重点大学计算机教材

HZ BOOKS

Java程序设计教程

辛运帏　饶一梅 编著

南开大学

机械工业出版社
China Machine Press

重点大学计算机教材

Java程序设计教程

辛运帏　饶一梅 编著
南开大学

机械工业出版社
China Machine Press

本书从Java语言的基本特点入手，全面介绍Java语言的基本概念和编程方法，并深入讨论Java的高级特性。全书共分为11章，涵盖Java中的基本语法和数据类型，同时涉及类的概念、异常处理、用户界面设计等内容。此外，本书还对Applet小应用程序、I/O数据流及线程等内容做了介绍。本书内容详尽，并配合大量示例，在每章的最后均列出若干习题，供读者参考。

本书既可供高等院校本科生用作Java程序设计课程的教材，也可作为程序设计自学者和专业技术人员的参考书。

图书在版编目（CIP）数据

Java程序设计教程/辛运帏，饶一梅编著. －北京：机械工业出版社，2007.8
（重点大学计算机教材）
ISBN 978-7-111-21780-0

Ⅰ.J… Ⅱ.①辛… ②饶… 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（2007）第096485号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：王 璐
北京瑞德印刷有限公司印刷 · 新华书店北京发行所发行
2007年8月第1版第1次印刷
184mm × 260mm · 17.75印张
定价：28.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书电话（010）68326294

前　言

大千世界事物万千，都有其内在规律。一个事物从面世到消亡会经历一个或长或短的过程，如果能顺应世界潮流，适应环境变化，则它的生存期就会延长，反之就会迅速逝去。正所谓顺者昌，逆者亡。这个规律是如此的残酷，却也是如此的现实。

计算机的面世，对我们日常工作、生活方式的改变不是三言两语就能说得清楚的。不管是与天地自然相比，还是与人类几千年的文明相比，计算机的历史都短暂得可以被忽略。即使这样，这个领域中已经有太多的东西被淘汰，但又有更多的新技术面世。在程序设计语言领域，Java正是这些新技术的突出代表。

Java的面世已经超过十年了，就一种语言来讲，这样的历史不可谓不长，不过可喜的是，Java并没有表现出“老态”。相反地，它不断地完善自身、扩展新功能、增加新元素，使它的应用领域越来越广。从小的应用程序设计，到大的软件系统开发，到处都是Java发挥潜能的舞台。例如，目前主流的软件设计架构J2EE就是使用Java技术开发企业级应用的一种事实上的工业标准。

Java应用面的拓展，导致Java语言的学习热潮并没有因为时间的推延而低沉下去，相反，越来越多的人开始学习Java，了解Java，试图使用Java完成各类任务。Java语言逐步成为程序员首选的程序设计开发工具。

本书就是为了满足广大读者的需求而编写的。我们注意到Java语言仍在不断完善之中，它的新旧版本之间存在若干差别。所以在编写过程中，我们不但对旧版本中的内容有所介绍，更主要的是介绍新版本中的一些特点，这样读者既可以了解Java语言的发展变化过程，又可以保证所设计程序保持代码级的兼容。

现在越来越多的高校为计算机专业的学生开设了Java语言这门课程，许多非计算机专业的学生也作为选修课选学Java语言。国内外的教科书种类繁多，定位也各有特长。基于这样的情况，本书在编写时注重吸取各教材的特点，分析高校中本科阶段学生的具体情况，并结合编者在教学过程中的感受，旨在向读者奉献一本有针对性的教材。

本书从Java语言的基本特点入手，同时配合介绍编程的思想和方法。第1~3章以介绍Java的特点及基本语法元素为主。Java语言和C++语言非常类似，所以如果读者熟悉C++语言的话，则会对第2、3章中的内容有似曾相识的感觉。

第4章和第6章讨论了面向对象程序设计语言中的重要概念——类，及其相关的知识。类的内容非常多，我们分两章来介绍。第4章侧重概念级内容的介绍，第6章则侧重应用方面的介绍，也就是如何使用类这种结构来编写代码满足我们的要求。

