普通高等院校“十二五”立项教材

Java 语言程序设计

主编◎邢静宇

吉林大学出版社

普通高等院校“十二五”立项教材

Java 语言程序设计

主　编　邢静宇
副主编　邱　雅　钱　鸽
编　者　（按姓氏笔画排序）
许　豪　河南南阳理工学院
邢静宇　河南南阳理工学院
邱　雅　河南南阳理工学院
单平平　河南南阳理工学院
钱　鸽　河南南阳理工学院

吉林大学出版社

图书在版编目(CIP)数据

Java语言程序设计 / 邢静宇主编. —— 长春 : 吉林大学出版社, 2014.12

ISBN 978-7-5677-3009-0

Ⅰ. ①J… Ⅱ. ①邢… Ⅲ. ①JAVA语言-程序设计-高等学校-教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2014)第304722号

书　　名：Java语言程序设计
作　　者：邢静宇　主编

责任编辑：李伟华　责任校对：魏丹丹　　封面设计：可可工作室
吉林大学出版社出版、发行　　北京楠海印刷厂印刷
开本：787×1092　毫米　1/16　　2015年1月 第1版
印张：19.5　字数：430千字　　2015年1月 第1次印刷
ISBN 978-7-5677-3009-0　　定价：38.00元

社址：长春市明德路501号　邮编：130021
发行部电话：0431-89580028/29
网址：http://www.jlup.com.cn
E-mail:jlup@mail.jlu.edu.cn

前　言

Java是一门面向对象的语言，Java的不断发展要归功于C、C++和C#等编程语言的不断挑战。C++、C#和Java等编程语言基本上都来源于C语言但又有很多区别。对于变量声明、参数传递、操作符、流控制等，Java使用了和C、C++、C# 相同的传统，但摈弃了许多难以控制的技术和非面向对象特性，如多继承和独立于类之外的方法等，是更为纯粹的面向对象语言。

从JDK诞生到现在已经有19年的时间了，沧海桑田一瞬间。转眼19年过去了，JDK已经发布了7个版本。在这19年里诞生了无数和Java相关的技术和标准。Java不仅可以用来开发大型的桌面应用程序，而且还特别适合于Internet的应用开发，也可以开发嵌入式移动应用。目前，很多新的技术领域都涉及Java，因此Java也是学习面向对象编程的首选语言。

本书的章节编排合理，内容循序渐进，注重知识的可用性，摈弃了不常用的语法与知识点，内容力求简明，并基本覆盖了Java语言中常用的基础知识点。

本书可以作为本科及高职高专计算机Java语言类的教材及职业培训教材，也可作为非计算机专业的Java语言的入门和普及教材，还可以作为Java语言初学者的参考书籍。

全书共分11章，具体内容如下所述：

第1章是Java语言概述，介绍了Java语言的特点，Java虚拟机及跨平台原理，Java开发环境的搭建及如何编写用户的第一个Java应用程序。

第2章是Java语言基础，主要给出了Java语言中的标识符和关键字，常量和变量，基本数据类型，运算符和表达式的定义和使用，同时，也介绍了Java中的注释，API文档的查看方法。

第3章是流程控制，主要讨论了Java编程中的顺序结构，分支结构，循环结构等流程控制内容。

第4章是面向对象基础，给出了面向对象的基本概念，包括类和对象。另外，还对static和final关键字进行了介绍。

第5章是面向对象高级特性，介绍了类的继承，类的多态性，接口和抽象类的概念及使用。

第6章是数组和字符串的处理，介绍了一维数组，二维数组和操作数组的工具类，以及Java中字符串类的使用。

第7章是异常处理，介绍了Java中异常的概念，异常的处理机制，异常处理原则及异常类的使用。

第8章是输入/输出，介绍了流的概念，Java的I/O包，字节流和字符流，文件的操作等。

第9章是用户图形界面，主要给出了Swing的功能，Swing的容器及基本组件的使用，及如何使用Action接口处理行为事件。

第10章是多线程，内容包括线程的概念，线程的生命周期，线程的创建和启动，线程的

优先级，如何控制线程等。

第 11 章是网络编程，主要包括 Java 对网络的支持，基于 TCP 的网络编程和基于 UDP 的网络编程。

本书主要由邢静宇编著，此外邱雅，单平平，许豪，钱鸽也都参与了编写工作，其中，第 1 章 Java 语言概述和第 10 章多线程由单平平编写；第 2 章 Java 语言基础，第 3 章流程控制和第 6 章数组和字符串的处理由邢静宇编写；第 4 章面向对象基础和第 9 章用户图形界面由钱鸽编写；第 5 章面向对象高级特性和第 8 章输入/输出由邱雅编写；第 7 章异常处理和第 11 章网络编程由许豪编写。

