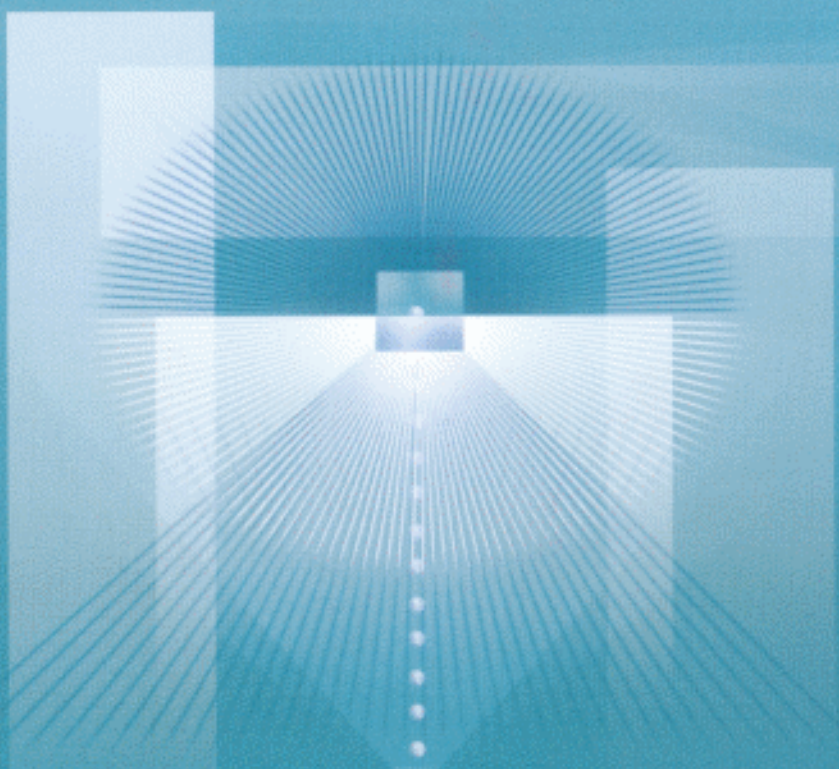
西安交通大學
XI'AN JIAOTONG UNIVERSITY
"十五"规划教材
21世纪大学计算机专业教材

Java程序设计

（修订本）

王志文 夏秦 李平均 编著 陆丽娜 审



西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

西安交通大学“十五”规划教材
21世纪大学计算机专业教材

Java 程序设计

（修订本）

王志文　夏　秦　李平均　编著

陆丽娜　审

西安交通大学出版社

·西安·

内 容 提 要

本书阐述了 Java 面向对象程序设计方法，共分 18 章，主要包括以下内容：Java 开发环境、Java 语言基础、Java 事件处理和异常机制、Java 类的创建与应用、Java Applet 基础、图形用户界面设计与布局管理器、输入/输出、网络编程、JDBC 数据库编程以及 RMI 与 CORBA 分布式编程技术。每一章都明确指出了应该掌握的重要内容，并附有课后练习题。

本书的特点是概念清晰、论述严谨、内容新颖、图文并茂、例程丰富，既重视基本原理和基本概念的阐述，又力图反映出 Java 语言的一些最新发展。本书可以作为高等院校计算机及相关专业的研究生、本科生教材，并可供各行各业从事计算机应用，特别是从事 Internet 网络应用编程的程序员使用。

图书在版编目(CIP)数据

Java 程序设计(修订本)/王志文，夏秦，李平均编著．
—2 版．—西安：西安交通大学出版社，2005．4
(西安交通大学“十五”规划教材)
ISBN 7-5605-1970-9

Ⅰ．J… Ⅱ．①王… ②夏… ③李… Ⅲ．JAVA 语言—程序设计—高等学校—教材 Ⅳ．TP312

中国版本图书馆 CIP 数据核字(2005)第 027618 号

书　　名	Java 程序设计(修订本)
编　　著	王志文　夏秦　李平均
责任编辑	贺峰涛　屈晓燕
出版发行	西安交通大学出版社
地　　址	西安市兴庆南路 25 号(邮编：710049)
网　　址	http://unit.xjtu.edu.cn/unit/jtupress
电　　话	(029)82668357　82667874(发行部) (029)82668315　82669096(总编部)
电子信箱	eibooks@163.com
印　　刷	陕西新世纪印刷厂
版　　次	2005 年 4 月第 2 版　2005 年 4 月第 1 次印刷
开　　本	727mm×960mm　1/16
印　　张	25.5
字　　数	473 千字
书　　号	ISBN 7-5605-1970-9/TP・398
定　　价	32.00 元