由于数组、字符串等的特殊性，我们既没有把它们混到基本语法结构的介绍中，也没有把它们归到类的一章中，而是单独成章——第5章。这似乎与Java语言的提出者们的最初考虑不谋而合。

Java的重要特性在接下来的几章中做了较全面的介绍。第7章所讲述的异常处理是保证Java程序鲁棒性的重要手段，也是它的特色之一。图形用户界面是应用程序的门面，它的设计手段将在第8章介绍。第9~11章分别介绍了Applet、I/O数据流及线程的概念，这些都是编写

应用程序时不可缺少的成分和技术手段。

本书内容涵盖了Java的基本内容，是进一步使用Java进行技术开发的基础，愿本书能成为读者进入Java殿堂的铺路石。

本书配有很多例题，可以帮助读者了解书中介绍的知识点。例题分两类：一类以“例”来标注，多是不完整的程序代码段、示例讲解等；另一类以“程序”来标注，基本上都是完整的可执行代码。对后一类程序，本书均在J2SE 5.0平台下完成调试，书中的大部分程序还给出了运行结果。读者可以在自己的机器上尝试运行书中的程序。

本书在每章的最后列出若干习题供读者参考。教师可根据各学校的具体课时安排，全部或选讲书中的内容。本书不仅适合Java语言的初学者使用，也可作为专业人员的参考书。

计算机技术是不断发展、不断完善的技术，Java语言也是如此。在本书编写的过程中，Java语言仍没有停止它完善的脚步。本书中有些内容是以目前的版本为标准，当推出更新版本的Java时，读者应参考最新标准。

在本书的编写过程中，编者得到了南开大学信息技术科学学院卢桂章教授、吴功宜教授、陈有祺教授、刘璟教授、周玉龙教授、朱耀庭教授等的亲切关怀和悉心指导。在此表示深深的感谢，也同样感谢读者在众多的Java参考书中选中了本书。

本书由辛运帏、饶一梅共同编写，温小艳调试了部分代码。由于作者的水平有限，书中难免有错误和不妥之处，恳请广大读者特别是同行专家们批评指正。

编者

2007年5月于南开园

目　录

第1章 Java语言入门

1.1 什么是Java语言

学习程序设计语言却没听说过Java的人已经是屈指可数了，但能如数家珍地说清Java的来龙去脉也并非易事，能够熟练运用这种语言去完成特定任务的人都是这个领域的高手。在本书的开篇部分，我们先了解一下Java的产生和发展历程。Java语言的初次亮相是在1991年，当时是美国Sun Microsystems（升阳计算机系统有限）公司的Jame Gosling领导的一个小组专为在家用消费类电子产品上进行交互式操作而设计的。开发人员最初给这个语言起名为Oak，面世之初它的影响与它的名字一样表现平平。

随着互联网的出现及广泛使用，人们需要面向网络应用的语言的呼声日趋强烈，设计人员及时改变了Oak语言的设计初衷，使这个语言能够开发网络应用程序，并能将程序嵌入到HTML页面中。这一改变可谓是划时代的，它给用户带来生动的界面及交互方式，并彻底扭转了Oak语言问世之初的尴尬境遇。设计人员又借神来之笔将Oak语言改名为Java，并配了一杯冒着热气的咖啡图案作为它的标志。1995年，从内到外都已面目一新的Java正式推出，立即引起了业界的巨大轰动。

