 finalize()。它的原型为：protected void finalize() throws Throwable。在 finalize()方法返回之后，对象消失，垃圾收集开始执行。原型中的 throws Throwable 表示它可以抛出任何类型的异常。

答案：B，D

面试例题 3：下列代码中，第几行的 sobj 符合垃圾收集器的收集标准？[SAP 公司 2005 年 10 月面试题]

```
1. Object sobj = new Object ( );
2. Object sobj = null ;
3. Object sobj = new Object ( );
4. sobj = new Object ( );
```

A．1　　B．2　　C．3　　D．4

解析：第 1 行和第 3 行。因为第 2 行将 sobj 赋值为 null，所以第 1 行的 sobj 符合垃圾收集器的收集标准。而第 4 行相当于将 sobj 赋值为 null，所以第 3 行的 sobj 也符合垃圾收集器的收集标准。

如果有一个对象的句柄 a，且你把 a 作为某个构造器的参数，即 new Constructor (a)，则即使你将 a 赋值为 null，a 也不符合垃圾收集器的收集标准。直到由上面构造器构造的新对象被赋空值时，a 才可以被垃圾收集器收集。

答案：A，C

面试例题 4：下列代码中，第几行的 obj 符合垃圾收集器的收集标准？[SAP 公司 2005 年 10 月面试题]

```
1. Object aobj = new Object ( );
2. Object bobj = new Object ( );
3. Object cobj = new Object ( );
4. aobj = bobj;
5. aobj = cobj;
6. cobj = null;
7. aobj = null;
```

A．1　　B．2　　C．3　　D．4

E．5　　F．6　　G．7

解析：第 7 行。注意，这类题型是可能遇到的最难题型了。

行 1～3 分别创建了 Object 类的 3 个对象：aobj、bobj、cobj。

行 4：此时对象 aobj 的句柄指向 bobj，所以该行的执行不能使 aobj 符合垃圾收集器的收集标准。

行 5：此时对象 aobj 的句柄指向 cobj，所以该行的执行不能使 aobj 符合垃圾收集器的收集标准。

行 6：此时仍没有任何一个对象符合垃圾收集器的收集标准。

行 7：对象 cobj 符合了垃圾收集器的收集标准，因为 cobj 的句柄指向单一的地址空间。在第 6 行的时候，cobj 已经被赋值为 null，但由于 cobj 同时还指向了 aobj（第 5 行），所以此时 cobj 并不符合垃圾收集器的收集标准。而在第 7 行，aobj 所指向的地址空间也被赋予了空值 null，这就说明了由 cobj 所指向的地址空间已经被完全赋予了空值。所以，此时 cobj 最终符合了垃圾收集器的收集标准。但对于 aobj 和 bobj，仍然无法判断其是否符合收集标准。

总之，在 Java 语言中，判断一块内存空间是否符合垃圾收集器收集标准的标准只有以下两个：

（1）给对象赋予了空值 null，以后再没有调用过。

（2）给对象赋予了新值，即重新分配了内存空间。

再次提醒一下，一块内存空间符合了垃圾收集器的收集标准，并不意味着这块内

存空间就一定会被垃圾收集器收集。

答案：G

扩展知识（变量的内存分配情况）

我们在使用垃圾回收时需要注意以下几点，或许可以作为写程序时的准则。

（1）不要试图去假定垃圾收集发生的时间，这一切都是未知的。比如，方法中的一个临时对象在方法调用完毕后就变成了无用对象，这个时候它的内存就可以被释放。

（2）Java 中提供了一些和垃圾收集打交道的类，而且提供了一种强行执行垃圾收集的方法——调用 System.gc()，但这同样是一个不确定的方法。Java 中并不保证每次调用该方法就一定能够启动垃圾收集，它只不过会向 JVM 发出这样一个申请，到底是否真正执行垃圾收集，一切都是个未知数。

（3）挑选适合自己的垃圾收集器。一般来说，如果系统没有特殊和苛刻的性能要求，可以采用 JVM 的默认选项。否则可以考虑使用有针对性的垃圾收集器，比如增量收集器就比较适用于实时性要求较高的系统中。若系统具有较高的配置，有比较多的闲置资源，可以考虑使用并行标记/清除收集器。

（4）关键的也是难把握的问题是内存泄漏。良好的编程习惯和严谨的编程态度永远是最重要的，不要让自己的一个小错误导致内存出现大漏洞。

（5）尽早释放无用对象的引用。大多数程序员在使用临时变量的时候，都是让引用变量在退出活动域（scope）后，自动设置为 null，暗示垃圾收集器来收集该对象，还必须注意该引用的对象是否被监听，如果是，则要去掉监听器，然后再赋空值。就是说，对于频繁申请内存和释放内存的操作，还是自己控制一下比较好，但是 System.gc()的方法不一定适用，最好使用 finalize 强制执行或者写自己的 finalize 方法。

面试例题 5：下面说法中哪项是正确的？

A．Java 虚拟机中的自动垃圾回收机阻止程序运行溢出内存

B．一段程序可以建议垃圾回收执行，但是不能强迫它执行

C．垃圾回收是一个独立的平台

D．当一个对象的所有引用都被置为空时，这个对象就可以变为能被垃圾回收

解析：如果一个程序保持创建一个引用对象时其他的任何引用没有被抛弃，也就会造成内存耗尽的结果。垃圾回收并不是一个独立的平台，它具有平台依赖。

Java 垃圾回收机制：GC 即垃圾收集机制，是指 JVM 用于释放那些不再使用的对象所占用的内存。Java 语言并不要求 JVM 有 gc，也没有规定 gc 如何工作。不过常用的 JVM 都有 gc，而且大多数 gc 都使用类似的算法管理内存和执行收集操作。Java 的垃圾回收机制是为所有的 Java 应用进程服务的，而不是为某个特定的进程服务的。因此，任何一个进程都不能命令垃圾回收机制做什么、怎么做或做多少。在 JVM 垃圾收集器收集一个对象之前，一般要求程序调用适当的方法释放资源，但在没有明确释放资源的情况下，Java 提供了默认机制终止该对象来释放资源，这个方法就是 finalize()。它的原型为：protected void finalize() throws Throwable。在 finalize()方法返回之后，对象消失，垃圾收集开始执行。原型中的 throws Throwable 表示它可以抛出任何类型的异常。

答案：B，D

8.2　内存管理

面试例题 1：Java 是如何管理内存的?

答案：Java 的内存管理就是对象的分配和释放问题。在 Java 中，程序员需要通过关键字 new 为每个对象申请内存空间（基本类型除外），所有的对象都在堆 （Heap）中分配空间。另外，对象的释放是由 GC 决定和执行的。在 Java 中，内存的分配是由程序完成的，而内存的释放是由 GC 完成的，这种收支两条线的方法确实简化了程序员的工作。但同时，它也加重了 JVM 的工作。这也是 Java 程序运行速度较慢的原因之一。因为 GC 为了能够正确释放对象，必须监控每一个对象的运行状态，包括对象的申请、引用、被引用、赋值等，GC 都需要进行监控。

监视对象状态是为了更加准确、及时地释放对象，而释放对象的根本原则就是该对象不再被引用。

为了更好地理解 GC 的工作原理，我们可以将对象考虑为有向图的顶点，将引用关系考虑为图的有向边，有向边从引用者指向被引用对象。另外，每个线程对象可以作为一个图的起始顶点，例如，大多程序从 main 进程开始执行，那么该图就是以 main 进程顶点开始的一棵根树。在这个有向图中，根顶点可达的对象都是有效对象，GC 将不回收这些对象。如果某个对象（连通子图）与这个根顶点不可达（注意，该图为有向图），那么我们认为这个（这些）对象不再被引用，可以被 GC 回收。

用有向图表示内存管理。对于程序的每一个时刻，都有一个有向图表示 JVM 的内存分配情况。图 8-1 就是左边程序运行到第 6 行的示意图。

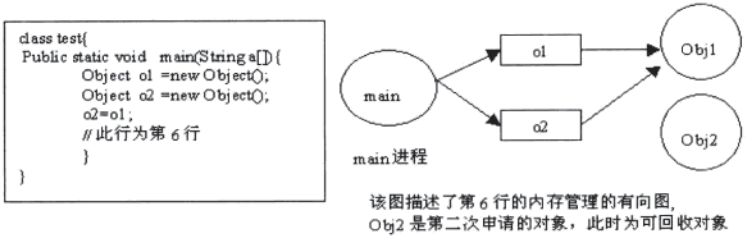


图 8-1　有向图表示内存管理

面试例题 2：什么是 Java 中的内存泄漏？

答案：在 Java 中，内存泄漏就是存在一些被分配的对象，这些对象有下面两个特点：① 对象是可达的，即在有向图中，存在通路可以与其相连；② 对象是无用的，即程序以后不会再使用这些对象。如果对象满足这两个条件，这些对象就可以判定为 Java 中的内存泄漏，这些对象不会被 GC 所回收，然而它们却占用内存。

在 C++中，内存泄漏的范围更大一些。有些对象被分配了内存空间，然后却不可达，由于 C++中没有 GC，这些内存将永远收不回来。在 Java 中，这些不可达的对象都由 GC 负责回收，因此，程序员不需要考虑这部分内存泄漏。

对于 C++，程序员需要自己管理边和顶点，而对于 Java 程序员，只需要管理边就可以了（不需要管理顶点的释放）。通过这种方式，Java 提高了编程的效率。

对于程序员来说，GC 基本是透明的、不可见的。虽然我们只有几个函数可以访问 GC，例如，运行 GC 的函数 System.gc()，但是根据 Java 语言规范定义，该函数不保证 JVM 的垃圾收集器一定会执行。因为不同的 JVM 实现者可能使用不同的算法管理 GC。通常，GC 线程的优先级别较低。JVM 调用 GC 的策略也有很多种，有的是内存使用到达一定程度时，GC 才开始工作，也有定时执行的，有的是平缓执行 GC，有的是中断式执行 GC。但通常来说，我们不需要关心这些，除非在一些特定的场合，GC 的执行影响应用程序的性能，例如，对于基于 Web 的实时系统，如网络游戏等，用户不希望 GC 突然中断应用程序的执行而进行垃圾回收，那么需要调整 GC 的参数，让 GC 能够通过平缓的方式释放内存，例如，将垃圾回收分解为一系列的小步骤执行。Sun 提供的 HotSpot JVM 就支持这一特性。

扩展知识

下面给出了一个简单的内存泄漏的例子。在这个例子中，循环申请 Object 对象，并将所申请的对象放入一个 Vector 中。如果仅仅释放引用本身，那么 Vector

仍然引用该对象，所以这个对象对 GC 来说是不可回收的。因此，如果对象加入到 Vector 后，还必须从 Vector 中删除，最简单的方法就是将 Vector 对象设置为 null。

```
Vector v=new Vector(10);
for (int i=1;i<100; i++)
{
    Object o=new Object();
    v.add(o);
    o=null;
}
//此时，所有的 Object 对象都没有被释放，因为变量 v 引用这些对象
```

面试例题 3：内存泄漏主要由什么引起？都有什么样的情况？

解析：忘记“释放”先前分配的内存，就可能造成内存泄漏。如果程序保留对永远不再使用的对象的引用，这些对象将会占用并耗尽内存，这是因为自动化的垃圾收集器无法证明这些对象将不再使用。如果存在一个对对象的引用，对象就被定义为活动的，因此不能删除。为了确保能回收对象占用的内存，编程人员必须确保该对象不能到达。这通常是通过将对象字段设置为 null 或者从集合（collection）中移除对象而完成的。注意，当局部变量不再使用时，没有必要将其显式地设置为 null，对这些变量的引用将随着方法的退出而自动清除。

概括地说，这就是内存托管语言中的内存泄漏产生的主要原因：保留下来却永远不再使用的对象引用。

答案：

典型的内存泄漏及其原因如下。

1．全局集合

在大的应用程序中有某种全局的数据储存库是很常见的，例如，一个 JNDI 树或一个会话表。在这些情况下，必须注意管理储存库的大小。必须有某种机制使得从储存库中移除不再需要的数据。

这可能有多种方法，但是最常见的一种是周期性运行的某种清除任务，该任务将验证储存库中的数据，并移除任何不再需要的数据。

另一种管理储存库的方法是使用反向链接（referrer）计数，然后集合负责统计集合中每个入口的反向链接的数目。这要求反向链接告诉集合何时会退出入口。当反向链接数目为零时，该元素就可以从集合中移除了。

2．缓存

缓存是一种数据结构，用于快速查找已经执行的操作结果。因此，如果一个操作执行起来很慢，对于常用的输入数据就可以将操作的结果缓存，并在下次调用该操作时使用缓存的数据。

缓存通常都是以动态方式实现的，其中新的结果是在执行时添加到缓存中的。典型的算法如下：

（1）检查结果是否在缓存中，如果在，就返回结果。

（2）如果结果不在缓存中，就进行计算。

（3）将计算出来的结果添加到缓存中，以便以后对该操作的调用可以使用。

该算法的问题（或者说是潜在的内存泄漏）出在最后一步。如果调用该操作时有相当多的不同的输入，就将有相当多的结果存储在缓存中。很明显，这不是正确的方法。

为了预防这种具有潜在破坏性的设计，程序必须确保对于缓存所使用的内存容量有一个上限。因此，更好的算法如下：

（1）检查结果是否在缓存中，如果在，就返回结果。

（2）如果结果不在缓存中，就进行计算。

（3）如果缓存所占的空间过大，就移除缓存最久的结果。

（4）将计算出来的结果添加到缓存中，以便以后对该操作的调用可以使用。

通过始终移除缓存最久的结果，我们实际上进行了这样的假设：在将来，比起缓存最久的数据，最近输入的数据更有可能用到。这通常是一个不错的假设。

新算法将确保缓存的容量处于预定义的内存范围之内。确切的范围可能很难计算，因为缓存中的对象在不断变化，而且它们的引用包罗万象。为缓存设置正确的大小是一项非常复杂的任务，需要将所使用的内存容量与检索数据的速度加以平衡。

解决这个问题的另一种方法是使用 java.lang.ref.SoftReference 类跟踪缓存中的对象。如果虚拟机的内存用尽而需要更多的堆，那么这种方法的保证这些引用能够被移除。

3．ClassLoader

Java 中 ClassLoader 结构的使用为内存泄漏提供了许多可乘之机。正是该结构本身的复杂性使 ClassLoader 在内存泄漏方面存在如此多的问题。ClassLoader 的特别之处在于它不仅涉及“常规”的对象引用，还涉及元对象引用，比如字段、方法和类。这意味着只要有对字段、方法、类或 ClassLoader 对象的引用，ClassLoader 就会驻留在 JVM 中，因为 ClassLoader 本身可以关联许多类及其静态字段，所以就有许多内存被泄漏。

下面谈一谈如何确定泄漏的位置。

发生内存泄漏的第一个迹象通常是：在应用程序中出现了 OutOfMemoryError。这通常发生在你最不愿意它发生的生产环境中，此时几乎不能进行调试。有可能是因为测试环境运行应用程序的方式与生产系统不完全相同，因而导致泄漏只出现在生产中。在这种情况下，需要使用一些开销较低的工具来监控和查找内存泄漏，还需要能够无须重启系统或修改代码就可以将这些工具连接到正在运行的系统上。可能最重要的是，当进行分析时，需要能够断开工具而保持系统不受干扰。

虽然 OutOfMemoryError 通常都是内存泄漏的信号，但是也有可能因为应用程序确实正在使用这么多的内存。对于后者，或者必须增加 JVM 可用的堆的数量，或者对应用程序进行某种更改，使它使用较少的内存。但是，在许多情况下，OutOfMemoryError 都是内存泄漏的信号。一种查明方法是不间断地监控 GC 的活动，确定内存使用量是否随着时间增加，如果确实如此，就可能发生了内存泄漏。

8.3　clone

面试例题 1：Object 类中有 clone 方法，但是 Object 又没有实现 cloneable 接口，这是为什么？对于一个没有实现 cloneable 的类来说，还可以用从 Object 类继承而来的 clone 方法实现一些基本的值的复制操作，那是不是可以说 clone 方法并没有对对象是否属于 cloneable 类型进行检验？

解析：

Object 类的 clone 是 protected 的，不能直接调用，可以被子类调用。Object 类的 clone 会知道对象大小，为它分配足够的内存空间，并将旧对象的内容复制到新的对象中。但是，Object.clone()执行其动作之前必须先检查 class 是否实现了 Cloneable 接口。

Cloneable 接口是一个标记接口，也就是没有任何内容，定义如下。

```
package java.lang;
public interface Cloneable{
}
```

这里分析一下这个接口的用法。

Java 中 clone 的含义是：假设 x 是一个非空对象，则

- x.clone()!=x 为 true，就是说它们不是同一个对象。
- x.clone().getClass()==x.getClass() 为 true，说明它们是同一个类型 Class。
- x.equals(x.clone()) 为 true，说明逻辑上应该相当。

clone 方法是在 Object 中定义的，而且是 protected 型的，只有实现了这个接口，才可以在该类的实例上调用 clone 方法，否则会抛出 CloneNotSupportedException。

Object 中默认的实现是一个浅复制，也就是表面复制，如果需要实现深层次复制，必须对类中可变域生成新的实例。

```
public class Unsupported{
   public Object clone(){
  Object obj;
       try {
       obj=super.clone();
  }
     catch (CloneNotSupportedException ex) {
       ex.printStackTrace(); //Exception was thrown
     }
  return obj;//如果不实现 implements Cloneable 接口，返回的是 null
   }
}
```

加上 implements Cloneable 就可以了。当然也可以不实现这个接口，但是覆盖 Clone 方法，其代码如下：

```
public class Unnormal{
   public Object clone(){
  return new Unnormal();
   }
}
```

这样肯定是没有问题的，不过已经和 Java 中的 clone 机制没有关系了。

下面举一个例子说明浅复制和深复制。

```
public class ShallowCopy implements Cloneable{
    private Date begin;
    public Date getBegin(){return this.begin;}
    public void setBegin(Date d){this.begin=d;}
    public Object clone(){
 Object obj=null;
       try
        {
         obj=super.clone();
         }
       catch (CloneNotSupportedException ex) {
       ex.printStackTrace();
       }
  return obj;
```

```
    }
}
public class DeepCopy implements Cloneable{
 private Date begin;
    public Date getBegin(){return this.begin;}
    public void setBegin(Date d){this.begin=d;}
    public Object clone(){
 DeepCopy obj=null;
 try
 {
  obj=(DeepCopy)super.clone();
  }
       catch (CloneNotSupportedException ex) {
       ex.printStackTrace();
    }
 obj.setBegin((Date)this.getBegin().clone());
 return obj;
   }
}
```

答案：clone 方法是在 Object 中定义的，而且是 protected 型的，只有实现了这个接口，才可以在该类的实例上调用 clone 方法，否则会抛出 CloneNotSupportException。说 clone 方法并没有对对象是否属于 cloneable 类型进行检验这个观点是不正确的。因为 cloneable 接口的出现跟接口的正常使用没有任何关系，特别是它并不指定 clone 方法——该方法从 Object 类中继承而来，该接口只是作为一个标记。这句话通俗地说就是以下 3 点：首先，clone 方法是 object 类的一个方法，所以任何一个类都会自动拥有这一个方法。其次，这并不说明该类就可以调用 clone 了，因为 Javac 或者 Java 需要程序员显式地指明该类可以调用 clone，方法就是写上这个字符串“implements Cloneable”。再次，这个字符串只是起一个指示作用，没有其他功能，这是由 Javac 或者 Java 规定的。

第9章

面向对象

面向对象其实是现实世界模型的自然延伸。现实世界中任何实体都可以看做是对象。对象之间通过消息相互作用。另外，现实世界中任何实体都可归属于某类事物，任何对象都是某一类事物的实例。如果说传统的过程式编程语言是以过程为中心、以算法为驱动的话，面向对象的编程语言则是以对象为中心，以消息为驱动。用公式表示，过程式编程语言为：程序=算法+数据；面向对象编程语言为：程序=对象+消息。

所有面向对象的编程语言都支持 3 个概念，即封装、多态性和继承，Java 也不例外。现实世界中的对象均有属性和行为，映射到计算机程序上，属性则表示对象的数据，行为表示对象的方法（其作用是处理数据或同外界交互）。所谓封装，就是用一个自主式框架把对象的数据和方法连在一起形成一个整体。可以说，对象是支持封装的手段，是封装的基本单位。Java 语言的封装性较强，因为 Java 无全程变量，也无主函数，在 Java 中绝大部分成员是对象，只有简单的数字类型、字符类型和布尔类型除外，而对于这些类型，Java 也提供了相应的对象类型以便与其他对象交互操作。

“这个世界是由什么组成的？”这个问题如果让不同的人来回答，会得到不同的答案。如果回答该问的人是一个化学家，他也许会告诉你：“还用问吗？这个世界是由分子、原子、离子等化学物质组成的”。如果他是一个画家，他也许会告诉你：“这个世界是由不同的颜色所组成的”。但如果让一个分类学家来考虑这个问题就有趣多了，他会告诉你：“这个世界是由不同类型的物与事所构成的”。好！作为面向对象的程序员，我们要从分类学家的角度去考虑问题“这个世界是由动物、植物等组成的。动物又分为单细胞动物、多细胞动物、哺乳动物等，哺乳动物又分为人、大象、老虎……就这样分下去了！”

现在从抽象的角度，我们给“类”下个定义吧！笔者的意思是，从抽象的角度，你回答我“什么是人类？”首先让我们来看看人类所具有的一些特征，这个特征包括

属性（一些参数、数值）及方法（一些行为，他能干什么）。每个人都有身高、体重、年龄、血型等一些属性。还有人会劳动、人都会直立行走、人都会用自己的头脑去创造工具等方法。人之所以能区别于其他类型的动物，是因为每个人都具有人这个群体的属性与方法。“人类”只是一个抽象的概念，它仅仅是一个概念，它是不存在的实体。但是所有具备“人类”这个群体的属性与方法的对象都叫人。这个对象“人”是实际存在的实体，每个人都是人这个群体的一个对象。老虎为什么不是人？因为它不具备人这个群体的属性与方法，老虎不会直立行走，不会使用工具等，所以说老虎不是人。

由此可见，类描述了一组有相同特性（属性）和相同行为（方法）的对象。在程序中，类实际上就是数据类型，例如，整数、小数等。整数也有一组特性和行为。面向过程的语言与面向对象的语言的区别就在于，面向过程的语言不允许程序员自己定义数据类型，而只能使用程序中内置的数据类型。而为了模拟真实世界，为了更好地解决问题，我们往往需要创建解决问题所必需的数据类型。

面向对象编程为我们提供了解决方案。以下的考题来自真实的笔试资料，希望读者先不要看答案，自我解答后再与答案加以比对，找出自己的不足。

9.1 面向对象的基本概念

面试例题 1：对象是类的一个实例，但是什么时候需要实例，什么时候不需要，该如何确认？有些类无须创建实例，直接定义就可以使用，系统也不报错。如何知道这个类必须用实例还是不需要用实例？[中国大陆某软件公司 Y2009 年 10 月面试题]

解析：对象是一个抽象的多变的概念。初级阶段仅理解为类的实例是可以的，到了后期，它就是一切你着眼的东西。

实例的作用是保存和传递不同的状态，当你需要这样的时候就用实例。例：A a1 = new A（）；a1.s1 = "a1";

A a2 = new A（）；a2.s1 = "a2";现在，a1 和 a2 保存着不同的状态。

如果你想使用 s1 和 doSome()，那么就要用 new 调用实例；

如果你想使用 s2 和 doMore()，那么直接在类型上调用。

即要使用实例的状态或方法时就用实例，要使用类型的状态或方法时就不用实例，直接使用类型。

当然你可以用 a1.doMore()，它看上去是在实例上调用了静态方法，其实质和 A.doMore()是一样的。

而在更宽广的条件下（比如，机器学习中的一些概念），那就不一定了。实例这个

词表达了一种特化的过程，而对象只是简单的是这种特化的产物。比如，这样的一个概念特化过程：Thing→Animal→Human→Chinese→Chen Ju，我们可以看到 Animal 是 Thing 的一个实例，它是一个对象。同时 Human 是 Animal 的一个实例，它也是一个对象。也就是说，一个对象可以是另外一个对象的实例。

答案：对象和实例从宏观的角度看，区别是：对象是同类事物的一种抽象表现形式，而实例是对象的具体化，一个对象可以实例化很多实例，对象就是一个模型，实例是照着这个模型生产的最终产品。实际上就是这样，一个对象可以实例化 *N* 个实例。就像根据一个模型可以制造多个实际的产品一样。

不用实例化的：

静态方法是为类提供的公共方法，也就是这个方法对所有属于这个类的对象适用。例如，假设为男人定义一个类，该类有个方法用来查询男人是否有喉结，那么从现阶段的生物进化来看，男人都是有喉结的。也就是不必具体到某个男人，就知道他有喉结。所以这个方法应定义为静态方法，因为是对所有的男人都适用。

要实例化的：

每个人都有身高、体重，当你要查询人的体重时，你肯定先要指定那个人的体重。这个过程就是相当于实例化，即具体到了个人。

有些类却无须创建实例，直接定义就可以使用。这种类型往往是面向公众的，如政府机构提供了一系列的服务，这些服务对每个人都是适用的，所以这个可以是静态的。如何知道这个类必须用实例还是不需要用实例？这个和问题域有关，如前面所说，当你研究男人有没有喉结时，这个不用实例化就能查询到。当你要查询体重时，就要具体到某个人，就是要实例化。

面试例题 2：Which is incorrect about the class？（关于类的描述，下面哪个是错误的？）[中国大陆某软件公司 Y2009 年 10 月面试题]

A．A class is a blueprint to objects.

B．We use the keyword class to create a class construct.

C．Once a class is declared, the class name becomes a type name and can be used to declare variables.

D．The class is same as the struct, and there are no different between class and struct.

解析：这道题的考点是类的概念。

答案：D

面试例题 3：Which is incorrect about the OOP?（下面关于面向对象技术的叙述哪个是错误的?）[中国大陆某软件公司 Y2006，2009 年 10 月面试题]

A．The central idea of OOP is to build programs using software objects.

B．The OOP focuses mainly on the step-by-step procedure as procedure-oriented programing.

C．The OOP offers many advantages: simplicity,modularity,modifiability, extensibility, and so on.

D．The key concept of object orientation is the attachment of procedure to data.

解析：本题属于 OOP 的概念面试例题。面向对象和面向过程不能混为一谈。

答案：B，D

9.2　类和对象

面试例题 1：以下代码编译时会产生错误的是_____。[Trend 公司 2005 年面试题]

```
class reverseIt4
{

 public static void main(String[] args)
 {
      EnclosingClass jb2;                  //-----1
     System.out.println(jb2.m);            //-----2
 }


}

class EnclosingClass                       //--------3
{
 public int m = 6;
  class InnerClass                         //-------4
 {
      int msquare;
      InnerClass()
      {
          msquare = m*m;
      }
 }
}
```

A．语句 1　　B．语句 2　　C．语句 3　　D．语句 4

解析：语句 3 和语句 4 显然是正确的，尽管它们的描述不是很规范（存在一个类中的类）。语句 1 声明了一个类，但是没有定义，于是问题就出现了。声明好比只是告诉编译器有一个人，但是如果不定义，这个人就是个抽象的人，没有身高、体重、年龄、职业的“空”人。所以，定义对象必须在声明的同时给它定义。正确的程序如下：

```
class reverseIt4
{

 public static void main(String[] args)
 {
      EnclosingClass jb = new EnclosingClass();

     System.out.println(jb.m);

 }


}

class EnclosingClass
{
 public int m = 6;
  class InnerClass
 {
      int msquare;
      InnerClass()
      {
           msquare = m*m;
      }
 }
}
```

答案：该题是问编译在哪儿出现问题，尽管问题出在 1 处，但编译器不会发现，编译器只有在 2 处才会发现问题。所以答案选 B。

面试例题 2：下面哪个不是 Object 类所定义的 public method?

A．finalize()　　B．clone()　　C．wait()　　D．sleep()

解析：Object 基类的方法如下：

```
clone()方法：创建并返回此对象的一个副本。
equals(Object obj)方法：指示某个其他对象是否与此对象“相等”。
finalize()方法：当垃圾回收器确定不存在对该对象的更多引用时，由对象的垃圾回收器调用此方法。
```

getClass()方法：返回一个对象的运行时类。
hashCode()方法：返回该对象的哈希值。
notify()方法：唤醒在此对象监视器上等待的单个线程。
notifyAll()方法：唤醒在此对象监视器上等待的所有线程。
toString()方法：返回该对象的字符串表示。
wait()方法：导致当前的线程等待，直到其他线程调用此对象的 notify() 方法或 notifyAll() 方法。
wait(long timeout)方法：导致当前的线程等待，直到其他线程调用此对象的notify()方法或notifyAll()方法，或者超过指定的时间量。
wait(long timeout, int nanos)方法：导致当前的线程等待，直到其他线程调用此对象的 notify() 方法或 notifyAll() 方法，或者其他某个线程中断当前线程，或者已超过某个实际时间量。

答案：D

面试例题 3：The code output（下列代码输出结果是什么？）

```
import java.util.*;
public class Test {
private String value = null;

public Test(String v) {
      value = v;
 }
 public boolean equals(Test o) {
      if (o == this)
            return true;
      if (o instanceof Test) {
            Test test = (Test) o;
            return value.equals(test.value);
      }
      return false;
 }

 public static void main(String[] args) {
      List list = new ArrayList();
      Test test1 = new Test("object");
      Test test2 = new Test("object");
      Test test3 = new Test("object");
      Object test4 = new Test("object");
      list.add(test1);

      System.out.println(list.contains(test2));
      System.out.println(test2.equals(test3));
      System.out.println(test3.equals(test4));
 }
}
```

A．false　true　false　　B．true　true　true

C．false　false　false　　D．false　true　true

解析：第一个输出 false 是因为 list.add(test1)；在 list 放入的是 test1，也就是“object”的内存地址，而不是“object”本身。所以这样肯定是输出 false。

第二个输出 true，很明显，按照题目的 equals 的执行顺序，结果是 true。

第三个是假，是调用 Object 的 Equals 方法。

答案：A

9.3　嵌套类

面试例题 1：请说明 static nested class 和 inner class 的不同。

答案：

1．nested（**嵌套**）class（**一般是 C++的说法**）

nested class 是合成型聚集关系（Composite Aggregation）的另一种表达方式，也就是说，nested class 也可以用 Aggregation 表达出来。但是，nested class 更加精确地表达了一种专用的、紧耦合的关系，尤其是在代码生成时，nested class 在 Java 中映射成 inline class。比如，计算机专用开关电源类可以作为计算机类的 nested class，但是电池组类就不一定适合作为计算机类的 nested class，因为电池组类表述的是一个过于通用的对象，可能还被包含（Aggregation）于模型中的其他设备对象。class A nested in class B，则说明 A 是一个 nested class，一般 A 是用来完成 B 中的某种重要功能的。

2．inner class（**一般是 Java 的说法**）

Java 内部类与 C++嵌套类最大的不同就在于是否有指向外部的引用上。

静态内部类（inner class）意味着：创建一个 static 内部类的对象，不需要一个外部类对象；不能从一个 static 内部类的一个对象访问一个外部类对象。

面试例题 2：关于下面类的定义，以下哪种说法是正确的？[中国台湾 2005 年 11 月面试题]

```
public class Test
{
    static class one{
      private static class two{
            public static void main(String[] aaa){
            System.out.println("two");
```

```
            }
        }
    }
}
```

A．此代码不能编译成功，因为该类嵌套了不止一层

B．此代码不能编译通过，因为 class two 是私有的

C．此代码可以编译通过，并且在运行的时候输出字符串 two

D．以上答案都不正确

解析：本题可以编译通过，并且在运行的时候输出字符串 two。

内部类或嵌套类在类层级上没有限制，所以选项 A 是错误的。

内部类可以是私有类，所以选项 B 也是错误的。

答案：C

9.4　集合类

面试例题 1：You need to store elements in a collection that guarantees that no duplicates are stored and all elements can be access in nature order, which interface provides that capability?（你需要存储元素到某种数据结构中，而且要确保元素在自然队列中没有重复，下面哪个接口能保证这种能力？）[美国某移动通信企业 M2009 年 12 月笔试题]

A．java.uil.Map　　C．java.util.List

B．java.util.Collection　　D．ava.util.Set

解析：本题涉及一连串的 Java 容器类库概念。面试者需要对这些概念有一个清晰的脉络。

（1）Java 容器类库一共有两种主要类型：Collection 和 Map。

Collection 和 Map 的区别在于容器内每个“槽”所存储的元素个数不同。Collection 类型中，每个“槽”只有一个元素；Map 类型中，持有 key-value 关联（key 叫做键，value 叫做值）像一个小型数据库。所有的 Java 容器类都可以自动调整自己的尺寸。

（2）各自旗下的子类关系如图 9-1 所示。

Collection 的子类如下：

① --List：将以特定次序存储元素。所以取出来的顺序可能和放入顺序不同。
- --ArrayList：擅长随机访问元素，但在 List 中间插入、删除、移动元素较慢。
- --LinkedList：插入、删除、移动元素方便，随机访问元素差。

② --Set：每个值只能保存 1 个对象，不能含有重复的元素。
- --HashSet：使用散列函数。

- -- TreeSet：使用红黑树。
- -- LinkedHashSet：使用链表结合散列函数。

③ --Queue：先进先出的容器。

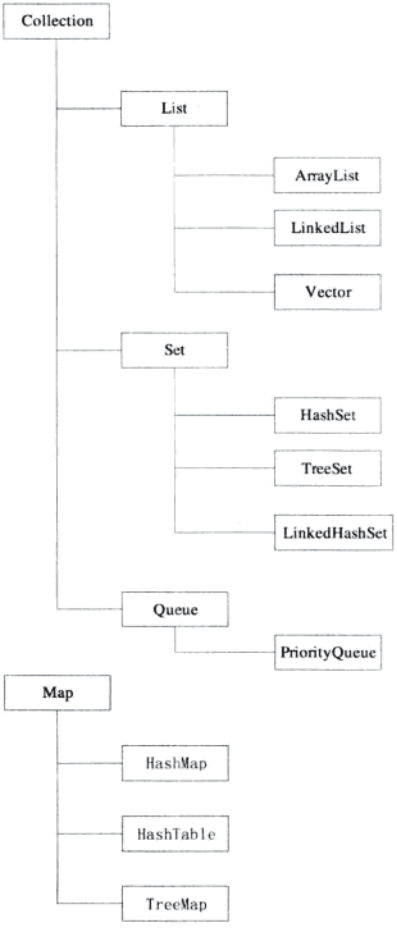


图 9-1　Collection 和 map 的子类

Map 的子类有以下 3 个。

- --HashMap。
- --HashTable。
- --TreeMap 。

（3）其他特征如下。

① List、Set、Map 将所有的对象一律视为 Object 类型。

② Collection、List、Set、Map 都是接口，不能实例化。继承自它们的 ArrayList、Vector、HashTable、HashMap、stack 是具体 class，这些才可被实例化。

③ Vector 容器确切地知道它所持有的对象隶属什么类别。Vector 不进行边界检查。

（4）关于 Collections。

Collections 是针对集合类的一个帮助类，它提供了一系列静态方法实现对各种集合的搜索、排序、线程完全化等操作。相当于对 Array 进行类似操作的类——Arrays。如：

```
Collections.max(Collection coll);          //取 coll 中最大的元素
Collections.sort(List list);               //对 list 中元素排序
```

（5）如何选择容器类。

① 容器类和 Array 的区别、择取。

容器类仅能持有对象引用（指向对象的指针），而不是将对象信息复制一份至数列某位置，一旦将对象置入容器内，便损失了该对象的类别信息。

② 在各种 Lists 中，最好的做法是以 ArrayList 作为默认选择。当插入、删除频繁时，使用 LinkedList()；Vector 总是比 ArrayList 慢；在各种 Sets 中，HashSet 通常优于 HashTree（插入、查找）。只有当需要产生一个经过排序的序列，才用 TreeSet。HashTree 的意义是用来维护其内元素的排序状态；在各种 Maps 中，HashMap 用于快速查找。当元素个数固定时，最好使用 Array，因为 Array 的效率是最高的。

答案：D

面试例题 2：下列代码的输出结果是什么？[中国某软件公司 LC2009 年 12 月笔试题]

```
public class Test {

 public static void main(String[] args) {
       Map<String, String> map = new HashMap<String, String>();
       map.put(String.valueOf(System.currentTimeMillis()) + "a", "1");
       map.put(String.valueOf(System.currentTimeMillis()) + "a", "2");
       map.put(String.valueOf(System.currentTimeMillis()) + "a", "3");
       for (Map.Entry<String, String> entry : map.entrySet()) {
            System.out.printf(entry.getValue());
       }
 }
}
```

A．123　　B．213　　C．123 顺序无法确定　　D．以上答案都不对

解析：本题中，程序期待的输出结果是 123 顺序无法确定。Map 中的键是 Set，值是 Collection，map.entrySet()产生的类型就是 Set，因而能用 foreach()，而 Set 顺序是随机的，因为输出的 1、2、3 的顺序是不确定的，但现在计算机的运行速度很快，因而极有可能 System.currentTimeMillis(毫秒级)相等，而 Set 是不允许有重复的，这样就会覆盖它的值，因而在机器上测试时只输出 3。

如果本题换一种写法：

```
import java.util.HashMap;
import java.util.Map;
public class Test {
 public static void main(String[] args) {
      Map<String, String> map = new HashMap<String, String>();
      map.put(String.valueOf(System.currentTimeMillis()) + "a", "1");
      map.put(String.valueOf(System.currentTimeMillis()) + "a", "2");
      map.put(String.valueOf(System.currentTimeMillis()) + "a", "3");
      for (Map.Entry<String, String> entry : map.entrySet()) {
           System.out.printf(entry.getValue());
      }
 }
}
```

java.lang.System.nanoTime(); 此方法来自 JDK 1.5 版本，用来测量代码段在同一线程上执行所消耗的时间，度量单位是“十亿分之一秒”（纳秒）。使用纳秒为单位获取时间，结果是 123 顺序无法确定，每次运行结果不同。

答案：C

面试例题 3：以下哪一个集合允许重复值？[中国某软件公司 LC2009 年 12 月笔试题]

A．HashMap　　B．ArrayList　　C．TreeMap　　D．HashSet

答案:B

面试例题 4：下列程序的输出结果是什么？[中国某著名软件培训公司 B2009 年 11 月笔试题]

```
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class Test {
    private static final String[] URLNAMES = {
    "http://www.sina.com.cn",                    //IP 地址为 202.108.33.94
    "http://www.nwu.edu.cn",                     //IP 地址为 124.115.173.252
    "http://javapuzzlers.com",                   //IP 地址为 208.97.154.9
```

```
        "http://www.google.com",                          //IP地址为64.233.189.147
        "http://Javapuzzlers.com",                        //IP地址为208.97.154.9
        "http://apache2-snort.skybar.dreamhost.com",      //IP地址为208.97.154.9

    };

    public static void main(String[] args) throws MalformedURLException {
        Set<URL> favorites = new HashSet<URL>();
        for (String urlName : URLNAMES)
            favorites.add(new URL(urlName));
        System.out.println(favorites.size());
    }
}
```

A．一定是4　　B．一定是5　　C．一定是6　　D．以上答案皆不对

解析：本题在联网的状态下会输出“4”，这是由于URL的equals比对方式。根据equals的文档说明：如果两个主机名可解析为同一IP地址，则认为两个主机相同(即使主机名不等)；如果有一个主机名无法解析，但两个主机名相等(不区分大小写)，或者两个主机名都为null，则也认为这两个主机相同。

也就是说，如果两个URL的IP地址是相同的，那么这两个URL就是相等的。根据题干：

```
http://javapuzzlers.com                          //IP地址为208.97.154.9
http://apache2-snort.skybar.dreamhost.com        //IP地址为208.97.154.9
http:// Javapuzzlers.com                         //IP地址为208.97.154.9
```

上面3个IP地址是相同的，都是208.97.154.9，所以在Set时都把它们当成同一个。答案为4。

如果在断网时，这些都是无法解析成为IP地址的，这时就要判断URL的名字，仅认为名字相同时才是相同的URL。“http:// Javapuzzlers.com”与“http://javapuzzlers.com”因为不区分大小写，所以默认两者相同。答案为5。

这道题告诉我们，不要把URL应用于Set和Map的key中，可以使用URI来代替，这样就不存在有无网络的问题了。

答案：D

面试例题5：在List.Map.Set等接口中，不能包含重复元素的接口是哪个？[德国某著名电子通讯公司E2009年9月笔试题]

A．List　　B．Map　　C．Set　　D．都不是

答案：C

面试例题 6：请说明 HashMap 和 Hashtable 的区别。

答案：它们都属于 Map 接口的类，实现了将唯一键映射到特定的值上。

HashMap 类没有分类或者排序。它允许一个 null 键和多个 null 值。

Hashtable 类似于 HashMap，但是不允许 null 键和 null 值。它也比 HashMap 慢，因为它是同步的。

Hashtable 继承自 Dictionary 类，而 HashMap 是 Java 1.2 引进的 Map interface 的一个实现。

HashMap 允许将 null 作为一个 entry 的 key 或者 value，而 Hashtable 不允许，还有就是，HashMap 把 Hashtable 的 contains 方法去掉了，改成 containsvalue(Returns true if this map maps one or more keys to the specified value（如果相对应的 map 对应指定的 value 有多个 key，则返回值为真）)和 containsKey(Returns true if this map contains a mapping for the specified key（如果对指定的 key，map 存在相应的映射则返回为真）)。因为 contains(Tests if some key maps into the specified value in this hashtable（ 检测哈希表中是否有 value 和 key 存在对应）)方法容易让人引起误解。

两者间最大的不同是，Hashtable 的方法是 Synchronize 的，而 HashMap 不是，在多个线程访问 Hashtable 时，不需要自己为它的方法实现同步，而 HashMap 就必须为之提供外同步。

Hashtable 和 HashMap 采用的 hash/rehash 算法类似，所以性能不会有很大的差异。

9.5 构造函数和析构函数

面试例题 1：下面哪个选项的说法是正确的？

A．构造函数不能被重载

B．构造函数不能被覆盖

C．一个构造函数可以返回一个私有的或一个对象的引用

D．构造函数代码执行时是从当前的类层级到它祖先的类

解析：重载构造函数是一个主要的技术，可以允许多种方式的初始化一个类。通过定义，构造函数是没有返回值的。所以选项 C 是错误的，这种说法并没有任何意义。选项 D 中构造函数代码的执行是从它最老的祖先类开始向下执行调用。可以写一个继承一个基类的类来测试，当创建一个子类的时候，会发现它的祖先类的构造函数先被调用。

答案：B

面试例题 2：下列有关派生类调用父类构造函数的描述中正确的是哪个？[美国某软件公司 SP2009 年 9 月笔试题]

A．派生类定义了自己的构造函数，就不会调用父类的构造函数

B．派生类必须通过 super 调用父类的含有参数的构造函数

C．派生类将会继承父类中所有的构造函数

D．创建派生类对象时，先调用派生类自己的构造函数，然后调用父类的构造函数

解析：派生类被构造时一定会先调用父类的构造函数，排除选项 A、D。你可以选择调用哪个构造函数，可以决定调用哪一个，但不能都不调用（至少必选一个），排除选项 C。若不指定，就会调用无参数的构造函数，以下代码调用的是无参构造函数，得到 A without any parameter B with a parameter 的结果，代码如下：

```
class A
{
   public A()
   {
   System.out.println("A without any parameter  ");
   }
   public A(int i)
   {
   System.out.println("A with a parameter");
   }
}
class B extends A
{
   public B()
   {
   System.out.println("B without any parameters.  ");
   }
   public B (int i)
   {
   System.out.println("B with a parameter");
   }
}
class Test
{
   public static void main(String args[]){
   B a =new B(100);
   }
}
```

如果选定调用 A(int i)，则会得到 A with a parameter B with a parameter 的结果，代码如下：

```
class A
{
   public A()
   {
   System.out.println("A without any parameter  ");
   }
   public A(int i)
   {
   System.out.println("A with a parameter");
   }
}
class B extends A
{
   public B()
   {
   System.out.println("B without any parameters.  ");
   }
   public B (int i)
   {
   super(i);
   System.out.println("B with a parameter");
   }
}
class Test
{
   public static void main(String args[]){
   B a =new B(100);
   }
}
```

答案：B

面试例题 3：以下程序输出结果是什么？[美国某软件公司 SP2009 年 9 月笔试题]

```
class X {
 Y b = new Y();
 X(){
       System.out.print("X");
 }
}

class Y {
 Y(){
```

```
        System.out.print("Y");
 }
}
public class Z extends X{
 Y y = new Y();
 Z(){
        System.out.print("Z");
 }
 public static void main(String[] args)
 {
       new Z();
 }
}
```

解析：派生类构造函数调用时必须先调用父类构造函数。程序解释如下：

```
class X {
 Y b = new Y();//步骤1：若Z继承X，构造Z，则先构造X，进入X类运行Y b = new Y(); 输出"Y"
 X(){
       System.out.print("X");//步骤2：接着输出"X"
 }
}
class Y {
 Y(){
       System.out.print("Y");
 }
}
public class Z extends X{
 Y y = new Y();//步骤3：再次构造Y，输出"Y"
 Z(){
       System.out.print("Z");//步骤4：最后调用Z本身的构造函数，输出"Z"
 }
 public static void main(String[] args)
 {
       new Z();
 }
}
```

答案：YXYZ

9.6 复制构造函数和赋值函数

面试例题1：Which of the following is true about "Copy Constructor"?（下面关于复制构

造函数的说法中哪一个是正确的？）[中国某杀毒软件公司 J2005,2009 年面试题]

A．They copy constructor into each other.（给每一个对象复制一个构造函数）

B．A default is provided, but simply does a member-wise copy.（有一个默认的副本构造函数，但简单地说这只是位复制）

C．They can't copy arrays into each other.（不能复制数组）

D．All of the above.（以上结果都正确）

解析：复制构造函数面试例题。

答案：B

面试例题 2：Which of the following class DOES NOT need a copy constructor?（下面所列举的类哪个不需要复制构造函数?）[中国台湾著名杀毒软件公司 2004，2009 年面试题]

A．A matrix class in which the actual matrix is allocated dynamically within the constructor and is deleted within its destructor.（一个矩阵类中，实际的矩阵利用构造函数动态分配，利用析构函数删除）

B．A payroll class in which each object is provided with a unique ID.（一个花名册类中每一个对象应唯一的 ID）

C．A word class containing a string object and vector object of line and column location pairs.（一个 word 类包含一个字符串对象和一个存在行、列位置对的 Vector 容器）

D．A library class containing a list of book object.（一个图书馆类由一系列书籍对象构成）

解析：按照题意，寻找一个不需要复制构造函数的类。

A 选项要定义复制构造函数。

在 B 选项中，若不自定义复制构造函数，势必造成两个对象的 ID 不唯一。至于说自定义了复制构造函数之后，如何保证新对象的 ID 唯一，那是面试例题所涉及的内容。方法有多种多样，比如可以使用当前的系统 tick 数作为新 ID。当然在语义上有损失，不是完全意义上的复制，但在这儿只能在保持语义和实现目的之间来一个折中。

选 C 的原因是其使用了默认的复制构造、string 子对象和 Vector 子对象，它们的类都是成熟的类，都有合适的赋值操作，复制构造函数避免了“浅复制”问题。

D 选项显然要定义复制构造函数。

答案：C

9.7 多态的概念

面试例题1：什么是多态?

答案：开门、开窗户、开电视。在这里的“开”就是多态。

多态性可以简单地概括为“一个接口，多种方法”。在程序运行的过程中才决定调用哪个函数。多态性是面向对象编程领域的核心概念。

多态（Polymorphisn）按字面的意思就是“多种形状”。多态性是允许你将父对象设置成为和它的一个或更多的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。简单地说，就是一句话：允许将子类类型的指针赋值给父类类型的指针。多态性在 Object Pascal 和 C++中都是通过虚函数（Virtual Function）实现的。

扩展知识（多态的作用）

虚函数就是允许被其子类重新定义的成员函数。而子类重新定义父类虚函数的做法称为“覆盖”（override），或者称为“重写”。这里有一个初学者经常混淆的一组概念：覆盖（override）和重载（overload）。覆盖是指子类重新定义父类的虚函数的做法。而重载是指允许存在多个同名函数，而这些函数的参数表不同（或许参数个数不同，或许参数类型不同，或许两者都不同）。其实，重载的概念并不属于“面向对象编程”，重载的实现是：编译器根据函数不同的参数表，对同名函数的名称做修饰，然后这些同名函数就成了不同的函数（至少对于编译器来说是这样的）。例如，有两个同名函数：function func(p:integer):integer 和 function func(p:string):integer，那么编译器做过修饰后的函数名称可能是这样的：int_func、str_func。对于这两个函数的调用，在编译器间就已经确定了，是静态的（记住：是静态）。也就是说，它们的地址在编译期就绑定了（早绑定），因此，重载和多态无关。真正和多态相关的是“覆盖”。当子类重新定义了父类的虚函数后，父类指针根据赋给它不同的子类指针，动态（记住：是动态）地调用属于子类的该函数，这样的函数调用在编译期间是无法确定的（调用的子类的虚函数的地址无法给出）。因此，这样的函数地址是在运行期绑定的（晚绑定）。结论就是：重载只是一种语言特性，与多态无关，与面向对象也无关。

引用一句 Bruce Eckel 的话：“不要犯傻，如果它不是晚绑定，它就不是多态。”

那么，多态的作用是什么呢？封装可以隐藏实现细节，使得代码模块化；继承可以扩展已存在的代码模块（类）。它们的目的都是为了代码重用。而多态则是为了实现另一个目的——接口重用！而且现实往往是：要有效重用代码很难，而真正最具有价值的重用是接口重用，因为“接口是公司最有价值的资源。设计接口比用一堆类来实现这个接口更费时间。而且接口需要耗费更昂贵的人力和时间。”其实，继承的为重用代码而存在的理由已经越来越薄弱，因为“组合”可以很好地取代继承的扩展现有代码的功能，而且“组合”的表现更好（至少可以防止“类爆炸”）。因此笔者个人认为，继承的存在在很大程度上是作为“多态”的基础而非扩展现有代码的方式。

那么什么是接口重用？举一个简单的例子，假设我们有一个描述飞机的基类（Object Pascal 语言描述）：

```
type
   plane = class
   public
      procedure fly(); virtual; abstract;              //起飞纯虚函数
      procedure land(); virtual; abstract;             //着陆纯虚函数
      function modal() : string; virtual; abstract;
      //查寻型号纯虚函数
   end;
```

然后，我们从 plane 派生出两个子类，直升机（copter）和喷气式飞机（jet）：

```
copter = class(plane)
private
   fModal : String;
public
   constructor Create();
   destructor Destroy(); override;
   procedure fly(); override;
   procedure land(); override;
   function modal() : string; override;
end;

jet = class(plane)
private
   fModal : String;
public
   constructor Create();
   destructor Destroy(); override;
   procedure fly(); override;
   procedure land(); override;
   function modal() : string; override;
end;
```

现在，要完成一个飞机控制系统，有一个全局的函数 plane_fly，它负责让传递给它的飞机起飞，那么只需要用以下语句：

```
procedure plane_fly(const pplane : plane);
begin
   pplane.fly();
end;
```

就可以让所有传给它的飞机（plane 的子类对象）正常起飞，不管是直升机还是喷气机，甚至是现在还不存在的而在以后会增加的飞碟。因为每个子类都已经定义了自己的起飞方式。

可以看到，plane_fly 函数接受参数的是 plane 类对象的引用，而实际传递给它的都是 plane 的子类对象，现在回想一下开头所描述的“多态”：多态性是允许你将父对象设置成为和一个或更多的它的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。很显然，“parent = child;”就是多态的实质，因为直升机“是一种”飞机，喷气机也“是一种”飞机，因此，所有对飞机的操作都可以对它们操作，此时，飞机类就作为一种接口。多态的本质就是将子类类型的指针赋值给父类类型的指针（在 OP 中是引用），只要这样的赋值发生了，多态也就产生了，因为实行了“向上映射”。

应用多态的例子非常普遍，在 Delphi 的 VCL 类库中，最典型的就是 TObject 类有一个虚拟的 Destroy 虚构函数和一个非虚拟的 Free 函数。Free 函数中是调用 Destroy 的。因此，当对任何对象（都是 TObject 的子类对象）调用“.Free();”之后，都会执行“TObject.Free();”，它会调用所使用的对象的析构函数“Destroy();”。这就保证了任何类型的对象都可以正确地被析构。

多态性是面向对象最重要的特性。

面试例题 2：重载和覆盖有什么不同？

答案：虚函数总是在派生类中被改写，这种改写被称为“override”。

override 是指派生类重写基类的虚函数，就像某个类中重写了另一个类中的某个函数，重写的函数必须有一致的参数表和返回值。override 一直没有合适的中文词汇来对应，在此译为“覆盖”更贴切。

overload 约定俗成地被翻译为“重载”，是指编写一个与已有函数同名但是参数表不同的函数。例如，一个函数既可以接受整型数作为参数，也可以接受浮点数作为参数。重载不是一种面向对象的编程，而只是一种语法规则。重载与多态没有什么直接关系。

第10章

继承与接口

接口在实际语言，如Delphi、Java、C++等中，都有广义和狭义之分。

广义接口从一般意义上说，凡是一个类提供给外部使用的部分都可以被称为接口。但是在引入继承和抽象类之前，这个广义接口并没有太大意义。广义接口的真正意义是在类的继承中体现多态的功能，这种接口又被称为抽象类接口。

狭义接口是指特定的函数集合，一般是用interface声明的，它表示一个方法集合，这个集合被称为一个命名接口。一个命名接口中的方法必须在一个类中实现后才能被使用，一个类继承实现一个接口，称为这个类实现了该接口，一个接口可以被多个类实现，一个类也可以继承多个接口，这样就形成了一种灵活的接口调用方式，从而实现更加灵活和节省资源的多态。

从上述内容可知，接口实际上是结合着多态而来的，它的最大任务就是实现多态。而多态又是面向对象最精华的理论，掌握了多态，也就掌握了面向对象的精髓。但掌握多态必须先理解和掌握接口，只有充分理解接口的意义，才能更好地应用多态。

在面试过程中，各大企业会考量你对虚函数、纯虚函数等知识点的掌握程度。因此，这是本书比较难掌握的一章。

10.1 基础知识

面试例题1：下面哪一项说法是正确的？

A．在一个子类中，一个方法不是public的就不能被重载

B．覆盖一个方法只需要满足相同的方法名和参数类型就可以了

C．覆盖一个方法必须要有相同的方法名参数和返回类型

D．一个覆盖的方法必须有相同的方法名、参数名和参数类型

解析：对于在同一可访问区内被声明的几个具有不同参数列（参数的类型、个数、顺序不同）的同名函数，程序会根据不同的参数列来确定具体调用哪个函数，这种机制叫重载，重载不关心函数的返回值类型。覆盖是指派生类中存在重新定义的函数，其函数名、参数列、返回值类型必须同父类中对应被覆盖的函数严格一致，覆盖函数和被覆盖函数只有函数体（花括号中的部分）不同，当派生类对象调用子类中该同名函数时会自动调用子类中的覆盖版本，而不是父类中的被覆盖函数版本，这种机制就叫做覆盖。

成员函数被重载的特征如下：

（1）相同的范围（在同一个类中）。

（2）函数名字相同。

（3）参数不同。

（4）virtual 关键字可有可无。

覆盖的特征如下：

（1）不同的范围（分别位于派生类与基类）。

（2）函数名字相同。

（3）参数相同。

（4）基类函数必须有 virtual 关键字。

答案：C

面试例题 2：关于函数重载，下列说法错误的是哪个？

A．重载函数的函数名必须相同

B．重载函数必须在参数个数或类型上有所不同

C．重载函数的返回值必须相同

D．重载函数的函数体可以有所不同

解析：overload 约定成俗被翻译为“重载”，是指编写一个与已有函数同名但是参数表不同的函数。例如，一个函数可以接受整型数作为参数，也可以接受浮点数作为参数。重载不是一种面向对象的编程，而只是一种语法规则，重载与多态没有什么直接关系。函数的重载是与函数的返回值无关的。

答案：C

面试例题 3：下面的说法中哪项是正确的？

A．静态方法不能被覆盖成非静态的方法

B．静态方法不能被声明成私有的

C．私有的方法不能被重载

D．一个重载的方法在基类中不通过检查不能抛异常

解析：静态的方法不能被覆盖。选项 B 和 C 的说法并不合理，没有合理的理由来说静态的方法不能被声明成私有的，或私有的方法不能被重载。选项 D 是对于一个覆盖方法异常限制的混杂版本来说的。

答案：A

面试例题 4：关于继承表述错误的是哪个？[美国著名数据库公司 SY2009 年 3 月面试题]

A．继承是一种通过扩展一个已有对象的实现，从而获得新功能的复用方法

B．泛化类（超类）可以显式地捕捉那些公共的属性和方法；特殊类（子类）则通过附加属性和方法来进行实现的扩展

C．继承会破坏封装性，因为会将父类的实现细节暴露给子类

D．继承本质上是“黑盒复用”，对父类的修改不会影响到子类

解析：继承（inheritance）是指 Child 类的对象可以使用仅对 Father 类的对象有效的方法或者属性，它使得这些方法和属性就好像是由 Child 类自己定义的一样。此时 Father 类是 Child 类的父类，Child 类是 Father 类的子类。在继承结构中，父类的内部细节对于子类是可见的。所以，通过继承的代码复用是一种“白盒式代码复用”。

组合（composition）是指通过对现有的对象进行拼装（组合）产生新的更复杂的功能。因为在对象之间，各自的内部细节是不可见的，所以，我们也说这种方式的代码复用是“黑盒式代码复用”。

继承和组合有各自的优缺点，比如，继承关系是客观世界及 OOP 中最基本的关系，而且继承是深化接口规范的基础，没有继承就没有多态等诸多 OO 特性。继承在编译期静态地定义其层次结构，使得在技术上来说，继承直白明了，易于使用，而且容易修改通过继承所复用的实现代码，但是不能在运行期间改变通过继承得到的实现代码。更糟糕的是，父类向子类开放了过多的权限（父类常常会定义一部分子类的表现特征，使得父类和子类之间的实现代码产生相互依赖），Effective Java 上面提到的“继承破坏封装性”（inheritance breaks encapsulation）是因为继承引起的代码依赖将会导致诸多问题，例如，通过继承而来的实现代码可能会不适用于新的问题域，从而使得需重写父类或者替换掉某些实现代码。同时，代码之间的依赖关系限制了程序的灵活性和可复用性。

相比而言，对象的组合是在运行期间通过对象之间的引用动态定义的。组合要求对象互相尊重（respect）对方的接口。不过这样做下来，因为对象之间只能唯一地通过接口相互作用，对象的封装性也就得到了良好的维护。同时，在运行期间，任何对象都能够被替换为其他相同类型的对象。更好之处在于，因为对象的实现代码只和其接口有关系，所以由潜在的代码依赖所带来的问题出现的机会就大大减少了。对于整

个系统的设计而言，使用对象组合将保证各个类能够被良好地封装起来，并且保证这些类能够只负责解决唯一的问题。这就使得类的层次结构能保持比较小的规模，而不至于变成难以控制的庞然大物。在基于对象组合进行设计的系统中，会有更多的对象、更少的类，系统的行为将由对象之间的交互关系来决定。

在理想的情况下，我们不需要创建新的组件来完成代码复用，而只需要通过对象组合的方法来拼装已存在的组件以获取新的功能。但这种情况很少出现，因为在实际情况中，现有的组件总是不够，而通过继承来复用代码往往要比通过组合对象容易得多。所以，继承和组合两种方法并存于实际的软件开发过程之中。

举个例子来说，我们通过 Array 来实现一个 Queue 类有两种方法，第一种是把 Queue 作为 Array 的一个子类，也就是通过类继承的方式来实现这个 Queue 类。

```
class Queue extends Array {
        // etc…
    }
```

