高等学校21世纪计算机教材

JAVA 程序设计

王萌　刘婧　来宾　编著

冶金工业出

JAVA

程序设计

[redacted]

高等学校21世纪计算机教材

JAVA程序设计

王萌　刘婧　来宾　编著

北　京

冶金工业出版社

2004

内 容 简 介

本书首先介绍了 Java 的基本知识和技能，从基本的数据类型和控制结构讲述，逐步过渡到方法、类与对象、继承机制和高级面向对象方法。并介绍了 Java 2 平台的一些高级特性，内容涉及字符串处理、Java Applet、图形用户界面设计、输入/输出处理、多媒体和多线程技术等。书中各章节都提供了丰富的例程，并且每一章均附有适量的综合测试题。另外，为了配合教学，书的最后一章还附加了实验指导。

本书内容丰富、条理清晰、文笔流畅，难易适当，适合于 Java 语言的初学者，可作为计算机相关专业程序设计的入门教材，同时也可供 Java 程序开发设计人员和编程爱好者参考。

图书在版编目（CIP）数据

JAVA 程序设计 / 王萌等编著. —北京：冶金工业出版社，2004.1
ISBN 7-5024-3411-9

I. J... II. 王... III. JAVA 语言—程序设计
IV. TP312

中国版本图书馆 CIP 数据核字（2003）第 111761 号

出版人　曹胜利（北京沙滩嵩祝院北巷 39 号，邮编 100009）
责任编辑　程志宏
湛江蓝星南华印务公司印刷；冶金工业出版社发行；各地新华书店经销
2004 年 1 月第 1 版，2004 年 1 月第 1 次印刷
787mm × 1092mm　1/16；　33 印张；　768 千字；　518 页；　1-5000 册
50.00 元
冶金工业出版社发行部　电话：（010）64044283　**传真：**（010）64027893
冶金书店　地址：北京东四西大街 46 号（100711）　**电话：**（010）65289081
（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、关于 Java 语言

Java 语言是 Sun 公司开发的一套程序设计语言，从推出到现在经过数个版本的更新，已经发展成为现在最重要的面向对象程序设计语言之一。Java 不仅是一种程序设计语言，它还是一个完善的开发平台，有丰富的编程接口（API）、成熟的平台机制。由于 Java 语言是完全面向对象的，因此它也越来越被人们当作面向对象程序设计的首选教学语言。Java 语言的开放性、成熟性和它的众多优点使它成为现代 IT 业界最重要的技术支柱之一。

二、关于本书

编程对于初学者来说是一个非常艰难的过程，因为程序设计应用的概念、技巧很多，涉及的知识面也很广泛。大量的实践经验证实，要为初学的读者解决这个问题，让其顺利地通过编程这扇大门，成长为一名优秀的编程技术人员，最重要一点就是从一开始就降低程序设计的门槛，使初学者不至于一进门就被排山倒海的概念、术语弄得头昏脑胀。学习 Java 编程也一样。本书的编写正是沿着“降低门槛”这个思路所做的尝试，尽量避免引入过多的概念和术语，而是采用类比举例的方式做形象化的阐述。并注重理论与实践的结合，在讲解理论知识之后，配合适量与实际生活联系紧密的例子帮助读者理解并巩固所学的理论知识。其中大多数例子都是完整的应用程序，读者可以自主地编译运行它们，在实践中加深对 Java 语言程序设计基本理论和思想的理解。