下面简单看看Java发展过程中的几个里程碑事件：1995年5月23日Sun公司发布了JDK 1.0，Java语言诞生；1996年1月第一个开发包JDK 1.0问世；1996年4月10个最主要的操作系统供应商申明将在其产品中嵌入Java技术；1997年2月18日Sun公司公布了新版本的开发包JDK 1.1；1997年4月2日，JavaOne会议召开，参与者逾一万人，这次会议的规模创当时全球同类会议规模之纪录；1998年12月4日公布了JDK 1.2开发包，这又是一个新的里程碑，从此Java转向Java 2 Platform；1998年12月8日Java 2企业平台J2EE发布；1999年6月Sun公司发布Java的三个版本，即标准版、企业版和微型版（J2SE、J2EE、J2ME），为程序员提供更多的选择；2000年5月8日JDK 1.3发布，主要是对各种已有API（应用程序编程接口）的加强和对新API的拓展；2000年5月29日JDK 1.4发布；2001年9月24日J2EE 1.3发布；2002年2月26日J2SE 1.4发布，主要考虑和修改了平台的性能，自此Java的计算能力有了大幅提升；2004年9月30日J2SE 1.5发布，这是Java语言发展史上的又一里程碑事件，为了表示这个版本的重要性，J2SE 1.5更名为J2SE 5.0；2005年6月，JavaOne大会召开，Sun公司公开Java SE 6。此时，Java的各种版本已经更名并取消其中的数字“2”，J2EE更名为Java EE，J2SE更名为Java SE，J2ME更名为Java ME。

Java语言在面世之初即受到广泛关注，更值得一提的是，最初的成功并没有让该语言进一步发展和完善的脚步停止。Sun公司不断丰富Java类库，为开发人员提供功能越来越强大的软件开发工具包（Software Development Kit，JDK），语言的性能随之日益提高，语言的应用面也得到了拓宽。有关Sun公司目前提供的工具包版本，读者可以从Sun公司的网站http://java.sun.com/javase上查询当前最新的版本信息，并获取相关资料。

Java究竟是怎样的一种语言呢？实际上，Java是一种功能强大的程序设计语言，既是开发环境，又是应用环境，它代表一种新的计算模式。下面以图1-1说明Java语言的基本概念。

Java是简单的面向对象的语言，具有分布性、安全性和鲁棒性的特点。它的最初版本是解释执行的，现在的版本中增加了编译执行，所以它兼备了高性能；它是多线程的、动态的语言；最主要的是它生成与平台无关的二进制格式的类文件，而由于有Java虚拟机，因此Java的代码可以跨平台执行，解决了困扰软件界多年的软件移植问题；Java语言既具有C++的功能，又具备对类型进行严格检查的安全特性，并且有越来越丰富的软件包提供给程序开发人员。

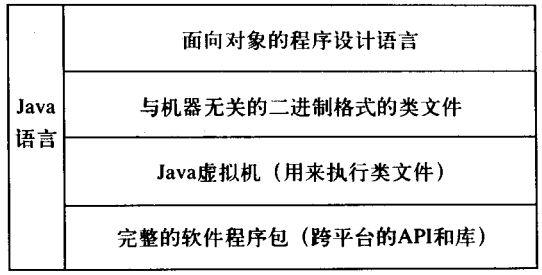

图1-1 Java的基本概念描述

1.1.1 Java语言的特点

自诞生之日起就引发了全世界软件界关注的Java语言，标志着一个新的计算时代——Java计算时代的到来。Java众多的突出特点使得它受到了大众的欢迎。实际上，Java符合目前面向对象程序设计的主流思想，它与Web及Internet结合紧密，支持动画、声音等功能，能实时处理信息。

归纳起来，Java语言有以下一些显著特点：

1. 语法简单，功能强大，强调面向对象的特性

Java是一种面向对象的程序设计语言，它的全部语句都含在类的范畴内。这是一种全新的设计格式，但具体到Java的各种语句却是沿袭了那些主流语言的写法，特别是类似于C++的语法，熟悉C++的程序设计人员对它不会感到陌生。Java比C++更简单一些，它去掉了C++中不常用且容易出错的地方。例如，Java中没有指针、结构和类型定义等概念，不再有全局变量，没有#include和#define等预处理器，也没有多重继承的机制。程序员不用自己释放占用的内存空间，因此不会引起因内存混乱而导致的系统崩溃。

