Java 2

适合PC用户和Java程序开发人员阅读

标准Java 2类库

使用手册

魏海萍　编著

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

标准 Java 2 类库使用手册

魏海萍　编著

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

内 容 简 介

本书详细介绍了标准 Java 类库中的大量类和接口，并提供了相关的编程示例。全书共分 16 章，前面的 4 章主要介绍了 Java 的基础知识，包括 Java 的综述、类、方法、继承、包、接口、异常处理和多线程。后面的 12 章详细讨论了标准 Java 类库及其应用技巧，包括 I/O、字符串、集合框架、网络、小程序、GUI 控件、图像以及动画，还介绍了 Java 2 平台 1.4 版中的新增特性，如新 I/O 系统、文件映射、规则表达式等。

本书叙述清晰，内容全面，是一本适合广大 Java 程序开发者和爱好者使用的工具书。

图书在版编目（CIP）数据

标准Java 2类库使用手册/魏海萍编著.—北京：电子工业出版社，2005.1
ISBN 7-121-00416-X

Ⅰ.标… Ⅱ.魏… Ⅲ.Java语言—程序设计—技术手册 Ⅳ.TP312-62

中国版本图书馆CIP数据核字（2004）第100613号

责任编辑：李 莹
印 刷：北京天竺颖华印刷厂
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
北京市海淀区翠微东里甲 2 号 邮编：100036
经 销：各地新华书店
开 本：787×1092 1/16 印张：26.625 字数：680 千字
版 次：2005 年 1 月第 1 次印刷
定 价：32.00 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

近10年，程序设计领域发生了许多变化。C#已经问世，C++已进行了标准变化，新的C标准已经制定出来，.NET框架已经发布，Java也随着Internet和Web的兴起而开始变得越来越流行、越来越重要。

除了是一种用在Internet上的主要语言之外，Java变得如此重要的另外一个原因是它改变了计算机语言的发展进程。许多因Java而变得重要的特征，现在也在其他语言中找到了用武之地。例如，C#语言就受到了Java强烈影响。掌握了Java就等于打开了通往最新编程革命的大门。总之，Java是世界上最重要的计算机语言之一。

本书是为已经具备一定编程经验与水平，至少能够编写简单Java程序的读者而编写的。对于正在学习Java语言的读者，本书正是任何Java基础教程的最佳配套参考书。对正在编写Java程序的读者，本书也是一本不可多得的参考书。

本书详细描述了标准Java类库及其使用技巧，包括Java 2平台1.4版中新引进的各种特性，如新I/O包、规则表达式、文件映射等。全书共有16章，前面的4章主要介绍了Java的基础知识，包括Java的综述、类、方法、继承、包、接口、异常处理以及多线程，后面的12章详细讨论了标准Java类库，包括字符串、I/O、网络、集合框架、小程序、GUI控件、图像以及动画。全书还提供了大量的编程示例。

本书的编写思想立足于介绍Java类库及其使用技巧，因此力求简洁，减少了基础知识的介绍。由于编者水平有限，加上时间仓促，书中难免有疏漏或欠妥之处，殷切期望广大读者及同行批评指正。

欢迎与我们联系

为了方便与我们联系，我们已开通了网站（www.medias.com.cn）。您可以在本网站上了解我们的新书介绍，并可通过读者留言簿直接与我们沟通，欢迎您向我们提出您的想法和建议。也可以通过电话与我们联系，电话号码（010）68252397。

目　录

第 1 章　Java 语言概述

从计算机语言的发展历史来看，可以说 C 是由 B 发展而来的，C 又发展成了C++，而C++则是 Java 形成的基础。要想弄清楚 Java，就需要明白它的诞生原因、它的形成基础以及它所继承的传统。同其他计算机语言一样，Java 既继承了前辈语言的丰富优点，又提供了新环境对它的独特要求。尽管 Java 与 Internet 的联机环境变得越来越紧密，但 Java 终究是一种计算机语言。它的革新与发展仍然源于计算机语言的两个基本发展因素：适应不断变化的环境与用途，以及实现程序设计方法上的精美化和改进。

1.1　Java 的诞生

Java 是由 Sun 公司的几位工程师花费了 1 年半的时间于 1991 年开发出来的。它的第一个版本称为“Oak”，并于 1995 年被重命名为“Java”。在此期间，又有许多人对该语言的设计和发展做出了重要的贡献。