本书在编写过程中得到了各编委的大力支持，同时，同行专家及相关行业人士也提出了很多宝贵意见，在此表示感谢。

尽管编委会成员在编书过程中付出了很多，但限于编者的水平和时间仓促，错误之处在所难免，希望读者提出宝贵意见和建议。

邢静宇

2014 年 7 月

目 录

第 1 章　Java 语言概述

1.1　Java 语言的概述

Java 是一个由 Sun 公司开发而成的新一代编程语言。使用它可在各式各样不同种机器、不同种操作平台的网络环境中开发软件。不论你使用的是哪一种 WWW 浏览器，哪一种计算机，哪一种操作系统，只要 WWW 浏览器上面注明了“支持 Java”，你就可以看到生动的主页。

Sun 的 Java 语言开发小组成立于 1991 年，其目的是开拓消费类电子产品市场，例如，交互式电视、烤面包箱等。Sun 内部人员把这个项目称为 Green，那时 WorldWideWeb 还在图纸上呢。该小组的领导人是 JamesGosling，是一位非常杰出的程序员。他出生于 1957 年，于 1984 年加盟 SunMicrosystem 公司，之前在一家 IBM 研究机构工作。他是 SunNeWs 窗口系统的总设计师。也是第一个用 C 实现的 EMACS 的文本编辑器 COSMACS 的开发者。

在研究开发过程中，Gosling 深刻体会到消费类电子产品和工作站产品在开发哲学上的差异：消费类电子产品要求可靠性高、费用低、标准化、使用简单，用户并不关心 CPU 的型号，也不欣赏专用昂贵的 RISC 处理器，他们需要建立在一个标准基础之上，具有一系列可选的方案，从 8086 到 80586 都可以选取。

(1)从 C 开始

为使整个系统与平台无关，Gosling 首先从改写 C 编译器着手。但是 Gosling 在改写过程中感到仅 C 是无法满足需要的，于是在 1991 年 6 月份开始准备开发一个新的语言，那么给它起一个什么名字呢？Gosling 回首向窗外望去，看见一棵老橡树，于是建一个目录叫 Oak，这就是 Java 语言的前身(后来发现 Oak 已是 Sun 公司另一个语言的注册商标，才改名为 Java，即太平洋上一个盛产咖啡的岛屿的名字)。

Gosling 在开始写 Java 时，并不局限于扩充语言机制本身，更注重于语言所运行的软硬件环境。他要建立一个系统，这个系统运行于一个巨大的、分布的、异构的网格环境中，完成各电子设备之间的通信与协同工作。Gosling 在设计中采用了虚机器码(Virtual Machine Code)方式，即 Java 语言编译后产生的是虚拟机，虚拟机运行在一个解释器上，每一个操作系统均有一个解释器。这样一来，Java 就成了平台无关语言。这和 Gosling 设计的 Sun NeWs 窗口系统有着相同的技术味道。在 NeWs 中用户界面统一用 Postscript 描述，不同的显示器有不同的 Postscript 解释器，这样便保证了用户界面的良好的可移植性。

Patrick Naughton 也是 Sun 公司的技术骨干，曾经是 Open Windows 项目的负责人。当 Naughton 加入该小组后，整个工作进展神速。经过 17 个月的奋战，整个系统胜利完成。它是由一个操作系统、一种语言(Java)、一个用户界面、一个新的硬件平台、三块专用芯片构成的。通常情况下，这样的项目在 Sun 公司要 75 个人干三年。项目完成后，在 Sun 公司内

部做了一次展示和鉴定，观众的反应是：在各方面都采用了崭新的、非常大胆的技术。许多参观者对 Java 留下了非常深刻的印象，特别得到 Sun 的两位领导人 Scott McNealy 和 Bill Joy 的关注，但 Java 的前途未卜。

(2)Java 语言的转折点

到了 1994 年，WWW 已如火如荼地发展起来。Gosling 意识到 WWW 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，它应是一种实时性较高、可靠安全、有交互功能的浏览器。于是 Gosling 决定用 Java 开发一个新的 Web 浏览器。

这项工作由 Naughton 和 Jonathan Payne 负责，到 1994 年秋天，完成了 Web Runner 的开发工作。Web Runner 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。Web Runner 改名为 HotJava，并于 1995 年 5 月 23 日发表后，在产业界引起了巨大的轰动，Java 的地位也随之而得到肯定。又经过一年的试用和改进，Java1.0 版终于在 1996 年年初正式发表。

