



中等职业学校电子信息类教材 计算机技术专业

Java 编程案例教程

向传杰 主编



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

本书配有电子教学参考资料包

<http://www.phei.com.cn>

中等职业学校电子信息类教材（计算机技术专业）

Java 编程案例教程

向传杰 主编

电子工业出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书将 Java 的主要知识点压缩到很短的章节中,用通俗易懂的文字讲解 Java 的编程思路和基本语法。

本书的基本内容涉及 Java 语言的基本语法、Jbuilder 9 的开发环境、Java 图形用户界面编程、Java 多线程编程、Java 小程序编程、数组流和文件编程、Java 图形开发实例和 Java 数据库编程。

本书浅显易懂,实例生动丰富,实战性强,贴近真实的项目开发,可作为中等职业学校和高等职业学校计算机技术专业的教材,也可作为 Java 软件项目开发人员的参考书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

Java 编程案例教程/向传杰主编. —北京:电子工业出版社, 2004.5

中等职业学校电子信息类教材. 计算机技术专业

ISBN 7-5053-9828-8

I. J… II. 向… III. Java 语言—程序设计—专业学校—教材 IV. TP312

中国版本图书馆 CIP 数据核字(2004)第 030761 号

责任编辑:朱怀永

印 刷:北京京科印刷有限公司

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

经 销:各地新华书店

开 本:787×1092 1/16 印张:13.5 字数:337.6 千字

印 次:2004 年 5 月第 1 次印刷

印 数:5 000 册 定价:17.00 元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。
联系电话:(010) 68279077。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。

前言



Java 语言发展很快,在很短的时间就流行起来了,成为目前 Web 以及信息系统开发的主力军。大多的 Java 书籍理论性都比较强,不适合于具有中级水平的读者。本书的编写力图将 Java 基本理论用浅显的语言表达,并将其溶入到一个个完整的 Java 项目中去。全书通过介绍 Java 的基本语法、JBuilder 9 开发环境以及一系列完整的 Java 系统的开发过程,由浅入深介绍了用 Java 语言实现多个程序系统。

本书提供了书中介绍的 Java 例子程序所有的源代码。读者可以在自己的机器上运行,参照本书各个章节熟悉各个程序的开发和使用,也可以在源代码的基础上进一步改进。本书源代码丰富,可以作为读者开发 Java 程序的基础代码,在此基础上修改即可实现自己开发信息系统。

本书理论讲解浅显易懂,Java 程序功能丰富,通过阅读本书和实际操作本书所提供的程序代码,读者可以迅速掌握开发 Java 程序的基本步骤和开发技巧,提高软件开发实战经验。

本书由向传杰和续爽执笔编写,所有程序代码由向传杰开发和调试。向传杰同志毕业于北京理工大学信息技术学院计算机科学与工程系,具有丰富的 Java 开发、ERP、电子商务、电子政务以及企业信息化项目的开发和管理经验。此外,杨勇教授、张勇高级工程师等对本书的编写思路调整和资料整理给予了作者很大的帮助。在此,对他们表示衷心的感谢。

为了便于教学,本书还配有教学指南、电子教案及习题答案(电子版),有需要的教师请与电子工业出版社联系,我们将免费提供。E-mail: ve@phei.com.cn

由于时间仓促,再加上作者编写水平有限,书中难免会有一些缺陷和不足之处,希望广大读者不吝赐教并提出宝贵意见。

编 者
2004 年 1 月





第1章 Java 基础	(1)
1.1 Java 语言介绍	(1)
1.1.1 Java 语言的背景和前景	(1)
1.1.2 Java 语言的特点	(2)
1.1.3 Java 虚拟机	(4)
1.1.4 垃圾收集	(4)
1.2 Java 程序结构	(4)
1.3 Java 编程规则	(6)
1.4 运行环境	(7)
本章小结	(7)
习题 1	(8)
第2章 Java 语言的基本语法	(9)
2.1 基本数据类型、标识符及关键字	(9)
2.1.1 Java 基本数据类型	(9)
2.1.2 Java 语言的标识符	(11)
2.1.3 关键字	(11)
2.2 表达式和运算符	(11)
2.2.1 赋值运算符	(11)
2.2.2 算术运算符	(12)
2.2.3 关系运算符与逻辑运算符	(13)
2.3 流程控制	(13)
2.3.1 条件控制语句	(13)
2.3.2 循环控制语句	(15)
2.4 Java 编码约定	(16)
本章小结	(17)
习题 2	(17)
第3章 JBuilder 9 开发环境	(18)
3.1 JBuilder 简介	(18)
3.2 JBuilder 9 的安装	(19)
3.2.1 安装环境	(19)
3.2.2 安装步骤	(19)