令人惊讶的是，开发 Java 的最初动机不是由于 Internet，而是因为编程人员需要一种平台独立，即体系结构无关的语言，以便用来开发嵌入到各种家电设备，如微波炉、遥控装置中的应用程序。众所周知，这些家电设备中用做控制器的 CPU 具有各种各样的类型。用 C、C++或其他计算机语言编写这类程序的麻烦在于，这些计算机语言是目标相关的，即特定于具体的 CPU 指令集。尽管程序员可以为任一类型的 CPU 都编译一个C++程序，但这要求一个以该 CPU 为目标的C++编译程序，而开发一个编译程序是非常耗时的，需要很高的开发成本。因此，程序员需要一种既容易又花费较少的解决方案。Java 的主要创始人 James Gosling 及其开发小组开始尝试开发一种具有可移植性和平台独立性的语言，以便程序员能够使用这种语言来生成可运行于不同环境下和不同 CPU 上的代码。这一努力最终导致了 Java 的诞生。

1.2　Java 对 Internet 的影响

Internet 把 Java 推向了程序设计的前沿，Java 反过来又对 Internet 产生了深远的影响，因为 Java 扩展了对象能够在计算机世界中自由移动的范畴。在网络中，服务器与客户计算机之间传递着两种对象：被动的信息和主动的动态程序。例如，在阅读电子邮件时，用户查看的就是被动的信息。即使在下载一个程序时，程序的代码本身仍只是被动的数据，除非用户运行它。可以传递到客户计算机上的第二种对象是动态的自运行程序。这类程序是客户计算机上由服务器启动的一个活动代理。例如，服务器可以提供一个程序来正确显示服务器正在发生的数据。

虽然动态的网络化程序十分受欢迎，但它们在安全和可移植性方面也受到了严重的挑战。在 Java 出现以前，计算机世界对大多数实体实际上是关闭的。然而，Java 的出现解决了这两方面的问题。

1.2.1 Java 小程序与应用程序

Java 可以用来开发两类程序:小程序与应用程序。Java 应用程序是指可以在计算机的操作系统下独立运行的 Java 程序,因而当用 Java 开发应用程序时,用 Java 所开发的应用程序与用其他任何一种计算机语言所开发的应用程序没有什么差别。Java 与其他语言的主要差别在于它能够用来开发小程序。小程序是指用 Java 开发的、通过 Internet 传输且由 Java 兼容的 Web 浏览器执行的 Java 程序。实际上,小程序就是小型的 Java 应用程序,而且图形、声音等文件也能够通过网络动态下载。两者的主要差别在于小程序是智能化的程序,而不只是动画或媒体文件。换句话说,小程序是能够对用户输入做出反应并动态变化的程序,而不只是反复执行同样动作的动画或声音。

下面来看一看小程序与应用程序之间的差别。Java 应用程序安装在客户端,并从命令行上启动。小程序通过 Internet 部署,并由 Web 浏览器执行。因此,在使用小程序时,一定要考虑小程序通过 Internet 下载的问题。由于小程序有可能在用户访问 Web 站点时被下载,所以浏览器必须提供运行小程序的受控制环境,以便限制小程序访问客户端,而只允许它们与源服务器进行通信。此外,浏览器要告诉小程序何时初始化、重画屏幕、启动、停止以及卸载。小程序只能使用浏览器在 HTML 页面中提供的小程序窗口来显示信息。

相反,由于应用程序在本地机上运行,所以能进行任何操作。小程序靠浏览器提供显示区,而应用程序在默认情况下不提供显示环境,即便能生成显示环境,也不能自动生成。小程序能从 HTML 页面中获取外部输入数据,而应用程序要求从命令行上输出数据。

虽然小程序令人兴奋,但如果 Java 不能解决两个与小程序相关的基本问题,即安全与可移植性,那么再聪明的小程序也不会有太大的用处。下面来看一看这两个问题对 Internet 究竟意味着什么。

1.2.2 安全

众所周知,每当下载一个"正常"的程序时,用户都可能会感染上计算机病毒。在 Java 出现以前,大多数用户不经常下载可执行程序,而且在运行下载的程序之前会扫描计算机病毒。即便如此,大多数用户仍担心他们的系统可能会感染上计算机病毒。除了计算机病毒之外,还存在另外一类需要加以防范的恶意程序。这类程序通过查找用户在本地系统文件中的内容来窃取用户的私有信息,比如信用卡账户、银行账户余额、密码等。Java 通过在网络应用程序与用户计算机之间设置一个"防火墙"来解决这两个问题。