第二种方法是将一个 Array 类型的实例作为 Queue 类的一个属性，也就是通过对象组合来实现 Queue 类。

```
class Queue extends Object {
        private Array anArray;
        // etc…
}
```

在这个例子中，Queue 类的效果会更好一些。因为这样 Queue 的实例就不用再继承多余的 Collection 类的方法和属性。

答案：D

面试例题 5：给定下面的代码。

```
class Base {}

class Agg extends Base{
      public String getFields(){
       String name = "Agg";
      return name;
      }
}


public class Avf{
public static void main(String argv[]){
      Base a = new Agg();
```

```
        //Here
        }
}
```

What code placed after the comment //Here will result in calling the getFields method resulting in the output of the string"Agg"?（下面哪个选项的代码替换到//Here 会调用 getFields 方法，使输出结果输出字符串“Agg”？）

A．System.out.println(a.getFields());

B．System.out.println(a.name);

C．System.out.println((Base) a.getFields());

D．System.out.println(((Agg) a).getFields());

解析：Base 类型要引用 Agg 类的实例需要把 Base 类显式地转换为 Agg 类，然后调用 Agg 类中的 getFields()方法。如果 a 是 Base 类的一个实例，它要调用 getFields()方法，那此方法在 Base 类中是不存在的，必须把 a 转换为 Agg 类的一个实例，这样才可以调用它的方法。

答案：D

面试例题 6：如果在下列代码中的 Here 处添加一段代码，问哪一个选项不能通过编译？[Trend 公司 2005 年 10 月面试题]

```
public class Upton{
public static void main(String argv[]){
        }
        public void amethod(int i){}
        //Here

}
```

A．public int amethod(int z){}　　B．public int amethod(int i,int j){return 99;}

C．protected void amethod(long l){}　　D．private void anothermethod(){}

解析：选项 A 不能通过编译。一个方法是显式地返回一个 int 值的方法，另一个是在同一个类中上述方法的一个重定义。方法中参数从 i 换做 z 对一个方法并没有任何影响。一个方法不能在同一个类中被覆盖。

答案：A

面试例题 7：下面代码的输出结果是多少？[中国台湾著名杀毒软件公司 T2005，2008 年 10 月面试题]

```
class A {
```

```
    public static void prt() {
        System.out.println("1");
    }

    public A() {
        System.out.println("A");
    }
}

public class B extends A {
    public static void prt() {
        System.out.println("2");
    }

    public B() {
        System.out.println("B");
    }

    public static void main(String[] args) {
        A a = new B();
        a = new A();
    }
}
```

解析：每新建一个对象，都会产生一个构造函数，因为产生构造函数的顺序是 A，B，A。所以结果是 A，B，A。

答案：A，B，A

面试例题 8：下列代码的输出结果是多少？[中国台湾著名杀毒软件公司 T2005，2009 年 10 月面试题]

```
class classA {
 public void printValue() {
       System.out.print("classA ");
 }
}
class classB extends classA {
 public void printValue() {
       System.out.print("classB ");
 }
}
public class Test {
public static void main(String[] args)  {
       classB objectB = new classB();
       objectB.printValue();
       classA as = (classA) objectB;
       as.printValue();
```

```
        as = new classA();
        as.printValue();
    }
}
```

解析：程序解释如下：

```
class classA {
    public void printValue() {
        System.out.print("classA ");
    }
}

class classB extends classA {
    public void printValue() {
        System.out.print("classB ");
    }
}

public class Test {

    public static void main(String[] args)  {
        classB objectB = new classB();  //new 出来的是 classB，输出是"classB "
        objectB.printValue();
        classA as = (classA) objectB;   //objectB 本来定义为 classB，输出是"classB "
        as.printValue();
        as = new classA();              //new 出来的是 classA()，所以输出是"classA "
        as.printValue();
    }
}
```

答案：classB classB classA

面试例题 9：下列程序的输出结果是什么？[中国著名门户网站公司 W2009 年 9 月校园招聘笔试题]

```
import java.util.*;
public  class  Test  {
    public  static  void  main(String[]  args)  {
        Father  father  =  new  Father();
        Father  child  =  new  Child();
        System.out.println(father.getName());
        System.out.println(child.getName());
    }
}

class  Father  {
    public  static  String  getName()  {
```

```
    return  "Father ";
    }
}

class  Child  extends  Father  {
 public  static  String  getName()  {
 return  "Child ";
    }
}
```

A．Father Father　　B．Father Child

C．编译失败　　D．以上答案都不对

解析：因为这两个 getName 方法是静态方法，所以在内存中的地址空间是固定的，根本不存在冲突的问题。也就是说，这两个方法在内存中占用了不同的空间，而具体执行哪一个，则要看是由哪个类来调用的，因为是静态方法，而且两个引用都是 father 的，所以只会调用 father 的方法。

答案：A

扩展知识：本题容易混淆，有些读者想当然地选择 Father Child 的输出结果（覆盖），如果是这样的结果，代码应该进行如下修改：

```
import java.util.*;
public  class  Test  {
public  static  void  main(String[]  args)  {
      Father  father  =  new  Father();
      Father  child  =  new  Child();
      System.out.println(father.getName());
      System.out.println(child.getName());
   }
}

class  Father  {
public     String  getName()  {
return  "Father ";
   }
}

class  Child  extends  Father  {
public     String  getName()  {
return  "Child ";
   }
}
```

10.2 Super

面试例题 1：以下代码的输出结果是下列哪个选项？[美国著名计算机软硬件公司 S2005 年面试题]

```
class Base{
Base(){
       System.out.println("Base");
       }
}

public class Checket extends Base{
 Checket(){
      System.out.println("Checket");
      super();

       }
public static void main(String argv[]){
    Checket c = new Checket();
       }


}
```

A．Compile time error　　B．Checket followed by Base

C．Base followed by Checket　　D．runtime error

解析：这是一个考查 super 关键字与构造函数的题目。子类的构造函数如果要引用 super，必须把 super 放在函数的首位，不然会出现以下报错：

```
Checket.java:10: call to super must be first statement in constructor
        super();
```

如果一定要引用 super 构造函数，则必须把 super()放在前面，代码如下：

```
class Base{
Base(){
       System.out.println("Base");
       }
}

public class Checket extends Base{
 Checket(){

       super();
       System.out.println("Checket");
```

```
        }
public static void main(String argv[]){
        Checket c = new Checket();
        }


}
```

答案：会出现编译报错，选项 A

面试例题 2：Java 里在类中用 super 调用父类构造函数时，为什么调用语句必须是子类的第一条语句？[德国著名软件公司 S2005 年面试题]

答案：如果想用 super 继承父类构造的方法，但是没有放在第一行的话，那么在 super 之前的语句，肯定是为了满足自己想要完成某些行为的语句，但是又用了 super 继承父类的构造方法。那么以前所做的修改就都回到以前，就是说又成了父类的构造方法了。如下面的程序所示：

```
class Father
{
public Father()
{String name=null;
int age=0;}
}
class Son extends Father
{
public Son()
{String name="学生";
super();
}
}
```

扩展知识（Java 中的 super 关键字）

在 Java 中，有时还会遇到子类中的成员变量或方法与超类（有时也称父类）中的成员变量或方法同名。因为子类中的成员变量或方法名优先级高，所以子类中的同名成员变量或方法就隐藏了超类的成员变量或方法，但是我们如果想要使用超类中的这个成员变量或方法，就需要用到 super。请看下面的类。

```
class Country
    {
      String name;
      void value()
      {
        name="China";
```

```
        }
    }
```

在下面的子类中，子类的成员变量和方法隐藏了超类的成员变量 name 和方法 value()。

```
class City extends Country
      String name;
      void value()
      {
        name="Hefei";
        super.value();
        System.out.println(name);
        System.out.println(super.name);
    }
```

为了在子类中引用超类中的成员变量 name 和方法 value()，在代码中使用了 super、super.name 和 super.value()，所以显示的结果为：

```
Hefei
China
```

如果想要使用超类的构造函数，则应当使用 super(参数列表)的形式。

10.3 this

面试例题 1：下面程序的结果是什么？[中国台湾著名软件公司 T 面试题]

```
class Tester
{
 int var;
 Tester(double var)
 {
     this.var = (int)var;

 }

 Tester(int var)
 {
     this("hello");
 }

 Tester(String s)
 {
```

```
        this();
        System.out.println(s);
    }
    Tester()
    {
        System.out.println("good-bye");
    }

    public static void main(String[] args)
    {
        Tester t = new Tester(5);
    }
}
```

答案：

```
good-bye
hello
```

扩展知识（变量的内存分配情况）

在 Java 中有两个非常特殊的关键字：this 和 super，它们在使用前都是不需要声明的。this 关键字使用在一个成员函数的内部，指向当前对象，当前对象指的是调用当前正在执行方法的那个对象。super 关键字是直接指向超类的构造函数，用来引用超类中的变量和方法。下面介绍一下 this 的使用方法。

先看下面的一段代码。

```
class PersonInformation
    {
      String name,gender,nationality,address;
      int age;
      void PersonInformation(String p_name,String p_gender,String
         p_nationality,String p_address,int p_age)
      {
        name=p_name;
        gender=p_gender;
        nationality=p_nationality;
        address=p_address;
        age=p_age;
      }
    }
```

在 PersonInformation()函数中，这个对象的方法提示可以直接访问对象的成员变量，而且在同一个范围中，定义两个相同名字的局部变量是不被允许的。如果确实想使类的成员变量与方法的参数或方法自己定义的局部变量同名，就需要

想一种方法使成员变量与跟它同名的方法参数或局部变量区分开来，这就要使用到 this 变量。下面改写一下上面的代码，使 PersonInformation 类的构造函数中每个参数都有与对象成员变量相同的名字，而成员变量的初值由参数给出。

```
class PersonInformation
   {
     String name,gender,nationality,address;
     int age;
     void PersonInformation(String name,String gender,String
        nationality,String address,int age)
     {
     this.name=name;
     this.gender=gender;
     this.nationality=nationality;
     this.address=address;
     this.age=age;
     }
   }
```

从上例中可以看出，该构造函数中必须使用 this。this 在方法体中用来指向引用当前正在执行方法的那个对象实例。在上例中，我们要区别参数 name 和成员变量 name，写成 name=name 显然是不允许的。在参数或局部变量名与类成员变量名相同的时候，由于参数或局部变量的优先级高，这样在方法体中参数名或局部变量名将隐藏同名的成员变量，因此，为了指明成员变量，必须使用 this 显式地指明当前对象。

有时候会遇到这种情况，全面访问当前对象，而不是访问某一个个别的实例对象，此时也可以使用 this，并利用 Java 中的 toString()方法（它能够返回一个描述这个对象的字符串）。如果把任何一个对象传递到 System.out.println 方法中，这个方法调用这个对象的 toString 方法，并打印出结果字符串，所以，可以用 System.out.println(this)方法来打印出固有参数的当前状态。

this 还有一个用法，就是构造函数的第一个语句，它的形式是 this(参数表)，这个构造函数就会调用同一个类的另一个相对的构造函数。请看下面的例子。

```
class UserInfo

{
     public UserInfo(String name)
     {
       this(name,aNewSerialNumber);
     }
     public Userinfo(String name,int number)
     {
```

```
        userName=name;
        userNumber=number;
      }
  }
```

如果调用 UserInfor newinfotable = new UserInfo("Wayne Zheng")，就会自动调用 UserInfo(String name,int number)构造函数。

可见，熟练掌握 this 在 Java 程序设计过程中是非常重要的。

面试例题 2：以下代码的运行结果是_____。[印度某软件公司 A2009 年 12 月面试题]

```
class Base{

  int i;

  Base(){
  add(1);
  System.out.println(i);
 }

 void add(int v){
   i+=v;
   System.out.println(i);
 }

 void print(){
   System.out.println(i);
  }
}
class MyBase extends Base{
  MyBase(){
   add(2);
 }
  void add(int v){
    i+=v*2;
    System.out.println(i);
  }
}
public class TestClu {
  public static void main(String[] args) {

   go(new MyBase());
  //System.out.println();
 }
   static void go(Base b){
   b.add(8);
  //b.print();
```

```
    }
}
```

A．12　　B．11　　C．22　　D．21

解析：

程序流程是这样的：在主函数中，首先执行 new MyBase()，在这个过程中，子类会首先调用父类的构造函数；在父类的构造函数 Base()中执行 add()方法。这里需要注意，这个 add 方法由于是在新建 MyBase 对象时调用的，将会首先查找 MyBase 类中是否有此方法。所以，Base()函数中的 add(1)实际上是：

```
void add(int v)
    {
    i+=v*2;
    System.out.println(i);
 }
```

此时 i 的值即为 2。在打印两个 2 之后，父类构造函数执行完毕，执行子类的构造函数，即 MyBase()，这里的 add(2)当然也是子类的。i 的值就变为了 6。

最后执行 go 函数，i 即为 22。

答案：C

10.4　不能继承的情况

面试例题 1：Anonymous Inner Class（匿名内部类） 是否可以 extends（继承）其他类，是否可以 implements（实现）interface（接口）？

答案：匿名的内部类是没有名字的内部类，不能 extends（继承） 其他类，但一个内部类可以作为一个接口，由另一个内部类实现。

final 类绝对不要因为性能的原因将类定义为 final 的（除非程序的框架要求）。如果一个类还没有准备好被继承，最好在类文档中注明，而不要将它定义为 final，这是因为没有人可以保证会不会由于什么原因需要继承它。

如果说整个类都是 final（在它的定义前冠以 final 关键字），就表明自己不希望从这个类继承，或者不允许其他任何人采取这种操作。换言之，出于这样或那样的原因，我们的类肯定不需要进行任何改变；或者出于安全方面的理由，我们不希望进行子类化（子类处理）。

除此以外，或许还考虑到执行效率的问题，并想确保涉及这个类各对象的所有行动都要尽可能地有效。如下所示：

```
//: Jurassic.java
//整个类都是final

class SmallBrain {}

final class Dinosaur {
  int i = 7;
  int j = 1;
  SmallBrain x = new SmallBrain();
  void f() {}
}

//!扩展Dinosaur类
// error: 不能扩展final类 (Dinosaur类)

public class Jurassic {
  public static void main(String[] args) {
    Dinosaur n=new Dinosaur();
    n.f();
    n.i=40;
    n.j++;
  }
}
```

注意，数据成员既可以是 final，也可以不是，取决于具体选择。应用于 final 的规则同样适用于数据成员，无论类是否被定义成 final。将类定义成 final 后，结果只是禁止进行继承——没有更多的限制。然而，由于它禁止了继承，所以一个 final 类中的所有方法都默认为 final。因为此时再也无法覆盖它们，所以与将一个方法明确声明为 final 一样，编译器此时有相同的效率选择。

可为 final 类内的一个方法添加 final 指示符，但这样做没有任何意义。

10.5　抽象类与接口

面试例题 1：下列关于抽象类说法错误的是哪个？[印度某软件公司 A2009 年 12 月面试题]

A．抽象类中可以不存在任何抽象的方法

B．抽象类可以被抽象类所继承，结果仍是抽象类

C．抽象类不能同时又是密封的

D．如果一个非抽象类从抽象类中派生，不一定要通过覆盖来实现继承的抽象成员

E．抽象类允许被声明。

解析：有时候，基类并不与具体的事物相联系，而是只表达一种抽象的概念，用以为它的派生类提供一个公共的界面。为此，Java 中引入了抽象类(abstract class)的概念。

C++程序员在这里最容易犯错误。C++中没有对抽象类进行直接声明的方法，而认为只要在类中定义了纯虚函数，这个类就是一个抽象类。

```
//C++中定义抽象类的做法
class Shape //这里不需要显式地声明抽象类
{
      public:
           Shape(){}
           ~Shape(){}
           virtual void Draw()=0; //定义纯虚函数
};
```

纯虚函数的概念比较晦涩，直观上不容易为人们所接受和掌握，因此，Java 抛弃了这一概念，而是通过显式地声明抽象类并使用 abstract 修饰符的办法：

```
abstract class Person
{
     public abstract void SayHello();
     public void about()
     {
          System.out.println("Abstract Demo");
     }
}
```

一个抽象类要注意以下几点：

（1）抽象类只能作为其他类的基类，它不能直接被实例化，而且对抽象类不能使用 new 操作符。抽象类如果含有抽象的变量或值，则它们要么是 null 类型，要么包含了对非抽象类的实例的引用。

（2）抽象类允许包含抽象成员，但这不是必需的（可以允许一个抽象类中没有任何抽象成员）；抽象类中可以有非抽象方法。

（3）抽象类不能同时又是 final 的。如果试图将一个 final 类作为其他类的基类，Java 将提示“The class can be either abstract or final, not both”。理所当然，final 类不能同时又是抽象类，因为抽象总是希望被继承的。

（4）如果一个非抽象类从抽象类中派生，则其必须通过覆盖来实现所有继承而来的抽象成员。

（5）抽象类可以被抽象类所继承，结果仍是抽象类。

（6）抽象类允许被声明。

以下通过一个例子来说明：

```
abstract class Person  //建立一个抽象类 Person
{
     public abstract void SayHello();  //其中存在抽象方法 SayHello
     public void about()               //以及一个非抽象方法 about
     {
          System.out.println("Abstract Demo");
     }
}

class Student extends Person           //建立实体类 Student 继承 Person
{
     public  void SayHello()           //必须覆盖抽象方法 SayHello
     {
          System.out.println("SayHello");
     }
}

class Nurse extends Person             //建立实体类 Nurse 继承 Person
{
     //没有覆盖抽象方法 SayHello。所以这个类是错误的
}

abstract class Pupil extends Person    //建立抽象类 Pupil 继承 Person
{
     public  void SayHello()           //可以覆盖抽象方法 SayHello
     {
          System.out.println("SayHello");
     }
}

abstract class Worker extends Person   //建立抽象类 worker 继承 Person
{
//抽象类继承抽象类，可以不覆盖抽象方法 SayHello。所以这个类是正确的
}
```

让我们研究汽车类的例子。如果我们从“交通工具”这个角度来理解 Vehicle 类，它应该表达一种抽象的概念，我们可以把它定义为抽象类。由轿车类 Car 和卡车类 Truck 来继承这个抽象类，它们作为可以实例化的类。

```
import java.io.*;
abstract class Vehicle
{
     public int wheels;
     protected float weight;
     public abstract void Logo();
```

```
    public Vehicle(int w, float g)
    {
        wheels = w;
        weight = g;
    }
    public void Speak()
    {
        System.out.println("the w vehicle is speaking!");
    }
}

class Car extends Vehicle
{
    int passengers;
    public Car(int w,float g,int p)
    {
     super(w, g);
        wheels=w;
        weight=g;
        passengers=p;
    }
    public  void Speak()
    {
        System.out.println("The car is speaking:Di-di!");
    }
    public  void Logo()
    {
        System.out.println("\n Car ");
    }
}

class Truck extends Vehicle
{
    int passengers;
    float load;
    public Truck(int w, float g, int p, float l)
    {
      super(w, g);
      wheels = w;
        weight = g;
        passengers = p;
        load = l;
    }
    public  void Speak()
    {
        System.out.println("The truck is speaking:Ba-ba!");
    }
```

```
    public  void Logo()
    {
        System.out.println("\n Truck ");
    }

}
public class Test {
    public static void main(String argv[])
    {
      Vehicle vehicle;
      Car car = new Car(1, 2, 3);
      Truck truck = new Truck(3,4,5,6);
      car.Logo();
      truck.Speak();

    }
}
```

答案：D

面试例题 2：What is an 'abstract' class?（什么是抽象类？）[美国著名数据分析软件公司 SA2008 年笔试题]

A．A class with no methods.（一个类没有方法）

B．A class with no concrete subclasses.（一个类没有具体的子类）

C．A class with at least one undefined message.（一个类至少有一个未定义的消息）

D．None of above.（以上都不是）

解析：选项 A、B 显然都不对，至于选项 C，一个抽象类中，即使一个抽象方法没有（空类）也是可以的。

```
abstract  class ppp
{

}
```

此外，即使所有的方法都实现了，只要在类前边加上 abstract，它依然是抽象类。

答案：D

面试例题 3：What will happen when you attempt to compile and run this code?（这段代码输出什么？）[美国著名数据分析软件公司 SA2008 年笔试题]

```
abstract class Base{
```

```
  abstract public void myfunc();
}

public class Abs extends Base{
public static void main(String argv[])
{
  Abs a = new Abs();
  a.amethod();
}
public void amethod(){
  System.out.println("A method");;
}
}
```

A．The code will compile and run, printing out the words"A method"（编译正常运行输出结果为 A method）

B．The compiler will complain errors in Base class.（Base 类编译错误）

C．The code will compile but complain at run time（代码编译正确，但运行出错）

D．The compiler will complain errors in Abs class. （Abs 类编译错误）

解析：这段程序编译错误，运行时编译器会提示：The type Test must implement the inherited abstract method Base.myfunc()。这是因为 Base 是一个抽象类，myfunc()是一个抽象方法，所以，Abs 类继承 Base()时必须继承这个方法。正确的代码如下：

```
abstract class Base{
  abstract public void myfunc();
}

public class Abs extends Base{
public void myfunc()                    //添加 myfunc()
{}
public static void main(String argv[])
{
  Abs a = new Abs();
  a.amethod();
}
public void amethod(){
  System.out.println("A method");;
}
}
```

答案：D

第 3 部分

数据结构和设计模式

Data structure and design pattern

本部分主要介绍求职面试过程中出现的第二个重要的板块——数据结构，包括字符串的使用、堆、栈、排序方法等。此外，随着外企研发机构大量迁入我国，外企针对软件工程知识的考核，包括设计模式、UML、敏捷软件开发，以及.NET 技术和完全面向对象语言 C#的面试题目将会有增无减，今后设计模式在面试中的比重会进一步提高。

第 11 章

数据结构基础

面试时间一般有两个小时，其中至少有 20～30 分钟左右是用来回答数据结构相关的问题，链表栈、堆、数组的排序和逆置是笔试必考的内容。

以下的考题来自真实的笔试资料，希望读者先不要看答案，自我解答后再与答案加以比对，找出自己的不足。

11.1 堆栈

面试例题 1：编号为 123456789 的火车经过如图 11-1 所示的轨道，从左边入口处移到右边出口处(每车都必须且只能进临时轨道 M 一次，且不能再回左边入口处) [中国台湾著名 CPU 生产公司 V2009 年 11 月笔试题]

按照从左向右的顺序，下面的结果不可能的是_____。

A．123876549　　B．321987654

C．321456798　　D．987651234

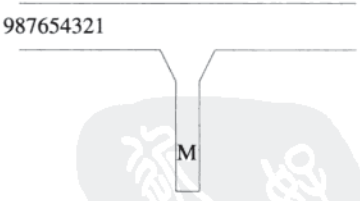


图 11-1　火车轨道示意图

解析：本题实际上考的是数据结构的栈。临时轨道 M 就是栈。

选项 A 中，123 逐个过去，45678 入栈，再出栈变成 87654，9 再过去；

选项 B 中，123 入栈，再出栈变成 321，456789 入栈再出栈，变成 987654；

选项 C 中，123 入栈，再出栈变成 321，4567 直接过去，89 入栈再出栈，变成 98；

选项 D 中，98765 在前，则 1234 必须全部先进栈，98765 过去后，剩下 1234 必须先回到左边，再通过才满足 1234。但是题意要求不可以再回到左边入口处，所以这个

选项不可行。

答案：D

扩展知识：如果 M 只能容纳 4 列车。上面选项应该选哪个才可行？

面试例题 2：Suppose you have an empty stack which have 4 items 1234 input in sequence, which of the following CANNOT be one output in sequence?（空栈置入 1234，输出结果不可能的是多少？）

A．1243　　B．2134　　C．1432　　D．4312

答案：D

面试例题 3：请看如下代码。

```
Image[] img=new Image[10];
   for(int i=0;i<img.length;i++){
   img[i] =Image.createImage("/res/"+i+".png");
   }
   img=null;
```

以上代码在执行过后，是否释放了堆栈空间？请说明理由。如果没有释放，写出正确的释放方法。

解析：

这个问题里有两个问题。

（1）“img=null;”能不能清掉 img 数组中图片对象的引用。

（2）执行 gc 的问题。

第一个问题，“img = null;”后没有原 img 数组中那些图片的引用，也就是没有办法再引用这些内存中的对象，所以，gc 应该可以把它们都列为可回收的对象。因此，没必要再遍历数组逐个清除引用。

第二个问题，J2ME 设备上 gc 的实现一般不同，但基本上还是可以主动调用一下的。显式地调用 gc 不是必要的，但清空引用是必要的。

答案：以上代码在执行过后，没有释放堆栈空间。

正确的释放方法如下：

```
if( img != null ){
   for(int i=0;i<img.length;i++){
      if( img[i]!=null ){
         img[i]=null;
```

```
        }
    }
    img=null;
}
```

面试例题 4：实现一个栈操作，将 1～12 月的英文单词压入栈中，然后再将其取出，使用 Java 代码实现。[德国著名 ERP 软件公司 S2005 年 10 月面试题]

解析：Stack 有时也可以称为“后进先出”（LIFO）集合。换言之，在堆栈里最后“压入”的东西将是以后第一个“弹出”的。和其他所有的 Java 集合一样，压入和弹出的都是“对象”，所以必须对自己弹出的东西进行“造型”。

一种很少见的做法是拒绝使用 Vector 作为一个 Stack 的基本构成元素，而是从 Vector 里“继承”一个 Stack。这样，它就拥有了一个 Vector 的所有特征及行为，另外加上一些额外的 Stack 行为，很难判断出设计者到底是明确地想这样做，还是属于一种固有的设计。

答案：months 数组的每一行都通过 push()继承进入堆栈，稍后用 pop()从堆栈的顶部将其取出。要声明的一点是，Vector 操作也可针对 Stack 对象进行，这可能是由继承的特质决定的——Stack“属于”一种 Vector。因此，能对 Vector 进行的操作也可针对 Stack 进行，例如 elementAt()方法。

```
import java.util.*;

public class Stacks {
 static String[] months = { "January", "February", "March", "April",
   "May","June","July","August","September","October","November",
   "December" };

 public static void main(String[] args) {
      Stack stk = new Stack();
      for (int i = 0; i < months.length; i++)
           stk.push(months[i] + " ");
      System.out.println("stk = " + stk);
      //把栈当做一个 Vector:
      stk.addElement("The last line");
      System.out.println("element 5 = " + stk.elementAt(5));
      System.out.println("popping elements:");
      while (!stk.empty())
           System.out.println(stk.pop());
 }
}
```

扩展知识（变量的内存分配情况）

本题我们也可以通过队列来实现，代码如下。

```
import java.util.*;

public class Stacks {
  static String[] months = { "January", "February", "March",
    "April", "May", "June", "July", "August", "September",
    "October", "November", "December" };

  public static void main(String[] args) {
      Vector vq = new Vector();
      //进队列对象放入尾部
      for (int i = 0; i < months.length; i++)
      vq.addElement(months[i] + " ");


      //出队列从头部取

      if (vq.isEmpty()) {
          System.out.println("kong");
      }



      while (!vq.isEmpty())
          {
          System.out.println(vq.firstElement());
          vq.removeElement(vq.firstElement());
          }
      vq.clear();



  }
} ///:~
```

可以发现，用 new 关键字分配的内存既不在栈中，也不在静态数据区中。

11.2　链表、哈希表

面试例题 1：以下关于链式存储结构的叙述中哪个是正确的？

A．链式存储结构不是顺序存取结构

B．逻辑上相邻的节点物理上必须邻接

C．可以通过计算直接确定第 i 个节点的存储地址

D．插入、删除运算操作方便，不必移动节点

解析：存储结构分以下四种：

（1）随机存取，即可以随意直接存取任意一个元素，可以通过下标直接存取任何一个元素如数组等；又如内存，可以通过地址直接访问任意一个空间。

（2）顺序存取，就是只能从前到后逐个访问。像链表这种结构，不能够直接通过下标访问，必须从表头开始，向后逐个搜索，就是顺序存取。这和磁带一样，想听后边的歌曲，就得把前边的磁带转过去，按照顺序来。

（3）索引存取是指为某个关键字建立索引表，从所有的表中得到地址，再直接访问。索引存取多用在数据管理过程中。

（4）散列存储是建立散列表，它相当于一种索引。

链式存储是顺序存储的，因为在逻辑上，存储的节点不在相邻的物理位置，要访问时需通过前一个节点的指针域来访问下一节点，只能按顺序进行存储和读取，而顺序存储是随机访问数据。

答案：D

面试例题 2：有 1 千万条有重复的短信，以文本文件的形式保存，一行一条，也有重复。请用 5 分钟时间找出重复出现最多的前 10 条。

解析：对于本题来说，某些面试者想用数据库的办法实现：首先将文本导入数据库，再利用 select 语句的方法得出前 10 个短信。但实际上用数据库是绝对满足不了 5 分钟解决这个条件的。这是因为 1 千万条短信即使 1 秒钟导入 1 万条（这已经算是很快的数据导入了），5 分钟才 3 百万条，即便真的能在 5 分钟内录完 1 千万条，也必须先建索引，否则 SQL 语句在 5 分钟内肯定得不出结果。但对 1 千万条记录建索引，在 5 分钟内也不能完成。所以用数据库的办法不行。

这种类型的题之所以会出现，这是因为互联网公司每时每刻都需要处理由用户产生的海量数据/日志，所以海量数据的题现在很热，互联网公司基本上都会考。重点考查你的数据结构设计与算法基本功。类似题目是如何根据关键词搜索访问最多的前 10 个网站。

答案：

方法 1：用哈希表的方法。可以将 1 千万条短信分成若干组，进行边扫描边建散列表的方法。第一次扫描，取首字节、尾字节、中间任意两字节作为 Hash Code，插入到 hash table 中，并记录其地址、信息长度和重复次数。同 hash code 且等长就疑似相

同，比较一下。相同记录只加 1 次进 hash table，但将重复次数加 1。一次扫描以后，已经记录各自的重复次数，进行第二次 hash table 的处理。用线性时间选择可在 O(n) 的级别上完成前 10 条的寻找。分组后每组中的 top10 必须保证各不相同，可用 hash 来保证，也可直接按 hash 值的大小来分类。

方法 2：采用从小到大排序的办法。根据经验，除非是群发的过节短信，否则字数越少的短信，出现重复的概率越高。建议从字数少的短信开始找起，比如一开始搜一个字的短信，找出重复出现的 top10 并分别记录出现次数，然后搜两个字的，以此类推。对于对相同字数的比较长的短信的搜索，除了 hash 之类的算法外，可以选择只抽取头、中和尾等几个位置的字符进行粗判，因为此种判断方式是为了加快查找速度，但未必能得到真正期望的 top10，因此，需要做标记；如此搜索一遍后，可以从各次 top10 结果中找到备选的 top10，如果这次 top10 中有刚才做过标记的，则对其对应字数的所有短信进行精确搜索，以找到真正的 top10 并再次比较。

方法 3：采用内存映射办法。首先，1 千万条短信按现在的短信长度将不会超过 1GB 空间，使用内存映射文件比较合适，可以一次映射（如果有更大的数据量，可以采用分段映射），由于不需要频繁使用文件 I/O 和频繁分配小内存，这将大大提高了数据的加载速度。其次，对每条短信的第 i（i 从 0 到 70）个字母按 ASCII 码进行分组，也就是创建树。i 是树的深度，也是短信第 i 个字母。

该问题主要是解决两方面的内容，一是内容加载，二是短信内容的比较。采用文件内存映射技术可以解决内容加载的性能问题(不仅仅不需要调用文件 I/O 函数，而且也不需要每读出一条短信都要分配一小块内存)，而使用树技术可以有效地减少比较的次数。

11.3　树、图

面试例题 1：There is binary search tree which is used to store characters 'A', 'B','C','D','E','F','G','H',which of the following is post-order tree walk result? （有一个二叉搜索树用来存储字符'A'，'B'，'C'，'D'，'E'，'F'，'G'，'H'，下面哪个结果是后序树遍历结果）

A．ADBCEGFH　　B．BCAGEHFD

C．BCAEFDHG　　D．BDACEFHG

解析：二叉搜索树（Binary Search Tree）或者是一棵空树，或者是具有下列性质的二叉树：对于树中的每个节点 X，它的左子树中所有关键字的值都小于 X 的关键字值，它的右子树中的所有关键字值都大于 X 的关键字值。这意味着该树所有的元素都可以用某种统一的方式排序。

例如，下面就是一棵合法的二叉搜索树：

它的左、右子树也分别为二叉搜索树。

二叉搜索树的查找过程和次优二叉树类似，通常采取二叉链表作为二叉搜索树的存储结构。中序遍历二叉搜索树可得到一个关键字的有序序列，一个无序序列可以通过构造一棵二叉搜索树变成一个有序序列，构造树的过程即为对无序序列进行排序的过程。每次插入的新节点都是二叉搜索树上新的叶子节点，在进行插入操作时，不必移动其他节点，只需改动某个节点的指针，由空变为非空即可。搜索、插入、删除的复杂度等于树高，即为 O(log(n))。

二叉树的一个重要的应用是它们在查找中的使用。二叉搜索树的概念相当容易理解，二叉搜索树的性质决定了它在搜索方面有着非常出色的表现：要找到一棵树的最小节点，只需要从根节点开始，只要有左子树就向左进行，终止节点就是最小的节点。找最大的节点则是往右进行。例如上面的例子中，最小的节点是 1，在最左边；最大的节点是 8，在最右边。

对于本题而言，二叉搜索树则必须满足对树中任一非叶子节点，其左子树都小于该节点值，右子树所有的节点值都大于该节点值。结合二叉树后序遍历的特点，最后一个肯定是根节点。

选项 A 中，

- (H)左子树((ADBCEGF)、右子树(空)(左子树必须都小于根 H，右子树都大于根 H)。
- (F)左子树(ADBCE)、右子树(G)。
- (E)左子树(ADBC)、右子树(空)。
- (C)剩下(ADB)不能区别左子树、右子树，所以选项 A 不成立。

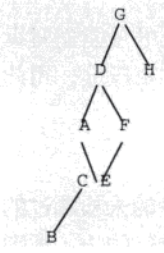
选项 B 中，

- (D、(BCA)、(GEHF))。
- GEHF、F 为根，剩下 GEH 不能根据 F 分成两个子段，所以 B 不成立。

选项 C 中，

- (G, (BCAEFD), (H))。

- (G, (D, (BCA), (EF)), (H))。
- (G, (D, (A, (), (BC)), (F, (E), ())), (H))。
- (G, (D, (A, (), (C, (B), ())), (F, (E), ())), (H))。



选项 C 成立；

选项 D 中，

- (G, (BDACEF), (H))。
- (G, (F, (BDACE), ()), (H))。
- (G, (F, (E, (BDAC), ()), ()), (H))。
- BDAC 子树，C 为根，据 C 不能将序列 BDA 划分为两个子序列，使得左子序列全小于 C，右子序列全大于 C。

所以选项 D 不成立。

答案：C

面试例题 2：Below is an undirected diagram with 7 nodes, From node 1, the depth-first and breadth-first traversal will obtain sequences individually of _____（如图 11-2 所示，深度优先和广度优先得到的节点排序分别是多少）

A．1354267,1534267

B．1347652,1726453

C．1534276,1354276

D．1247653,1247653

图 11-2　图的深度优先和广度优先遍历

解析：本题要先搞清楚什么是图的深度优先遍历？什么是图的广度优先遍历？

1．图的深度优先遍历的递归定义

假设给定图 G 的初态是所有的顶点均未曾访问过。在 G 中任选一顶点 v 为初始出发点(源点)，则深度优先遍历可定义如下：首先访问出发点 v，并将其标记为已访问过；

然后依次从 v 出发搜索 v 的每个邻接点 w。若 w 未曾访问过，则以 w 为新的出发点继续进行深度优先遍历，直至图中所有和源点 v 有路径相通的顶点(亦称为从源点可达的顶点)均已被访问为止。若此时图中仍有未访问的顶点，则另选一个尚未访问的顶点作为新的源点重复上述过程，直至图中所有的顶点均已被访问为止。

图的深度优先遍历类似于树的前序遍历。采用的搜索方法的特点是尽可能先对纵深方向进行搜索。这种搜索方法称为深度优先搜索(Depth-First Search)。相应地，用此方法遍历图就很自然地称之为图的深度优先遍历。

2．深度优先搜索的过程

设 x 是当前被访问顶点，在对 x 做过访问标记后，选择一条从 x 出发的未检测过的边(x，y)。若发现顶点 y 已访问过，则重新选择另一条从 x 出发的未检测过的边，否则沿边(x，y)到达未曾访问过的 y，对 y 访问并将其标记为已访问过；然后从 y 开始搜索，直到搜索完从 y 出发的所有路径，即访问完所有从 y 出发可达的顶点之后，才回溯到顶点 x，并且再选择一条从 x 出发的未检测过的边。上述过程直至从 x 出发的所有边都已检测过为止。此时，若 x 不是源点，则回溯到在 x 之前被访问过的顶点，否则图中所有和源点有路径相通的顶点(即从源点可达的所有顶点)都已被访问过，若图 G 是连通图，则遍历过程结束，否则继续选择一个尚未被访问的顶点作为新源点，进行新的搜索过程。

3．广度优先遍历的递归定义

设图 G 的初态是所有的顶点均未访问过。在 G 中任选一顶点 v 为源点，则广度优先遍历可以定义为：首先访问出发点 v，接着依次访问 v 的所有邻接点 w1，w2，…，wt，然后依次访问与 w1，w2，…，wt 邻接的所有未曾访问过的顶点。以此类推，直至图中所有和源点 v 有路径相通的顶点都已访问到为止。此时从 v 开始的搜索过程结束。

若 G 是连通图，则遍历完成，否则，在图 C 中另选一个尚未访问的顶点作为新源点继续上述的搜索过程，直至 G 中所有的顶点均已被访问为止。

广度优先遍历类似于树的按层次遍历。采用的搜索方法的特点是尽可能先对横向进行搜索，故称其为广度优先搜索（Breadth-First Search）。相应的遍历也就自然地称为广度优先遍历。

4．广度优先搜索的过程

在广度优先搜索过程中，设 x 和 y 是两个相继要被访问的未访问过的顶点。它们的邻接点分别记为 x1，x2，…，xs 和 y1，y2，…，yt。

为确保先访问的顶点其邻接点亦先被访问，在搜索过程中使用 FIFO 队列来保存已访问过的顶点。当访问 x 和 y 时，这两个顶点相继入队。此后，当 x 和 y 相继出队时，我们分别从 x 和 y 出发搜索其邻接点 x1，x2，…，xs 和 y1，y2，…，yt，对其中未访

者进行访问并将其入队。这种方法是将每个已访问的顶点入队，故保证了每个顶点至多只有一次入队。

答案：C

11.4　排序基础知识

面试例题 1：在第一趟排序之后，一定能把数据表中最大或最小元素放在其最终位置上的排序算法是____。

A．冒泡排序　　B．基数排序　　C．快速排序　　D．归并排序

解析：该题考查排序比较问题。冒泡法第一次就实现最大（最小）元素的到位。

答案：A

扩展知识：排序基础知识

排序面试题是各大 IT 公司必考的题目。

所谓排序，就是整理文件中的记录，使之按关键字递增(或递减)次序排列起来。其确切定义如下：

输入：n 个记录 R_1，R_2，…，R_n，其相应的关键字分别为 K_1，K_2，…，K_n。

输出：R_{i1}，R_{i2}，…，R_{in}，使得 $K_{i1} \leqslant K_{i2} \leqslant \cdots = K_{in}$(或 $K_{i1} \geqslant K_{i2} \geqslant \ldots \geqslant K_{in}$)。

（1）被排序对象——文件

被排序的对象——文件由一组记录组成。

记录则由若干个数据项（或域）组成，其中有一项可用来标识一个记录，称为关键字项。该数据项的值称为关键字（Key）。

（2）排序运算的依据——关键字

用做排序运算依据的关键字可以是数字类型，也可以是字符类型。

关键字的选取应根据问题的要求而定。

在高考成绩统计中，将每个考生作为一个记录，每条记录包含准考证号、姓名、各科的分数和总分数等项内容。若要唯一地标识一个考生的记录，则必须用“准考证号”作为关键字。若要按照考生的总分数排名次，则需用“总分数”作为关键字。

1．排序的稳定性

当待排序记录的关键字均不相同时，排序结果是唯一的，否则排序结果不唯一。

在待排序的文件中，若存在多个关键字相同的记录，经过排序后，这些具有相同关键字的记录之间的相对次序保持不变，该排序方法是稳定的；若具有相同关键字的记录之间的相对次序发生变化，则称这种排序方法是不稳定的。排序算法的稳定性是针对所有输入的实例而言的，即在所有可能的输入实例中，只要有一个实例使得算法不满足稳定性要求，则该排序算法就是不稳定的。稳定的排序如表 11-1 所示。

表 11-1　稳定的排序

稳定的排序	时间复杂度	空间复杂度
气泡排序（bubble sort）	最差、平均都是 $O(n^2)$；最好是 O(n)	1
鸡尾酒排序 (Cocktail sort，双向的冒泡排序)	最差、平均都是 $O(n^2)$；最好是 O(n)	1
插入排序 （insertion sort）	最差、平均都是 $O(n^2)$；最好是 O(n)	1
归并排序 （merge sort）	最差、平均和最好都是 O(nlog n)	O(n)
桶排序 （bucket sort）	O(n)	O(k)
基数排序（Radix Sort）	O(dn)（d 是常数）	O(n)
二叉树排序 （Binary tree sort）	O(n log n)	O(n)
图书馆排序（Library sort）	O(n log n)	(1+ε)n

不稳定的排序如表 11-2 所示。

表 11-2　不稳定的排序

不稳定的排序	时间复杂度	空间复杂度
选择排序 （selection sort）	最差、平均都是 $O(n^2)$	1
希尔排序 （shell sort）	O(n log n)	1
堆排序 （heapsort）	最差、平均情况和最好情况都是 O(n log n)	1
快速排序 （quicksort）	平均是 O(n log n)，最坏情况下是 $O(n^2)$	O(log n)

2．排序方法的分类

（1）按是否涉及数据的内、外存交换分。

在排序过程中，若整个文件都放在内存中处理，排序时不涉及数据的内、外存交换，则称之为内部排序（简称内排序）；反之，若排序过程中要进行数据的内、外存交换，则称之为外部排序。

注意：

① 内排序适用于记录个数不很多的小文件。

② 外排序则适用于记录个数太多、不能一次将其全部记录放入内存的大文件。

（2）按策略划分内部排序方法。

可以分为五类：插入排序、选择排序、交换排序、归并排序和分配排序。

① 排序算法的基本操作。

大多数排序算法都有两个基本的操作：

- 比较两个关键字的大小。
- 改变指向记录的指针或移动记录本身。

其中第 2 种基本操作的实现依赖于待排序记录的存储方式。

3. 待排文件的常用存储方式

（1）以顺序表（或直接用向量）作为存储结构。

排序过程：对记录本身进行物理重排（即通过关键字之间的比较判定，将记录移到合适的位置）。

（2）以链表作为存储结构。

排序过程：无须移动记录，仅需修改指针。通常将这类排序称为链表（或链式）排序。

（3）用顺序的方式存储待排序的记录，但同时建立一个辅助表（如包括关键字和指向记录位置的指针组成的索引表）。

排序过程：只需对辅助表的表目进行物理重排（即只移动辅助表的表目，而不移动记录本身）。适用于难以在链表上实现，仍需避免排序过程中移动记录的排序方法。

4. 排序算法的性能评价

（1）评价排序算法好坏的标准。

评价排序算法好坏的标准主要有两条：

① 执行时间和所需的辅助空间。

② 算法本身的复杂程度。

（2）排序算法的空间复杂度。

若排序算法所需的辅助空间并不依赖于问题的规模 n，即辅助空间是 O(1)，则称之为就地排序(In-PlaceSou)。非就地排序要求的辅助空间一般为 O(n)。

（3）排序算法的时间开销。

大多数排序算法的时间开销主要是关键字之间的比较和记录的移动。有的

排序算法其执行时间不仅依赖于问题的规模，还取决于输入实例中数据的状态。

面试例题2：以下哪种排序属于稳定排序?

A．归并排序　　B．快速排序　　C．希尔排序　　D．堆排序

解析：只有归并排序是稳定排序，其他三个都是不稳定的。

答案：C

扩展知识：各种排序方法比较

按平均时间将排序分为四类。

（1）平方阶($O(n^2)$)排序。一般称为简单排序，例如直接插入、直接选择和冒泡排序。

（2）线性对数阶(O(nlgn))排序。如快速、堆和归并排序。

（3）$O(n^{1+£})$阶排序。£是介于 0 和 1 之间的常数，即 0<£<1，如希尔排序。

（4）线性阶(O(n))排序。如桶、箱和基数排序。

简单排序中直接插入最好，快速排序最快，当文件为正序时，直接插入排序和冒泡排序均最佳。

因为不同的排序方法适应不同的应用环境和要求，所以选择合适的排序方法应综合考虑下列因素:

- 待排序的记录数目 n。
- 记录的大小（规模）。
- 关键字的结构及其初始状态。
- 对稳定性的要求。
- 语言工具的条件。
- 存储结构。
- 时间和辅助空间复杂度等。

不同条件下，排序方法的选择:

① 若 n 较小（如 n≤50），可采用直接插入或直接选择排序。

当记录规模较小时，直接插入排序较好。否则因为直接选择移动的记录数少于直接插入，应选直接选择排序为宜。

② 若文件初始状态基本有序（指正序），则应选用直接插入、冒泡或随机的快速排序为宜。

③ 若 n 较大，则应采用时间复杂度为 O(nlgn)的排序方法：快速排序、堆排序或归并排序。

快速排序是目前基于比较的内部排序中被认为是最好的方法，当待排序的关键字是随机分布时，快速排序的平均时间最短。

堆排序所需的辅助空间少于快速排序，并且不会出现快速排序可能出现的最坏情况。这两种排序都是不稳定的。

若要求排序稳定，则可选用归并排序。但本章介绍的从单个记录起进行两两归并的排序算法并不值得提倡，通常可以将它和直接插入排序结合在一起使用，即先利用直接插入排序求得较长的有序子文件，然后再两两归并之。因为直接插入排序是稳定的，所以改进后的归并排序仍是稳定的。

面试例题 3：下列排序算法中，哪种排序在某趟结束后不一定选出一个元素放到其最终的位置上。

A．选择　　B．冒泡　　C．归并　　D．堆

答案：C

面试例题 4：就平均性能而言，下列排序算法中哪种排序最快？

A．希尔　　B．冒泡　　C．交换　　D．快速

答案：D

面试例题 5：从平均角度而言，哪种结构获取任意一个指定值最快？

A．二叉排序树　B．哈希表　　C．栈　　D．队列

解析：一般来说，哪个需要的额外空间越多，哪个就越快。

哈希表和哈希函数是大学数据结构中的知识，实际开发中，我们经常用到 Hashtable 这种结构，当遇到键-值对存储时，采用 Hashtable 比 ArrayList 查找的性能高。为什么呢？我们在享受高性能的同时，需要付出高额外空间的代价。那么使用 Hashtable 是否就是很好的选择呢？就此疑问，做分析如下。

1．对于键-值查找性能高

数据结构描述线性表和树时，记录在结构中的相对位置是随机的，记录和关键字之间不存在明确的关系，因此，在查找记录的时候，需要进行一系列的关键字比较，这种查找方式建立在比较的基础之上，在 Java 中（Array、ArrayList、List）的这些集合结构采用了上面的存储方式。比如，现在我们有一个班同学的数据，包括姓名、性别、年龄、学号等。假设数据如下：

姓名	性别	年龄	学号
张三	男	15	1

```
李四    女        14          2
王五    男        14          3
```

如果我们按照姓名来查找，假设查找函数 FindByName(string name)：

① 查找“张三”。

只需在第一行匹配一次。

② 查找“王五”。

在第一行匹配，失败；

在第二行匹配，失败；

在第三行匹配，成功。

上面两种情况分别分析了最好的情况和最坏的情况，那么平均查找次数应该为(1+3)/2=2 次，即平均查找次数为(记录总数+1)的 1/2。尽管有一些优化的算法可以使查找排序效率增高，但是复杂度会保持在 log2n 的范围之内。

如何更快地进行查找呢？我们所期望的效果是一下子就定位到要找记录的位置上，这时候的时间复杂度为 1，查找最快。如果我们事先为每条记录编一个序号，然后让他们按号入位，我们又知道按照什么规则对这些记录进行编号的话，如果我们再次查找某个记录的时候，只需要先通过规则计算出该记录的编号，然后根据编号，在记录的线性队列中就可以轻易找到记录了。

注意，上述的描述包含了两个概念，一个是用于对学生进行编号的规则，在数据结构中，称之为哈希函数；另外一个是按照规则为学生排列的顺序结构，称之为哈希表。

仍以上面的学生为例，假设学号就是规则，老师手上有一个规则表，在排座位的时候也按照这个规则来排序。要查找李四，首先该教师会根据规则判断出李四的编号为 2，就是在座位中的 2 号位置直接走过去，“李四，哈哈，原来你在这！”

流程如下：

从上面可以看出哈希表可以描述为两个表，一个表用来装记录的位置编号，另一个表用来装记录。此外，存在一套规则，该规则用来表述记录与编号之间的联系。这个规则通常是如何制定的呢？

① 直接定址法：对于整型的数据，GetHashCode()函数返回的就是整型，其实就是基于直接定址的方法，比如有一组 0～100 的数据，用来表示人的年龄，那么，采用直接定址的方法构成的哈希表为：

```
0     1    2    3    4    5
0岁   1岁  2岁  3岁  4岁  5岁
```

.....

这样的一种定址方式简单方便，适用于元数据能够用数字表述或者原数据具有鲜

明顺序关系的情形。

② 数字分析法：有这样一组数据，用于表述一些人的出生日期。

```
年    月   日
75    10   1
75    12   10
75    02   14
```

分析一下，年和月的第一位数字基本相同，造成冲突的概率非常大，而后面三位差别比较大，所以采用后三位。

③ 平方取中法：取关键字平方后的中间几位作为哈希地址。

④ 折叠法：将关键字分割成位数相同的几部分，最后一部分位数可以不相同，然后取这几部分的叠加和（去除出进位）作为哈希地址，比如有这样的数据

```
20144545473
```

可以为

```
      5473
+     4454
+      201
=    10128
```

去除进位 1，取 0128 为哈希地址。

⑤ 取余法：取关键字被某个不大于哈希表表长 m 的数 p 除后所得余数为哈希地址。H(key)=key MOD p (p<=m)。

⑥ 随机数法：选择一个随机函数，取关键字的随机函数值为它的哈希地址，即 H(key)=random(key)，其中 random 为随机函数。通常用于关键字长度不等时。

总之，哈希函数的规则是：通过某种转换关系，使关键字适度地分散到指定大小的顺序结构中。数据越分散，则以后查找的时间复杂度越小，空间复杂度越高。

2．使用 hash 付出的代价

hash 是一种典型的以空间换时间的算法，比如原来一个长度为 100 的数组，对其进行查找，只需要遍历且匹配相应的记录即可，从空间复杂度上看，假如数组存储的是 byte 类型数据，那么该数组占用 100byte 空间。现在我们采用 hash 算法，我们前面说的 hash 必须有一个规则：约束键与存储位置的关系，那么就需要一个固定长度的 hash 表，此时，仍然是 100byte 的数组，假设我们需要的 100byte 用来记录键与位置的关系，那么总的空间为 200byte，而且用于记录规则的表的大小会根据规则的大小而不定。

hash 表最突出的问题在于冲突，就是两个键值经过哈希函数计算出来的索引位置很可能相同。

答案：B

面试例题 6：Which of the following operation performs NOT faster on an ordered data over a disordered data （有序队列数据相对于无序队列数据，下列哪种操作并非快一些？）

A．Find the minimum（找出最小值）

B．Calculate the average value（估算平均值）

C．Find the median（找出中间值）

D．Find the one with maximal occurrence（找出最大出现的可能性）

解析：对于这四种情况分别分析如下：

对于寻找最小值：有序队列数据的时间复杂度是 O(1)，无序队列数据的时间复杂度是 O(n)；

对于估算平均值：有序队列数据的时间复杂度是 O(n)，无序队列数据的时间复杂度是 O(n)；

对于找出中间值：有序队列数据的时间复杂度是 O(1)，无序队列数据的时间复杂度是 O(n)（O(n)的算法类似于快排）；

对于找出最大出现的可能性：有序队列数据的时间复杂度是 O(n)，无序队列数据的时间复杂度是 O(nlgn)（使用平衡查找结构而不是哈希表）。

答案：B

面试例题 7：表序列为（b c d e f g q r s t），则在二分法查找关键字 b 的过程中，先后进行比较的关键字依次是_____。

A．f c b　　B．f d b　　C．g c b　　D．g d b

解析：二分法查找是指已知有序队列中找出与给定关键字相同的数的具体位置。原理是分别定义三个指针 low、high、mid，分别指向待查元素所在范围的下界、上界以及区间的中间位置，即 mid＝(low＋high)/2，让关键字与 mid 所指的数比较，若相等，则查找成功并返回 mid，若关键字小于 mid 所指的数，则 high =mid-1，否则 low=mid+1，然后继续循环，直到找出或找不到为止。对本题而言，要比较三个关键字，分别是 f、e、b。具体情况如图所示：

答案：A

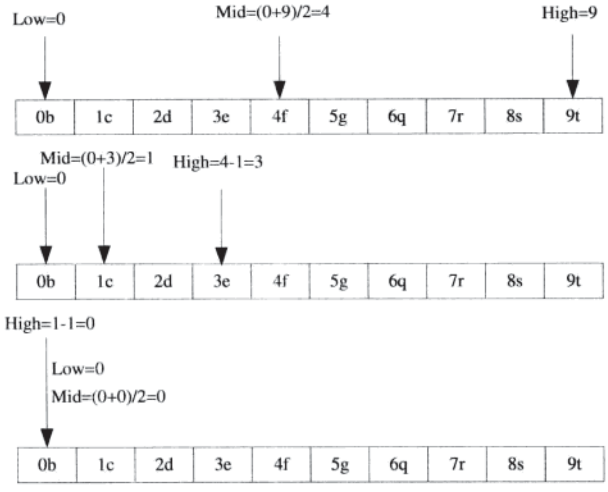


图 11-3　查找关键字

11.4.1　冒泡排序

面试例题 1：请用 Java 写出一个冒泡排序程序，要求输入 10 个整数，输出排序结果。

解析：本题考查交换排序中冒泡排序的问题。

答案：程序如下：

```
public class Test{

    public static void bubbleSort(int[] source) {

        for (int i = source.length - 1; i > 0; i--) {
            for (int j = 0; j < i; j++) {
                if (source[j] > source[j + 1]) {
                    swap(source, j, j + 1);
                }
            }
        }

    }

    private static void swap(int[] source, int x, int y) {
            int temp = source[x];
            source[x] = source[y];
```

```
                source[y] = temp;
            }

    public  static  void  main(String[]  args)  {
         int [] a = {4, 2, 1, 6, 3, 6, 0, -5, 1, 1 };
        int i;
        bubbleSort(a);

        for(i=0;i <10;i++)
         System.out.printf("%d  ",a[i]);
    }
}
```

扩展知识：交换排序的算法思想

交换排序的基本思想是：两两比较待排序记录的关键字，发现两个记录的次序相反时即进行交换，直到没有反序的记录为止。

应用交换排序基本思想的主要排序方法有：冒泡排序和快速排序。

1．排序方法

将被排序的记录数组 R[1..n]垂直排列，每个记录 R[i]看做是重量为 R[i].key 的气泡。根据轻气泡不能在重气泡之下的原则，从下往上扫描数组 R，凡扫描到违反本原则的轻气泡，就使其向上“飘浮”，如此反复进行，直到最后任何两个气泡都是轻者在上、重者在下为止。

（1）初始。

R[1..n]为无序区。

（2）第一趟扫描。

从无序区底部向上依次比较相邻的两个气泡的重量，若发现轻者在下、重者在上，则交换二者的位置，即依次比较(R[n]，R[n-1])，(R[n-1]，R[n-2])，…，(R[2]，R[1])；对于每对气泡(R[j+1]，R[j])，若 R[j+1].key<R[j].key，则交换 R[j+1]和 R[j]的内容。

第一趟扫描完毕时，“最轻”的气泡就飘浮到该区间的顶部，即关键字最小的记录被放在最高位置 R[1]上。

（3）第二趟扫描。

扫描 R[2..n]。扫描完毕时，“次轻”的气泡飘浮到 R[2]的位置上……

最后，经过 n-1 趟扫描可得到有序区 R[1..n]。

注意：

第 i 趟扫描时，R[1..i-1]和 R[i..n]分别为当前的有序区和无序区。扫描仍是

从无序区底部向上，直至该区顶部。扫描完毕时，该区中最轻的气泡飘浮到顶部位置 R[i]上，结果是 R[1..i]变为新的有序区。

2．排序算法

（1）分析。

因为每一趟排序都使有序区增加了一个气泡，在经过 n-1 趟排序之后，有序区中就有 n-1 个气泡，而无序区中气泡的重量总是大于或等于有序区中气泡的重量，所以，整个冒泡排序过程至多需要进行 n-1 趟排序。

若在某一趟排序中未发现气泡位置的交换，则说明待排序的无序区中所有的气泡均满足轻者在上、重者在下的原则。因此，冒泡排序过程可在此趟排序后终止。为此，在下面给出的算法中，引入一个布尔量 exchange，在每趟排序开始前，先将其置为 FALSE。若排序过程中发生了交换，则将其置为 TRUE。各趟排序结束时检查 exchange，若未曾发生过交换，则终止算法，不再进行下一趟排序。

（2）具体算法。

```
void BubbleSort(SeqList R)
{ //R(1..n)是待排序的文件，采用自下向上扫描，对R做冒泡排序
  int i, j;
  Boolean exchange;  //交换标志
  for(i=1;i<n;i++){  //最多做n-1趟排序
    exchange=FALSE;  //本趟排序开始前，交换标志应为假
    for(j=n-1;j>=i; j--)          //对当前无序区R[i..n]自下向上扫描
     if(R[j+1].key<R[j].key){  //交换记录
       R[0]=R[j+1];  //R[0]不是哨兵，仅做暂存单元
       R[j+1]=R[j];
       R[j]=R[0];
       exchange=TRUE;             //发生了交换，故将交换标志置为真
      }
    if(!exchange)                 //本趟排序未发生交换，提前终止算法
        return;
  } //endfor(外循环)
 } //BubbleSort
```

3．算法分析

（1）算法的最好时间复杂度。

若文件的初始状态是正序的，一趟扫描即可完成排序。所需的关键字比较次数 C 和记录移动次数 M 均达到最小值：

$C_{min}=n-1$

$M_{min}=0$

冒泡排序最好的时间复杂度为 O(n)。

（2）算法的最坏时间复杂度。

若初始文件是反序的，需要进行 n-1 趟排序。每趟排序要进行 n-i 次关键字的比较($1 \leqslant i \leqslant n-1$)，且每次比较都必须移动记录三次来达到交换记录位置。在这种情况下，比较和移动次数均达到最大值：

$C_{max}=n(n-1)/2=O(n^2)$

$M_{max}=3n(n-1)/2=O(n^2)$

冒泡排序的最坏时间复杂度为 $O(n^2)$。

（3）算法的平均时间复杂度为 $O(n^2)$。

虽然冒泡排序不一定要进行 n-1 趟，但由于它的记录移动次数较多，故平均时间性能比直接插入排序要差得多。

（4）算法稳定性。

冒泡排序是就地排序，且它是稳定的。

11.4.2 选择排序

面试例题 1：请用 Java 写出一个选择排序，要求输入 10 个整数，输出排序结果。

解析：选择排序（Selection Sort）是一种简单直观的排序算法。它的工作原理如下：首先在未排序序列中找到最小元素，存放到排序序列的起始位置，然后，再从剩余未排序元素中继续寻找最小元素，放到排序序列末尾。以此类推，直到所有的元素均排序完毕。