(3)Sun 被 Oracle 收购

在 2009 年 4 月 20 号，甲骨文宣布收购 Sun 公司。在美国时间 2010 年 3 月 17 日，在 Oracle 收购 Sun Microsystems 之后，Java 之父 James Gosling 首度在公开场合露面，他一如既往保持着对 Java 的高度关注，并表示 Java 在 Oracle 的掌管下令人放心，随后他还透露了 Java 的发展方向。

Gosling 表示，尽管目前大家看到的大多是 Oracle 在企业端 Java 的努力，但 Oracle 同样也在致力于 Java 在桌面端、嵌入式、移动领域、高性能计算机及其他系统方面的发展。他说，所有这一切的原则是网络，网络将这些应用和功能链接。

谈到企业端 Java，Gosling 表示 Java EE 6 (Java Platform, Enterprise Edition 6)将是下一代企业软件的基础，Java 社区及许多开发者在 2009 年 11 月促使了 Java EE 6 specification 的认可，并发布和升级了一些 Java API，Gosling 对此表示感谢。

Gosling 表示，Java EE 6 以模块化为中心，引入了 profiles 的概念，但是有两个 profiles，一个是 full profile，另一个是 Web profile。Web profile 是第一个被定义的 Java EE profile，对于现代 Web 应用开发它是一个功能全面的中型堆栈。

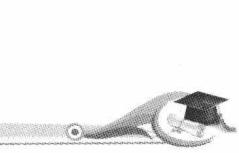
Gosling 还提到了 Java EE 6 specification 中新增的依赖注入(dependency injection)特性。依赖注入可以允许你在代码中注入依赖，你将可以使用 JDK 5 [Java Development Kit 5]中的注释特性来析出模板代码[boilerplate]，从而 EJB [Enterprise JavaBeans]的麻烦一扫而光。有趣的是，就在几年前这些问题还是 Java 社区内争论的焦点呢。

同时，Gosling 宣布了 GlassFish 应用服务器的升级新版本为 Version 3，它也是 Java EE 6 的参考实现(reference implementation)。GlassFish 是全球最流行的下载型应用服务器，每个月的下载量为 100 万。

另外，Gosling 表示 Oracle 也在积极推进 NetBeans IDE，使它积极运用到企业端、移动领域和桌面端开发。

(4)Java 语言的应用前景

工业界不少人预言："Java 语言的出现，将会引起一场软件革命"，这是因为传统的软件往往都是与具体的实现环境有关，换了一个环境就需要作一番改动，耗时费力，而 Java 语言能在执行码(二进制码)上兼容，这样以前所开发的软件就能运行在不同的机器上，只要所用



的机器能提供 Java 语言解释器即可。

Java 语言有着广泛的应用前景，大体上可以从以下几个方面来考虑其应用：

1. 软件的需求分析：可将用户的需求进行动态的、可视化描述，以提供设计者更加直观的要求。而用户的需求是各色各样的，不受地区、行业、部门、爱好的影响，都可以用 Java 语言描述清楚。

2. 软件的开发方法：由于 Java 语言的面向对象的特性，所以完全可以用 O—O 的技术与方法来开发，这是符合最新的软件开发规范要求的。

3. 软件最终产品：用 Java 语言开发的软件可以具有可视化、可听化、可操作化的效果，这要比电视、电影的效果更为理想，因为它可以做到“即时、交互、动画与动作”，要它停就停，要它继续就继续，而这是在电影与电视播放过程中难以做到的。

4. Java 语言的动画效果：远比 GUI 技术达到效果逼真，尤其是利用 WWW 提供的巨大动画资源空间，可以共享全世界的动态画面的资源。

5. 交互操作的设计（选择交互、定向交互、控制流程等）。

6. Internet 的系统管理功能模块的设计，包括 Web 页面的动态设计、管理和交互操作设计等。

7. Intranet（企业内部网）上的软件开发（直接面向企业内部用户的软件）。

8. 与各类数据库连接查询的 SQL 语句实现。

9. 其他：使用 Java 语言对开发效益、开发价值都有比较明显的影响。

1.2　Java 语言的特点

Java 的不断发展要归功于 C、C++ 和 C# 等编程语言的不断挑战。C++、C# 和 Java 等编程语言基本上都来源于 C 语言但又有很多区别。业内人士经常将 C 比作爷爷，C++ 比作儿子，C# 和 Java 等语言比作孙子。对于变量声明、参数传递、操作符、流控制等，Java 使用了和 C、C++、C# 相同的传统，而 C++ 主要是对 C 的扩展并融入了面向对象的思想，C# 和 Java 语言是纯粹地面向对象的编程语言并吸收了 C、C++ 语言的很多优点，摒弃了很多缺点，但 C# 编程语言的运行依赖于 Windows 平台，而 Java 语言不依赖于任何平台，因此使得熟悉 C、C++、C# 的程序员能够很方便地转向 Java 编程。