前　言

Java 是 Sun Microsystems 公司开发的强有力的新型程序设计语言。Java 语言之所以著名，不仅因为 Java Applets 可以在 Web 页面中运行，而且因为它确实是一种强有力的易于使用的面向对象程序设计语言。Java 语言可以处理许多常见的、但却相当复杂的问题，而这些问题是程序员在开发应用程序时经常碰到的。Java 通过它所提供的线程类来支持多线程程序设计，它还可以在后台执行垃圾回收功能，自动释放不再被使用的内存空间。Java 应用程序设计接口（Application Programming Interface，简称 API）包含在 Sun 公司所提供的 Java 开发工具（Developer's Kit）中，这些接口为程序员在编制复杂的 Internet 应用程序时，提供了对所需工具的平台无关性访问（platform-independent access）。

Java 使得平台无关性处理的理想成为现实。Java Applets 可以运行在任何可以运行 Java 型 Web 浏览器的机器上，而单个的 Java 程序则可以被编译为与平台无关的字节代码，然后，这些字节代码可以在任何配有 Java 解释器的机器上运行。可以说，Java 是第一种真正实现平台无关处理的流行高级程序设计语言。

为适应当前 Internet 的迅猛发展及相关专业人士学习 Java 语言的需要，尤其是根据高等院校的本科学生教学要求，结合笔者多年对 Java 语言的跟踪以及自身的应用开发体会，特编写《Java 程序设计》一书。本书共分 18 章，涉及的内容包括 Java 语言的发展、开发环境、基本语法、面向对象编程、网络编程、I/O 操作、JDBC 数据库编程以及 RMI 和 CORBA 等分布式编程技术。

编写教材最难处理的就是内容的取舍。Internet 网络技术的飞速发展以及软件开发模型的层次化走向使得 Java 语言也在不断地更新。在非常有限的篇幅中，应当将哪些最为重要的内容交给学生？经验证明，最重要的是要在教材中把该课程涉及的基本原理讲述清楚。尽管理论联系实际非常必要，但教材不能当作工程实践的手册指南使用。新的但尚未成熟的内容不宜写入教材。

本书承蒙西安交通大学陆丽娜教授审稿，对本书内容结构、编写大纲等方面提出了十分宝贵的意见；另外，西安交通大学计算机系统结构与网络研究所的唐亚哲、陈妍、朱海萍等许多同志对本书的编写也给予了关心和支持，在此一并表示衷心的感谢。

由于作者水平有限，加之 Java 语言的发展和变化非常迅速，书中难免有缺点、错误存在，欢迎同行专家和读者批评指正。

作者的电子邮件地址为：wangzhiwen51@sina.com。

王志文
于西安交通大学

第1章 Java概述

Java编程语言的诞生对于后来的网络应用产生了无比深远的影响，它的一系列技术特点使得它迅速流行并成为目前应用领域最为广泛的编程语言，尽管Java语言与传统的C/C++有些类似，但它们之间还是存在相当程度的区别。作为介绍Java编程语言的开始，本书将介绍Java程序运行机制以及它的运行时刻环境，这些内容对于理解Java程序的跨平台特性有着非常重要的意义。

Java语言诞生于1991年，是由Sun公司成功开发的新一代编程语言，它的最大特点在于使用它能够在各种不同种类机器、不同种类操作平台的网络环境下进行软件的开发。Java是一种跨平台语言，适用于分布式计算环境的面向对象的编程语言，它具有可移植性、高度的安全性、简单性、与体系结构无关性以及动态执行等一系列优良特征。近年来，随着Internet的兴起和分布式计算环境框架技术的成熟，Java正在逐步成为Internet应用的主要开发语言并成为全球信息网络舞台上一颗璀璨夺目的明星，它彻底地改变了应用软件的开发模式，在计算机领域里掀起了又一次技术革命高潮，为迅速发展的信息世界增添了新的活力。尤其是Java2平台的推出，进一步促进和加快了Java技术的应用。

需要声明的是：本书内容都是基于Java2平台而言的。

1.1 Java的诞生

1991年，Sun公司为了开拓消费类电子产品市场（例如交互式电视、互动影像等），成立了一个专门独立开发小组，命名为“Green”。该小组的领导人是James Gosling，他是一位非常杰出的程序员。在研究开发中，Gosling深刻体会到了消费类电子产品的特点，它们要求可靠性高、费用低、标准化、使用简单。为了使整个系统脱离具体平台的依赖——平台无关性，Gosling决定开始着手改写C编译器，但是他很快就发现仅仅依靠C是难以满足需求的，于是，他决定重新设计并开发出一种新的编程语言。

