“用实例学编程”丛书

用实例学 Java 2

Java 2 by Example

Second Edition

[美] Jeff Friesen 著

钟 萍 张玉峰 等译

叶喜涛 审校

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

内 容 简 介

本书结合大量具体实例介绍了面向对象的程序设计和Java基础。本书所介绍的Java语言的基础内容包括：运算符、表达式、语句，以及带有类和对象的面向对象程序设计、继承和动态方法等。本书还包含对面向对象分析和设计方法应用OOP的概念，并且示范了在集合中组织数据和应用Java的内置数学函数。此外，附录A中提供了与每章末尾习题相应的答案，以帮助读者很好地理解该章内容。

通过这种方法，读者可以学习数百个说明每个概念使用方法的生动实例。

本书语言简明通俗、内容生动翔实，可作为Java开发人员和非开发人员的参考书。

版权贸易合同登记号：图字：01-2001-4962

图书在版编目（CIP）数据

用实例学Java 2 /（美）弗里森（Friesen, J.）著；钟萍等译. - 北京：电子工业出版社，2002.7
（"用实例学编程"丛书）
书名原文：Java 2 by Example Second Edition
ISBN 7-5053-7759-0

Ⅰ.用... Ⅱ.①弗... ②钟... Ⅲ.JAVA语言 - 程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第045367号

责任编辑：马 岚 杜闽燕
印 刷 者：北京天竺颖华印刷厂
出版发行：电子工业出版社 www.phei.com.cn
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：38.5 字数：961千字
版 次：2002年7月第1版 2002年7月第1次印刷
定 价：59.00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

目　录

第一部分　学习这门语言

第三部分 附 录

第一部分
学习这门语言

第1章 Java入门

随着微软公司C#（读做C-sharp）语言和.NET平台的发布，你可能对应该花时间学习C#/.NET还是Java的问题感到疑惑。为了帮你做出更好的选择，本章将开始一个了解Java语言及其部分应用程序接口（API）的旅程。通过本章的学习，你会明白Java是什么、Java开发工具、不同的Java程序类型以及Java与C++语言的比较（如果你是一位具有C++语言背景的程序员，这个比较很有用处）。学完本章和本书的其余部分之后，你会发现花时间学习Java是很值得的。

1.1 Java是什么

Java是什么？Java是一个编程语言与执行环境的结合体。编程语言和执行环境协作，可以让开发程序无需修改就能运行于多种不同的硬件和操作系统的组合之上；这样，不怀好意的程序员就很难（不是不可能）创建Java程序来引入病毒、盗窃口令信息、破坏机器设备以及执行其他各种恶意操作。Java是怎样实现这些任务的呢？本节将给出答案。在探究这些答案之前，我们先分析Sun微系统公司对Java的定义。

说明： Sun微系统公司是Java语言的创始者，同时也是Java技术发展的原动力和推动者。读者可以访问Sun公司的Java网站（http://java.sun.com）来获得最新的Java信息。

1.1.1 基本定义

Sun公司将Java定义为“一种简单的、面向对象、支持网络、解释型、健壮、安全、结构中立、可移植、高性能、多线程和动态的语言”。这个定义太长了！下面我们一点一点地分析这个定义。

Java是一种简单的语言，它让初学者的学习曲线平滑，并能够快速成为一个多产的程序员。Java设计者意识到很多开发人员具有C或C++的开发经验，这个认识使得Java的语法基于C或C++建立，从而简化了很多程序员关于Java语言的学习。但是Java取消了一些C++语言或C和C++语言都有的复杂功能，如C++语言的运算符重载、多重继承和析构函数；以及C和C++语言的指针和自动强制类型转换。删除这些功能（新手常常难以理解）简化了Java。

Java是面向对象的语言。Java开发人员利用程序语言在源代码中建立问题模型，可以集中精力解决问题。你可以用一个对象来表示每个问题实体，并且说明这些对象之间的交互关系。对象将实体的状态和动作结合起来。相反，非面向对象语言的程序员致力于将问题转化为符合语言的等价表示。因此，即使非面向对象的程序和面向对象的程序都可以解决同样问题，但是一个非面向对象程序的源代码要比对应的面向对象程序的源代码更难以理解和维护。