3.3 JBuilder9 的开发环境介绍	(24)
3.3.1 菜单	(25)
3.3.2 工程窗口	(30)
3.3.3 文件窗口	(31)
3.3.4 内容窗口	(31)
3.3.5 工具条	(34)
3.3.6 结构窗口	(34)
3.3.7 消息窗口	(35)
3.4 用 JBuilder9 开发一个 Java 程序	(35)
3.4.1 用 JBuilder9 调试 Java 程序的流程	(35)
3.4.2 程序实例	(37)
本章小结	(43)
习题 3	(43)
第 4 章 Java 图形用户界面编程	(44)
4.1 Java 图形用户界面编程知识	(44)
4.2 图形用户界面组件编程实例	(44)
4.2.1 Button (按钮)	(44)
4.2.2 Label (标签)	(49)
4.2.3 Checkbox (复选框)	(51)
4.2.4 CheckboxGroup (单选框)	(55)
4.2.5 Choice (下拉列表框)	(58)
4.2.6 List (列表) / Label (标签)	(60)
4.2.7 MenuBar (菜单条)	(63)
4.2.8 PopupMenu (弹出式菜单)	(65)
4.2.9 ScrollBar (滚动条)	(68)
4.2.10 TextField (单行文本) / TextArea (文本区域)	(70)
4.2.11 ProgressBar (进度条) / Password Field (密码文本框)	(72)
4.3 一个科学计算器系统的开发	(74)
本章小结	(82)
习题 4	(82)
第 5 章 Java 多线程编程	(83)
5.1 多线程基本理论	(83)
5.1.1 多线程的定义	(83)
5.1.2 线程的组成	(83)
5.2 Java 编程中的线程	(84)
5.2.1 Java 线程的模型	(84)
5.2.2 创建线程类	(84)
5.3 线程的基本控制	(86)
5.3.1 线程的调度	(86)

5.3.2 线程的控制	(88)
5.4 Java 线程之间的同步	(89)
5.5 Java 线程之间的通信	(92)
5.6 Java 线程编程实例	(93)
本章小结	(98)
习题 5	(98)
第 6 章 Java 小程序编程	(99)
6.1 Applet 技术	(99)
6.2 JBuilder 9 开发 Applet	(100)
6.2.1 基本开发步骤	(100)
6.2.2 程序结构分析	(104)
6.3 Applet 程序实例	(106)
本章小结	(110)
习题 6	(110)
第 7 章 数组	(111)
7.1 数组的创建	(111)
7.1.1 数组的声明	(111)
7.1.2 创建数组	(111)
7.1.3 数组的初始化	(112)
7.2 多维数组	(113)
7.3 数组的拷贝	(114)
7.4 数组编程实例	(114)
7.5 Vector 类	(119)
7.5.1 Vector 类介绍	(119)
7.5.2 Vector 类编程实例	(120)
本章小结	(124)
习题 7	(124)
第 8 章 流、网络 and 文件编程	(125)
8.1 流式 I/O	(125)
8.1.1 流的基础知识	(125)
8.1.2 InputStream 方法	(125)
8.1.3 OutputStream 方法	(126)
8.1.4 基本的流类	(127)
8.2 Java 网络编程	(127)
8.2.1 Java 的网络编程基础	(128)
8.2.2 一个 TCP/IP 通信的实例	(128)
8.2.3 一个 UDP 通信的实例	(135)
8.3 Java 文件操作	(142)