Green项目小组开发的最初产品包括四个组成部分：Oak，GreenOs（一种专门

的操作系统),用户接口模块以及硬件模块,这四个部分被集成到一个名为“*7”的类似 PDA 的电子设备中。成功地演示了这个系统以后,1993 年,Sun 公司成立了一个名为“FirstPerson”的子公司,不幸的是,这个系统在随后的几次大型商业活动中都惨遭失败。用 Java 创始人 James Gosling 的话说,他们发现“消费类电子工程市场并不是真实的,他们所期盼的远远超过了目前所能够做到的”。尽管如此,Green 项目小组坚持他们的使命,并得到了来自大公司的源源不断的支持,一群才干横溢的计算机专家可以集中精力研究和解决在网络应用(networking application)体系框架构造过程中出现的各种问题,最终,他们在技术上获得了成功。

1994 年,WWW 已经如火如荼地发展起来了。Gosling 此时意识到 WWW 迫切需要一个中性的浏览器,它不依赖于任何硬件平台和软件平台,而且它还必须是一个实时性较高、可靠安全、有交互功能的浏览器。同时,尽管 WWW 页面的内容丰富,但是由于它是静态的,因此需要一种机制来使它具有动态性,增强动感。嵌入式语言支持是一种解决方案,但是这种语言必须很简洁,此外,安全性对于 WWW 服务而言也是一个十分令人头疼的问题,对于这些问题,Oak 具有先天的巨大优势。于是,Gosling 决定首先将 Java 与 WWW 相结合。

Gosling 决定使用 Java 开发一个新的 Web 浏览器。到 1994 年秋天,完成了 WebRunner 的开发工作。WebRunner 是 HotJava 的前身,这个原型系统展示了 Java 可能带来的广阔市场前景。1995 年 1 月,Oak 被正式更名为 Java(由于早先已经存在一种称为 Oak 的计算机语言,它被设计用来在应用框架内部工作,为了避免商标侵权行为而更改名字),它于 1995 年 5 月 23 日发表后,在业界引起了巨大的轰动。一些著名的计算机公司纷纷购买了 Java 语言的使用权,如 MicroSoft,IBM,Netscape,Novell,Apple,DEC,SGI 等,因此,Java 语言被美国的著名杂志 PC Magazine 评为 1995 年 10 大优秀科技产品,随之而来的是出现了大量用 Java 编写的软件产品,并受到工业界的重视与好评,认为“Java 是 20 世纪 80 年代以来计算机界的一件大事”。

有人预言:Java 将是网络上的“世界语”,今后所有用其他语言编写的软件通通都要用Java语言来改写。

1.2 Java 的技术特点

Java 不仅仅只是一个广泛使用的网络编程语言,它还代表了一种全新的计算概念和程序设计思想,它的主要技术特点表现在以下几个方面:

1.2.1　简明的语法结构

Java 采用了与 C 相类似的语法结构。如果是一个熟悉 C 和 C++的程序员就会发现，Java的语法结构几乎和 C++完全相同(实质上还是存在少量差别)，而业界公认 C 语言是一种效率很高、灵活性极强的计算机语言，这也是使用 C 语言开发软件的程序员数量十分庞大的根本原因。因此，采用类似 C 的语法结构，程序员能够很容易地将 C 及 C++程序和 Java 程序进行相互转换。

Java 语言的发明者拥有丰富的 C 和 C++语言的使用经验，他们在吸收 C 和 C++优秀特征的同时，舍弃了其中一些容易引起问题，且不是绝对必要的部分功能。

指针是 C 和 C++的重要特征之一，甚至有人说过，指针是 C 的灵魂。指针是 C 和 C++语言完成一些高级程序设计的基础，使用指针，可以灵活高效地解决很多底层问题。然而，指针也不是完美无缺的，指针好比是一把双刃剑，一旦使用失误，就有可能给用户带来灾难性的后果。Java 语言决定取消指针，而仅仅保留了引用(reference)。这样，通过引用的使用，不但保留了指针的灵活性，而且避免了许多由于指针的存在而潜在的隐患。

另外，Java 语言的使用也比较简单。由于 Java 语言是一种面向对象的语言，它通过提供最基本的方法来完成指定的任务，只需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、多重继承等模糊的概念，并且通过实现自动垃圾回收，大大简化了程序设计者的内存管理工作。

1.2.2　平台独立性

众所周知，Internet 由世界范围内数以万计的各种各样的计算机系统组成。这些系统中的软件和硬件可能千差万别，各不相同，要让应用软件在网络上任何一种计算机系统中都能够正常地运行，就像是专门为该系统特别设计的一样，就必须使软件具有平台的独立性，也就是说，软件本身不受计算机硬件设备和操作系统的限制，软件代码具有相当的独立性，它可以在不同的计算机环境下良好地运行。