Java是一个纯粹的面向对象的语言，强调的是面向对象的特性，对软件工程技术能够提供很强的支持。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及动态的接口模型。与其他面向对象的语言一样，Java具备继承、封装及多态性这些通常的特性，更提供了一类对象的原型，程序员可以通过继承机制，实现代码的复用。另外Java的继承机制独特，在设计时去掉了不安全的因素，因此使用Java可以编制出非常复杂但逻辑清晰的系统。值得一提的是，Java的解释器只占用很少的内存，适合在绝大多数类型的机器上运行。

2. 分布式与安全性

从诞生之日起Java就与网络联系在一起，它强调网络特性，内置TCP/IP、HTTP、FTP协议类库，便于开发网络应用系统。Java程序在语言定义阶段、字节码检查阶段及程序执行阶段进行的三级代码安全检查机制，对参数类型匹配、对象访问权限、内存回收、Java小应用程序的正确使用等都进行了严格的检查和控制，可以有效地防止非法代码的侵入，阻止对内存的越权访问，能够避免病毒的侵害。

3. 与平台无关

提到Java，不得不提一句著名的口号“一次编写，到处运行”。这句话很好地反映了Java的平台无关性，也说明Java实现了编程人员多年的梦想。Java可以跨平台使用的特点使其更适合于网络应用。Java语言规定了统一的数据类型，有严格的语法定义，它最后的目标码都是一致的，不会依据机器的不同而不同，也不会依赖于编译器。

Java的这个特性，为Java程序跨平台的无缝移植提供了很大的便利。Java编译器将Java程序编译成二进制代码，即字节码（bytecode）。不同的操作系统有不同的虚拟机（JVM），虚拟机类似于一个小巧而高效的CPU。字节码就是虚拟机的机器指令，与平台无关。字节码有统一的格式，不依赖于具体的硬件环境。在任何安装Java运行时环境的系统上，都可以执行这些代码。也就是说，只要安装了Java运行时系统，Java程序就可在任意的处理器上运行。这些字节码指令对应于Java虚拟机中的表示，Java解释器得到字节码后，对它进行转换，使之能够在不同的平台运行。运行时环境针对不同的处理器指令系统，把字节码转换为不同的具体指令，保证了程序的“到处运行”。

4. 解释和编译两种运行方式

Java程序可以经解释器得到字节码，所生成的字节码经过了精心设计和优化，因此运行速度较快，从而突破了以往解释性语言运行效率低的瓶颈。在现在的Java版本中又加入了编译功能（即just-in-time编译器，简称JIT编译器），用生成器将字节代码转换成本机的机器代码，然后以较高速度执行，使执行效率大幅度提高，基本达到了编译语言的水平。

5. 多线程

单线程程序某一时刻只能做一件事情，而多线程程序允许在同一时刻同时做多件事情。大多数高级语言（包括C、C++等）都不支持多线程，用它们只能编写顺序执行的程序（除非有操作系统API的支持）。Java内置了语言级多线程功能，提供现成的Thread类，只要继承这个类就可以编写多线程的程序，使用户程序并行执行。Java提供的同步机制可保证各线程对共享数据的正确操作，完成各自的特定任务。在硬件条件允许的情况下，这些线程可以直接分布到各个CPU上，充分发挥硬件性能，减少用户等待的时间。

通过使用多线程，程序设计者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，这样就很容易实现网络上的实时交互行为。

6. 动态执行

Java执行代码是在运行时动态载入的，程序可以自动进行版本升级。在网络环境下，Java语言编写的代码用于瘦客户机架构可减少维护工作。另外，类库中增加的新方法和其他实例不会影响到原有程序的执行。

7. 丰富的API文档和类库

Java为用户提供了详尽的API文档说明。Java开发工具包中的类库包罗万象，应有尽有，这使程序员的开发工作可以在一个较高的层次上展开。这也正是Java受欢迎的重要原因之一。