选择排序的交换操作介于 0 和(n-1)次之间；选择排序的比较操作为 n(n-1)/2 次之间；选择排序的赋值操作介于 0 和 3(n-1)次之间；其平均复杂度为 $O(n^2)$。

答案：完整代码如下：

```
public class Test{

   public static void selectSort(int[] source) {

      for (int i = 0; i < source.length; i++) {
         for (int j = i + 1; j < source.length; j++) {
            if (source[i] > source[j]) {
               swap(source, i, j);
            }
```

```
            }
        }

    }

    private static void swap(int[] source, int x, int y) {
            int temp = source[x];
            source[x] = source[y];
            source[y] = temp;
        }

public  static  void  main(String[]  args)  {
    int [] a = {4, 2, 1, 6, 3, 6, 0, -5, 1, 1 };
    int i;
    selectSort(a);

    for(i=0;i <10;i++)
     System.out.printf("%d  ",a[i]);
}
}
```

11.4.3　插入排序

面试例题 1：请用 Java 写出一个插入排序程序，要求输入 10 个整数，输出排序结果。

解析：一般来说，插入排序都采用 in-place 在数组上实现。具体算法描述如下：

① 从第一个元素开始，该元素可以认为已经被排序。

② 取出下一个元素，在已经排序的元素序列中从后向前扫描。

③ 如果该元素（已排序）大于新元素，则将该元素移到下一位置。

④ 重复步骤③，直到找到已排序的元素小于或者等于新元素的位置。

⑤ 将新元素插入到该位置中。

⑥ 重复步骤②。

如果目标是把 n 个元素的序列升序排列，那么采用插入排序存在最好情况和最坏情况。最好情况就是：序列已经是升序排列了，在这种情况下，需要进行的比较操作需(n-1)次即可。最坏情况就是：序列是降序排列，那么此时需要进行的比较共有 n(n-1)/2 次。插入排序的赋值操作是比较操作的次数加上(n-1)次。平均来说，插入排序算法复杂度为 O(n2)。因而，插入排序不适合对于数据量比较大的排序应用。但是，如果需要排序的数据量很小，例如，量级小于千，那么插入排序还是一个不错的选择。

答案：完整代码如下：

```
public class Test{

    public static void selectSort(int[] source) {

       for (int i = 1; i < source.length; i++) {
           for (int j = i; (j > 0) && (source[j] < source[j - 1]); j--) {
              swap(source, j, j - 1);
           }
        }

    }

    private static void swap(int[] source, int x, int y) {
              int temp = source[x];
              source[x] = source[y];
              source[y] = temp;
           }

 public  static  void  main(String[]  args)  {
      int [] a = {4, 2, 1, 6, 3, 6, 0, -5, 1, 1 };
     int i;
     selectSort(a);

     for(i=0;i <10;i++)
      System.out.printf("%d  ",a[i]);
 }
}
```

11.4.4 Shell 排序

面试例题 1：请用 Java 写出一个 Shell 排序程序，要求输入 10 个整数，输出排序结果。

解析：

希尔排序（Shell Sort）是插入排序的一种，因 D.L.Shell 于 1959 年提出而得名。Shell 排序的中心思想是将数据进行分组，然后对每一组数据进行排序，在每一组数据都有序之后，就可以对所有的分组利用插入排序进行最后一次排序。这样可以显著减少数据交换的次数，以达到加快排序速度的目的。

答案：完整代码如下：

```
public class Test {
  public static int[] a = { 4, 2, 1, 6, 3, 6, 0, -5, 1, 1 }; // 预设数据数组

  public static void main(String args[]) {
    int i;                    // 循环计数变量
```

```
    int Index = a.length;     // 数据索引变量

    System.out.print("排序前: ");
    for (i = 0; i < Index - 1; i++)
      System.out.printf("%3s ", a[i]);
    System.out.println("");

    ShellSort(Index - 1);     // 选择排序
    // 排序后的结果
    System.out.print("排序后: ");
    for (i = 0; i < Index - 1; i++)
      System.out.printf("%3s ", a[i]);
    System.out.println("");
  }

  public static void ShellSort(int Index) {
    int i, j, k;              // 循环计数变量
    int Temp;                 // 暂存变量
    boolean Change;           // 数据是否改变
    int DataLength;           // 分割集合的间隔长度
    int Pointer;              // 进行处理的位置

    DataLength = (int) Index / 2;      // 初始集合间隔长度

    while (DataLength != 0)            // 数列仍可进行分割
    {
      // 对各个集合进行处理
      for (j = DataLength; j < Index; j++) {
        Change = false;
        Temp = a[j];                   // 暂存 Data[j]的值，待交换值时用
        Pointer = j - DataLength;      // 计算进行处理的位置

        // 进行集合内数值的比较与交换值
        while (Temp < a[Pointer] && Pointer >= 0 && Pointer <= Index) {
          a[Pointer + DataLength] = a[Pointer];
          // 计算下一个欲进行处理的位置
          Pointer = Pointer - DataLength;
          Change = true;
          if (Pointer < 0 || Pointer > Index)
            break;
        }
        // 与最后的数值交换
        a[Pointer + DataLength] = Temp;

        if (Change) {
          // 打印目前排序结果
          System.out.print("排序中: ");
          for (k = 0; k < Index; k++)
```

```
            System.out.printf("%3s ", a[k]);
          System.out.println("");
        }
      }
      DataLength = DataLength / 2; // 计算下次分割的间隔长度
    }
  }
}
```

扩展知识：希尔排序基本思想

先取一个小于 n 的整数 d_1 作为第一个增量，把文件的全部记录分成 d_1 个组，所有距离为 d_1 的倍数的记录放在同一个组中。先在各组内进行直接插入排序；然后取第二个增量 $d_2<d_1$ 重复上述的分组和排序，直至所取的增量 $d_t=1(d_t<d_{t-1}<\ldots<d_2<d_1)$，即所有的记录放在同一组中进行直接插入排序为止。

该方法实质上是一种分组插入方法。

给定实例的 Shell 排序的排序过程：假设待排序文件有 10 个记录，其关键字分别是：

49，38，65，97，76，13，27，49，55，04。

增量序列的取值依次为：

5，3，2，1

排序过程演示如图 11-4 所示。

Shell 排序的算法实现：

```
void ShellPass(SeqList R, int d)
{//希尔排序中的一趟排序，d 为当前增量
  for(i=d+1;i<=n; i++)     //将 R[d+1.. n]分别插入各组当前的有序区
    if(R[i].key<R[i-d].key){
      R[0]=R[i];j=i-d;     //R[0]只是暂存单元，不是哨兵
      do {//查找 R[i]的插入位置
         R[j+d]; =R[j];    //后移记录
         j=j-d;            //查找前一记录
      }while(j>0&&R[0].key<R[j].key);
      R[j+d]=R[0];         //插入 R[i]到正确的位置上
    } //endif
} //ShellPass

void  ShellSort(SeqList R)
{
 int increment=n;          //增量初值，不妨设 n>0
 do {
      increment=increment/3+1;     //求下一增量
```

```
        ShellPass(R, increment);      //一趟增量为 increment 的 Shell 插入排序
    }while(increment>1)
} //ShellSort
```

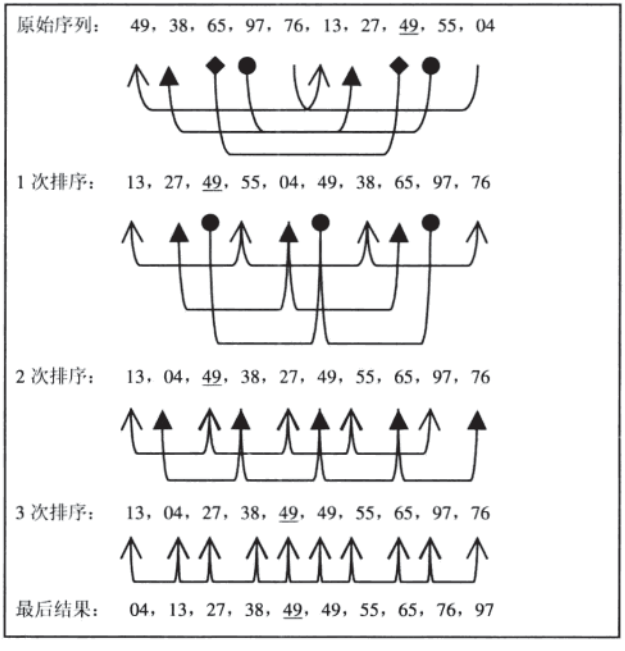


图 11-4　Shell 排序示例

注意：当增量 d=1 时，ShellPass 和 InsertSort 基本一致，只是由于没有哨兵而在内循环中增加了一个循环判定条件“j>0”，以防下标越界。希尔排序的算法分析如下：

（1）增量序列的选择。

Shell 排序的执行时间依赖于增量序列。好的增量序列的共同特征如下：

① 最后一个增量必须为 1。

② 应该尽量避免序列中的值(尤其是相邻的值)互为倍数的情况。

有人通过大量的实验给出了目前较好的结果：当 n 较大时，比较和移动的次数约在 $n^{1.25}$ ~ $1.6n^{1.25}$ 之间。

（2）Shell 排序的时间性能优于直接插入排序。

希尔排序的时间性能优于直接插入排序的原因如下：

① 当文件初态基本有序时，直接插入排序所需的比较和移动次数均较少。

② 当 n 值较小时，n 和 n2 的差别也较小，即直接插入排序的最好时间复杂度 O(n)和最坏时间复杂度 0(n2)差别不大。

③ 在希尔排序开始时增量较大，分组较多，每组的记录数目少，故各组内直接插入较快，后来增量 d_i 逐渐缩小，分组数逐渐减少，而各组的记录数目逐渐增多，但由于已经按 d_{i-1} 作为距离排过序，使文件较接近于有序状态，所以新的一趟排序过程也较快。因此，希尔排序在效率上较直接插入排序有较大的改进。

（3）稳定性。

希尔排序是不稳定的。参见上述实例，该例中两个相同关键字 49 在排序前后的相对次序发生了变化。

11.4.5 二分排序

面试例题 1：下面的程序是哪种排序？

```
public class Test{
 public  static  void  main(String[]  args)  {
     int [] a = {4, 2, 1, 6, 3, 6, 0, -5, 1, 1 };
    int i,j;
    int low,high,mid;
    int temp;
    for(i=1;i <10;i++)
    {
       temp=a[i];
       low=0;
       high=i-1;
       while(low <=high)
       {
          mid=(low+high)/2;
          if(a[mid]>temp)
             high=mid-1;
          else
             low=mid+1;
       }
       for(j=i-1;j>high;j--)
          a[j+1]=a[j];
       a[high+1]=temp;
    }

    for(i=0;i <10;i++)
     System.out.printf("%d  ",a[i]);
 }
}
```

A．归并排序　　B．快速排序　　C．希尔排序　　D．二分法排序

解析：从二分字面上理解的话，快速排序和归并排序都与二分相关：快速排序按照标值二分，小的在前，大的在后；而归并则是按照下标二分，再分别对两个部分归并排序，先分后和，在和的过程中排序。

此外，还有一种二叉树排序，它的原理是：构造二叉树，小的在左，大的在右，将待排序的数依次插入，然后前序遍历即可，这种方法利用二叉树的特殊构造把每个数插入到恰当的位置，跟二分的概念略有关系。

目前比较公认的二分法排序是指折半插入排序。当直接插入排序进行到某一趟时，对于 r[i].key 来讲，前边 i-1 个记录已经按关键字有序。此时不用直接插入排序的方法，而改为二分折半查找，找出 r[i].key 应插的位置，然后插入。这种方法就是折半插入排序（二分法排序）。二分法排序中，关键字的比较次数由于采用了折半查找而减少，数量级为 O(n log n)，但是元素移动次数仍为 $O(n^2)$ 。故折半插入排序时间复杂度仍为 $O(n^2)$ 。二分法排序方法是稳定的。

答案：D

11.4.6　快速排序

面试例题 1：The following is an improved Java quick sort algorithm. Please fill in the blank.（下面的程序是一个 Java 快速排序问题，请填空。）

```
public class Test{
public static void qsort_asc(int source[], int low, int high) {
        int i, j, x;
        if (low < high) {
            i = low;
            j = high;
            x = source[i];
            while (i < j) {
                while (i < j && source[j] > x) {
                    j--;
                }
                if (i < j) {
                    source[i] = source[j];
                    i++;
                }
                while (i < j && source[i] < x) {
                    i++;
                }
                if (i < j) {
```

```
                source[j] = source[i];
                j--;
            }
        }
        source[i] = x;
        qsort_asc(_____);
        qsort_asc(_____);
    }
}

public static void main(String[] args) {
    int [] a = {4, 2, 1, 6, 3, 6, 0, -5, 1, 1 };
    int i;
    qsort_asc(a, 0, a.length-1);

    for(i=0;i <10;i++)
     System.out.printf("%d ",a[i]);
}
}
```

解析：本题考查数据结构的快速排序问题。

答案：

qsort_asc(source, low, i - 1);

qsort_asc(source, i + 1, high);

扩展知识：快速排序的算法思想

快速排序是 C.R.A.Hoare 于 1962 年提出的一种划分交换排序。它采用了一种分治的策略，通常称其为分治法（Divide-and-ConquerMethod）。

（1）分治法的基本思想。

分治法的基本思想是：将原问题分解为若干个规模更小但结构与原问题相似的子问题。递归地解这些子问题，然后将这些子问题的解组合为原问题的解。

（2）快速排序的基本思想。

设当前待排序的无序区为 R[low..high]，利用分治法可将快速排序的基本思想描述为：

① 分解。在 R[low..high]中任选一个记录作为基准（Pivot），以此基准将当前无序区划分为左、右两个较小的子区间 R[low..pivotpos-1]和 R[pivotpos+ 1..high]，并使左边子区间中所有记录的关键字均小于或等于基准记录（不妨记为 pivot）的关键字 pivot.key，右边子区间中所有记录的关键字均大于或等于 pivot.key，而基准记录 pivot 则位于正确的位置（pivotpos）上，它无须参加后续的排序。

注意：划分的关键是要求出基准记录所在的位置 pivotpos。划分的结果可以简单地表示为（注意 pivot=R[pivotpos]）：

R[low..pivotpos-1].keys ≤ R[pivotpos].key ≤ R[pivotpos+1..high].keys

其中 low ≤ pivotpos ≤ high。

② 求解。通过递归调用快速排序对左、右子区间 R[low..pivotpos-1]和 R[pivotpos+1..high] 快速排序。

③ 组合。因为当“求解”步骤中的两个递归调用结束时，其左、右两个子区间已有序。对快速排序而言，“组合”步骤无须做什么，可看做是空操作。

快速排序算法 QuickSort

```
void QuickSort(SeqList R , int low , int high)
 { // 对 R[low..high] 快速排序
   int pivotpos ; // 划分后的基准记录的位置
   if(low<high){// 仅当区间长度大于 1 时才需排序
      pivotpos=Partition(R , low , high) ;    // 对 R[low..high] 做划分
      QuickSort(R , low , pivotpos-1) ;       // 对左区间递归排序
      QuickSort(R , pivotpos+1 , high) ;      // 对右区间递归排序
    }
 } //QuickSort
```

注意：

只需调用 QuickSort(R，1，n)，即可完成对 R[l..n]的排序。

划分算法 Partition

（1）简单的划分方法。

① 具体做法。

第一步：（初始化）设置两个指针 i 和 j，它们的初值分别为区间的下界和上界，即 i=low，i=high。选取无序区的第一个记录 R[i](即 R[low])作为基准记录，并将它保存在变量 pivot 中。

第二步：令 j 自 high 起向左扫描，直到找到第 1 个关键字小于 pivot.key 的记录 R[j]，将 R[j])移至 i 所指的位置上，这相当于 R[j]和基准 R[i](即 pivot)进行了交换，使关键字小于基准关键字 pivot.key 的记录移到了基准的左边，交换后，R[j]中相当于 pivot；然后令 i 指针自 i+1 位置开始向右扫描，直至找到第 1 个关键字大于 pivot.key 的记录 R[i]，将 R[i]移到 i 所指的位置上，这相当于交换了 R[i]和基准 R[j]，使关键字大于基准关键字的记录移到了基准的右边，交换后，R[i]中又相当于存放了 pivot；接着令指针 j 自位置 j-1 开始向左扫描，如此交替改变扫描方向，从两端各自往中间靠拢，直至 i=j 时，i 便是基准 pivot 最终的位置，将 pivot 放在此位置上就完成了一次划分。

② 划分算法。

```
int Partition(SeqList R, int i, int j)
 {// 调用 Partition(R, low, high)时，对 R[low..high] 做划分，
  // 并返回基准记录的位置
   ReceType pivot=R[i] ;  // 用区间的第 1 个记录作为基准
   while(i<j){            // 从区间两端交替向中间扫描，直至 i=j 为止
     while(i<j&&R[j].key>=pivot.key)    //pivot 相当于在位置 i 上
       j-- ;  //从右向左扫描，查找第 1 个关键字小于 pivot.key 的记录 R[j]
     if(i<j) // 表示找到 R[j] 的关键字 <pivot.key
        R[i++]=R[j] ;      // 相当于交换 R[i] 和 R[j] ，交换后 i 指针加 1
     while(i<j&&R[i].key<=pivot.key)    //pivot 相当于在位置 j 上
        i++ ;  // 从左向右扫描，查找第 1 个关键字大于 pivot.key 的记录 R[i]
     if(i<j)   // 表示找到了 R[i] ，使 R[i].key>pivot.key
        R[j--]=R[i];       // 相当于交换 R[i] 和 R[j] ，交换后 j 指针减 1
    } //endwhile
   R[i]=pivot ;            // 基准记录已被最后定位
   return i ;
 } //partition
```

快速排序执行过程

快速排序执行的全过程可用递归树来描述，如图 11-5～图 11-7 所示。

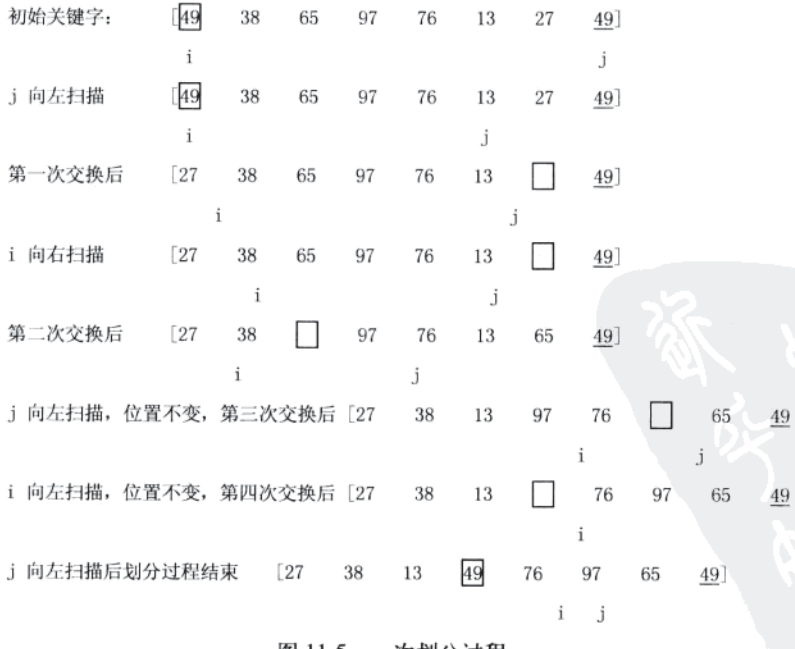


图 11-5 一次划分过程

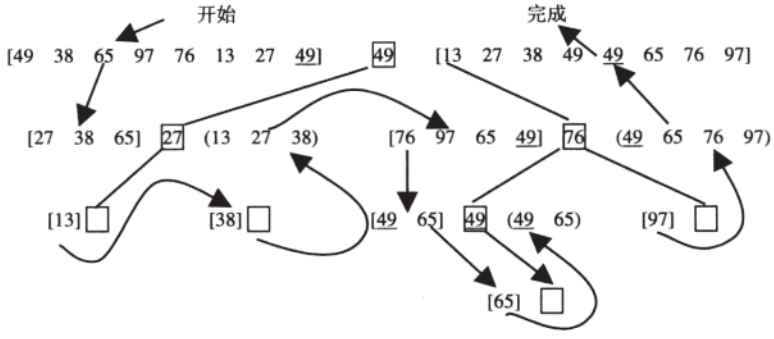


图 11-6　QuickSort 执行时的递归树

[49	38	65	97	76	13	27	49]	//初始关键字
[27	38	13]	49	[76	97	65	49]	//第一次划分完成后，对应递归树第 2 层
[13]	27	[38]	49	[49	65]	76	[97]	//对上一层个无序区划分完成后，对应递归树第 3 层
13	27	38	49	49	[65]	76	97	//对上一层个无序区划分完成后，对应递归树第 4 层
13	27	38	49	49	65	76	97	//最后的排序结果

图 11-7　对递归树的每层上个无序区划分之后的状态

时间复杂度

快速排序法是一种不稳定的排序方法。其平均时间复杂度为 O(n*lg n/lg 2)；最差情况下时间复杂度为 O(n^2)。

11.4.7　归并排序

面试例题 1：请用 Java 写出一个归并排序程序，要求输入 10 个整数，输出排序结果。

解析：归并排序指的是将两个已经排序的序列合并成一个序列的操作。归并排序具体工作的原理如下（假设序列共有 n 个元素）：

① 将序列每相邻的两个数字进行归并操作(merge)，形成 floor(n / 2)个序列，排序后每个序列包含两个元素。

② 将上述序列再次归并，形成 floor(n / 4)个序列，每个序列包含四个元素。

③ 重复步骤②，直到所有的元素排序完毕。

答案：完整代码如下：

```
public class Test{
```

```
public static void merge(int array[],int start1,int end1,int start2,int end2){
      int i,j;//i,j 分别为表 1 和表 2 的游标
      {
       i = start1;
       j = start2;
      }
      int k=0;int []temp = new int[end2-start1+1]; //建立一个临时长度为两个子列表长度
                                                    //之和的数组
      while(i<=end1&&j<=end2)          //通过循环，依次从两个子列表中找出较大元素放入临时数组中
      {
       if(array[i]>array[j])
        temp[k++]=array[j++];
       else
        temp[k++]=array[i++];
      }
      //把剩下的元素依次放入临时数组中（肯定是只剩下一方）
      while(i<=end1)
       temp[k++]=array[i++];
      while(j<=end2)
       temp[k++]=array[j++];
      k=start1;
      for(int element: temp){//把临时数组元素复制给原数组
       array[k++]=element;
      }

     }

 public static void mergeSort(int array[],int start,int end)
 {
  if(start<end)
  {
   int mid = (start+end)/2;
   //两路归并
   mergeSort(array,start,mid);
   mergeSort(array,mid+1,end);
   merge(array,start,mid,mid+1,end);

   //多路归并
   /*
   int mid = (start+end)/4;
   mergeSort(array,start,1*mid);
   mergeSort(array,1*mid+1,2*mid);
   mergeSort(array,2*mid+1,3*mid);
   mergeSort(array,3*mid+1,end);
   merge(array,start,1*mid,1*mid+1,2*mid);
   merge(array,2*mid+1,3*mid,3*mid+1,end);
```

```
    merge(array,start,2*mid,2*mid+1,end);
    */
   }
  }

 public  static  void  main(String[]  args)  {
      int [] a = {4, 2, 1, 6, 3, 6, 0, -5, 1, 1 };
    int i;
    mergeSort(a, 0, a.length-1);

    for(i=0;i <10;i++)
     System.out.printf("%d  ",a[i]);
 }
}
```

扩展知识：

假设我们有一个没有排好序的序列，首先我们使用分割的方法将这个序列分割成一个一个已经排好序的子序列。然后利用归并的方法将一个个的子序列合并成排序好的序列。分割和归并的过程示意图如图 11-8 所示。

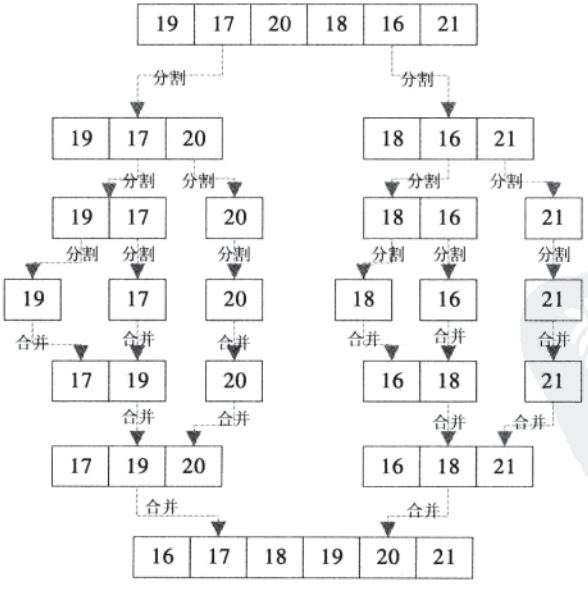


图 11-8　分割和归并的过程示意图

由图 11-8 得知：我们首先把一个未排序的序列从中间分割成 2 部分，再把

2 部分分成 4 部分，依次分割下去，直到分割成一个一个的数据，再把这些数据两两归并到一起，使之有序，不停地归并，最后成为一个排好序的序列。

如何把两个已经排序好的子序列归并成一个排好序的序列呢？可以参看下面的方法。假设我们有两个已经排序好的子序列：

```
序列A：17，19，20
序列B：16，18，21
```

那么可以按照下面的步骤将它们归并到一个序列中。

```
(1) 设定一个新的数列C[6]。
(2) A[0]和B[0]比较，A[0] = 17，B[0] = 16，A[0] > B[0]，那么C[0] = 16
(3) A[0]和B[1]比较，A[0] = 17，B[1] = 18，A[0] < B[1]，那么C[1] = 17
(4) A[1]和B[1]比较，A[1] = 19，B[1] = 18，A[1] > B[1]，那么C[2] = 18
(5) A[1]和B[2]比较，A[1] = 19，B[2] = 21，A[1] < B[2]，那么C[3] = 19
(6) A[2]和B[2]比较，A[2] = 20，B[2] = 21，A[2] < B[2]，那么C[4] = 20
(7) 最后将B[2]复制到C中，那么C[5] = 21，归并完成。
```

归并排序算法是一种 O(nlogn)的算法。它的最差时间复杂度、平均时间复杂度、最好时间都是 O(nlogn)。但是它需要额外的存储空间，这在某些内存紧张的机器上会受到限制。归并算法是由分割和归并两部分组成的，对于分割部分，如果我们使用二分查找，时间是 O(nlogn)，在最后归并的时候，时间是 O(n)，所以总的时间是 O(nlogn)。

第12章

字符串、数组、范型

求职者在进行笔试时，几乎没有不考字符串、数组和范型的。字符串也是一种相对简单的数据结构，结合指针，容易多次引起面试官反复发问。笔者曾不止一次在笔试或面试时遇到字符串的试题。事实上，字符串也是一个考验程序员编程规范和编程习惯的重要考点。范型是 Java SE 1.5 的新特型，泛型的本质是参数化类型，也就是说，所操作的数据类型被指定为一个参数。这种参数类型可用在类、接口和方法的创建中，分别称为范型类、泛型接口、泛型方法。Java 语言引入范型的好处是安全简单。Java 范型编程也是 Java 程序员面试的热点之一。大家不能忽视这些细节，因为这些细节会体现你在操作系统、软件工程、边界内存处理等方面的知识掌控能力。

12.1 字符串基础问题

面试例题 1：下面代码的输出结果是什么？[中国著名软件培训公司 B2009 年 3 月面试题]

```
public class Test {
 public static void main(String[] args) {

      String s1 = "abc";
      String s2 = s1;
      String s5 = "abc";
      String s3 = new String("abc");
      String s4 = new String("abc");
      System.out.println("s1 == s5: " + (s1==s5));
      System.out.println("s1 == s2: " + (s1==s2));
      System.out.println("s1.equals(s2): " + s1.equals(s2));
      System.out.println("s3 == s4: " + (s3==s4));
```

```
        System.out.println("s1.equals(s4): " + s1.equals(s4));
        System.out.println("s3.equals(s4): " + s3.equals(s4));
    }
}
```

解析：“==”判断符号左右两个变量（Object）是否指向同一内存地址；“equals()”判断两个 object 是否“一样”（所有成员的值相同）。

例 1：

```
string a = "abc";
string b = "abc";
a==b //true;
a.equals(b) //true;
```

例 2：

```
string a = new string("abc");
string b = new string("abc");
a==b // false!!!!!!!
a.equals(b) //true
```

在例 1 中，"abc"是放在常量池（constant pool）中的，所以，虽然 a、b 都等于"abc"，但是内存中只有一份副本，所以“==”返回 true。

在例 2 中，new 方法决定了两个不同的 string"abc"被创建放在了内存 heap 区，分别被 a 和 b 所指向，因此，“==”返回了 false。

答案：输出结果如下：

```
s1 == s5: true
s1 == s2: true
s1.equals(s2): true
s3 == s4: false
s1.equals(s4): true
s3.equals(s4): true
```

面试例题 2：下面语句共创建了几个对象？[中国著名软件培训公司 B2009 年 3 月面试题]

```
String s = "a" + "b" + "c" + "d" + "e";
```

A．没有创建对象　　B．1 个对象　　C．2 个对象　　D．3 个对象

解析：就创建了 1 个对象。

String s = "a" + "b" + "c" + "d" + "e"; 语句中，赋值符号右边的"a"、"b"、"c"、"d"、"e"都是常量，对于常量，编译时就直接存储它们的字面值，而不是它们的引用，在编译时就直接将它们连接的结果提取出来变成了"abcde"。

该语句在 class 文件中就相当于 String s = "abcde"。然后当 JVM 执行到这一句时，就在 String pool 里找，如果没有这个字符串，就会产生 1 个对象。可以对比源代码和反编译代码如下：

```
public class a {
  public static void main(String[] args) {
       String s = "a" + "b" + "c" + "d" + "e";
       System.out.println(s);
  }
}
//反编译如下：
Java code
Compiled from "a.java"
public class a extends java.lang.Object{
public a();
  Code:
  0: aload_0
  1: invokespecial #1;   //Method java/lang/Object."<init>":()V
  4: return
public static void main(java.lang.String[]);
  Code:
  0: ldc #2;             //String abcde 直接是 abcde
  2: astore_1
  3: getstatic #3;       //Field java/lang/System.out:Ljava/io/PrintStream;
  6: aload_1
  7: invokevirtual #4;   //Method java/io/PrintStream.println:(Ljava/lang/String;)V
  10: return
}
```

String 的内部结构是通过 StringBuilder 实现的。所以，没有生成“ab”、“abc”等对象 s 引用在堆栈里，肯定不是对象，所以只创建了一个对象“abcde”。

答案：B

扩展知识 1：Constant Pool 常量池的概念

在 Java 编译好的 class 文件中，有个区域称为 Constant Pool，它是一个由数组组成的表，类型为 cp_info constant_pool[]，用来存储程序中使用的各种常量，包括 Class、String、Integer 等各种基本的 Java 数据类型。

String Pool 是对应于在 Constant Pool 中存储 String 常量的区域。习惯性地称它为 String Pool。

对于 Constant Pool，表的基本通用结构如下：

```
  cp_info {
```

```
        u1 tag;
        u1 info[];
}
```

tag 是一个数字，用来表示存储的常量类型。例如 8 表示 String 类型，5 表示 Long 类型，info[]根据类型码 tag 的不同会发生相应的变化。

对于 String 类型，表的结构为：

```
CONSTANT_String_info {
        u1 tag;
        u2 string_index;
}
```

tag 固定为 8 时，string_index 是字符串内容信息，其类型为：

```
CONSTANT_Utf8_info {
        u1 tag;
        u2 length;
        u1 bytes[length];
}
```

tag 固定为 1 时，length 为字符串的长度，bytes[length]为字符串的内容。

为了详细理解常量池的结构，看如下代码：

```
public class Test {
   public static void main(String[] args) {
      String s1 = "aaa111";
      String s2 = "aaa111";
   }
}
```

在上面程序中存在 2 个相同的常量"aaa111"，对于值相同的 String 常量，在 Constant Pool 中只会创建一个，所以在编译好的 class 文件中,我们只能找到一个对"aaa111"的表示。

在程序执行的时候，Constant Pool 会储存在 Method Area 中，而不是堆中。另外，对于" "内容为空的字符串常量，会创建一个长度为 0、内容为空的字符串放到 Constant Pool 中，而且 Constant Pool 在运行期是可以动态扩展的。

扩展知识 2：关于 String 类的说明

（1）String 使用 private final char value[]来实现字符串的存储，也就是说，String 对象创建之后，就不能再修改此对象中存储的字符串内容，就是因为如此，才说 String 类型是不可变的（immutable）。

（2）String 类有一个特殊的创建方法，就是使用" "双引号来创建。例如 new String("jinder")实际创建了 2 个 String 对象，一个是"jinder"通过" "双引号创建的，另一个是通过 new 创建的。只不过它们创建的时期不同，一个是编译期，一个是运行期。

（3）Java 对 String 类型重载了加号（+）操作符，可以直接使用“+”对两个字符串进行连接。

（4）运行期调用 String 类的 intern()方法可以向 String Pool 中动态添加对象。

（5）String 的创建方法一般有如下几种：

① 直接使用" "引号创建。

② 使用 new String()创建。

③ 使用 new String("someString")创建以及其他的一些重载构造函数创建。

④ 使用重载的字符串连接操作符“+”创建。

下面看四个例子：

例 1：

```
String s1 = "aaa777";
String s2 = "aaa777";
System.out.println(s1 == s2); //结果为 true
```

"aaa777"是编译期常量，编译时已经能确定它的值，在编译好的 class 文件中，它已经在 String Pool 中了，此语句会在 String Pool 中查找等于"aaa777"的字符串（用 equals(Object)方法确定），如果存在，就把引用返回，赋值给 s1；若不存在，就会创建一个"aaa777"放在 String Pool 中，然后把引用返回，赋值给 s1。

由于 String Pool 只会维护一个值相同的 String 对象，上面两句得到的引用是 String Pool 中的同一个对象，所以它们引用相等。

例 2：

```
String s1 = new String("aaa777");
String s2 = "aaa777";
System.out.println(s1 == s2); //结果为 false
```

在 Java 中，使用 new 关键字会创建一个新对象，在本例中，不管在 String Pool 中是否已经有值相同的对象，都会创建一个新的 String 对象存储在 heap 中，然后把引用返回，赋值给 s1。本例中使用了 String 的 public String(String original)构造函数。

由于 s1 是用 new 创建出的新对象，存储在 heap 中，s2 指向的对象存储在 String Pool 中，它们肯定不是同一个对象，只是存储的字符串值相同，所以返回 false。

例 3:

```
String s1 = new String("aaa777");
s1 = s1.intern();
String s2 = "aaa777";
System.out.println(s1 == s2);
```

当调用 intern 方法时，如果 String Pool 中已经包含一个等于此 String 对象的字符串（用 equals(Object)方法确定），则返回池中的字符串，否则将此 String 对象添加到池中，并返回此 String 对象在 String Pool 中的引用。由于执行了 s1 = s1.intern()，会使 s1 指向 String Pool 中值为"aaa777"的字符串对象，s2 也指向了同样的对象，所以结果为 true。

例 4:

```
String s1 = new String("777");
String s2 = "aaa777";
String s3 = "aaa" + "777";
String s4 = "aaa" + s1;
System.out.println(s2 == s3); //true
System.out.println(s2 == s4); //false
System.out.println(s2 == s4.intern()); //true
```

由于进行连接的两个字符串都是常量，编译期就能确定连接后的值了，编译器会进行优化，直接把它们表示成"aaa777"存储到 String Pool 中， 由于上边的 s2="aaa777"已经在 String Pool 中加入了"aaa777"，此句会把 s3 指向和 s2 相同的对象，所以它们引用相同。此时仍然会创建出 "aaa"和"777"两个常量，存储到 String Pool 中。

由于 s1 是变量，在编译期不能确定它的值是多少，所以会在执行的时候创建一个新的 String 对象存储到 heap 中，然后赋值给 s4。

面试例题 3：下列程序的输出结果是多少？[美国某著名 CPU 生产公司 A2008 年 11 月笔试题]

```
class Test{
 public static void main(String args[]){
      String str = "ABCDEFGH";
      String str1 = str.substring(3, 5);
      System.out.println(str1);
 }
}
```

A．CD　　B．CDE　　C．DE　　D．DEF

解析：很多人对该题会想当然地给出 DEF 的答案，但是一定要注意 Java 中的 substring 是前包括后不包括的。所以是 DE。

答案：C

面试例题 4：如下代码编译后的输出结果是多少？[中国著名金融软件公司 2005 年面试题]

```
import java.util.*;
public class  Test3
{

 public static void main(String[] args) {
      String s = "hello";
      String t = "hello";
      char c[] = {'h','e','l','l','o'} ;
      if(s.equals(t))
           System.out.println("true");
      else
           System.out.println("false");

      if(t.equals(c))
           System.out.println("true");
      else
           System.out.println("false");

      if(t.equals(new String("hello")))
           System.out.println("true");
      else
           System.out.println("false");

      if(s==t)
           System.out.println("true");
      else
           System.out.println("false");

   }
}
```

解析：检测两个字符串是否相同用到 equals 函数。如果两个是栈内的字符串，直接使用“==”比较也是可以的。char 和 string 是两种不同的类型，所以比较结果不同。

答案：

true

false

true

true

扩展知识（Java 中 string 与 char 如何转换）

代码如下。

```
import java.util.*;
public class  Test3
{

  public static void main(String[] args) {
       String s = "hello";
       String t = "hello";
       String str = "hello";
       char c[] = {'h','e','l','l','o'} ;

       char  ch[]=str.toCharArray();    //string 转换成 char

       if(ch==c)
            System.out.println("true");
       else
            System.out.println("false");

       if(ch.equals(c))
            System.out.println("true");
       else
            System.out.println("false");

       String  s2  =  new  String(c);    //char 转换成 string

       if(s2==(s))
            System.out.println("true");
       else
            System.out.println("false");

       if(s2.equals(s))
            System.out.println("true");
       else
            System.out.println("false");


   }
}
```

12.2　StringBuffer

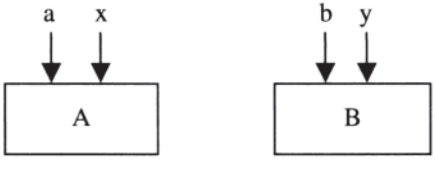
面试例题 1：对于如下代码，以下哪个结论是正确的？[中国著名金融软件公司 2005，2009 年面试题]

```
//import java.lang.*;
public class Foo
{

 public static void main(String[] args)
 {
      StringBuffer a = new StringBuffer("A");
      StringBuffer b = new StringBuffer("B");
      operate(a,b);
      System.out.println(a+","+b);
 }

 static void operate(StringBuffer x,StringBuffer y)
 {
      x.append(y);
      y=x;
 }

}
```

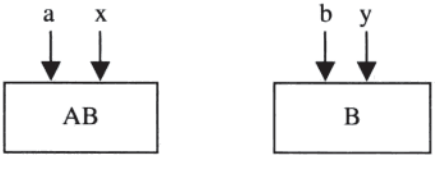
A．The code compiles and prints "A,B".　　B．The code compiles and prints "A,A".

C．The code compiles and prints "B,B".　　D．The code compiles and prints "AB,B".

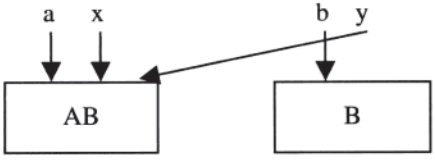
解析：operate 函数传入两个 StringBuffer 类型的句柄的副本，一个是 x，一个是 y，分别指向 A、B。



执行“x.append(y);”后，AB 附加在一起，而指针 x、y 没有发生变化。



执行“y=x;”后，y 指向 AB。



函数结束后，x、y 自然消亡。a 指向的是 AB，b 指向的是 B。

答案：D

扩展知识（String 与 StringBuffer 的区别）

1．String 的创建

String s = "hello";

JVM 先根据内容“hello”查找对象，如果没有找到，则在 heap 上创建新对象，并将其赋予 s1，否则使用已经存在的对象。

String s = new String("hello");

JVM 直接在 heap 上创建新的对象，所以在 heap 中会出现内容相同而地址不同的 String 对象。

2．String 的比较

“==”是比较地址，“equals”是比较内容。

举例如下。

```
String s1 = "hello";
String s2 = "hello";
String s3 = new String("hello");

s1 == s2;            // true   地址相同
s1 == s3;            // false  地址不同
s1.equals(s2);       // true   内容相同
s1.equals(s3);       // true   内容相同
```

3．intern()方法

查找内容相同（equals()）的字符串，举例如下。

```
String s1 = "hello"; // hello 不存在，JVM 创建新对象 (1)
String s2 = new String("hello");
// 创建新对象 (2)，这时 heap 中存在两个内容为 hello 的对象
s1 == s2;            // false // 地址不同
s1.equals(s2);       // true  // 内容相同
```

```
s2 = s2.intern();     // true   // 找到对象(1) 并赋予 s2
s1 == s2;             // true   // 注意：此时 s1,s2 同指向(1)
```

4．效率比较：String 与 StringBuffer

例 1:

（1）String result = "hello" + " world";

（2）StringBuffer result = new String().append("hello"). append(" world");

（1）的效率好于（2），这是因为 JVM 会做如下处理:

- 将 result 字符串做 “"hello" + " world";” 处理，然后才赋值给 result。只开辟了一次内存段。
- 编译 StringBuffer 后还要做 append 处理，花的时间要长一些。

例 2:

（1）public String getString(String s1, String s2) {
　　return s1 + s2;
　}

（2）public String getString(String s1, String s2) {
　　return new StringBuffer().append(s1).append(s2);
　}

（1）的效率与（2）一样，这是因为 JVM 会做如下处理:

开辟一个内存段，再合并（扩展）内存，所以两者执行的过程是一致的，效率相当。

例 3:

（1）String s = "s1";
　　s += "s2";
　　s += "s3";

（2）StringBuffer s = new StringBuffer().append("s1"). append("s2").append("s3");

（1）的效率好于（2），因为 String 是不可变对象，每次 “+=” 操作都会构造新的 String 对象。

例 4:

（1）StringBuffer s = new StringBuffer();
　for (int i = 0; i < 50000; i ++) {
　　s.append("hello");
　}

（2）StringBuffer s = new StringBuffer(250000);

```
for (int i = 0; i < 50000; i ++) {
    s.append("hello");
}
```

（1）的效率好于（2），因为 StringBuffer 内部实现的是 char 数组，默认初始化长度为 16，每当字符串长度大于 char 数组长度的时候，JVM 会构造更大的新数组，并将原先的数组内容复制到新数组。（2）避免了复制数组的开销。

StringBuffer 面试关键点：

（1）简单地认为.append()效率好于“+”是错误的。

（2）不要使用 new 创建 String。

（3）注意.intern()的使用。

（4）在编译期能够确定字符串值的情况下，使用“+”效率最高。

（5）避免使用“+=”来构造字符串。

（6）在声明 StringBuffer 对象的时候，指定合适的 capacity，不要使用默认值（18）。

（7）注意以下二者的区别，后者开辟了两个内存段。

- String s = "a" + "b";
- String s = "a";
  s += "b";

面试例题 2：请说出下面代码块存在的问题。

```
String tmp = "";

for(int i=0;i<9999;tmp +="x")
{
}
```

解析：String 是一个支持非可变性的类，这种类的特点是状态固定（不存在任何修改对象的方法）。在该对象的生存周期内，它的值是永远不变的（它是线程安全的），更容易设计、实现和使用，不易出错，更加安全。

由于 String 类是支持非可变性的，所以，当执行 tmp += "x"的时候，实际上是另外创建了一个对象，而 tmp 原来指向的那个对象就成了垃圾（当它没有其他引用的时候），这样，一个循环就会产生 n 个对象，从而造成内存的浪费。

先看一下 String 类中 substring 方法的实现。

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

```
        if (beginIndex < 0) {

            throw new StringIndexOutOfBoundsException(begin-
              Index);
        }

        if (endIndex > count) {

            throw new StringIndexOutOfBoundsException(endInd-
              ex);
        }

        if (beginIndex > endIndex) {

            throw new StringIndexOutOfBoundsException(endIn-
              dex - beginIndex);
        }

        return ((beginIndex == 0) && (endIndex == count)) ?
            this :

            new String(offset + beginIndex, endIndex - beginIndex, val-
              ue);
    }
```

它重新创建了一个新的对象，符合“支持非可变性”的原则，但这也显示出了非可变类的真正唯一的缺点，就是“对于每一个不同的值都要求一个单独的对象”。

那为什么 String 要设计成非可变类呢？String 是 Java 中使用最频繁的一个类，只有使其支持非可变性，才有可能避免一系列的问题。例如：

```
String a,b,c;
a="test";
b=a;
c=b;
processA(){
    …
}
ProcessB(){
    …
}
ProcessC(){
    …
}
```

当 String 支持非可变性的时候，它们的值很好确定，不管调用哪种方法，都互不影响。如果它不支持非可变性，则无法想象其结果。

StringBuffer 作为 String 的“配套类（companying class）”，它的作用就是为了解决上述问题的，StringBuffer 扮演了 String 的可变配套类角色。非可变类本质上是线程安全的，它们不要求做同步处理，可以将其共享给所有的用户，让所有的“客户端程序员”都可以直接使用此类而不需要做任何额外的工作。

答案：修改成可变类 StringBuffer。

面试例题 3：以下程序创建了几个对象？

```
String A,B,C
A="a";
B="b";
A=A + B;
StringBuffer D=new StringBuffer("abc");
D = D.append("567");
```

A．4　　B．3　　C．5　　D．6

解析：

要知道 String s = new String("abc")创建了几个 String Object，首先必须了解引用变量与对象的区别。

（1）引用变量与对象。除了一些早期的 Java 书籍，我们都可以从书中比较清楚地学习到两者的区别。“A aa;”语句声明一个类 A 的引用变量 aa（常称为句柄），而对象一般通过 new 创建。所以题目中 D 仅仅是一个引用变量，它不是对象。而字符串文字“abc”是一个 String 对象。

（2）Java 中所有的字符串文字（字符串常量）都是一个 String 的对象。有人（特别是 C 程序员）在一些场合喜欢把字符串当做字符数组，这也没有办法，因为字符串与字符数组存在一些内在的联系。事实上，它与字符数组是两种完全不同的对象。

```
System.out.println("Hello".length());
char[] cc={'H','i'};
System.out.println(cc.length);
```

（3）字符串对象的创建。由于字符串对象的大量使用（它是一个对象，一般而言，对象总在 heap 分配内存），Java 中为了节省内存空间和运行时间（如比较字符串时，“==”比 equals()快），在编译阶段就把所有的字符串文字放到一个文字池（pool of literal strings）中，而运行时文字池成为常量池的一部分。文字池的好处就是该池中所有相同的字符串常量被合并，只占用一个空间。对两个引用变量使用“==”判断它们的值（引用）是否相等，即指向同一个对象。

```
String s1 = "abc" ;
String s2 = "abc" ;
if( s1 == s2 ) System.out.println("s1,s2 refer to the same object");
else   System.out.println("trouble");
```

这里的输出显示两个字符串文字保存为一个对象。就是说，上面的代码只在 pool 中创建了一个 String 对象。

现在看“String s = new String("abc");”语句，这里“abc”本身就是 pool 中的一个对象，而在运行时执行 new String()时，将 pool 中的对象复制一份放到 heap 中，并且把 heap 中的这个对象的引用交给 s 持有。这条语句就创建了两个 String 对象。

```
String s1 = new String("abc") ;
String s2 = new String("abc") ;
if( s1 == s2 ){ //不会执行的语句}
```

这时用“==”判断就可知，虽然两个对象的“内容”相同（用 equals()判断），但两个引用变量所持有的引用不同。上面的代码创建了 3 个 String Object，pool 中一个，heap 中两个。

下面对题目中的代码一一分析。

（1）“StringBuffer D = new StringBuffer("abc");”产生了两个对象，“abc”本身与经过 new 创建出来的不是一个对象，可以用“==”来检验。

（2）对本题：

“A = "a";”，引用的不是对象，此处创建了一个对象和一个引用 A；

“B = "b";”，说明同上；

“A = A + B;”，说明同上，此处创建了一个对象，并由引用 A 来引用，那么原来 A 所指向的对象就成为垃圾对象，被回收；

“StringBuffer D = new StringBuffer("abc");”，StringBuffer 的特点是改变对象本身而不是创建新的对象，因此，此处及“D = D.append("567");”都是对同一个对象进行处理。而字符串对象也是一个对象，故有两个对象。

结论：若讨论整个过程共产生了多少个对象，可以得出是 5 个。

答案：C

12.3　正则表达式

面试例题 1：String s = "32fdsfd8fds0fdsf9323k32k"，从中找出 3280932332，你会怎么做？[中国香港某著名门户网站 F2005 年 9 月面试题]

解析：可以使用正则表达式。“[^0-9]”是正则表达式，s.replaceAll ("[^0-9]", "")的意思是将 s 中所有非 0～9 的字符替换为空串。

答案：代码如下。

```
public class GetNumber1 {
 static String s = "32fdsfd8fds0fdsf9323k32k";

 /**
  * @param args
  */
 public static void main(String[] args) {
      String a = s.replaceAll("[^0-9]", "");
      System.out.print(a);
 }

}
```

面试例题 2：String str="2006-04-15 02:31:04"，要把这个串变成 20060415023104，你会怎么做？[中国台湾著名杀毒软件公司 T2005 年 9 月面试题]

解析：本题考核的是对 Java 正则表达式的理解。读者可能会问，“不用正则表达式就不能解决此问题了吗？”当然也可以采用下述方法。

```
public static void main(String[] args) throws Exception{
String str ="2006-04-15 02:31:04";
str=str.replaceAll("-","");
str=str.replaceAll(":","");
str=str.replaceAll(" ","");
System.out.println(str);
}
```

不过这没有技术含量，面试分不会太高。这只是在考相关的知识掌握有多深，replaceAll 是最浅的。会用正则表达式给招聘单位的印象会较好。

答案：可以采用正则表达式，代码如下。

```
class  Test3
{
   public static void main(String[] args)
   {
      String str ="2006-04-15 02:31:04";
      String str2 ="";
      String []result=str.split("\\D");
      for(int i=0;i<result.length;i++)
         {
```

```
            System.out.print(result[i]);
            str2+=result[i];
           }
        System.out.println(str2);
    }
}
```

12.4　数字流和数组

面试例题 1：Which of the following are valid array declaration for strings of 50 chars?（对于 50 个字符的字符串数组，下面哪个声明是正确的？）[中国台湾著名杀毒软件公司 T2005 年 9 月面试题]

A．char c[][];　　B．String []s;　　C．String s[50];　　D．Object s[50];

解析：本题涉及数组定义问题。如下声明方法：

```
int [][] iArray;
int []iArray[];
int iArray[][];
```

都是可以的。但是要注意，数组不能直接指定列数或者行数，比如 int iArray[3][4]。正确的定义数组的行数和列数应该是在创建数组对象时，如 int iArray[][] = new int[3][4]。

也可以这样：

```
int iArray[][] = { {1,2,3},{2,2,4}};
```

再就是二维数组可以列数不同。

例如：

```
int iArray[][] = new int[2][];
iArray[0] = new int [3];
iArray[1] = new int [4];
```

数组不能直接指定列数或者行数，所以选项 C 和选项 D 是错误的。选项 A 声明的是二维数组。

答案：B

面试例题 2：语句

```
char foo='和'
```

是否正确？（假设代码以 GB2312 编码存储，并且以 javac -encoding GB2312 命令编译）

A．编译并运行正确　　B．编译错误

C．编译正确但运行错误　　D．以上答案都不对

答案：A

面试例题 3：一个数组中，“支配者”是在数组中出现频率超过一半的整数，例如[3，4，3，2，-1，3，3，3]，其中数值“3”出现过 5 次，5 除以 8 大于 0.5。所以数值“3”是一个支配者。写一个函数，在给定的整数数组中找出支配者是多少，如果一个数组中没有支配者，则返回-1。[中国东北某著名软件公司 D2009 年 10 月笔试题]

解析：本题可以这样做，先对数组做一个排序；然后统计相同数的值，如果相同数的数量超过了总数的一半，自然其为支配数。比如，对于数组[3，4，3，2，-1，3，3，3]，排序后为[-1，2，3，3，3，3，3，4]；-1 的数量为 1，不到总数的一半，舍弃；2 的数量为 1，不到总数的一半，舍弃；3 的数量为 5，超过总数的一半，成为“支配者”。

答案：代码如下：

```
import java.util.*;
public class Test {
    public static void main(String[] args) {
        int[] ints = {3,2,3,3,0,2,3};
        int j = judge(ints);
        //System.out.println(j);
        if(j==-1)
            System.out.println("No dominator");
        else
            System.out.println("The dominator is :" + ints[j]);
    }
    public static int judge(int[] ints) {
        Arrays.sort(ints);
        int counter = 1;
        for(int i=0;i<(ints.length - 1);i++) {
            if(ints[i] == ints[i+1]) {
                counter++;
                if(((double)counter) / ints.length > 0.5) {
                    return i;
                }
            } else {
                counter = 1;
            }
        }
        return -1;
    }
}
```

扩展知识：本题也可用集合来实现，算法复杂度为 $O(n^2)$。外循环 n 次，内循环平均 n/2 次。代码如下：

```
import java.util.HashSet;
public class Test {
    private HashSet<Integer> elements = new HashSet<Integer>();
    //一个集合存放所有出现的元素
    public void dominator(int [] arr){
    boolean found = false;//决定最终有没有支配者

        for(int i=0;i<arr.length;i++){
            //如果前面已经处理过此元素，则跳过
            if(elements.contains(new Integer(arr[i])))
                continue;
            //如果集合中不包含此元素，那么就往后面计数
            int counter = 0;
            for(int j=i+1;j<arr.length;j++){
                if(arr[i] == arr[j]){
                    counter++;
                }
            }
            double ratio = new Double(counter+1)/(arr.length);
            if(ratio>0.5){
                found = true;//找到，则退出
                System.out.println("Dominator found at position: "+i);
                break;
            }else{//当前元素不是支配者，则将其加入集合
                elements.add(new Integer(arr[i]));
            }
        }
        if(!found){
            System.out.println("No Dominator found!");
        }
    }

    public static void main(String[] args) {
        int[] array={3,2,3,3,0,2,3};
        new Test().dominator(array);
    }
}
```

面试例题 4：下列程序的输出结果是什么？[中国著名金融软件公司 SZ2009 年 10 月笔试题]

```
import java.util.*;
import java.math.BigInteger;
import java.lang.*;

public  class  Test  {
 public  static  void  main(String[]  args)  throws NumberFormatException {
      BigInteger  one=new  BigInteger("1");
      BigInteger  two=new  BigInteger("2");
      BigInteger  three=new  BigInteger("3");
      BigInteger  sum=new  BigInteger("0");
      sum.add(one);
      sum.add(two);
      sum.add(three);
      System.out.println(sum.toString());
    }
}
```

A．6　　B．0　　C．1　　D．3

解析：本题考的是 Java 中的大数类。

（1）BigInteger 属于 java.math.BigInteger，因此，在每次使用前都要 import（输入）这个类，否则会提示找不到提示符。

（2）BigInteger 构造方法有很多，如：

```
BigInteger(String val)
//将 BigInteger 的十进制字符串表示形式转换为 BigInteger
BigInteger(String val, int radix)
//将指定基数的 BigInteger 的字符串表示形式转换为 BigInteger
```

如要将 int 型的 2 转换为 BigInteger 型，要写为：

```
BigInteger two=new BigInteger("2"); //注意 2 的双引号不能省略
```

（3）BigInteger 类模拟了所有的 int 型数学操作，如 add()==+，divide()==-等，但注意进行数学运算时，不能直接使用数学运算符进行运算，必须使用其内部方法，而且其操作数也必须为 BigInteger 型。

如：two.add(2)就是一种错误的操作，因为 2 没有改为 BigInteger 型。

对本题而言，sum 返回一个 biginteger 的值并不改变原来的变量，所以 sum 的值始终为 0。如果想得到 6 的结果，必须显式地返回，代码如下：

```
import java.util.*;
import java.math.BigInteger;
import java.lang.*;
public  class  Test  {
 public  static  void  main(String[]  args)  throws NumberFormatException {
```

```
        BigInteger  one=new  BigInteger("1");
        BigInteger  two=new  BigInteger("2");
        BigInteger  three=new  BigInteger("3");
        BigInteger  sum=new  BigInteger("0");
        sum = sum.add(one);
        sum = sum.add(two);
        sum = sum.add(three);
        System.out.println(sum.toString());
    }
}
```

答案：B

面试例题 5：约瑟环问题。17 世纪的法国数学家加斯帕在《数目的游戏问题》中讲的一个故事：15 个教徒和 15 个非教徒在深海上遇险，必须将一半的人投入海中，其余的人才能幸免于难，于是想了一个办法：30 个人围成一个圆圈，从第一个人开始依次报数，每数到第九个人，就将他扔入大海，如此循环进行，直到仅余 15 个人为止。问怎样排法，才能使每次投入大海的都是非教徒。[中国深圳某著名软件公司 SX2009 年 12 月面试题]

解析：题目中 30 个人围成一圈，可以用一个循环的链来表示，使用结构数组来构成一个循环链。结构中有两个成员，其一为指向下一个人的指针，以构成环形的链；其二为该人是否被扔下海的做标记，为 1 表示还在船上。从第一个人开始对还未扔下海的人进行计数，每数到 9 时，将结构中的标记改为 0，表示该人已被扔下海了。这样循环计数，直到有 15 个人被扔下海为止。

还有一种办法：使用一个 Boolean 数组来模拟，非教徒站的位置为 false，教徒站的位置为 true。一开始，所有的位置为 true，每数到 9 时，则自动将 true 设为 false，如果该位置已经为 false，则设置下一个 true 位置为 false，如果已经循环到底，则重新开始。

答案：代码如下：

```
public class Test {
    public static void main(String[] args) {
        Boolean[] usaJapa = new Boolean[30];
        for(int i=0; i<usaJapa.length; i++) {
            usaJapa[i] = true;
        }


        int leftCount = usaJapa.length;
        int countNum = 0;
        int index = 0;
        while(leftCount>1) {
            countNum ++;
```

```
            if(countNum == 9) {
                countNum = 0;
                usaJapa[index] = false;
                leftCount --;
            }
            index ++;
            if(index == usaJapa.length)index=0;

        }
        for(int i=0; i<usaJapa.length; i++) {
            System.out.print(i + "=" + usaJapa[i] + "  ");
        }

    }
}
```

12.5 字符串其他问题

面试例题 1：下列程序的输出结果是什么？[中国台湾某软件公司 YH2009 年 4 月校园招聘笔试题]

```
import java.util.*;
 public class Test {
 public static void main(String[]  args)
 {
      String i = "";
      if(i == i + 0)
      {
           System.out.println( "Hello World ");
      }
 }
}
```

A．Hello world　　B．空　　C．编译错误　　D．以上答案都不对

解析：数值类型(如 int 类型)+0 的话，必然使 i!=i+0 为 true。而 String 的“+”运算并不是进行数值计算，而是进行字符串的连接。所以 i!=i+0 是 false。

答案：B

面试例题 2：对多个字符串进行排序，用 Java 编程语言来实现，不能使用现有的类。在排序中，字符串"Bc"，"Ad"，"aC"，"Hello"，"X man"，"little"，"During"，"day"能够排序成"Ad"，"aC"，"Bc"，"During"，"day"，"Hello"，"little"，"X man"，也就是说，排序的过程并不是传统地按照字符串排序，在排序中还需要将小写字母一并排序，也