长期以来，软件的平台独立性一直是软件业发展的需求和编程人员苦苦追求的目标，而 Java 就是一种具有平台独立性的编程语言。可以说，跨平台性是 Java 语言得以迅速发展并获得成功的最主要原因，此时，程序开发者无需考虑对用户所使用的不同平台提供专门的特殊支持，真正实现了“一次编写，到处运行”的跨平台性目标。

我们知道，计算机处理的只能是 0，1 二进制位串，程序员编写出源程序以后，通过编译程序转换为计算机所能够理解的 0，1 指令串。在使用其他编程语言编写应用程序时，如 C 和C++等，编译程序产生的指令串是专门针对某一特定处理器

的，而 Java 编译程序的工作机理与上述传统方式完全不同，它是将源程序编译成中间代码（即所谓的虚拟机代码），该代码并不针对任何特定的处理器，它完全是一种中性代码，与具体的处理器无关。如图 1－1 所示，它给出了 Java 编译程序的工作原理。

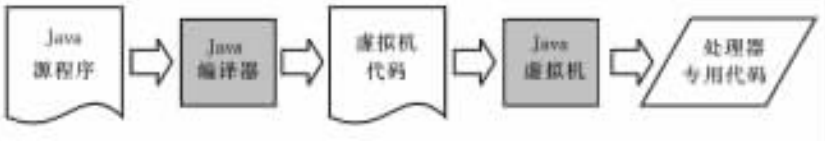


图 1－1　Java 编译程序的工作原理

除了给出基于 JVM（Java Virtual Machine，Java 虚拟机）的虚拟代码外，Java 还规定了同一种数据结构在所有实现中所占据的内存空间的大小。通过数据类型的空间大小方面的统一标准，Java 成功地保证了其程序的平台独立性，或者说是所谓的“体系结构中立性”（Architecture Neutral）。

正是由于采用了虚拟机制，这才使得一个 Java 编写的程序可以运行在任何一个安装了 Java 虚拟机的系统上。

任何事情都存在两面性，即好的方面与不利的方面总是并存的，Java 虚拟机也不例外，由于虚拟机代码的引入而带来了一个不利因素——速度问题。因为浏览器或者运行环境必须把虚拟机代码翻译成为物理处理器能够识别的专用代码，所以，Java 虚拟代码的运行速度不如专用代码的速度快。Java 程序的执行速度一般而言要比功能相当的 C 或 C＋＋程序的执行速度慢得多。幸运的是，随着系统硬件性能的大幅度提高，Java 程序的运行速度相应地得到了极大的改善，而且这方面的损失可以通过高移植性所带来的良好效益得以弥补。

1.2.3　面向对象特征

面向对象编程技术是当今世界软件开发中最为常用的技术之一，Java 就是一种新型的面向对象的程序设计语言，它集中于对象和接口的设计，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，不仅使应用程序开发变得容易和简单，而且程序的代码量也得以大大减少。

1.2.4　面向网络环境

Java 从一开始就作为一种面向网络的程序设计语言进行开发。我们已经看到由于虚拟机的出现而带来的许多优点，虚拟机避免了程序伤害其下载的计算机

且允许程序快速下载，并允许他们在其并不依附的支撑操作系统上运行。

这些优点都根植于 Java 语言的内部，在每日的编程过程中，程序员无需关心它们。既然 Java 从一开始就是面向网络环境的，所以我们在 API 中有了灵活的扩充功能，我们因此可以在 Internet 上进行自由地交互。通过 API，我们可以使用较高层次的抽象(abstraction)实体，例如 URL(Uniform Resource Locators)，或者在较低层次上直接进行通信，交换报文分组。

1.2.5　动态性

一个 C 或 C++程序，在经过编译以后，只是由机器指令组成的单一文件。对于大型程序而言，可执行文件的大小可以兆字节(megabytes)为单位进行计量。一个程序员真正开发一个大型项目时，他可能需要使用别人先前编写过的代码，而原来的源代码经过编译并被存储在库(library)中。当一个程序被编译时，新编写的源代码被链接到这个已存在的库上，然后整个映像数据被灌注到一个巨大的可执行文件中。

如果在程序中发现了一个“臭虫”(bug)，或者由于其他原因，程序员需要改变其中某一模块的功能，那又该怎么办呢？唯一的途径就是重新链接使用该库的所有应用程序。不过 Java 根本不会存在这类问题，因为它是在运行时刻才会将程序需要的类模块组合起来，这就意味着模块独立于其要编译的程序而存在。