三、本书结构

本书共分 13 章，具体结构如下：

第 1 章：Java 语言概述。主要介绍了 Java 语言的发展、面向对象概念与方法及其开发运行环境，并针对不同水平的读者分别讲了 Java 程序初体验。

第 2 章：Java 语言基础。主要介绍了 Java 语言的标识符、保留字、数据类型、类型转换机制、运算符、表达式、包装类及其输入输出基本知识。

第 3 章：控制语句。总体介绍了流程控制的用途与分类、再分别讲述了分支控制语句、循环控制语句、跳转语句、Java 的方法及异常处理。

第 4 章：数组。主要介绍了数组的基本概念，一维数组、二维数组及其应用。

第 5 章：面向对象程序设计。主要介绍了类与对象、对象的生命周期、类的继承机制、接口和包的基础知识及应用。

第 6 章：字符串处理。主要介绍了 String 字符串、StringBuffer 字符串的定义与应用及 main 方法的参数。

第 7 章：输入/输出处理。主要介绍了输入/输出基础、文件的顺序访问、随机访问、并涉及目录及文件管理和其他常用流处理。

第 8 章：Java Applet。主要介绍了 Java Applet 的基础知识及其创建、执行和通信。

第 9 章：图形用户界面设计。主要介绍了 GUI 设计概述，用 AWT、Swing 创建图形用户界面，及文本与字体和图形设计的知识。

第 10 章：Java 多媒体技术。主要从图像、动画、声音等方面介绍 Java 的多媒体技术，并辅以实例说明。

第 11 章：多线程技术。主要介绍了多线程概述、多线程的实现与控制、线程的互斥与同步及线程间通信。

第 12 章：一个综合的例子——MiniEditor。主要介绍了 MiniEditor 的功能需求分析、基本设计思路、类划分及其具体实现。

第 13 章：Java 语言程序设计上机实验指导。主要介绍了 Java 语言实验机器与环境、JDK1.4.2 的安装及简介并辅以 7 个例子详细说明。

另外，本书最后还附有三个附录供读者参考：

附录 A，javadoc 的注释保留字；附录 B，Java 保留字；附录 C，流程图与算法的结构化描述。

四、本书特点

本书内容丰富，条理清晰，文笔流畅，难易适当，以循序渐进、深入浅出的方式引导读者逐步掌握 Java 程序设计精髓。书中注重程序设计理论与编程实践的结合，通过配合大量与实践生活紧密联系的例子和完整的应用程序使读者加深对 Java 语言程序设计基本理论和思想的理解。同时书中第 1~11 章都附有习题可以让读者边学边练，更好的掌握 Java 程序设计。

五、适用对象

本书适合 Java 语言的初学者，可作为计算机相关专业程序设计的入门教材，同时也可供 Java 程序开发设计人员和编程爱好者参考。

读者在学习本书过程中如有好的意见或建议，请与作者联系：**december.wang@263.net**，也可以登录网站 **http://www.cnbook.net**，在该网站的论坛进行探讨。

由于作者水平所限，时间仓促，书中缺点和错误在所难免，恳请广大读者批评指正。

编　者

2003 年 9 月

目　录

第 1 章　Java 语言概述

程序设计是计算机领域中最重要的分支之一，也是计算机相关专业的必修课程。程序设计语言的选择直接关系到软件系统开发的难度大小和软件产品品质的优劣。从计算机诞生开始就有许多的程序设计语言涌现，而经过多年实践的检验，只有为数不多的高级语言流行到现在，Java 语言就是其中的一种。那么 Java 语言到底是怎样的一门语言？它与计算机发展有些什么样的联系？它与其他的高级语言有些什么样的相同点和不同点？本章将试图简明扼要的回答这些问题。

本章首先回顾 Java 语言的发展历史，之后介绍 Java 语言的特点以及 Java 应用程序在 Java 平台下的工作机制。通过本章学习，读者将为阅读本书学习 Java 语言打下必要的基础。

1.1　Java 语言的发展

1.1.1　Java 语言的发展历程

Java 的起源有些偶然，为了使读者对 Java 的发展历史有一个较全面的认识，这里提一下 Sun 的一个开发项目——Green。这个项目是为具备简化处理芯片的家用消费电子产品（如 PDA,微波炉，洗衣机等）开发一个嵌入分布式系统，以对它们进行控制。同时，Green 项目的开发人员还希望开发出的嵌入式软件能够具有良好的可移植性，让同样的程序可以很方便地在不同的控制芯片上运行。