8.3.1 创建一个新的 File 对象	(142)
8.3.2 随机存取文件	(142)
8.3.3 串行化 (Serializable)	(143)
8.3.4 简单文本文件操作实例	(143)
8.3.5 对象序列化文件存取实例	(148)
本章小结	(157)
习题 8	(157)
第 9 章 Java 图形开发实例	(158)
9.1 一个简单的图元操作系统开发的说明	(158)
9.2 系统的开发步骤	(160)
9.3 Frame1 程序说明	(167)
9.4 系统的发布	(184)
本章小结	(187)
习题 9	(187)
第 10 章 Java 数据库编程	(188)
10.1 Java 数据库编程基础	(188)
10.1.1 JDBC API	(189)
10.1.2 Java 操作数据库的基本过程	(191)
10.1.3 常用数据库的 Java 链接基本操作	(192)
10.2 Java 数据库编程的实例	(194)
10.3 JBuilder 数据库应用程序开发	(199)
10.4 SQL	(204)
10.4.1 SQL 数据表的操作	(205)
10.4.2 SQL 数据表的数据操作	(205)
本章小结	(206)
习题 10	(206)
参考文献	(207)

第1章 Java 基础



本章要点:

- ◆ Java 语言介绍
- ◆ Java 程序结构
- ◆ Java 语言的标识符
- ◆ Java 语言的关键字
- ◆ Java 语言的数据类型
- ◆ Java 编程的规则
- ◆ 运行环境

1.1 Java 语言介绍

1.1.1 Java 语言的背景和前景

1991 年, 美国 Sun 公司的一个叫 Green 的项目组为家用消费类电器产品开发了一个分布式代码系统。这个系统就是想把电子邮件(E-mail)发给家里的电冰箱、电视机等家用电器, 对它们进行远程控制和信息交流, 比如在出差的时候也可以开关家里的电视机。开始, 他们准备采用 C++ 进行开发, 但 C++ 太复杂, 安全性也差, 最后决定使用基于 C++ 开发一种新的语言——Oak, 这就是 Java 的前身。Oak 是一种用于网络的精巧而安全的语言, Sun 公司曾依此投标一个交互式电视项目, 但结果失败了。恰巧这时 Mark Ardreesen 开发的 Mosaic 和 Netscape 启发了 Oak 项目组成员, 他们用 Java 语言编制了 HotJava 浏览器, 得到了 Sun 公司首席执行官 (CEO) Scott McNealy 的支持, 触发了 Java 进军 Internet。网络的出现和迅速发展改变了 Java 的命运, 使它逐渐成为 Internet 上最受欢迎的开发和编程语言。

在 Java 出现以前, Internet 上的信息内容都是一些乏味死板的 HTML 文件。这对于那些迷恋于 Web 浏览的人们来说简直不可容忍。他们迫切希望能在 Web 中看到一些交互式的内容, 开发人员也极希望能够在 Web 上创建一类无需考虑软硬件平台就可以执行的应用程序, 当然这些程序还要有极大的安全保障。对于用户的这种要求, 传统的编程语言显得无能为力, 而 Sun 的工程师们敏锐地察觉到了这一点, 他们开始将 Java 技术应用于 Web 上, 并且开发出了 HotJava 的第一个版本。当 Sun 公司 1995 年正式以 Java 这个名字推出的时候, 几乎所有的 Web 开发人员都想到: 噢, 这正是我想要的。于是 Java 成了一颗耀眼的明星, 丑小鸭一下子变成了白天鹅。有人预言: Java 语言将是网络上的“世界语”, 今后所有的用其他语言开发的软件统统都要用 Java 语言来改写。

Java 语言是新一代面向对象程序的设计语言, 它的应用前景非常好, 而且特别适合于



Internet 应用程序开发，它的平台无关性将直接威胁到 Windows 技术的垄断地位。Java 作为软件开发的一种革命性的技术，其地位已被确立。一些著名的计算机公司纷纷购买了 Java 语言的使用权，如 IBM, Netscape, Novell, Apple, DEC, SGI 等，以及最不情愿的 Microsoft；Java 已得到了工业界的认可，众多的软件开发商开始支持 Java 的软件产品；Intranet 正在成为企业信息系统最佳的解决方案，而其中 Java 将发挥不可替代的作用；世界上成千上万的软件开发人员都转向使用 Java 语言。Java 语言正在不断发展和完善，Sun 公司是其主要的发展推动者，较通用的编译环境有 JDK(Java Develop Kit)与 JWS(Java Workshop)。还有很多其他公司正在开发 Java 语言的编译器与集成环境，如 IBM 的 Visual age for Java 以及 Borland 公司的 JBuilder 等。Java 语言的正确性与效率都在不断提高，用户用 Java 编程和用 C++编程一样方便和高效。