Java 的动态性与 Windows 系统下的动态链接库 DLL(Dynamic Link Library)比较类似，运行中的程序只有在需要时才会加载相应的模块，这不仅可以加快程序的运行速度，而且可以减少程序运行的空间开销。

1.2.6　安全性

作为网络编程语言，Java 具有强大的安全结构和策略，代码在编译和实际运行过程中都会接受一层层的安全检查，这样可以防止恶意程序的攻击和病毒的入侵。为了实现这些安全性目标，Java 采取的主要措施包括有：取消指针操作，消除了复写内存单元和破坏有用数据的可能性；内存布局由 JVM 决定，并依赖于 Java 的运行时刻环境和 JVM 所在宿主机平台的特性，内存管理自动化；在字节码装载过程中使用了字节码检验器，确保了指令中参数类型的正确性、对象域访问的合理性以及操作数的边界检查(例如数组边界的自动检查)；使用类装入器，将本地类与从网络上下载的类分别置于不同空间，并对类之间的引用进行严格地审查和控制；利用沙箱模型，严格控制代码的访问权限。Java 软件包提供了多种网络协议(例如 FTP，HTTP 以及 Telnet 等)的用户接口，用户可以在网络传输中使用多种加密技术来保证网络传输的安全性和完整性。

1.2.7 稳定性

大多数程序设计语言在使用时，如果编写的程序有严重错误，那么，常常会造成系统死机。但是，Java 由于引进了对运行错误的异常处理机制，所以，当用 Java 语言编写的程序出错时，系统会转入异常处理，并自动寻找相应的异常处理方法，而不是使程序中断或系统死机。这一措施大大加强了 Java 系统的稳定性。

Java 对内存的垃圾收集机制其实是一种内存保护机制，这是 Java 系统稳定性的又一保证。这个机制使 Java 应用程序只能访问和修改有限的被许可的一部分内存区，它改变了传统的应用程序可以访问内存任何区域、可以修改任何内存单元的做法，而这种做法恰恰最容易造成系统出问题。另外，Java 不使用指针操作，从而避免了对内存地址的管理混乱，也避免了对有用数据的破坏，这也是对 Java 系统稳定性的有力支撑。

1.2.8 多线程

接触过多任务系统的读者一定知道进程这个术语，进程是指一个正在运行中的程序，它有自己管理的一组系统资源和一个独立的存储空间。进程的特点是它所涉及的数据、内存是独立的，所以，多进程系统就一定带有进程之间的通信机制，而为了实现进程通信，就要费去很多时间，也让系统付出许多开销。线程也是指一个正在运行中的程序，但线程有别于进程，即多个线程不仅可以共用同一个内存区域，而且可以共享同一组系统资源。对每个线程来讲，只有堆栈和寄存器数据是独立的，所以，线程之间进行通信和切换时，系统开销要比进程机制小得多。

Java 通过多线程运行机制可以很好地支持多任务和并行处理业务。Java 程序可以有多个执行线程，如果底层的操作系统支持多线程，Java 的线程通常被映射到实际的操作系统线程中，这意味着在多机环境下，用 Java 写的程序可以并行执行。Java 提供了 Thread 类及其一组内置的方法，使程序员在进行多线程程序设计时，只需继承这个类和调用相应的方法，就可以生成线程、执行线程或者查看线程的执行状态，由于 Java 的多线程特性，用户可编制多道作业调度程序，从而可以实现从服务器下载文本文件时，一边显示图形和一边收听音乐的多媒体处理效果。

1.2.9 类库丰富

Java 提供了大量的类库以满足网络化、多线程、面向对象系统的需要，系统提供的主要类和程序包有：

(1) 语言包提供的支持包括字符串处理、多线程处理、异常处理、数学函数处理等，可以用它简单地实现 Java 程序的运行平台。

(2) 实用程序包提供的支持包括哈希表、堆栈、可变数组、时间和日期等。

(3) 输入输出包用统一的“流”模型来实现所有格式的 I/O,包括文件系统、网络、输入/输出设备等。

(4) 低级网络包用于实现 Socket 编程。

(5) 抽象图形用户接口包实现了不同平台的计算机的图形用户接口部份,包括窗口、菜单、滚动条、对话框等,使得 Java 可以移植到不同平台的机器。

(6) 网络包支持 Internet 的 TCP/IP 协议,提供了与 Internet 的接口。它支持 URL 连接,WWW 的即时访问,并且简化了用户/服务器模型的程序设计。

除此以外,程序员或者中间产品供应商可以根据特定的应用(例如:多媒体应用、数据库操作等)需求自行灵活制定相应的类,这些类既可以用于以后的程序开发,也可以作为商品提供给其他软件开发者。