Position Feb 2009	Position Feb 2008	Defta in Postion	Programming Language	Ratings Feb 2009	Dafta Feb 2008	Status
1	1	⇌	Java	19.401%	−2.08%	A
2	2	⇌	C	15.837%	+0.98%	A
3	5	⬆⬆	C++	9.633%	+0.36%	A
4	3	⬇	(Visual) Basic	8.843%	−2.76%	A
5	4	⬇	PHP	8.779%	−1.11%	A
6	8	⬆⬆	C#	5.062%	+0.55%	A
7	7	⇌	Python	4.567%	−0.20%	A

Position Feb 2009	Position Feb 2008	Defta in Postion	Programming Language	Ratings Feb 2009	Dafta Feb 2008	Status
8	6	⬇⬇	Perl	4.117%	−2.09%	A
9	9	⇌	Delphi	3.624%	+0.83%	A
10	10	⇌	JavaScript	3.540%	+1.21%	A
11	11	⇌	Ruby	3.278%	+1.42%	A
12	12	⇌	D	1.259%	+0.07%	A
13	13	⇌	PL/SOL	0.988%	+0.01%	A
14	14	⇌	SAS	0.835%	−0.11%	A
15	22	⬆⬆⬆⬆⬆⬆⬆	Logo	0.813%	+0.50%	A−
16	17	⬆	Pascal	0.689%	+0.24%	A
17	29	⬆⬆⬆⬆⬆⬆⬆⬆⬆⬆⬆	ABAP	0.574%	+0.42%	B
18	21	⬆⬆⬆	ActionScript	0.539%	+0.22%	B
19	26	⬆⬆⬆⬆⬆⬆⬆	RPG(AS/400)	0.505%	+0.33%	B
20	18	⬇⬇	Lua	0.487%	+0.10%	B

图 1-1　各种编程语言的使用排名

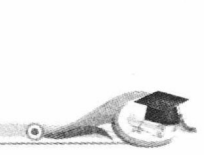
Java 语言的特点包括：

(1)简单

Java 语言简单是指这门语言既易学有好用，Java 语言的简单性主要体现在以下三个方面：第一，Java 的风格类似于 C++，因而 C++程序员是非常熟悉的。从某种意义上讲，Java 语言是 C 及 C++语言的一个变种，因此，C++程序员可以很快就掌握 Java 编程技术。第二，Java 摒弃了 C++容易引发程序错误的地方，如指针。第三，Java 提供了丰富的类库。但是，不要将简单误解为这门语言很干瘪。你可能很赞同这样的观点，英语要比阿了伯语言容易学，但这并不意味着英语就不能表达丰富的内容和深刻的思想，许多文学若贝尔奖的作品都是英文写的。如果你学习过 C++语言，你会感觉 Java 很眼熟，因为 Java 中许多基本语句的语法和 C++一样，像常用的循环语句，控制语句等和 C++几乎一样，但不要误解为 Java 是 C++的增强版，Java 和 C++是两种完全不同的语言，他们各有各的优势，将会长期并存下去，Java 语言和 C++语言已成为软件开发者应当掌握的语言。如果从语言的简单性方面看，Java 要比 C++简单，C++中许多容易混淆的概念，或者被 Java 弃之不用了，或者以一种更清楚更容易理解的方式实现，例如，Java 不再有指针的概念。

(2)面向对象

面向对象其实是现实世界模型的自然延伸。现实世界中任何载体都可以看做是对象，对象之间通过消息相互作用。如果说传统的过程式编程语言是以过程为中心、以算法为驱动的话，面向对象的编程语言则是以对象为中心、以消息为驱动。用公式表示，过程式编程语言为：程序＝算法＋数据；面向对象编程语言为：程序＝对象＋消息。Java 支持面向对象



程语言的三个概念：封装、多态性和继承。在 Java 中绝大部分成员是对象，只有简单的数字类型、字符类型和布尔类型除外。而对于这些类型，Java 也提供了相应的对象类型以便与其他对象交互操作。基于对象的编程更符合人的思维模式，使人们更容易编写程序。