如图 1-1 所示是 Java 的 Logo，图 1-2 所示是 Java 平台吉祥物。

图 1-1　Java 的 Logo

图 1-2　Java 平台吉祥物—Duke

起初，项目组准备采用 C++，但后来发现 C++并不适合用来开发嵌入式系统，主要原因是 C++太复杂，安全性差。这归咎于 C++以直接引用方式引用系统资源，这不但带给系统与程序开发人员很大的负担，而且容易导致软件的执行发生错误。虽然程序的运行错误对于一般的软件系统来说最多只是造成系统的死机或者丢失数据，但是用于控制电子产品的嵌入式软件绝对不允许这样的错误产生，因为如果一个以软件控制的电烤箱，软件发生错误，其直接结果可能就是一场足以让人丧生的火灾。

此外，由于电子消费性产品种类繁多，所使用的控制芯片也不尽相同，所以一个很迫

切的问题是如何让程序开发人员开发出适合各式电子消费产品的软件。如果可以达到这个目标，就可以开发出相对独立于芯片的软件系统，当电子制造商要升级芯片的时候，程序开发人员不必重新设计同样的软件，因此可以避免很多无谓的重复劳动。这个跨平台的目标也是 C++无法胜任的。因为即便 C++可以在重新编译之后运行于不同的平台，但是却不得不为各种不同的芯片设计专属的 C++编译器，这也是相当费时费力的。

最后，Green 项目小组决定放弃使用 C++，重新开发一种适合开发跨平台嵌入式软件的语言。它就是 Java 的前身——Oak。在开发 Oak 时，项目开发人员舍弃了 C++那些可能造成系统错误的语言机制，包括如指针、运算符重载以及资源的直接引用，并且将内存管理的机制直接引入语言本身，使得开发人员不必为内存配置问题担心。Oak 被开发出来之后，Sun 公司曾依此投标一个交互式电视项目，但结果是被对手 SGI 打败。这个时候似乎 Oak 已经没什么发展前景了，恰巧 Mark Ardreesen 开发的 Mosaic 和 Netscape 启发了 Oak 项目组成员，他们用 Java 编制了 HotJava 浏览器。而当时刚好是互联网开始火热的时期，各厂商都开始体会到国际互联网所带来的无限商机，Green 项目的开发人员认为 Oak 的跨平台特性很适合开发网络上的应用程序，因此决定将 Oak 更名为 Java，并在 1996 年 1 月发布了第一个 Java 编译器。1995 年的 Java 还只是个编程语言，6 年之后，Java 发生了巨大的变化，已经发展成一个强大的开发平台——一个整合了编程语言、标准程序库和运行环境的完整平台（如图 1-3 和图 1-4 所示）。并且，Java 2 平台针对不同用户应用的需要，发布了三种版本：J2SE（Java 2 platform Standard Edition，标准版）、J2EE（Java 2 platform Enterprise Edition，企业版）和 J2ME（Java 2 platform Micro Edition，缩微版），分别用于桌面应用、企业级应用和移动设备应用的开发。

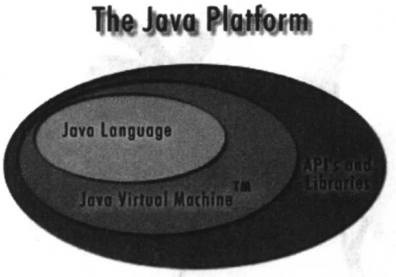

图 1-3 Java 已经发展成为一个开发平台

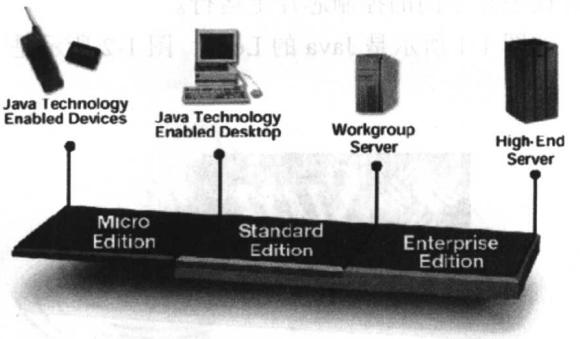

图 1-4 Java2 平台的三种版本

2001 年 2 月 5 日，Sun 公司在美国举行了 SUN ONE（Sun Open Net Enviorment，开放网络环境）的新闻发布会，更促进了 Java 的发展。SUN ONE 是 SUN 公司提供的适于开放的智能化 Web 服务的新一代架构，支持跨越多网络的系统，其中包括传统 Web、无线 Web 和家庭网络。SUN ONE 的宗旨是要保证可以使用任何工具开发智能 Web 服务，使之运行于任何平台无缝地实现交互操作。从本质上讲，SUN ONE 体系结构给出一些开放和流行的标准、技术和协议，Java 就是其中的核心技术。