Java 有着广泛的应用前景，包括所有面向对象的应用程序开发、计算过程的可视化的软件开发、动态画面图形图像开发、交互式操作的设计、Internet 的系统管理开发、Intranet 上的软件开发、数据库开发以及其他类型的程序开发等。

1.1.2 Java 语言的特点

Java 语言的特点到底如何呢？归纳总结，它主要具有如下这样一些特点：

1. 简单

Java 语言最初是为对家用电器进行集成控制而设计的一种语言，因此它必须简单明了。Java 的风格类似于 C++，因而 C++程序员对它会是非常熟悉的。从某种意义上讲，Java 语言是 C 及 C++语言的一个变种，因此，C++程序员可以很快掌握 Java 编程技术。Java 摒弃了 C++中容易引发程序错误的地方，如指针和内存管理。Java 提供了丰富的类库，使程序设计变得非常容易，可以很方便地建立自己的系统。

2. 与操作系统无关性

与操作系统无关性是指 Java 能运行于不同的操作系统。Java 引进虚拟机 (JVM) 原理，并运行于虚拟机，实现 Java 在不同平台上的运行。使用 Java 编写的程序能在世界范围内共享。Java 的数据类型与机器无关，Java 虚拟机是建立在硬件和操作系统之上，实现 Java 二进制代码的解释执行功能，提供于不同平台的接口上的。与体系结构无关的特性使得 Java 应用程序可以在配备了 Java 解释器和运行环境的任何计算机系统上运行，这成为 Java 应用软件便于移植的良好基础。但仅仅如此还不够。如果基本数据类型设计依赖于具体实现，也将为程序的移植带来很大不便。通过定义独立于平台的基本数据类型及其运算，Java 数据得以在任何硬件平台上保持一致。

3. 安全性

Java 的安全性可从两个方面得到保证：一方面，在 Java 语言里，像指针和释放内存等 C++功能被删除，避免了非法内存操作；另一方面，当 Java 用来创建浏览器时，语言功能和一些浏览器本身提供的功能结合起来，使它更安全。Java 语言在机器上执行之前，必须要经过很多次的测试。Java 语言要经过代码校验，检查代码段的格式，检测指针操作、对象操作是否有效。Java 的编程类似于 C++，但是 Java 舍弃了 C++的指针对存储器地址的直接操作，程序运行时，内存由操作系统分配，这样可以避免病毒通过指针侵入系统。Java 对程序提供了安全管理器，防止程序的非法访问。



4. 面向对象

面向对象可以说是 Java 最重要的特性。Java 语言的设计完全是面向对象的，它不支持类似 C 语言那样的面向过程的程序设计技术，Java 支持静态和动态的代码继承和重用。单从面向对象的特性来看，Java 类似于 SmallTalk，但其他特性尤其是适用于分布式计算环境的特性远远超越了 SmallTalk。Java 吸取了 C++ 面向对象的概念。将数据封装于类中，利用类的优点，实现了程序的简洁性和便于维护性。类的封装性、继承性等有关对象的特性，使程序代码只需一次编译，然后通过上述特性反复利用。程序员只需把主要精力用在类和接口的设计和应用上。Java 提供了众多的一般对象的类，通过继承即可使用父类的方法。在 Java 中，类的继承关系是单一的、非多重的，一个子类只有一个父类，子类的父类又有一个父类。Java 提供的 Object 类及其子类的继承关系如同一棵倒立的树形，根类为 Object 类，Object 类功能强大，经常会使用到它及其他派生的子类。

5. 分布式

Java 包括一个支持 HTTP 和 FTP 等基于 TCP/IP 协议的子库。因此，Java 应用程序可凭借 URL 打开并访问网络上的对象，其访问方式与访问本地文件系统几乎完全相同。为分布环境尤其是 Internet 提供动态内容无疑是一项非常宏伟的任务，但 Java 的语法特性却使我们很容易地实现这项目标。Java 建立在扩展 TCP/IP 网络平台上，库函数提供了用 HTTP 和 FTP 协议传送和接受信息的方法。这使得程序员使用网络上的文件和使用本机文件一样容易。