(3)与平台无关

与平台无关是 Java 语言最大的优势。Java 应用程序在 Java 语言编译器中进行编译后，形成二进制代码，我们称之为字节码。但这些字节码不能直接由操作系统识别执行，而只能由一个叫 Java 虚拟机(JVM)的 Java 字节码解释器来识别。在 Java 应用程序执行时，由 Java 虚拟机(JVM)对这些字节码进行逐步解释，并转换成当前操作系统的命令，然后再执行。由于 Java 应用程序的执行只与 Java 虚拟机(JVM)直接相关，任何一台机器只要配备了 JVM，就可以运行这个程序，而不管这种字节码是在何种平台上生成的。因此，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，实现了平台无关性。其他语言编写的程序面临的一个主要问题是操作系统的变化，处理器升级以及核心系统资源的变化，都可能导致程序出现错误或无法运行。Java 的虚拟机成功地解决了这个问题，Java 编写的程序可以在任何安装了 Java 虚拟机 JVM 的计算机上正确地运行，实现了“一次写成，处处运行”。

(4)解释型

我们知道 C，C＋＋等语言，都是只能对特定的 CPU 芯片进行编译，生成机器代码，该代码的运行就和特定的 CPU 有关。例如，在 C 语言中，我们都碰到过类似下面的问题：int 型变量的值是 10，那么下面代码的输出结果是什么呢？ printf(“%d，%d”，x，x＝x＋1)。如果上述语句的计算顺序是从左到右，结果是 10，11。但是，有些机器会从右到左计算，那么结果就是 11，11。Java 不像 C＋＋，它不针对特定的 CPU 芯片进行编译，而是把程序编译为称做字节码的一个“中间代码”。字节码是很接近机器码的文件，可以在提供了 Java 虚拟机 JVM 的任何系统上被解释执行。Java 被设计成为解释执行的程序，即翻译一句，执行一句，不产生整个的机器代码程序。翻译过程如果不出现错误，就一直进行到完毕，否则将在错误处停止执行。

(5)多线程

Java 在两方面支持多线程。一方面，Java 环境本身就是多线程的。若干个系统线程运行负责必要的无用单元回收，系统维护等系统级操作；另一方面，Java 语言内置多线程控制，可以大大简化多线程应用程序开发。Java 提供了一个类 Tread，由它负责启动运行，终止线程，并可检查线程状态。Java 的线程还包括一组同步原语。这些原语负责对线程实行并发控制。利用 Java 的多线程编程接口，开发人员可以方便地写出支持多线程的应用程序，提高程序执行效率。Java 的特点之一就是内置对多线程的支持。多线程允许同时完成多个任务。实际上多线程使人产生多个任务在同时执行的错觉，因为，目前的计算机的处理器在同一时刻只能执行一个线程，但处理器可以在不同的线程之间快速地切换，由于处理器速度非常快，远远超过了人接收信息的速度，所以给人的感觉好像多个任务在同时执行。C＋＋没有内置的多线程机制，因此必须调用操作系统的多线程功能来进行多线程程序的设计。

(6)可靠性和安全性

第一，Java 是强类型的语言。要求显式的方法声明，这保证了编译器可以发现方法调用错误，保证程序更加可靠；第二，Java 不支持指针，这杜绝了内存的非法访问；第三，Java 的自动单元收集防止了内存丢失等动态内存分配导致的问题；第四，Java 解释器运行时实施检

查，可以发现数组和字符串访问的越界，第五，Java 提供了异常处理机制，程序员可以把一组错误代码放在一个地方，这样可以简化错误处理任务。

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时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检验器的检查，然后，Java 解释器将限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，当你准备从网络上下载一个程序时，你最大的担心是程序中是否含有恶意的代码，比如试图读取或删除本地机上的一些重要文件，甚至该程序是一个病毒程序等。当你使用支持 Java 的浏览器时，你可以放心地运行 Java 的小应用程序 Java Applet，不必担心病毒的感染和恶意的企图，Java 小应用程序将限制在 Java 运行环境中，不允许它访问计算机的其他部分。最后，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，值得 Java 成为安全的编程语言。

(7)动态

Java 程序的基本组成单元就是类，有些类是自己编写的，有一些是从类库中引入的，而类又是运行时动态装载的，这就使得 Java 可以在分布环境中动态地维护程序及类库，而不像 C++那样，每当其类库升级之后，相应的程序都必须重新修改，编译。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有更灵活的方式和扩展性。Java 的这些特点，使得 Java 开发的网络应用系统可以在各种平台上运行。Java 程序的应用范围，已经从小小的浏览器，扩展到涉及金融、商贸、电子、制造业、娱乐等多个领域，应用程序小到个人的单机游戏，大到跨国企业的经营管理，既有在前台客户机上运行的，也有面向后台服务器的。Java 已经成为计算机应用程序开发的主要工具。

