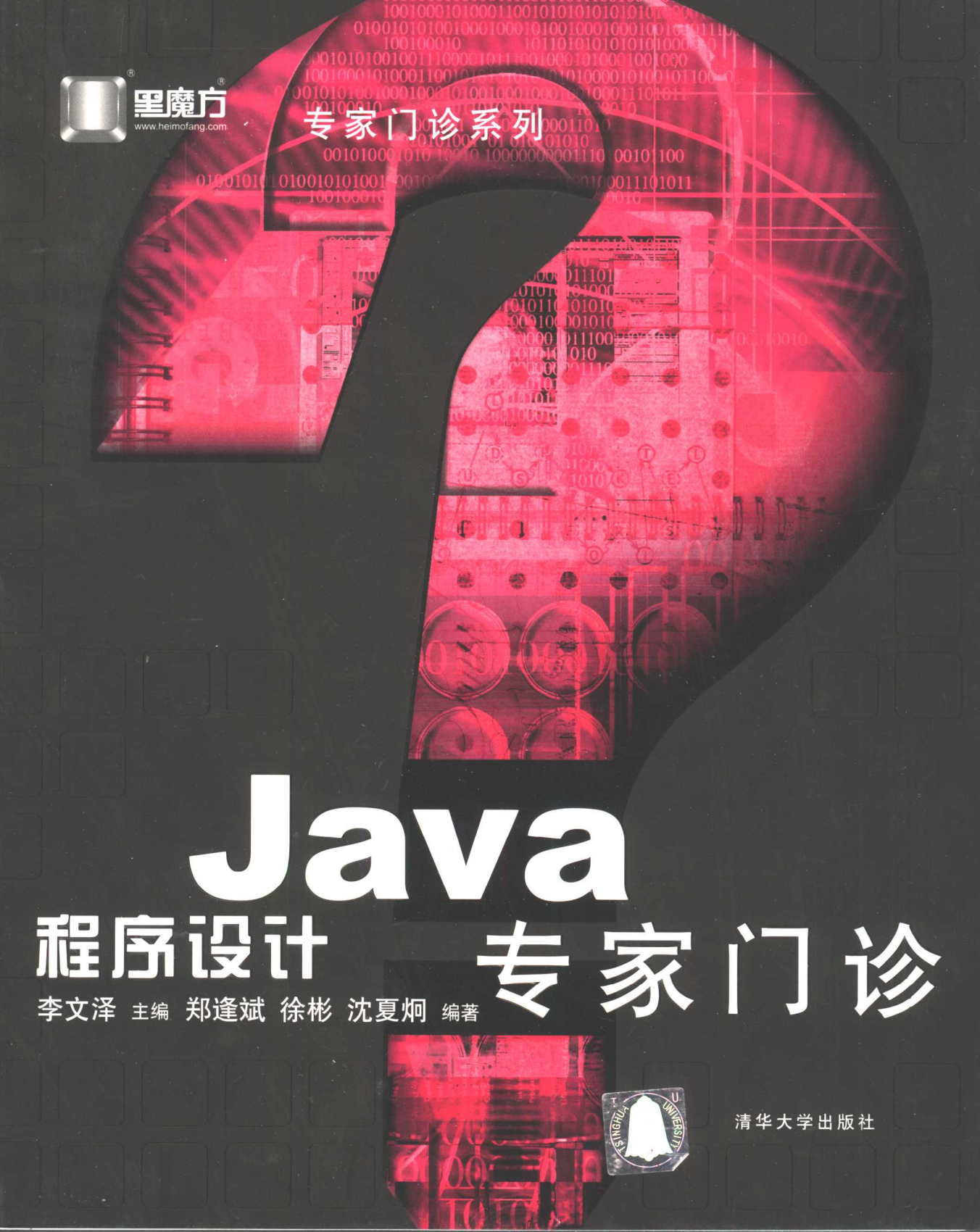
黑魔方
www.heimofang.com
专家门诊系列
Java
程序设计
专家门诊
李文泽 主编 郑逢斌 徐彬 沈夏炯 编著
清华大学出版社

专家门诊系列

Java 程序设计 专家门诊

李文泽 主编 郑逢斌 徐彬 沈夏炯 编著

清华大学出版社

内容简介

本书重点介绍了在 Java 语言的学习过程中所遇到的各种问题，这些问题都是非常基础的，但也是非常重要的，是很多学习 Java 的人无法从课堂上弄明白的问题。本书按照普通 Java 教材的顺序循序渐进地回答读者学习过程中经常遇到的问题。使读者不仅知其然，而且还能知其所以然。如果您是第一次学习 Java，通过本书就能够完全掌握 Java。如果您已经学过 Java，却仍感到有很多疑问，相信本书能够为您扫清这些阻碍您前进的障碍。