6. 健壮性

Java 致力检查程序在编译和运行时的错误。利用类型检查帮助可以查出许多早期开发出现的错误。Java 自己操作内存，减少了内存出错的可能性。Java 还实现了真数组，避免了覆盖数据的可能性。Java 提供：Null 指针检测、数组边界检测、异常出口、Byte code 校验。这些功能大大提高了开发 Java 应用程序的周期。

7. 解释性

Java 解释器(运行系统)能直接运行目标代码指令。链接程序通常比编译程序所需资源少，所以程序员可以在创建源程序上花上更多的时间。

8. 高性能

如果解释器速度不慢，Java 可以在运行时直接将目标代码翻译成机器指令。Sun 用直接解释器，一秒钟内可调用 300 000 个过程。翻译目标代码的速度与 C 和 C++ 的性能没什么区别。

9. 多线程

Java 提供的多线程功能使得在一个程序里可同时执行多个小任务。线程是一个大进程里分出来的小的独立的进程。因为 Java 实现了多线程技术，所以比 C 和 C++ 更健壮。多线程带来的更大的好处是更好的交互性能和实时控制性能。当然实时控制性能还取决于系统本身，如 UNIX, Windows, Macintosh 等，在开发难易程度和性能上都比单线程要好。任何用过当前浏览器的人，都感觉为调一幅图片而等待是一件很烦恼的事情。在 Java 里，你可用一个单线程来调一幅图片，同时，你可以访问 HTML 里的其他信息而不必等待。

10. 动态

Java 的动态特性是其面向对象设计方法的展现，它支持在运行中动态装入类，这是 C++



语言进行面向对象程序设计所无法实现的。在 C++ 程序设计过程中，每当在类中增加一个实例变量或一种成员函数后，引用该类的所有子类都必须重新编译，否则将导致程序崩溃。Java 从如下几方面采取措施来解决这个问题：Java 编译器将符号引用信息在字节码中保存下来，并传递给解释器，再由解释器在完成动态连接类后，将符号引用信息转换为数值偏移量。这样，一个在存储器中生成的对象不在编译过程中决定，而是延迟到运行时由解释器确定。这样，对类中的变量和方法进行更新时就不至于影响现存的代码。解释执行字节码时，这种符号信息的查找和转换过程仅在新的名字出现时才进行一次，随后代码便可以全速执行。在运行时确定引用的好处是可以使用已被更新的类，而不必担心会影响原有的代码。如果程序连接了网络中另一系统中的某一类，该类的所有者也可以自由地对该类进行更新，而不会使任何引用该类的程序崩溃。Java 还简化了使用一个升级的或全新的协议的方法。如果你的系统运行 Java 程序时遇到了不知怎样处理的程序，Java 能自动下载你所需要的功能程序。

1.1.3 Java 虚拟机

Java 虚拟机是在真实机器中用软件模拟实现的一种想象机器。Java 虚拟机代码被存储在 .class 文件中，每个文件都包含最多一个 public 类。Java 虚拟运行机规范为不同的硬件平台提供了一种编译 Java 技术代码的规范，该规范使 Java 软件独立于平台。编译器在获取 Java 应用程序的源代码后，将其生成字节码，它是为 JVM 生成的一种机器码指令。每个 Java 解释器，不管它是 Java 技术开发工具，还是可运行 Applet 的 Web 浏览器，都可执行 JVM。

1.1.4 垃圾收集

许多编程语言都允许在程序运行时动态分配存储器，分配存储器的过程由于语言句法不同而有所变化，但总是要将指针返回到存储区的起始位置，当分配存储器不再需要时（存储器指针已溢出范围），程序或运行环境应取消分配存储器。在 C，C++ 或其他语言中，程序员负责取消分配存储器。有时，这是一件很困难的事情。因为程序员并不总是事先知道存储器应在何时被释放。当在系统中没有能够被分配的存储器时，可导致程序瘫痪，这种程序被称作具有存储器漏洞。

Java 编程语言解除了程序员取消分配存储器的责任，它可提供一种系统级线程以跟踪每一存储器的分配情况。Java 编程语言不为程序员提供自己管理内存的功能，因为 Java 运行环境会自动收回那些没有引用与之相连的对象所占的内存。这种内存回收的过程就叫垃圾回收。垃圾回收包括：