就是说，a 字符串要在 B 或 b 之前。[中国台湾某软件公司 YH2009 年 4 月校园招聘笔试题]

解析：本题的难点在于比较字符串中字符的 ASCII 码值。

如果要写一个 StringCmp 方法来实现字符串比较大小，首先要确定思想，对字符串进行排序，判断两个字符串中第一个字母的 ASCII 码是否相等，如果相等，就判断字符串中下一个字母的 ASCII，以此类推，但是这里注意的是，a 的 ASCII 比 Z 要大，所以，在判断前还需要判断是否为小写字母，如果是小写，则转换成大写进行判断，最后可以返回一个 int 类型的值进行判断。

排序的方法题目没有要求，这里用的是冒泡法。

答案：完整的代码如下：

```
public class Test{
 public static void main(String args[]) {
     String aSource[] = { "dad", "bood", "bada", "Admin",  "  ", "Good",
 "aete", "cc", "Ko", "Beta", "Could" };
 boolean isChanged;
 int nMaxIndex = aSource.length - 1;
 String sTemp = null;
 try {
     do {
         isChanged = false;
         for (int i = 0; i < nMaxIndex; i++) {
             if (StringCmp(aSource[i], aSource[i + 1]) > 0) {
                 sTemp = aSource[i];
                 aSource[i] = aSource[i + 1];
                 aSource[i + 1] = sTemp;
                 isChanged = true;
             }
         }
         nMaxIndex--;
     } while (isChanged);
 } catch (Exception e) {
     System.out.print(e.getMessage());
 }
 for (int i = 0; i < aSource.length; i++) {
     System.out.println(aSource[i]);
 }
}

/**
* 字符串比较函数
*/
 public static int StringCmp(String s1, String s2) throws Exception {
```

```
        int nResult = s1.length() - s2.length();
        int nLen = Math.min(s1.length(), s2.length());
        int nCharOrder1, nCharOrder2;
        char aChar1[] = s1.toCharArray();
        char aChar2[] = s2.toCharArray();
        try {
            for (int i = 0; i < nLen; i++) {
                if (aChar1[i] != aChar2[i]) {
                    System.out.println(aChar1[i]);
                    System.out.println(aChar1[i]-96);
                    System.out.println(aChar1[i]-64);
                    //如果大于a，则减96，否则减64
                    nCharOrder1 = aChar1[i] > 96 ? aChar1[i] - 96
                            : aChar1[i] - 64;
                    nCharOrder2 = aChar2[i] > 96 ? aChar2[i] - 96
                            : aChar2[i] - 64;
                    if (nCharOrder1 != nCharOrder2) {
                        nResult = nCharOrder1 - nCharOrder2;
                        break;
                    } else {
                        nResult = aChar1[i] - aChar2[i];
                        break;
                    }
                }
            }
        } catch (Exception e) {
            throw e;
        }
        return nResult;
    }
}
```

面试例题 3：编写一个截取字符串的函数，输入为一个字符串和字节数，输出为按字节截取的字符串。但是要保证汉字不被截半个，如“"我 ABC",4”，应该截为“我 AB”，输入“"我 ABC 汉 DEF", 6”，应该输出为“我 ABC”，而不是“我 ABC+汉的半个”。[美国著名软件公司 GS2009 年 11 月面试题]

解析：本题不能直接使用 String 类的 substring(int beginIndex, int endIndex)方法，因为它是按 char 计算的。'爱'和'a'都被作为一个字符来看待，length 都是 1。string.getbytes()是按 byte 计算的，搞清楚这个应该就能懂了，一个汉字的 getbytes().length==2。如果原始字符不为 null，也不是空字符串，将原始字符串转换为 GBK 编码格式，要截取的字节数大于 0，且小于原始字符串的字节数。按照字符来分解字符串的，如果遇到中文汉字，截取字节总数减 1。

答案：代码如下：

```
import java.io.UnsupportedEncodingException;
public class Test {
    public static boolean isChineseChar(char c)
            throws UnsupportedEncodingException {
        return String.valueOf(c).getBytes("GBK").length > 1;
    }
    public static String cutstring(String orignal, int count)
            throws UnsupportedEncodingException {
        if (orignal != null && !"".equals(orignal)) {
            orignal = new String(orignal.getBytes(), "GBK");
            if (count > 0 && count < orignal.getBytes("GBK").length) {
                StringBuffer buff = new StringBuffer();
                char c;
                for (int i = 0; i < count-1; i++) {
                    c = orignal.charAt(i);
                    buff.append(c);
                    if (Test.isChineseChar(c)) {
                        --count;
                    }
                }
                return buff.toString();
            }
        }
        return orignal;
    }

    public static void main(String[] args) {
        String s = "我gfr是Chinese";
        try {
            System.out.println("cut1bit" + Test.cutstring(s, 1));
            System.out.println("cut2bit" + Test.cutstring(s, 2));
            System.out.println("cut4bit" + Test.cutstring(s, 6));
            System.out.println("cut6bit" + Test.cutstring(s, 7));
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }
}
```

12.6　范型与容器

面试例题 1：以下程序的输出结果是多少?[美国著名 CPU 公司 A2008 年校园招聘题目]

```
import java.util.*;
```

```
public class Test {
  public static void main(String[] args) {
      Queue<Integer> queue = new PriorityQueue<Integer>(10,
              new Comparator<Integer>() {
                  public int compare(Integer i, Integer j) {
                      int result = i % 2 - j % 2;
                      if (result == 0)
                          result = i - j;
                      return result;
                  }
              });

      for (int i = 0; i < 10; i++) {
          queue.offer(i);
      }

      for (int i = 0; i < 10; i++) {
          System.out.println(queue.poll());
      }
  }
}
```

解析： java.util.Queue 接口用来支持队列的常见操作。该接口扩展了 java.util.Collection 接口。Queue 通常使用 offer()来加入元素，使用 poll()来获取并移出元素。它们的优点是通过返回值可以判断成功与否，如果要使用前端而不移出该元素，使用 element()或者 peek()方法。Queue 接口的常见方法有 element()检索，但是不移除此队列的头。此方法与 peek 方法的唯一不同是，如果此队列为空，它会抛出一个异常。如果可能，将指定的元素插入此队列。使用可能有插入限制（例如容量限定）的队列时，offer 方法通常要优于 Collection.add()方法，因为后者只能通过抛出异常使插入元素失败。peek()检索，但是不移除此队列的头，如果此队列为空，则返回 null。poll()检索并移除此队列的头，如果此队列为空，则返回 null。remove()检索并移除此队列的头。此方法与 poll()方法的不同在于，如果此队列为空，它会抛出一个异常。

值得注意的是，LinkedList 类实现了 Queue 接口。因此，我们可以把 LinkedList 当成 Queue 来用。

```
import java.util.*;

public class Test {
  public static void main(String[] args) {
      Queue<String> queue = new LinkedList<String>();
      queue.offer("one");
      queue.offer("two");
```

```
        queue.offer("three");
        System.out.println(queue.size());
        String str;
        while ((str = queue.poll()) != null) {
            System.out.println(str);
        }

        System.out.println(queue.size());
    }
}
```

至于本题，只是一个优先队列的排序问题，按奇偶排序，即先偶后奇的顺序排列进行输出。

答案：0 2 4 6 8 1 3 5 7 9

面试例题 2：下列代码有什么错误？ [中国某著名杀毒软件公司 JS2010 年 1 月笔试题]

```
public class Test {
public static void main(String[] args) {
    List<String> list = new ArrayList<String>();
    list.add("Welcome ");
    list.add("to ");
    list.add("Xian ");
    for (Iterator  i=list.iterator();i.hasNext(); ) {
        String s = i.next();
        System.out.println(s);
    }
}
}
```

解析：本题会发生编译错误。

这道题考查对 Java 泛型的理解，在代码中可以看到 List 泛型参数是 String，也就是说，List 只允许存放 String 类型的数据。该题的错误之处主要在于

```
for(Iterator  i=list.iterator();i.hasNext();)
```

这一句上，通过 API 文档可以看到 Iterator 迭代器也是需要采用泛型参数的。如果没有加泛型参数，i.next();返回的结果是 Object 类型，所以

```
String  s=i.next();
```

就会报类型不匹配的错误。为了类型安全，应该改为：

```
import java.util.*;
public class Test {
```

```
    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();
        list.add("Welcome ");
        list.add("to ");
        list.add("Xian ");
        for (Iterator<String> i = list.iterator(); i.hasNext();) {
            String s = i.next();
            System.out.println(s);
        }
    }
}
```

答案：程序不正确，代码

```
for (Iterator  i=list.iterator();i.hasNext(); )
```

应该改成

```
for (Iterator<String> i = list.iterator(); i.hasNext();)
```

面试例题 3：Which are not containers in Java?（下面哪个不是 Java 中的容器）[美国著名 CPU 公司 A2008 年校园招聘题目]

A．ScollPane　　B．Canvas　　C．Dialog　　D．Applet

解析：容器就是能放组件的东西。因此，Canvas（画布）不是容器。至于 Dialog，其实它也是容器的一种。

答案：B

面试例题 4：You need to store elements in a collection that guarantees that no duplicates are stored and all elements can be access in nature order, which interface provides that capability?（需要存储某些元素在一个容器里，而且要保证没有重复的数据在一个自然队列。下面哪个接口能实现这种功能）[美国著名 CPU 公司 A2008 年校园招聘题目]

A．java.util.SortedMap　　B．java.util.Set

C．java.util.List　　D．java.util.SortedSet

解析：选 java.util.SortedSet，Set 保证唯一，Sorted 保证排序。

答案：D

第13章

设计模式

地上本没有路，走的人多了，也就成了路。设计模式如同此理，它是经验的传承，并不成体系；它是被前人发现，经过总结形成了一套某一类问题的一般性解决方案，而不是被设计出来的定性规则；它不像算法那样可以照搬照用。

设计模式关注的重点在于通过经验提取的“准则或指导方案”在设计中的应用，因此，在不同层面考虑问题的时候就形成了不同问题域上的模式。模式的目标是把共通问题中的不变部分和变化部分分离出来。不变的部分就构成了模式。因此，模式是一个经验提取的“准则”，并且在一次一次的实践中得到验证。不同的层次有不同的模式，小到语言实现，大到架构。在不同的层面上，模式提供不同层面的指导。

与建筑结构一样，软件中亦有诸多的“内力”。与建筑设计一样，软件设计也应该努力疏解系统中的内力，使系统趋于稳定、有生气。一切软件设计都应该由此出发。任何系统都需要有变化，否则任何系统都会走向死亡。作为设计者，应该适应变化，而不是逃避变化。

经典设计模式一共有 23 种，面试时重要的不是你熟记了多少个模式的名称，关键还在于付诸实践的运用。为了有效地设计，而去熟悉某种模式所花费的代价是值得的，因为你很快会在设计中发现这种模式真的很好，很多时候它令你的设计更加简单。

其实公司需要一个真正出色的设计师，懂得判断运用模式的时机。很多才踏入软件设计领域的人员往往对设计模式很困惑，对于他们来说，由于没有项目的实际经验，面向对象的思想也还未曾建立，设计模式未免过于高深了。其实，即使是非常有经验的程序员，也不敢夸口对各种模式都能合理地应用。

面试中关于设计模式的题目主要以 C#或 Java 的形式出现，本章将对各大公司（尤其是外企）的设计模式、Java 等题目进行详细分析，帮助读者应对关于设计模式方面的面试考量。

13.1 UML

面试例题 1：which is true about UML（下面关于 UML 的说法哪种是正确的？）[美国著名数据库公司 SA2010 年 1 月笔试题]

A．UML is a standardlized approach for use case modeling.（UML 是一种用例模型的标准接口）

B．UML supplies a set of notations used in the design of applications（UML 是一种应用软件设计使用方法的标记）

C．UML is a methodology for designing and maintaining computer systems（UML 是一种用来设计和维护计算机系统的方法）

D．UML stands for Unified Modeling Language.（UML 代表统一建模语言）

答案：D

面试例题 2：When using the writeObject method to store the state of n object，how can you protect sensitive data from being accessed in the stored object?（如何保证敏感数据不被存储对象访问？）[美国著名数据库公司 SA2010 年 1 月笔试题]

A．Declare the sensitive fields as public（宣布敏感域公有）

B．Declare the sensitive fields as private transisent（宣布敏感域作为私有）

C．Declare the sensitive fields as static（宣布敏感域静态）

D．Declare the sensitive fields as protected（宣布敏感域私有）

答案：B

面试例题 3：对于关联端点(Association end)的表述，不正确的选择是哪个？[中国某软件公司 Y2009 年 10 月面试题]

A．关联端点是关联的一个结构部分，它定义了在关联中类的参与

B．在同一个关联中一个类可以连接到多个端点

C．在关联中的关联端点有不同的位置而且有名字，并且通常是可互换的

D．关联端点一旦脱离它的关联独立存在也不再有含义

解析：关联是两个或多个特定类元之间的关系，它描述了这些类元的实例的联系。参与其中的类元在关联内的位置有序。在一个关联中同一个类可以出现在多个位置上。关联的每个实例（链接）是引用对象的有序表，关联的外延即这种链接的一个集合。在链接集合中给定的对象可以出现多次，或者在关联的定义允许的情况下可以在同一

个链中（在不同的位置）出现多次。关联将一个系统组织在一起，如果没有关联，那只有一个无连接类的集合。

关联的结构：关联可以有一个名称，但是它的大部分描述建立在关联端点中，每个端点描述了关联中类对象的参与。关联端点只是关联描述的一部分，不是可区分的语义或可用符号表示的概念。

关联的名称：关联可以有一个名称，在包含的所有关联和类中，它必须是唯一的（关联类既是一个关联，又是一个类，所以关联和类分享同一个命名空间）。关联不是必须要有一个名称，它的端点的角色名称提供了在同一个类中辨别多个关联的另一种途径。按照习惯，名字以类在表中出现的顺序读出：employee 为 company 工作；salesman 卖 car 给消费者。

关联的端点：关联包含一张有多个关联端点的有序表（也就是说，这些端点是可以被区分的并且是不可替换的）。每个关联端点定义了在关联中给定位置的一个类（角色）的参与。同一个类可以出现在多个位置上，而位置通常是不可交换的。每个关联端点指定了的应用于对应对象的参与特性，如在关联中一个独立的对象在链接中会出现多少次（多重性）。某些特性，如导航性只应用于二元关联，但是多数可以应用于 n 元关联。

在 UML 的语义中，位置的关系（类元的关联）称为链。链在系统执行过程中可以被创建和销毁，服从每个关联端点可变性的限制（类元本身的对应或所属关系）。

答案：B

13.2　常见设计模式

面试例题 1：写一个 Singleton 出来。

答案：Singleton 模式的主要作用是保证在 Java 应用程序中，一个类 Class 只有一个实例存在。

Singleton 模式通常有两种形式。

第一种形式：定义一个类，它的构造函数为 private 的，它有一个 static 的 private 的该类变量，在类初始化时实例化，通过一个 public 的 getInstance 方法获取对对象的引用，继而调用其中的方法。

```
public class Singleton {
    private Singleton(){}
    //在自己内部定义自己一个实例，是不是很奇怪?
    //注意，这是private，只供内部调用
```

```
    private static Singleton instance = new Singleton();
    //这里提供了一个供外部访问本 class 的静态方法，可以直接访问
    public static Singleton getInstance() {
       return instance;
    }
}
```

第二种形式：

```
public class Singleton {
    private static Singleton instance = null;
    public static synchronized Singleton getInstance() {
    //这种方法比上面有所改进，不用每次都进行生成对象，只是第一次
    //使用时生成实例，提高了效率
    if (instance==null)
       instance=new Singleton();
    return instance;     }
}
```

面试例题 2：In the selling system of the beefsteak Coffee Stall, there is a series of flavor beefsteak, such as philli beefsteak, curry beefsteak, cheese beefsteak. Now we try to use the design pattern to describe the beefsteak sell system. （在一个牛排咖啡摊（the beefsteak Coffee Stall）所使用的系统里，有一系列的“风味（Flavor）牛排”。比如说菲利牛排、咖哩牛排、奶酪牛排等。现在我们想用设计模式来描述牛排销售系统，该用哪种设计模式？）[德国著名 ERP 公司 S2005 年 10 月面试题]

为了设计上面的东西该选用：

A．单键模式　　B．桥接模式　　C．享元模式　　D．观察者模式

解析：本题应该选择享元模式。Flyweight 在拳击比赛中指最轻量级，即“蝇量级”，有些作者翻译为“羽量级”。这里使用“享元模式”更能反映模式的用意。

享元模式以共享的方式高效地支持大量的细粒度对象。享元对象能做到共享的关键是区分内蕴状态（Internal State）和外蕴状态（External State）。内蕴状态是存储在享元对象内部并且不会随环境改变而改变。因此，内蕴状态可以共享。

外蕴状态是随环境改变而改变的、不可以共享的状态。享元对象的外蕴状态必须由客户端保存，并在享元对象被创建之后，在需要使用的时候再传入到享元对象内部。外蕴状态与内蕴状态是相互独立的。

享元模式在编辑器系统中大量使用。一个文本编辑器往往会提供很多种字体，而通常的做法就是将每一个字母做成一个享元对象。享元对象的内蕴状态就是这个字母，而字母在文本中的位置和字模风格等其他信息则是外蕴状态。比如，字母 a 可能出现

在文本的很多地方，虽然这些字母 a 的位置和字模风格不同，但是所有这些地方使用的都是同一个字母对象。这样，字母对象就可以在整个系统中共享。

在这个牛排摊所使用的系统里，有一系列的牛排“风味（Flavor）”。客人到摊位上购买牛排，所有的牛排均放在台子上，客人自己拿到牛排后就离开摊位。牛排有内蕴状态，也就是牛排的本身风味；牛排没有环境因素，也就是说没有外蕴状态。如果系统为每一碟牛排都创建一个独立的对象，那么就需要创建出很多的细小对象来。这样就不如把牛排按照种类（即“风味”）划分，每一种风味的牛排只创建一个对象，并实行共享。

使用牛排摊主的话来讲，所有的牛排都可按“风味”划分成如 curry beefsteak、cheese beefsteak 等，每一种风味的牛排不论卖出多少碟，都是相同的、不可分辨的。所谓共享，就是牛排风味的共享，制造方法的共享等。因此，享元模式对牛排摊来说，就意味着不需要为每一份单独调制。摊主可以在需要时，一次性地调制出足够一天出售的某一种风味的牛排。

代码如下：

```
import java.util.*;
abstract class Order
{
  // 将牛排卖给客人
  public abstract void Serve();
  // 返回牛排的名字
  public abstract String GetFlavor();
}

 class Flavor extends Order
{
  private String flavor;

   // 构造函数，内蕴状态以参数方式传入
   public Flavor(String flavor)
   {
     this.flavor = flavor;
   }
   // 返回牛排的名字
   public  String GetFlavor()
   {
     return this.flavor;
   }
   // 将牛排卖给客人
   public  void Serve()
   {
       System.out.println("Serving flavor " + flavor);
   }
```

```
}

class FlavorFactory
{
  private Hashtable flavors = new Hashtable();
  public Order GetOrder(String key)
  {
    if(! flavors.containsKey(key))
      flavors.put(key, new Flavor(key));

      return ((Order)flavors.get(key));
  }
  public int GetTotalFlavorsMade()
  {
    //return flavors.count;
    return flavors.size();
    //return 3;
  }
}

public class Client
{
  private static FlavorFactory flavorFactory;
  private static int ordersMade = 0;
  public static void main(String[] args)
  {
    flavorFactory = new FlavorFactory();
     TakeOrder("philli beefsteak");
     TakeOrder("curry beefsteak");
     TakeOrder("cheese beefsteak");
     TakeOrder("curry beefsteak");
     TakeOrder("cheese beefsteak");
     TakeOrder("philli beefsteak");
     TakeOrder("cheese beefsteak");
     TakeOrder("cheese beefsteak");
     TakeOrder("philli beefsteak");
     TakeOrder("curry beefsteak");
     TakeOrder("curry beefsteak");
     TakeOrder("philli beefsteak");

     System.out.println("\nTotal Orders made: " + orders- Made);

     System.out.println("\nTotal Flavor objects made: " +
       flavorFactory.GetTotalFlavorsMade());
```

```
  }
  private static void TakeOrder(String aFlavor)
  {
    Order o = flavorFactory.GetOrder(aFlavor);
    // 将牛排卖给客人
    o.Serve();

    ordersMade++;
  }

}
```

答案：C

面试例题 3：下面程序的输出结果是多少？[德国著名 ERP 公司 S2005 年 10 月面试题]

```
//: LocalCopy.java
// 通过 clone 方法创建本地副本
import java.util.*;

class MyObject implements Cloneable {
 int i;

 MyObject(int ii) {
      i = ii;
 }

 public Object clone() {
      Object o = null;
      try {
           o = super.clone();
      } catch (CloneNotSupportedException e) {
           System.out.println("MyObject can't clone");
      }
      return o;
 }

 public String toString() {
      return Integer.toString(i);
 }
}

public class LocalCopy {
 public static MyObject g(MyObject v) {
      // 传递一个句柄，修改外部对象
```

```
        return v;
    }

    public static MyObject f(MyObject v) {
        v = (MyObject) v.clone(); // 本地复制

        return v;
    }

    public static void main(String[] args) {
        MyObject a = new MyObject(11);
        MyObject b = g(a);
        a.i++;
        // 测试句柄等值，
        // 非对象等值
        if (a == b)
            System.out.println("a == b");
        else
            System.out.println("a != b");
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        MyObject c = new MyObject(47);
        MyObject d = f(c);
        c.i++;
        if (c == d)
            System.out.println("c == d");
        else
            System.out.println("c != d");
        System.out.println("c = " + c);
        System.out.println("d = " + d);
    }
}
```

解析：这道题的考点是设计模式中的原型模式。实际上，题中分别提出两个概念：浅复制和深复制。

浅复制是指当对象的字段值被复制时，字段引用的对象不会被复制。例如，如果一个对象有一个指向字符串的字段，并且我们对该对象做了一个浅复制，那么两个对象将引用同一个字符串。而深复制是对对象实例中字段引用的对象也进行复制的一种方式，所以，如果一个对象有一个指向字符串的字段，并且对该对象做了一个深复制的话，我们将创建一个新的对象和一个新的字符串——新对象将引用新字符串。需要注意的是，执行深复制后，原来的对象和新创建的对象不会共享任何东西，改变一个对象对另外一个对象没有任何影响。

题目中方法 g 是浅复制，只是复制对象，但并不新开辟内存，相当于给一个仓库

再配一把钥匙。而方法 f 是深复制，不但复制对象句柄，而且新开辟内存，相当于重新建一座仓库。所以，当原仓库发生改变时，不会影响新仓库的使用。在本题中对象 a、b 就是复制钥匙，共享仓库，所以当 a 对仓库发生改变时（由 11 变到 12），b 对象的结果同时改变。而 c、d 是复制仓库的关系，所以当 c 对老仓库发生改变时（由 47 变到 48），d 对象所指向的新仓库不会改变。

答案：输出结果如下。

a == b
a = 12
b = 12
c != d
c = 48
d = 47

扩展知识

原型模式的结构图如图 13-1 所示。

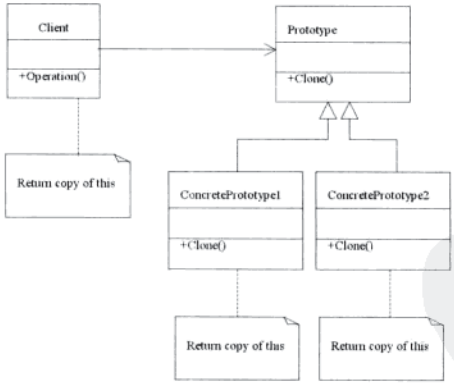


图 13-1　原型模式的结构图

下面介绍 Prototype 模式的优点与缺点。

1．Prototype 模式的优点

（1）Prototype 模式允许动态增加或减少产品类。由于创建产品类实例的方法是产品类内部具有的，因此，增加新产品对整个结构没有影响。

（2）Prototype 模式提供了简化的创建结构。工厂方法模式常常需要有一个与产品类等级结构相同的等级结构，而 Prototype 模式就不需要这样。

（3）Prototype 模式具有给一个应用软件动态加载新功能的能力。由于 Prototype 的独立性较高，很容易动态加载新功能而不影响老系统。

（4）产品类不需要非得有任何事先确定的等级结构，因为 Prototype 模式适用于任何的等级结构。

2．Prototype 模式的缺点

Prototype 模式最主要的缺点就是每一个类必须配备一个克隆方法，而且这个克隆方法需要对类的功能进行通盘考虑，这对全新的类来说不是很难，但对已有的类进行改造时，不一定是一件容易的事。

面试例题 4：简述 Struts 下的设计模式?

答案：MVC (Model/View/Controller)模式是一种使用得较多的设计模式。MVC 包括三类对象，Model 是应用对象，View 是它在屏幕上的表示，Controller 定义用户界面对用户输入的响应方式。

模型是应用程序的主体部分。模型表示业务数据或者业务逻辑；视图是应用程序中用户界面相关的部分，是用户看到并与之交互的界面；控制器工作就是根据用户的输入，控制用户界面数据显示和更新 model 对象状态。MVC 模式的出现不仅实现了功能模块和显示模块的分离，同时，它还提高了应用系统的可维护性、可扩展性、可移植性和组件的可复用性。

不一定所有的 JSP 应用都该采用 MVC 模式。但对于大型应用来说，最好还是采用 MVC 模式。若不使用 MVC 模式，用户界面设计时往往将这些对象混在一起，而 MVC 则将它们分离以提高灵活性和复用性。

13.3 软件工程

面试例题 1：三层次（或多层次）软件架构相比二层次软件架构有什么优势？[加拿大某通讯公司 N2008 年面试题]

解析：在规划、设计企业信息系统时，需要考虑的主要是信息系统的基础架构，它的主要特点如下：

- 先进性；
- 开放性，依从主流的技术标准；
- 安全性；
- 与现有系统的兼容性，异种系统之间的互连；

- 技术的成长性；
- 应用系统开发工具。

已经存在的计算机软件远远跟不上我们的要求，尤其是目前大量使用的两层次的应用软件更是一个瓶颈，主要问题归纳如下：

- 部门协作使得信息通道多样，连接复杂，安全管理困难；
- 大量信息在各级、各地间公用；
- 各企业内公用信息和私有信息并存；
- 要求将各种系统资源（数据库、消息传递服务器、邮件服务器）集成在一起；
- 要求快速反应，能够移动工作，并随时随地访问各种信息，相互通信。

回顾一下大多数企业采用的技术，企业信息系统应用架构主要以二层次（Two-tier）平台为基础，是面向数据的应用结构。其理想的应用程序环境是 100 个客户端以下，通常只有一个数据源（RDBMS），提供基于局域网的连接，并只能提供较低的安全保障。在这种体系结构中，应用程序的界面和逻辑都放在客户端，从而对客户端的要求比较高。

从开发和维护的角度考虑，当超过 100～150 个客户端时，每个客户端的开销会呈非线性增长，同时二层次的数据库系统互操作能力不强，会给部门和部门间带来许多应用限制。当客户端的数量超过一定限度时，开发维护的代价会显著增长，单凭硬件升级，并不能解决问题。

在企业信息系统中，登录点是大量的且来自各方面的。各个基层单位的查询点的计算机系统情况多种多样，要求有高效的查询能力和统一的界面。在传统意义上，这是很难实现的。企业业务工作的数据源是多种的，分布广泛的，数据量十分庞大，要做到快速查询和网上安全传输，对应用系统的体系结构要求比较高。同时，企业信息系统还要求严格的保密性，信息按照保密级别严格分类，维护整体性、实用性、可扩展性和先进性的系统设计原则。

所以，不难看出，传统的二层次（Two-tier）的体系结构不适合企业信息系统的发展，我们必须实现新的、更适应要求的软件产品来满足企业的需要。

下面介绍三层次（或多层次）软件架构与中间件。

三层次的浏览器/服务器架构是基于 Web 的、先进的体系结构，在这种架构中，利用成熟的 Web 应用服务器（WAS）和事务处理服务器，为应用程序提供 Web 运行环境。数据资源和客户机将被“应用服务器”分隔开，应用服务器上存储着应用逻辑。这种结构着重于客户机对应用服务的请求，有别于二层次架构着重于数据请求。

其实，中间件是一个比较笼统的说法，顾名思义，我们可以将三层次架构的中间层的所有服务器软件统称为中间件，其主要的任务是响应客户端的请求，进行复杂的企业逻辑运算，访问企业的后台数据资源，生成满足客户需要的结果并返回给客户。除了这些基本功能之外，中间件软件还应该提供下面的功能。

- 应用服务器为应用程序提供各种服务；
- 程序加载、程序启动、内存管理、负载平衡、出错恢复及强大的应用管理功能；
- 高性能地处理大量并发访问，及时快速响应；
- 屏蔽异构平台，具有强大的和后台各种资源（应用系统、各种数据库）的连接。

在三层次结构中，客户端对数据源的直接访问被对应用程序的请求所替代，客户端访问的是应用程序，由应用程序对数据进行查询和存取，这样就能保证数据不被非法使用和篡改。目前我们比较多的是基于 Web 的三层次结构应用，一般以标准的 TCP/IP 网络为平台，可很好地与网络开发语言如 Java 等有效地集成，使异种资源的结合变得容易。同时，这种结构也提高了系统的性能，简化了用户的管理。

答案：三层次架构要求的初期投资比二层次的体系结构高，但是它具有极高的长期扩展性，随着客户数量、应用复杂度的增加，开发和维护的费用基本上随之呈线性增长。三层次架构的浏览器/服务器架构着重于客户机对应服务的请求，而二层次架构仅局限于数据的简单请求。这些优点将有效地帮助企业信息系统实现近期和长远的目标。

面试例题 2：总结自己在项目中的架构设计经验，有什么感悟？

答案：好的软件架构也是为变化而设计的。用户的需求日新月异、千变万化，大的项目一般都会有第一期、第二期、第三期……怎样才能让自己的架构设计能满足不断变化的用户需求呢？“为变化而设计”，应该是解决这个问题的战略方针。

怎样才能实现“为变化而设计”这个战略方针呢？答案是分层。分层就是实现战略目标的战术、战斗方法。

下面是分层的方法和步骤。

首先，要确定软件的生命周期。我们应该避免为一个 5 分钟的问题提供一个 50 年的解决方案，也不要为一个 50 年的问题提供一个 5 分钟的解决方案。我们在实际开发过程中发现，每个项目都有它的生命周期，有 1 年的、2 年的、5 年的、10 年的，也有 1 天的，甚至 5 分钟的。我们应该根据软件的生命周期来设计。

其次，要确认软件中变和不变的因素，并根据变化频度进行分类。依据变化频度的分类进行层次划分，并标出各个层次的依赖关系。对于不变的部分，要设计得稳固、牢靠。对于变化的部分，根据变化的频度灵活设计，以便于修改。

第 4 部分

操作系统、数据库、网络

OS、DB、Network

本部分主要介绍求职面试过程中出现的第三个重要的板块——操作系统、数据库、网络知识。作为一个程序员，尤其是系统管理方面的程序员，对这几部分有深刻的理解和领悟是相当重要的。

第14章

操作系统

操作系统面试例题主要涉及进程、线程、内存管理、垃圾回收，以及缓存等诸方面。以下的考题来自真实的笔试资料，希望读者先不要看答案，自我解答后再与答案加以比对，找出自己的不足。

14.1 基础知识

面试例题 1：Which of the following best describes a multitasking operating system?（下面关于多任务操作系统描述正确的是哪个？）[中国著名通讯公司 ZX2009 年 11 月校园招聘笔试题]

A. An operating system that can run multiple tasks concurrently on multiple CPUs.（在多个 CPU 上同时运行多个任务）

B. An operating system that can run a single task concurrently across multiple CPU.（在多个 CPU 上同时运行 1 个任务）

C. An operating system that can run multiple tasks concurrently on a single CPU.（在 1 个 CPU 上同时运行多个任务）

D. An operating system that constantly switches CPU time between loaded processes.（在 1 个 CPU 上运行 1 个任务）

解析：先解释两个概念：多用户和多任务。

多用户：容许在同一时间有很多人使用同一部机器，只要每个用者都有他的一个终端（terminal）。

多任务：为了使很多程序可以一同运行，这个特点叫做多任务。

多任务又分为协同多任务和抢占式多任务。

协同多任务：在一个多任务处理环境下，只有当其他程序认可时，一个运行程序才可获得处理机时间。每一个应用程序必须协同地放弃对处理器的控制，以便其他应用程序运行。

抢占式多任务处理：多任务处理的一种形式。必要时，操作系统可以中断当前正运行的任务以便运行另一个任务。

那么，CPU 的多任务抢占机制是如何实现的？

CPU 是通过设置时间中断来实现抢占机制的，在进入保护模式之前，先初始化中断向量表，在时钟中断入口处放置任务切换代码。然后设置好时钟中断的时间。

当某一线程的执行用光了时间片时，时钟中断产生，CPU 转去执行中断处的任务切换代码，保存当前线程的状态，得到并恢复下一个线程的状态，然后转去执行那个线程，以此类推。

为了尽可能地提高 CPU 的利用率，当某一线程必须转入等待状态，如要获取某一低速外设的数据时，这里的读操作程序会主动放弃当前没有用光的时间片，主动调用任务切换中断代码，把 CPU 的时间让给其他已经准备好可以继续的线程。

答案：C

面试例题 2：Which permission, when applied to a directory in the file system, will allow a user to enter the directory?（下列哪种权限许可用户进入一个文件系统的目录？）

A．Read（读权限）　　B．Write（写权限）

C．Execute（执行权限）　　D．Access Control（访问控制权限）

解析：只有具备执行权限才允许用户进入一个文件系统的目录。

答案：C

面试例题 3：IA 硬件架构，Linux 32 位操作系统下，一个应用层的程序最多能分配和访问的内存大小为哪个？[中国著名通讯公司 ZX2009 年 11 月校园招聘笔试题]

A．1GB　　B．2GB　　C．3GB　　D．4GB

E．8GB　　F．硬件能支持到多少，就能访问多少

解析：所有的 32 位应用程序都有 4GB 的进程地址空间，因为 32 位地址最多可以映射 4GB 的内存。对于 Windows，应用程序可以访问 2GB 的进程地址空间（32 位 Linux 可以访问 3GB 地址空间），这就是称为用户模式的虚拟地址空间。这 2GB 的用户模式虚拟地址空间位于 4GB 地址空间的低一半，而与之相对应的高一半 2GB 地址空间由操作系统内核使用，因此，被称为内核模式的虚拟地址空间。在一个进程中，所有的

线程读共享相同的 2GB 用户模式虚拟地址空间。

为了缓解地址空间的不足，微软提供了一个权宜的解决方案，所有从 Windows 2000 Server 开始的操作系统版本都提供了一个 boot.ini 启动开关（一般大小为 3GB），可以为应用程序提供访问 3GB 的进程地址空间的能力，从而将内核模式的地址空间限定为 1GB。以下就是一个开启了 3GB 选项的 boot.ini 文件示例：

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)WINDOWS="Windows Server 2003, Enterprise" /fastdetect /3GB
```

答案：C

面试例题 4：假设就绪队列中有 10 个进程，系统将时间片设为 200ms，CPU 进程切换要花费 10ms。则系统开销所占比率为多少？[中国著名通讯公司 ZX2009 年 11 月校园招聘笔试题]

A．1%　　B．5%　　C．10%　　D．20%

解析：注意，10 个进程是在就绪队列里的，10 个进程等效于 2 个进程。

A-B-A-B……系统中 A 和 B 不停地切换执行，取一个周期，A-B-

20ms/400ms = 5%。

答案：B

14.2　进程

面试例题 1：At time 0, process A has arrived in the system, in that order; at time 30, both progress B and C have just arrived; at time 90, both progress D and E have also arrived.The quantum or timeslice is 10 units.（在 0 时刻，进程 A 进入系统，按照这个顺序，在 30 时刻，进程 B 和进程 C 也抵达；在 90 时刻，进程 D 和进程 E 也抵达。一个时间片是 10 个单元）[美国著名操作系统综合软件公司 M2009 年 9 月笔试题]

Process A requires 50 units of time in the CPU;

Process B requires 40 units of time in the CPU;

Process C requires 30 units of time in the CPU;

Process D requires 20 units of time in the CPU;

Process E requires 10 units of time in the CPU;

（进程 A 需要占用 CPU 50 个单元；
进程 B 需要占用 CPU 40 个单元；
进程 C 需要占用 CPU 30 个单元；
进程 D 需要占用 CPU 20 个单元；
进程 E 需要占用 CPU 10 个单元；）

Which of the process will be the LAST to complete, if scheduling policy is preemptive SJF (Short Job First)（如果按照短作业优先级的方法，哪个进程最后结束。）

解析：牢记“短作业优先”=“最短剩余时间作业优先”。本题考的是可剥夺式处理机的调度问题。时间图如图 14-1 所示。

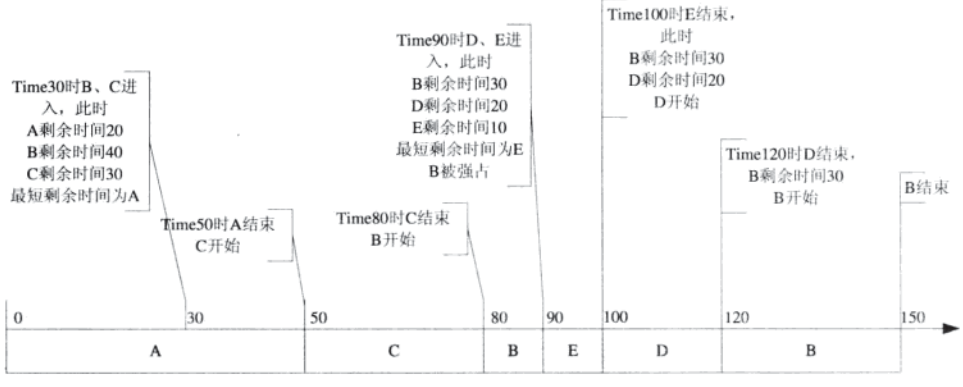


图 14-1　短作业问题

答案：如果按短作业优先级的方法，进程 B 最后结束。

扩展知识：读者试着扩展思维，想想如果本题是不可剥夺式处理机的调度方法，哪个进程最后结束。

面试例题 2：请描述进程与线程之间的关系，线程的优点和不足是什么？

解析：以一个比喻来理解进程与线程之间的关系。假设有一个公司，公司里有很多各司其职的职员，那么我们可以认为这个正常运作的公司就是一个进程，而公司里的职员就是线程。一个公司至少有一个职员吧，同理，一个进程至少包含一个线程。在公司里，可以一个职员干所有的事，但是效率很显然是高不起来的，一个人的公司也不可能做大。一个程序中也可以只用一个线程去做事，事实上，一些过时的语言如 Fortune、Basic 都是如此，但是像一个人的公司一样，效率很低，如果做大程序，效率

更低——事实上，现在几乎没有单线程的商业软件。公司的职员越多，老板就得发越多的薪水给他们，还得耗费大量精力去管理他们，协调他们之间的矛盾和利益。程序也是如此，线程越多，耗费的资源也越多，需要 CPU 花时间去跟踪线程，还得解决诸如死锁、同步等问题。总之，如果不想公司被称为“皮包公司”，就得多几个员工；如果不想让程序显得稚气，就在程序里引入多线程。

答案：Windows 是一个多任务的系统，如果使用的是 Windows 2000 及其以上版本，可以通过任务管理器查看当前系统运行的程序和进程。什么是进程呢？当一个程序开始运行时，它就是一个进程，进程所指的包括运行中的程序及程序所使用到的内存和系统资源。而一个进程又是由多个线程所组成的。线程是程序中的一个执行流，每个线程都有自己的专有寄存器（栈指针、程序计数器等），但代码区是共享的，即不同的线程可以执行同样的函数。多线程是指程序中包含多个执行流，即在一个程序中可以同时运行多个不同的线程来执行不同的任务，也就是说，允许单个程序创建多个并行执行的线程来完成各自的任务。浏览器就是一个很好的多线程的例子。在浏览器中可以在下载 Java 小应用程序或图像的同时滚动页面，在访问新页面时，播放动画和声音，打印文件等。

多线程的好处在于可以提高 CPU 的利用率。任何一个程序员都不希望自己的程序在很多时候没事可干，在多线程程序中，当一个线程必须等待的时候，CPU 可以运行其他的线程而不是等待，这样就大大提高了程序的效率。

然而也必须认识到线程本身可能影响系统性能的不利方面，以正确使用线程。

（1）线程也是程序，所以线程需要占用内存，线程越多，占用内存也越多。

（2）多线程需要协调和管理，所以需要 CPU 花时间跟踪线程。

（3）线程之间对共享资源的访问会相互影响，必须解决竞用共享资源的问题。

（4）线程太多会导致控制太复杂，最终可能造成很多 Bug。

面试例题 3：What are these ways in which a thread can enter the waiting state?（进程进入等待状态有哪几种方式？）[德国著名 ERP 软件公司 S2005 年面试题]

解析：本题属于操作系统面试例题。

答案：

进程进入等待状态有 3 种方式：

- CPU 调度给优先级更高的 thread，原先 thread 进入等待状态。
- 阻塞的 thread 获得资源或者信号，进入等待状态。
- 时间片轮转的情况下，如果时间片到了，也将进入等待状态。

14.3　线程与串行化

面试例题 1：sleep() 和 wait() 有什么区别?

答案：

1．sleep()

sleep()是使线程停止一段时间的方法。在 sleep 时间间隔期满后，线程不一定立即恢复执行。这是因为在那个时刻，其他线程可能正在运行，而且没有被调度为放弃执行，除非“醒来”的线程具有更高的优先级，或者正在运行的线程因为其他原因而阻塞。

2．wait()

当线程交互时，如果线程对一个同步对象 x 发出一个 wait()调用，该线程会暂停执行，被调对象进入等待状态，直到被唤醒或等待时间到。

面试例题 2：以下哪个方法不改变线程的状态?

A．start()　　B．run()　　C．isAlive()　　D．sleep()

解析：run()当然可以改变线程的状态，比如设置了优先级，线程可能转换为等待状态。而设置优先级的方法是 setPriority()。

如果有个线程 Thread thread = new Thread()，那么 thread.start()是启动新线程。thread.sleep()是将当前线程休眠（注意，sleep 是静态方法）。

答案：C

面试例题 3：在 Win32 环境中线程有 3 种基本模式，分别是什么？它们的关系和各自的优缺点是什么?

解析：在 Win32 环境中，线程有 3 种基本模式：单线程、单元线程和自由线程。

为了对线程模式有一定的了解，可以将其想象为从一间屋子搬到另一间屋子。如果采用单线程方法，则需要自己完成从打包到扛箱子，再到拆包的所有工作。如果使用单元线程模式，则表示邀请了好朋友来帮忙，每个朋友在一个单独的房间里工作，并且不能帮助在其他房间工作的人。他们各自负责自己的空间和空间内的物品搬运。如果采用自由线程方法，仍然邀请相同的朋友来帮忙，但是所有的朋友可以随时在任何一个房间工作，共同打包物品。与此类似，房子就是运行所有线程的进程，每个朋友都是一个代码实例，搬运的物品为应用程序的资源和变量。

不同线程模式有自己的优点和缺点。单元线程比单线程要快，因为有多个组件实例

在工作。在某些情况下，自由线程比单元线程更快、更有效，这是因为所有的事情同时发生，并且可以共享所有的资源。但是，当多线程更改共享资源时，这可能会出现问题。假设一个人开始使用箱子打包厨房用具，此时另一个朋友进来了，要使用同一个箱子打包浴室的东西。第一个朋友在箱子上贴上了“厨房用具”标签，另一个朋友用“洗漱用品”标签覆盖了原标签。结果，当你拆包时，就会发生将厨房用品搬到浴室的情况。

答案：3 种基本模式分别为单线程、单元线程和自由线程。

1．单线程

简单的应用程序很可能是单线程应用程序，仅包含与应用程序进程对应的线程。进程可以被定义为应用程序的实例，拥有该应用程序的内存空间。大多数 Windows 应用程序都是单线程的，即用一个线程完成所有的工作。

2．单元线程

单元线程是一种稍微复杂的线程模式。标记用于单元线程的代码可以在其自己的线程中执行，并限制在自己的单元中。线程可以被定义为进程所拥有的实体，处理时将调度该进程。在单元线程模式中，所有的线程都在主应用程序内存中各自的子段范围内运行。此模式允许多个代码实例同时但独立地运行。例如，在 .NET 之前，Visual Basic 仅限于创建单元线程组件和应用程序。

3．自由线程

自由线程是最复杂的线程模式。在自由线程模式中，多个线程可以同时调用相同的方法和组件。与单元线程不同，自由线程不会被限制在独立的内存空间。当应用程序必须进行大量相似而又独立的数学计算时，你可能需要使用自由线程。在这种情况下，你需要生成多个线程使用相同的代码示例来执行计算。可能 C++ 开发人员是仅有的编写过自由线程应用程序的应用程序开发人员，因为像 Visual Basic 6.0 这样的语言几乎不可能编写自由线程应用程序。

面试例题 4：设计 4 个线程，其中两个线程每次对 j 增加 1，另外两个线程每次对 j 减少 1。循环 100 次，写出程序。

答案：以下程序使用内部类实现线程，对 j 增减的时候没有考虑顺序问题。

```
public class ThreadTest1{
 private int j;
 public static void main(String args[]){
      ThreadTest1 tt=new ThreadTest1();
      Inc inc=tt.new Inc();
      Dec dec=tt.new Dec();
```

```
        for(int i=0;i<2;i++){
            Thread t=new Thread(inc);
            t.start();
            t=new Thread(dec);
            t.start();
        }
    }
    private synchronized void inc(){
        j++;
        System.out.println(Thread.currentThread().getName()+"-inc:"+j);
    }
    private synchronized void dec(){
        j--;
        System.out.println(Thread.currentThread().getName()+"-dec:"+j);
    }

    class Inc implements Runnable{
        public void run(){
            for(int i=0;i<100;i++){
                inc();
            }
        }
    }
    class Dec implements Runnable{
        public void run(){
            for(int i=0;i<100;i++){
                dec();
            }
        }
    }
}
```

面试例题 5：同步和异步有何异同，在什么情况下分别使用它们？举例说明。

答案：如果数据将在线程间共享，例如，正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程。在很多情况下采用异步途径往往更有效率。

面试例题 6：启动一个线程是用 run()还是 start()?

答案：启动一个线程是调用 start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由 JVM 调度并执行，但并不意味着线程就会立即运行。run()方法可以产生必须退出的标志来停止一个线程。

面试例题 7：简述线程的基本概念、线程的基本状态。

答案：线程指在程序执行过程中，能够执行程序代码的一个执行单位。每个程序至少都有一个线程，也就是程序本身。Java 中的线程有 4 种状态，分别是：运行、就绪、挂起、结束。

面试例题 8：简述 synchronized 和 java.util.concurrent.locks.Lock 的异同。

答案：主要相同点是，Lock 能完成 synchronized 所实现的所有功能。主要不同点是，Lock 有比 synchronized 更精确的线程语义和更好的性能。synchronized 会自动释放锁，而 Lock 一定要求程序员手工释放，并且必须在 finally 从句中释放。

面试例题 9：能否为线程设置一个易读的名字?

A．不能

B．能，可用 Thread.setName()

C．Java 自己把名字限制死了，都是 Thread=1 形式的

D．以上答案都不对

解析：可以使用 setName()方法给线程设置名字，如下所示：

```
Thread.currentThread().setName("hello");
System.out.println(Thread.currentThread().getName());
```

答案：B

面试例题 10：请问下面这段代码会不会输出 found?

```
public class Test {
 volatile static int test1 = 0;
 public static void main(String args[]) {
      Thread t1 = new TestThread("test1");
      Thread t2 = new TestThread("test2");
      t1.start();
      t2.start();
      try {
           t1.join();
           t2.join();
      } catch (Exception e) {
           e.printStackTrace();
           System.exit(-1);
      }
      System.out.println(Test.test1);
 }
```

```
}

class TestThread extends Thread {
 public TestThread(String n) {
      super(n);
 }

 public void run() {
      synchronized (Test.class) {
           for (int i = 0; i < 100000000; i++) {
                int oldV = Test.test1;
                Test.test1++;
                int newV = Test.test1;
                if (newV - oldV > 1) {
                     System.out.println("found");
                }

           }
      }
      System.out.println(this.getName() + "   thread   end   " + Test.test1);
 }
}
```

解析：首先要说明的是 TestThread 会对共享变量 test1 做 1000000 次以下操作：

```
int   oldV=MyTestVolatile.test1;
MyTestVolatile.test1++;
int   newV=MyTestVolatile.test1;
```

如果是单线程，肯定每次循环都是 oldV+1=newV，而且最终肯定是 MyTestVolatile.test1=1000000；

如果是在多线程下，且不加 Volatile，那么由于 MyTestVolatile.test1 的 local copy 做++操作，在大多数时候没有来得及执行 flush 的时候，两个线程保持不可见彼此的 local copy，所以每个线程内大多数时候肯定每次循环都是 oldV+1=newV，所以你运行不加 Volatile 的这个程序时，肯定绝大多数时候都不会执行 System.out.println("found")。

如果用 Volatile 的版本，会发现因为每次写操作都要在下一个读操作前 flush 进 master copy，所以出现 found 的概率大大提高。

如果更严格，执行 synchronized 操作，这样对 class 级别的排他锁不仅达到了 Volatile 的作用，而且由于完全排他，导致 MyTestVolatile.test1++的操作不会被并发进行，所以最终结果是绝对的 200000，而且绝对永远不会出现 found。

答案：本题执行 synchronized 操作，输出结果不会出现 found。

面试例题 11：下面哪些关键字通常用来对对象加锁，该标记使得对对象的访问是排他的？[中国杭州某著名 B2B 软件公司 T2009 年 10 月笔试题]

A．transient　　B．synchronized　　C．serialize　　D．static

解析：虽然本题只是一道选择题，但涉及知识面很广：Java 中对象的串行化、transient 关键字、锁机制。synchronized 关键字用于对象加锁。

答案：B

扩展知识：Java中对象的串行化（serialization）和transient关键字

对象的串行化

1．串行化的概念和目的

（1）什么是串行化。

对象寿命随着生成该对象的程序终止而终止。有时候，可能需要将对象的状态保存下来，在需要时再将对象恢复。对象的这种能记录自己的状态以便将来再生的能力叫做对象的持续性（persistence）。对象通过描述自己状态的数值来记录自己，这个过程叫对象串行化（serialization）。串行化的主要任务是写出对象实例变量的数值。如果变量是另一对象的引用，则引用的对象也要串行化。这个过程是递归的，串行化涉及一个复杂树结构的单行化，包括原有对象、对象的对象等。

（2）串行化的目的。

Java 对象的串行化的目标是为 Java 的运行环境提供一组特性，如下所示：

① 尽量保持对象串行化的简单扼要，但要提供一种途径使其可根据开发者的要求进行扩展或定制。

② 串行化机制应严格遵守 Java 的对象模型。对象的串行化状态中应该存有所有的关于种类的安全特性的信息。

③ 对象的串行化机制应支持 Java 的对象持续性。

④ 对象的串行化机制应有足够的可扩展能力以支持对象的远程方法调用（RMI）。

⑤ 对象串行化应允许对象定义自身的格式及其自身的数据流表示形式，可外部化接口来完成这项功能。

2．串行化方法

从 JDK 1.1 开始，Java 语言提供了对象串行化机制，在 java.io 包中，接口 Serialization 用来作为实现对象串行化的工具，只有实现了 Serialization 的类的

对象才可以被串行化。

Serializable 接口中没有任何方法。当一个类声明要实现 Serializable 接口时，只是表明该类参加串行化协议，而不需要实现任何特殊的方法。

3．串行化的注意事项

（1）串行化能保存的元素。

串行化只能保存对象的非静态成员变量，不能保存任何成员方法和静态的成员变量，而且串行化保存的只是变量的值，对于变量的任何修饰符都不能保存。

（2）transient 关键字。

对于某些类型的对象，其状态是瞬时的，这样的对象是无法保存其状态的。例如，一个 Thread 对象或一个 FileInputStream 对象，对于这些字段，我们必须用 transient 关键字标明，否则编译器将报措。

另外，串行化可能涉及将对象存放到磁盘上或网络上的数据，这时候就会产生安全问题。因为数据位于 Java 运行环境之外，不在 Java 安全机制的控制之中。对于这些需要保密的字段，不应简单地不加处理地保存下来，为了保证安全性，应该在这些字段前加上 transient 关键字。在 java.io 包中，接口 Serializable 用来作为实现对象串行化的工具，只有实现了 Serializable 的类的对象才可以被串行化。

transient 是 Java 语言的关键字，用来表示一个域不是该对象串行化的一部分。当一个对象被串行化的时候，transient 型变量的值不包括在串行化的表示中，非 transient 型的变量是被包括进去的。

下面是一个例子：当用户输入账号密码的时候，仅串行化账号，而不串行化密码。

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Date;

public class Test implements java.io.Serializable {

    private static final long serialVersionUID = 1L;
    private Date loggingDate = new Date();
    private String uid;
    private transient String pwd;

    Test(String user, String password) {
        uid = user;
        pwd = password;
```

```
    }

    public String toString() {
        String password = null;
        if (pwd == null) {
            password = "NOT SET";
        } else {
            password = pwd;
        }
        return "logon info: \n   " + "user: " + uid + "\n   logging date : "
                + loggingDate.toString() + "\n   password: " + password;
    }

    public static void main(String[] args) {

        Test logInfo = new Test("Jinder", "Rh-ab703");
        System.out.println(logInfo.toString());
        try {
            ObjectOutputStream o=new ObjectOutputStream(newFileOutputStream(
                    "logInfo.out"));
            o.writeObject(logInfo);
            o.close();
        } catch (Exception e) {// deal with exception

        }

        // To read the object back, we can write
        try {
            ObjectInputStream in = new ObjectInputStream(new FileInputStream(
                    "logInfo.out"));
            Test logInfo1 = (Test) in.readObject();
            System.out.println(logInfo1.toString());
        } catch (Exception e) {// deal with exception

        }
    }
}
```

扩展知识：Java中对象的synchronized关键字

多线程共享同一存储空间，在带来方便的同时，也会造成访问冲突。Java 语言提供了 synchronized 关键字以解决这种冲突，有效地避免了同一个数据对象被多个线程同时访问。使用 synchronized 关键字要注意以下几点：

（1）synchronized 关键字可以作为函数的修饰符，也可作为函数内的语句。synchronized 可作用于 instance 变量、object reference（对象引用）、static 函数和 class literals（类名称字面常量）身上。

（2）无论 synchronized 关键字加在方法上还是对象上，它取得的锁都是对象，而不是把一段代码或函数锁定，而且同步方法很可能还会被其他线程的对象访问。

（3）每个对象只有一个锁（lock）与之相关联。

（4）实现同步是要以很大的系统开销作为代价的，甚至可能造成死锁，所以尽量避免无谓的同步控制。

synchronized 关键字的作用域有两种：

（1）某个对象实例内，synchronized aMethod(){}可以防止多个线程同时访问这个对象的 synchronized 方法。如果一个对象有多个 synchronized 方法，只要一个线程访问了其中的一个 synchronized 方法，其他线程不能同时访问这个对象中任何一个 synchronized 方法。这时，不同的对象实例的 synchronized 方法是不相干扰的。也就是说，其他线程照样可以同时访问相同类的另一个对象实例中的 synchronized 方法。

（2）某个类的范围中，synchronized static aStaticMethod(){}防止多个线程同时访问这个类中的 synchronized static 方法。它可以对类的所有对象实例起作用。

synchronized 方法控制对类成员变量的访问：每个类实例对应一把锁，每个 synchronized 方法都必须获得调用该方法的类实例的锁方能执行，否则所属线程阻塞，方法一旦执行，就独占该锁，直到从该方法返回时才将锁释放，此后被阻塞的线程方能获得该锁，重新进入可执行状态。这种机制确保了同一时刻对于每一个类实例，其所有的声明为 synchronized 的成员函数中至多只有一个处于可执行状态（因为至多只有一个能够获得该类实例对应的锁），从而有效地避免了类成员变量的访问冲突（只要所有可能访问类成员变量的方法均被声明为 synchronized）。

下面举一个例子来加深印象：一个电影院有 20 张票要卖，它有 3 个售票员。我们用 sleep()函数来模拟售票，假设不使用 synchronized 关键字。

```
public class Test {
  public static void main(String[] args) {
      SellThread sell = new SellThread();
      Thread sell1 = new Thread(sell, "sellman1");
      Thread sell2 = new Thread(sell, "sellman2");
      Thread sell3 = new Thread(sell, "sellman3");
      sell1.start();
      sell2.start();
```

```
        sell3.start();
    }
}

class SellThread implements Runnable {
    private int i = 20;

    public void run() {
        while (true) {
            if (i > 0) {
                try {
                    Thread.sleep(100);
                } catch (Exception e) {
                }

                System.out.println(Thread.currentThread().getName() + " sell "
                        + i--);
            }
        }
    }
}
```

结果一共卖掉了 22 张票，估计电影院会为无缘无故多收门票钱而很高兴，不过一会也许就要面对愤怒的顾客了……

如果这时使用 synchronized 关键字，就可以避免这种冲突，修改程序如下：

```
public class Test {
    public static void main(String[] args) {
        SellThread sell = new SellThread();
        Thread sell1 = new Thread(sell, "sellman1");
        Thread sell2 = new Thread(sell, "sellman2");
        Thread sell3 = new Thread(sell, "sellman3");
        sell1.start();
        sell2.start();
        sell3.start();
    }
}

class SellThread implements Runnable {
    private int i = 20;
    String a = "now ok!";

    public void run() {
        while (true) {
            synchronized (a) {
                if (i > 0) {
                    try {
```

```
                    Thread.sleep(100);
                } catch (Exception e) {
                }

                System.out.println(Thread.currentThread().getName()
                        + " sell " + i--);
            }
        }
    }
}
```

这样只会卖 20 张票了，synchronized()的括号中需要的是一个 class 的对象，所以不能直接在括号中写上 i，于是这里定义了一个 String 的对象 a。a 可以认为是一个标志位，a 标志位默认任何人都能使用，这样售票员 sellman1 的卖票线程拿到了 a 以后就开始卖票了，同时他把 a 这个对象标志位置为不可用，然后其他售票员卖票的线程发现他们拿不到 a 这个对象，就只能先搁置了，一直到 sellmanl 的卖票线程释放了 a，a 的标志位就又变成可用，这个时候其他售票员（线程）就可以竞争了，看谁先拿到 a 这个对象，不过 String a 和卖票没什么关系，所以我们可以用 this 来代替 synchronized()中的 a，它和 a 的效果一样表示谁拿到了 this 对象才能执行。

这里有两个容易误解的地方：

（1）一个线程拿到 synchronized 括号中的对象之后，其他也需要拿到这个对象才能运行的线程不能被执行了。其实其他线程也是可以执行的，但它们执行到了需要 synchronized 中对象的时候，发现对象的标志位不可用，只能又被搁置了，所以 synchronized 使线程同步的时候是以牺牲效率为代价的。

（2）一个线程拿到 synchronized 括号中的对象之后，其他任何线程都不能执行了，假如其他不需要 synchronized 的对象才能继续执行的线程，还是可以和拿到 synchronized 的括号中的对象的线程一起运行的。有的方法在前面被加上了 synchronized，其实这个时候就是把这个方法的调用者即 this 的标志位置 0 了，这样就不能和其他需要 this 才能运行的线程一起执行，但可以和其他不需要这个 this 对象的线程一起运行。

面试例题 12：synchronized 关键字做同步，可以使用在什么上？[中国杭州某著名 B2B 软件公司 T2009 年 10 月笔试题]