本书的最大特点就是针对每个问题从解题思路、具体步骤、专家指点几个方面进行介绍，让读者在解决一个问题的同时取得举一反三的效果，本书所给出的问题在论坛中也经常出现，都是学习 Java 的人比较关心的问题，具有很好的参考价值。此外，本书强调理论与实践相结合，注重内容的实用性，提供并详细讲解了大量的程序实例。

本书内容丰富、语言通俗易懂、实用性非常强，适合 Java 程序设计的初学者，对于有一定经验的 Java 读者也具有极高的参考价值。本书可作为课堂教学、培训辅导、国际认证考试的教材，也可作为程序开发人员的自学参考书。

图书在版编目（CIP）数据

Java 程序设计专家门诊 / 李文泽主编. —北京：清华大学出版社，2006.4
ISBN 7-302-12485-X

Ⅰ.J…　Ⅱ.李…　Ⅲ.JAVA 语言-程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2006）第 006261 号

出 版 者：清华大学出版社　　**印 刷 者**：北京鑫丰华彩印有限公司
地　　址：北京清华大学学研大厦　　**装 订 者**：三河市金元印装有限公司
http://www.tup.com.cn　　**发 行 者**：新华书店总店北京发行所
邮　　编：100084　　**开　　本**：185×230　**印张**：29.5　**字数**：643 千字
社 总 机：010-62770175　　**版　　次**：2006 年 4 月第 1 版　2006 年 4 月第 1 次印刷
客户服务：010-62776969　　**书　　号**：ISBN 7-302-12485-X/TP·8004
责任编辑：魏江江　　**印　　数**：1～4000
装帧设计：吴文越　　**定　　价**：46.00 元

前言

谢谢您阅读本书，为了能更好地帮助您学习本书的知识，请仔细阅读下面的内容。

读者对象：

本书是为那些有一定的编程经验、并想要进入 Java 编程领域的初学者实现入门和提高编写的。全书用通俗易懂的语言，由浅入深地对 Java 的各个方面进行了介绍，既可以满足那些通过课堂还无法掌握 Java 的初学者的入门需要，也可以让那些学有余力的读者有充分思考的空间。因此，本书无论是作为课堂教学、培训辅导，还是国际认证考试都将使读者受益无穷。

写作环境：

本书采用的 Java 开发环境是 JDK1.5，操作系统可以使用 Windows2000/XP/2003。代码编辑工具可以使用 Editplus、JCreator 或者 UltraEdit 等。建议读者使用 JCreator LE 版。

本书特色：

- 精选目前国内外最流行的程序设计语言——Java 作为本书的选题，并以丰富的内容来解决读者学习该语言时可能遇到的各种问题。
- 以专业的论坛为基础，选择其中典型的问题和技巧，为读者解决各类疑难问题。
- 明确定位，面向初、中级读者，由“入门”起步，侧重“提高”，愿新手老手都能成为行家里手。
- 围绕用户实际使用之需取材谋篇，着重问题的剖析和操作技巧的指点，解决读者的疑难问题，并使读者能举一反三。
- 追求明晰精练的风格，用醒目的步骤提示和生动的屏幕画面使读者如临操作现场，轻轻松松地掌握 Java 语言。

内容提示：

本书重点介绍了 Java 语言的 8 个方面：Java 基础、Java 与面向对象、异常处理、Java 多线程、Java 输入/输出、Swing 编程、Applet 编程、网络编程。每个方面作为 1 章，每 1 章都涵盖了 Java 学习过程中的重点和围绕这些重点所遇到的问题。其中，每个小节都以提出并解决一个问题的方式进行展开，并通过以下 6 个步骤进行介绍。

第 1 步：给出为什么会提出这个问题。

第 2 步：分析问题的特征，归纳同类问题相应的特性及解决方法，就像老师上课时，在讲解 1 道题目时，常会先进行分析，总结规律和解题思路，以便学生可以按照这种思路解决同

样类型的问题。

第 3 步：给出解决这个问题的实际操作步骤或者源代码。

第 4 步：对解决问题的方法进行综合性的总结，告诉读者应该如何运行源程序或最终文件，以及运行后的效果图或者现象等。

第 5 步：告诉读者通过该问题的解决过程，需要掌握什么内容或技巧，你能够学到什么，以及其他相关的知识点。

第 6 步：列举与该问题类似或相关的问题，并给出解决问题的关键性提示。

在这 6 个步骤中，还会增加以下内容，以丰富图书内容、辅助读者阅读理解、增加内容的生动和活泼性。

注意——提醒操作中应注意的有关事项，避免错误的发生，让您少一些傻眼的时刻和求救的烦恼。

提示——提示可以进一步参考的章节，以及有关某个内容的详细信息，使您可深可浅，收放自如。

技巧——指点一些捷径，透露一些高招，让您事半功倍、技高一筹。对于只需要一两句话就可以解决的问题，都在此处，再作为一个小节讲解。

试一试——精心设计各种操作练习，您只要照猫画虎，试上一试，不仅能在您的电脑上展现出书中的美妙画面，还能了解书中未详述的其他实现方法和可能出现的其他操作结果。随处可见的“试一试”让您边学边用，时有所得，常有所悟。