1.1.2 Java语言的新特点

Java编程语言是一种通用的面向对象的语言，具有并行性、以类为基准的强类型特点。除原来的基本特点之外，在新版本中还加入了如下新特性：

1. 新的泛型类型和枚举类型

一个集合可以存放任何类型的对象，虽然很便利，但相应地从集合中获取对象的时候我们也不得不对它们进行强制的类型转换。从Java 5.0版本起引入了泛型类型（Generic），它允许指定集合内元素的类型，这样可以在编译时进行类型检查，这是强类型语言给程序设计人员提供的方便之处。

从JDK 5.0版本起，新加入了一个基于“类”的数据类型——枚举类型（Typesafe Enums），程序员可以使用任意的方法和字段来创建一个枚举类型。

2. 增强的for循环

在集合的API中，Iterator类是使用率非常高的一个类，它提供了一种顺序访问集合元素的能力。新增强的for循环能在顺序访问集合元素的时候代替Iterator，简化对集合和数组的操作。

3. 自动装箱和自动拆箱

所谓自动装箱（Autoboxing）是指基本类型自动转为包装类，如int转为Integer。自动拆箱（Unboxing）是指包装类自动转为基本类型，如Integer变为int。自动装箱/拆箱功能大大方便了基本类型数据和它们的包装类型的使用，使代码更加简洁易懂。

4. 不定参数

不定参数（Vararg）机制允许在一个方法中传入多个不定数量的参数，这样可以避免在一个方法中需要接收变长的参数时还要人为地将其封装到一个数组中。

5. 静态引入

以前版本中要使用静态成员（方法和变量）时，必须给出提供这个方法的类。静态引入（Static Import）可以使被导入类的所有静态变量和静态方法在当前类直接可见，使用这些静态成员时无需再给出它们的类名。

6. 元数据

元数据（Metadata）提供了使用原程序的注解来产生样本代码的能力，这是一种说明性的编程风格：只要程序员说应该干什么，工具就可以生成代码，而且在此环境下也不需要一些另外的文件来记录源文件的改变，所有的信息都可以保留在源文件中。

除了上面这些突出特点以外，Java还在虚拟机、基类、集成类、用户界面等方面有一些新的改进，例如：提供了格式化输出（Formatter）的工具，用于调整数字型、字符串、日期数据的通用格式；还提供了格式化的输入（Scanner）方法，java.util.Scanner类可以将文本转换为原语（Primitive）或者字符串。在程序流程处理方面，新版本提供了并行工具（Concurrency Utility），其中有三个包java.util.concurrent、java.util.concurrent.atomic和java.util.concurrent.locks是为了专用于开发并行的类和应用而提供的。增加的这些包对于核心类库来说，可以使程序员不再需要手工维护这些工具，同时这些包利用处理器提供的并行支持为高级的并行程序设计提供底层的原语，使程序员可以用Java语言实现一些高性能、高伸缩性的并行算法。

另外，Java在字符集支持方面提供了国际化处理，其中主要是对增补字符的支持。目前字符串的处理是基于Unicode标准4.0版，这影响到java.lang包中的Character和String类。Java在平台的监控和管理方面，其功能也有显著改善。新增的java.lang.management包提供了对Java虚拟机的监控和管理的接口的API；新增的java.util.logging.LoggingMXBean接口为记录设备（logging facility）提供了管理接口；Java虚拟机有内置的测试设备让用户对它使用JMX进行监控和管理。另外，在用于数据源连接的JDBC的支持上，Java也有所更新。