在使用 Java 兼容的 Web 浏览器时,用户可以放心地下载 Java 小程序,不必担心病毒感染或恶意入侵。Java 是通过把 Java 程序限定在 Java 执行环境中,并禁止它访问客户计算机的其他部分来实现这种保护的。许多人认为,Java 最重要的特点就是用户下载小程序时不必担心造成伤害和违反安全规定。

1.2.3 可移植性

Java 的格言是"一次编写,到处运行",也就是说,同一个程序能够在任何一种平台上运行。Java 的这种可移植性是利用平台独立的字节码来实现的。正如前面曾经提到的,Internet 上连接了许多类型的计算机、操作系统和 CPU。为了让程序能够被动态地下载到已连接到 Internet 的各种平台上,就需要一些机制来生成具有可移植性的可执行代码。事实上,用来帮助

保证安全的相同机制也可以用来帮助产生可移植性。

1.3　字节码

能使 Java 得以解决安全和可移植性问题的关键在于，Java 编译程序所输出的不是可执行代码，而是字节码。字节码是一种被设计成由 Java 运行系统执行的、高度优化的指令集。Java 运行系统又叫做 Java 虚拟机(JVM)。JVM 的标准形式是字节码的解释程序。众所周知，编译后的 C/C++程序就变成了可执行代码。实际上，由于性能的缘故，大多数的现代计算机语言被设计成编译性的语言，而不是解释性的语言。然而，正是由于 Java 程序由 JVM 来执行的机制，才帮助解决了与通过 Internet 下载程序有关的主要问题。

把 Java 程序翻译成字节码能够帮助它们在各种环境下更轻松地运行，原因很简单：每种平台只需实现解释字节码的 JVM 即可。只要存在给定系统的 JVM，任何 Java 程序就都能够在该系统上运行。虽然 JVM 的实现细节在不同的平台之间有所不同，但它们都解释同样的字节码。如果把一个 Java 程序编译为本地执行的代码，就需要为 Internet 上所连接的每种 CPU 类型都编译该 Java 程序的一个不同版本。很显然，这不是一个切实可行的解决方案。因此，字节码的解释是产生真正可移植程序的最简单方式。

Java 程序的这种解释机制也使 Java 程序变得十分安全。由于每个 Java 程序的执行都处于 JVM 的控制之下，所以 JVM 能够包容该程序，并防止它在系统外产生副作用。正是由于 Java 中存在这样的限制，安全性才得到了加强。

当程序被解释执行时，通常会比编译后的可执行代码执行得慢。但是，就 Java 而言，两者之间的差别不是很大。字节码的运用使 JVM 执行 Java 程序的速度比预期的快得多。

虽然 Java 被设计成一种解释性语言，但是从技术上说，字节码也可以在运行时被毫无限制地编译成本地执行代码。为此，Sun 公司为字节码提供了运行时编译器(JIT)，这个编译程序被包含在 Java 2 中。当 JIT 为 JVM 的一部分时，它把字节码逐块地编译成可执行代码。需要特别注意的是，整个 Java 程序无法被立即编译成可执行代码，因为 JIT 只能在运行时完成各种运行时检查。相反，JIT 在执行时会根据需要编译代码。但是，这种运行时编译执行技术仍能大幅度地提升性能。即使在动态编译被应用于字节码时，可移植性和安全仍能得到保证，因为执行编译的运行系统仍负责控制执行环境。不论 Java 程序是按传统方式解释执行，还是在运行时编译执行，它的功能都不变。

1.4　Java 的结构特点

要讨论 Java 的起源，就不能不看一看 Java 的体系结构特点，否则让人觉得不够完整。虽然可移植性和安全是开发 Java 的两大本质驱动力，但其他因素也起到十分关键的作用。根据原先的 Java 白皮书，Java 的设计目标是成为“简单、面向对象、分布式、解释性、健壮、安全、体系结构无关、可移植、高性能、多线程的语言”。除了前面已经讨论过的安全与可移植性之外，下面来看一看 Java 的其他几个特点又意味着什么。

1. 简单性

Java 的设计目标是让专业程序员学起来轻松和用起来有效。对于一名有经验的程序员来

说,Java 掌握起来并不十分困难。如果知道面向对象程序设计的基本概念,学习 Java 会更容易。有经验的C++程序员转向 Java 不会有任何困难。由于 Java 继承了 C/C++的语法结构和C++的许多面向对象特征,大多数 C/C++程序员在学习 Java 时会觉得很轻松。