分析——分析为什么要这样做，指出操作的关键，介绍其他操作的结果。使读者知其然，也知其所以然，从而举一反三。

本书作者：

参加本书编写的作者是从事多年教学工作并具有丰富经验的 Java 一线教师。

本书主编：李文泽，编写准备知识，第 1 章，第 2 章和第 5 章。

本书编委：徐彬，编写第 4 章；沈夏炯，编写第 6 章；郑逢斌，编写第 7 章；曹蕾，编写第 3 章和附录；贾真，编写第 8 章。

IT 书吧（www.itbook8.com）提供本书相关资料下载。

目录

准备知识

0.1 为什么会产生 Java，它有哪些特点

Java 是一种可以编写跨平台应用软件的面向对象的程序设计语言，它最初是由 Sun 公司的 James Gosling 等人于 1991 年提出设想并开始设计开发的，第 1 个版本花费了大约 18 个月的时间，开始起名为 Oak，于 1995 年更名为 Java。到目前为止，Java 已经成为一种广泛使用的网络编程语言。因为它所编写的程序不但可以保证安全、稳健且和平台无关的通过网络传播，而且更容易编写、管理和维护，代价也更低。

说起来多少有些令人吃惊，Java 的最初推动力并不是网络，而是源于对平台无关性的需要。Java 的前身主要是作为一种用于小家用电器的编程语言，解决诸如电视机、电话、微波炉等家用电器的控制和通信问题。因为在这种小型家用电器中用作控制器的 CPU 芯片是多种多样的，当时主流的编程语言 C 和 C++所开发的程序只能一次应用于某一个特定的芯片。尽管可以为所有的 CPU 芯片编写各自的程序，但是创建适用于各种 CPU 芯片的编译器是一项非常费时费力的工作。为了找到一种简单而且经济的方案，Sun 公司的 Gosling 等人便开始一起致力于开发一种可移植、跨平台的语言，该语言能够生成运行于不同环境、不同 CPU 芯片上的代码。然而，由于这些智能化家电的市场需求没有预期的高，Sun 公司放弃了该项计划。就在 Oak 语言几乎夭折的时候，Internet 技术得到了空前的发展，Gosling 等人通过和 Web 技术的对比，看到了 Oak 语言在计算机网络上的广阔应用前景，于是改造了 Oak，并在 1995 年 5 月 23 日正式以 "Java" 的名称对外发布。同年，Sun 公司决定免费向公众发放 Java 的开发工具包，并和当时著名的网景公司合作，将 Java 的虚拟机加入到 Nescape 浏览器当中。这一系列举措都使得 Java 伴随着 Internet 的迅猛发展而发展起来，并逐渐成为重要的 Internet 编程语言。

要想更好地了解 Java 为什么会如此成功，就必须了解它产生的原因、推动它发展的动力，以及它对其他语言的继承。Java 的产生与过去 40 年中计算机语言的改进和不断发展密切相关。而每一次语言设计的革新都是因为先前的语言不能解决目前遇到的基本问题而引起，Java 也不例外。归纳起来，计算机语言的革新和发展需要两个基本因素：第一就是它必须适应正在变化的环境和需求，第二就是它必须能够实现编程艺术的完善与提高。

说到这里，我们不得不提到 C 语言。20 世纪 70 年代中期，C 语言的出现深刻的影响了整个计算机界，并从根本上改变了编程的方法和思路。可以说，C 语言是当时人们不断追求结构化、高效率的直接结果。然而程序设计语言的发展在这之前又是怎样的局面呢？主流的编程语言如：汇编语言虽然能够写出高效率的程序，但是不便于学习，而且使用和调试汇编程序也相当困难；BASIC 语言虽然容易学习，但功能不够强大，更不用说结构化，这使它根本无法应用到较大的程序中；FORTRAN 语言在科学计算应用方面确实可以编写出相当高效的程序，但它不适于编写系统程序。尽管当时人们对编程语言实现结构化的要求