Java是一种支持网络的语言。Java内置许多TCP/IP网络库，使它容易与运行在网络机器上的各种各样的TCP/IP进程（如HTTP和FTP）建立连接，而且这些库访问网络对象与访问本地文件系统上的对象所用的机制相同。因此在理解了Java如何访问文件之后，就自动理解了它如何处理网络访问，这样可节省学习时间。在电子商务世界中，理解网络是很重要的。

Java是一种解释型的语言。Java编译器将Java源代码转换为字节码（bytecode）指令。这些字节码执行（也称运行）于虚拟机———一个模仿计算机的计算机程序。该虚拟机通过解释（一种机制，首先读字节码、理解其含义，然后在相应平台上执行等价的微处理器指令。平台指虚拟机运行时所用的底层计算机体系结构和操作系统）来执行字节码。

说明： 开发人员经常将Java虚拟机表示成JVM或Java VM。

Java是一种健壮的语言。换句话说，它有助于开发人员编写不易出故障的代码，对于那些对错误要求极严格的程序员很有好处。Java通过强制执行比C和C++语言更严格的类型检查、取消指针（指针使用不当会造成程序崩溃和安全性破坏）、提供真正带边界检查的数组防止访问不存在的数组元素，并使用一个后台工作的无用内存收集机制来负责释放动态分配的内存等方法，来部分实现其健壮性。

说明： 在使用动态分配和释放内存的语言中（如C++），时常出现程序员忘记释放无用内存的情况，这样会导致内存泄漏，最后使得运行程序“慢慢停下来”。由于Java具有无用内存收集机制负责释放动态分配的内存，因此内存泄漏很少出现（Java中也会发生内存泄漏，但是本书将介绍如何避免出现这种情况）。

Java是安全的语言。编译器（compiler）与JVM的解释器相配合，提供多种检查以确保病毒和其他“脏”程序不能影响计算机的文件。该功能在网络环境中具有很重要的意义；在网络环境中可能会下载和执行Java软件，而且事先不知道程序的作者和程序的功能。如果缺乏安全性，你就不知道（直到事后）执行一个Java程序可能导致你的信用卡或者口令信息被窃、硬盘上的文件被删除或者打印机大量的纸被浪费。Java的安全功能十分灵活，可以从十分安全（如小应用程序applet）到不安全（如应用程序）。

说明： 安装安全管理器以后可以增强应用程序的安全性，安全管理器是一种特殊的对象类型，它与JVM一起提高安全性。但有关安全管理器和其他安全话题的讨论超出了本书的范围。

Java是结构中立的语言。一般只需要编写一次程序、编译一次该程序，然后（从理论上说）就能够运行于任何支持Java的体系结构——从最小的消费类电子设备（如蜂窝电话或寻呼机）到个人计算机和最大的主机。虽然大多数情况下这样的工作组织得很好，但是开发人员需要记住几个注意事项（gotcha）：其中的一些将在本书中进行讲解。结构中立（又称平台无关性）是通过编译器产生基于JVM的执行代码，而非特定计算机结构的执行代码来实现的。

Java是一种可移植的语言。它要求基本数据类型（如整数和浮点数）在任何平台上都具有相同的大小。这与C和C++等语言形成鲜明对比（在这些语言中一个平台与另一个平台上的整数类型大小不同），而且Java还使用抽象类和接口来“保护”程序，避免不同平台的结构差异。比如要操作图形，Java程序将调用一个抽象类的功能。相比之下，该抽象类才与真正执行图形操作的特定平台上的类相互通信。最后，Java的大部分（如编译器）是使用Java编写的；这样，