1.1.2 Java 语言的特点

Java 是一种简单的、面向对象的、分布式的、健壮的、体系结构中立的、安全的、可

移植的、解释的、高性能的、多线程的、动态的语言。

1. 简单性

Java语言作为一种面向对象的语言，通过提供最基本的方法来完成指定的任务。用户只需要理解一些基本的概念，就可以用它编写出适合于各种情况的应用程序。Java略去了运算符重载、多重继承等模糊而且很少用到的概念，并且通过实现自动垃圾回收机制，大大简化了程序开发人员的内存管理工作。另外，Java也适合于在小型机上运行，它的基本解释器及类的支持只有40KB左右，加上标准类库和线程的支持也只有215KB左右。

Java语言的简单性主要体现在：

（1）Java的风格与C++十分相似，因此C++程序员可以很快掌握用Java做应用开发的技术。

（2）Java舍弃了C++中容易引发程序错误的机制，如指针和内存管理。

（3）Java提供了丰富的类库供开发人员使用。

2. 面向对象

Java被人们称为面向对象的语言，面向对象是什么含义呢？事实上，面向对象（OOP，Object-oriented Programming）是一种软件开发方式，通过用问题范畴中的元素与对象描述问题，而不是用计算机中执行的一系列步骤来描述。这样，好的设计就能得到复用、可扩展和可维护的组件。有关“面向对象”的详细内容请参考1.2节。

面向对象可以说是Java最重要的特性。Java语言的设计集中于对象及其接口，它提供了简单的类机制以及动态的接口模型。对象中封装了它的状态变量以及相应的方法，实现了模块化和信息隐藏；而类则提供了一类对象的原型，并且通过继承机制，子类可以使用父类所提供的方法，实现了代码的复用。

3. 分布式

分布式包括数据分布和操作分布。数据分布是指数据可以分散在网络的不同主机上，操作分布指把一个计算分散在不同主机上处理。

Java支持WWW客户机/服务器计算模式，因此，它支持这两种分布性。对于前者，Java提供了一个叫做URL的对象，利用这个对象，可以打开并访问具有相同URL地址上的对象，访问方式与访问本地文件系统相同；对于后者，Java的applet小程序可以从服务器下载到客户端，即部分计算在客户端进行，提高系统执行效率。

Java提供了一整套网络类库，开发人员可以利用类库进行网络程序设计，方便的实现Java的分布式特性。

4. 健壮性

健壮性反映程序的可靠性。Java的几个内置的特性使程序的可靠性得到改进：

（1）Java 是强类型语言。编译器和类载入器保证所有方法调用的正确性，防止隐式类版本不兼容性。

（2）Java没有指针，不可能引用内存指针，搞乱内存或数组越界访问。

（3）Java 进行自动内存回收，编程人员无法意外释放内存，不需要判断应该在何处释放内存。

（4）Java 在编译和运行时，都要对可能出现的问题进行检查，以消除错误的产生。

另外在编译的时候还可揭示出可能出现但尚未被处理的异常，以防止系统的崩溃。

5. 体系结构中立

体系结构中立性指 Java 的平台中立字节码（byte code）。Java 程序不是被编译成依附于平台的二进制码，而是字节码。只要有 Java 运行环境的机器都能执行这种字节码。目前，Java 运行环境已经有在 Solaris，Win32，Unix/Linux，MacOS 等系统上的移植版本。