非常强烈，然而 BASIC 和 FORTRAN 等程序设计语言都没有考虑结构化问题，而是使用 GOTO 语句作为对程序进行控制的主要方法。这样一来，用这些语言编写的程序往往就成了“意大利面条式的程序代码”，一大堆混乱的跳转语句和条件分支语句使得程序难以理解，并且给调试工作也带来了巨大的困难。Pascal 语言正是为缓解这一矛盾而出现的，不过虽然 Pascal 是结构化语言，但是它的设计效率很低，仍然无法编写系统程序代码。而当时的情况是，早期的计算机语言开发软件的效率和功能已经根本无法满足人们对软件的需求量日益增加的需要。也就是在这个时候，C 语言出现了，它功能强大、高效、满足结构化要求、简单易学，而且它的设计、实现、开发是由真正从事编程工作的程序员在考虑了现实编程工作的实际需要后而完成的。它的特性经由实际运用该语言的人们不断去提炼、测试、思考、再思考，使得 C 语言最终成为程序员们最喜欢使用的程序设计语言。幸运的是，Java 也继承了这个思想。

正当 C 语言成为主流的计算机编程语言的时候，问题又出现了，也就是在 20 世纪 80 年代初，随着软件规模的增大，软件设计和开发的复杂性日益突出，对于软件的管理甚至需要依赖于工程学的方法才能够实现。此时，C 语言已经显得力不从心。因为，许多工程项目的复杂性都超过了结构化方法的极限。为了对程序的复杂性更好地进行管理，面向对象编程(OOP)方法诞生了，这种方法的核心思想是通过使用继承、封装和多态性来帮助组织复杂的程序。C++是这一时期非常重要的面向对象程序设计语言，它是在 C 语言的基础之上通过增加面向对象的特性来扩充 C 语言，因此它包括了 C 语言的所有特征、属性和优点。这也是 C++作为编程语言成功的一个关键因素。因为 C++的发明并不是企图创造一种全新的编程语言，而是对一个已经高度成功的编程语言进行改进。正是由于 C++既有面向对象的特征，又有 C 语言高效和格式上的优点，使得它直到现在也是一种可以被广泛应用的编程语言。然而，语言的发展并没有结束，互联网的迅猛发展使得程序设计语言又产生了新的变化，这一变化最终导致了 Java 的出现。

导致 Java 出现的根本力量是对程序可移植性的迫切要求。尽管这种要求并不是在互联网出现之后才有的，而是和程序设计语言的出现几乎同步，但是由于各种原因，这种需求并不是那么迫切。然而，随着 Internet 和 Web 技术的出现和繁荣，对程序可移植性的要求变得重要且紧迫起来。这是因为，互联网中存在着大量不同的、分布式的系统，这其中包括了各种类型的计算机、操作系统和 CPU 芯片。人们虽然实现了各种类型的平台都可以和互联网进行连接，共享资源，但是还无法在各种平台上运行同样的程序。因此，可以这么说，中立体系结构编程语言的需要是促使 Java 诞生的主要动力，而 Internet 最终导致了 Java 的成功。

实际上，Java 的大部分特性都是从 C 和 C++中继承而来的，这也是 Java 成功的一个方面。Java 的设计人员觉得，在新语言中使用熟悉的 C 语法及模仿 C++面向对象的特性，将使他们的编程语言对经验丰富的 C 和 C++程序员有更大的吸引力。而且，Java 的设计和

测试工作是由真正从事编程工作的人员完成，它源于设计它的人员的需要和经验，因而也是一种程序员自己的语言。可以说，Java 是一种在各方面性能表现都很好的编程语言，它的特点可以表现在以下几个方面。

1/ 简单

Java 是一种面向对象的程序设计语言，只要学过 C 或者 C++，那么学习 Java 也会非常容易，因为 Java 继承 C 和 C++的语法以及许多 C++面向对象的特性。当然，即使没有学过其他程序设计语言，学习 Java 也不是很难，因为 Java 的设计目的就是让程序员或者准程序员觉得既易学又好用。Java 简单的另一个方面是它摒弃了 C++中许多容易出错和混淆的概念，如指针、运算符重载、多重继承等，取而代之的是通过其他更清楚、更易理解的方式来实现。此外，Java 还通过实现自动垃圾回收机制而大大简化了程序设计人员对内存的管理工作，减少了错误的发生。

2/ 面向对象

Java 语言借鉴了近几十年来所有面向对象程序设计语言的优点。提供了简单的类机制和动态灵活的接口模型，使得开发工作完全集中于对象本身和接口的设计。通过对对象状态以及行为的封装实现了信息的有效隐藏和模块化要求。通过类的继承机制实现了向上对问题域的高度抽象以及向下对程序代码的有效复用。可以说，Java 保证了对象模型既简单又容易扩展。

3/ 稳健性与安全性

在设计 Java 的时候，最优先考虑的就是怎样才能创建出稳健的、安全可靠的程序，这是因为在网络这个多平台的环境中，程序必须得到可靠的执行。为了实现这个目标，Java 被设计成一种严格类型检查的语言，它不但在编译时对代码进行检查，而且在运行程序时也会对可能出现的问题进行检查，这样就可以及早地发现错误并进行消除。其实，程序失败的原因无外乎以下两种：内存管理错误和误操作引起的异常情况。