虽然与平台通信的那部分Java是用C或C++编写的，但是边界很清楚。将Java移植到不同平台的工作要容易得多，因为只有用C或C++编写的部分需要进行移植。

Java是一种高性能的语言。虽然Java的旧版本比较慢（因为字节码需要解释的固有特性），但是通过引入即时（Just-In-Time，JIT）编译器，Java的执行速度得到了很大的提高。现在的JVM包含JIT编译器，该编译器在Java程序运行的时候将字节码编译为平台本身的微处理器指令，这样提升了运行性能。

Java是多线程的语言。多线程将一个程序的全部执行划分为多个独立的执行路径（称为线程），它们可以同时运行（如果底层平台使用多个微处理器——每个线程运行在一个处理器上）或者基本上同时运行（如果底层平台使用一个微处理器——该处理器为全部线程所共享）。例如一个多线程的程序可以响应用户输入，同时检索大量的信息。虽然类似C和C++这样的语言也支持多线程（在API级别），但是需要进行更多的工作，因为与Java不同，C或C++都没有同步访问共享数据的语言支持。缺乏同步机制就有可能出现一个线程读取一个数据项，而该数据项正在被其他线程写入的情况，这样读线程仅仅获得新值的一部分，是一个错误的结果。总而言之，如果仔细使用，多线程将会增强Java程序的性能。

Java是一种动态的语言。Java能够适应连续进化的环境，通过将程序与库的连接推迟到运行时刻，以及通过使用接口来表示对象完成功能而不是具体的实现（“实现”部分由实现接口的类来完成，对象是从该类创建得到的）。只要程序与类的接口保持不变，而且程序通过这些接口来通信，对象所属类的行为就可以进行变化，并且不必分解代码。另外与C++不同的是，Java具有用任意对象的地址返回对象真正类型的能力。该能力使Java更具有动态性。

1.1.2 程序开发和执行环境

Java程序由源代码开始。启动一个编辑器，输入Java程序的源代码，然后将这些代码保存到文件中。在选择文件名时，注意将文件扩展名指定为.java。如果不指明文件的扩展名，一些Java编译器（Sun公司的javac编译器最明显）将不能识别这个源文件。

Java编译器负责将Java源文件中正文的内容转换为字节码（即JVM指令）和相关数据。由于源代码是基于类的，所以编译器将遇到的每个类生成的字节码置于自己的类文件中。图1.1说明了这个编译过程。

如果编译成功，则创建执行程序。在执行这个程序时，JVM的类加载器（class loader）逻辑把其中称为开始类的类文件加载到JVM。

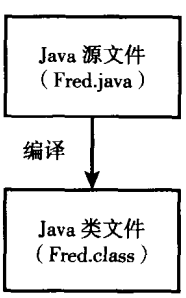

图 1.1 Java编译器将（保存在Fred.java中）名为Fred类的源代码转换为等价的字节码和数据。编译器在一名为Fred.class的类文件中保存这些字节码（和数据）

在类加载器完成开始类文件的加载之后，它请求 JVM 的字节码验证器（bytecode verifier）逻辑检查该类文件的字节码是有效的，并且没有破坏 Java 的安全约束条件。如果字节码验证器发现一个安全性破坏因素，它将强制结束 JVM。

如果验证成功，则 JVM 的解释器依次解释开始类文件的字节码指令，程序开始运行。一般解释器与 JIT 编译器一同执行这个任务。当解释器发现字节码序列被重复执行时，它让 JIT 编译器将这个字节码序列编译成平台的本地指令（native instruction），以后在遇到这些字节码序列时就执行这些本地指令。这样做可使性能显著提高。

在解释字节码的时候，解释器可能会遇到要求执行位于其他类文件中的字节码的情况，这个文件可能是 Java 扩展类库的类文件。如果出现这种情况，解释器将联系类加载器定位并加载这个类文件。在新的类文件加载之后，JVM 的字节码验证器逻辑验证类文件的字节码，如果字节码验证器在这些类文件的字节码中发现一个安全违规，它将结束 JVM 和程序；如果验证成功，则解释器开始解释这些字节码。该请求、加载、验证、解释的过程持续运行直到程序结束，结束的条件可以是正常或者出现错误（从简单的被0除错误到发生复杂的输入/输出访问错误）。图 1.2 说明了类加载、字节码验证和解释的过程。