- ① 为每一个对象保存与之相连的所有引用的数目；
- ② 定期回收那些引用数目为零的对象所占的内存。

在 Java 虚拟机的空闲周期，垃圾收集线程检查并释放那些可被释放的存储器。垃圾收集在 Java 技术程序的生命周期中自动进行，它解除了取消分配存储器的要求，并避免了存储器漏洞。

1.2 Java 程序结构

Java 语言的源程序代码由一个或多个编译单元（compilation unit）组成，每个编译单元只能包含下列内容（空格和注释除外）：



- ① 一个程序包语句 (package statement)。
- ② 引入语句 (import statements)。
- ③ 类的声明 (class declarations)。
- ④ 界面声明 (interface declarations)。

每个 Java 的编译单元可包含多个类或界面,但是每个编译单元最多只能有一个类或者界面是公共的。Java 的源程序码被编译后,便产生了 Java 字节代码。Java 的字节代码由一些不依赖于机器的指令组成,这些指令能被 Java 的运行系统(runtime system)有效地解释。Java 的运行系统工作起来如同一台虚拟机。在当前的 Java 实现中,每个编译单元就是一个以.java 为后缀的文件。每个编译单元有若干个类,编译后,每个类生成一个.class 文件。.class 文件是 Java 虚拟机能够识别的代码文件。

一个典型的 Java 程序结构如下:

```
//程序包说明
package nihao;
//引入语句
import java.awt.*;
//类的定义
public class Application1 {
    //主方法
    public static void main(String[] args) {
        System.out.println("我要好好学习 java 编程!");
    }
}
```

文件名 Application1.java 和类名 Application1 必须一致,编译完后生成一个.class 文件 Application1.class。

另外 Java 语言有三种类型,Java(TM) 支持三种类型的注释。

1. int x; // a comment 一行中"//"后的部分为注释内容

2. int x;

/*

The variable x is an integer;

*/

"/*" 和 "*/" 之间的所有内容为注释内容,编译器对此注释内容不做处理

3. int x;

/**

x -- an integer representing the x coordinate

*/

int x; "/*" 和 "*/" 之间的所有内容为注释内容,编译器对此注释内容不做处理,此种注释为文档工具 javadoc 生成文档时所用。



1.3 Java 编程规则

Java 代码编写的一般性指导规则:

(1) 类的设计

设计类对应使类尽可能短小精悍, 而且只解决一个特定的问题。下面是对类设计的一些建议: 数量众多的方法涉及到类型差别极大的操作, 考虑用几个类来分别实现; 许多成员变量在特征上有很大的差别, 考虑使用几个类。

(2) main()方法

对于自己创建的每一个类, 都考虑置入一个 `main()`, 其中包含了用于测试那个类的代码。为使用一个项目中的类, 没必要删除测试代码, 这些代码可作为如何使用类的一个示例使用。

(3) 方法的设计

应将方法设计成简要的功能性单元, 用它描述和实现一个不连续的类接口部分。理想情况下, 方法应简明扼要。若长度很大, 可考虑通过某种方式将其分割成较短的几个方法。这样做也便于类内代码的重复使用 (有些时候, 方法必须非常大, 但它们仍应只做同样的一件事情)。

(4) 私有化

让一切东西都尽可能地“私有化”——`private`。如果使库的某一部分“公共化”(一个方法、类或一个字段等等), 就永远不能把它拿出。若强行拿出, 就可能破坏现有的代码, 使它们不得不重新编写和设计。若只公布自己必须公布的, 就可放心大胆地改变其他任何方法。在多线程环境中, 隐私是特别重要的一个因素, 只有 `private` 字段才能在不同步使用的情况下受到保护。

(5) 内部类

任何时候只要发现类与类之间结合得非常紧密, 就需要考虑是否采用内部类, 从而改善编码及维护工作。

(6) 注释

尽可能细致地加上注释, 并用 `java doc` 注释文档语法生成自己的程序文档。

(7) 常数

避免使用“魔术数字”, 因为这些数字很难与代码很好地配合。如以后需要修改它, 无疑会成为一件很困难的事, 因为根本不知道“100”到底是指“数组大小”, 还是一个整数。所以, 我们应创建一个常数, 并为其定义使用时具有说服力的描述性名称, 并在整个程序中都采用常数标识符。这样可使程序更易理解以及更易维护。

(8) 数组

创建大小固定的对象集合时, 应将它们传输至一个数组 (若准备从一个方法里返回这个集合, 更应如此操作)。这样一来, 我们就可享受到数组在编译期进行类型检查的好处。