使用 C 和 C++编写的程序虽然效率很高，但是很多复杂的工作都需要程序员自己来实现。比如内存的管理，C 和 C++的程序员必须手工分配并且释放所有的动态内存，这样做就会导致一些问题，如程序员可能忘记释放原来分配过的内存，或者释放了其他部分程序正在使用的内存。为了从根本上消除这些问题，Java 提供了自动垃圾回收机制来帮助程序员管理内存的分配和释放，从而将程序员从复杂、乏味的内存管理工作中解脱出来。

此外，类似于“被零除”、“数组下标越界”或者“文件未找到”这样的误操作也经常在程序中发生。如果程序员在 C 或者 C++中要避免这类问题发生，必须编写一大堆难以理解的指令。而 Java 完全不必这样去做，它通过集成的面向对象的异常处理机制来对可能出现但未被处理的异常进行提示，从而更好的对运行时错误进行管理以防止系统崩溃。另外，Java 在编译时还可以捕获类型声明中的许多常见错误，从而防止动态运行时不匹配问题的出现。

在 Java 中，限制对指针的使用也是保证程序安全可靠的重要原因之一。Java 规定一切

对内存的访问都必须通过对象的实例变量来实现，实例变量中的内容其实就是指针，只不过 Java 不允许程序员对其进行删除和计算，这样一来就可以避免从外部在未授权的情况下访问对象的私有成员，也可以避免由于对指针的错误操作而引起的不必要的麻烦。

下载小应用程序并能确保它对客户机的安全性不会造成危害也是 Java 安全可靠的一个重要的方面。也就是说，每当使用一个兼容 Java 的 Web 浏览器时，你可以安全地下载 Java 小应用程序。这是因为，Java 将小应用程序限制在 Java 的运行环境中，不允许它访问客户机的其他部分，就像是在网络应用程序和计算机之间提供了一道防火墙，从而不必担心病毒的感染或其他恶意的企图了。

4 / 多线程

Java 的设计目标之一是为了满足人们能够在互联网上创建运行平稳的交互式程序的需要，这就要求 Java 编写的应用程序能够同时执行多个任务。为了更好地管理多任务，Java 提供了对多线程编程的内嵌支持。多线程使应用程序能够并行执行，而且同步机制保证了对共享数据的正确操作。通过使用多线程，程序设计人员可以分别用不同的线程完成特定的功能，而不需要采用全局的事件循环机制，这样就能很容易地实现网络上的实时交互功能。而且，多线程还能够最大限度地利用 CPU 资源。这在用 Java 操作交互的、网络环境下的应用程序时也显得非常重要。

5 / 体系结构中立和可移植性

程序员面临的一个主要问题是，不能保证今天编写的程序明天能否在同一台机器上顺利运行，更不能保证在这台机器上编写的程序能否在其他机器上顺利运行。这是因为操作系统升级、处理器升级以及核心系统资源的变化，都可能导致程序无法继续运行。Java 通过虚拟机的实现来保证 Java 语言编写的程序对每种计算机都一样的运行。比如，简单的类型都是不变的：整数总是 32 位，长整数总是 64 位。令人奇怪的是，诸如 C 及 C++等主流的编程语言却不是这样。由于这些语言定义如此自由，每种编译器及开发环境便各有不同了，这使程序的移植成为非常棘手的问题。Java 程序的移植却很容易，而且不需要进行重新编译。这是因为 Java 能够通过自身的编译器将源程序生成与体系结构无关的字节码指令，只要安装了 Java 虚拟机，这种字节码指令就可以根据对应于 Java 虚拟机中的表示，转换成本地的机器码，然后就可以在任意的平台上运行了。

与平台无关的特性使 Java 程序可以方便地被移植到网络上的不同机器。同时，Java 的类库中也实现了与不同平台的接口，使这些类库可以自由地移植。可以这么说，Java 真正实现了“一次编译、到处运行”的设计目标，使程序的可移植性从梦想变成了现实。

6 / 解释性和高性能

当运行 Java 程序时，它首先被编译成字节码，然后通过 Java 解释器直接对 Java 字节码进行解释执行。虽然这个过程是解释执行的，但是由于字节码的设计并不专门针对任何一种特定的机器，而是非常类似于机器指令的指令编码。因而通过 Java 的虚拟机就可以很

容易的直接将字节码转换成对应于特定 CPU 的机器码，从而得到较高的性能。这种转换的效率比其他解释性语言如 Basic、Perl 等要高得多，甚至在非常低档的 CPU 上也能顺利运行。不过 Java 毕竟是解释性的语言，它解释执行的意义在于能够实现程序一经编译便可在众多不同的计算机上执行的跨平台运行。虽然这比 C 程序慢了许多，但在大多数应用中都是可以接受的。而且，现在 Java 也已经有了专门的代码生成器，可以很容易使用 JIT 编译技术将字节码直接转换成高性能的本机代码。值得一提的是，Java 运行时系统在提供这个特性的同时仍具有平台独立性，因而“高效且跨平台”对 Java 来说不再矛盾。