A．方法上　　B．对象上　　C．原子类型　　D．AB 选项皆对

答案：D

第15章

数据库和SQL语言

数据库面试例题主要涉及范式、事物、存储过程、SQL语言及索引等诸方面。以下的考题来自真实的笔试资料，希望读者先不要看答案，自我解答后再与答案加以比对，找出自己的不足。

15.1 数据库理论问题

面试例题1：设有关系R(S,D,M)，其函数依赖集F={S→D,D→M}，则关系R至多满足_____。[美国著名搜索引擎公司Y2008年面试题]

A．1NF　　B．2NF　　C．3NF　　D．BCNF

解析：这是数据库模式的4个范式面试例题。

1NF：第一范式。如果关系模式R的所有属性的值域中每一个值都是不可再分解的值，则称R属于第一范式模式。如果某个数据库模式都是第一范式的，则称该数据库模式属于第一范式的数据库模式。

第一范式的模式要求属性值不可再分裂成更小部分，即属性项不能由属性组合和组属性组成。

2NF：第二范式。如果关系模式R为第一范式，并且R中每一个非主属性完全函数依赖于R的某个候选键，则称R为第二范式模式。如果某个数据库模式中每个关系模式都是第二范式的，则称该数据库模式属于第二范式的数据库模式。（注：如果A是关系模式R的候选键的一个属性，则称A是R的主属性，否则称A是R的非主属性。）

3NF：第三范式。如果关系模式R是第二范式，且每个非主属性都不传递依赖于

R 的候选键，则称 R 是第三范式的模式。如果某个数据库模式中的每个关系模式都是第三范式，则称为 3NF 的数据库模式。

BCNF：BC 范式。如果关系模式 R 是第一范式，且每个属性都不传递依赖于 R 的候选键，那么称 R 是 BCNF 的模式。

4NF：第四范式。设 R 是一个关系模式，D 是 R 上的多值依赖集合。如果 D 中成立非平凡多值依赖 X→→Y，X 必是 R 的超键，那么称 R 是第四范式的模式。

上题属于传递依赖，所以至多满足第二范式。

答案：B

面试例题 2：存储过程和函数的区别是什么？[美国著名搜索引擎公司 Y2008 年面试题]

答案：存储过程是用户定义的一系列 SQL 语句的集合，涉及特定表或其他对象的任务，用户可以调用存储过程。而函数通常是数据库已定义的方法，它接收参数并返回某种类型的值，而且不涉及特定用户表。

面试例题 3：What is database transaction?（什么是数据库事务？）

答案：数据库事务是指作为单个逻辑工作单元执行的一系列操作，这些操作要么全做，要么全不做，是一个不可分割的工作单位。

事务的开始与结束可以由用户显式地控制。如果用户没有显式地定义事务，则由 DBMS 按默认的规定自动划分事务。事务具有原子性、一致性、独立性及持久性等特点：

（1）事务的原子性是指一个事务要么全部执行，要么不执行。也就是说，一个事务不可能只执行一半就停止了。比如你从银行取钱，这个事务可以分成两个步骤：① 存折减款，② 拿到现金。不可能存折钱少了，而钱却没出来。这两步必须同时完成，要么都不完成。

（2）事务的一致性是指事务的运行并不改变数据库中数据的一致性。例如，完整性约束了 a+b=10，一个事务改变了 a，那么 b 也应该随之改变。

（3）事务的独立性是指两个以上的事务不会出现交错执行的状态。因为这样可能会导致数据不一致。

（4）事务的持久性是指事务运行成功以后，系统的更新是永久的，不会无缘无故地回滚。

面试例题 4：Explain the difference between clustered and non-clustered indexes. How does index affect the query?（解释聚簇索引和非聚簇索引之间的区别）

答案：经典教科书对聚簇索引的解释是：聚簇索引的顺序就是数据的物理存储顺序，而对非聚簇索引的解释是：索引顺序与数据物理排列顺序无关。正是因为如此，所以一个表最多只能有一个聚簇索引。

在 SQL Server 中，索引是通过二叉树的数据结构来描述的，我们可以这么理解聚簇索引：索引的叶节点就是数据节点。而非聚簇索引的叶节点仍然是索引节点，只不过有一个指针指向对应的数据块。

聚簇索引确定表中数据的物理顺序。聚簇索引类似于电话簿（电话簿按照字母簿排序），后者按姓氏排列数据。由于聚簇索引规定数据在表中的物理存储顺序，因此，一个表只能包含一个聚簇索引，但该索引可以包含多个列（组合索引），就像电话簿按姓氏和名字进行组织一样。

聚簇索引对于那些经常要搜索范围值的列特别有效。使用聚簇索引找到包含第一个值的行后，便可以确保包含后续索引值的行在物理相邻。例如，如果应用程序执行的一个查询经常检索某一日期范围内的记录，则使用聚簇索引可以迅速找到包含开始日期的行，然后检索表中所有相邻的行，直到到达结束日期。这样有助于提高此类查询的性能。同样，如果对从表中检索的数据进行排序时经常要用到某一列，则可以将该表在该列上聚簇（物理排序），避免每次查询该列时都进行排序，从而节省成本。

非聚簇索引与课本中的索引类似。数据存储在一个地方，索引存储在另一个地方，索引带有指针指向数据的存储位置。索引中的项目按索引键值的顺序存储，而表中的信息按另一种顺序存储（这可以由聚簇索引规定）。如果在表中未创建聚簇索引，则无法保证这些行具有任何特定的顺序。

有索引就一定检索得快吗？答案是否。有些时候用索引还不如不用索引快，比如说，我们要检索表中共 8000 条记录，如果不用索引，需要访问 8000 条×1000B/8KB=1000 个页面；如果使用索引，首先检索索引，访问 8000 条×10B/8KB=10 个页面，得到索引检索结果，再根据索引检索结果去对应数据页面，由于是检索所有的数据，所以，需要再访问 8000 条×1000B/8KB=1000 个页面，将全部数据读取出来，一共访问了 1010 个页面，这显然不如不用索引快。

面试例题 5：游标的作用是什么？如何知道游标已经到了最后？[中国著名金融软件公司 SZ 面试题]

答案：游标用于定位结果集的行。通过判断全局变量@@FETCH_ STATUS，可以判断游标是否到了最后，通常此变量不等于 0 表示出错或到了最后。

面试例题 6：触发器分为事前触发和事后触发，这两种触发有何区别？语句级触发和行

级触发有何区别？[中国著名金融软件公司 SZ 面试题]

答案：事前触发器运行于触发事件发生之前，而事后触发器运行于触发事件发生之后。通常，事前触发器可以获取事件之前和新的字段值。语句级触发器可以在语句执行前或后执行，而行级触发在触发器所影响的每一行触发一次。

面试例题 7：执行数据库查询时，如果要查询的数据有很多，假设有 1 000 万条，用什么办法可提高查询效率（速度）？在数据库方面或 Java 代码方面有什么优化的办法？

答案：

1．在数据库设计方面

（1）建立索引。

（2）分区（MySQL，比如按时间分区）。

（3）尽量使用固定长度的字段。

（4）限制字段长度。

2．在数据库 I/O 方面

（1）增加缓冲区。

（2）如果涉及表的级联，不同的表存储在不同的磁盘上，以增加 I/O 速度。

3．在 SQL 语句方面

（1）优化 SQL 语句，减少比较次数。

（2）限制返回的条目数（MySQL 中用 limit）。

4．在 Java 方面

如果是反复使用的查询，使用 PreparedStatement 减少查询次数。

15.2　SQL 语言常见问题

面试例题 1：列表如下：[美国著名数据库公司 SA2009 年 10 月笔试题]

A	B	C
3	7	9
4	8	4
5	6	9

① 选出 A、B 字段值，并按 B 的升序排列。

② 插入一条记录（7，9，8）。

③ 选出 C 值，并按 C 的降序排列，消除重复值。

写出三条标准的 SQL 语句实现上面的功能。

答案：代码如下：

```
if object_id('list')is not null drop table list
go
create table List(A int, B int,C int)
insert list select 3 , 7, 9
insert list select 4 , 8 , 4
insert list select 5, 6 , 9
--1
select a,b from list order by b asc
--2
insert list select 7,9,8
--3
select distinct c from list order by c desc
/*a          b
----------- -----------
5           6
3           7
4           8

c
-----------
9
8
4*/
```

面试例题 2：Given the following CREATE TABLE statement:（如下列表）

```
CREATE TABLE department
  (deptid INTEGER,
   deptname CHAR(25),
   budget NUMERIC(12,2))
```

Which of the following statement prevents two departments from being assigned the same DEPTID, but allows null values? （下列哪个选项防止两个部门（相同名称）被分配在 DEPTID 字段，但允许 NULL 值？）[美国著名数据库公司 SA2009 年 10 月笔试题]

A．ALTER TABLE department ADD CONSTRAINT dpt_cst PRIMARY KEY（改变 department 表，增加主键约束）

B．CREATE INDEX dpt_idx ON department(deptid)（创建 department 索引）

C．ALTER TABLE department ADD CONSTRAINT dpt_cst UNIQUE（创建唯一约束）

D．CREATE UNIQUE INDEX dpt_idx ON department(deptid) --（在 department 唯一索引）

解析：A 选项有主键，肯定不允许为 null。B、D 选项也显然不对。unique 约束能够约束一列保证该列值唯一，但允许该列有空值。故选 C。

答案：C

面试例题 3：Use the ERD to help you answer the following two questions（使用 ERD 图（如图 15-1 所示）回答以下两个问题）[美国著名数据库公司 SA2009 年 10 月笔试题]

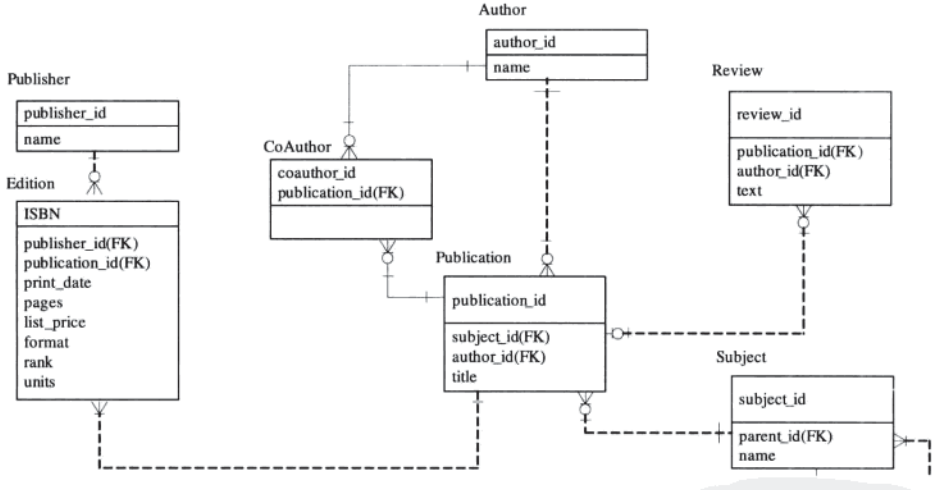


图 15-1　ERD 图

问 1：Find all ISBN values in the EDITION table, for all FORMAT='Hardcover' books。（在 EDITION 表中找到所有的 ISBN 表，其 FORMAT 值为 Hardcover 的书）

问 2：Find the sum of all LIST_PRICE values in the EDITION table, for each publisher。（在 EDITION 表中找出 LIST_PRICE 值的总和，以及每个 publisher 值）

答案：上面两题的 SQL 语言表达分别是：

```
问 1
select ISBN from EDITION where FORMAT='Hardcover'
问 2
select publisher_id,sum(LIST_PRICE) from EDITION group by publisher_id
```

面试例题 4：两个 SQL 表描述如下：[中国著名通讯公司 HW2009 年校园招聘面试题]

```
部门表[department]
2 个字段：id、name
编号(主键) 名称
---------------------
1 财务
2 销售
3 客服
员工表[employee]
3 个字段：id、name、id_department
编号(主键) 姓名 所属部门编号(外键)
--------------------------------------
1 张三 1
2 李四 1
3 王五 3
4 赵六 3
```

用一个 SQL 语句输出以下统计表：

```
编号(主键) 名称 员工数目
---------------------------------
1          财务  2
2          销售  0
3          客服  2
```

答案：编写代码如下：

```
select d.*,e.isnull(num,0) num from department d
left join
(
   select id_department,count(*) num from employee  group by id_department
)e on e.id_department =d.id
```

面试例题 5：假设你是 SQL Server 2005 服务器中 DB1 数据库的管理员。你收到一个警告说 DB1 的日志文件所在驱动器接近最大容量了。已知，虽然事务日志文件每 5 分钟进行一次备份，但磁盘空间占用还是在有规则的增长。因此，你认为可能是一个未提交的事务引起的。为了查明原因，你需要去确定在 DB1 数据库中，最早开始的活动事务的开始时间及服务器进程 ID，你应该如何做？[中国著名通讯公司 HW2009 年校园招聘面试题]

A．连接到 DB1 数据库，执行 DBCC OPENTRAN 命令，查看 SPID 和 Start time 行

B．连接到 master 数据库，执行 DBCC OPENTRAN，查看 SPID 和 Start time 行

C．用 SQL Server Management Studio 语句打开活动监视器，选择进程信息页，然后用数据库 = DB1 和打开的事务=yes 为条件进行筛选，查看结果中的进程 ID 和上一批列

D．打开查询窗口，连接到 master 数据库，执行如下 SQL 语句

```
SELECT TOP 1 spid, last_batch FROM sys.sysprocesses WHERE dbid = db_id('DB1')
AND open_tran > 0 ORDER BY last_batch
```

解析：

选项 A 正确。

选项 B 错误，DBCC OPENTRAN 不带参数默认为当前数据库。

选项 C 中，查询结果为所有当前未提交事务的进程中 last_batch（最近一次提交命令的时间）最小的进程。该进程不一定是题目要求的最早开始的事务。

选项 D 错误，理由同选项 C。

答案：A

第16章

计算机网络及分布式系统

网络面试例题主要涉及局域网、广域网和IP管理等诸方面。

以下的考题来自真实的笔试资料，希望读者先不要看答案，自我解答后再与答案加以对比，找出自己的不足。

16.1 网络结构

面试例题1：局域网甲内的主机A开启了P2P下载工具（如bt、emule等），它如何同局域网乙中的主机B建立连接？[中国杭州某网络公司A2009年9月笔试题]

A．通过主机B的内网IP建立连接

B．通过主机B的物理地址建立连接

C．通过NAT穿越技术建立连接

D．无法建立连接

解析：NAT（网络地址转换）是一种将一个IP地址域映射到另一个IP地址域的技术，从而为终端主机提供透明路由。NAT包括静态网络地址转换、动态网络地址转换、网络地址及端口转换、动态网络地址及端口转换、端口映射等。NAT常用于私有地址域与公用地址域的转换，以解决IP地址匮乏问题。在防火墙上实现NAT后，可以隐藏受保护网络的内部拓扑结构，在一定程度上提高网络的安全性。如果反向NAT提供动态网络地址及端口转换功能，还可以实现负载均衡等功能。

答案：C

面试例题2：使用视频软件进行聊天时，视频数据几乎都通过UDP协议传输。关于UDP协议，下列说法错误的是（　　）。[中国杭州某网络公司A2009年9月笔试题]

A．数据通过 UDP 协议传输存在丢包的可能，安全性不如 TCP 协议

B．UDP 协议传输执行速度一定比 TCP 快

C．UDP 协议的数据传输是面向无连接的，TCP 协议的数据传输是面向有连接的。

D．视频、聊天等数据的传输都可以使用 UDP 协议。

解析：TCP 和 UDP 的差别在于适用性上，并非在于速度上。泛泛地去说谁比谁快都是不恰当的。特定环境下造成的差别只能具体情况具体分析，TCP 在启动速度上较 UDP 缓慢（同样的情况也发生在 HTTP 上）。如果是大文件，TCP 的优势就体现了（但也未必就比 UDP 快）。从这里也看出来，UDP 是一个很“贪婪”的协议，容易造成网络拥塞，影响不同业务流的公平性。

至于 TFTP 和 FTP，TFTP 省去了协商的步骤，也没有什么主动链接和被动链接的麻烦，每次固定传输 512B，但是局限性就在于不能传输大文件，最多传输 32MB，这是因为 block 号从 0 开始到 65535 结束，当超过 65535 后又从 0 开始。FTP 如果在线路好的情况下会“跑”得越来越快，直到极限，因为 TCP 的载荷可达 1460B 或许更多。

TCP和UDP速度的问题要在不同的要求下进行不同的讨论，比如在高可用线路上，UDP肯定更快、更有效率，如果对传输保证要求高，肯定会用TCP，当然线路不稳定会适得其反，容易引起拥塞。还有就是取决于上层的应用，如果视频流用上TCP，则效率会低点，当线路不好的时候或许比用UDP传输的视频画面还要模糊，既然上层视频解码有容错机制，那么就可以用UDP传输。

答案：B

扩展知识：UDP和TCP在应用上的区别

考虑到 3 个方面：应用类型、系统资源开销、网络实际状况。

（1）从**网络传输角度**看应用，应用类型可分成若干种。

- 延迟敏感（VoIP 等）。
- 带宽敏感（例如文件传输、备份、大数据量 SQL 等）。
- 群组通讯（IPTV、路由公告等）。

（2）系统资源开销。

- 计算密集型（科学计算、数据库检索等）。
- 资源集约型（嵌入设备管理维护等）。
- 大规模并发用户类型（QQ、MSN 等）。

（3）网络状况。

- 任务型（企业内部网络等）。
- 混杂型（Internet 等）。

- 故障型（卫星链路等）。

实际应用中，延迟敏感型会偏向于 UDP，因为 TCP 并无延迟控制良策，反而通过自定义的协议栈和应用缓冲机制，有机会提供比较好的效果；带宽敏感型会比较偏向于 TCP，因为应用开发要达到最佳效果，通常会更加专注于文件系统的缓存优化，而把传输任务交送给 TCP 处理；群组通讯型会比较偏向于 UDP，因为组播通讯并没有确定的回应人，所以组播传输几乎完全基于 UDP

计算机密集型和带宽敏感型类似，通常会偏向于 TCP，因为希望上层优化开发需要花费的时间更多，通常会把传输交给 TCP。DNS 协议也属于这个类型，但属于一个比较例外。它既采用 UDP，又采用 TCP，主要考虑到 DNS 的 Client 很多是延迟敏感型和资源集约型类型；资源集约型相对会偏向于 UDP，因为剥离 TCP 堆栈以后，内核可以简化很多。一些简化的控制机制也很容易在 UDP based 协议上实现，例如，SNMP 这类协议广泛采用 UDP；大规模并发用户类型不同于技术流派的问题，当初 QQ 采用 UDP 4000/8000 并非仅仅考虑到系统负载之类的状况，而是因为混杂型、故障型的原因更多，就好像 MSN 采用 TCP，同样也可以承担大规模并发用户。但是 QQ 的设计实践很好地适应了中国教育网、电信、网通互联的故障型状况，在最短时间内获得了最大批的用户。而现在，QQ 很多情形下都采用 TCP 80 443 来穿越防火墙，主要通讯已经成为 TCP 行为。

任务型通常会按照具体任务来确定，TCP 是首选；混杂型属于个人爱好，例如，P2P 的应用，用 UDP、TCP 均可以写好；故障型通常会采用改进协议栈的 TCP 或者 UDP。因为丢包率、延迟抖动、乱序等问题的存在，标准的 TCP 的时钟、重传机制未必能够很好适应。这时候就会修改 TCP 协议栈或者采用自定义开发协议栈（通常基于 GRE 或者 UDP）。在这种情况下，使用 UDP 其实并没有特别的性能优势，但由于 UDP 广泛被各类网络允许通过，所以才使用 UDP。

可以看到，从传输性能角度考虑，通常不是选择 UDP 的理由，而更容易选择 TCP，因为应用开发人员可以更集中精力提高上层文件处理性能和管理效率，而不必为了有限的潜在性能而去优化协议栈，就好像 TFTP，TFTP 的“T”代表琐碎、不重要的意思，即 TFTP 设计用来传输一些零碎的物件，所以，保障可用性高于性能需求。

面试例题 3：Which one is Class B Address?（下列属于 B 类地址是哪个？）[美国某著名数据库公司 SA2009 年 11 月笔试题]

A．10.10.10.1　　B．191.168.0.1　　C．192.168.0.1　　D．202.113.0.1

解析：A 类 IP 地址由 1B（每个字节是 8 位）的网络地址和 3B 的主机地址组成，

网络地址的最高位必须是“0”，即第一段数字范围为 1～127。每个 A 类地址可连接 16 387 064 台主机，Internet 有 126 个 A 类地址。

B 类 IP 地址由 2B 的网络地址和 2B 的主机地址组成，网络地址的最高位必须是“10”，即第一段数字范围为 128～191。每个 B 类地址可连接 64 516 台主机，Internet 有 16 256 个 B 类地址。

C 类地址是由 3B 的网络地址和 1B 的主机地址组成，网络地址的最高位必须是“110”，即第一段数字范围为 192～223。每个 C 类地址可连接 254 台主机，Internet 有 2 054 512 个 C 类地址。

D 类地址用于多点播送，第一个字节以“1110”开始，第一个字节的数字范围为 224～239，是多点播送地址，用于多目的地信息的传输或作为备用。全零（“0.0.0.0”）地址对应于当前主机，全“1”的 IP 地址（“255.255.255.255”）是当前子网的广播地址。

E 类地址以“11110”开始，即第一段数字范围为 240～254。E 类地址保留，仅作为实验和开发用。

答案：B

16.2　TCP/IP

面试例题 1：If we divide the network 40.15.0.0 into two subnets, and the first one is 40.15.0.0/17, then the second subnet will be ____ .（如果把一个网络 40.15.0.0 分为两个子网，第一个子网是 40.15.0.0/17，那么第二个子网将会是_____。）[中国台湾杀毒软件公司 2005 年 10 月面试题]

A．40.15.1.0/17　　B．40.15.2.0/16

C．40.15.100.0/17　　D．40.15.128.0/17

解析：让主网分成两个网段，子网掩码分别是 0xff 0xff 0x80 0x00 和 0xff 0xff 0x00 0x00。

答案：D

面试例题 2：If a worm scans the hosts in Class A IP address space on a home PC, it is quite probably that the host will received a lot of ____.（如果一个蠕虫病毒攻击了一个家用 PC 机的 A 类地址主机，这个地址主机最有可能接收很多_____。）[中国台湾杀毒软件公司 2005 年 10 月面试题]

A．HTTP response packet（HTTP 回应包）

B．DNS response packet（DNS 回应包）

C．ICMP destination unreachable packet（网间控制报文协议目的无法抵达包）

D．ARP response（地址解析协议回应）

解析：大量发出的 IP 请求肯定很多不可达到，返回不可达到错误。

答案：C

面试例题 3：Before an IP datagram arrived at the destination, it ____ .（在一个 IP 数据包到达目的地址之前，它____。）[中国台湾杀毒软件公司 2005 年 10 月面试题]

A．may be fragmented but never reassembled（可能成为碎片，但是不会重组）

B．may be fragmented or reassembled（可能成为碎片，或者重组）

C．can't be fragmented or reassembled（不能成为碎片，或者重组）

D．can't be fragmented but may be reassembled（不能成为碎片，但是可能会重组）

解析：包未达到终点不可能重组，但可以分散成碎片。

答案：A

面试例题 4：In TCP/IP protocol stack, which of following is taken as an indication of congestion?（在 TCP/IP 协议栈里，如果出现阻塞情况，下面哪种情况最有可能发生？）[中国台湾杀毒软件公司 2005 年 10 月面试题]

A．Link failure（连接错误）　　B．Free buffer（释放缓存）

C．Packet loss（丢包）　　D．Packet error（包错误）

解析：本题涉及网络阻塞问题，阻塞导致丢包。

答案：C

面试例题 5：Which protocol is FTP based on for file transfer?（文件传输基于下列哪种协议？）[中国台湾杀毒软件公司 2005 年 10 月面试题]

A．TCP　　B．UDP

C．Both A and B　　D．None of above

解析：FTP 是有连接的服务，所以必须基于 TCP 协议。

答案：A

面试例题 6：What is the maximum number of host can exist in a class C network?（一个 C 类网络最多能容纳多少台主机？）[中国台湾杀毒软件公司 2005 年 10 月面试题]

A．256　　B．254　　C．255　　D．128

解析：子网中 IP 为 0～255，其中 0 和 255 不能用，所以是 254 个。

答案：B

面试例题 7：Which of the following is the E-mail transfer protocol of the Internet?（下面哪一个缩写是电子邮件传输协议？）[中国台湾杀毒软件公司 2005 年 10 月面试题]

A．HTTP　　B．POP　　C．SSL　　D．SMTP

解析：SMTP 的全称是“Simple Mail Transfer Protocol”，即简单邮件传输协议。它是一组用于从源地址到目的地址传输邮件的规范，通过它来控制邮件的中转方式。SMTP 协议属于 TCP/IP 协议簇，它帮助每台计算机在发送或中转信件时找到下一个目的地。SMTP 服务器就是遵循 SMTP 协议的发送邮件服务器。

答案：D

面试例题 8：Ethernet switch forwarding packet bases on_____。（以太网转换控制包是基于_____。）[中国台湾杀毒软件公司 2005 年 10 月面试题]

A．Destination MAC address（目的网卡地址）

B．Source MAC address（源网卡地址）

C．Destination IP address（目的 IP 地址）

D．Source and destination IP address（源和目的 IP 地址）

答案：C

面试例题 9：If the TCP based server program crashed before the client data arrived on the connection that established earlier, the TCP/IP stack may return a ____ .（如果 TCP 服务器在客户端发出数据报之前已经崩溃了，TCP/IP 栈可能返回一个_____。）[中国台湾杀毒软件公司 2005 年 10 月面试题]

A．RST　　B．FIN　　C．SYN　　D．ACK

解析：SYN 包是 TCP 连接的第一个包，是非常小的一种数据包。SYN 攻击包括大量此类的包，由于这些包看上去来自实际不存在的站点，因此无法有效地进行处理。SYN 攻击就是利用 TCP 连接的三次握手机制，发起攻击端只来一、两次握手，而被攻击端就一直在试图完成 TCP 连接，造成资源不足。

答案：C

面试例题 10：在 Windows 2000 操作系统中，配置 IP 地址的命令是__(1)__。若用 ping 命令来测试本机是否安装了 TCP/IP 协议，则正确的命令是__(2)__。如果要列出本机当前建立的连接，可以使用的命令是__(3)__。[中国大陆某杀毒软件公司 2010 年 1 月面试题]

(1) A．winipcfg　　B．ipconfig　　C．ipcfg　　D．winipconfig

(2) A．ping 127.0.0.0　　B．ping 127.0.0.1　　C．ping 127.0.1.1　　D．ping 127.1.1.1

（3）A．netstat -s　　B．netstat –o　　C．netstat -a　　D．netstat -r

答案：B，B，C

面试例题 11：发起 TCP 连接的三次握手中的第一次发送的包类型为下列哪个？[中国大陆某杀毒软件公司 2010 年 1 月面试题]

A．SYN　　B．ACK　　C．FIN　　D．RST

解析：在 TCP/IP 协议中，TCP 协议提供可靠的连接服务，采用三次握手建立一个连接。

第一次握手：建立连接时，客户端发送 SYN 包（SYN=j）到服务器，并进入 SYN_SEND 状态，等待服务器确认。

第二次握手：服务器收到 SYN 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（SYN=k），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态。

第三次握手：客户端收到服务器的 SYN＋ACK 包，向服务器发送确认包 ACK (ack=k+1)，此包发送完毕，客户端和服务器进入 ESTABLISHED 状态，完成三次握手。

完成三次握手后，客户端与服务器开始传送数据，在上述过程中，还有一些重要的概念。

（1）未连接队列：在三次握手协议中，服务器维护一个未连接队列，该队列为每个客户端的 SYN 包（SYN=j）开设一个条目，该条目表明服务器已收到 SYN 包，并向客户发出确认，正在等待客户的确认包。这些条目所标识的连接在服务器处于 Syn_RECV 状态，当服务器收到客户的确认包时，删除该条目，服务器进入 ESTABLISHED 状态。

（2）* SYN：同步标志。

同步序列编号（Synchronize Sequence Numbers）栏有效。该标志仅在三次握手建立 TCP 连接时有效。它提示 TCP 连接的服务端检查序列编号，该序列编号为 TCP 连接初始端（一般是客户端）的初始序列编号。在这里，可以把 TCP 序列编号看做是一个范围从 0 到 4、294、967、295 的 32 位计数器。通过 TCP 连接交换的数据中每一个字节都经过序列编号。在 TCP 报头中的序列编号栏包括 TCP 分段中第一个字节的序列编号。

（3）*ACK：确认标志。

确认编号（Acknowledgement Number）栏有效。大多数情况下，该标志位是置位的。TCP 报头内的确认编号栏内包含的确认编号（w+1，Figure-1）为下一个预期的序列编号，同时提示远端系统已经成功地接收所有的数据。

（4）*RST：复位标志。

复位标志有效。用于复位相应的 TCP 连接。

(5) *URG：紧急标志。

紧急（The urgent pointer） 标志有效。紧急标志置位。

(6) *PSH：推标志。

该标志置位时，接收端不将该数据进行队列处理，而是尽可能快地将数据转由应用处理。在处理 telnet 或 rlogin 等交互模式的连接时，该标志总是置位的。

(7) *FIN：结束标志。

带有该标志置位的数据包用来结束一个 TCP 会话，但对应端口仍处于开放状态，准备接收后续数据。

答案：A

16.3　网络其他问题

面试例题 1：网络中常见的 ping 命令基于什么协议？[中美合资通讯软件公司 HS 面试题]

解析：ICMP 是“Internet Control Message Protocol”（Internet 控制消息协议）的缩写。它是 TCP/IP 协议族的一个子协议，用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。

在网络中经常会使用到 ICMP 协议，只不过觉察不到而已。比如经常使用的用于检查网络通不通的 ping 命令，这个“ping”的过程实际上就是 ICMP 协议工作的过程。还有其他的网络命令，如跟踪路由的 Tracert 命令，也是基于 ICMP 协议的。

ICMP 协议对于网络安全具有极其重要的意义。ICMP 协议本身的特点决定了它非常容易被用于攻击网络上的路由器和主机。

比如，可以利用操作系统规定的 ICMP 数据包最大尺寸不超过 64KB 这一规定，向主机发起“Ping of Death”（死亡之 Ping）攻击。“Ping of Death” 攻击的原理是：如果 ICMP 数据包的尺寸超过 64KB 上限，主机就会出现内存分配错误，导致 TCP/IP 堆栈崩溃，致使主机死机。

此外，向目标主机长时间、连续、大量地发送 ICMP 数据包，也会最终使系统瘫痪。大量的 ICMP 数据包会形成“ICMP 风暴”，使得目标主机耗费大量的 CPU 资源处理，疲于奔命。

虽然 ICMP 协议给黑客以可乘之机，但是 ICMP 攻击也并非“无药可医”。只要在日常网络管理中未雨绸缪，提前做好准备，就可以有效地避免 ICMP 攻击造成的损失。

对于“Ping of Death”攻击，可以采取两种方法进行防范。第一种方法是在路由器上对 ICMP 数据包进行带宽限制，将 ICMP 占用的带宽控制在一定的范围内，这样即使

有 ICMP 攻击，它所占用的带宽也是非常有限的，对整个网络的影响非常小；第二种方法就是在主机上设置 ICMP 数据包的处理规则，最好是设定拒绝所有的 ICMP 数据包。

答案：ping.exe 的原理是，向指定的 IP 地址发送一定长度的数据包，按照约定，若指定的 IP 地址存在的话，会返回同样大小的数据包，当然，若在特定的时间内没有返回，就是“超时”，就认为指定的 IP 地址不存在。由于 ping 使用的是 ICMP 协议，有些防火墙软件会屏蔽掉 ICMP 协议，所以有时候 ping 的结果只能作为参考，ping 不通并不能就一定说明对方 IP 不存在。

ping 命令是一个非常有用的网络命令，大家常用它来测试网络联通情况。但同时它也是把“双刃剑”，别人使用 ping 命令能探测到你计算机上的很多敏感信息，造成不安全。出于安全考虑，防止 ping 的方法也有很多，比如防火墙，又比如创建一个禁止所有计算机 ping 本机 IP 地址的安全策略。

由于 ping 使用的是 ICMP 协议，有些防火墙软件会屏蔽掉 ICMP 协议。IPSec 安全策略是关于如何“防 ping”的，其原理是通过新建一个 IPSec 策略来过滤掉本机所有的 ICMP 数据包实现的。这样，确实是可以有效地“防 ping”，但同时也会留下后遗症。因为 ping 命令和 ICMP 协议（Internet Control Message Protocol）有着密切的关系，在 ICMP 协议的应用中包含有 11 种报文格式，其中 ping 命令就是利用 ICMP 协议中的“Echo Request”报文进行工作的。但 IPSec 安全策略防 ping 时采用格杀勿论的方法，把所有的 ICMP 报文全部过滤掉，特别是很多有用的其他格式的报文也同时被过滤掉了。因此，在某些有特殊应用的局域网环境中，容易出现数据包丢失的现象，影响用户正常办公。建议使用防火墙。

面试例题 2：已知当前局域网网关的 IP 为 192.168.1.1，在局域网上一台计算机执行 ping 192.168.1.1 命令得到以下结果：[美国著名综合软件公司 H2010 年 1 月面试题]

Reply from 192.168.1.1:bytes=32 time<1ms TTL=255

那么我们可以判断从网关服务器发出的网络包，最多被传递（）级路由器后就被抛弃。

A．32　　B．1　　C．255　　D．8

解析：看 TTL 值，每经过一个路由器 TTL 减 1，到 0 时就丢弃。

答案：C

面试例题 3：下面关于 TCP/IP 协议中 TCP 和 UDP 的表述，正确的是（　　）？[美国著名综合软件公司 H2010 年 1 月面试题]

A．TCP 包头部有源端口号，UDP 包头部没有源端口号

B．TCP 包头部有校验码（Checksum），UDP 包头部没有校验码

C．TCP 和 UDP 包都是固定长度的，由协议栈的 MTU 决定

D．TCP 和 UDP 包头部结构中都不包含目的地址的 IP

解析：TCP、UDP 为传输层协议，不包含 IP 地址，IP 协议中才有 IP 地址。

答案：D

面试例题 4：An application-gateway:(M)（多选：一个应用网关（　　）？）[美国著名综合软件公司 H2010 年 1 月面试题]

A．can be a proxy-server（可以是一个代理服务器）

B．can only check the IP-info of messages send through the network（只能检查 IP 信息）

C．can be of use to link applications outside the company network with applications used within the company network（可以被用来连接公司内部网络应用和公司外部网络应用）

D．can NEVER be part of a Firewall construct（永远不可能是防火墙构造的一部分）

解析：应用网关当然可以是代理服务器，所以选项 A 正确；至于选项 B，显然错误，应用网关不只可以检查 IP 信息；选项 C 也是正确的；选项 D 显然错误，应用网关当然可以是防火墙构造的一部分。

答案：A C

面试例题 5：What is the default network mask for a class A network?（A 类网络默认的子网掩码是多少？）[美国著名综合软件公司 H2010 年 1 月面试题]

A．255.255.255.255　　B．255.255.255.0

C．255.255.0.0　　D．255.0.0.0

解析：IP 地址有 5 种类型，由网络号的第一组数字来表示。

1．网络分类

A 类地址的第一组数字为 1～126。注意，数字 0 和 127 不作为 A 类地址，数字 127 保留给内部回送函数，而数字 0 则表示该地址是本地宿主机，不能传送。

B 类地址的第一组数字为 128～191。

C 类地址的第一组数字为 192～223。

D 类 IP 地址的第一段数字范围为 224～239，D 类地址用做多目的地信息的传输，作为备用。

E 类 IP 地址的第一段数字范围为 240～254，E 类地址保留，仅作为 Internet 的实验和开发之用。

例如：某校的网络号是 202.206.64～79，它的第一组数字为 202，因此 202.206.64.34 是 C 类地址。而 159.266.1.1 则是 B 类地址。

“IP 地址在全世界范围内唯一”。看到这句话你可能有这样的疑问，像 192.168.0.1 这样的地址在许多地方都能看到，并不唯一，这是为何？Internet 管理委员会规定如下地址段为私有地址，私有地址可以自己组网时用，但不能在 Internet 网上用，Internet 网没有这些地址的路由，有这些地址的计算机若要上网，必须转换成为合法的 IP 地址（也称为公网地址），这就像有很多世界公园，每个公园内都可命名相同的大街，如香榭丽舍大街，但是这些大街只是在本公园里不重名。下面是 A、B、C 类网络中的私有地址段：

```
10.0.0.0~10.255.255.255
172.16.0.0~172.131.255.255
192.168.0.0~192.168.255.255
```

2．子网掩码

子网掩码不能单独存在，它必须结合 IP 地址一起使用。子网掩码只有一个作用，就是将某个 IP 地址划分成网络地址和主机地址两部分。

子网掩码的设定必须遵循一定的规则。与 IP 地址相同，子网掩码的长度也是 32 位，左边是网络位，用二进制数字“1”表示；右边是主机位，用二进制数字“0”表示，其中“1”有 24 个，代表与此相对应的 IP 地址左边 24 位是网络号；“0”有 8 个，代表与此相对应的 IP 地址右边 8 位是主机号。这样，子网掩码就确定了一个 IP 地址的 32 位二进制数字中哪些是网络号，哪些是主机号。这对于采用 TCP/IP 协议的网络来说非常重要，只有通过子网掩码，才能表明一台主机所在的子网与其他子网的关系，使网络正常工作。

子网掩码可以确定一个网络层地址哪一部分是网络号，哪一部分是主机号。子网掩码的作用就是获取主机 IP 的网络地址信息，用于区别主机通信根据不同的情况选择不同的路径。其中 A 类网络的子网掩码为 255.0.0.0；B 类网络为 255.255.0.0；C 类网络地址为：255.255.255.0。

通过 IP 地址的二进制与子网掩码的二进制进行与运算来确定某个设备的网络地址，也就是说，通过子网掩码分辨一个网络的网络部分和主机部分。子网掩码一旦设置，网络地址和主机地址就固定了。与 IP 地址相同，子网掩码的长度也是 32 位，也可以使用十进制的形式。例如，二进制形式的子网掩码：11111111111111111111100000000，采用十进制的形式为：255.255.255.0。

3．子网掩码和 IP 地址的关系

子网掩码是用来判断任意两台计算机的 IP 地址是否属于同一子网络的根据。

最简单的理解就是两台计算机各自的 IP 地址与子网掩码进行 AND（与）运算后，如果得出的结果是相同的，则说明这两台计算机是处于同一个子网络上的，可以进行直接的通信。

运算演示之一：

```
IP 地址   192.168.0.1
子网掩码   255.255.255.0
AND 运算
转化为二进制数进行运算：
IP 地址  11000000.10101000.00000000.00000001
子网掩码  11111111.11111111.11111111.00000000
AND 运算后
11000000.10101000.00000000.00000000
转化为十进制数后为：
192.168.0.0

运算演示之二：
IP 地址   192.168.0.254
子网掩码   255.255.255.0
AND 运算
转化为二进制数进行运算：
IP 地址  11000000.10101000.00000000.11111110
子网掩码  11111111.11111111.11111111.00000000
AND 运算后
11000000.10101000.00000000.00000000
转化为十进制数后为：
192.168.0.0

运算演示之三：
IP 地址   192.168.0.4
子网掩码   255.255.255.0
AND 运算
转化为二进制数进行运算：
IP 地址  11000000.10101000.00000000.00000100
子网掩码  11111111.11111111.11111111.00000000
AND 运算
11000000.10101000.00000000.00000000
转化为十进制数后为：
192.168.0.0
```

通过以上对三组计算机 IP 地址与子网掩码的 AND 运算后，可以看到它的运算结果是一样的，均为 192.168.0.0。所以，计算机就会把这三台计算机视为同一子网络，然后进行通信。一般的代理服务器的内部网络就是这样规划的。

4．可用 IP 地址

局域网内部的 IP 地址是我们自己规定的（当然和其他的 IP 地址是一样的），这个是由子网掩码决定的。通过对 255.255.255.0 的分析可知：前三位 IP 码分配的一个 C 类网络是固定的；能变化的就只剩下最后的一位了，那么显而易见的，IP 地址只能有 2^8-1，即 256–1=255，一般末位为 0 或者是 255 的都有其特殊的作用。

假设你的子网掩码是 255.255.128.0，那么你的局域网内 IP 地址的前两位肯定是固

定的。这样，你就可以按照下面的计算来看看同一个子网内到底能有多少台机器：

```
① 10 进制数 128 = 二进制数 10000000
② IP 码要和子网掩码进行 AND 运算
③
IP 地址  11000000.10101000.1*******.********
子网掩码  11111111.11111111.10000000.00000000
AND 运算
11000000.10101000.10000000.00000000
转化为十进制数后为：
192 . 168. 128 . 0

④ 可知内部网可用的 IP 地址为：
11000000.10101000.10000000.00000000  到
11000000.10101000.11111111.11111111

⑤ 转化为十进制数：
192.168.128.0 到 192.168.255.255

⑥ 0 和 255 通常作为网络的内部特殊用途，且通常不使用。

⑦ 于是最后的结果（可用的 IP 地址）为：
192.168.128.1～192.168.128.254
192.168.129.1～192.168.129.254
192.168.130.1～192.168.130.254
192.168.131.1～192.168.131.254
……
192.168.139.1～192.168.139.254
192.168.140.1～192.168.140.254
192.168.141.1～192.168.141.254
192.168.142.1～192.168.142.254
192.168.143.1～192.168.143.254
……
192.168.254.1～192.168.254.254
192.168.255.1～192.168.255.254

⑧ 内网中可用的 IP 地址总数为(255-128+1)×(254-1+1) =128×254 = 32512
```

答案：D

面试例题 6：在 TCP/IP 协议中，采用_____来区分不同的应用进程。[美国著名综合软件公司 H2010 年 1 月面试题]

A．端口号　　B．IP 地址　　C．协议类型　　D．MAC 地址

解析：区分不同进程用的是端口号。根据 TCP/IP 分层模型，进程属于应用层，依靠端口号来区分。而 IP 地址是区分不同主机的，TCP/UDP 是区分传输层的。

答案：A

第 5 部分

Java 开源

Java opensource

EJB 组件曾经被认为是一个重量级的组件。EJB 3.0 规范的重要目标就是简化 EJB 的开发，提供一个相对轻量级的组件方案。Spring 基于轻量内核，然后通过集成第三方的服务器来提供完整的架构。

轻量级的组件，并不意味着提供服务的容器是轻量的。对于 Spring，你需要寻找并粘合各种服务，然后让它们能够稳定地在一起工作，如果应用对技术的需求较多，伸缩性要求也较高，你就会不断地在应用服务中加入其他服务，如资源池、集群等。当加入这些以后，Spring 的解决方案已经同 Java EE Application Server 解决方案一样重量级了。

其实，当 EJB 3.0 推出，当 Spring 2.0 的程序需要特别的 Javac 进行编译时，重和轻模糊了。

追求简单、轻量，是每一个应用架构的目标。对于企业应用的构建来说，不论轻重，只有架构合适的应用平台，才能最终适应项目的需要。

第17章

J2EE 技术

从整体上讲，J2EE 是使用 Java 技术开发企业级应用的一种事实上的工业标准（Sun 公司出于其自身利益的考虑，至今没有将 Java 及其相关技术纳入标准化组织的体系），它是 Java 技术在不断适应和促进企业级应用过程中的产物。目前，Java 平台有 3 个版本：适用于小型设备和智能卡的 J2ME（Java 2 Platform Micro Edition）、适用于桌面系统的 J2SE 和适用于企业级应用的 J2EE。Sun 推出 J2EE 的目的是为了消除传统 Client/Server 模式的弊病，迎合 Browser/Server 架构的潮流，为应用 Java 技术开发服务器端应用提供一个平台独立的、可移植的、多用户的、安全的和基于标准的企业级平台，从而简化企业应用的开发、管理和部署。J2EE 是一个标准，而不是一个现成的产品。各个平台开发商按照 J2EE 规范分别开发了不同的 J2EE 应用服务器，J2EE 应用服务器是 J2EE 企业级应用的部署平台。由于它们都遵循了 J2EE 规范，因此，使用 J2EE 技术开发的企业级应用可以部署在各种 J2EE 应用服务器上。

为了推广并规范化使用 J2EE 架构企业级应用的体系架构，Sun 同时给出了一个建议性的 J2EE 应用设计模型：J2EE Blueprints。J2EE Blueprints 提供了实施 J2EE 企业级应用的体系架构、设计模式和相关的代码，通过应用 J2EE Blueprints 所描述的体系模型，能够部分简化架构企业级应用这项复杂的工作。J2EE Blueprints 是开发人员设计和优化 J2EE 组件的基本原则，同时为围绕开发工作进行职能分工给出了指导性策略，以帮助应用开发设计人员合理地分配技术资源。

17.1 Spring 轻量级架构

面试例题 1：Spring 都有哪些特点？为什么要使用 Spring？

解析：先解释两个概念。

- 控制反转（Inverse of Control，IOC）。
- 依赖注入（Dependency Injection，DI）。

控制反转是对组件对象控制权的转移，从程序代码本身转移到了外部容器，通过容器来实现对象组件的装配和管理。

注入类型：接口注入、设置注入、构造子注入。而 Spring 则是 IOC 的一个容器。

在 Spring 中，所谓依赖注入，即在运行期由容器将依赖关系注入到组件之中，就是在运行期，由 Spring 根据配置文件，将其他对象的引用通过组件提供的 setter 方法进行设定。

从核心而言，Spring 是一个 DI 容器，其设计哲学是提供一种无侵入式的高扩展性框架。即无须代码中涉及 Spring 专有类，即可将其纳入 Spring 容器进行管理。

作为对比，EJB 则是一种高度侵入性的框架规范，它制定了众多的接口和编码规范，要求实现者必须遵从。侵入性的结果就是，一旦系统基于侵入性框架设计开发，那么之后任何脱离这个框架的企图都将付出极大的代价。为了避免这种情况，实现无侵入性的目标，Spring 大量引入了 Java 的 Reflection 机制，通过动态调用的方式避免硬编码方式的约束，并在此基础上建立了其核心组件 BeanFactory，以此作为其依赖注入机制的实现基础。

org.Springframework.beans 包中包括了这些核心组件的实现类，核心中的核心为 BeanWrapper 和 BeanFactory 类。这两个类从技术角度而言并不复杂，但对于 Spring 框架而言，却是关键所在。通过在配置文件中加以设定，就可以在运行期动态地创建对象并设定其属性（依赖关系）。

Spring 的主要目的就是使 J2EE 易用和促进良好的编程习惯。

Spring 是让已有的技术更加易用，比如，它没有底层事务协调处理，但是提供了一个抽象层覆盖了 JTA 和任何其他事务策略。Spring 并没有和其他的开源项目竞争，不过还是在一些领域有新的方案提供，比如它的 Web Framework、轻量级的 IoC 容器和 AOP 框架。

答案：Spring 的特点：

① Spring 不同于其他的 Framework，它要提供的是一种管理你的业务对象的方法。

② Spring 有分层的体系结构，意味着你能选择仅仅使用它的任何一个独立的部分，而其他的仍然使用你的相关实现。

③ 它的设计从一开始就是要帮助你编写易于测试的代码，Spring 是使用测试驱动开发（TDD）工程的理想框架。

④ Spring 不会给你的工程添加对其他的框架依赖；同时 Spring 又可以称得上是一个一揽子解决方案，提供了一个典型应用所需要的大部分基础架构。

之所以要使用 Spring，是因为

① Spring 能有效地组织你的中间层对象。

② Spring 能消除在许多工程中常见的对 Singleton 的过多使用。

③ 通过一种在不同应用程序和项目间一致的方法来处理配置文件，消除各种自定义格式的属性文件的需要，仅仅需要看看类的 JavaBean 属性。Inversion of Control 的使用帮助完成了这种简化。

④ 能够很容易培养你面向接口而不是面向类的编程习惯。

⑤ Spring 的设计会让使用它创建的应用尽可能少地依赖于它的 APIs，在 Spring 应用中的大多数业务对象没有依赖于 Spring。

⑥ 使用 Spring 构建的应用程序易于单元测试。

⑦ Spring 使 EJB 成为一个实现选择，而不是必需的选择。你可以用 POJOs 或 local EJBs 来实现业务接口，却不会影响到调用代码。

⑧ Spring 提供一些 Web 应用上的 EJB 的替代方案，比如用 AOP 提供声明性事务管理。

⑨ Spring 为数据存取提供了一个一致的框架，不论是使用 JDBC 还是 O/R mapping 的产品。

17.2 Hibernate

面试例题 1：Hibernate 的工作原理是什么？为什么要使用 Hibernate?

解析：Hibernate 可以理解为是一个中间件。它负责把 Java 程序的 SQL 语句接收过来并发送到数据库，而数据库返回来的信息由 Hibernate 接收后直接生成一个对象传给 Java。

首先必须有数据库文件，假设数据库里有一个 employee 员工表，SQL 语句如下：

```
create table employee(id Number(10),name varchar2(20))
```

其次，Hibernate 有两个特有的文件，一个是以.cfg.xml 结尾的文件，一个是以.hbm.xml 结尾的文件。下面分别详述：

1．关于 cfg.xml 文件

cfg.xml 文件的作用就是连接数据库。文件内部其实就是一个由 user、password、url、driver 组成的连接库的基本信息。文件的内容如下：

```
<Hibernate-configuration>
 <session-factory>
  <property name="connection.username">sss</property>
  <property name="connection.url">jdbc:oracle:thin:@192.168.0.1:1809:ris</property>
  <property name="dialect">org.Hibernate.dialect.Oracle9Dialect</property>
  <property name="connection.password">sss</property>
  <property name="connection.driver_class">oracle.jdbc.driver. OracleDriver</property>
```

```
    <mapping resource="employee.hbm.xml"/>
        </session-factory>
</Hibernate-configuration>
```

简单分析一下这个文件：

① <Hibernate-configuration>包含的是程序里面的 configuration 实例信息。通过这个实例的方法 configure，我们可以从 mapping 中得到对应的表的信息和类的信息。

② <session-factory>这个标签是程序中通过 configure 的方法 BuildSessionFactory 所得到的一个 SessionFactory 对象，这个对象可以理解为一个 statement。我们对数据库的所有操作都是通过它的一系列方法来实现的。

③ property dialect 都是 Hibernate 的一些属性设置，我们可以设置很多 Property，其中一些是必需的，一些是可选的。

2．关于 hbm.xml 文件

hbm.xml 文件是对数据库中表的映射文件，可以由这个文件指出哪个类对应着哪个表，而且还指出哪个类中的属性对应着表中的哪个字段。文件内容如下：

```
<Hibernate-mapping>
 <class name="src.employee" table="employee">
  <id name="id" column="id">
   <generator class="increment"/>
  </id>
  <property name="name" column="name"/>
 </class>
</Hibernate-mapping>
```

上面文件里面有一行语句

```
<generator class="increment"/>
```

可以实现 id 自动增加。也就是说，如果往数据库中插入一个 name，那么 id 就自动加 1。

这个文件解释了<Hibernate-mapping>所包含的就是 cfg.xml 文件中说的 mapping。Java 类中配置得到的 mapping 就是从这个文件里面读取出来的。如：

```
<class name="src.employee" table="employee">
```

就是先指定了类对应的表。文件中的语句就是指定表中的字段与类中的属性的对应关系。

下面再说说要实现功能所需的 Java 类，有两个：一个是 employee 类，一个是 Test 类。

employee 类是一个简单的 javaBean。它是先和数据库字段对应，然后取值的：

```
package src;
public class employee{
```

```
 private int id;
 private String name;
 public void setId(int id){
  this.id=id;
 }
 public void setName(String name){
  this.name=name;
 }
 public int getId(){
  return id;
 }
 public String getName(){
  return name;
 }
}
```

再看一下 Test 类：

```
package src;
import org.hibernate.*;
import org.hibernate.cfg.*;
public class Test{
 public static void main(String bb[]){
  try{
   SessionFactory sessfac=new Configuration().configure().BuildSessionFactory();
   Session session1=sessfac.opensession();
   Transaction transaction1=session1.beginTransaction();
   for(int i=0;i<3;i++){
    employee employee1=new employee();
    employee1.setName("begin go "+ i);
    session1.save(employee1);
    }
   transaction1.commit();
   session1.close();
  }catch(Exception e){
   e.printStackTrace();
  }
 }
}
```

对代码分析如下：

① 语句

```
SessionFactory sessfac=new Configuration().configure().BuildSessionFactory();
```

得到 configuration 的实例。然后通过 configure()读取 mapping 对应的 hbm.xml 文件的信息。

② 通过 BuildSessionFactory 得到 SessionFactory 对象。

③ 通过 opensession()建立连接。Session 就是指一个 Session 被建立，这里等于是一个 connection 被建立好。

④ 通过 Session 对象开启事务（Transaction）相当于 conn.setAutoCommit (false); 先不递交，最后通过另外一个方法递交；

⑤ 循环语句把 employee 实例化。

⑥ 实例化后就可以用其中的方法，如 setName 方法等。

⑦ 每次都要用 session 来 save 一下，一个对象 set 之后要保存在 session 中。

⑧ 最后递交 commit()。

之所以没有 SQL 语句，是因为 Hibernate 的特性，对数据库的操作就是对对象的操作，这就是 OR-Mapping 的本质。

答案：Hibernate 工作原理如下：

① 读取并解析配置文件。

② 读取并解析映射信息，创建 SessionFactory。

③ 打开 Session。

④ 创建事务 Transation。

⑤ 持久化操作。

⑥ 提交事务。

⑦ 关闭 Session。

⑧ 关闭 SesstionFactory。

使用 Hibernate 的原因如下：

① 对 JDBC 访问数据库的代码做了封装，大大简化了数据访问层烦琐的重复性代码。

② Hibernate 是一个基于 JDBC 的主流持久化框架，是一个优秀的 ORM 实现，它在很大程度上简化了 DAO 层的编码工作。

③ Hibernate 使用 Java 反射机制，而不是字节码增强程序来实现透明性。

④ Hibernate 的性能非常好，因为它是一个轻量级框架，映射的灵活性很出色。它支持各种关系数据库，从一对一到多对多的各种复杂关系。

面试例题 2：在 Hibernate 中，使用二级缓存机制要注意什么？

解析：Cache 是在提升系统性能方面常用的方法。Hibernate 中对于 Cache 有一级缓存和二级缓存的概念，一级缓存是必需的，位于 Session 部分，二级缓存则不是必需的，是由 sessionFactory 控制的进程级缓存，由开发人员自行指定。二级缓存可指定使用何种开源的 Cache 工具，Hibernate 3 以后的版本默认使用的是 Ehcache，也可以切换为 Oscache、JbossCache。查询时使用缓存的实现过程为：

① 查询一级缓存中是否具有需要的数据。

② 如果没有，查询二级缓存。

③ 如果二级缓存中也没有，此时再执行查询数据库的工作。

此 3 种方式的查询速度依次降低。

Hibernate 会自行维护缓存中的数据，以保证缓存中的数据和数据库中的真实数据的一致性。无论何时，当你调用方法传递或获得一个对象时，该对象都将被加入到 Session 的内部缓存中。当 flush()方法随后被调用时，对象的状态会和数据库取得同步。也就是说，删除、更新、增加数据的时候，同时更新缓存。

答案：Hibernate 中使用二级缓存时要注意的几点：

二级缓存能够明显提高系统的性能，当然，如果数据量特别巨大，此时不适合于二级缓存，原因是缓存的数据量过大可能会引起内存资源紧张，反而降低性能。

对于数据更新频率过高的数据，频繁地同步缓存中数据的代价可能和查询缓存中的数据从中获得的好处相当，坏处和益处相抵消，此时缓存的意义也不大。

财务数据等是非常重要的数据，绝对不允许出现或使用无效的数据，所以此时为了安全起见，最好不要使用二级缓存。因为此时“正确性”的重要性远远大于“高性能”的重要性。

因为数据表中的数据量虽然大，但是经常使用的往往只是较新的那部分数据，此时，也可为其配置二级缓存。但是必须单独配置其持久化类的缓存策略，比如最大缓存数、缓存过期时间等，将这些参数降低至一个合理的范围（太高会引起内存资源紧张，太低了，则缓存的意义不大），同时也会消耗更多的内存，可以通过配置文件来指定内存中能够加载的最多元素，这有利于避免消耗过多的内存。

面试例题 3：Hibernate 有哪些主键?