1.3 Java 虚拟机及跨平台原理

Java 虚拟机(JVM)是 Java Virtual Machine 的缩写，它是一种抽象化的计算机，通过在实际的计算机上仿真模拟各种计算机功能来实现的。Java 虚拟机有自己完善的硬件架构，如处理器、堆栈、寄存器等，还具有相应的指令系统。JVM 屏蔽了与具体操作系统平台相关的信息，使得 Java 程序只需生成在 Java 虚拟机上运行的目标代码(字节码)，就可以在多种平台上不加修改地运行。

(1)Java 虚拟机的体系结构

我们知道，一个 JVM 实例的行为不光是它自己的事，还涉及到它的子系统、存储区域、数据类型和指令这些部分，它们描述了 JVM 的一个抽象的内部体系结构，其目的不光规定实现 JVM 时它内部的体系结构，更重要的是提供了一种方式，用于严格定义实现时的外部行为。每个 JVM 都有两种机制，一个是装载具有合适名称的类(类或是接口)，叫做类装载子系统；另外的一个负责执行包含在已装载的类或接口中的指令，叫做运行引擎。每个 JVM 又包括方法区、堆、Java 栈、程序计数器和本地方法栈这五个部分，这几个部分和类装载机制与运行引擎机制一起组成的体系结构，如图 1-2 所示。

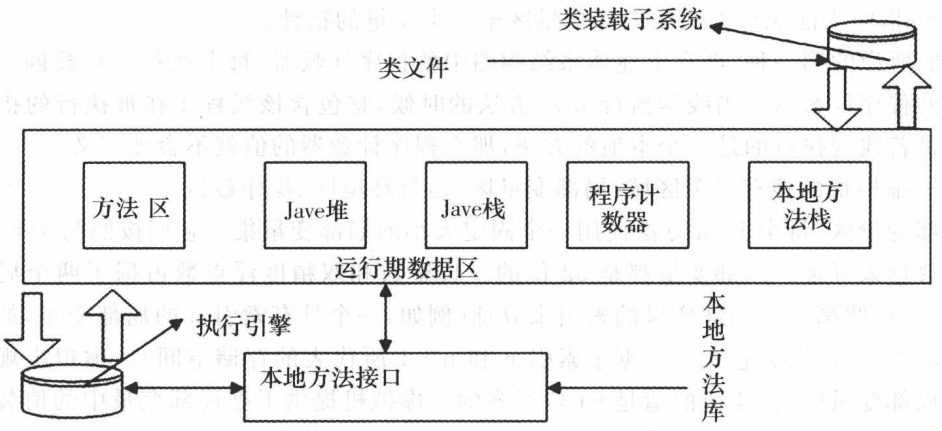


图 1-2　JVM 的体系结构

JVM 的每个实例都有一个它自己的方法域和一个堆，运行于 JVM 内的所有的线程都共享这些区域；当虚拟机装载类文件的时候，它解析其中的二进制数据所包含的类信息，并把它们放到方法域中；当程序运行的时候，JVM 把程序初始化的所有对象置于堆上；而每个线程创建的时候，都会拥有自己的程序计数器和 Java 栈，其中程序计数器中的值指向下一条即将被执行的指令，线程的 Java 栈则存储为该线程调用 Java 方法的状态；本地方法调用的状态被存储在本地方法栈，该方法栈依赖于具体的实现。下面分别对这几个部分进行说明。

执行引擎处于 JVM 的核心位置，在 Java 虚拟机规范中，它的行为是由指令集所决定的。尽管对于每条指令，规范很详细地说明了当 JVM 执行字节码遇到指令时，它的实现应该做什么，但对于怎么做却言之甚少。Java 虚拟机支持大约 248 个字节码。每个字节码执行一种基本的 CPU 运算，例如，把一个整数加到寄存器，子程序转移等。Java 指令集相当于 Java 程序的汇编语言。

对于本地方法接口，实现 JVM 并不要求一定要有它的支持，甚至可以完全没有。Sun 公司实现 Java 本地接口(JNI)是出于可移植性的考虑，当然我们也可以设计出其他的本地接口来代替 Sun 公司的 JNI。但是这些设计与实现是比较复杂的事情，需要确保垃圾回收器不会将那些正在被本地方法调用的对象释放掉。