7/ 分布式

能够在 Internet 的分布式环境中运行 Java 程序也是 Java 的设计目标之一。特别是它提供了用于处理 TCP/IP 协议的专用类库，这样用户就可以通过 URI 地址存取资源，并且还可以很方便的在网络上访问其他对象。

8/ 动态性

Java 从一开始就被设计成动态地、可扩展的面向对象程序设计语言，这一优势可使它适合于一个不断发展的环境。程序员可以在类库中自由地加入新的方法和实例变量而不会影响用户程序的执行。并且 Java 通过接口来支持多重继承，使之比严格的类继承具有更灵活的方式和扩展性。此外，Java 的动态性特征也为小应用程序在运行时可以动态地更新提供了保障。

由于 Java 和 C++之间的相似性，容易使人们将 Java 简单地想象为“C++的增强版本”。但其实这是一种误解。Java 在各个方面都与 C++有重要的不同。Java 的出现也并不是用来取代 C++的，设计 Java 是为了解决某些特定的问题，而设计 C++是为了解决另外一类完全不同的问题。两者将长时间共存。那么 Java 和 C++都有哪些区别呢？下面仅列出一些比较明显的区别：

1/ 数据类型及类

Java 是完全面向对象的语言，所有函数和变量必须是类的一部分。除了基本数据类型之外，其余的都作为类对象，包括数组和字符串。对象将数据和方法结合起来，把它们封装在类中，这样每个对象都可拥有自己的属性和行为。而 C++允许将函数和变量定义为全局的。此外，Java 中取消了 C/C++中的结构体和联合，消除了不必要的麻烦。

2/ 类型转换

Java 和 C/C++都具有自动类型转换的机制，但是在强制类型转换方面，则有所不同。例如，在 C++中可以隐含的将浮点数赋予整型变量，通过去掉其尾数来实现。Java 则不支持这种自动强制类型转换，如果需要，必须由程序显式进行强制类型转换。

3/ 指针

Java 语言让程序设计人员无法直接访问和修改指针，而且通过自动内存管理功能有效地防止类似于 C/C++语言中对指针的误操作。但是这并不是说 Java 没有指针，而是这种指针只能供 Java 虚拟机内部使用，而不能由程序员使用罢了，这些限制都大大增强了 Java

程序的安全性。

4/ 预处理功能

C/C++在编译过程中都有一个预编译阶段。预处理阶段可以为开发人员提供方便，但增加了编译的复杂性。但是 Java 不支持预处理功能，Java 虚拟机通过提供引入语句(import)来实现和 C/C++的预编译功能相似的作用。

5/ 多重继承

C++支持多重继承，它允许一个类继承多个父类。尽管多重继承功能很强，但使用复杂，而且会引起许多麻烦，编译程序实现它也很不容易。Java 为了避免这些问题，取消了对多重继承的支持，为了平衡对语言本身造成的损失，Java 通过允许一个类可以实现多个接口的方式来达到类似于 C++多重继承的功能。

6/ 自动内存管理

Java 程序中所有的对象都是用 new 操作符建立在内存的堆中，这个操作符类似于 C++的 new 操作符，但是对无用对象的删除方式却是完全不同的。在 Java 中，当一个对象不再被使用时，不用程序员自己来进行删除，Java 会自动进行无用内存回收操作，这个操作是以线程的方式在后台运行的。而 C++中必须由程序员自己编写代码来释放内存资源，这显然会增加程序设计人员的负担。

7/ 字符串

在 C/C++中不支持字符串变量，Java 中虽然有字符串类型，但是是以对象的形式存在的。这样做有许多好处，比如：我们在整个系统中对建立字符串和访问字符串元素的方法时一致的；以类对象的方式存在，可以在程序执行时进行有效的检查，并有助于排除运行时错误的发生；对字符串可以通过 "+" 进行连接操作，从而便于字符串和非字符串之间的合并。

8/ 异常

Java 提供了强大的异常处理机制用于捕获程序运行时错误，但是却不需要程序员编写非常复杂的代码。因为，几乎程序中所有可能遇到的异常都被 Java 自动地进行处理，我们只需要在可能发生异常的代码前后用 try...catch 语句声明就可以了，显然这种处理工作增强了系统的容错能力。而 C++则没有如此方便的针对异常的处理机制。

9/ 操作符重载

Java 不支持操作符重载。操作符重载被认为是 C++的突出特征。在 Java 中，虽然类大体上可以实现这样的功能，但操作符重载的方便性仍然丢失了不少。Java 不支持操作符重载是为了保持 Java 尽可能简单。