1.3　Java 与 C/C++的差异

熟悉 C 语言和C++语言的读者一定想搞清这个问题,实际上,Java 确实是从 C 语言和C++语言继承了许多成份,甚至可以将 Java 看成是类 C 语言发展和衍生的产物。比如 Java 语言的变量声明、操作符形式、参数传递、流控制等方面和 C 语言、C++语言完全相同。尽管如此,Java 和 C 语言、C++语言又有许多差别,主要表现在如下几个方面:

(1) Java 中对内存的分配是动态的,它采用面向对象的机制,采用运算符 new 为每个对象分配内存空间,而且,实际内存还会随程序运行情况而改变。程序运行中,Java 系统自动对内存进行扫描,对长期不用的内存空间作为“垃圾”进行收集,使得系统资源得到更充分利用。按照这种机制,程序员不必关注内存管理问题,这使 Java 程序的编写变得简单明了,并且避免了由于内存管理方面的差错而导致系统出问题。而 C 语言是通过 malloc()和 free()这两个库函数分别实现分配内存和释放内存空间的,C++语言中则通过运算符 new 和 delete 来分配内存及释放内存。在 C 和 C++这种机制中,程序员必须非常仔细地处理内存的使用问题。一方面,如果对已释放的内存再作释放或者对未曾分配的内存作释放,都会造成死机;而另一方面,如果对长期不用的或不再使用的内存不释放,则会浪费系统资源,甚至因此造成资源枯竭。

(2) Java 不在所有类之外定义全局变量,而是在某个类中定义一种公用静态的变量来完成全局变量的功能。

(3) Java 不用 goto 语句,而是用 try-catch-finally 异常处理语句来代替goto语句处理出错的功能。

(4) Java 不支持头文件,而 C 和 C++语言中都用头文件来定义类的原型、全局变量、库函数等,这种采用头文件的结构使得系统的运行维护相当繁杂。

(5) Java 不支持宏定义,而是使用关键字 final 来定义常量,在 C++中则采用宏定义来实现常量定义,这不利于程序的可读性。

(6) Java 对每种数据类型都分配固定长度。比如,在 Java 中,int 类型总是 32 位的,而在 C 和 C++中,对于不同的平台,同一个数据类型分配不同的字节数,同样是 int 类型,在 PC 机中为二字节即 l6 位,而在 VAX-11 中,则为 32 位。这使得C 语言造成不可移植性,而在 Java 则具有跨平台性。

(7) 类型转换不同。在 C 和 C++中,可以通过指针进行任意的类型转换,常常带来不安全性,而在 Java 中,运行时系统对对象的处理要进行类型相容性检查,以防止不安全的转换。

(8) 结构和联合的处理。在 C 和 C++中,结构和联合的所有成员均为公有,这就带来了安全性问题,而 Java 中根本就不包含结构和联合,所有的内容都封装在类里面。

(9) Java 不再使用指针。指针是 C 和 C++中最灵活,也是最容易产生错误的数据类型。由指针所进行的内存地址操作常会造成不可预知的错误,同时通过指针对某个内存地址进行显式类型转换后,可以访问一个 C++中的私有成员,从而破坏安全性。而 Java 对指针进行完全的控制,程序员不能直接进行任何指针操作。

1.4 Java 程序运行机制

Java 作为一种半编译半解释的语言,其程序的运行过程既有别于编译型语言,也有别于解释型语言。

图 1-2 给出了 Java 运行系统的功能结构图,从图中可以看出 Java 运行系统的功能是对字节码进行解释和执行,从工作过程看它主要分为三个步骤。

第一步是由类装载器完成字节码的装载,运行程序所涉及的所有代码都被装载。前面我们讲过,字节码中保留了地址的符号引用信息,装载时,运行系统通过建立符号引用信息和内存地址之间的对照表来确定程序的内存分配。