这种与平台无关的特性使 Java 程序可以方便的被移植到网络中的不同机器上，而并不需要重新编译与连接。

6. 安全性

当 Java 用于网络、分布式环境下时就必须要注重安全性。Java 通过自己的安全机制防止了病毒程序的产生和下载程序对本地系统的威胁破坏。当 Java 字节码进入解释器时，首先必须经过字节码校验器的检查，然后 Java 解释器将决定程序中类的内存布局，随后类装载器负责把来自网络的类装载到单独的内存区域，避免应用程序之间相互干扰破坏。此外，客户端用户还可以限制从网络上装载的类只能访问某些文件系统。上述几种机制结合起来，使得 Java 成为安全的编程语言。

7. 平台无关性

Java 是平台无关的语言是指用 Java 写的应用程序不用修改就可在不同的软硬件平台上运行。平台无关有两种：源代码级和目标代码级。C 和 C++具有一定程度的源代码级平台无关，表明用 C 或 C++写的应用程序不用修改只需重新编译就可以在不同平台上运行。

Java 主要靠 Java 虚拟机（Java Virutal Machine，JVM）在目标代码级别上实现它的平台无关性。JVM 是一种虚拟机器，它被建立在具体操作系统之上，本身具有一套虚机器指令，并有自己的栈、寄存器组等等，JVM 通常是在软件层次上实现而不是在硬件层次上实现。（目前，Sun 公司已经设计实现了 Java 芯片，主要使用在网络计算机 NC 上。另外，Java 芯片的出现也会使 Java 更容易嵌入到家用电器中）。JVM 是 Java 平台无关的关键，在 JVM 上，有一个 Java 解释器用来解释 Java 编译器编译后的目标代码。Java 开发人员在编写完软件后，通过 Java 编译器将 Java 源程序编译为 JVM 的字节码。任何一台机器只要安装了 JVM，就可以运行这个程序，而不管这种字节码是在什么平台上生成的（如图 1-5 所示）。另外，Java 采用的是基于 IEEE 标准的数据类型。通过 JVM 保证数据类型的一致性，也确保了 Java 的平台无关性。

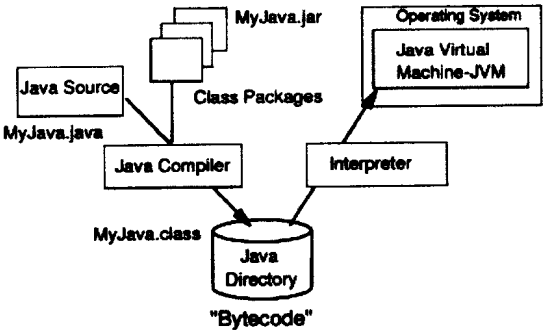

图 1-5 Java 的平台无关性

Java 的平台无关性具有深远意义。首先，它使得编程人员 Write once，run anywhere（开

发一次软件在任意平台上运行）变成事实，这将大大加快和促进软件产品的开发和推广。其次 Java 的平台无关性刚好迎合了“网络计算机”思想。如果大量常用的应用软件（如字处理软件等）都用 Java 重新编写，并且放在某个 Internet 服务器上，那么具有 NC 的用户将不需要占用大量空间安装软件，只需要一个 JVM，每当需要使用某种应用软件时，下载该软件的字节代码即可，运行结果也可以发回服务器。

8. 解释执行

Java 源程序被编译成类文件（.class 文件），它相当于程序的字节码表现。在一个 Java 类文件中，所有对方法和实例变量的引用都按名称进行，并且在第一次执行代码的时候加以分辨。这使得代码更通用，不容易受修改的影响。

Java 解释器直接对 Java 字节码进行解释执行。字节码本身携带了许多编译信息，程序的连接过程很简单。Java 解释器（Java Runtime Enviroment,Java 执行环境）能直接运行目标代码指令。连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多花时间。

9. 高性能

和其他解释执行的语言如 BASIC、TCL 不同，Java 字节码的设计使之能很容易地直接转换成对应于特定 CPU 的机器码，从而得到较高的性能。

10. 多线程

线程是操作系统的一种新概念，它又被称作轻量进程，是比传统进程更小的可并发执行的单位。C 和 C++采用单线程体系结构，而 Java 却提供了多线程支持。

多线程机制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，编程人员可以分别用不同的线程完成特定的行为，而不需要采用全局的事件循环机制，这样就很容易地实现网络上的实时交互行为。