无论怎样，Java 和 C++都是面向对象语言。也就是说，它们都能够实现面向对象的思想（封装、继承和多态）。由于 C++为了照顾大量的 C 语言使用者而兼容了 C，使得自身仅仅成为了带类的 C 语言，多多少少影响了其面向对象的彻底性。Java 则是完全的面向对

象语言，它语法更清晰，规模更小，更易学。它是在对多种程序设计语言进行了深入细致研究的基础上出现的，不但摒弃了其他语言的不足之处，而且从根本上解决了C++语言中所固有的缺陷。

可以这么说，环境的变化促使Java这种独立于平台的语言注定成为Internet上的分布式编程语言。同时，Java也改变了人们的编程方式，特别是Java对C++使用的面向对象范例进行的增强和完善。所以，Java不是孤立存在的一种语言，而是计算机语言多年来的演变结果。仅这个事实就足以证明Java在计算机语言历史上的地位。Java对Internet编程的影响就如同C语言对系统编程的影响一样，因而有人预言："Java语言的出现，将会引起一场软件革命"一点也不为过。

0.2 如何安装JDK，它和JRE有什么区别

Sun公司提供了自己的Java开发环境，通常称之为JDK(Java Development Kit)。随着时间的推移和技术的进步，Java在继续向前发展，JDK的版本也在不断地升级，从最初的JDK1.0版本到目前最新的JDK5.0版本。每一次升级都会带来新的技术和更加丰富的类库。其中，JDK1.2版本标志着Java语言成熟的开始，为了和以前旧版本进行区别，这之后发布的JDK都称为Java 2，并正式命名为J2SDK(Java2 Software Development Kit)。Java 2支持更多的新特性，如：增加了Swing和集合框架，添加了关键字assert，新的基于通道的IO子系统，修改了网络类和线程类，同时也增强了Java虚拟机和各种编程工具等。当然，在新发布的Java 2,版本5.0中，又增加了新的功能，

Sun公司提供了多种操作系统下的JDK下载，各种操作系统下的JDK的各种版本在使用上基本相似，我们可以根据自己的需要从Sun公司的官方网站(http://java.sun.com)上下载一个JDK的最新版本，如图0.2.1所示。

或许你在下载的时候曾经感到疑惑，Sun公司的官方网站上怎么会提供那么多可以下载的链接呢？是不是网页中提供的每个开发套件都要下载才可以呢？当然不用，"NetBeans IDE+JDK5.0"是Sun公司将自己提供的IDE工具和JDK开发包捆绑在一起供程序员使用的，下载这个套件就可以同时拥有Java的集成开发工具和开发包。读者暂不用去考虑下载这组套件。因为一开始就使用IDE工具来进行Java程序的开发并不是一件很轻松的事情，反而会增加初学者学习的困难。建议读者仅下载JDK5.0这个开发工具包，如果你用的操作系统是Windows，那么选择"Download JDK for Windows"就可以了。

此时，想必读者仍然会提出新的问题，因为读者从下载页面中还可以看到在JDK5.0下载套件下面，Sun公司还提供了一个JRE5.0的下载套件，这个套件是干什么用的呢？是不是它们都可以作为Java的开发包供程序员使用呢？这两者一定有所不同，否则Sun公司不会将两者区分开来。那么JDK和JRE两者到底有什么区别呢？