C++中令人费解而又不常用的方面要么已经从 Java 中删去,要么已经用一种更简洁、更任意处理的方式进行了实现。例如,Java 中删去了运算符重载,而加进了自动内存回收之类的特性。实际上,如果看一看 Java 的特性集和计算机语言发展史,就会发现 Java 中的一切都是借用现成的东西得来的。

- 类来自C++和 SmallTalk,而 Java 只限于单实现继承;
- 接口来自 Objective-C,而 Java 支持多接口继承;
- 包来自 Modula,而 Java 中增加了层次式名称空间和逻辑开发单元;
- 并发性来自 Mesa,而 Java 内置了多线程支持;
- 异常处理来自 Modula-3,而 Java 方法中增加了抛出异常声明;
- 动态链接和自动内存回收来自 Lisp,因而 Java 可以在需要时装入类,而在不需要时释放它们。

2. 面向对象特性

虽然 Java 受到其前辈语言的影响,但它没有被设计成兼容于其他语言的源代码。这使得 Java 开发小组没有任何约束地进行自由设计。结果之一是简洁、可用、注重实效的面向对象处理方法。Java 中的对象模型简单、容易扩展,而像整数之类的简单数据类型仍保留为高性能的非对象类型。

3. 健壮性

健壮性反映程序的可靠性。Web 的多平台类型环境对程序提出了另外的要求,因为程序必须能够在各种系统中可靠地运行。因此,在设计 Java 时,设计小组优先考虑了编写健壮程序的能力。Java 的几个内部特性使程序的可靠性得到加强。

- Java 是一种类型严格的语言。它不但在编译时检查程序的代码,同时还在运行时检查程序的代码。因此,Java 中不可能出现太多难以跟踪的隐含错误。Java 的一个关键特点是,知道程序员所编写的代码在各种条件下将用可预言的方式来表示。
- Java 没有指针,不能引用内存指针,因而避免了扰乱内存或超过数组边界的错误。Java 采用了引用概念,而没有采用指针概念,因而不能通过删除引用直接操纵内存空间。
- Java 进行内存自动回收,因而程序员不必操心内存的分配和重新分配问题。
- Java 鼓励多用接口,少用类。接口定义一组行为,类则实现这些行为。程序传递的是接口,而不是类,从而隐藏了实现细节。实现细节发生改变时,只要新类实现旧接口,其余一切照样正常工作。

为了更充分地理解 Java 的健壮性,下面来看一看导致程序出现故障的两个主要原因:存储器管理错误及错误处理的异常条件(运行时错误)。在传统的程序设计环境中,存储器管理是一项困难而又笨拙的任务。例如,在 C/C++中,程序员必须手工分配和释放所有的动态内存,这有时可能会引起一些问题,因为程序员可能会忘了释放以前分配的内存,而更糟糕的是,程序员可能会试图释放里面所保存的部分代码仍在使用着的内存。Java 通过替程序员管理内存的分配与重新分配,实际消除了这些问题。事实上,重新分配是自动完成的,因为 Java 提供

了自动内存回收，即所谓的垃圾收集机制来释放未使用的对象。传统程序设计环境中的异常条件，经常会导致诸如“除以零”或“无法找到文件”之类的错误情况，这种情况必须使用笨拙而又不易理解的结构来管理。Java 通过提供面向对象的异常处理功能，帮助程序员避免了这类情况。在编写得很完善的 Java 程序中，所有运行时错误都能够、也应该由程序来管理。

4. 多线程

线程可被理解为执行环境。例如，联机浏览时，浏览器在后台下载图形的同时显示新网页，每个任务可在不同线程里面执行。程序同时运行多个任务或线程时就叫做多线程。

Java 是为满足现实世界对创建交互式网络程序的需要而设计的。Java 通过支持多线程编程来满足这一要求。多线程编程能使程序员编写可并发地执行多项任务的程序。Java 运行系统带有一个用于多进程同步的复杂而又完善的解决方案，该解决方案能使程序员平滑地构造运行中的交互式系统。Java 的多线程方法能使程序员把精力集中在程序的特殊行为上，而不是集中在多任务的子系统上。