第二步是由字节码检验器对字节码进行安全性检查。这种检查可以排除字节码中可能存在的违反访问权限、不规范数据类型、错误指针、堆栈出错操作、非法调用等问题。

第三步是字节码的翻译和执行。这可以取两种途径之一来实现。一种是走右边支路,即通过代码生成器先将字节码翻译成适用于本系统的机器码,然后再送硬

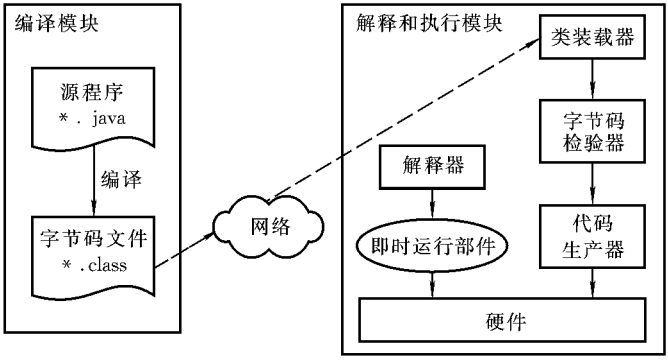


图 1-2　Java 运行系统功能图

件执行，这是一种编译型工作方式；另一种是走左边支路，通过解释器将字节码翻译成机器码，然后由即时运行部件立即将机器码送硬件执行，这是一种解释型工作方式，Java 运行系统一般采用第二种方式。只有对那些运行速度要求高的软件，才采用编译器工作方式，这时需使用特定的代码生成器完成编译，用这种方式时，程序运行速度和 C 语言编写的程序相当。而在解释型工作方式下，速度只是 C 或 C++语言程序的 1/15。

1.5　Java 运行时刻环境

在全面了解 Java 语言的所有细节之前，有必要对 Java 的基本结构有一个比较深刻的认识。在开始编写第一个 Java 程序以前，先介绍一下 Java 运行时刻环境(The Java Runtime Environment)。

1.5.1　Java 虚拟机

Java 虚拟机(JVM)是可运行 Java 代码的假想计算机。只要根据 JVM 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。对于一般程序员来讲，Java 虚拟机的机制和规范尽管不是至关重要的，但是，作为 Java 技术人员或资深程序员而言，了解 Java 虚拟机的概念和轮廓不仅是有益的，而且是必要的。

JVM 的设计目标是提供一个基于抽象规格描述的计算机模型，为解释程序开发人员提供很好的灵活性，同时也确保 Java 代码可在符合该规范的任何系统上运行。JVM 对其实现的某些方面给出了具体的定义，特别是对 Java 可执行代码，即字节码的格式给出了明确的规格。这一规格包括操作码和操作数的语法和数值、

标识符的数值表示方式、Java 类文件中的 Java 对象及常量缓冲池在 JVM 的存储映像。这些定义为 JVM 解释器开发人员提供了所需的信息和开发环境。Java 的设计者希望给开发人员以随心所欲使用 Java 的自由。

JVM 定义了控制 Java 代码解释、执行和具体实现的五种规格。

(1) JVM 指令系统

JVM 指令系统同其他计算机的指令系统极其相似,Java 指令也是由操作码和操作数两部分组成。操作码为 8 位二进制数,操作数紧随在操作码的后面,其长度根据需要而不同。操作码用于指定一条指令操作的性质(在这里我们采用汇编符号的形式进行说明)。JVM 采用了"big endian"的编码方式来处理这种情况,即高位 bits 存放在低字节中。这同 Motorola 及其他的 RISC CPU 采用的编码方式是一致的,而与 Intel 采用的"little endian"的编码方式,即低位 bits 存放在低位字节的方法不同。

(2) JVM 寄存器

所有的 CPU 都包含有用于保存系统状态和处理器所需信息的寄存器组。如果虚拟机定义较多的寄存器,便可以从中得到更多的信息而不必对堆栈或内存进行访问,这有利于提高运行速度。然而,如果虚拟机中的寄存器比实际 CPU 的寄存器多,在实现虚拟机时就会占用处理器大量的时间来用常规存储器模拟寄存器,这反而会降低虚拟机的效率。针对这种情况,JVM 只设置了 4 个最为常用的寄存器(均为 32 位)。它们是:

- pc 程序计数器。
- optop 操作数栈顶指针。
- frame 当前执行环境指针。
- vars 指向当前执行环境中第一个局部变量的指针。

(3) JVM 堆栈结构

作为基于堆栈结构的计算机,Java 堆栈是 JVM 存储信息的主要方法。当 JVM 得到一个 Java 字节码应用程序后,便为该代码中类的每一个方法创建一个堆栈框架,以保存该方法的状态信息,每个堆栈框架包括局部变量、执行环境以及操作数栈三类信息。

(4) JVM 碎片回收堆

Java 类的实例所需的存储空间是在堆栈上分配的。解释器具体承担为类实例分配空间的工作。解释器在为一个实例分配完存储空间后,便开始记录对该实例所占用的内存区域的使用。一旦对象使用完毕,便将其回收到堆栈中。

在 Java 语言中,除了 new 语句外没有其他方法为对象申请和释放内存。对内存进行释放和回收的工作是由 Java 运行系统承担的。这允许 Java 运行系统的设