1.1.3 Java的几种特殊机制

1. Java虚拟机

Java虚拟机（Java Virtual Machine, JVM）是运行Java程序必不可少的机制。Java虚拟机规范中给出了JVM的定义：JVM是在一台真正的机器上用软件方式实现的一台假想机。这个规范提供了编译所有Java代码的硬件平台。因为编译是针对假想机的，所以该规范能让Java程序独立于平台，并适用于每个具体的硬件平台，以保证能运行为JVM编译的代码。JVM使用的代码存储在.class文件中。JVM的某些指令很像真正的CPU指令，包括算术运算、流控制和数组元素访问等。

我们都知道，一台没有安装任何软件系统的计算机称为裸机，这样的机器是不具备任何

“计算”功能的，计算机必须在安装了相应的系统之后才能具备各种各样的功能。由于计算机所用芯片的差异，选择安装的系统也会不同，因此同一种软件产品往往只能安装于某一类或某几种计算机上，这就是我们通常所说的平台依赖性。程序员在编写软件时，也会根据所运行机器的独特性，在代码级做特殊处理。

通常，编写好的代码并不能直接运行在系统的CPU上，需要经过编译、链接等系统的特殊处理，生成可执行文件，也就是机器能够识别的机器码，然后才能正确运行。对于Java语言来讲，用来接受编译后指令的系统正是JVM，也就是说编译后的Java程序指令由JVM执行，因此可以把JVM看作编译后的Java程序和硬件系统之间的接口，它屏蔽了硬件平台的差异性，正因为有了这个机制，Java做到了“一次编写，到处运行”。而从程序员角度来看，JVM是一个虚拟的处理器，它不仅解释执行编译后的Java指令，而且还能进行安全检查。它是Java程序能在多平台间进行无缝移植的可靠保证，同时也是Java程序的安全检验引擎。

JVM可以使用软件方式实现，也可以使用硬件方式实现。它的具体实现包括：指令集(等价于CPU的指令集)、寄存器组、类文件格式、栈、垃圾收集堆、内存区。

Java程序经编译后生成的代码称为字节码，这也是JVM要识别的代码。Java是强类型语言，即它预定义的各种基本数据类型的长度都是不变的，不会根据机器的不同而不同，所以类型检查可以在编译时由字节码校验器来完成。它的字节码是压缩形式的，这些都有助于提高运行效率。

Java虚拟机规范对运行时数据区域的划分及字节码的优化并不做严格的限制，它们的实现依平台的不同而有所不同，这可以充分发挥各硬件系统的优势。JVM的实现叫做Java运行时系统或运行时环境（Runtime Environment），简称为运行时。Java运行时必须遵从Java虚拟机规范，这样Java编译器生成的类文件才可被所有Java运行时系统下载。嵌入了Java运行时系统的应用程序，就可以执行Java程序了。目前有许多操作系统和浏览器都嵌入了Java运行时环境。

Java解释器经过不断的优化，字节码的执行速度已有很大提高，改善了执行效率。另外，在具体技术处理上，Sun公司也有许多可圈可点之处。在提高执行效率方面比较有代表的是Hotspot技术，它提供对代码的运行时选择，为的是从根本上解决Java程序的效率问题。在程序执行时，Hotspot对每个字节码指令进行分析，根据它的执行次数，动态决定它的执行方式。比如，一段指令需要多次重复执行，则立即编译为可执行代码。如果是只执行一次的简单指令，且解释执行的效率更高，则使用解释执行的方式。有了这项技术，Java的效率问题基本上可以得到解决。

此外，JIT编译器也是另一个技术特点，在字节码执行之前可以先经过JIT编译器进行编译，生成针对具体平台的本机执行代码。它的执行效率可比解释执行的效率大幅度提高。现在许多厂商都提供JIT编译器，这项技术已非常成熟。由于字节码与平台无关，所以经过编译的Java仍不失跨平台的特点。

2. 垃圾收集

程序运行必须要有相应的内存空间，高级程序设计语言都为用户提供了相应的内存管理机制，程序员可以根据需要动态地分配或是释放内存。虽然这种机制给用户提供了很大的自由度，但也带来了潜在的危险。这种危险表现在两方面，一是程序员忘记释放不再使用的内存，二是程序员释放了仍在使用的内存。第一种情况最多是引发“内存漏洞”问题，导致内存空间的不足，并不会引来致命性的错误。第二种情况往往会导致系统的崩溃，并不可恢复。