5. 体系结构无关性

体系结构无关性源于 Java 的平台独立性字节码。Java 程序并不被编译成平台特定的二进制代码，而是可以在任何平台上运行，无需重新编译和链接。

Java 的设计者所面对的主要问题是代码的使用寿命和可移植性问题。程序员所面对的主要问题是今天编写的程序能不能保证明天还能运行，即便是在同一台计算机上。操作系统的升级、处理器的更新换代、核心系统资源的改变等因素都可能使程序出错。Java 的设计者在尝试改变这种状况时，在 Java 和 JVM 中做了几项硬性规定。他们的目标是“编写一次，在任何地方、任何时间都能运行”。这个目标在很大程度上得到了实现。

6. 解释性

Java 程序是解释性的，不被编译成自然执行码，而是被转换成平台独立的字节码。这种字节码可以被传输给任何具有 Java 运行环境(JRE)和 JVM 的平台，并且在执行时不需要重新编译和链接。有人可能会认为这个过程会致使速度降低，但正如下一节将要介绍的，情况并非如此。平台独立的字节码实际上还包含其他信息，并可以在运行时根据编译时无法决定的决策进行执行优化。

7. 高性能

有人或许会认为“高性能”与“解释性”是相互矛盾的，但 Java 的平台独立性字节码可以在运行时转换成 CPU 特定的机器码，并能按照和 C/C++ 编译代码相同的速度运行。Java 带有两个可以自动完成这项任务的运行时转换工具：第一代工具是运行时编译程序(JIT)，第二代工具是 HotSpot。这两个工具的作用相同，即在运行时把 Java 的字节码直接转换成高性能的本机代码。同时，提供这个特性的 Java 运行时系统又不会丧失平台独立性的优点。

8. 分布式

Java 被设计成支持 Internet 并能像访问本地对象一样方便地访问分布式对象，因为它能处理 HTTP 之类的标准 TCP/IP 协议。事实上，使用 URL 访问资源与访问文件没有太大的

差别。Java 的最初版本(Oak)就包含了在寻址空间内传递消息的特性。这个特性使得两台不同计算机上的对象能够远程地执行过程。Java 把这些接口封装在一个叫做远程方法调用(RMI)的包中。这个特性给客户/服务器编程带来了前所未有的抽象。

最近,Java 还能通过 CORBA 和 Web Services 协议方便而直接地调用远程计算机上的方法。对每个协议,系统会自动处理所有转换与传输。

9. 安全

Java 有许多内置的安全特性,Java 2 的 1.4 版(简称 Java 1.4)中又增加了更多的标准安全特性。这些安全特性在几个不同方面起着作用。保证健壮型的许多因素也保证程序执行的安全性。例如,没有内存指针,因而不会扰乱内存。其他安全特性,包括字节码验证器、类装载器和安全管理器被内置在执行模型中,以保证代码执行安全,以及防止非信任代码执行硬盘读取之类的信任操作。安全的最后一个方面与验证、授权及加密有关,以保护隐私权和保证数据完整性。

10. 动态性

Java 还是一种动态语言,这里指的是库。具体地说,库可以不断更新和改进,而 Java 程序不需要重新编译和链接。此外,Java 程序本身带有大量运行时类型信息。程序员可以使用这些信息验证和解析运行时对对象的动态访问。这使得在安全得到保障的情况下动态地链接代码成为了可能。这对小程序环境的健壮性来说是至关重要的,因为在这类环境中字节码可以在一个正运行着的系统上动态地更新。

1.5 面向对象的程序设计

面向对象的程序设计(OOP)是 Java 的核心。事实上,所有 Java 程序都是面向对象的,这不同于C++中的情况。面向对象的程序设计对 Java 来说是必需的。在编写 Java 程序之前,有必要先了解一下 Java 的基本原则。

1.5.1 编程模型

众所周知,所有计算机程序均由两种元素组成:代码与数据。从概念上说,程序员可以围绕着程序的代码或数据来组织程序。更精确地说,有些程序是围绕着“什么正在发生”而编写的,而有些程序则是围绕着“谁正在受影响”而编写的。第一种编程方式叫做“面向过程的模型”。按这种模型编写的程序以一系列的线性步骤(即代码)为特征。面向过程的模型可被理解为作用于数据的代码。C 之类的过程化语言成功地运用了这种模型。但是,随着程序变得越来越大、越来越复杂,这种模型的局限性就显现了出来。

为了管理越来越高的复杂性,出现了第二种编程方式,即“面向对象的模型”。面向对象的程序设计围绕着程序的数据(即对象)和针对该对象而严格定义的接口来组织程序。面向对象编程的特点是数据控制代码的访问。通过把控制权转移到数据上,面向对象的模型在组织方式上有以下好处:抽象、封装、继承和多态。下面分别介绍它们。