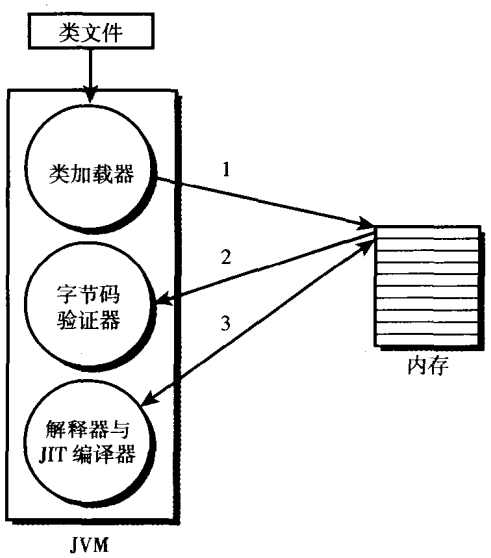

图 1.2　Java 程序的执行过程如下：JVM 的类加载器将每个类文件（组成这个程序的类文件）加载到内存，JVM 的字节码验证器验证这些在内存中的指令，然后 JVM 的解释器（与 JIT 编译器一起）执行这些指令

1.1.3　简要的历史回顾

前面介绍了 Java 是什么、Java 程序开发机制和执行环境，下面开始介绍 Java 的由来。

Java 是怎样产生的？Java 的起源可追溯到 1990 年 11 月，当时 Sun 公司的雇员 Patrick Naughton，Mike Sheridan 和 James Gosling 接受了一项任务，任务的内容是预计信息处理技术下

一步的主要发展趋势。他们都断定采用数字控制的消费类电子设备和计算机的结合将有重要意义，因此Sun公司的绿色计划（green project）诞生了。

绿色计划除了开发出一种令人耳目一新的、能够运行UNIX的简化版本（small-footprint）之外，还促使开发出一种基于C和C++的语言。它的创造者James Gosling以其在Sun公司办公室窗外生长的一棵橡树命名，将这种语言称为Oak。后来发现已经有一种名为Oak的语言存在，于是将其改名为Java。

说明： 读者可以访问Sun公司的网页http://java.sun.com/people/jag/green/index.html，随着卡通Java吉祥物Duke的出现，了解到更多关于绿色计划Oak或Java的早期情况。

绿色计划也遇到了困难。在20世纪90年代早期，智能消费类电子设备的市场发展非常缓慢——相对Sun公司的需求而言太慢了。另外，由于Sun公司失去了一份将Java包含到另外一个公司中的重要合同，绿色计划几乎被取消。但是在1994年，随着处于襁褓中的Web（World Wide Web）的流行，Java的未来出现光明。Sun公司注意到这是一个机会，可以将Java用于向网页提交动态内容，于是委派绿色计划小组开发一种基于Java的Web浏览器（这就是后来的HotJava），该浏览器可以在网页中嵌入（运行）Java程序（后来被称为applet)。Netscape（网景）公司发现了这个可以提高浏览器市场份额的机会，便与Sun公司签订合同将Java放在Netscape公司的Web浏览器中（后来微软公司也仿效网景公司的做法，与Sun公司签订合同将Java包含到IE Web浏览器中）。

在1995年5月，Sun公司将Java的正式描述提交到一次重要会议。企业界发现由于Web的流行，Java变得更具吸引力（这并不妨碍Java针对商业目的的设计）。由于Java很适合目前的电子商务项目，特别是在客户/服务器形式的电子商务解决方案的服务器上，因此世人对Java的兴趣与日俱增。

1.2 开发工具

开发Java程序需要相应的开发工具。无论是免费获得的自由软件，还是昂贵的真正的商业产品（通常要花很多钱），在Solaris，Mac OS，Linux，Windows和其他平台之上都有许多创建Java程序的开发工具可用。