解析：Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，使得 Java 程序员可以随心所欲地使用对象编程思维来操纵数据库。Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的客户端程序使用，也可以在 Servlet/JSP 的 Web 应用中使用。最具革命意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP，完成数据持久化的重任。

Hibernate 的核心接口一共有 5 个，分别为：Session、SessionFactory、Transaction、Query 和 Configuration。这 5 个核心接口在任何开发中都会用到。通过这些接口不仅可以对持久化对象进行存取，还能够进行事务控制。各接口之间的交互关系如图 17-1 所示。

下面对这 5 个核心接口分别加以介绍。

Session 接口：Session 接口负责执行被持久化对象的 CRUD 操作（CRUD 的任务是完成与数据库的交流，包含了很多常见的 SQL 语句）。需要注意的是，Session 对象

是非线程安全的，同时，Hibernate 的 Session 不同于 JSP 应用中的 HttpSession。这里当使用 Session 这个术语时，其实指的是 Hibernate 中的 Session，而以后会将 HttpSesion 对象称为用户 Session。

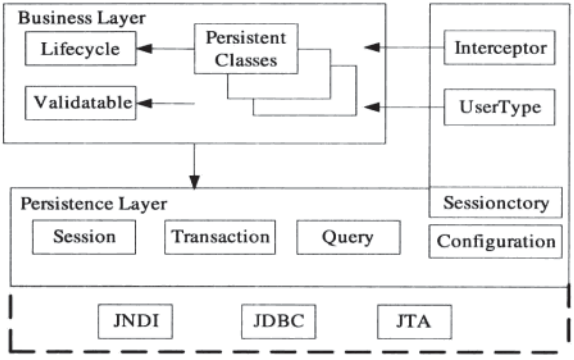


图 17-1　各接口之间的交互关系

SessionFactory 接口：SessionFactroy 接口负责初始化 Hibernate。它充当数据存储源的代理，并负责创建 Session 对象。这里用到了工厂模式。需要注意的是，SessionFactory 并不是轻量级的，因为一般情况下，一个项目通常只需要一个 SessionFactory 就够，当需要操作多个数据库时，可以为每个数据库指定一个 SessionFactory。

Configuration 接口：Configuration 接口负责配置并启动 Hibernate，创建 SessionFactory 对象。在 Hibernate 启动的过程中，Configuration 类的实例首先定位映射文档位置、读取配置，然后创建 SessionFactory 对象。

Transaction 接口：Transaction 接口负责事务相关的操作。它是可选的，开发人员也可以设计编写自己的底层事务处理代码。

Query 和 Criteria 接口：Query 和 Criteria 接口负责执行各种数据库查询。它可以使用 HQL 语言或 SQL 语句两种表达方式。

Hibernate 主键介绍：

Assigned：Assigned 方式由程序生成主键值，并且要在 save()之前指定，否则会抛出异常。特点是主键的生成值完全由用户决定，与底层数据库无关。用户需要维护主键值，在调用 session.save()之前要指定主键值。

Hilo：Hilo 使用高低位算法生成主键，高低位算法使用一个高位值和一个低位值，然后把算法得到的两个值拼接起来作为数据库中的唯一主键。Hilo 方式需要额外的数据库表和字段提供高位值来源。默认情况下，使用的表是 hibernate_unique_key，默认字段叫做 next_hi。next_hi 必须有一条记录，否则会出现错误。特点是需要额外的数据

库表的支持，能保证同一个数据库中主键的唯一性，但不能保证多个数据库之间主键的唯一性。Hilo 主键生成方式由 Hibernate 维护，所以，Hilo 方式与底层数据库无关，但不应该手动修改 hilo 算法使用的表值，否则会引起主键重复的异常。

Increment：Increment 方式对主键值采取自动增长的方式生成新的主键值，但要求底层数据库支持 Sequence 方式，如 Oracle、DB2 等。需要在映射文件 xxx.hbm.xml 中加入 Increment 标志符的设置。特点是由 Hibernate 本身维护，适用于所有的数据库，不适合多进程并发更新数据库，适合单一进程访问数据库，不能用于群集环境。

Identity：Identity 当时根据底层数据库来支持自动增长，不同的数据库用不同的主键增长方式。特点是与底层数据库有关，要求数据库支持 Identity，如 MySQL 中是 auto_increment, SQL Server 中是 Identity，支持的数据库有 MySQL、SQL Server、DB2、Sybase 和 HypersonicSQL。Identity 无须 Hibernate 和用户的干涉，使用较为方便，但不便于在不同的数据库之间移植程序。

Sequence：Sequence 需要底层数据库支持 Sequence 方式，例如 Oracle 数据库等。特点是需要底层数据库支持序列，支持序列的数据库有 DB2、PostgreSql、Qracle、SAPDb 等在不同数据库之间移植程序，特别是从支持序列的数据库移植到不支持序列的数据库需要修改配置文件。

Native：Native 主键生成方式会根据不同的底层数据库自动选择 Identity、Sequence、Hilo 主键生成方式。特点是根据不同的底层数据库采用不同的主键生成方式。由于 Hibernate 会根据底层数据库采用不同的映射方式，因此，便于程序移植，项目中如果用到多个数据库时，可以使用这种方式。

UUID：UUID 使用 128 位 UUID 算法生成主键，能够保证网络环境下主键的唯一性，也就能够保证在不同数据库及不同服务器下主键的唯一性。特点是能够保证数据库中主键的唯一性，生成的主键占用比较多的存储空间。

Foreign GUID：Foreign 用于一对一关系中。GUID 主键生成方式使用了一种特殊算法，保证生成主键的唯一性，支持 SQL Server 和 MySQL。

答案：Hibernate 有如下主键：Hilo，Increment，Identity，Sequence，Native，UUID，Foreign GUID。

面试例题 4：Hibernate 有几种查询数据的方式？

答案：有三种查询方式。

① HQL：Hibernate Query Language，它跟 SQL 非常类似。

② Criteria Query，以对象的方式添加查询条件。

③ SQL，直接使用 SQL 语句操作数据库。

面试例题 5：Hibernate 在性能提升上其实有很多种做法，请描述你在项目中提升性能通常采用的方法。

答案：

（1）Lazy Load

Lazy Load 是常用的一种提升性能的方法。这个其实很容易理解。在不采用 Lazy Load 的情况下，Hibernate 在获取一个 PO 的时候，将同时获取 PO 中的属性，PO 中的集合及集合中对象的属性、集合。这样很容易看出，如果对象的关联结构有深层次的话，最后整个库可能都被加载出来了，而在实际使用中往往可能只需要用到 PO 中的一两个属性而已。在 Hibernate 3 以后，默认的 lazy 就已经设置为 true 了，这个时候包括 PO 中的属性都是采用 Lazy Load 的方式，只有在调用到这个属性时才会从缓存或数据库中加载。当然，集合也同样如此。

在 Lazy Load 上推荐不要什么字段都采用 Lazy Load 的方式。对于一些基本属性的字段，建议将其 lazy 设置为 false，而对于一些可能需要消耗内存的字段，如 clob 这样的字段，建议将其 lazy 设置为 true，对于集合，则全部设置为 lazy=true。

是否采用 Lazy load 对系统的性能会有非常明显的影响，同时尽量不要将 Detached Object 放入 HTTP 的 session 中。

（2）Cache

Cache 是在提升系统性能方面常用的方法。在 Hibernate 中通常有非常好的对于 Cache 的支持方法。Hibernate 中对于 Cache 有一级缓存和二级缓存的概念。一级缓存是必需的，位于 Session 部分；二级缓存则不是必需的，由开发人员自行指定。二级缓存可指定使用何种开源的 Cache 工具。Hibernate 3 以后的版本默认使用的是 EhCache，也可以切换为 OsCache、JBossCache，它们最重要的区别在于对 cluster 的支持上。

二级缓存能够明显地提高系统的性能，当然，同时也会更加消耗内存。可以通过配置文件来指定内存中能够加载的最多元素，这有利于避免消耗过多的内存。

二级缓存的设置在 Hibernate 中非常简单，只需要在相应的 hbm.xml 中增加 Cache 元素，指明使用何种策略，如 read-only、read-write 等，也可以通过直接在 hibernate.cfg.xml 中增加 class-cache 的方式来进行全局指定。

（3）高效的查询语句

查询语句是否高效，对于系统的性能也会造成明显的影响。为了方便系统性能的调优，建议大家对查询语句进行统一管理，如统一采用 NamedQuery 的方式。在这样的情况下可以在系统运行时请教数据库专家，由他们来分析系统中查询语句的执行效率及提出改进策略。而对于开发人员来讲，在查询语句方面最能够注意的就是采用占位符式的查询。

占位符式的数据库查询对于所有的 SQL 语句都要进行语法分析，而其分析通常会受到语句中的大小写、空格及参数不同的影响，在其语法分析器认为不同的情况下将再次进行分析，这就不可避免地降低了响应的速度。而采用占位符式的语法查询则可保证语法分析器只进行一次分析，在参数不同的情况下并不会出现重复解析的现象。其次就是要统一查询语句的编写风格，包括大小写、空格等。

（4）配置

在 hibernate.cfg.xml 中的一些配置也会对性能产生一定的影响，如 jdbc.fetch_size 的设置等，还有像采用连接池方面的设置。对于 B/S 应用的情况，建议尽量采用应用服务器提供的 JNDI 的方式。

17.3 EJB

面试例题 1：EJB 2.0 有哪些内容？分别用在什么场合？

答案：规范内容包括 Bean 提供者、应用程序装配者、EJB 容器、EJB 配置工具、EJB 服务提供者和系统管理员。其中，EJB 容器是 EJB 之所以能够运行的核心。EJB 容器管理着 EJB 的创建、撤销、激活、去活及与数据库的连接等重要的核心工作。

面试例题 2：EJB 与 Java Bean 的区别是什么？

答案：Java Bean 是可复用的组件，对 Java Bean 并没有严格的规范，从理论上讲，任何一个 Java 类都可以是一个 Bean。但通常情况下，由于 Java Bean 是被容器所创建的（如 Tomcat），所以 Java Bean 应具有一个无参的构造器，另外，Java Bean 通常还要实现 Serializable 接口用于实现 Bean 的持久性。Java Bean 实际上相当于微软 COM 模型中的本地进程内的 COM 组件，它是不能被跨进程访问的。Enterprise Java Bean 相当于 DCOM，即分布式组件。它是基于 Java 的远程方法调用（RMI）技术的，所以 EJB 可以被远程访问（跨进程、跨计算机）。但 EJB 必须被部署在诸如 WebSpere、WebLogic 这样的容器中，EJB 客户从不直接访问真正的 EJB 组件，而是通过其容器访问。EJB 容器是 EJB 组件的代理，EJB 组件由容器所创建和管理。客户通过容器来访问真正的 EJB 组件。

面试例题 3：EJB 是基于哪些技术实现的？并说出 Session Bean 和 Entity Bean 的区别，以及 Stateful Bean 和 Stateless Bean 的区别。

答案：EJB 包括 Session Bean、Entity Bean、MessageDriven Bean，基于 JNDI、RMI、JAT 等技术实现。Session Bean 在 J2EE 应用程序中被用来完成一些服务器端的业务操

作，例如，访问数据库、调用其他 EJB 组件。Entity Bean 被用来代表应用系统中用到的数据。对于客户机，Session Bean 是一种非持久性对象，它实现某些在服务器上运行的业务逻辑。而同样对于客户机，Entity Bean 是一种持久性对象，它代表一个存储在持久性存储器中的实体的对象视图，或是一个由现有企业应用程序实现的实体。Session Bean 还可以再细分为 Stateful Session Bean，这两种 Session Bean 都可以将系统逻辑放在 method 之中执行，不同的是，Stateful Session Bean 可以记录呼叫者的状态，因此，通常来说，一个使用者会有一个相对应的 Stateful Session Bean 的实 Stateless 体。Session Bean 虽然也是逻辑组件，但是它不负责记录使用者状态，也就是说，当使用者呼叫 Stateless Session Bean 的时候，EJB Container 并不会找寻特定的 Stateless Session Bean 的实体来执行这个 method。换言之，很可能数个使用者在执行某个 Stateless Session Bean 的 methods 时，会是同一个 Bean 的 Instance 在执行。从内存方面来看，Stateful Session Bean 与 Stateless Session Bean 比较，Stateful Session Bean 会消耗 J2EE Server 较多的内存，然而 Stateful Session Bean 的优势却在于它可以维持使用者的状态。

面试例题 4：EJB 包括 Session Bean 和 Entity Bean，说出它们的生命周期，以及 EJB 是如何管理事务的。

答案：Session Bean：Stateless Session Bean 的生命周期是由容器决定的。当客户机发出请求要建立一个 Bean 的实例时，EJB 容器不一定要创建一个新的 Bean 实例供客户机调用，而是随便找一个现有的实例提供给客户机。当客户机第一次调用一个 Stateful Session Bean 时，容器必须立即在服务器中创建一个新的 Bean 实例，并关联到客户机上，以后此客户机调用 Stateful Session Bean 的方法时，容器会把调用分派到与此客户机相关联的 Bean 实例。

Entity Bean：Entity Beans 能存活相对较长的时间，并且状态是持续的。只要数据库中的数据存在，Entity Beans 就一直存活，而不是按照应用程序或者服务进程来说的。即使 EJB 容器崩溃了，Entity Beans 也是存活的。Entity Beans 生命周期能够被容器或者 Beans 自己管理。

EJB 通过以下技术管理事务：对象管理组织（OMG）的对象事务服务（OTS），Sun Microsystems 的 Transaction Service（JTS）、Java TransactionAPI（JTA），开发组（X/Open）的 XA 接口。

面试例题 5：EJB 的角色和 3 个对象是什么？

答案：一个完整的基于 EJB 的分布式计算结构由 6 个角色组成，这 6 个角色可以由不同的开发商提供，每个角色所做的工作必须遵循 Sun 公司提供的 EJB 规范，以保

证彼此之间的兼容性。这 6 个角色分别是 EJB 组件开发者（Enterprise Bean Provider）、应用组合者（Application Assembler）、部署者（Deployer）、EJB 服务器提供者（EJB Server Provider）、EJB 容器提供者（EJB Container Provider）和系统管理员（System Administrator）。3 个对象是 Remote（Local）接口、Home（LocalHome）接口和 Bean 类。

面试例题 6：EJB 容器提供哪些服务？

答案：主要提供声明周期管理、代码产生、持续性管理、安全、事务管理、锁和并发行管理等服务。

面试例题 7：EJB 规范规定 EJB 中禁止的操作有哪些？

答案：

（1）不能操作线程和线程 API（线程 API 指非线程对象的方法，如 notify、wait 等）。

（2）不能操作 AWT。

（3）不能实现服务器功能。

（4）不能对静态属性存取。

（5）不能使用 I/O 操作直接存取文件系统。

（6）不能加载本地库。

（7）不能将 this 作为变量和返回。

（8）不能循环调用。

面试例题 8：说说你所熟悉或听说过的 J2EE 中的几种常用模式，以及对设计模式的一些看法。

答案：

Session Facade Pattern：使用 Session Bean 访问 Entity Bean。

Message Facade Pattern：实现异步调用。

EJB Command Pattern：使用 Command Java Beans 取代 Session Bean，实现轻量级访问。

Business Interface：通过远程（本地）接口和 Bean 类实现相同接口规范业务逻辑一致性。

EJB 架构的设计好坏将直接影响系统的性能、可扩展性、可维护性、组件可重用性及开发效率。项目越复杂，项目队伍越庞大，则越能体现良好设计的重要性。

面试例题 9：Remote 接口和 Home 接口的主要作用分别是什么？

答案：Remote 接口定义了业务方法，用于 EJB 客户端调用业务方法。Home 接口是 EJB 工厂用于创建、移除、查找 EJB 实例所使用的方法。

面试例题 10：简介 Bean 实例的生命周期。

答案：对于 Stateless Session Bean、Entity Bean 和 Message Driven Bean，一般存在缓冲池管理，而对于 Entity Bean 和 Stateful Session Bean，存在 Cache 管理，通常包含创建实例、设置上下文、创建 EJB Object（create）、业务方法调用和 remove 等过程。对于存在缓冲池管理的 Bean，在 create 之后，实例并不从内存清除，而是采用缓冲池调度机制不断地重用实例。而对于存在 Cache 管理的 Bean，则通过激活和去激活机制保持 Bean 的状态并限制内存中实例的数量。

面试例题 11：简介 EJB 的激活机制。

答案：以 Stateful Session Bean 为例，其 Cache 大小决定了内存中可以同时存在的 Bean 实例的数量。根据 MRU 或 NRU 算法，实例在激活和去激活状态之间迁移。激活机制是当客户端调用某个 EJB 实例业务方法时，如果对应 EJB Object 发现自己没有绑定对应的 Bean 实例，则从其去激活 Bean 存储中（通过序列化机制存储实例）回复（激活）此实例。状态变迁前会调用对应的 ejbActive 和 ejbPassivate 方法。

面试例题 12：EJB 包括几种类型？

答案：EJB 包括会话（Session）Bean，实体（Entity）Bean 和消息驱动的（Message Driven）Bean。会话 Bean 又可分为有状态（Stateful）和无状态（Stateless）两种。实体 Bean 可分为 Bean 管理的持续性（BMP）和容器管理的持续性（CMP）两种。

面试例题 13：简介客服端调用 EJB 对象的几个基本步骤。

答案：设置 JNDI 服务工厂及 JNDI 服务地址系统属性；查找 Home 接口；从 Home 接口调用 Create 方法创建 Remote 接口；通过 Remote 接口调用其业务方法。

面试例题 14：什么是可伸缩性？什么是重/轻量架构，谁是实用系统的架构主选？

答案：所谓可伸缩性，是指在小型规模单台服务器情况下，应用系统可以良好地运转，系统的访问量或功能增加后，整个系统只需要通过增加服务器硬件就可以实现性能扩展，无须修改太多的软件。对于可伸缩性平台（如 JBoss）来说，理论上，没有最大负载或最多在线人数这样的概念。

重/轻量其实是使用难易程度，从根本上说，重/轻量应该和可伸缩性不矛盾，特别是在 EJB 3.0 推出以后，这个问题应该得到比较好的解决。

但是，在目前情况下，编写一个 Java Beans 要比编写一个 EJB 容易得多，那么，是重/轻量还是可伸缩性应该成为系统架构的主要依据呢？

重量一般是基于 JBoss/EJB 的完整 J2EE 系统架构（具有可伸缩性，目前不易于学习）。轻量一般是基于 Tomcat 的 Struts+Hibernate/Spring+ Hibernate（目前无太大可伸缩性，但是易于学习、使用）。

因为轻量解决方案易于学习新技术，容易使用，选中率比较高。但是让人产生对系统的可伸缩性的担忧。鉴于这种情况，我认为有必要强调一下可伸缩性的重要性，且不能因为要跟进新的设计思想和技术，而盲目地采用一个无可伸缩性的设计方案。

其实，“轻量”应该是一个中性词，但是因为大量新的设计思想比较容易通过轻量方案获得成型软件，如 Spring、Naning、Hibernate 等，逐渐地，“轻量”好像变成了一个褒义词。如果从可伸缩性的标准看，轻量还可能是一个贬义词，轻量意味着丧失重量系统中的分布式网络计算的设计考量，那么可伸缩性就要打问号。

从长远来看，随着 EJB 3.0 中间件轻量化、SOA 架构理念的普及，轻量/重量的区别已经模糊，如果还是以轻量/重量作为架构选择的标准，甚至标榜自己的系统，无疑是不明智的。

可伸缩性应该依然是实用企业系统架构的主选。可伸缩性是站在软件公司的客户企业立场为这些客户企业考虑的，但是他们经常因为被认为是外行，被挡在了软件系统架构选择的门外。

17.4 JDBC

面试例题 1：对于 JDBC 连接，下面哪个表述是正确的？

A．连接是由 JDBC Driver 管理的　　B．连接的建立及关闭比较消耗资源

C．连接不需要在代码中显式地关闭　　D．连接是基于 ACM 创建的

解析：JDBC 是一种可用于执行 SQL 语句的 Java API（应用程序设计接口）。它由一些 Java 语言编写的类和界面组成。JDBC 为数据库应用开发人员、数据库前台工具开发人员提供了一种标准的应用程序设计接口，使开发人员可以用纯 Java 语言编写完整的数据库应用程序。

Java 和 JDBC 的结合可以让开发人员在开发数据库应用时真正实现很好的移植性。JDBC 扩展了 Java 的能力，如使用 Java 和 JDBCAPI 就可以公布一个 Web 页，页中带有能访问远端数据库的 Applet；或者企业可以通过 JDBC 让所有的职工（他们可以使

用不同的操作系统，如 Windows、Machintosh 和 UNIX）在 Intranet 上连接到几个全球数据库上，而这几个全球数据库可以是不相同的。程序员可以编写或改写一个程序，然后将它放在服务器上，而每个用户都可以访问服务器得到最新的版本。对于信息服务行业，Java 和 JDBC 提供了一种很好的向外界用户更新信息的方法。

简单地说，JDBC 能完成下列三件事：

- 同一个数据库建立连接。
- 向数据库发送 SQL 语句。
- 处理数据库返回的结果。

JDBC 直接调用 SQL 命令，同时它也是构造高层 API 和数据库开发工具的基础。高层 API 和数据库开发工具应该是用户界面更加友好，使用更加方便，更易于理解的。但所有这样的 API 将最终被翻译为像 JDBC 这样的底层 API。SQL 语言嵌入 Java 的预处理器，虽然 DBMS 已经实现了 SQL 查询，但 JDBC 要求 SQL 语句被当做字符串参数传送给 Java 程序。而嵌入式 SQL 预处理器答应程序员将 SQL 语句混用：Java 变量可以在 SQL 语句中使用来接收或提供数值。然后 SQL 的预处理器将把这种 Java / SQL 混用的程序翻译成带有 JDBCAPI 的 Java 程序。

本题中，答案 A 描述不准确。连接管理的操作是通过驱动管理器（Driver Manager）实现的。

选项 B 是正确的，JDBC 连接的建立及关闭是极其耗资源的。为了确保 JDBC 资源不在出现异常或错误等情况下被不正常关闭，应该在使用完 JDBC 资源之后关闭且释放它们。JDBC 连接池提供了 JDBC 连接定义和数目有限的连接，假如数量不够，就需要长时间的等待。不正常关闭 JDBC 连接会导致等待回收无效的 JDBC 连接。只有正常关闭和释放 JDBC 连接，JDBC 资源才可以被快速重用使性能得到改善。失败的关闭和释放 JDBC 连接可能导致其他用户的连接经历长时间的等待。虽然超时的 JDBC 连接会被 WebSphereApplicationServer 退回而被回收，但必须等待这种情形发生。使用完 JDBC 资源后关闭它们，还可以显式地关闭 JDBCResultSets。假如没有显式的关闭语句，则在完成了相关语句之后会释放 ResultsSets。所以要确保构建的代码在所有的情况下，甚至在异常和错误条件下，都能关闭和释放 JDBC 资源。

D 选项也不对，JDBC 连接基于 Java API。

解析：B

面试例题 2：在 JDBC 连接数据库编程应用开发中，利用哪个类可以实现连接数据库（　　）？

A．Connection 接口　　B．PreparedStatement 类

C．CallableStatement 类　　D．Statement 类

解析：本题涉及以下知识点：JDBC 数据库连接问题。PreparedStatement 类、CallableStatement 类、Statement 类这三类区别的问题。三个选项中只有 Connection 接口可以用来连接数据库。

答案：A

扩展知识1：JDBC连接

JDBC 2.0 API 被划分为两部分：JDBC 2.0 核心 API 和 JDBC 2.0 标准扩展 API。核心 API 在 java.sql 里面，这时原来的版本就实现了基本的功能。标准扩展 API 在 javax.sql 里面，由 JDBC 2.0 规范新规定的一些接口在这里面。当然，JDBC 2.0 也对原来版本的 java.sql 核心做了一些改动，不过不是很大。原来 JDBC 1.0 的程序可以不加修改地在 JDBC 2.0 上运行。

JDBC 2.0 的扩展 API 增加了一些数据访问和数据源访问的重大功能。这中间有一些主要用来做企业计算。用 JDBC 2.0 的新的扩展包，JDBC 提供了一个从 Java 2 平台的通用数据访问的方法。

首先，我们来看看 JDBC 标准扩展的 API 是怎样和 JDBC 2.0 结合在一起的。JDBC 2.0 包括以下两个包。

（1）java.sql 包：这个包里面是 JDBC 2.0 的核心 API。它包括原来的 JDBC API（JDBC 1.0 版本），再加上一些新的 2.0 版本的 API。这个包在 Java 2 Platform SDK 里面提供。

（2）javax.sql 包：这里面是 JDBC 2.0 的标准扩展 API。这个包是一个全新的，在 Java 2 Platform SDK、Enterprise Edition 里面单独提供。

JDBC 2.0 的核心 API 包括 JDBC 1.0 的 API，并在此基础上增加了一些功能，对某些性能做了增强，使 Java 语言在数据库计算的前端提供了统一的数据访问方法，效率也得到了提高。

JDBC 是向后兼容的，JDBC 1.0 的程序可以不加修改地运行在 JDBC 2.0 上。但是，假如程序中用到了 JDBC 2.0 的新特性，就必须要运行在 JDBC 2.0 版本上。

概括地说，JDBC 核心 API 的新特性在两个方面做了工作，一个是支持一些新的功能；另一个就是支持 SQL 3 的数据类型。

（1）在支持新功能方面：包括结果集可以向后滚动，批量地更新数据。另外，还提供了 UNICODE 字符集的字符流操作。

（2）在支持 SQL 3 的数据类型方面：包括新的 SQL 3 数据类型，增加了对持久性对象的存储。

JDBC 的新特性在数据存取、交互操作等方面较以往更容易了。例如，数据块的操作能够显著地提高数据库访问的性能。新增加的 BLOB、CLOB 和数组接口能够使应用程序操作大块的数据类型，而不必使客户端在存储之前进行其他的处理。这样，就显著地提高了内存的使用效率。

标准扩展 API 分为如下几个方面。

- DataSource 接口：和 Java 名字目录服务（JNDI）一起工作的数据源接口。
- Connection pooling（连接池）：可以重复使用连接，而不是对每个请求都使用一个新的连接。
- Distribute transaction（分布式的事务）：在一个事务中涉及多个数据库服务器。
- Rowsets：Java Bean 组件包含结果集，主要用来将数据传给瘦客户，或者提供一个可以滚动的结果集。

下面具体介绍。

1．DataSource 接口是一个更好的连接数据源的方法

JDBC 1.0 原来是用 DriverManager 类来产生一个对数据源的连接。

一个 DataSource 对象代表了一个真正的数据源。根据 DataSource 的实现方法，数据源既可以是关系数据库，也可以是电子表格，还可以是一个表格形式的文件。当一个 DataSource 对象注册到名字服务中，应用程序就可以通过名字服务获得 DataSource 对象，并用它来产生一个与 DataSource 代表的数据源之间的连接。

关于数据源的信息和如何来定位数据源，例如，数据库服务器的名字、在哪台机器上、端口号等，都包含在 DataSource 对象的属性中。这样，对应用程序的设计就更方便，因为并不需要硬性地把驱动的名字固定到程序中。通常，驱动名字中都包含了驱动提供商的名字，而在 DriverManager 类中通常是这么做的。如果数据源要移植到另一个数据库驱动中，代码也很容易修改。所需要做的修改只是更改 DataSource 的相关属性，而使用 DataSource 对象的代码不需要做任何改动。

由系统管理员或者有相应权限的人来配置 DataSource 对象。配置 DataSource 时，包括设定 DataSource 的属性，然后将它注册到 JNDI 名字服务中去。在注册 DataSource 对象的过程中，系统管理员需要把 DataSource 对象和一个逻辑名字关联起来。名字可以是任意的，通常取能代表数据源且容易记住的名字。在下面的例子中，名字起为：InventoryDB，按照惯例，逻辑名字通常都在 jdbc 的子上下文中。这样，逻辑名字的全名就是：jdbc/InventoryDB。

一旦配置好了数据源对象，应用程序设计者就可以用它来产生一个与数据源的连接。下面的代码片段示例了如何用 JNDI 上下文获得一个数据源对象，然后如何用数据源对象产生一个与数据源的连接。开始的两行用的是 JNDI API，第三行用的才是 JDBC 的 API。

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/InventoryDB");
Connection con =ds.getConnection("myPassword", "myUserName");
```

在一个基本的 DataSource 实现中，DataSource.getConnection 方法返回的 Connection 对象和用 DriverManager.getConnection 方法返回的 Connection 对象是一样的。因为 DataSource 提供的方便性，我们推荐使用 DataSource 对象来得到一个 Connection 对象。我们希望所有的基于 JDBC 2.0 技术的数据库驱动都包含一个基本的 DataSource 的实现，这样就很容易在应用程序中使用它。

对于普通的应用程序设计者，是否使用 DataSource 对象只是一个选择问题。但是，对于那些需要用连接池或者分布式事务的应用程序设计者来说，就必须使用 DataSource 对象来获得 Connection（其中的原因在下面我们会提到）。

2．Connection pooling（连接池）

连接池是这么一种机制：当应用程序关闭一个 Connection 的时候，这个连接被回收，而不是被破坏，因为建立一个连接是很浪费资源的操作。如果能把回收的连接重新利用，会减少新创建连接的数目，从而显著地提高运行的性能。

假设应用程序需要建立到一个名字为 EmployeeDB 的 DataSource 的连接，使用连接池得到连接的代码如下：

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/EmployeeDB");
Connection con =ds.getConnection("myPassword", "myUserName");
```

除了逻辑名字以外，我们发现其代码和上面所举例子的代码是一样的。逻辑名字不同，就可以连接到不同的数据库。DataSource 对象的 get Connection 方法返回的 Connection 是否是一个连接池中的连接，完全取决于 DataSource 对象的实现方法。如果 DataSource 对象实现与一个支持连接池中间层的服务器一起工作，DataSource 对象就会自动返回连接池中的连接，这个连接也是可以重复利用的。

是否使用连接池获得一个连接，在应用程序的代码上是看不出不同的。在使用这个 Connection 连接上也没有什么不一样的地方，唯一不同的是在 Java 的

finally 语句块中来关闭一个连接。在 finally 中关闭连接是一个好的编程习惯。这样，即使方法抛出异常，Connection 也会被关闭并回收到连接池中。代码如下所示：

```
try{…
}catch () {…
}finally{ if (con!=null) con.close();}
```

3．分布式事务

获得一个用来支持分布式事务的连接与获得连接池中的连接是很相似的。同样，不同之处在于 DataSource 实现上的不同，而不是在应用程序中获得连接的方式上有什么不同。假设 DataSource 的实现可以与支持分布式事务的中间层服务器一起工作，得到连接的代码如下所示：

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/EmployeeDB");
Connection con = ds.getConnection("myPassword","myUserName");
```

由于性能上的原因，如果一个 DataSource 能够支持分布式的事务，它同样也可以支持连接池管理。

从应用程序设计者的观点来看，是否支持分布式的事务的连接对程序本身来说没什么不同，唯一的不同是在事务的边界上（开始一个事务的地方和结束一个事务的地方），开始一个事务或者结束一个事务都是由事务服务器来控制的。应用程序不应该做任何可能妨碍服务的事情。应用程序不能够直接调用事务提交（commit）或者回滚（rollback）操作，也不能够使用事务的自动提交模式（在数据库操作完成的时候自动调用 commit 或者 rollback）。

对于通常的 Connection 来说，默认的是 auto-commit 模式。而对于支持分布式事务的 Connection 来说，默认的不是 auto-commit 模式。注意，即使 Connection 是支持事务的，它也可以用于没有事务的情况。关于事务边界的限制，只在针对分布式事务的情况下才成立。

4．结果集

结果集对象是一行行数据的容器。根据其目的，可以通过多种方法实现。RowSet 及其相关的接口与 JDBC 2.0 的标准扩展 API 有点不同，它们并不是驱动的一部分，RowSet 是在驱动的上层实现的，可以由其他任何人来实现它们。

任何类型的 RowSet 都实现了 RowSet 接口，RowSet 接口扩展了 ResultSet 接口。这样 RowSet 对象就有了 ResultSet 对象所有的功能，能够通过 getXXX 方法得到数据库中的某列值，通过 updateXXX 方法可以修改某列值，可以移动

光标，使当前行变为另一行。

当然，我们更感兴趣的是 RowSet 接口提供的新功能。作为一个 Java Bean 组件，RowSet 对象可以增加或者删除一个 listener（监听者），可以 get 或者 set 其属性值。在这些属性中，有一个是字符串，表示一个对数据库的 Query 请求，RowSet 接口定义了设定参数的方法，也提供了执行这个请求的方法。这意味着 RowSet 对象能够执行查询请求，可以根据它产生的结果集进行计算。同样，RowSet 也可以根据任何表格数据源进行计算。所以，它不局限于关系数据库。

从数据源得到数据之后，RowSet 对象可以和数据源断开连接，RowSet 也可以被序列化。这样，RowSet 就可以通过网络传递给瘦客户端。

RowSet 可以被重新连接到数据源，这样，所做的修改就可以存回到数据源中去。如果产生了一个 listener，当 RowSet 的当前行移动或数据被修改的时候，监听者就会收到通知。例如，图形用户界面组件可以注册成为监听者，当 RowSet 更改的时候，图形用户界面接到通知，就可以修改界面，来符合它所表示的 RowSet。

根据不同的需要，RowSet 接口可以通过多种方法来实现。与 Cached RowSet 类不一样的是，JDBCRowSet 类总是保持和数据源的连接。相当于在 ResultSet 外围加了一层，使基于 JDBC 技术的驱动看起来像是一个简单的 Java Bean 组件一样。

扩展知识2：PreparedStatement类、CallableStatement类和Statement类

（1）Prepared（准备好的，精制的），从这里可以知道 PreparedStatement 是预先编译的语句，而 Statement 则不是预先编译的。PreparedStatement 继承于 Statement，通常的 JDBC 实现过程中，PreparedStatement 最终还是通过 Statement 的相关方法来执行 SQL 的（可以做少量优化），其最主要的优势在于可以减少 SQL 的编译错误（在 JDBC 中就可以捕获部分异常而不是由数据库服务器执行时返回错误代码），增加 SQL 安全性（减少 SQL 注入的机会）。

（2）PrepareStatement 中执行的 SQL 语句中是可以带参数的，而 Statement 则不可以。比如：

```
    PreparedStatement pstmt=con.prepareStatement("UPDATE EMPLOYEES SET SALARY = ?
WHERE ID = ?");
    pstmt.setBigDecimal(1, 1000.00);
    pstmt.setInt(2, 1111);
    pstmt.executeUpdate();
```

（3）对于数据库来说，单个 PrepareStatement 因为还需要参数代入过程，所以比单个 Statement 执行速度慢是很正常的。但是对于一个真实的应用来说，单个请求执行速度快几十毫秒还是慢几十毫秒，根本无关紧要。

当处理批量 SQL 语句时，这个时候就可以体现 PrepareStatement 的优势，由于采用 Cache 机制，则预先编译的语句就会放在 Cache 中，下次执行相同的 SQL 语句时，则可以直接从 Cache 中取出来。例如，网站有这么一个常用查询，每天可能要执行几十万次，如果用 PreparedStatement，理论上数据库只编译一次。只要有一个用户执行过此查询，其他用户执行的时候就会相对变快，尤其是比较复杂的 SQL 语句，可能会比较明显。如果用 Statement，由于有很多条件相同，理论上数据库要更多的编译，这样占的数据库内存也要多一些。

（4）CallableStatement 对象为所有的 DBMS 提供了一种以标准形式调用已储存过程的方法。已储存过程储存在数据库中，对已储存过程的调用是 CallableStatement 对象所含的内容。这种调用是用一种换码语法来写的，有两种形式：一种形式是带结果参数，另一种形式是不带结果参数。其中，结果参数是一种输出（OUT）参数，是已储存过程的返回值。两种形式都可带有数量可变的输入（IN 参数）、输出（OUT 参数）或输入和输出（INOUT 参数）的参数。问号将用做参数的占位符。

通常，创建 CallableStatement 对象的程序员应当知道所用的 DBMS 是支持已储存过程的，并且知道这些过程都是些什么。然而，假如需要检查，多种 DatabaseMetaData 方法都可以提供这样的信息。例如，假如 DBMS 支持已储存过程的调用，则 supportsStoredProcedures 方法将返回 true，而 getProcedures 方法将返回对已储存过程的描述。

CallableStatement 中定义的所有方法都用于处理 OUT 参数或 INOUT 参数的输出部分：注册 OUT 参数的 JDBC 类型（一般 SQL 类型）、从这些参数中检索结果，或者检查所返回的值是否为 NULL。

面试例题 3：JDBC 的 Connection 接口不包含下面哪个方法？

A．createStatement()　　B．prepareStatement(String sql)

C．createPrepareStatement(String sql)　　D．prepareCall(String sql)

解析：JDBC 的 Connection 接口包含如下方法：

```
① String nativeSQL(String string) throws SQLException;
② void setAutoCommit(boolean boolean0) throws SQLException;
③ boolean getAutoCommit() throws SQLException;
```

```
④ void commit() throws SQLException;
⑤ void rollback() throws SQLException;
⑥ void setCatalog(String string) throws SQLException;
⑦ String getCatalog() throws SQLException;
⑧ void setTransactionIsolation(int int0) throws SQLException;
⑨ int getTransactionIsolation() throws SQLException;
⑩ Statement createStatement(int int0, int int1) throws SQLException;
⑪ PreparedStatement prepareStatement(String string, int int1, int int2) throws
SQLException;
⑫ CallableStatement prepareCall(String string, int int1, int int2) throws SQLException;
⑬ Map getTypeMap() throws SQLException;
⑭ void setTypeMap(Map map) throws SQLException;
```

以上这些方法都是关于数据库连接访问的，如设置是否自动提交、预处理 STATEMENT、提交、回滚等相应的方法。

答案：C

17.5 JDO

面试例题 1：Class.forName 的作用是什么？

答案：调用该访问返回一个以字符串指定类名的类的对象。

面试例题 2：JDO 是什么？

答案：JDO 是 Java 对象持久化的新规范，是 Java Data Object 的缩写。JDO 也是一个用于存取某种数据仓库中的对象的标准化 API，它提供了透明的对象存储。因此，对开发人员来说，存储数据对象完全不需要额外的代码（如 JDBC API 的使用）。这些烦琐的例行工作已经转移到 JDO 产品提供商身上，使开发人员解脱出来，从而把时间和精力集中在业务逻辑上。另外，JDO 很灵活，因为它可以在任何数据底层上运行。JDBC 只是面向关系数据库（RDBMS），JDO 更通用，提供到任何数据底层的存储功能，比如关系数据库、文件、XML，以及对象数据库（ODBMS）等，使得应用的可移植性更强。

面试例题 3：数据连接池的工作机制是什么？

答案：J2EE 服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其标记为忙。如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量由配置参数决定。当使用的池连接调用完成后，池驱动程序将此连接标记为空闲，其他调用就可以使用这个连接。

第18章

Java 中的 Web 设计

关于 Web 设计的面试题目，涉及 Session、Servlet、JSP、Javascript 和 XML 等方面。以下的考题来自真实的笔试资料，希望读者先不要看答案，自我解答后再与答案加以对比，找出自己的不足。

18.1 JSP

面试例题 1：forward 和 redirect 的区别是什么？

答案：forward 是服务器请求资源，服务器直接访问目标地址的 URL，把 URL 的响应内容读取过来，然后把这些内容再发给浏览器，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。

redirect 就是服务端根据逻辑发送一个状态码，告诉浏览器重新去请求事先访问过的那个地址，一般来说，浏览器会用刚才请求的所有参数重新请求，所以 Session、request 参数都可以获取。

面试例题 2：JSP 有哪些内置对象？作用分别是什么？

答案：JSP 共有以下 9 种基本内置组件（可与 ASP 的 6 种内部组件相对应）。

request：用户端请求，此请求会包含来自 GET/POST 请求的参数。

response：网页传回用户端的回应。

pageContext：网页的属性是在这里管理的。

Session：与请求有关的会话期。

application：Servlet 正在执行的内容。

Out：用来传送回应的输出。

Config：Servlet 的构架部件。

Page：JSP 网页本身。

Exception：针对错误网页，未捕捉的例外。

面试例题 3：JSP 有哪些动作？它们的作用分别是什么？

答案：JSP 共有以下 6 种基本动作。

jsp:include：在页面被请求的时候引入一个文件。

jsp:useBean：寻找或者实例化一个 JavaBean。

jsp:setProperty：设置 JavaBean 的属性。

jsp:getProperty：输出某个 JavaBean 的属性。

jsp:forward：把请求转到一个新的页面。

jsp:plugin：根据浏览器类型为 Java 插件生成 OBJECT 或 EMBED 标记。

面试例题 4：JSP 中动态 INCLUDE 与静态 INCLUDE 的区别有哪些？

答案：动态 INCLUDE 用 jsp:include 动作实现，如：

<jsp:include page="included.jsp" flush="true" />

它总是会检查所含文件中的变化，适合用于包含动态页面，并且可以带参数。

静态 INCLUDE 用 include 伪码实现，如：

<%@ include file="included.htm" %>

它不会检查所含文件的变化，适用于包含静态页面。

面试例题 5：两种跳转方式分别是什么？有什么区别？

答案：两种跳转方式分别为如下。

```
<jsp:include page="included.jsp" flush="true">
<jsp:forward page= "nextpage.jsp"/>
```

前者页面不会转向 include 所指的页面，只是显示该页的结果，主页面还是原来的页面。执行完以后还会回来，相当于函数调用，并且可以带参数。后者完全转向新页面，不会再回来，相当于 GOTO 语句。

面试例题 6：下面不属于 JSP 内置对象的是（　　）？

A．config　　B．page　　C．cookie　　D．exception

解析：JSP 共有以下 9 种基本内置组件，包括：

① request：用户端请求，此请求会包含来自 GET/POST 请求的参数；
② response：网页传回用户端的回应；

③ pageContext：网页的属性是在这里管理的；
④ session：与请求有关的会话期；
⑤ application servlet：正在执行的内容；
⑥ out：用来传送回应的输出；
⑦ config：servlet 的构架部件；
⑧ page：JSP 网页本身；
⑨ exception：针对错误网页，未捕捉的例外。

答案：C

面试例题 7：使用 JSP 代码和使用 JavaScript 代码进行表单数据验证有什么不同?

答案：JavaScript 为脚本程序，可以在表单提交前在客户端验证，提高了验证速度，不会使表单内的信息丢失。JSP 验证需要向服务器请求，服务器对请求页面进行重新编译，并验证数据的合法性，使得服务器负担加重，且验证结果返回较慢，容易使填写的表单信息丢失。

18.2　Servlet

面试例题 1：Servlet 是什么?

答案：Servlet 是使用 Java Servlet 应用程序设计接口（API）及相关类和方法的 Java 程序。除了 Java Servlet API，Servlet 还可以使用用以扩展和添加到 API 的 Java 类软件包。Servlet 在启用 Java 的 Web 服务器上或应用服务器上运行并扩展了该服务器的能力。Java Servlet 对于 Web 服务器就好像 Java Applet 对于 Web 浏览器。Servlet 装入 Web 服务器并在 Web 服务器内执行，而 Applet 装入 Web 浏览器并在 Web 浏览器内执行。Java Servlet API 定义了一个 Servlet 和 Java 性能的服务器之间的一个标准接口，这使得 Servlet 具有跨服务器平台的特性。

Servlet 通过创建一个框架来扩展服务器的能力，以提供在 Web 上的请求和响应服务。当客户机发送请求至服务器时，服务器可以将请求信息发送给 Servlet，并让 Servlet 建立起服务器返回给客户机的响应。当启动 Web 服务器或客户机第一次请求服务时，可以自动装入 Servlet。装入后，Servlet 继续运行直到其他客户机发出请求。Servlet 的功能涉及范围很广，例如，Servlet 可完成如下功能。

（1）创建并返回一个包含基于客户请求性质的动态内容的完整的 HTML 页面。
（2）创建可嵌入到现有 HTML 页面中的一部分 HTML 页面（HTML 片段）。
（3）与其他服务器资源（包括数据库和基于 Java 的应用程序）进行通信。
（4）用多个客户机处理连接，接收多个客户机的输入，并将结果广播到多个客户机上。例如，Servlet 可以是多参与者的游戏服务器。

（5）在允许单连接方式传送数据的情况下，在浏览器上打开服务器至 Applet 的新连接，并将该连接保持在打开状态。在允许客户机和服务器简单、高效地执行会话的情况下，Applet 也可以启动客户浏览器和服务器之间的连接，可以通过定制协议或标准（如 IIOP）进行通信。

（6）将定制的处理提供给所有服务器的标准例行程序。例如，Servlet 可以修改如何认证用户。

面试例题 2：说一说 Servlet 的生命周期及与 CGI 的区别。

答案：Servlet 有良好的生存期的定义，包括加载和实例化、初始化、处理请求及服务结束。这个生存期由 javax.Servlet，以及 Servlet 接口的 init、service 和 destroy 方法表达。Servlet 被服务器实例化后，容器运行其 init 方法，请求到达时运行其 service 方法，service 方法自动派遣运行与请求对应的 doXXX 方法（doGet、doPost）等，当服务器决定将实例销毁的时候调用其 destroy 方法。

与 CGI 的区别在于，Servlet 处于服务器进程中，它通过多线程方式运行其 service 方法，一个实例可以服务于多个请求，并且其实例一般不会销毁。而 CGI 对每个请求都产生新的进程，服务完成后就销毁，所以效率上低于 Servlet。

面试例题 3：请描述 Java Servlet API 中 forward()与 redirect()的区别?

答案：前者仅是容器中控制权的转向，在客户端浏览器地址栏中不会显示出转向后的地址。后者则是完全的跳转，浏览器将会得到跳转的地址，并重新发送请求链接。这样，从浏览器的地址栏中可以看到跳转后的链接地址。所以，前者更加高效，在前者可以满足需要时，尽量使用 forward()方法，并且，这样也有助于隐藏实际的链接。在有些情况下，比如，需要跳转到一个其他服务器上的资源，则必须使用 sendRedirect()方法。

面试例题 4：请描述 Servlet 生命周期。

答案：Web 容器加载 Servlet，生命周期开始。通过调用 Servlet 的 init()方法进行 Servlet 的初始化。通过调用 Service()方法实现根据请求的不同调用不同的方法。结束服务后，Web 容器调用 Servlet 的 destroy()方法。

面试例题 5：如何实现 Servlet 的单线程模式?

答案：要实现单线程，可以在配置文件中修改 isThreadSafe 属性，修改如下。

```
<%@ page isThreadSafe="false"%>
```

面试例题 6：Servlet 页面间对象传递的方法有几种？

答案：可以通过 request、Session、application、cookie 等方法实现页面间的对象传递。

面试例题 7：JSP 和 Servlet 有哪些相同点和不同点？它们之间的联系是什么？

答案：JSP 是 Servlet 技术的扩展，本质上是 Servlet 的简易方式，更强调应用的外表表达。JSP 编译后是“类 Servlet”。Servlet 和 JSP 最主要的不同点在于，Servlet 的应用逻辑是在 Java 文件中，并且完全从表示层中的 HTML 里分离开来。而 JSP 的情况是 Java 和 HTML 可以组合成一个扩展名为.jsp 的文件。JSP 侧重于视图，Servlet 主要用于控制逻辑。

面试例题 8：举例说说你所知道的会话跟踪技术。

答案：会话作用域也就是 Session，在应用程序中，对于每个新会话，都会创建 Session-scope 对象，并且在会话结束后将其释放。因此，每个活动的会话都有一个对象。会话作用域用于从多个脚本中调用的对象，但只影响一个用户会话。你可以只在需要时才为对象赋予会话作用域。

page 代表与一个页面相关的对象和属性。一个页面由一个编译好的 Java Servlet 类（可以带有任何的 include 指令，但是没有 include 动作）表示。这既包括 Servlet，又包括被编译成 Servlet 的 JSP 页面。

request 代表与 Web 客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面，涉及多个 Web 组件（由于 forward 指令和 include 动作的关系）。

Session 代表与用于某个 Web 客户机的一个用户体验相关的对象和属性。一个 Web 会话可能也经常会跨越多个客户机请求。

application 代表与整个 Web 应用程序相关的对象和属性。这实质上是跨越整个 Web 应用程序，包括多个页面、请求和会话的一个全局作用域。

面试例题 9：Web 程序运行时，1 个客户的请求动作可能会跟他之前的请求有联系，也就是说，Web 程序需要维持一定的状态。Web 容器通过会话机制将一个客户的一系列 Web 请求联系起来。除了（　　），Web 容器通过 3 种途径实现会话。

A．Cookie 机制　　B．URL 重写　　C．Session 机制　　D.隐藏表单输入

解析：Servelt 规范规定：会话的维持由 Servlet 容器与客户端浏览器通过对话协调进行。一次会话是从客户端发送请求开始的，在 Server 端，Servlet 引擎用 httpsession 的一个实例（Session）来记录一次会话。Server 端追踪 Session 按照如下步骤：

① 客户端浏览器发送请求。

Server 端准备了一个 test 用的 cookie 和 jsessionid（string 字段用来标识 Session），

并且把响应页面内的所有 URL 改写（在 URL 尾部加上了 jsessionid），然后发送响应到客户端。

② 客户端再次发送请求（包含了是否允许 cookie 的信息）。注意，这次发送的请求的 URL 是已经被 Server 改写了的（尾部添加了 jsessionid），所以会话是被跟踪了的。Server 响应请求，并且判断测试用的 cookie 是否被客户端浏览器接受。如果 cookie 不被接受，那么继续通过改写 URL 来达到跟踪 Session 的目的。如果 cookie 被接受，则在这次响应中，cookie 将作为首选机制来跟踪 Session。

这样，在两次 Client 和 Server 对话之间，Session 被跟踪了，也就是说会话被维持了。这次会话的结束时刻为：Session 过期或者客户端关闭浏览器。

答案：D

面试例题 10：以下关于 session 的说法正确的是（　　）？

A．session 有超时间隔限制，且间隔不可调整

B．session 用于保持用户状态

C．session 可以通过 cookie 保持

D．用户数据不存储在 session 中

答案：B

面试例题 11：请描述 cookie 和 session 机制的区别与联系。

答案：具体来说，cookie 机制采用的是在客户端保持状态的方案，而 session 机制采用的是在服务器端保持状态的方案。同时也看到，由于采用服务器端保持状态的方案在客户端也需要保存一个标志，所以 session 机制可能需要借助于 cookie 机制来达到保存标志的目的，但实际上它还有其他选择。

正统的 cookie 分发是通过扩展 HTTP 协议来实现的，服务器通过在 HTTP 的响应头中加上一行特殊的指示，以提示浏览器按照指示生成相应的 cookie。然而纯粹的客户端脚本，如 JavaScript 或者 VBScript 也可以生成 cookie。而 cookie 的使用是由浏览器按照一定的原则在后台自动发送给服务器的。浏览器检查所有存储的 cookie，如果某个 cookie 所声明的作用范围大于或等于将要请求的资源所在的位置，则把该 cookie 附在请求资源的 HTTP 请求头上发送给服务器。

session 机制是一种服务器端的机制，服务器使用一种类似于散列表的结构（也可能就是使用散列表）来保存信息。

保存这个 session ID 的方式可以采用 cookie，这样在交互过程中浏览器可以自动按照规则把这个标志发回给服务器。一般这个 cookie 的名字都是类似于 session ID。但

cookie 可以被人为地禁止，则必须有其他机制，以便在 cookie 被禁止时仍然能够把 session ID 传递回服务器。

18.3　JavaScript

面试例题 1：以下代码有什么区别?

```
var a = [];
var a = {};
```

答案：

一个是数组对象，类似 var a = new Array();

一个是对象，类似 var a = new Object();

面试例题 2：Javascript 是面向对象的吗？怎么体现 JavaScript 的继承关系？

解析：谈 JavaScript 的面向对象设计有点牵强，毕竟 JavaScript 语言本身没有高级语言严谨。JavaScript 事实上不存在真正意义上的面向对象。为了让 JavaScript 的编程更方便、更易于管理，使其像高级语言的面向对象上面靠拢。

Javascript 的语法和 C++很接近，不过在类实现中没有使用关键字 Class，实现继承的时候也没有采用传统的 Public 或者 Implement 等所谓的关键字来标识类的实现。Javascript 类实际上是为了方便编程把一些常用的变量、函数集中到一个“集合”里，使用的时候可以方便地查找到与这个“集合”相关的变量或功能。这个“集合”打包了与之相关的变量和功能，实现了功能（成员函数）和变量（属性或字段）的封装。

（1）类的定义：类的定义形式和函数是一样的。

其实函数和类只有在使用的时候才能分辨出来，比如，在一个页面中分别使用它们：

```
function ClassName(){
}
<script language="javascript">
function functionA(){
alert('hello');
}
//A 作为函数来使用
functionA();
//A 作为类来使用，从类 A 派生出一个 obj 对象
var obj=new functionA();
</script>
```

（2）类的成员。

```
function ClassName(){
  //定义公有变量
  this.property1=0;
  //定义公有方法
  this.method1=function(){
  //判断 this.a 是否赋值
  if(this.a != undefined)
   alert(this.a);
}
//另外一种方法的定义形式
this.method3=funcA;
//定义私有成员
var pram1=1;
var method2=function(){
alert('');
}
}
//funcA 是方法 method3 的处理函数
function functionA(){
  alert('');
}
```

对上面代码的解释如下：

this 开头的都是类的成员，而且都是公有（public）的，比如：property1 是类的属性、method1 是类的方法；类的成员不需要使用 var 来定义，没有使用 this 前缀的都是私有变量，比如：pram1 是私有变量，method2 是私有方法；类的属性可以不在类中定义，不要初始化的属性可以不定义，在其他地方仍可以调用，比如：在 method1 里面要输出 a 属性，在整个类里面就没有定义 a 属性，我们在创建对象的时候可以给它赋值。

```
var obj=new ClassName()
obj.a="hello javascript";
obj.method1();
```

类的方法可以通过 this.method=function(){}来定义，比如 method1 方法；也可以通过 this.method=funcName 来定义，将方法指定让某个函数来处理，比如 method3。

（3）类的继承。

Javascript 没有使用“:”或者 Java 的 public 那样类似的关键字，只是通过 newclassname.prototype=new baseclass 这样的方法来完成。

```
function classfunctionA(){
```

```
this.property1='hello';
this.method1=function (){
alert(this.property1);
}
}

function classB(){
}
//继承 classA
classB.prototype=new classfunctionA();

//给 classB 添加 PI 属性
classB.prototype.PI=3.1415926;
//给 classB 添加 showPI 方法
classB.prototype.showPI=function(){
alert(this.PI);
}
```

通过使用 prototype 对象，将类 classA 的实例赋值给 prototype 对象，从而 classB 继承了 classA 的所有成员。比如：

```
classB.prototype=new classfunctionA();
```

同时也可以通过 prototype 在类外给类重新添加成员（这是其他高级语言所没有的功能）。比如：

```
classB.prototype.PI
```

和

```
classB.prototype.showPI
```

（4）类方法的重载。

类方法的重载在类的构造函数中使用比较多，比如：类中有两个同名的方法而不同参数或不同参数的类型。JavaScript 从形式上来说不支持类方法重载，我们可以通过它的 arguments 属性来完成对类方法的重载。

```
function classfunctionA(){
 //获取参数个数
 //注意：this.arguments.length 是错误的
 var num=classA.arguments.length;

this.method1=function(){
if(num==0){
```

```
alert(0);
}
if(num==1){
alert(1);
}
}
}
```

类或函数名的 arguments 属性返回一个数组包含所有的参数，如：classA.arguments.length 可以获取参数的个数，classA.arguments[0]获取第一个参数的值。根据需要，通过获取参数个数或参数值，从而同一个函数或类具有了不同的功能，如：

```
var obj= new classfunctionA();
obj.method1();//输出了 0
var obj1= new classA(5);
obj1.method1();//输出了 1
```

（5）对象的定义。

```
//对象是类的实例，定义一个类作为一个对象的模板
function functionA(){
this.a=1;
this.b=2;
this.add=function(){
return this.a+this.b;
}
}
//定义一个对象
var obj=new functionA();
//赋值类属性
obj.a=5;
obj.b=6;
//调用类方法
var sum=obj.add();

//另外一种定义方法：
var obj={
a:1,
b:2,
add:function(){
return this.a + this.b;
}
}
```

面试例题 3：请写出 HTML 中打开一个模式窗口和非模式窗口的 Javascript 代码。

答案：模式窗口和非模式窗口分别是：

window.showModalDialog()
window.showModelessDialog()

面试例题 4：在 HTML 中，有一个城市的下拉列表框的代码为:

```
<select id="country">
    <option value="1">广东 <option>
    <option value="2">海南 <option>
</select>
```

请用 Javascript 分别写出如何得到当前选中城市的 value 和名称(名称是广东和海南)。

答案：

```
document.getElementById("country").value
document.getElementById("country").options
[document.getElementById("country").selectedIndex].text
```

面试例题 5：下面的 Javascript 代码输出结果是多少?