(9) interfaces 和 abstract 类

尽量使用 `interfaces`, 不要使用 `abstract` 类。若已知某样东西准备成为一个基础类, 那么第一个选择应是将其变成一个 `interfaces` (接口)。只有在不得不使用方法定义或者成员变量的时候, 才需要将



其变成一个 abstract (抽象) 类。接口主要描述了客户希望做什么事情, 而一个类则致力于 (或允许) 具体的实施细节。

(10) 优化

警惕“过早优化”。首先让程序运行起来, 再考虑变得更快, 但只有在自己必须这样做、而且经证实某部分代码中的确存在一个性能瓶颈的时候, 才应进行优化。除非用专门的工具分析瓶颈, 否则很有可能是在浪费自己的时间。性能提升的隐含代价是自己的代码变得难于理解, 而且难于维护。

(11) 阅读代码

阅读代码的时间比写代码的时间长得多。思路清晰的设计可获得易于理解的程序, 但注释、细致的解释以及一些示例往往具有不可估量的价值。无论对编程人员自己, 还是对看到代码的其他人, 它们都是相当重要的。

(12) 磨刀不误砍柴工

良好的设计能带来很大的回报。简言之, 对于一个特定的问题, 通常会花较长的时间才能找到一种最恰当的解决方案。但一旦找到了正确的方法, 以后的工作就轻松多了, 再也不用经历数小时、数天或者数月的痛苦寻找。由于自己倾注了大量心血, 最终获得一个出色的设计方案, 成功的快感也是令人心动的。坚持抵制草草完工的诱惑, 因为那样做往往得不偿失。

1.4 运行环境

Java 不仅提供了一个丰富的语言和运行环境, 而且还提供了一个免费的 Java 开发工具集 (Java Developers Kits, 简称 JDK)。编程人员和最终用户可以利用这个工具来开发 Java 程序或调用 Java 内容。JDK 包括以下工具:

(1) Java

Java 语言编译器, 输出结果为 Java 字节码。

(2) Java

Java 字节码解释器。

(3) Javap Disassembler

Java 字节码分解程序, 本程序返回 Java 程序的成员变量及方法等信息。

(4) Java Applet Viewer

小应用程序浏览工具, 用于测试并运行 Java 小应用程序。

我们介绍的 JBuilder 就是一个非常优秀的 Java 集成开发环境。



本章小结

本章介绍了 Java 编程的基础知识, 包括 Java 的历史和发展、Java 语言的程序结构、Java 的编程原则以及运行环境等。



习题 1

1. Java 语言的特点如何?
2. Java 语言的程序结构是什么?
3. 为什么 Java 比 C++安全?
4. Java 编程的原则有哪些?

第2章 Java 语言的基本语法

本章要点:

- ◆ 基本数据类型、标识符及关键字
- ◆ 表达式和运算符
- ◆ 流程控制
- ◆ Java 编码约定

2.1 基本数据类型、标识符及关键字

Java 语言的基本数据类型和 C 语言十分相似,基本数据类型是 Java 语言自身定义的数据类型,通常不允许用户修改。

2.1.1 Java 基本数据类型

Java 编程语言包括八个原始数据类型和一个特殊类型。原始类型可分为四种。

(1) 逻辑类 boolean

逻辑值有两种状态,即人们经常使用的“真”和“假”或“true”和“false”,这样的值是用 boolean 类型来表示的。boolean 有两个值,即“true”和“false”。以下是一个有关 boolean 类型变量的声明和初始化:

```
boolean truth = true; //声明一个逻辑类型变量 truth, 赋初值 true
```

(2) 文本类 char

使用 char 类型可表示单个字符。一个 char 代表一个单个字符。一个 char 变量必须包含在单撇号(')内。如, 'a'表示一个 char 字符。

例如,

```
char char1 = 'A'; // 申明并初始化一个字符变量 char1
```

```
char ch1, ch2; // 申明两个字符变量
```

(3) 整数类 byte, short, int, long

每个整数数据类型的范围见表 2.1 所示。

表 2.1 整数数据类型的范围

整数长度	类 型	范 围
8 位	byte	$-2^7 \dots 2^7 - 1$
16 位	short	$-2^{15} \dots 2^{15} - 1$