1.2.1 从JDK到SDK

在1995年秋天，Sun公司发布最初的Java开发工具箱（JDK）。JDK 1.0分Windows 95/NT和Solaris平台，由编译器、相关工具、例子和文档组成。但是JDK 1.0只能创建简单的程序，因此Sun公司集成一种新的事件处理模型（以及其他功能），通过增强Java对部件开发的支持，推出了JDK 1.1。

JDK 1.1的一个主要问题是其图形用户界面（Graphical User Interface，GUI）子系统，这个称为抽象窗口工具箱（Abstract Windowing Toolkit，AWT）的子系统的功能很有限，因为它是绑定在底层平台之上的。另外，使用AWT创建的GUI在Solaris，Windows和Mac OS等不同平台上的显示效果不同。这些限制促使Sun公司开发了一个扩展、增强的AWT，即Swing。Swing可

以开发出复杂的GUI，并且它们既可以看上去与平台相关，又可以在任何Java支持的平台上生成相同的效果。在JDK中包含Swing是一个重要的里程碑——产生Java 2。

Sun公司对名字进行了重大调整，不是将其下一代Java版本称为JDK 1.2，而是变成Java 2 SDK（软件开发工具箱）1.2版。改变名字是因为开发人员希望在大型项目的客户端和服务器端都使用Java，而且他们也希望在小型消费类电子设备上使用Java。设想将这三个领域所需的功能集成到一个产品中，其结果将是一个巨大的开发工具箱，而且并不是所有功能都是每个Java开发人员所需要的。为了简化SDK，Sun公司发布了三个独立的Java 2版本：

- Java 2 Platform, Standard Edition
- Java 2 Platform, Enterprise Edition
- Java 2 Platform, Micro Edition

本书主要使用Java 2平台标准版SDK来创建客户端的程序，但是这个开发工具箱也支持服务器端的交互。相比之下，Java 2平台企业版SDK更强调服务器端的开发。最后，Java 2平台Micro版SDK适合为消费类电子设备（如蜂窝电话、智能卡、寻呼机以及汽车导航系统等）编写程序的开发人员。

在Sun公司将产品分为三个开发工具之后。每个开发工具独立地发展版本。例如，在Java 2平台标准版1.2之后，Sun公司发布了1.3版以及现在的1.4版。你可以访问Sun公司的Products & APIs（产品和API）网页（http://www.javasoft.com/products/?frontpage-main），下载并了解每种开发工具箱最新版本的信息，另外由于本书主要介绍Java 2平台标准版1.4版，你还可以访问Sun公司的Summary of New Features and Enhancements（新特性和增强功能摘要）网页（http://java.sun.com/j2se/1.4/docs/relnotes/features.html）来了解1.4版中的新增内容。

说明： JDK 1.1的出现促使Apple公司将Java移植到其基于Mac操作系统的平台上。Apple公司的Java实现称为Macintosh Java运行时（Macintosh Runtime for Java，MRJ），在本书编写的时候，Apple公司该产品的最新版本是MRJ 2.2.5。

MRJ 2.2.5的基础是Sun公司的JDK 1.1.8，因此Mac上的开发人员不能使用JDK 1.1.8之外的Java特性（包括SDK 1.4的特性）。但Apple公司2001年发布了Mac OS X，将Java的新特性引入Mac。虽然Mac OS X包含Java 2平台标准版的1.3版，但是不支持1.4版（在编写本章的时候），这个状况将会改善（而且会尽快得到改善）。

要获取MRJ 2.2.5的副本，请访问http://www.apple.com/java，但如果要了解Mac OS X上Java版本的最新信息，可以阅读Apple公司的Java Development for Mac OS X PDF文档，该文档可以在http://developer.apple.com/macosx/pdf/macosx_java.pdf上找到。

下载 SDK 1.4

如果想直接使用Sun改进过的SDK 1.4（而不是集成了SDK 1.4的开发工具），可以从下面的网页下载SDK 1.4：http://www.javasoft.com/j2se/。这个网页包含最新的版本、相关技术以及链接到Java 2平台标准版产品家族的先前产品目录。