Java 方法区与传统语言中的编译后代码或是 Unix 进程中的正文段类似。它保存方法代码(编译后的 java 代码)和符号表。每个类文件包含了一个 Java 类或一个 Java 界面的编译后的代码。可以说类文件是 Java 语言的执行代码文件。为了保证类文件的平台无关性，Java 虚拟机规范中对类文件的格式也作了详细的说明。

Java 的堆是一个运行时数据区，类的实例(对象)从中分配空间，它的管理是由垃圾回收来负责的：不给程序员显式释放对象的能力。Java 不规定具体使用的垃圾回收算法，可以根据系统的需求使用各种各样的算法。

Java 虚拟机的寄存器用于保存机器的运行状态，与微处理器中的某些专用寄存器类似。Java 虚拟机的寄存器有四种：

pc：Java 程序计数器；

optop：指向操作数栈顶端的指针；

frame：指向当前执行方法的执行环境的指针；

vars：指向当前执行方法的局部变量区第一个变量的指针。

我们所说的第一种，即在上述体系结构图中的程序计数器，每个线程一旦被创建就拥有了自己的程序计数器。当线程执行 Java 方法的时候，它包含该线程正在被执行的指令的地址。但是若线程执行的是一个本地的方法，那么程序计数器的值就不会被定义。

Java 虚拟机的栈有三个区域：局部变量区、运行环境区、操作数区。

局部变量区：每个 Java 方法使用一个固定大小的局部变量集。它们按照与 vars 寄存器的字偏移量来寻址。局部变量都是 32 位的。长整数和双精度浮点数占据了两个局部变量的空间，却按照第一个局部变量的索引来寻址(例如，一个具有索引 n 的局部变量，如果是一个双精度浮点数，那么它实际占据了索引 n 和 n+1 所代表的存储空间)。虚拟机规范并不要求在局部变量中的 64 位的值是 64 位对齐的。虚拟机提供了把局部变量中的值装载到操作数栈的指令，也提供了把操作数栈中的值写入局部变量的指令。

运行环境区：在运行环境中包含的信息用于动态链接，正常的方法返回以及异常捕捉。

操作数栈区：机器指令只从操作数栈中取操作数，对它们进行操作，并把结果返回到栈中。选择栈结构的原因是：在只有少量寄存器或非通用寄存器的机器(如 Intel486)上，也能够高效地模拟虚拟机的行为。操作数栈是 32 位的。它用于给方法传递参数，并从方法接收结果，也用于支持操作的参数，并保存操作的结果。

上面对虚拟机的各个部分进行了比较详细的说明，下面通过一个具体的例子来分析它的运行过程。

虚拟机通过调用某个指定类的方法 main 启动，传递给 main 一个字符串数组参数，使指定的类被装载，同时链接该类所使用的其他的类型，并且初始化它们。例如对于程序：

```
class HelloApp
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
        for (int i = 0; i < args.length; i++ )
        {
            System.out.println(args[i]);
        }
    }
}
```

编译后在命令行模式下键入：java HelloApp run virtual machine

将通过调用 HelloApp 的方法 main 来启动 java 虚拟机，传递给 main 一个包含三个字符串“run”、“virtual”、“machine”的数组。现在我们略述虚拟机在执行 HelloApp 时可能采取的步骤。

开始试图执行类 HelloApp 的 main 方法，发现该类并没有被装载，也就是说虚拟机当前不包含该类的二进制代表，于是虚拟机使用 ClassLoader 试图寻找这样的二进制代表。如果这个进程失败，则抛出一个异常。类被装载后同时在 main 方法被调用之前，必须对类 HelloApp 与其他类型进行链接然后初始化。链接包含三个阶段：检验，准备和解析。检验检查

被装载的主类的符号和语义，准备则创建类或接口的静态域以及把这些域初始化为标准的默认值，解析负责检查主类对其他类或接口的符号引用，在这一步它是可选的。类的初始化是对类中声明的静态初始化函数和静态域的初始化构造方法的执行。一个类在初始化之前它的父类必须被初始化。整个过程如图 1-3 所示：