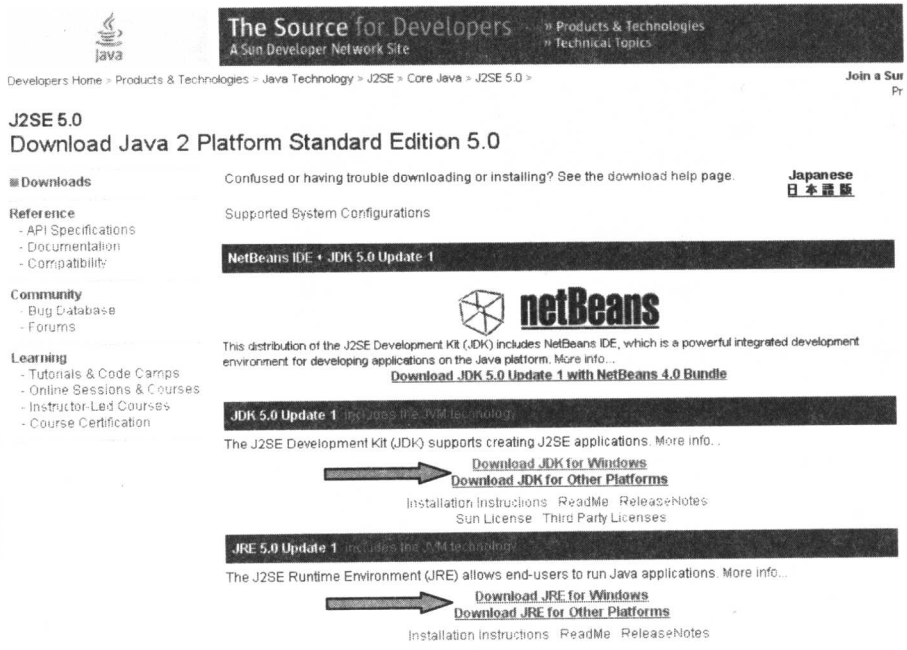

图 0.2.1　JDK 下载界面

其实，如果大家对 Flash 比较熟悉的话，就能很快明白它们的区别了。比如在 Flash 中，我们用 Flash 软件包设计一个 Flash 程序，这个软件包就类似于 Java 中的 JDK。如果需要在网页中观看 Flash 作品，则必须在浏览器中加上 Flash 的插件，浏览器和这个插件就类似于 JRE。

简单的说，JDK 就是面向开发人员使用的软件开发工具包，它提供了 Java 的开发环境和运行环境。如果你下载并完全安装了 JDK，那么你不仅可以开发 Java 程序，也同时拥有了运行 Java 程序的平台。JRE(Java Runtime Enviroment)是指 Java 的运行环境，是面向 Java 程序的使用者，而不是开发者。如果你仅下载并安装了 JRE，那么你的系统只能运行 Java 程序。

下面为读者演示 JDK5.0 的安装过程。

首先运行从 Sun 公司官方网站上下载下来的 JDK5.0 程序压缩包，如图 0.2.2 所示。

接着，选择接受使用许可书中的条款，并选择 Next 进入下一步，如图 0.2.3 所示。

在定制安装(Custom Setup)界面中，我们可以设置需要安装的组件，这里面包含了开发工具包“Development Tools”、演示程序“Demos”、API 库的源代码“Source Code”、以及包含了标准类库的运行时环境“Public JRE”4 个组件。其中，除了“Development Tools”是必选的组件之外，其他 3 个都是可选的。此外，还可以在这个安装界面中设置 Java 开发

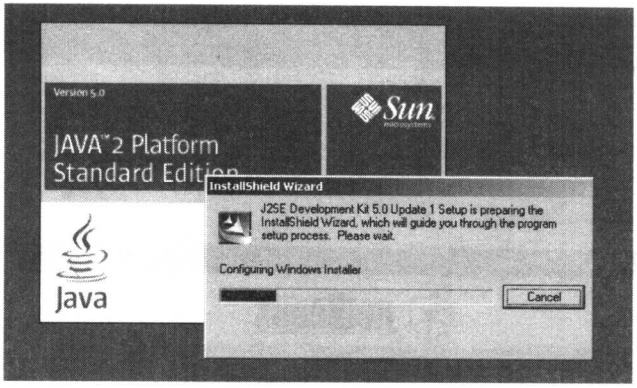

图 0.2.2 安装界面

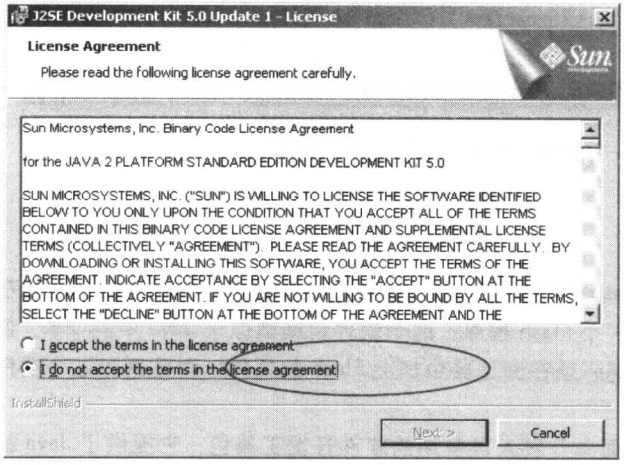

图 0.2.3 许可书条款

工具包安装到什么位置，默认情况下会安装到 C:\Program Files\Java\jdk1.5.0_01\目录下，如图 0.2.4 所示。为了便于将来使用开发工具包中的命令，Java 的初学者可以修改这个路径，使其简单而且直接，如：安装路径可以设置为 C:\JDK。需要注意的是，如果读者使用的是 JDK1.5 以前的版本，那么默认安装路径会和 JDK1.5 有所不同，如：在安装 JDK1.4.0 工具包时的默认安装路径就是 C:\jdk1.4.0\。

接着，安装程序会开始安装我们设置好的组件，在安装到 JRE 时会提示如下界面，如图 0.2.5 所示。这个界面在 JDK1.5.0 以前的版本中是没有的，JDK 的最新版在 JRE 中加入了对欧洲语言的更多的支持，正如在下图中看到的那样。此外，JRE 的安装路径也是可以修改的，通常选择默认设置。