```
<script type="text/javascript">
var x = 1;
var y = 0;
var z = 0;
function add(n){n=n+1;}
y = add(x);
function add(n){n=n+3;}
z = add(x);
</script>
```

求 y 和 z 的值是多少?

A．undefined undefined　　B．4 4

C．1 4　　D．1 1

解析：本题的迷惑性很大，解题时一定要看清楚。function add(n)是没有返回值的，所以 y 和 z 都会是 undefined。

答案：A

扩展知识 1:

如果将 add 函数改为有返回值的情况:

```
function add(n){return n=n+1;}
```

和

```
function add(n){return n=n+3;}
```

y 和 z 都会是 4，因为后面定义的 add 会覆盖前面定义的 add。

扩展知识 2：再把本题修改一下，求 y+z 之和是多少？

```
<script type="text/javascript">
var x = 1;
var y = 0;
var z = 0;
function add(n){n=n+1;}
y = add(x);
function add(n){n=n+3;}
z = add(x);
s=y+z;
</script>
```

y 和 z 都是 undefined，那么 s 自然也就不会是一个数字，所以 s 应该是 NaN。

面试例题 6：下面 JavaScript 代码的输出结果是多少？

```
var str = "abcdefgdafoadfjladsf";
var strmp = str.split("f");
var c = strmp[4]+"5*10"+5*10;
```

解析：var str = "abcdefgdafoadfjladsf"; var strmp = str.split("f"); 的意思是被拆成字符数组。strmp[0]= "abcde"，strmp[1]= "gda"，strmp[2]= "oad"，strmp[3]= "jlads"，strmp[4]= " "

var c = strmp[4]+"5*10"+5*10;//实际上相当于""+"5*10"+50，最后结果为 5*1050。

答案：5*1050

面试例题 7：下面的 JavaScript 代码

```
Var a =[[0,1],[2,3]]
Var b = a[0][1];
Var c = a[0,1]
Var d = [0]
```

b/c/d 是多少？

解析：b = a[0][1]，所以 b 的值是 1；Var c = a[0,1]，所以 c 的值是 1；d 的值是 0。b/c/d=1/1/0，被除数为 0，结果为 NaN，不是一个数。

答案：NaN

面试例题 8：下面 JavaScript 代码

```
var str = new Array(0,1,2,3,4,5,6,7,8,9,10)
var b = str[5]%str[1]
```

请问 b 是多少?

解析：结果为 0，str[5]为 5，str[1] 为 1，5 除以 1，所以余数为 0。

答案：0

面试例题 9：下面这段 JavaScript 代码

```
function test(){
alert(1);
}

test();
function test(){
alert(2);
}
```

请问输出结果是多少?

A．1　　B．2　　C．null　　D．undefined

解析：如果只是重复绑定函数，后面的优先级应该比前面定义的高，这些内容是一个 Java 程序员该会的东西，输出结果为 2。

答案：B

面试例题 10：请问在 JavaScript 语言中 alert('\n'==0)的输出结果是多少?

A．null　　B．true　　C．undefined　　D．false

解析：alert('\n'==0)的结果是 true。

对于“==”、“!=”这两种情况而言，以下情况认为是==的：

① 如果两个表达式的类型不同，则试图将它们转换为字符串、数字或 Boolean 量。

② 负零等于正零。

③ null 与 null 和 undefined 相等。

④ 相同的字符串、数值上相等的数字、相同的对象、相同的 Boolean 值或者（当类型不同时）能被强制转化为上述情况之一，均被认为是相等的。

其他比较均被认为是不相等的。如 NaN 与包括其本身在内的任何值都不相等。

JavaScript 手册说得很清楚：如果两表达式的类型不同，则试图将它们转换为字符串、数字或 Boolean 量。至于转变的顺序，是这样规定的：一般情况下，如果类型不同时，主要看两个不同的类型是否都可以转换为字符串或数字，如果不能转为第①种

的时候，就都转为 Boolean 后再比较，如果可以转为字符串或数字，那么在这个时候就不需要区分其顺序了，只需要如第④点说的那样，看它们转换后是否满足：相同的字符串、数值上相等的数字、相同的对象、相同的 Boolean 值，如果是不相同的，不管是哪一种，它们的结果都为 false。

所以这个关键就是看两个不同的类型是否都可以转换为字符串、数字或对象，如果其中有一个不能转换为这两种中的任何一种，那么它们就会转为 Boolean。

像'\n'==0 这种情况是转换为 Boolean，因为\n 既不是字符串，又不是数字，而 0 又只能转为数字、字符串或 Boolean。而且像这种情况的\n 只能转为 false。两个 false 相等，则输出结果是 true。

答案：B

面试例题 11：尝试实现注释部分的 JavaScript 代码，可在其他任何地方添加更多的代码（如不能实现，说明不能实现的原因）。

```
var Obj = function(msg){
this.msg = msg;
this.shout = function(){
  alert(this.msg);
}
this.waitAndShout = function(){
//隔五秒钟后执行上面的 shout 方法
```

解析：本题出得很好。如果写大型的页面，不同的人写出来的代码是有很大差距的。

答案：可以实现，其代码如下：

```
var Obj = function(msg) {
    this.msg = msg;
    this.shout = function() {
        alert(this.msg);
    }
    this.waitAndShout = function() {
    //隔五秒钟后执行上面的 shout 方法
    var me = this;
    setTimeout(function(){me.shout();me.waitAndShout();},5000);
    }
}
```

面试例题 12：Read the following javascript code（读下列 JavaScript 代码）：

```
someText = 'Web2.0';
pattern = /(\w+)(\d)\.(\d)/i;
outCome = pattern.exec(someText);
```

What is outCome[0]?（下列哪个全部匹配？）

A．Web2.0　　B．false　　C．null　　D．Web

解析：exec 是正则表达式的方法，执行效果是对字符串做正则匹配，以数组形式返回匹配的字符串段。outCome[0]是全部匹配；outCome[1]是第 1 个括号的匹配；outCome[2]是第 2 个括号的匹配；outCome[3]是第 3 个括号的匹配。输出代码如下：

```
someText = 'Web2.0';
pattern = /(\w+)(\d)\.(\d)/i;
outCome = pattern.exec(someText);
alert(outCome[0])//Web2.0
alert(outCome[1])//Web
alert(outCome[2])//2
alert(outCome[3])//0
```

答案：A

扩展知识：请问下面代码 outCome_exec 和 outCome_matc 的输出结果是多少？

```
var someText="web2.0 .net2.0";
var pattern=/(\w+)(\d)\.(\d)/g;
var outCome_exec=pattern.exec(someText);
var outCome_matc=someText.match(pattern);
```

JavaScript 中与正则表达式有关的匹配字符串的函数主要有 RegExp 类的方法 exec(string)以及 String 类的方法 match(regex)，二者的特点如下：

（1）exec 是正则表达式方法，它的参数是字符串，如下所示：

```
var re=new RegExp(/\d/);
re.exec( "jinder22" );
```

match 是字符串类方法，它的参数是正则表达式对象，如下所示：

```
"jinder22".match(\d);
```

（2）exec 和 match 返回的都是数组

exec 方法的正则表达式没有分组，如果匹配，则返回只有一个元素的数组，这个数组唯一的元素就是该正则表达式匹配的第一个串；如果没有匹配，则返回 null。

下面两个 alert 函数弹出的信息是一样的：

```
var str= "dog,log" ;
var p=/og/; //没有 g 属性
alert(p.exec(str))
alert(str.match(p))
```

都是"og"。在这种场合下，exec 等价于 match，但如果正则表达式是全局匹配（g 属性）的，那么以上代码的结果就不一样：

```
var str= "dog,log" ;
var p=/og/g; //注意 g 属性
alert(p.exec(str))
alert(str.match(p))
```

分别是

```
"og"
"og,og"
```

因为 exec 永远只返回第一个匹配，而 match 在正则表达式中指定了 g 属性的时候，会返回所有的匹配。

（3）exec 如果找到了匹配，而且包含分组的话，返回的数组将包含多个元素，第一个元素是找到的匹配，之后的元素依次为该匹配中的第一、第二、...个分组（反向引用），如下的代码将弹出"dog2,og"：

```
var str= "dog2,log8" ;
var p=/d(og)\d/;
alert(p.exec(str))
```

其中第一个元素是匹配的字符串"dog2"，之后的元素是括号中匹配的"og"。一些演示代码如下：

第 1 组：

```
var someText="web2.0 .net2.0";
var pattern=/(\w+)(\d)\.(\d)/;
var outCome_exec=pattern.exec(someText);
var outCome_matc=someText.match(pattern);
document.write(outCome_exec)
document.write(outCome_matc)
```

结果为

```
web2.0,web,2,0
web2.0,web,2,0
```

第 2 组：

```
var someText="web2.0 .net2.0";
var pattern=/(\w+)(\d)\.(\d)/g;
var outCome_exec=pattern.exec(someText);
var outCome_matc=someText.match(pattern);
document.write(outCome_exec)
```

```
document.write(outCome_matc)
```

结果为

```
web2.0,web,2,0
web2.0,net2.0
```

分析如下：

outCome_exec 的值：pattern 中的 g 属性对 exec 函数无效。所以第 1 组和第 2 组的情况都一样：exec 匹配第一个可以匹配的字串"web2.0"，作为其返回数组的第一个元素，另外，由于 pattern 中包含三个分组((\w +)、(\d)、(\d))，因此，该数组还将包含三个元素，依次是"web"、"2"、"0"，所以该 exec 执行后的最终结果是["web2.0","web","2","0"]。pattern 没有 g 属性，那么它将与 outCome_exec 的结果一样。

outCome_matc 的值：对于第 1 组情况，pattern 没有 g 属性，那么它将与 outCome_exec 结果一样。对于第 2 组情况，由于 pattern 是全局匹配的，因此，match 匹配了所有可以匹配的字串，第 2 组 outCome_matc 的结果为["web2.0","net2.0"]。

面试例题 13：JavaScript 怎样选中一个 checkbox，怎样设置它无效？

解析：对于本题，某些面试者想当然地用 checkbox 的 checked 为 false。这样并不能达到本题的目的，这是因为：

```
document.getElementByName("checkbox")[i].checked=false;
```

是设置为不选择，但还可以再选择操作。

```
document.getElementByName("checkbox")[i].disabled=true;
```

是设置失效，使图标变灰了（类似只读），不能再进行选择操作。

答案：设置 checkbox 的 disabled 为 true，比如下面的代码：

```
<html>
<head>
<script type="text/javascript">
function x(){
   document.all.cb1[0].disabled=true;
}
</script>
</head>
<body onload="x()">
<input name="cb1" type="checkbox">
</body>
```

```
</html>
```

面试例题 14：用 JavaScript 创建一个 XML 文件 c:/test.xml，为该文件增加三个节点 <a id="1"/>，<b id="2"/>，<c id="3">，将 b 的 attribute 改为 4 并保存。

答案：代码如下：

```
<script language=javascript>
//创建 XML
function CreateXML()
{
var fso;
fso = new ActiveXObject("Scripting.FileSystemObject");
xmlfile = fso.CreateTextFile("c:\\"+"123.XML",true,true);
xmlfile.Close();
}
//写 XML 内容
var buffer="<customer><a id='1'/><b id='2'/><b id='3'/></customer>"
var xmlDoc = new ActiveXObject("Msxml2.DOMDocument.4.0");
xmlDoc.load("c:\\123.xml");
xmlDoc.async = false;
var arr = buffer.split(">");
xmlDoc.loadXML(buffer);
window.alert(xmlDoc.xml);
//windows 认为在客户端对硬盘写操作时不安全的，这个操作应该放在服务器上
//xmlDoc.save("c:\\123.xml");

</script>
```

面试例题 15：怎样通过代码实现图片每秒闪烁 2 次?

答案：

```
<SCRIPT LANGUAGE="JavaScript">
function fun()
{
var img = document.getElementById("img");
img.style.display=img.style.display=="none"?"":"none"
}
</SCRIPT>
<imgid="img"src="http://www.example.com/img/logo.gif"onload="setInterval(fun,250)">
```

18.4 XML

面试例题 1：XML 和 HTML 的区别是什么？请写出包含一个人的姓名、年龄、性别的 XML 简单元素。

解析：XML 的书写格式与 HTML 十分相似，这是因为 XML 与 HTML 都是标准通用标记语言（SGML）的下一代语言。写法虽然相似，两者却有着重要的区别。大家可以理解为：HTML 是描述文本结构和样式的，主要用于文本的显示和控制；而 XML 是一个协议语言，用来描述数据和数据结构，是一种描述数据的标准，同时必须依赖高级语言才能实现其作用。

一个 XML 文档是由一些简单或复杂元素构成的，其结构与 HTML 基本相同，都包含元素、属性和值。XML 元素都是用一个标签开始，并用一个相应的标签表示结束。复杂元素中可以包含子元素，而简单元素中就只能包含属性和内容。

答案：在 XML 文档与 HTML 文档的写法规则上，存在的差别包括以下几个方面。

（1）XML 文档中区分大小写，而 HTML 文档中并不区分大小写。

（2）在 HTML 文档中当存在列表键时，可以省略结尾标记，而在 XML 文档中，所有的标记都不能省略。

（3）在 XML 文档中，属性值必须写在引号中，而在 HTML 文档中并没有严格规定。

一个 XML 简单元素的事例如下：

```
<?xml version="1.0" encoding="gb2312"?>
<people>
    <person>
        <name>zhangsan</name>
        <sex>male</sex>
        <age>24</age>
    </person>
    <person>
        <name>Lisi</name>
        <sex>male</sex>
        <age>22</age>
    </person>
</people>
```

面试例题 2：XML 的命名空间是什么？有什么作用？

答案：XML 的命名空间是 W3C 推出的一个标准，是用来统一命名 XML 文档中的元素和属性的机制。使用命名空间可以明确表示出 XML 文档中的元素、属性及其他标记，可以避免名称之间冲突所带来的问题。其实质就是在元素、属性名之前加上特定的命名空间，使它们都具有自己的领域，同时也方便程序员查询和提取数据。在 XML 文档中，一般都用 URL 来表示命名空间，但是一般的 URL 都比较长，放在元素和属性名称前难于书写。因此，会使用一个简短的字符来代替，并称为命名空间的前缀。

面试例题 3：XML 有哪些解析技术？它们之间的区别是什么？

答案：有 DOM、SAX 等解析技术。

DOM：处理大型文件时，其性能下降得非常厉害。这个问题是由 DOM 的树结构所造成的，这种结构占用的内存较多，而且 DOM 必须在解析文件之前把整个文档装入内存，适合对 XML 的随机访问。

SAX：不同于 DOM，SAX 是事件驱动型的 XML 解析方式。它顺序读取 XML 文件，不需要一次全部装载整个文件。当遇到像文件开头、文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理 XML 文件，适合对 XML 的顺序访问。

面试例题 4：你在项目中用到了 XML 技术的哪些方面？是如何实现的？

答案：用到了数据存储和信息配置两方面。在做数据交换平台时，将不同数据源的数据组装成 XML 文件，然后将 XML 文件压缩打包加密后通过网络传送给接收者，接收后再对 XML 文件中的相关信息进行处理。在做软件配置时，利用 XML 可以很方便地进行，软件的各种配置参数都存储在 XML 文件中。

面试例题 5：An Intranet Web application provides news for thousands of users inside one company. The application gets XML data from a database and then thansforms the data into HTML with server-side XSLT. Currently the web application performs poorly. Which of the following actions should the application developer to address the problem?（一个企业内部网络应用软件为公司内部数千名用户提供新闻。这个应用软件从数据库中得到 XML 数据，然后用 XSLT 转换成 HTML。现在这个软件的效率低下，如果你是工程师，你将如何解决这个问题？）

A．Update the database to a higher version.（升级数据库到更高版本）

B．Use well formed XHTML tags in the application（在这个软件中使用完好的 XHTML 表格标号）

C．Supply the XML dat and the Style Sheet to Web browsers（把 XML 和样式表提供给浏览器来进行转换）

D．still use XSL-FO to make the transformation in database .（仍然在数据库中使用 XSL-FO 来执行转换）

解析：XSL-FO 是用于格式化 XML 数据的语言，全称为 Extensible Stylesheet Language Formatting Objects（格式化对象的可扩展样式表语言），是 W3C 参考标准，现在通常叫做 XSL。

XSL-FO 文档是带有输出信息的 XML 文件。它们包含着有关输出布局以及输出内容的信息。XSL-FO 文档存储在以.fo 或.fob 为后缀的文件中。以.xml 为后缀存储的

XSL-FO 文档也很常见，这样做可以使 XSL-FO 文档更易被 XML 编辑器存取。

XSLT 是一种用于将 XML 文档转换为 XHTML 文档或其他 XML 文档的语言。

XSLT 指 XSL 转换（XSLTransformations）。XSLT 使用 XPath 在 XML 文档中进行导航。XPath 是一个 W3C 标准。在转换过程中，XSLT 使用 XPath 来定义源文档中可匹配一个或多个预定义模板的部分，一旦匹配被找到，XSLT 就会把源文档的匹配部分转换为结果文档。

这个系统现在的运行效率低下，不是出了问题，而且题目还特别提到了 XSLT 转换 XML 的工作是在服务器端进行的，这也许是一个暗示。选项 C 是在说把 XML 和样式表（应该是指的 XSLT）提供给浏览器（即让浏览器来进行转换）是比较有效的解决方案；至于选项 D，如果还是在服务器端进行转换，用 XSLT 和 XSL-FO 不会有本质上的改善。

答案：C

面试例题 6：下列陈述中哪个是不正确的？

A．如果一个程序员想在选择了系统关闭菜单后让一个框架关闭，那么这个过程必须写代码来实现

B．框架的默认布局管理是 BorderLayout 布局

C．框架的布局管理在被指定后就不能被改变

D．GridBagLayout 利用了 GridBagConstraints 类的扩展

解析：答案 C 不正确，因为不只是框架的布局，还有其他任何容器的布局管理，你想在任何时候改变都可以。

答案：C

18.5　APPLET

面试例题 1：下列陈述中哪个是正确的？（多选）

A．一个 APPLET 默认的布局管理是 FlowLayout

B．一个 FRAME 默认的布局管理是 FlowLayout

C．在 SETSIZE 方法被调用之前，一个布局管理必须分配给一个 APPLET

D．流动布局管理器的优点是使任意组件达到最佳大小

解析：一个应用程序默认的布局管理是 BorderLayout，如果一个 APPLET 没有应用任何布局管理，那么它默认的布局管理是 FlowLayout。Java 语言中提供的布局管理器种类有：边界式布局、卡片式布局、流式布局和网格式布局等。它们各有不同的特点，可根据实际需要选用。

答案：A，D

第19章

Struts 结构设计

Struts 跟 Tomcat、Turbine 等诸多 Apache 项目一样，是开源软件，这是它的一大优点，使开发者能更深入地了解其内部实现机制。除此之外，Struts 的优点主要集中体现在两个方面：TagLib 和页面导航。TagLib 是 Struts 的标记库，灵活运用能大大提高开发效率。页面导航使系统的脉络更加清晰，通过一个配置文件，即可把握整个系统各部分之间的联系，这对于后期的维护有着很大的好处，尤其是当另一批开发者接手这个项目时，这种优势体现得更加明显。

以下的考题来自真实的笔试资料，希望读者先不要看答案，自我解答后再与答案加以比对，找出自己的不足。

19.1 AWT

面试例题 1：请编程实现以下功能。

产生一个包含两个按钮的 Frame：一个是 Sleep 按钮，一个是 Hello 按钮。Hello 按钮只是简单地在控制台上打印“Hello World”。Sleep 按钮使它所在的线程（AWT 事件处理线程）休眠 5 秒钟。在你按下 Sleep 按钮 5 秒钟之内，无论你单击多少次 Hello 按钮，程序都不会有任何响应。如果你在这期间按下了 Hello 按钮 5 次，那么“Hello World”将会立即被连续打印 5 次。

请用 AWT 多线程实现。

解析：Java 程序除了小部分非常简单的控制台程序都是基于多线程的，原因在于 Java 的 Abstract Windowing Toolkit（AWT）和它的扩展 Swing。AWT 用一个特殊的线程处理所有的操作系统级的事件，这个特殊的线程是在第一个窗口出现的时候产生的。

因此，几乎所有的 AWT 程序都至少有两个线程在运行，分别是 main 函数所在的线程，以及处理来自 OS 的事件和调用注册监听者的响应方法（也就是回调函数）的 AWT 线程。必须注意的是，所有注册的监听者方法运行在 AWT 线程上，而不是人们一般认为的 main 函数线程（这也是监听器注册的线程）上。

这种架构有个问题：虽然监听器的方法是运行在 AWT 线程上的，但是它一般会非常频繁地访问它的外部类，也就是运行在 main 线程上的类的成员变量。当这两个线程竞争（compete）访问同一个对象实例（Object）时，会引起非常严重的线程同步问题。适当地使用关键字 synchronized 是保证两个线程安全访问共享对象的必要手段。

除此之外，AWT 线程不但处理监听器方法，还响应来自操作系统的事件。这就意味着，如果监听器方法占用大量的 CPU 时间来进行处理，则你的程序将无法响应操作系统级的事件（例如，鼠标单击事件和键盘事件）。这些事件将会被阻塞在事件队列中，直到监听器方法返回。具体的表现就是系统的死锁。该题就是让我们通过线程的 Sleep 操作模拟死锁的过程。

答案：

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class Hang extends JFrame

{

   public Hang()

   {JButton b1 = new JButton( "Sleep" );

      JButton b2 = new JButton( "Hello" );
      b1.addActionListener
      (new ActionListener()
         {public void actionPerformed(.ActionEvent event )
            {try
               {
                 Thread.currentThread().sleep(5000);
               }
               catch(Exception e){}
            }
         }
      );

      b2.addActionListener
      (new ActionListener()
```

```
            {public void actionPerformed( ActionEvent event )
               {System.out.println("Hello world");
               }
            }
        );
        getContentPane().setLayout( new FlowLayout() );
        getContentPane().add( b1 );
        getContentPane().add( b2 );
        pack();
        show();
    }
    public static void main( String[] args )
{
new Hang();
    }
}
```

19.2 Struts 体系结构

面试例题 1：J2EE 号称多层结构，为什么多层比两层好？典型的 J2EE/Java EE 至少划分几个层次？你如何根据自己项目的特点进行选择框架设计？

解析：这是面试题中非常开放的问题，100 个人恐怕有 100 种答案。这里的答案仅供参考。

答案：高质量的 J2EE/Java EE 系统标准实际就是 OO 设计的标准，松耦合是 OO 设计的主要追求目标之一，那么无疑解耦性成为衡量 J2EE/JEE 质量的首要标准。在实际选择中，还需要兼顾可伸缩性、性能、开发效率等方面综合考虑。为什么多层比两层好？因为多层结构解耦性好，使得维护与拓展方便、灵活。

典型的 J2EE/Java EE 至少划分 3 个层次，分别是表现层、业务逻辑组件层和持久层。

面试例题 2：表现层架构一般实现了哪些功能？不同表现层架构各自的特点都是什么？

解析：目前 J2EE 表现层的框架技术是：Struts、Tapestry 和 JSF。从诞生时间上看，Struts 应该比较早，使用得非常广泛。Tapestry 3.0 逐渐引起广泛的重视。随着 JSF 1.1 标准的推出，JSF 开始正面出击，逐渐受到重视。

其实，JSF 和 Tapestry 两者还是各有侧重点的，不过差别比较细微，但是这种细微点在实现一个大工程时可能带来不同的感受和变化。

图 19-1 从一个高度抽象了表现层框架应有的技术架构，可以说是所有表现层框架技术都必须实现的功能架构图。

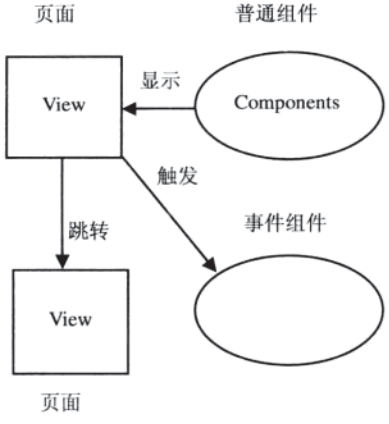


图 19-1　表现层框架技术架构

答案：一个表现层框架无外乎要实现图中的 3 个功能。

（1）在当前页面能够显示一个组件对象的内容，而不是像纯 JSP 那样，需要在 JSP 页面写入“调用对象方法”的 Java 代码。

（2）当用户按下页面的提交按钮或链接后，事件发生，这时应该触发服务器端，并将当前页面的参数提交给服务器。这种机制表现在 Form 表单提交和有参数的链接。

（3）从一个页面视图直接跳转到另外一个页面视图，起单纯的导航作用。

通过表 19-1 来比较这 3 种框架在实现图 19-1 中各个功能时的技术细节，从而得出它们的异同点和偏重点。

表 24-2　3 种框架的异同点和偏重点

	Struts	Tapestry 3.0	JSF
在 View 显示的组件要求	组件必须继承 ActionForm	分显式调用和隐式调用，组件必须继承 BaseComponent	普通 POJO 无须继承 Managed Bean
组件在 View 显示粒度	View 页面只能显示与表单对应	可将组件嵌入页面任何一行，对使用组件数量无限制	同 Tapestry
页面分区 Tiles	使用 Tiles 标签库实现，需要另外的 tiles-def.xml 配置文件	组件有自己的视图页面，通过调用组件即直接实现多个页面组合。强大自然的页面组合是其特点	通过组件+标签库实现 Subview，但如需重用 Layout，还要结合 Tiles

续表

	Struts	Tapestry 3.0	JSF
页面跳转	使用标签库 html:link 中写明目标 URL，URL 名称需要对照配置文件的 path 命名，与组件 Action 耦合	URL 名称是目标的组件名称，不涉及 URL 和路径等操作，方便稳固	类似 Struts，也需要在配置文件中查找，与组件分离
参数传递	使用 html:link 时传递参数超过一个以上处理麻烦	直接调用组件，直接赋予参数，没有参数个数限制	参数分离传递给组件
事件触发	通过表单提交 submit 激活，不能细化到表单里的字段	能够给表单的每个字段贴一个事件，事件组件必须实现 PageListener 接口	同 Tapestry，事件组件必须实现 ActionListener 接口

扩展知识（Struts 组件编程模型）

Struts 实现组件编程时有一些复杂，经常为一个页面中需要引入多个组件而头疼，因为 Struts 中无法直接引入多个组件，必须绕一些圈子。

一般分两种情况。如果同一个 Action 就可以对付这些组件，那么在这种情况下有两个办法。

（1）将多个组件装入一个 ActionForm 中，如使用 MapForm 等机制。

（2）手工将多个组件装入 request、session 等 scope 中，然后根据其名称在 JSP 中获得。

这两种方法都有缺点。第一种办法的缺点是经常会把一个框架弄得面目全非，变成一个大杂烩，违反了 OO 分派封装的原则。第二种办法其实又回到 JSP 编程。

对于第二种情况，如果这些组件必须预先由不同的 Action 来处理，每个组件必须经过 Action →ActionForm 流程，在这种情况下有两种办法：

（1）使用 Tiles，将不同流程输出到同一个页面的不同区域。这是一种并行处理方式。

（2）对多个流程首尾相连，第一个 Action forward 结果是第二个 Action，最后输出一个 JSP，在这个 JSP 中就可以使用前面多个流程的多个 ActionForm 了。这属于串行方式。

第20章

Java 架构技术及相关中间件

在软件开发的过程中，人们越来越意识到软件重用的重要性。异构的系统、不同的实现方案使软件的重用变得复杂。在中间件产生以前，应用软件不得不直接面对非常底层的东西。不同的硬件体系、不同的操作系统、不同的网络协议实现和不同的数据库等，这些使得应用程序复杂多变。面对易变的东西，软件设计师们已经习惯于通过添加中间层的方式来隔离变化。把应用软件所要面临的共性问题进行提炼、抽象，在操作系统之上添加一个可复用的部分，供成千上万的应用软件重复使用。这一技术思想最终构成了中间件。一方面，中间件要应对底层不同的环境，针对不同的环境进行不同的调用；另一方面，中间件要对上层提供统一的接口，保证在不同的环境中为上层提供相同行为的服务。具体地说，中间件屏蔽了底层操作系统的复杂性，使程序开发人员面对一个简单而统一的开发环境，减少程序设计的复杂性，将注意力集中在自己的业务上，不必再为程序在不同系统软件上的移植而重复工作，大大减少了技术上的负担。

中间件带给应用系统的不只是开发的简便、开发周期的缩短，也减少了系统的维护、运行和管理的工作量，还减少了计算机总体费用的投入。由于采用了中间件技术，应用系统的总建设费用可以减少 50%左右。在网络经济、电子商务大发展的今天，从中间件获得利益的不只是 IT 厂商，IT 用户同样是赢家。其次，中间件作为新层次的基础软件，其重要作用是将不同时期、在不同操作系统上开发的应用软件集成起来，彼此像一个天衣无缝的整体协调工作，这是操作系统、数据库管理系统本身做不了的。中间件的这一作用使得在技术不断发展之后，我们以往在应用软件上的劳动成果仍然物有所用，节约了大量人力、财力的投入。

面试中将对中间件的类型、各种应用有所考核，读者应事先做好这方面的准备。

20.1 WebLogic

面试例题 1：如何给 WebLogic 指定内存的大小?

答案：在启动 WebLogic 的脚本中（位于所在 Domain 对应服务器目录下的 startServerName），增加 set MEM_ARGS=-Xms32m -Xmx200m，可以调整最小内存为 32MB，最大为 200MB。

面试例题 2：如何设定 WebLogic 的热启动模式（开发模式）与产品发布模式?

答案：可以在管理控制台中修改对应服务器的启动模式为开发或产品模式之一，或者修改服务的启动文件或 commenv 文件，在里面增加：set PRODUCTION_ MODE=true。

面试例题 3：如何在启动时不需输入用户名与密码?

答案：修改服务启动文件，增加 WLS_USER 和 WLS_PW 项。也可以在 boot.properties 文件中增加加密过的用户名和密码。

面试例题 4：在 WebLogic 管理控制台中对一个应用域（或者说是一个网站，Domain）进行 JMS 及 EJB 或连接池等相关信息配置后，实际保存在什么文件中?

答案：保存在此 Domain 的 config.xml 文件中，它是服务器的核心配置文件。

面试例题 5：如何查看在 WebLogic 中已经发布的 EJB?

答案：可以使用管理控制台，在它的 Deployment 中可以查看所有已发布的 EJB。

面试例题 6：在 WebLogic 中发布 EJB 需要涉及哪些配置文件?

答案：不同类型的 EJB 涉及的配置文件不同，都涉及的配置文件包括 ejb-jar.xml 和 weblogic-ejb-jar.xml，CMP 实体 Bean 一般还需要 weblogic-cmp-rdbms-jar.xml。

面试例题 7：EJB 需要直接实现它的业务接口或 Home 接口吗？请简述理由。

答案：远程接口和 Home 接口不需要直接实现，它们的实现代码是由服务器产生的，程序运行中对应实现类会作为对应接口类型的实例被使用。

面试例题 8：说说在 WebLogic 中开发消息 Bean 时的 persistent 与 non-persistent 的差别。

答案：persistent 方式的 MDB 可以保证消息传递的可靠性，也就是即使 EJB 容器出现问题，而 JMS 服务器依然会将消息在此 MDB 可用的时候发送过来，而 non-persistent 方式的消息将被丢弃。

面试例题 9：如何远程启动 WebLogic 服务？

答案：用 Telnet 远程控制服务器，远程启动 WebLogic 服务，启动后关闭 Telnet，WebLogic 服务也跟着停止，这是因为使用 Telnet 启动的进程会随着 Telnet 进程的关闭而关闭。所以，我们可以使用 UNIX 下的一些命令来做到不关闭。

使用如下命令：

```
nohup startWeblogic.sh&
```

如果想要监控标准输出，可以使用：

```
tail -f nohup.out
```

面试例题 10：如何限制公网用户访问 WebLogic 的控制台？

答案：WebLogic（版本 6.1）应用部署在内部网上，通过防火墙映射到公网上，但公网用户通过键入域名：www.xxx.com/console，就可进入 WebLogic 的登录页面，用户可猜测管理员的密码。屏蔽公网用户对 WebLogic 控制台的访问有以下方法。

方法 1：在控制台上单击左边的 Domain，将 Console Enabled 选项去掉，这样就完全不能使用 console 了。

方法 2：将“console”改名，改“Console Context Path”的“console”为一个稀奇古怪的名字就可以了。

方法 3：不要给 WebLogic 公网 IP，通过一个有公网 IP 的 Apache 等 Proxy 来访问 WebLogic。

方法 4：启动 Administration Port。

方法 5：应用不发布在 Admin Server 上，Admin Server 在外网不可见。

面试例题 11：如何停止 WebLogic 服务？

答案：直接杀死进程不是标准的做法，应该使用如下 Java 命令。

```
java -classpath weblogic.jar;%CLASSPATH% weblogic.Admin -url
  <host_name>:<port_number> SHUTDOWN -username <system _user_name>
  -password <system_user_password>
```

例如：

```
java -classpath weblogic.jar;%CLASSPATH% weblogic.Admin -url
  192.168.0.1:7001

SHUTDOWN -username system -password password
```

其中，如果 SHUTDOWN 关不掉，可以使用 FORCESHUTDOWN 代替 SHUTDOWN 来强制关掉服务器。

面试例题 12：应用管理 JNDI 里面加和不加“java:comp/env/”前缀有什么区别？

答案：“java:comp/env”是标准的 J2EE 环境查找规则，使用这种方式必须做一次环境名到 JNDI 名的映射，这种隔离使得在写程序时不必关注真正的 JNDI 名字，这与把 JNDI 名放到配置文件里是一样的，用法如下。

下例中把 java:comp/env/my/datasource 映射到 my.ora.datasource。

```
web.xml
<resource-ref>
<res-ref-name>my/datasource</res-ref-name>
<res-type>javax.sql.DataSource</res-type>
<res-auth>CONTAINER</res-auth>
</resource-ref>
```

不使用这个前缀的，其实就是直接的 JNDI 名，如下所示。

```
weblogic.xml
<reference-descriptor>
<resource-description>
<res-ref-name>my/datasource</res-ref-name>
<jndi-name>my.ora.datasource</jndi-name>
```

面试例题 13：WebLogic 如何设置 session 超时时间？

答案：

1．web.xml

设置 Web 应用程序描述符 web.xml 里的<session-timeout>元素。这个值以分钟为单位，并覆盖 weblogic.xml 中的 TimeoutSecs 属性。

```
<session-config>
  <session-timeout>54</session-timeout>
</session-config>
```

此例表示 Session 将在 54 分钟后过期。

若<session-timeout>设置为-2，表示将使用在 weblogic.xml 中设置的 TimeoutSecs 这个属性值。

若<session-timeout>设置为-1，表示 Session 将永不过期，而忽略在 weblogic.xml 中设置的 TimeoutSecs 属性值。该属性值可以通过 console 控制台来设置。

2．weblogic.xml

设置 WebLogic 特有部署描述符 weblogic.xml 的<session-descriptor>元素的 TimeoutSecs 属性。这个值以秒为单位。

```
<session-descriptor>
  <session-param>
    <param-name>TimeoutSecs</param-name>
    <param-value>3600</param-value>
  </session-param>
</session-descriptor>
```

默认值是 3 600 秒。

20.2　WebSphere

面试例题 1：Tomcat 和 WebSphere、WebLogic 有什么区别和联系？

答案：Tomcat 只是入门级别的 Web 服务器，支持 JSP 和 Servlet。

WebSphere 是 IBM 公司的专业级别的应用服务器，除了支持 JSP 和 Servlet 以外，还支持数据库连接池管理、EJB 容器、LDAP 和 TIVOLI 等，功能比 Tomcat 强大。

WebLogic 是世界上第一位的应用服务器，支持数据库连接池、EJB 容器、集群管理，完全遵循 J2EE 的规范。

面试例题 2：两台 Tomcat 或 WebLogic 服务器处于同一网段，有什么好的通信手段？每次通信内容不超过 10KB，要求不能超过毫秒级，即一定要小于 100ms。

答案：可以从以下几个步骤设置。

（1）Socket 创建后一定要设置属性，把 SendBuffer 和 RecBuffer 调整一下。buffer 小，反应速度快，但如果数据大的话就会卡住。建议设置为单个数据包的 4 倍左右。TcpNoDelay 一定要设置，设置成 true，否则这里就要延迟 100ms。

（2）传输流最好用 BufferedStream，同样在创建时设置缓存，和 socket 设置的系统缓存一样大就可以了。

（3）数据接收发送要分开线程，处理也要分开。将收到的数据放入 LinkedList 中，由另外一个线程去处理。

（4）socket 要设置超时，两个 socket 之间要轮循报告状态，用这种方法来检测流是不是断开（通过这种方式判断是不是 TCP 断开了）。

（5）一定要有重连动作，因为 socket 可能会断开。

20.3　WebService

面试例题 1：WebService 是什么？

解析：对这个问题，至少有两种答案。从表面上看，WebService 就是一个组件，它能够通过 TCP/IP 网络被其他应用程序调用。它执行一种特殊功能，包括从计算和信用卡认证到复杂的排序之间的所有操作，并且向调用应用程序返回数值。Microsoft 指出："WebService 的独特之处就是能够通过 Web 调用它们"，暴露出一个能够通过 Web 进行调用的 API。这就是说，能够用编程的方法通过 Web 来调用这个应用程序。把调用这个 WebService 的应用程序叫做客户。例如，想创建一个 WebService，它的作用是返回当前的天气情况。那么可以建立一个 ASP 页面，它接受邮政编码作为查询字符串，然后返回一个由逗号隔开的字符串，包含了当前的气温和天气。要调用这个 ASP 页面，客户端需要发送下面的这个 HTTP GET 请求。

http://host.company.com/weather.asp? zipcode=20171

返回的数据就应该是：

21，晴

这个 ASP 页面就应该算做是 WebService 了。因为它基于 HTTP GET 请求，暴露出了一个可以通过 Web 调用的 API。当然，WebService 还有更多的东西。

答案：WebService 是建立可互操作的分布式应用程序的新平台。作为一个 Windows 程序员，可能已经用 COM 或 DCOM 建立过基于组件的分布式应用程序。COM 是一种非常好的组件技术，但是也很容易举出 COM 并不能满足要求的情况。WebService 平台是一套标准，它定义了应用程序如何在 Web 上实现互操作性。可以用任何喜欢的语言，在任何喜欢的平台上写 WebService，只要能通过 WebService 标准对这些服务进行查询和访问。

面试例题 2：什么是 SOAP?

答案：简单对象访问协议（SOAP）是一种轻量的、简单的、基于 XML 的协议，它被设计成在 Web 上交换结构化的和固化的信息。SOAP 可以与现存的许多因特网协议和格式结合使用，包括超文本传输协议（HTTP）、简单邮件传输协议（SMTP）和多用途网际邮件扩充协议（MIME）。它还支持从消息系统到远程过程调用（RPC）等大量的应用程序。

SOAP 包括以下 3 个部分。

- OAP 封装：它定义了一个框架，该框架描述了消息中的内容是什么，谁应当处理它，以及它是可选的还是必需的。
- SOAP 编码规则：它定义了一种序列化的机制，用于交换应用程序所定义的数据类型的实例。
- SOAP RPC 表示：它定义了用于表示远程过程调用和应答的协定。

第21章

Java 测试

软件测试在软件质量安全控制上的地位不可替代。美国的软件企业将 40%的工作量花在软件测试上，测试费用占项目总费用的 30%～50%。如微软 Windows 2000 团队动用的测试人员比项目经理和开发人员的总和还要多。之所以如此重视软件测试，是因为通过必要的测试，软件缺陷数可至少降低 75%，而软件的投资回报率能达到 350%。

我国随着软件产业的高速发展，软件测试的重要性也已逐渐被企业关注，国内软件企业已有组建测试队伍的意识。但现阶段我国软件测试人才极为紧缺。在今后若干年，软件测试工程师将在面试过程中逐渐成为焦点。软件测试工程师面试考题非常宽泛：语言、数据库、数据结构、算法、计算机网络、测试基本理论、测试流程和测试技术等都有涉及。

21.1 白盒测试

面试例题 1：白盒测试技术中测试用例的设计，_____是覆盖最弱的一组。

A．语句覆盖　　B．路径覆盖　　C．条件组合覆盖　　D．判定覆盖

解析：本题考的是白盒测试技术中的逻辑覆盖。

一个或者多个条件组成一个判定，一个程序中可以有多个判定。首要的是建立一个二维的真值表，各列为判定和条件，各行为每组值的 T 或者 F。逻辑覆盖包含以下 6 种。

（1）语句覆盖。

为了暴露程序中的错误，至少每个语句应该执行一次。这也是最弱的逻辑覆盖标准。

（2）判定覆盖。

每个判定的每种可能结果都要执行一次。建立判定表以后，要保证每种判定的结

果中都包含了 T 和 F，才满足判定覆盖。

（3）条件覆盖。

不但每个语句需要执行一次，而且判定表达式中的每个条件都要取到可能的结果。建立判定表以后，要保证每种条件的结果中都包含了 T 和 F，才满足条件覆盖。

（4）判定/条件覆盖。

每个判定以及每个判定中的每个条件都取到可能的结果。建立判定表以后，要保证每个判定结果包含 T 和 F，而且每种条件的结果包含 T 和 F。也就是综合了上面的条件覆盖和判定覆盖。

（5）条件组合覆盖。

每个判定中的条件的各种组合至少出现一次。也就是说，先把程序中的条件列出来，排列组合写出所有的可能性，看有没有哪组值同时满足这些排列组合。

（6）路径覆盖。

每条可能的路径都至少执行一次，看源程序中的判断都有哪些组合，比如 T、FF、FT 等，看看哪个满足包含了所有的组合。这些不同的组合就代表了程序中执行的不同路径。

答案：A

面试例题 2：Function club is used to simulate guest in a club. With 0 guests initially and 50 as max occupancy, when guests beyond limitation, they need to wait outside;when some guests leave the waiting list will decrease. The function will print out number of guests in the club and waiting outside. The function declaration as follows:

void club(int x); positive x stands for guests arrived, nagative x stands for guests left from within the club.（club 函数用来模拟一个俱乐部的顾客。初始化情况下是 0 个顾客，俱乐部最大规模只能有 50 个顾客，当用户超过了最大规模，他们必须等在外面。当一些顾客离开了等待队列将减少。这个 club 函数将打印在俱乐部里面的顾客人数和外面的等待人数。函数声明如下：

```
void club(int x);
```

正数 x 代表客人来了，负数 x 代表客人离开了俱乐部）

For example, club (40) prints 40,0; and then club (20) prints 50,10; and then club (-5) prints 50,5; and then club (-30) prints 25,0; and then club (-30) prints N/A; since it is impossible input.（举例而言，club (40)打印 40，0；接着 club (20)打印 50，10；接着 club (-5)打印 50，5；接着 club (-30)打印 25，0；接着 club (-30)打印 N/A，因为这是不可能

实现的。）

To make sure this function works as defined, we have following set of data to pass into the function and check the result are correct.（为了确保函数工作正常，我们使用下列数据来测试函数是否正常，你认为该选哪个选项？）

```
a 60
b 20 50 -10
c 40 -30
d 60 -5 -10 -10 10
e 10 -20
f 30 10 10 10 -60
g 10 10 10
h 10 -10 10

A. a d e g
B. c d f g
C. a c d h
D. b d g h
E. c d e f
```

解析：本题实际上是考边界条件的测试情况，看有没有覆盖所有的边界条件。设 A 为已在俱乐部的成员，B 为排队的人。

```
        A                B
Case1   <=0              0
Case2   <50(Up/Down)     0
Case3   50               >0(Up/Down)
```

对于 a～h 的各种情况：

```
a 60                适用 C3
b 20 50 -10          适用 C2\C3
c 40 -30            适用 C2
d 60 -5 -10 -10 10     适用 C3\C2
e 10 -20            适用 C1
f 30 10 10 10 -60      适用 C1\C2\C3
g 10 10 10          适用 C2
h 10 -10 10         适用 C2
```

看看条件，肯定要包含 e，因为只有这个 case 能测 N/A 的情况，排除了 B、C、D 三项；再看选项 A 和选项 E，差别在 a 和 c、g 和 f 的选取上，很显然，d 包含 a，f 包含 g，所以排除选项 A，最终确定选项 E。

答案：E

面试例题 3：Why is test automation important?（自动化测试为何重要？）

解析：自动化测试从根本上提高了 QA 们的职业素质，让 QA 们彻底摆脱了重复繁重的测试工作，而更着重于 QA 的流程中已经完成项目质量的重复保证上。此外，某些人对 QA 有些偏见：对 QA 的普遍认识就是只是测试而已。因为他们能看到 QA 最直接的劳动就是在反反复复、勤勤恳恳地测试。

答案：自动化测试可以让测试人员从枯燥无味的手工重复性测试中解放出来，并且提高工作效率，通过自动化测试结果来分析功能和性能上的缺陷。

面试例题 4：Describe criterions that testing is completed。（描述一个测试结束的准则。）

答案：一个测试结束的标准是查看已提交的 BUG 是否已经全部解决并已验证关闭。一般来说，BUG 验证率在 95%以上，并且没有大的影响功能的 BUG 处于未解决状态，就可以测试通过。

面试例题 5：What kinds of content should be included in a Test Plan?(For example, available human resource)（在一个测试计划中能包含哪些内容？如可用的人力资源）

解析：在一个测试计划中可以包含需要测试的产品的特点和主要功能模块，列出需要测试的功能点，并标明侧重点；测试的策略和记录（测试工具的确认、测试用例等文档模板、测试方法的确定）；测试资源配置（确定测试每一阶段的任务和所需资源）。

面试例题 6：Please describe differences between functional testing and usability testing（请描述功能测试和可用性测试之间的区别。）

解析：本题涉及几个测试的重要概念：

Functional testing （功能测试）也称为 behavioral testing（行为测试），根据产品特征、操作描述和用户方案，测试一个产品的特性和可操作行为以确定它们满足设计需求。本地化软件的功能测试用于验证应用程序或网站对目标用户能否正确地工作。使用适当的平台、浏览器和测试脚本，以保证目标用户的体验将足够好，就像应用程序是专门为该市场开发的一样。

功能测试也叫黑盒子测试或数据驱动测试，只需考虑各个功能，不需要考虑整个软件的内部结构及代码。一般从软件产品的界面、架构出发，按照需求编写出来的测试用例，输入数据在预期结果和实际结果之间进行评测，进而提出使产品更加符合用户使用的要求。

可用性测试是用户在和系统（网站、软件应用程序、移动技术或任何用户操作的设备）交互时对用户体验质量的度量。可用性（Usability）是交互式 IT 产品/系统的重

要质量指标，指的是产品对用户来说有效、易学、高效、好记、少错和令人满意的程度，即用户能否用产品完成他的任务，效率如何，主观感受怎样，实际上是从用户角度所看到的产品质量，是产品竞争力的核心。为了获得可用性，可以参考以下 4 个原则。

① 及早以用户为中心：设计人员应当在设计过程的早期就致力于了解用户的需要。

② 综合设计：设计的所有方面应当齐头并进地发展，而不是顺次发展。使产品的内部设计与用户界面的需要始终保持一致。

③ 及早并持续性地进行测试：当前对软件测试的唯一可行的方法是根据经验总结出的方法，即若实际用户认为设计是可行的，它就是可行的。通过在开发的全过程引入可用性测试，可以使用户有机会在产品推出之前就设计提供反馈意见。

④ 反复式设计：大问题往往会掩盖小问题的存在。设计人员和开发人员应当在整个测试过程中反复对设计进行修改。

答案：功能测试主要是黑盒测试，由测试人员进行，主要验证产品是否符合需求设计的要求。

可用性测试主要是由用户（或者测试人员模拟用户行为）来进行的测试，主要是对产品的易用性进行测试，包括有效性（effectiveness）、效率（efficiency）和用户主观满意度（satisfaction）。其中有效性指用户完成特定任务和达到特定目标时所具有的正确和完整程度；效率指用户完成任务的正确和完整程度与所使用资源（如时间）之间的比率；满意度指用户在使用产品的过程中所感受到的主观满意和接受程度。

面试例题 7：The following is a pseudocode fragment that has no redeeming feature except for the purpose of this question.（下面语句是一段伪代码）

```
module nonsense[]
/*a[] and b[] are global variables */
begin
int i,x
i=1
read(x)
while(i <x)do begin
a[i]=b[i]*x
if a[i]>50 then
print("array a is over the limit")
else
print("ok")
i=i+1
end
print("end of nonsense")
end
```

Please list test cases to cover all branches in this fragment（请设计一个测试用例涵盖该段代码各个分支。）

答案：白盒测试有几种测试方法：条件覆盖、路径覆盖、语句覆盖、分支覆盖。其中分支覆盖又称判定覆盖，使得程序中每个判断的取真分支和取假分支至少经历一次，即判断的真假均曾被满足。本题的目的逻辑分支图如图 21-1 所示。

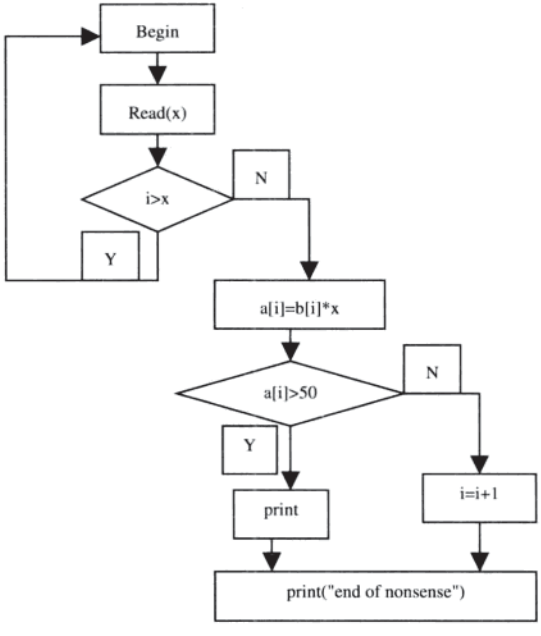


图 21-1　目的逻辑分支图

根据判定覆盖准则，测试用例涵盖该段代码各个分支如下：

① i>x
② i<x && a[i]>50
③ i<x && a[i]<50
④ i=x && a[i]>50
⑤ i=x && a[i]<50

21.2　性能测试

面试例题 1: A student needs to score at least 60 points to pass. If they score at least 80 points they will achieve a Merit and if they score 100 points they will achieve the Maximum.

Which of the following would be the most likely set of values identified by the boundary Value Testing?_____

(一个学生需要 60 分才能及格；80 分就可以得优；100 分就是满分了。下面四个选项中哪一个边界测试是最好的，为什么？)

A．-1,0,59,60,79,80,99,100　　B．0,59,79,100

C．0,1,59,69,70,80,100　　D．60,80,100

解析：边界值测试就是找到边界，然后在边界及其边界附近（这里应该包括边界两侧）选点。因此，边界 0（隐含需求边界），60，80，100 要测试，边界另一侧的-1，59，79，99 也要测试。对于选项 B、D，只覆盖了边界的一侧，而选项 C 中的 69 和 70 跟边界无关，所以选项 A 相对最好。

答案：A

扩展知识：除边界值测试外，面试前最好了解以下几个相关概念。

（1）健壮性测试：健壮性测试是边界值分析的一种简单扩展。除了变量的五个边界值分析之外，还要分析变量值比最高值高出一点和比最低值低一点的情况下会出现什么反应。

（2）最坏情况测试：边界值分析时是在单缺陷的假设下进行的。如果不做此假设，那么就会出现同时有多个变量取边界值的情况。最坏情况测试的测试用例的获取是对每个变量先进行包含五个边界值元素集合的测试，然后对这些集合进行笛卡儿积计算，以生成测试用例。

（3）特殊值测试：这种测试不需要使用任何测试方针，只使用最佳工程判断。因此，该方法与测试人员的能力密切相关。

（4）随机测试：这种方法不是永远选取有界变量的最小值、略高于最小值、正常值、略低于最大值、最大值，而是使用随机数生成器生成测试用例值。这种测试用例的获取需要用程序来得出，而且还涉及测试覆盖率的问题。

面试例题 2：int FindMiddle(int a ,int b,int　c)和 int CalMiddle(int a ,int b,int　c)分别为两个 C 函数，它们都号称能返回三个输入 int 中间的那个 int。你无法看到它们的源代码，你如何判断哪个函数的质量好？

答案：从编程习惯上看，笔者认为 int FindMiddle(int a ,int b,int　c)比较好，因为名字比较明确，就是找中间那一个数，让人一看就明白。

这道题是考软件测试的分析能力，比如一些特殊情况要特殊处理，例如：先测试

000，看它们的测试结果，再测 001，随便输入一些不是数字的数，测一下它们的排错功能，如果它们的结果一样，那就该测它们的算法效率。比如可以计算 10000 个数测试用时，代码如下：

```
System.out.println(new Date().toString());
for(int i=0; i < 10000; i++){
  for(int j=0; j < 10000; j++){
    for(int k=0; k < 10000; k++){
      FindMiddle(i, j, k);
    }
  }
}
System.out.println(new Date().toString())

System.out.println(new Date().toString());
for(int i=0; i < 10000; i++){
  for(int j=0; j < 10000; j++){
    for(int k=0; k < 10000; k++){
      CalMiddle(i, j, k);
    }
  }
}
System.out.println(new Date().toString())
```

面试例题 3：please write a test plan and test case the elevator functionalities .（请写一个电梯功能的测试用例和测试方案。）

解析：电梯调度算法的基本原则就是如果在电梯运行方向上有人要使用电梯，则继续往那个方向运动，如果电梯中的人还没有到达目的地，则继续向原方向运动。在此将电梯一次从下到上视为一次运行（注意不一定从底层到顶层），同理，电梯一次从上到下也视为一次运行（注意不一定从顶层到底层）。

当电梯向上运行时：

① 位于当前层以下的向上请求都被忽略留到下次向上运行时处理；
② 位于当前层以上的向上请求都被记录留到此次运行处理；
③ 无论何层的向下请求都被忽略留到下次向下运行时处理；

当电梯向下运行时：

① 位于当前层以上的向下请求都被忽略留到下次向下运行时处理；
② 位于当前层以下的向下请求都被记录留到此次运行处理；
③ 无论何层的向上请求都被忽略留到下次向上运行时处理；

答案：如果只考虑单部电梯的实现情况，是很简单的。电梯移动引发状态转换，电梯的主要状态有：

① UpRun，电梯上行中。
② UpStopClose，电梯上行楼层暂停未开门。
③ UpStopOpen，电梯上行楼层暂停已开门。
④ UpRun，电梯下行中
⑤ UpStopClose，电梯下行楼层暂停未开门。
⑥ UpStopOpen，电梯下行楼层暂停已开门。
⑦ FullStopClose，电梯楼层停止未开门。
⑧ FullStopOpen，电梯楼层停止未开门。

根据用户的请求会转换电梯所在位置的状态，请求分为：

① 电梯内同向请求。
② 前方楼层同向请求。
③ 前方楼层逆向请求。
④ 电梯内逆向请求。
⑤ 后方楼层逆向请求。
⑥ 后方楼层同向请求。
⑦ 电梯内关门请求。
⑧ 电梯内开门请求。
⑨ 停止时同层同向请求。
⑩ 停止时同层逆向请求。

多部电梯的情况相对复杂，每部电梯是一个电梯类的实例，在电梯运行到楼层处时检查请求并进行状态转换，但这样的效果：

① 电梯只考虑局部状态没有全局观（只看一步），像苍蝇。
② 时间一长电梯扎堆（抢任务），像蜜蜂。
③ 电梯经常空跑（任务被其他电梯完成），像蚂蚁。

对多部电梯情况进行适当优化：

① 增加时间变量
② 增加相应配合(只派一部电梯)
③ 电梯空闲时到重要地点（底层或顶层）。

效果：

① 电梯行为过分依赖于电梯内请求。
② 电梯不适应突发性集中请求（层层停没效果）。
③ 电梯上行的行为与下行不同（底层到各层、各层到底层、停层不是聚集点）。
④ 公平服务（先来先向服务）。
⑤ 反向快速响应（提前转向）。

⑥ 最长等待者的最短等待（后来者的跳跃型响应）。
⑦ 最少移动（资源最小调度）。

更高级的算法是：

① 梯内人优先（判断梯内直达请求）。
② 梯内人优先（减少无效开门，考虑超载直达）。
③ 不响应下行聚集型请求（减少无效开门，考虑超载直达）。

甚至测试可以考虑电梯环境：

① 公寓型：只有底层（停车场）到各层或各层到底层，爆发型请求不多。
② 写字楼型：有若干次爆发性请求（上下班或午休情况）需要测试，有邻层转移要求（同一公司租用）。
③ 酒店型：某些层（大堂、餐厅）有阶段持续请求。
④ 商场型：一般来说均匀调度。

面试例题 4：please design a test plan and test case to test a simplied mobile phone: (写一个测试用例和测试方案来测试手机。)

```
including following function（包括如下功能）
Calling
SMS
Address book
```

答案：

对于 Calling：
① 是否有拨打电话这个功能，能否接通电话。
② 拨打正常号码。
③ 拨打不正常号码，是否有相应提示。

对于 SMS：
① 是否有发送或接收短信功能。
② 输入正常号码发送短信。
③ 输入位数不对等不正常号码是否有提示。
④ 发送短信能否保存在发件箱，能否保存草稿，短信能否删除等。

对于 Address book：
① 是否有电话本这个功能。
② 新建一个联系人，联系人信息为空是否有提示。
③ 新建联系人与已有联系人姓名或电话号码等信息重复是否有提示。
④ 删除联系人是否成功，删除时是否有提示信息。

21.3　游戏

面试例题 1：在用户体验测试中，发现问题可能存在误测，误测包括两种：第 1，问题被遗漏了；第 2，不是问题却被认为是问题。我们的测试通常通过两种不同的测试方法来进行，依据完成不同的设计思路，但是有一些共同的：第 1，他们都能发现所有存在的问题；第 2，仍然会有百分之三的误测率；第 3，不会出现同一问题对两个系统都误测的可能性。为了测试一个非常重要的系统，我们会把两个测试方法进行交叉测试，测试结果采取并集，我们可以这样推理：采用这种方法的时候是不会出现任何误测的情况。

以下哪项最恰当地描述了上述推理？

A．上述推理是必然的，即如果前提为真，则结论一定真

B．上述推理很强，但不是必然，即如果前提条件为真，则为结论提供很强的证明，但附加信息仍可削弱论证

C．上述推理很弱，前提条件尽管与结论相关，但最多只为结论提供不充分证明

D．推论不成立，因为它把某些必要的条件当做了充分的条件

解析：因为从前提条件中只知道有两种误测，但并未说明两种误测能够在测试时就知道，因此，当误测发生时，我们无法区分是哪种误测，因此，即使使用两种测试方法进行交叉测试取并集，最多只能保证不被遗漏，但是不能解决不是问题却被认为是问题。

答案：D

面试例题 2：假定现在有一个轮船航海游戏，参与人数为 2～4 人，游戏的胜利方式是自己的四艘小轮船都航到终点，获胜的人将可以离开游戏。为了增强趣味性，投放了一种收费道具：1～4 遥控骰子，该道具需要在 1 盘游戏开始前决定是否接受，无论使用与否，都将在游戏后消失，一次游戏最多带两个骰子，一个可以用三次。道具投放之后，有玩家投诉该游戏道具严重破坏游戏的平衡性。由此，开发组获取某一周内所有的数据，每场游戏数据包含：游戏场次编号，游戏进行时间，游戏参与人数，游戏花费时间，第一胜利者用户 ID，该用户使用道具情况，第二胜利者用户 ID，该用户使用道具情况，第三胜利者用户 ID，该用户使用道具情况，第四胜利者用户 ID，该用户使用道具情况。

问题：请您给出一个确认游戏平衡是否失控的分析过程（包含数据信息的分析）。

答案：判定游戏是否失控的分析可以有多种：

① 由于均衡性的质疑源于道具使用，所以将收集到的数据分为两种，一种是游戏中都没有使用道具和都使用道具的，另外一种是游戏中既有使用道具的，也有没有使用道具的。我们在此只分析第②种；

② 将上述第②种情况中所有使用道具的游戏得分求平均值和所有没有使用道具的游戏得分平均值进行比较，查看差距范围，以评价平衡性（均值分析）。

③ 统计上述第②种情况中各局游戏的第一名使用道具和不使用道具的百分比值，根据比值评价平衡性（极值分析）。

面试例题 3：请详细叙述俄罗斯方块中包含的人生态度。

答案 1：位置感是成功的关键，有些废物因为位置正常变成精英，有些精英因为位置错位变成废物。

答案 2：寸有所长，尺有所短。精确定位，韬光养晦。一次失意说明不了什么。

答案 3：俄罗斯方块显示了人生是由一点一滴积累出来的。正所谓“冰冻三尺非一日之寒”，所以我们做什么都要持之以恒，坚定信心；坚持下去，才可以取得最终的胜利。任何事情如果不坚持的话，到最终也只是镜花水月而已。

答案 4：追求平整和平衡，人生才会放下负担。

面试例题 4：什么是游戏的不平衡性？

解析：Sid Meier 曾经说过：“一个游戏是很多有趣的选择的集合。”因此，得出的是如果游戏失去平衡，就会减少这些选择而影响游戏性。一个理想的游戏应该经过一系列的选择，最后以胜利或其他完成的条件结束。有时一些选择明显成为唯一的选择，或明显是无效的。如果在某一阶段，游戏出现仅有唯一的选择，而游戏却没有结束，就说明游戏的平衡性有了问题。

游戏大富翁(Monopoly)中就有很好的游戏不平衡性的例子。在游戏的后期，玩家们总是尽量拖长呆在监狱里的时间。显然，玩家在游戏后期的最好策略就是进监狱不出来而且也不付钱，希望别人进入自己的领土而破产。在玩大富翁的最后阶段，无须再做选择，游戏基本结束了。没有人再选择是否购买财产，也很少有机会再根据游戏规则建设新的财产（因为房子已经被用完），而且因为资产已经被几个人集中，所以也不再有交易可做。一旦产生这种情形，游戏就变成每个玩家有一定的概率获胜而基本上结束了。此时玩家可以做得很少，除非靠运气得胜。这情景与游戏前期及中期大相径庭，那时玩家往往忙于大施战术、巧妙夺取利益、陷害对手或谨慎购买“重量级”黄色、绿色或深蓝色的地产。

答案：游戏的不平衡来自于玩家在游戏中选择权的减少。例如，在一个策略游戏里，如果某一种部队的作用和费用相比过于划算，就会造成其他部队几乎或完全没有作用。这种情况不仅只留给玩家一个选择（无从选择），而且使玩家受到很多不相关的干扰。这些干扰实际上让游戏变得比较迷乱，减损了游戏性，而且让玩家感到灰心。

所有游戏的不平衡性最终归结为游戏没有选择性。

第 6 部分

综合面试题

Compositive interview questions

本部分主要介绍求职面试过程中出现的第五个重要的板块——英语面试、电话面试和智力测试。其中，英语面试不同于普通的英语面试。就一个程序员而言，最好能够用英文流利地介绍自己的求职经历，这是进外企非常重要的一步。此外，还必须对几个常用的问题有相关的解答，比如你最大的缺点是什么。有些问题即便是用中文，你都很难回答，更何况是用英文去回答。但是求职过程本身就是一个准备的过程，精心地准备，等待机会——机会总是垂青于那些精心准备的人。

第22章

英语面试

如果你是一个具有战略眼光，且期待进入国际性跨国大企业的求职者，本章值得你仔细研读。

英语面试主要考查两个部分：英语口语能力和你做人的特质。外企会有专门的人力资源经理和你聊天，会问你关于人生、经历、团队合作、成功收获、失败教训等一系列问题。这里和一般的技术类面试不同，不是考验你的技术能力，而是考验你的语言能力及情商，所以事先要预测一下面试官可能提出的问题。本章取材于实际英语面试中关于工作问题、个人特质、未来企划等知识点，并给出了详细的参考答案。请读者结合个人经历修改这些答案以应对可能出现的英语面试。

22.1 面试过程和技巧

现在，不管是国企还是外企，在招聘时都非常看重应聘者的英语交际能力，公司往往通过英语面试，对应聘者的英语交际能力进行考查。我们对参加英语面试的应聘者提出4个建议。

建议一：精心设计一个自然的开场白。例如：

C（应聘者）：May I come in？（我能进来吗？）

I（考官）：Yes, please. Oh, you are Jin Li, aren't you？（请进。哦，你是李劲吧？）

C：Yes, I am.（对，我是。）

I：Please sit here, on the sofa.（请坐在沙发上。）

C：Thank you.（谢谢。）

采用“Excuse me. Is this personnel department？”（请问，这里是人力资源部吗？）或“Excuse me for interrupting you. I'm here for an interview as requested”（不好意思打搅了，

我是应约来应聘的。）类似的开场白都比较合适。在确定了面试场所和面试官之后，简洁地用“Good morning / afternoon”向面试官打招呼，也能令在场的人提升对你的印象。

建议二：不要害怕外表冷冰冰的考官。

一些用人单位与面试者的最初交流是比较冷冰冰的。例如：

I：Your number and name, please.（请告知你的号码和名字。）

C：My number is sixteen and my name is Zhixin Zhang.（我是 16 号，我叫张志新。）

这时，面试者不要觉得有压力，面试官的态度冷漠并不是针对你个人，你只需要照实简洁地回答即可。在某些情况下，面试官会问一些看起来比较普通和随意的问题，但实际上是暗藏深意的。例如：

How did you come? A very heavy traffic?（你怎么来的？路上很堵吧？）

你的回答可以是这样：“Yes, it was heavy but since I came here yesterday as a rehearsal, I figured out a direct bus line from my school to your company, and of course, I left my school very early so it doesn't matter to me.”（是的，很堵。但我昨天已经预先来过一次并且找到了一条从我们学校直达贵公司的公交路线，并且在今天提早从学校出发。因此，路上的拥堵并没有影响到我。）这样的回答就点出了你的计划性及细心程度。

建议三：抓住考官问题的关键点回答。

如果应聘者突然听不懂面试官的话，或者问题太复杂，该如何回答？下面是一名学生参加一知名化妆品公司英文面试中的一段。

I：You are talking shop. How will you carry out marketing campaign if we hire you?（你很内行。如果我们雇用你的话，你打算如何开始你的市场工作？）

C：Sorry,sir. I beg your pardon.（对不起，能重复一下吗？）

I：I mean that how will you design your work if we hire you?（我的意思是如果我们决定录用你，你打算如何开展市场工作？）

C：Thank you, sir. I'll organize trade fair and symposium, prepare all marketing materials, and arrange appointments for our company with its business partners（谢谢。我将组织贸易展销会和研讨会，预备所有的营销材料，为公司及其商业伙伴安排见面会。）

I：Do you know anything about this company?（你对本公司的情况了解吗？）

C：Yes, a little. As you mentioned just now, yours is an America-invested company. As far as I know, ××Company is a world-famous company which produces cosmetics and skincare products. Your cosmetics and skincare products are very popular with women in all parts of the world.（是的，了解一点点。正如你刚才所提到的那样，贵公司是一家美资公司。据我所知，贵公司是一家世界闻名的生产美容护肤品的公司。你们的美容护肤品深受世界各地妇女的欢迎。）

在没听清对方问题的情况下，可以要求对方重复，除了“I beg your pardon.”还可

以用“Would you please rephrase your sentence？”的表达方式。

建议四：巧用过渡语，表明自己用心听问题。

面试者在面试时可以用一些类似“As you mentioned”（正如你所说的）或者“As far as I know”（据我所知）的句子，表示你一直在认真听对方的谈话。此外，你还可以选择“As it is shown in my resume”（正如我的简历所提到的）或“As my previous experience shows”（如我之前的工作经验所示）之类的表达法。

另外，在面试之前，面试者对应聘公司应有所了解，比如公司的规模、业务、未来发展等。这些往往被学生忽略了。对公司文化理解深刻，是你超出其他应聘者的一个亮点。但如果实在不了解，就应根据所知诚实回答。

22.2 关于工作（About Job）

面试例题 1：Can you sell yourself in two minutes? Go for it.（你能在两分钟内自我推荐吗？大胆试试吧！）

A：With my qualifications and experience, I feel I am hardworking, responsible and diligent in any project I undertake. Your organization could benefit from my analytical and interpersonal skills.（依我的资格和经验，我觉得我对所从事的每一个项目都很努力、负责、勤勉。我的分析能力和与人相处的技巧，对贵单位必有价值。）

面试例题 2：Tell us about your project experiences.（告诉我你的项目经验是什么。）

A：Northwest University Personnel Managing System is a system of auto-manage the persons' information. It is designed by PowerDesign. The project is a C/S architecture system. It is based on a Microsoft SQL database, and the UI is developed by Delphi 7. In this project, I designed the schema of database, programmed database connectivity using Delphi 7 and ADO.