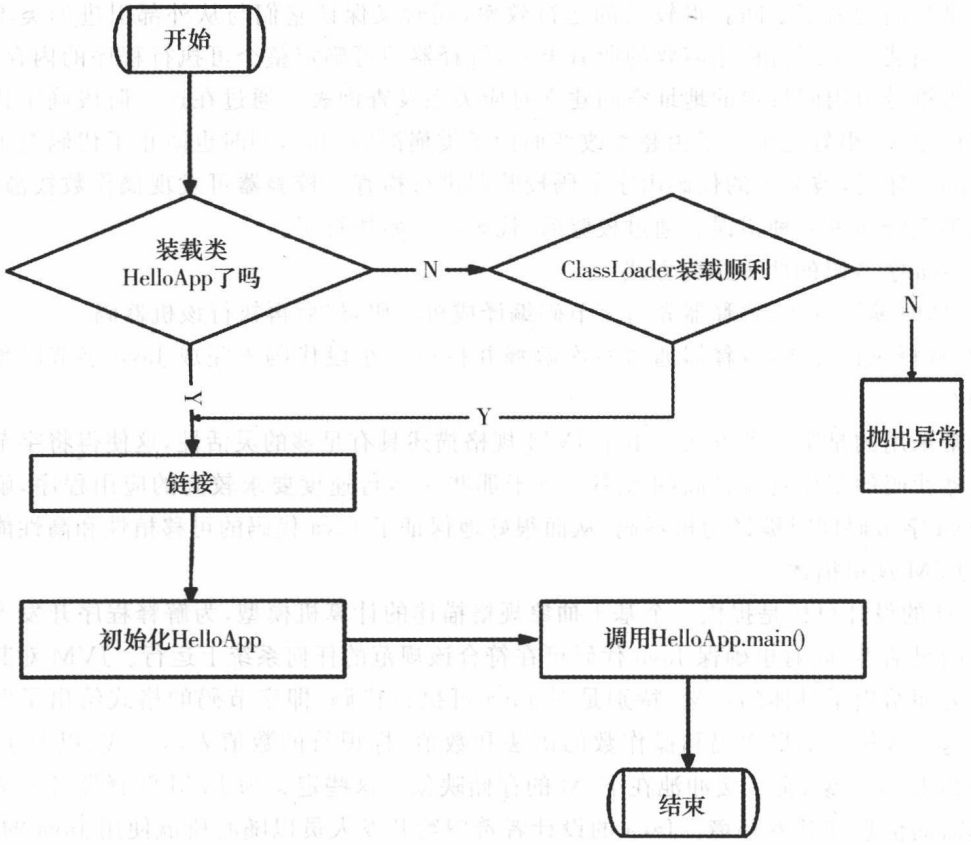


图 1-3　虚拟机的运行过程

(2) Java 跨平台的原理

Java 的跨平台是通过 Java 虚拟机(JVM)来实现的。

1. Java 源文件的编译过程

Java 应用程序的开发周期包括编译、下载、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。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 编译器却不将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建内存布局，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。

2. Java 解释器的执行过程

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。解释器为符号引用同特定的地址空间建立对应关系及查询表。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，非法数据类型转换等多种错误。通过校验后，代码便开始执行了。

3. Java 字节码的两种执行方式

(1)即时编译方式：解释器先将字节码编译成机器码，然后再执行该机器码。

(2)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。

通常采用的是第二种方法。由于 JVM 规格描述具有足够的灵活性，这使得将字节码翻译为机器代码的工作具有较高的效率。对于那些对运行速度要求较高的应用程序，解释器可将 Java 字节码即时编译为机器码，从而很好地保证了 Java 代码的可移植性和高性能。

4. JVM 规格描述

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、以及 Java 类文件中的 Java 对象、常量缓冲池在 JVM 的存储映象。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 是为 Java 字节码定义的一种独立于具体平台的规格描述，是 Java 平台独立性的基础。

5. Java 程序执行与 C/C++程序执行的对比分析

如果把 Java 源程序想象成我们的 C++源程序，Java 源程序编译后生成的字节码就相当于 C++源程序编译后的 80×86 的机器码(二进制程序文件)，JVM 虚拟机相当于 80×86 计算机系统，Java 解释器相当于 80×86CPU。在 80×86CPU 上运行的是机器码，在 Java 解释器上运行的是 Java 字节码。Java 解释器相当于运行 Java 字节码的“CPU”，但该“CPU”不是通过硬件实现的，而是用软件实现的。Java 解释器实际上就是特定的平台下的一个应用程序。只要实现了特定平台下的解释器程序，Java 字节码就能通过解释器程序在该平台下运行，这是 Java 跨平台的根本。当前，并不是在所有的平台下都有相应的 Java 解释器程序，这也是 Java 并不能在所有的平台下都能运行的原因，它只能在已实现了 Java 解释器程序的平台下运行。

1.4 Java 的开发环境

JDK 是 Java 开发工具包(Java Development Kit)的缩写，是一种最基本的工具和开发环