在Current Releases下面，可以找到SDK的链接、Java运行时环境（Java Runtime Environment，JRE）以及文档。单击SDK链接继续SDK的下载过程。

说明： JRE表示Java运行时环境。JRE基本上是没有开发工具和示例的SDK，它要比SDK小，一般被下载和安装在运行Java程序的计算机，而非开发程序的计算机上（Java程序可以运行于开发程序的计算机上，因为SDK包含JRE的副本）。

由于文件大小的原因，SDK的文档是与SDK独立打包的。如果文档被包含在SDK中，则可能需要一次下载50多兆的内容，而且文档是周期性进行修改的。为了获得更新后的文档，而要求开发人员重新下载整个SDK是不合理的。在下载了SDK以后，如果你选择下载文档，则需要单击文档的链接；如果不愿意下载文档，则可以选择到下面的网址去“冲浪”，在线浏览文档的网址是http://java.sun.com/j2se/1.4/docs/。

在单击了SDK链接之后，可以将Java 2平台标准版1.4版的网页作为浏览器的默认页面。网页中心的顶部显示下载Solaris SPARC/x86，Linux x86以及Microsoft Windows等平台SDK 1.4版的链接，单击适合你所用平台的链接。

假设要下载Windows平台的SDK 1.4，可以在SDK for Windows网页选择将SDK 1.4作为一个大包下载，或者作为“磁盘大小倍数的片段”（每个片段为1.44 MB或更小）。下载网页上提供了将多个片段组合成等价的打包信息。

提示： 如果是通过调制解调器连接到Internet的，则可能希望选择用较小的磁盘大小倍数片段的选项，因为一些Internet服务提供商（ISP）在一段时间后，会断开链接。这种做法对将SDK作为一个打包来下载很不好，通过56 Kb/s的调制解调器下载也要花很长的时间。相比之下，将SDK按多个片段进行下载只会损失一个1.44 MB的片段（下载这个片段只需要几分钟）。

如果决定使用打包下载选项，而且单击了Continue按钮，就进入了许可证协议网页。单击Accept按钮接受许可证的条款，然后继续下载过程。出现的下一网页包含几个按钮，用于下载SDK。如果你的计算机位于防火墙之后，单击HTTP下载按钮，否则单击包含FTP下载标签的按钮。无论哪种情况，都会显示Save As对话框（通过浏览器），同时选择保存SDK安装文件（j2sdk1_4_0-win.exe）的位置。

在下载完SDK安装文件之后，运行这个程序，并按照屏幕提示安装SDK 1.4。如果需要改变包含SDK工具的位置，则在Windows上，通过修改PATH环境变量，使其包含jdk1.4\bin目录（虽然术语SDK被广泛应用，但是由于历史原因，安装程序创建一个默认的jdk目录作为安装Java的父目录）。假设SDK被安装在c:\jdk1.4上，你可以在autoexec.bat文件（针对Windows 95/98/ME用户）的当前PATH设置之后附加c:\jdk1.4（如PATH=%PATH%，c:\jdk1.4\bin）或者使用控制面板的环境工具（针对Windows NT/2000/XP）。

SDK 1.4版本的Windows教程

安装好SDK 1.4之后，你可能想知道现在计算机上多了哪些文件。在jdk1.4目录下，将找到下面几个文件：README.txt，readme.html，COPYRIGHT，LICENSE，Uninst.isu以及src.jar。

README.txt和readme.html文件中包含入门所需的有用信息（如系统配置要求，以及指向各种包含其他信息的网站链接），这些信息以文本或者HTML的格式出现。要了解SDK，最好在安装之后阅读其中一个文件。COPYRIGHT和LICENSE文件包含Sun公司委托版权（mandatory copyright）以及许可证协议信息。一般忽略这些文件，但是快速浏览一下其中的内容也会有好处。你可能会注意到一个名字很奇怪的文件Uninst.isu，该文件包含卸载SDK时所用的信息。不要删除这个文件，没有这个文件将不能卸载SDK。