（西北大学人事管理系统是一个自动的人事信息管理系统，它是用 PowerDesign 设计的。整个项目是 C/S 系统架构。它的后台基于微软的 SQL 数据库，前台设计由 Borland 公司的软件 Delphi 7 完成。在这个项目中，我负责系统的架构及数据库的连接。）

A：Based on ASP.NET+SQL2000, we finished Northwest University Network Course-selected System. Everybody in this school can select, cancel, query course in network. The project is a B/S architecture system; the code is developed by Visual C#, and run on the. NET plat. In this project, I used the ADO interface which is provided by the Database program and after that, I joined the testing of whole system.

（基于 ASP.NET+SQL 2000 的平台，我们实现了西北大学网络选课系统。学校内的任何人都能够在网上选择、取消、查询课程。整个项目是 B/S 的系统架构。项目基于.NET

平台，前端代码是用 C#完成的。在项目中，我们使用 ADO 接口实现数据库的支持。整个系统架设结束后，我参加了系统的测试工作。）

面试例题 3：Give me a summary of your current job description.（对你目前的工作做个概括的说明。）

A：I have been working as a computer programmer for five years. To be specific, I do system analysis, trouble shooting and provide software support.（我干了 5 年的电脑程序员。具体地说，我做系统分析、解决问题及提供软件方面的支持。）

面试例题 4：Why did you leave your last job?（你为什么离职呢？）

A：Well, I am hoping to get an offer of a better position. If opportunity knocks, I will take it.（我希望能获得一份更好的工作。如果机会来临，我会抓住。）

A：I feel I have reached the "glass ceiling" in my current job. I feel there is no opportunity for advancement.（我觉得目前的工作已经达到顶峰，即没有升迁的机会。）

面试例题 5：How do you rate yourself as a professional?（作为一位专业人员，你如何评估自己呢？）

A：With my strong academic background, I am capable and competent.（凭借我良好的学术背景，我可以胜任自己的工作，而且我认为自己很有竞争力。）

A：With my teaching experience. I am confident that I can relate to students very well.（依我的教学经验，我相信能与学生相处得很好。）

A：My background has been focused on preparing me for the IT field, so I can exhibit my ability right away. I already have obtained the educational technology and skills. I am confident of my ability to learn quickly in any assignment which I'm not familiar for the moment.（我的背景使我非常适合 IT 领域，我会在此展现我的能力。我已经获得了教育方面的能力和技巧，我确信我的能力能够迅速地学习那些我暂时不了解的任务。）

A：I realize that there are many other college students who have the ability to do this job. I also have that ability. But I also bring an additional quality that makes me the very best person for the job——my attitude for excellence. I am a multi-tasked individual who work well under pressure.　My ability to be trained in any area would definitely be a good reason to hire me to work for this firm.（我知道有很多大学生有能力去做这个工作。我也有这个能力。但是相对于这些人而言，我的积极态度决定了我是做这项工作的最佳人选。我是一个可以承担多种任务压力的人。我可以胜任公司的各个领域，并为公司努力工作。）

面试例题 6：What contribution did you make to your current (previous) organization?（你对目前/从前的工作单位有何贡献？）

A：I have finished three new projects, and I am sure I can apply my experience to this position.（我已经完成了 3 个新项目，我相信我能将我的经验用在这份工作上。）

面试例题 7：What do you think you are worth to us?（你如何知道你对我们有价值呢？）

A：I feel I can make some positive contributions to your company in the future.（我觉得未来我对贵公司能做些积极的贡献。）

面试例题 8：What make you think you would be a success in this position?（你如何知道你能胜任这份工作？）

A：My graduate school training combined with my internship should qualify me for this particular job. I am sure I will be successful.（我在研究所的训练，加上实习工作，使我适合这份工作。我相信我能成功。）

面试例题 9：Are you a multi-tasked individual?（你是一位可以同时承担数项工作的人吗？） Do you work well under stress or pressure?（你能承受工作上的压力吗？）

A：Yes, I think so. The trait is needed in my current(or previous) position and I know I can handle it well.（这种特点就是我目前（先前）工作所需要的，我知道我能应付自如。）

面试例题 10：What will it take to attain your goals, and what steps have you taken toward attaining them?（你将通过什么手段达到你的成功？你已经采取了哪些步骤？）

A：I have finished writing a book recently. And currently I am learning a lot of network certification. I obtained my first certification through self-study, so I am learning and keeping up with the latest technology trends. To be successful in this career, one should have to be a excellent problem solver,critical thinker, and team oriented.（我刚刚写完一本书。目前我正在做一个网络认证，是完全通过自学完成的第一个认证，所以我一直在学习，与最新的技术同步。为了在职场取得成功，我们必须成为一个优秀的问题解决专家，一个具有批判性的思考者，并以团队利益为主导。）

面试例题 11：What steps do you follow to study a problem before making a decision?（在对一项问题进行研究并给出答案之前，你会遵循什么样的步骤？）

A：Following standard models for problem-solving and decision-making can be very helpful. Here are the steps and how I solve a problem with a group project:

（1）Define the problem to be solved and decision to be made.

（2）Have a plan. To solve a problem,you must establish a plan of attack which leads to a specific goal.

（3）Solve the problem.

（遵循标准的解决问题和做决策的模式会有很大帮助。以下就是我解决问题的步骤：

（1）给要解决的问题和要做的决定做一个限定。

（2）为要解决的问题拟一个计划，为具体的目标制订一个进攻计划。

（3）解决问题。

22.3　关于个人（About Person）

面试例题 1：What is your strongest trait(s)?　（你个性上最大的特点是什么？）

A：Helpfulness and caring.（乐于助人和关心他人。）

A：Adaptability and sense of humor.（适应能力和幽默感。）

A：Cheerfulness and friendliness.（乐观和友爱。）

面试例题 2：How would your friends or colleagues describe you?　（你的朋友或同事怎样形容你？）

A：（Pause a few seconds）（稍等几秒钟再答，表示慎重考虑。）

They say Mr. Chen is an honest, hardworking and responsible man who deeply cares for his family and friends.（他们说陈先生是位诚实、工作努力、负责任的人，他对家庭和朋友都很关心。）

A：They say Mr. Chen is a friendly, sensitive, caring and determined person.（他们说陈先生是位很友好、敏感、关心他人和有决心的人。）

A：They say I am an active, innovative man, a good team-worker, with rich IT knowledge and developing experience.（他们说我是一个积极、革新的人，是一个很好的同事，并且具备丰富的 IT 知识和研发经验。）

面试例题 3：What personality traits do you admire?　（你欣赏哪种性格的人？）

A：(I admire a person who is) honest, flexible and easy-going.（诚实、不死板而且容易相处的人。）

A：(I like) people who possess the "can do" spirit.（有“实际行动”的人。）

面试例题 4：What leadership qualities did you develop as an administrative personnel?（作为行政人员，你有什么样的领导才能？）

A：I feel that learning how to motivate people and to work together as a team will be the major goal of my leadership.（我觉得学习如何把人们的积极性调动起来，以及如何配合协同的团队精神，是我行政工作的主要目标。）

A：I have refined my management style by using an open-door policy.（我以开放式的政策改进我的行政管理方式。）

面试例题 5：How do you normally handle criticism?（你通常如何处理别人的批评？）

A：Silence is golden. Just don't say anything; otherwise the situation could become worse. I do, however, accept constructive criticism.（沉默是金。不必说什么，否则情况更糟。不过我会接受建设性的批评。）

A：When we cool off, we will discuss it later.（我会等大家冷静下来再讨论。）

面试例题 6：What do you find frustrating in a work situation?（在工作中，什么事令你不高兴？）

A：Sometimes, the narrow-minded people make me frustrated.（胸襟狭窄的人有时使我泄气。）

A：Minds that are not receptive to new ideas.（不能接受新思想的那些人。）

面试例题 7：How do you handle your conflict with your colleagues in your work?（你如何处理与同事在工作中的意见不和？）

A：I will try to present my ideas in a more clear and civilized manner in order to get my points across.（我要以更清楚和文明的方式提出我的看法，使对方了解我的观点。）

面试例题 8：How do you handle your failure?（你怎样对待自己的失败？）

A：None of us was born "perfect". I am sure I will be given a second chance to correct my mistake.（我们大家生来都不是十全十美的，我相信我有机会改正我的错误。）

面试例题 9：Are you more energized by working with data or by collaborating with other individuals?（你能使组里气氛活跃，并且易于沟通吗？）

A：The best thing about working in a group or a team is combining the great minds from different facets. Compared with when you're working alone, communication can generate vitality in the project you're working on. No matter how much wisdom you've got together, without communication, you can't go very far. The perfect situation would be a

combination of communication and people, and I'm confident of my abilities in both areas.（在一个团队或一个组里工作，你最重要的一件事就是集思广益，云集大家的智慧，而不要只是一个人闷头单干。与此同时，沟通是很重要的，可以产生活力。无论你汇集了多高的智慧，如果没有沟通，你将不会成功。完美的境地是把人和沟通有机地组合。我确信我能在这两方面都做得很好。）

面试例题 10：What provide you with a sense of accomplishment?（什么会让你有成就感？）

A：Doing my best job for your company.（为贵公司竭力效劳。）

A：Finishing a project to the best of my ability.（尽我所能完成一个项目。）

面试例题 11：If you had a lot of money to donate, where would you donate it to? Why?（假如你有很多钱可以捐赠，你会捐给什么单位？为什么？）

A：I would donate it to the medical research because I want to do something to help others.（我会捐给医药研究方面，因为我要为他人做点事。）

A：I prefer to donate it to educational institutions.（我乐意捐给教育机构。）

22.4　关于未来（About Future）

面试例题 1：What is most important in your life right now?（眼下你生活中最重要的是什么？）

A：To get a job in my field is most important to me.（对我来说，能在这个领域找到工作是最重要的。）

A：To secure employment hopefully with your company.（能在贵公司任职对我来说最重要。）

面试例题 2：What current issues concern you the most?（目前什么事是你最关心的？）

A：The general state of our economy and the impact of China' entry to WTO on our industry.（目前中国经济的总体情况及中国加入 WTO 对我们行业的影响。）

面试例题 3：How long would you like to stay with this company?（你会在本公司服务多久呢？）

A：I will stay as long as I can continue to learn and to grow in my field.（只要我能在我的行业里继续学习和成长，我就会留在这里。）

面试例题 4：Could you project what you would like to be doing five years from now？（你能预料 5 年后你会做什么吗？）

A：As I have some administrative experience in my last job, I may use my organizational and planning skills in the future.（我在上一个工作中积累了一些行政经验，我将来也许要运用我组织和计划上的经验和技巧。）

A：I hope to demonstrate my ability and talents in my field adequately.（我希望能充分展示我在这个行业的能力和智慧。）

A：Perhaps, an opportunity at a management position would be exciting.（也许有机会，我将会从事管理工作。）

如果不愿正面回答，也可以说：

It would be premature for me to predict this.（现在对此问题的预测，尚嫌过早。）

面试例题 5：What range of pay-scale are you interested in？（你喜欢哪一种薪水标准？）

A：Money is important, but the responsibility that goes along with this job is what interests me the most.（薪水固然重要，但伴随工作而来的责任更吸引我。）

假如你有家眷，可以说：

To be frank and open with you, I like this job, but I have a family to support.（坦白地说，我喜欢这份工作，不过我必须要负担我的家庭。）

面试例题 6：What specific goals,including those related to your occupation, have you established for your life (career)？（你为你的职业生涯制定了什么样的具体目标？）

A：My specific goals related to my occupation is to work for a company where I can apply my technical and business skills I obtained from college and my past experience. To take advantage of the continuous learning process that goes along with the many technological advances.（关于我的职业的具体目标是：为一个可以让我施展我在大学及过往经验中积累的技术与商业技巧的公司工作。通过技术上的不断学习推进技术创新。）

面试例题 7：What attracts you to the position you have applied for? Why do you want to work for VNCC? What do you think you will be doing in your first Year at VNCC?（你应征的这个职位有什么吸引你的地方？为什么你想为 VNCC 工作？你认为在 VNCC 的第一年你将会做些什么？）

A：A company's success relies on the knowledge and skills of people. For me, the answer to find a suitable company to work for depends how much the company cares about

their own people.

VNCC Personal Development is supported throughout career, to meet the demands of the business and themselves.

VNCC assigned all staff with a personal performance counselor, which is fantastic for staff to set up their objectives and review their performance regularly. These objectives are designed to help them deliver against business requirements, whilst developing their skills. Strengths can be developed and weaknesses addressed. All these have attracted me in applying to VNCC.

I remember going to my careers centre at university and being told that VNCC was one of the top firms to work for in my interests. I knew there will be a lot of competition for a job there and the rigorous interview process certainly reflected this. It was worth it though and being part of the team here at VNCC must be something to be proud of !

The first year in VNCC should be very challenging. However, bare in mind, as a first year trainee, it is more likely to undertake a number of common tasks whilst sometimes being given tough tasks.

Although I believe I have come a long way to cope that based on my current experience, I realize that I still have tremendous amount to learn from VNCC.

VNCC definitely is the right choice and challenge that are second to none.

（一个公司的成功依靠的是员工的知识和技能。对我来说，找到一个合适的公司而为之工作的答案就是这个公司有多关心他们的员工。

VNCC 个人发展贯穿整个职业生涯，满足了业务和个人发展的需求。

VNCC 为全体员工指定了绩效顾问，有效地使员工建立起他们的目标，并能定期回顾他们的表现。这些设计出来的目标能帮助他们应对业务需求，并且提高自身技能。

优势将被挖掘，不足也将暴露出来。所有的这些都吸引我来应征 VNCC 公司。我记得去学校就业指导中心时，被告知 VNCC 是最符合我兴趣爱好的工作地。我知道，为了这里的一个职位将会有很多竞争，严谨的面试过程很好地反映了这个事实。

但是这是值得的，作为 VNCC 团队中的一员，都值得为此感到自豪。

在 VNCC 的第一年应该是充满挑战性的。无论如何，作为第一年参加工作的新人将记住，很可能承担很多琐碎的任务，并且有时候可能被给予棘手的任务。虽然我认为按照我现在的经验我有能力去完成，但是我也意识到，我在 VNCC 仍然有很多东西需要去学习。

VNCC 绝对是首屈一指的正确选择和挑战。）

第23章

电 话 面 试

求职时，经常会遭遇电话面试，戏称“触电”。笔者曾经在开会、洗澡、吃饭、坐车时都接到过电话。问的问题也是五花八门，千奇百怪。

不要轻视电话面试，打电话的人通常也是具有否决权的。俗话说“阎王好见，小鬼难缠”，这里的“阎王”是最终录用你的 HR，而“小鬼”可能是企业授权进行电话面试的外包公司。“小鬼”虽不具备最终录用你的权力，但仍然可以否决你（我就在通过笔试后，倒在 SAP 公司的电话面试上，当时电话面试我的是一家外包公司）。

电话面试前，要调整好电话，保证通话质量清晰，不要关机、欠费或超出服务区。如果是越洋电话面试（我曾经参加过 Motorola 公司和 Sybase 公司的越洋电话面试），要选择座机，以保证通话质量。电话面试通常用英文，所以平时一定要注意英语口语的练习。

23.1 电话面试之前的准备工作

一般来说，正规外企在电话面试之前会发邮件来确认你是否有空，并且确认你的电话号码是多少。通常的格式如下：

To help us schedule your interview,please respond with the following information:

When are you available to speak with us?

We generally schedule phone appointments at least three days in advance.

Please suggest several one-hour time slots when you will be available, keeping in mind that we need at least three days of lead time to schedule your interview. Please provide your time zone for the appointment as well.

What phone number should we call?

Confirm the phone number where you can be reached for the phone interview.

（为了帮助我们确认你的面试时间，请回答下面的问题：

你什么时候有时间接受我们的电话面试？

通常在 3 天内我们可以与你做一个电话的交流。

请提出几个你的空闲时间段，大概一个小时左右。我们需要 3 天的时间来为你的面试做一个时间表。

我们应拨打哪个电话号码？

确认电话号码以助于你能准确地接到面试电话。）

23.2　电话面试交流常见问题

电话面试往往是非常突然的。投递简历后就会接到这样的电话，它可能在你洗澡时、开会时、吃饭时打来。电话面试的组织方有可能是第三方中介，因此，面试问题多半是格式化的、单调的问题。电话开始时，考官会问你有没有空，如果你想准备一下，不妨告诉他没有空，让他 10 分钟后再打过来，以便做一些相应的准备。

面试例题 1：Introduce yourself, please.（介绍一下你自己。）

对于上面这个问题，你可以选择重点来回答。对于你什么时候上的小学，什么时候初中毕业就不用讲了。你可以这么说：

A：I was admitted by Northwest University with excellent student record,which was quite satisfied by the teaching staff.（我是从西北大学毕业的，我的成绩优异。来自西北大学不要说 I come from Northwest University，最好说 I was admitted by Northwest University。）

My field of study is Software and Theory, which is a famous one in China.（我的研究方向是计算机软件与理论，这个专业在中国是很著名的。你说专业的时候可以不说 major，你可以说 field of study，即研究方向。）

My undergraduate study gave me a wide range of vision. I fulfilled the courses like English, Network and Programming. I have a deep understanding in Programming.（我的本科教育给了我宽广的视野，我涉猎的课程有英语、网络和程序设计。我对编程方面有很深的见解。）

I developed several professional interests, like History and Micro Economics at my

spare time.（我有很多个人爱好，空闲的时间我研究历史和微观经济学。）

The several years working experience give me full play to my creativity, diligence and intelligence. I believe I can do my job well.（这几年的工作经历使我充满创造力，勤勉，有智慧。我相信我能把工作做好。）

面试例题 2：What is your greatest weakness?（你最大的弱点是什么？）

你不应该说你没有任何弱点，以此来回避这个问题。每个人都有弱点，最佳策略是承认你的弱点，但同时表明你在予以改进，并有克服弱点的计划。可能的话，你可说出一项可能会给公司带来好处的弱点，如可说："I'm such a perfectionist that I won't stop until a job is well done."（"我是一个完美主义者。工作做得不漂亮，我是不会撒手的。"），或者对于一个学生而言，缺乏工作经验不是很大的缺点，你可以直言不讳。

A：I'm lacking of working experience. But I'm taking a course.（我缺乏工作经验，但我正在学习。）

A：I'm lacking of supervision. But I'm reading a book.（我缺乏远见，但我会用阅读来弥补。）

A：I'm just graduated.（我只是刚毕业。）

面试例题 3：Do you know anything about this company?（你对本公司的情况了解吗？）

A：Yes,a little.As you mentioned just now，yours is an America-invested company. As far as I know, ××Company is a world-famous company which produces database and applications software products. Your products like PowerBuilder and database products are very popular with company in all parts of the world.（是的，了解一点点。正如你刚才所提到的那样，贵公司是一家美资公司。据我所知，××公司是一家世界闻名的生产数据库产品和应用软件的公司。你们的产品 PowerBuilder 和数据库产品深受世界各地公司的欢迎。）

面试例题 4：What kind of programming do you study in your project? What have you learned in this process?（在你的项目中你用到了哪种程序？在此过程中你学到了什么？）

A：Based on ASP.NET+SQL 2000 we finished Northwest University Network Course-selected System. Everybody in this school can select, cancel, query course in network. The project is a B/S architecture system; the code is developed by Visual C#, and run on the .NET plat. In this project, I used the ADO interface which is provided by the

Database program. And after that, I joined the testing of whole system.（基于 ASP.NET 和 SQL 2000 平台基础，我们完成了西北大学网络选课系统。学校中的每一个人都可以在网上选择、取消、查询课程。这个项目是一个 B/S 结构系统；代码是用 C#编写的，在.NET 平台上运行。在项目中，我使用 ADO 接口来支持数据库程序。在此之后，我参加了对整个系统的测试。）

In order to achieve the function of background database, I have designed the database including:primary key, foreign key, database connection, data view and so on. I use the SQL language to search, delete, update data. In this process I encountered a lot of difficulties. But I conquered them as best I could and finally I promoted and enriched myself.（为了实现后台数据库，我进行了数据库的设计，包括主键和外键的设计、连接、视图及其他，并运用 SQL 语句实现数据的查找、删除和修改。其中我遇到了大量的困难，但我尽我所能去克服它们，最终获得了提升和自我经验的丰富。）

面试例题 5：In this progress what are your most challenge ? How did you deal with it ？（在这个过程中你遇到的最大挑战是什么？你是怎么解决的？）

A：In the Network Course-selected System, I once met an intractable problem that the server response the client too slowly, especially when lots and lots of people upload their message at the same time. I have to find out what cause that situation. It was very urgent for me at that time, because there were about 20 thousands students in the school waiting for selecting courses, so I was undertaking great pressure. But after I checked the web log very carefully I finally found that the sticking point was on the submission button. The button has appended too many futile functions and received too much data which caused data redundance. I suggested it is challenge for me to overcome.（在网络选课系统中，我遇到了一个非常棘手的问题，即服务器响应客户端的数据非常缓慢，特别是人比较多的时候。我不得不去寻找是什么造成了这样的结果。而那时对我来说是非常紧急的，大约 2 万多名学生正等着选课呢。我承担着很大的压力，我检查了网络日志，最后发现症结出现在“提交”按钮上。一个提交按钮附加了太多的功能，读取数据量过大，导致了数据冗余。去解决这样一个问题对我来讲是一个挑战。）

A：First I try to disassembled a complicated function into several simple functions.Second, some operations are handled in database instead of web, for example set up a trigger or store procedure. We also can read web log to see how many thread success. If too many thread can not carry out, that imply reading data need to be optimized.（我的解决办法是：把功能分解，放到不同的按钮上面。我们可以设置触发器或者存储过程来解

决这些问题。通过日志，主要看在同一时刻有多少并发线程，执行成功的线程有多少。如果很大一部分并发线程没有成功，这说明数据的读取操作没有优化。）

面试例题 6：What were your main social worker or volunteers duties during the four years of college life? （大学四年期间你做过哪些社工或者志愿者活动？）

What was your greatest contribution during this period?（在此期间你最大的贡献是什么？）

What key skills have you developed?（这件事情你收获了什么技能？）

A：Coordinated with Operations Director to organise maths exhibitions for students at schools, museums and discovery centres. Instructed and guided the students at exhibitions（在学校主管的配合下，完成了学校、博物馆和科学发现中心的数学展示。）

I and two other colleagues organised a 3-days maths exhibition for around 180 pupils in Xian 74 School. The exhibition was very successful such that the school was willing to pay for much for our visit on a regular basis.（我和两个同事在西安 74 中为 180 名学生组织一次为期 3 天的数学展示，展览非常成功，学校为我们的优秀付出了高额的报酬。）

I have learnt to plan in detail and in advance. Effective communication between the school, and I would ensure the success of the event. In addition, giving presentations for pupils developed my presentation skills. （我学习了如何在细节上预先设计计划，与学校之间的有效沟通，我能确保事件的成功，此外，我给学生提供演示报告提升了我的表演才能。）

A：Internship in International Business Division of YNN bank ? Teamwork in drafting the tender for the project to develop a " Supply Chain Management (SCM)" software ? Negotiated the joint risk investment between KK Computer Co., Ltd. and Monash International Ltd. (India) ? Contact with officials from KK Computer Co Science and Technology Bureau.（在 YNN 银行国际商务部实习，协助起草了“供应链软件管理”投资计划，在 KK 电脑公司和 Monash 互联网股份有限公司之间斡旋共同风险投资谈判，与 KK 电脑公司科技办事处联系。）

During the negotiation between Monash and KK, I successfully helped KK to persuade Monash to agree to set up the major research and manufacturing base in China instead of in India. I also successfully assisted to persuade Monash to invest more capital in the joint venture.（在 Monash 和 KK 谈判期间，我成功地帮助 KK 说服了 Monash 在中国建立研发和生产中心而不是在印度，此外，我成功地说服了 Monash 在合资中增加了投资比例。）

I have learnt negotiation skills from other colleagues. Sometimes I needed to work on weekends and had meeting till midnight, but it was understandable as commitment is essential to the business.（我从其他同事那里学来了谈判技巧，有时候我需要周末加班甚至开会到深夜，但是出于对工作的忠诚，我认为这都是可以接受的。）

A： Customer service representative and the store key holder. Ensured financial performance of the store, accuracy of the cash flow ? Resolved customer queries and handled problems under emergency（我在一家商店做客户服务代表并掌管钥匙，确保商店正常的资金链和现金流的稳定，处理客户要求并在压力下处理问题。）

The store system once didn't automatically adjust goods price when it is postproduction . It resulted in wrong prices charged to customers. I immediately reported this to the area manager, and meanwhile handwrote down all transactions. It prevented the complaints from customers and also a loss to the store.（这个商店的系统某一次不能自动调整处理商品的价格，这就导致以错误的价钱卖给客户商品。发现问题后，我立即报告给区域经理，同时手动处理所有的交易。这样及时阻止了客户的抱怨和商店的损失。）

Persuading customer to rent or buy movies based on my good product knowledge made me more diplomatic. Customer-focus is critical to achieve store's commercial goal. I also learnt to handle problems under emergency. （基于我丰富的商业知识，我成功地劝说用户租或买电影光盘，我的商业手段在此期间更加圆滑，并成功地达到了各种商业目的。我也学会了如何在紧急情况下处理难题的能力。）

注：有的把 experience（表述）美化一下，可以很 impressive（吸引人），譬如你的职位是 prime time part time job， 说白了就是收银员。但可以说成 Customer service representative，这样可以表现你的商业意识。

A： Internship in Panda Communications Company Ltd. Research & Development Centre. I Studied the detailed service path of the mobile systems. Analysed Base Station Controller platform. Tested upgrading patches for Mobile Switching Centre.（在熊猫通讯软件公司实习期间，我学习手机系统集成线路的各种知识理论，分析各种控制平台系统，测试手机评测中心的各种升级补丁包。）

Working with different teams in the R&D centre as HQ wanted to integrate a few operating systems of defferent division together.（在研发中心与不同的团队工作，集成不同部门的一系列不同的操作系统。）

I appreciated more the importance of the coordination and teamwork. In this case, delay of one team affected the whole process. I also learnt to understand the complexity of project management.（我十分感谢团队协作和共同进步的重要性，在项目中，一个人的延误会

导致整个进程的拖延。此外，我也开始学习理解项目管理的复杂度。）

面试例题 7：Please give details of a technical project undertaken as part of your graduate degree （请讲一下你上研究生时涉及的项目细节。）

A：My final year project is closely related to 3G network - to examine a new scheduling method for transmitting data packages from network operator to users such that the transmission rate is maximised. The project involves the study of mathematical model of conventional and new scheduling methods; programming and designing of simulations under MATLAB environment; and also analysing of simulation results. It combines statistics, program designing, and results analysing.（我最后一年的项目紧扣 3G 网络：通过检查网络操作者传输数据包的新时序方法，可以将传输率最大化，效益最高。这个项目包括常规的数学建模的学习和新测试方法以及在 MATLAB 环境下的模拟编程设计；分析了测试结果，包括统计学、程序设计、结果分析。）

面试例题 8：How do you realize the job you apply? What is your blue print in future?（你如何理解你所应聘的职位？你未来几年的规划是什么？）

A：I apply for a QA job (QA means quality assurance). I think I'll communicate with R&D, respect their thoughts,and listen to their idea. I'll get to know the schema of the project deeply, collaborate with my colleagues well and I hope I'll reveal my ability and talents in my field adequately.QA is a profession requires much patience, sometimes you cannot get the conclusion you want although you have tested it time and time again, and it will waste your time and you maybe feel frustrated. I think if you wish to be a automatic thorough independent professional QA, these is a long road for you to go,a lot of troubles for you to overcome, but I 'm sure I can succeed.

（我应聘的是 QA 职位，QA 意味着质量保证。我们应该很好地与研发部门交流。我计划对项目的结构做深入的了解，与我的同事们很好地合作。我希望能充分展示我在这个行业的能力和智慧。QA 是一个需要有耐心的职位，有时即便测试千回，也不能得到你想要的结果，这会浪费很多时间，自己也会觉得很无聊。我想，做一个自动化的、彻底的、独立的、专业的测试人员肯定会有些坎坷，但我坚信我能成功。）

A：Perhaps, an opportunity at a management position would be exciting. As I have some administrative experience in my job, I may use my organizational and planning skills in the future.（也许有机会，我将会从事管理工作。我在上一个工作中积累了一些行政经验，我将来也许要运用我在组织和计划上的经验和技巧。）

面试例题 9：What is the largest-scale company you have worked for, and the smallest one? What have you done for them?（你所服务的最大的组织是什么？最小的是什么？都做些什么工作?）

A：In this three years I worked for my tutor. To be specific, I do system analysis, trouble shooting and provide software support.（在 3 年研究生的阶段中我为我的导师工作。具体地说，我做系统分析、解决问题及软件供应方面的支持。）

面试例题 10：What is a Project Manager in your eyes?（在你眼中的项目管理者是什么样的？）

A：The person or firm responsible for the planning, coordination and controlling of a project from inception to completion, meeting the project's requirements and ensuring completion on time, within cost and to required quality standards. A system analyst with a diverse set of skills–management, leadership, technical, conflict management, and customer relationship–who is responsible for initiating, planning, executing, and closing down a project.（项目管理者是那种负责项目从概念到实施整个过程的计划、协调、控制的人或公司，他（们）能够满足项目的需求，并确保项目在有限成本内按时完成，达到要求的质量标准。要启动、计划、执行并完成一个项目，系统分析师需要具备一些不同的技能，如管理能力、领导能力、技术、冲突管理及客户关系。他必须为项目的起始、计划、执行，以及项目的完成或中断负责。）

A：The Project Manager defines, plans, schedules, and controls the project. The project plan must include tasks, deliverables and resources – the people who will perform the tasks. The manager will monitor and coordinate the activities of the team, and will review their deliverables.（项目管理者定义、计划、安排、控制整个项目。一个项目计划必须包含目标、可行性、资源及可以执行此项目的人选。管理者应该监控和协调整个项目组的行动，并时时检查项目的可行性。）

A：PM is coordinator, assistant and best friend of team members, planner, monitor, reporter of the project or progress.（项目管理者是一个协调者，一个很好的助理，是研发人员最好的朋友，是一个策划者、领导者，同时还是项目流程的报告者。）

面试例题 11：What is your favorite working atmosphere?（你理想的工作环境是什么？）

A：I 'd like to work in a harmonious surrounding.Everybody pays great effort to do his job for the company. Accomplish every project with the best of my ability. What provide me

with a sense of accomplishment, your effort would win approbation. Working as a team member allows me to gain from others within the group. （我喜欢工作在一个和谐的环境里。每个人能尽其最大的努力为公司做事，尽其最大的努力完成每一个项目。完成一份工作，我们会得到满足，并具有成就感，你的努力能够得到公司的嘉许。在这个组里作为一员，我能从他人身上受益，我也会令他人受益。）

A：My ideal job is one that allows me to combine my technical attributes, business skills, and critical thinking skills in an attempt to help solve problems and create solutions.（我理想的工作是允许我把技术品质和事业技巧结合在一起，尝试帮助解决问题和寻求解决方案。）

A：I will try my best to arrange my time, I think I will find balance point between work and relax.（我将尽我所能去安排时间，我相信我会在工作和休闲之间寻找平衡。）

面试例题 12：Do you have the qualifications and personal characteristics necessary for success in your chosen career?（你拥有在你所选择的职业里获得成功的资质和必备的个性特点吗？）

A：I believe I have a combination of qualities to be successful in this career. First, I have a strong interest, backed by a solid, well-rounded, State-of-the-art education, especially in a career that is technically oriented. This basic ingredient, backed by love of learning, problem-solving skills, well-rounded interests, determination to succeed and excel, strong communication skills, and the ability to work hard, are the most important qualities that will help me succeed in this career. To succeed, you also need a natural curiosity about how systems work —— the kind of curiosity I demonstrated when I upgraded my two computers recently. Technology is constantly changing, so you must a fast learner just to keep up or you will be overwhelmed. All of these traits combine to create a solid team member in the ever-changing field of information systems. I am convinced that I possess these characteristics and am ready to be a successful team member for your firm.（我相信我具备在这一职业中取得成功的一系列资质。首先，我有极强的爱好，以扎实、广泛、艺术性的教育为背景，特别是在一个以技术为主导的职业领域中。这些基本条件，加上对学习的热爱、解决问题的能力、广泛的兴趣、成功与求胜的决心、很强的沟通技巧、努力工作的能力，都是能够帮助我在职业生涯中取得成功的重要品质。为了成功，我们需要有一种对系统如何运行的出自本能的好奇心，这种好奇心是最近我在升级我的两个计算机的时候发现的。新技术不断取代旧的，所以，我们必须成为能跟上步伐的快速学习者，否则将被他人打倒。所有的这些优点联合起来，塑造成一个能在不断

变换的信息社会中生存的坚实的团队中的一员。我确信我拥有这些品质，我有信心成为贵公司成功团队中的一员。）

面试例题 13：Any question?（你有什么要问我的吗？）

A：No sir.（没有，先生。）

A：Since I can go to work in May, I'd like to know when the HR will send me the offer.（既然我可以在 5 月份入职，我想知道什么时候人力资源部会给我发录用通知。）

面试例题 14：How do you estimate our interview?（你如何评价我们的面试过程？）

A：I've already passed interview for the second turn and also the calling interview from the Vs, but the HR notified me to go there for work in July, and the interviews between teams are different. So I've to review the interview procedure.

It was really astonished when I was told to take the interview for the second time, but I thought that there would be sorts of spasmodic problems and disaster in a programmer's career. What can I do is only to face it peaceful. So it doesn't matter at all.

（我已经通过了两轮的面试及从美国来的电话面试，但是 HR 通知我 7 月入职，并说不同 Team 的面试是不一样的，所以要求我把面试流程重新走一遍。

我得到重新面试的消息非常惊讶。但我想，作为一个测试人员，会遇到很多突发的问题和灾难，只要平和面对就是了。所以我想这也算不了什么。）

第24章

智力测试

智力测试其实是考查应聘者在限制条件下解决问题的能力。这类题目会出现于跨国企业的招聘面试中，对考查一个人的思维方式及思维方式转变能力有极其明显的作用。而据一些研究显示，这样的能力往往也与工作中的应变与创新状态息息相关。所以回答这些题目时，必须冲破思维定式，试着从不同的角度考虑问题，不断进行逆向思维，换位思考，并且把题目与自己熟悉的场景联系起来，切忌思路混乱。

24.1 关于数字的智力测试

面试例题 1：1000 瓶药水，其中至多有 1 瓶有剧毒，现在给你 10 只狗在 24 小时内通过狗试药的方式找出哪瓶药有毒或者全部无毒（狗服完药 X 小时后才会毒发。19<X<23）

答案：按 10 进制方式给狗编号分别是 1～10。按 2 进制方式给药水编号。按照药水第几位数字为 1，给相应的狗服药，如下：

```
第 1 瓶    0000000001 第 1 位数字为 1，给 1 号狗服药。
第 2 瓶    0000000010 第 2 位数字为 1，给 2 号狗服药。
第 3 瓶    0000000011 第 1、2 位数字为 1，给 1、2 号狗服药。
第 4 瓶    0000000100 第 3 位数字为 1，给 3 号狗服药。
……
第 99 瓶   0001100011 第 1、2、6、7 位数字为 1，给 1、2、6、7 号狗服药。
……
第 455 瓶  0111000111 第 1、2、3、7、8、9 位数字为 1，给 1、2、3、7、8、9 号狗服药。
……
第 1000 瓶 1111101000 第 4、6、7、8、9、10 位数字为 1，给 4、6、7、8、9、10 号狗服药。
```

最后看哪只狗毒发，则通过狗的编号得出药瓶号码。比如，1、2、3、7、8、9 号狗毒发，则 455 瓶（编号 0111000111）为毒药。

面试例题 2：If F(n)=50!* 5^n, (n is positive integer), what number is the least value of n such that F(n) and F(n+1) have the same number of trailing 0's (F(n) is represented in decimal)?（如果 F(n)=50!* 5^n，n 是正整数，请问 n 最少是多少时，F(n) 和 F(n+1)才拥有相同多的 0 作为尾数。）

A．34　　B．35　　C．36　　D．37

解析：如果想要得到相同数量的尾数 0，必须要算清楚要有多少个 2*5=10。1～50 的阶乘总共有 25+12+6+3+1=47 个 2 的因子，10+2=12 个 5 的因子，多出 47-12=35 个 2 的因子。所以必须配套 35 个 5 来得出 10。

答案：B

面试例题 3：公司有 1 000 个苹果和 10 个箱子，事先将 1 000 个苹果分别装入 10 个箱子，问：怎样做才能在客户无论需要多少苹果时，都可以整箱整箱地提供给客户？

答案：1，2，4，8，16，32，64，128，256，489。道理很简单，第 *N* 只箱子应该装的数量是前面 1～*N*-1 只箱子装的数量的总和加 1。

面试例题 4：安娜在超市买东西，人家找给她 0.59 美元的零钱。零钱共 11 枚硬币（硬币种类包括面值 1 分硬币、面值 2 分硬币、面值 5 分硬币和面值 1 角硬币），其中只有一种硬币的总数为 3，那么这种硬币一定是（　　）？

A．1 分硬币　　B．2 分硬币　　C．5 分硬币　　D．1 角硬币

解析：根据题目列方程如下：

59=1n + 2m + 5k +(11-n-m-k)*10

59=110-9n-8m-5k

化简得

9n+8m+5k=51

然后看哪个整数等于 3。

如果 n=3，8m+5k=24，唯一的正整数解为 m=3，k=0，那么至少有两组三个钱币相同；

如果 m=3，9n+5k=27，唯一的整数解是 n=3，k=0；

如果 k=3，9n+8m=36，n=4，m=0；

只有最后一种情况即只有一个整数是 3，也就是 4 个 1 分、3 个 5 分、4 个 1 角，其他都有两组总数是 3 个硬币，无法得到答案。

因此，答案应该是 5 分硬币。

答案：C

面试例题 5：发月饼，第一个人拿 1 个和剩下的 1/9，第二个人拿 2 个和剩下的 1/9，以此类推，全部发完后，每个人所得相同且分发时不许把 1 个月饼切开。请问月饼多少，人多少?

解析：

方法 1：

假设有 X 个人，Y 份月饼，根据第一个人得 1 份，第二个得 2 份，则有最后那个人（X）所分月饼为 X 份，因为剩余一定为 0。根据大家所得相同，有 X=1+(Y-1)/9。

第一个人的为 1+（Y-1)/9＝X （1）

第二个为 2+（Y-X-2）/9＝X （2）

解方程（1）、（2）得

X=8

Y＝64

即有 8 个人，64 块月饼。

方法 2：

假设有 X 个人，Y 份月饼，根据第一个得 1 份，第二个得 2 份，则有最后（X）所分月饼为 X 份，因为剩余为 0，由题意可知，每个人都有 X 份，即得出

X*X=Y（1）

第一个的月饼为

1+（Y-1）/9＝X （2）

解方程（1）、（2）得

X=8

Y＝64

即有 8 个人，64 块月饼。

答案：8 个人，64 块月饼。

面试例题 6：有一个 100 层高的大厦，你手中有两个相同的玻璃围棋子，从这个大厦的某一层扔下围棋子就会碎。用你手中的这两个玻璃围棋子找出一个最优的策略，来得知那个临界层面。

分析：设总共楼层为 h，a(n)（如 a(1)、a(2)、…）表示每一次抛棋子所在的层次，

则对于任一次抛掷 a(n)，必须沿着上一次抛掷所在的层向上逐个尝试，即最多必须抛掷 a(n) - a(n-1) - 1 + n 次。

以此类推，考虑平均值，将各次求和，消去之后得到

(a(n) - n + (1+n)*n/2) / n

由于 a(n) = h，所以等式化为 h/n + n/2 + 1/2。

其最小值发生于 n = (h*2)^(1/2)的时候。

代入 h = 100，得到 n 约等于 14。

即在最坏情况下，约需 14 次完成。

答案：最多 14 次。

从 14 层开始扔第一次，如果碎了，那么从第 2 层开始扔，一层层加，直到 13 层，一共 14 次。

如果没有碎，在 27 层再扔一次。以此类推，从 15 层到 26 层一共 12 次，加上前面的 14 层、27 层 2 次，所以说也是 14 次。

依次这样扔：

14

14+13=27

27+12=39

39+11=50

50+10=60

60+9=69

69+8=77

77+7=84

84+6=90

90+5=95

96

97

98

99

最多 14 次。

24.2　关于推理的智力测试

面试例题 1：故事背景：

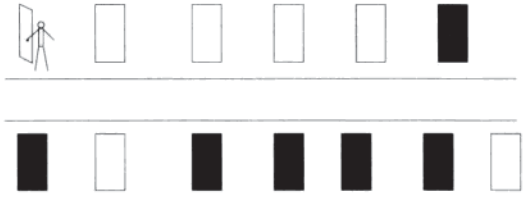
一所监狱关着 N 个人，每人都被关在单独的房间里，房门外面标着从 1 开始的号

码，表示犯人的号码。突然有一天，国家发生了战争，监狱长想“我是应该释放这些犯人呢?还是应该放弃他们离开这里呢？”（如果是一群无知的犯人，释放他们反而会给居民们带来更大的危险。如果是一群有脑子的犯人，说不定还能对抗敌人，为国家做点贡献）。最后，监狱长决定用一个智力题考验监狱里的所有犯人。

题目如下：

给每个犯人的房间门外侧涂上颜色，颜色要么是“白色”，要么是“黑色”。从 1 号犯人开始，一次只能一个人，出来看一下其犯人们的门是什么颜色。但是看不了自己的门，因为自己的门当时是翻开的，背面靠墙了。此时犯人要猜测自己的门是什么颜色，并告诉监狱长，然后回到自己的房间。监狱长把每个人的答案记下来，并且统计有多少人猜对了。

如果有一半的人猜对了，就把监狱里的所有人全部释放。如果猜对的人不到一半，监狱长就放弃这些犯人，让他们饿死。



1 号犯人是被关的时间最长的犯人，也是这个监狱犯人里的带头大哥。监狱长给 1 号犯人一个机会，让他出门后看完其他人的门后，可以使用监狱广播 1 分钟，把他的“逃命方案”告诉所有的犯人，然后再回房间。

假设你是这个 1 号犯人，你会如何设计你的“逃命方案”呢?

答案：

设黑色门为 X，白色门为 Y，则总数为 X+Y。1 号犯人出来之后，他只会遇到以下三种状况：X 为 1 号犯人所见的数量为多数的颜色门数，Y 为另一颜色门数，下式中左侧 1 表示 1 号犯人自己的门数，X+Y+1=总数。

① ||X-Y|-1| >= 2 的情况最简单，开门犯人自己的门颜色不影响大局，可直接要求所有的犯人猜自己的门是处于多数的颜色（X>Y 即猜 X，Y>X 即猜 Y），即可保证半数以上的人猜对，然后过关。

② ||X-Y|-1| ==1 的情况比较复杂，此时可能是 X==Y，也可能是 X==Y+2，但若 1 号犯人要求其他所有的犯人猜自己的门是处于多数的颜色(X>Y 即猜 X，Y> X 即猜 Y)，同时猜自己的门为处于少数的颜色，可以保证((X+Y)/2+1)的犯人猜对，然后过关（假若 X==Y，即 1 号犯人属于其所见多数门的反色，猜对，同时 X 个犯人猜对，结果 X+1

猜对，X==Y，即多于总数半数；假若 X==Y+2，即 1 号犯人属于其所见多数门色，猜错，同时另 X 个犯人猜对，Y 个犯人猜错，即对了 X 个，错 Y+1 个，但是 X=Y+2，即 X>Y+1，多余总数半数）。

③ ||X-Y|-1| == 0 这种情况，此时可能是 X==Y+1，也可能是 X==Y-1，但是很明显，1 号犯人的房门颜色是处于多数的那一方，因此，可以要求其他犯人假如发现除自己外，所有房门的颜色相加黑白数目相等的话，就猜自己的房门颜色为 1 号房门的颜色，而若发现所有房门的颜色相加数目不等，就猜自己房门为少数数目颜色，这样可以保证(X+1)/2X 的正确率，然后过关。

面试例题 2：三个枪手同时爱上了一个姑娘，为了决定他们谁能娶这个姑娘，他们决定用枪进行决斗。小沈的命中率是 30%，小武比他好些，命中率是 50%，最出色的枪手是小白，他从不失误，命中率是 100%。由于这个显而易见的事实，为公平起见，他们决定按这样的顺序：小沈先开枪，小武第二，小白最后。然后这样循环，直到他们只剩下一个人。那么这三个人中谁活下来的机会最大呢？他们都应该采取什么样的策略？

答案：在小沈先开枪的情况下，排名第 2 的小武一定会射小白，因为小白必定会先射命中率较高的小武。小沈最想看到的结果是小白死了，而自己又射死小武，但他又不敢先射小武，因为万一小武死了，自己肯定被小白射。所以，小沈的最优方案是也射小白，然后再射小武。小沈有两种对应方法：

方法一：小沈先射小白

A 阶段：

若小白死了（机会 30%），小武会射小沈。

小沈的死亡机会（也就是小武的幸存概率）：

0.3*0.5+0.3*0.5*0.7*0.5+0.3*0.5*0.7*0.5*0.7*0.5+...=0.220875；

小沈没死就会射小武，小武的死亡机会（也就是小沈的幸存概率）：

0.3*0.5*0.3+0.3*0.5*0.7*0.5*0.3+0.3*0.5*0.7*0.5*0.7*0.5*0.3+...=0.0662625；

B 阶段：

若小白没死（机会 70%），小武会射小白。

小白的死亡机会为 0.5。这时又分为两种情况：

情况 1：

```
到这时，如小白还没死
则小武会死（机会 100%）
小沈会射剩下的小白
小白的幸存机会：
0.35*0.7=0.245
```

然后小白就会杀死小沈，并成为唯一幸存者。
小沈成为唯一幸存者的机会（0.35*0.3=0.105）

情况 2：

如小白死了
小沈会射小武
小沈的幸存机会：
0.5*0.3+0.5*0.7*0.5*0.3+0.5*0.7*0.5*0.7*0.5*0.3+...=0.220875
如小武没死，小武会射小沈
小武的幸存机会：
0.5*0.7*0.5+0.5*0.7*0.5*0.7*0.5+0.5*0.7*0.5*0.7*0.5*0.7*0.5+...=0.2576875

总结：

小沈会成为唯一幸存者的机会=0.0662625+0.105+0.220875= 0.3921375；

小武会成为唯一幸存者的机会=0.220875+0.2576875=0.4785625；

小白会成为唯一幸存者的机会=0.245；

这三个人中小武活下来的机会最大。

方法二：如小沈一开始故意射失，小武会射小白，小武和小白的死亡机会均为 0.5。

情况 3：

如小武杀死小白，小沈会射小武
小武的死亡机会为
0.5*0.3=0.15
小沈成为唯一幸存者的机会为
0.5*0.3+0.5*0.7*0.5*0.3+0.5*0.7*0.5*0.7*0.5*0.3+...=0.220875
小武成为唯一幸存者的机会为
0.5*0.7*0.5+0.5*0.7*0.5*0.7*0.5+0.5*0.7*0.5*0.7*0.5*0.7*0.5+...=0.2576875

情况 4：

如小白杀死小武
小沈会射小白
小白的死亡机会为 0.5*0.3=0.15
小沈成为唯一幸存者的机会为 0.5*0.3=0.15
小白成为唯一幸存者的机会为 0.5*0.7*1=0.35

总结：

小沈会成为唯一幸存者的机会=0.220875+0.15=0.370875；

小武会成为唯一幸存者的机会=0.2576875；

小白会成为唯一幸存者的机会=0.35；

这三个人中小沈活下来的机会最大。在小沈开枪顺序排第一的情况下，小沈应采用方法二。

面试例题 3：屋里 4 盏灯，屋外 4 个开关，一个开关仅控制一盏灯，屋外看不到屋里。怎样只进屋一次，就知道哪个开关控制哪盏灯?

答案：四个开关 A、B、C、D。开 A 和 B，过 5 分钟，关上 A，B 保持开着，然后开 C，进去。又亮又热的是已经开了一会儿的 B，又亮又不热的是刚开的 C，不亮但很热的是刚关了的 A，又不亮又不热的是没有动过的 D。

24.3　综合智力测试

面试例题 1：史密斯先生有遗产 2000 元要分给他的两个孩子。遗嘱规定如下：

① 由哥哥先提出分钱的方式，如果弟弟同意，那就这么分。

② 如果弟弟不同意，1000 元会捐给地震灾区，由弟弟提出剩下 1000 元的分钱方式。

③ 如果哥哥同意弟弟的方式，就分掉这剩下的 1000 元。

④ 如果哥哥不同意，遗嘱规定剩下的 1000 元中的 800 元捐给灾区，然后分别只给他们每人 100 元。

问：哥哥会提出什么样的分钱方式使其利益最大化？（分配最小单位元）

附带条件：两人都极聪明且唯利是图。

A．1900　100　　B．1899　101　　C．1000　1000　　D．1101　899

解析：这是一道风险评估题，属于博弈论的范畴。

如果两次都没有通过，那么两人最少获得 100 元，所以 100 元是无风险的。

第二次弟弟分配，只要给哥哥 101 元，弟弟获得 899 元。因为 100 元是无风险的，若少于或等于 100 元，哥哥“完全可以”不同意，然后获得 100 元。

若给哥哥的大于 100 元，给得越多，哥哥不同意的可能性越小。所以第二次分配的时候，弟弟可能获得的最大利益是 899 元，不可能再多，但是可能会更少。

回头再考虑哥哥第一次分配的情况，最低风险同样是 100，给弟弟的越多，弟弟同意的可能性越大。因为弟弟在第二次分配的时候可能最大获利 899 元，所以只要小于这个数目，弟弟不同意的可能就会因为可以冒险获得更大的利益。

但是只要等于 899 元，弟弟就没必要冒险，因为不可能获得比 899 元更多，况且哥哥也有不同意的可能。

所以哥哥分配的方案应该是：

哥哥：1101 元

弟弟：899 元

答案：D

扩展知识：对这道面试题来说，我们只是就题论题，考虑的是两人都极其聪明理智且唯利是图。但是如果真的在实际生活中出现了这样的博弈，考虑的东西就会更多些，因为有人的地方就有变数，人的情感、性格不是机械的数字所能左右的。这些变数都会影响到实际博弈的结果。

比如说，现实生活中某些人总希望别人得到的比自己少，而不管自己得到多少。于是哥哥提出 1899 和 101 的方案，如果弟弟不同意，那么弟弟提出的方案哥哥也不同意。那最后双方都得到 100。对于两个人来说都是双亏的抉择，当然这是一种极端的情况。

所以，我们现实考虑问题的时候，要考虑最通常的情况，即每个人都希望拿得尽量多，如果万一拿得比对方少，也不要差距太大（古人云“不患寡患不均”）。因此，实际生活中的情况可能是这样的：

（1）假如进入第二轮分配，弟弟会冒险地提出 899、101 的分配方案吗？显然不会，这个风险太大了，他完全没有胜算。因为哥哥可能宁可不要多的一块钱，也要和弟弟的差距不大。他必须拿出一定的钱来换取哥哥至少有 50%以上的可能性来同意他的分配方案，只要在保证分配方案得到对方点头的情况下，利益最大才能得到体现。因此，要博弈就不该仅仅考虑自己的利益，这放在商业上也是这样的。

（2）弟弟的方案最保险的是每人 500，但既然要考虑利益最大，那就不可能是公平地进行分配，那弟弟的分配方案应该是让哥哥得到 100～500 的中间值，也就是哥哥 300，自己得到 700。

（3）如果哥哥考虑到弟弟可能提出这种方案的话，也就是让弟弟得到 700，自己得到 1300。这种分配方案，弟弟同意的可能性过半，所谓博，就在于此，不能一味去追求极端的情况。

这道题出得非常好，它告诉了我们好伙伴之间怎样合理分配利益，这是两者能长久合作的基础之一。大家都是做企业的，获取最大利润无可争议。既然我们是兄弟般的合作者，分配的方法应该让双方心服口服，不要以为只有自己是智者，也不要压人。总之，要有理有节达到共赢的局面。

面试例题 2：During Olympic Games this summer, lots of people visited Beijing. However, many visitors have complains/difficulties towards trip since most hotels are very expensive or already fully booked. On the other hand, some local people are eagerly to rent out their apartments during Olympics, but don't know how to effectively publish the information,

even after Olympic, a lot of visitors and house renters are still facing the same problems（本届夏季奥运会期间，有许多人来北京观光旅游，然而许多游客不是抱怨住宿费太高，就是旅馆爆满很难定到房间。同时一些当地人一方面希望奥运期间自己的房子能够出租，另一方面又不知道如何更有效地打出租广告。即使在奥运高峰之后，游客和房主还会面临同样的问题。）

You're assigned to solve this problem, and what are the action(s) you want to take for your first step（假定你被派去解决这一问题，你**第一步**会采取什么措施？）

A. Contact some companies and ask for some initial funding（和一些公司联系请求基本基金援助）

B. Work with your colleagues to do user study to figure out more on who may be your customers （和同事精诚合作，分析潜在的客户群）

C. Work with your Developer and Tester to build a platform to help these customers（与开发商和监测人员联系共同搭建平台为这些顾客提供援助）

D. Find some of your friends outside the company, discuss about the project details, and see if they can help you.（找些业外朋友谈谈具体细节，看他们能否想办法帮忙）

解析：这个题目是一个开放性的题目，以下答案仅供参考。笔者认为题目问的是你第一步做什么？那就是需求分析，找清楚我们的潜在客户群是哪些旅游者和租主。选项 A、C、D 都不是第一步要做的。

答案：B

面试例题 3：You are the PM of online shopping project. This project is running well and will be released to customer in two months. But the marketing team demands to add two new features in this release and emphasize these two features are critical because our major competitors already have similar features . At the same time, you found out that all developers and tester already had work items planned for next two months. What are the right possible action(s) to take to deal with this issue?（你是一个在线购物工程的项目经理。这个项目运行正常并打算在两个月内投放市场，但是营销团队要求加入两项新功能且强调它们是非常重要的，因为我们的主要竞争对手已经拥有了这两项新功能。可同时你却发现所有的研发者和测试者在未来两个月内都已经安排好了工作内容。下面哪些是处理这个问题比较可行的方法？）

（1）Discuss the issue with the management team and try to request new resources to fulfill these two features(和团队集体讨论并想办法寻找新的人员去增加这两个新功能。)

（2）Reject the new features request directly so that you can ship current release on time

（否决增加这两个新功能的要求，以使项目按期投入使用。）

（3）Decrease the testing criteria to reduce the test duration so that you can add the two new features （降低测试标准以减少测试期限，以便能增加这两个新功能。）

（4）Go through all the left work items with marketing team and other partners to punt low priority items to next release（将下一个项目的工作推后，优先和销售团队以及其他搭档一起完成增加新功能的工作。）

（5）Discuss this issue with stakeholders (management team, marketing team etc) and propose to postpone current release（和所有参与项目的人讨论（管理团队以及营销团队等）并提议延期投放该项目。）

A．1, 3 and 4　　B．2, 3 and 4　　C．1, 3 and 5　　D．1, 4 and 5

解析：2、4 是首先被否决的选项。因为这些是公司高层决定的事情，仅是一个项目经理，不可能越俎代庖做这样的决定。

你首先考虑的是，应该与管理团队和营销团队讨论，在讨论之前应该做一些准备：

（1）与项目团队集体讨论有什么可替代方案。如果要增加功能，会需要多少时间和资源，对现有项目和其他项目有什么影响。

（2）尽可能考察竞争对手的功能。

（3）听取营销团队的功能需求。

（4）如果降低测试标准会节省多少时间，对项目的稳定有多少影响做出基本评估。

综合以上情况，提出可能的几种解决方法，可以有倾向性意见，但不能做结论。重点在于客观描述采用每种方案在时间、资源需求、稳定性等方面的情况，同时做好在每种方案下，怎么调度人力和资源的准备。然后在与管理团队和营销团队的讨论中，提出准备的情况，如果一次讨论不能决定，可能需要重复这些步骤再次讨论。最终应该由讨论达成共识，或者由管理团队做出决策。

因此 1、3、5 条是可行方案。至于第 3 条，有人觉得合适，有人觉得离谱，这是由于看问题角度不同。从工程角度看，每个程序员都像艺术家一样，希望每个作品都是完美的，可站在更高层面上看呢？这里没法分析具体情况，但可以肯定的是，高层董事会有一个自身利益诉求。最看重并非一个项目是否完美（即使获全球最佳项目奖也没用），而是项目带来的收益（效率、业务延伸、收入等）。那么在一个项目中可能包含有很多功能，最被看重的功能可能只是项目（工作量）中的一小部分，只要确保这部分，其他部分的瑕疵可能是被容忍的，完全可以留待今后完善和修改。这样，未必不能减少一部分测试。

答案：C

[General Information]
书名=Java程序员面试宝典 第二版
作者=欧立奇，朱梅，段韬编著
页数=348
出版社=电子工业出版社
出版日期=2011.06
SS号=12815546
DX号=000008126529
URL=http://book2.duxiu.com/bookDetail.jsp?dxNumber=000000812652 9&d=B55AFE46D5481BB0B83A9A4C29B48CEC