这里举一个小例子来说明多线程的好处。很多人都有过这样的经历，当用浏览器试图打开一个网页的时候，等待一幅图片的下载完成有时需要一两分钟的时间，而在它被下载完成之前网页上的其他文字内容却不能浏览到，这确实是一件令人不愉快的事情。而在 Java 里，完全可以解决这个问题：可以用一个单线程来下载图片，而同时也可以访问 HTML 页面里的其他信息。

11. 动态性

Java 是个动态语言，这里指的是类库。Java 的设计使它适合于一个不断发展的环境。在类库中可以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。

1.1.3 Java 程序的工作机制

Java 程序源代码被编译成字节码之后，由 Java 运行环境（JRE）解释执行，这就是 Java 程序的工作机制。

JVM（Java Virtual Machine，Java 虚拟机）是最典型的 Java 运行环境，下面就介绍一些 JVM 的相关知识。

1. Java 虚拟机 JVM

Java 不仅是一种语言，更重要是一种区别于传统思想构建的，遵循“网络就是计算机”

这一信条的平台技术。Java 平台将面向对象系统扩展成包括程序和数据的网络计算机（NC），而这个平台的核心就是 Java 虚拟机，许多使 Java 成为万能开发平台的属性都源于 Java 虚拟机的概念和实现。

Java 虚拟机是一个假想的机器,它可以执行 Java 字节码，在实际的计算机上通过软件技术模拟来实现。

只要根据 JVM 的规格描述将解释器移植到特定的计算机上,就能保证经过编译的任何 Java 代码能够在该系统上运行。

2. Java 虚拟机的工作机制

Java 应用程序的开发周期包括编译、载入、解释和执行几个部分。Java 编译器将 Java 源程序翻译成 JVM 的可执行代码——字节码。这一编译过程同 C/C++的编译有些不同。当 C 编译器编译生成一个目标代码时，该代码是为在某一特定硬件平台运行而产生的。因此在编译的过程中，编译器通过查表将所有对符号的引用转换为特定的内存偏移量，用以保证程序运行。Java 编译器却不把对变量和方法的引用编译为数值引用，也不确定程序执行过程中的内存布局，而是将这些符号引用信息保存在字节码中，由解释器在执行代码的过程中建立内存布局,之后再通过查表来确定一个方法所在的地址。这样就有效地保证了 Java 的可移植性和安全性。

运行 JVM 字节码的工作是由解释器来完成的。解释执行过程分为三步进行：代码的载入、代码的校验和代码的执行。

1）代码的载入

载入代码的工作由“类装载器”（class loader）完成。

类装载器负责载入运行一个程序所要用到的所有代码。当类装载器载入一个类时，该类被放在自己的命名空间（namespace）中。除了通过符号引用自己命名空间以外的类，类之间不会互相影响。当载入了运行所需的类之后，解释器就可以确定整个可执行程序的内存布局。解释器在符号引用与地址空间之间建立对应关系并以查询表的形式导出。

2）代码的校验

被载入的代码在被执行之前必须接受字节码校验器的检查。

为了执行 Java 程序，虚拟机用类装载器从磁盘或者网络上获取字节码，而在传输过程中可能会出现错误，所以需要这样一种校验机制。每个类文件被送进一个字节码校验器，确保该类的格式是正确的，从而确保它可以被正常执行。这里需要指出的是，字节码的校验阶段会增加类载入的时间，但实际上能使程序运行更快，因为类的校验只进行一次。

3）代码的执行

通过校验后，代码便开始被执行了。

虚拟机的执行单元完成字节码中指定的指令。最简单的执行单元是一个解释器（Interpreter），读取字节码，解释其意义，并完成相关的操作。执行方式有两种：

（1）解释执行方式。

解释器通过每次解释并执行一小段代码来完成 Java 字节码程序的所有操作。

（2）即时（JIT- Just In Time）编译方式。

解释器先将字节码转变成用户机器的本地代码指令，然后再执行该代码指令。JIT 编译器运行在用户机器上，对用户来说是透明的。