计者自己决定碎片回收的方法。在 SUN 公司开发的 Java 解释器和 Hot Java 环境中，碎片回收用后台线程的方式来执行。这不但为运行系统提供了良好的性能，而且使程序设计人员摆脱了自己控制内存使用的风险。

(5) JVM 存储区

JVM 有两类存储区：常量缓冲池和方法区。常量缓冲池用于存储类名称、方法和字段名称以及串常量；方法区则用于存储 Java 方法的字节码。对于这两种存储区域具体实现方式在 JVM 规格中没有明确规定。这使得 Java 应用程序的存储布局必须在运行过程中确定，依赖于具体平台的实现方式。

1.5.2　Java 平台

如图 1－3 所示，一个完整的 Java 平台包括实际计算机、适配器、Java 虚拟机(JVM)、Java 基本软件和 Java 应用程序接口五个部分组成。

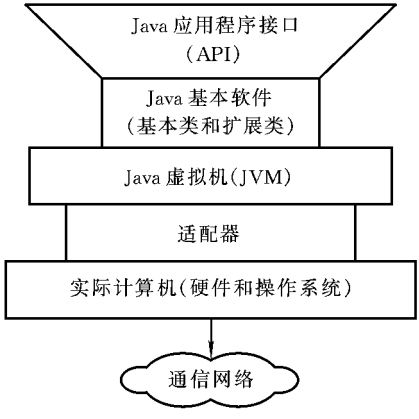


图 1－3　Java 平台的组成

JVM 是 Java 平台的核心。JVM 的下面是和实际计算机的接口，此接口也被称为适配器。可以想到，对于不同类型的计算机而言，适配器会有很大差别。而如果在针对 Java 设计的操作系统上建立平台，则可以省去适配器，因为在这种情况下，适配器的功能已经由操作系统提供了。JVM 的上面是 Java 基本软件和 Java 应用程序接口 API(Application Program Interface)。

Java 基本软件通常也称为基本类(有关类的概念在后面讲述)。实际上，基本类和 API 的规模不是固定的，许多 Java 平台对这两部分内容进行了补充，即除了基本类和 API 外，还有大量的扩展类和扩展 API，API 是 Java 平台和程序员之间的界面，程序员编写的 Java 应用程序可以在任何一个这样的 Java 平台上基于 API 运行。

1.5.3 字节码介绍

Java 应用程序的开发周期包括编译、装载、解释和执行几个部分。Java 编译程序将 Java 源程序翻译为 JVM 可执行代码——字节码(Bytecode)。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对象的代码时,该代码是为在某一特定硬件平台运行而产生的。因此,在编译过程中,编译程序通过查表将所有对符号的引用转换为特定的内存偏移量,以保证程序运行。Java 编译器却不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留在字节码中,由解释器在运行过程中创立内存布局,然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。解释执行过程分三步进行:代码的装入、代码的校验和代码的执行。装入代码的工作由"类装载器"(class loader)完成。类装载器负责装入运行一个程序需要的所有代码,这也包括程序代码中的类所继承的类和被其调用的类。当类装载器装入一个类时,该类被放在自己的名字空间中。除了通过符号引用自己名字空间以外的类,类之间没有其他办法可以影响其他类。在本台计算机上的所有类都在同一地址空间内,而所有从外部引进的类,都有一个自己独立的名字空间。这使得本地类通过共享相同的名字空间获得较高的运行效率,同时又保证它们与从外部引进的类不会相互影响。当装入了运行程序需要的所有类后,解释器便可确定整个可执行程序的内存布局。解释器为符号引用与特定的地址空间建立对应关系及其查询表。通过在这一阶段确定代码的内存布局,Java 很好地解决了由超类改变而使子类崩溃的问题,同时也防止了代码对地址的非法访问。

随后,被装入的代码由字节码校验器进行检查。校验器可发现操作数栈溢出、非法数据类型转化等多种错误。通过校验后,代码便开始执行了。

1.5.4 编译与执行过程

半编译半解释是 Java 语言的一个特点。我们知道编译是指一次性地把一个高级语言编写的源程序翻译成可以运行的目标程序,而翻译好的目标程序作为一个独立的文件可以无数次地运行。编译过程所需要的存储空间大,同时,编译所需要的时间较长,但目标程序执行时速度快。当前大多数语言属于这种类型,如 C 语言、FORTRAN 语言、PASCAL 语言等等。因为不需要多次翻译,所以这种方法特别适用于重复执行的程序。解释是指对高级语言编写的源程序每翻译一句然后再执行一句,翻译和运行过程交叉进行,如果要再运行一次,那就必须重新翻译,重新执行,翻译完即执行完。这类语言最典型的例子是 BASIC。解释型语言适用