Java针对这个问题提出了新的解决办法：只允许程序员根据需要分配内存，而释放内存

的任务交给系统来完成。系统有一套完善的监督机制，提供一个后台系统级线程，记录每次内存分配的情况，并统计每个内存指针的引用次数。当发现有不再使用的内存时，会及时将其释放，而仍在使用的内存，绝不会提前释放。垃圾收集线程只利用Java虚拟机运行时环境闲置的时间完成内存释放，并不会给系统造成负担。从用户角度来看，在Java程序生存期内，垃圾收集将自动进行，无需程序员释放内存，从而消除了内存漏洞。编制程序时，程序员可以将注意力放在更需注意的地方，不仅如此，程序的调试也更加方便。

3. 代码安全

应该说，Java是目前最安全的程序设计语言之一，这归功于它的三级代码安全检查机制。也就是它的三个环节：语言定义环节、字节码检查环节及程序执行环节。图1-2说明了Java程序环境，我们来看看它是如何在这个环境中保证代码安全的。

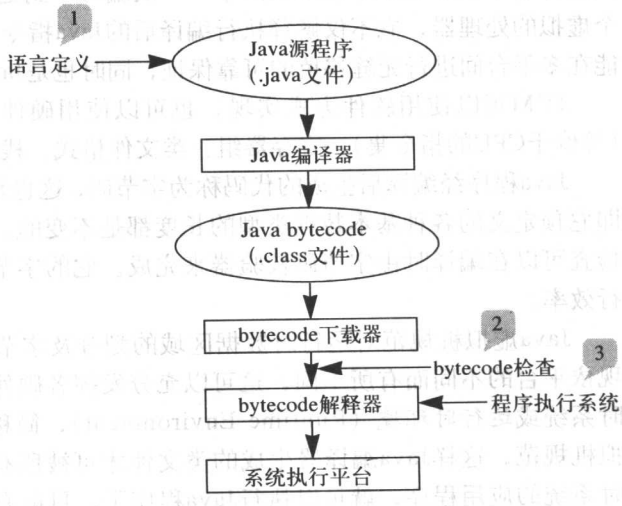

图1-2 Java代码安全

语言定义环节是最初的屏障。Java是强类型语言，不仅规定了各基本数据类型的长度，还规定了完善的访问权限（通过public、protected、private、final等修饰符定义），对于变量、常量的检查都非常到位。语言中取消了指针、预定义等，提供自动垃圾收集机制，增强了鲁棒性。

在做字节码检查时，根据对象的访问权限进行相应的检查，并保证方法调用时参数类型一致，同时控制系统堆栈溢出。

在程序执行环节主要是对Java小应用程序Applet进行安全检查，限制它的不正当使用，保证本机的安全。因为Applet是通过网络从其他机器上下载到本机执行的，程序中可能隐藏了某些非法操作，所以系统要对其进行严格的三级代码安全检查，即验证、分析和跟踪监测。第一级验证是在类下载时完成的，检查从哪里下载文件、是否有权限进入本机系统。然后进行第二级检查——字节码校验，此时要分析下载的字节码是否合乎规则。如果字节码的格式不合要求，则拒绝执行，只有完全合乎规则的字节码才允许执行。执行的时候，安全管理器始终监测所执行的每步操作，检查其合法性。Java的安全检查可以全面提高操作系统的安全等级，经过这三级安全检查的Java程序不会受到病毒的侵害。目前很多系统在做安全检查时只做第一步，即看代码下载的权限是否合乎要求。与之相比，Java的三级安全检查机制要完备得多，它不仅进行静态检查，还要进行动态跟踪。

具体地说，在不同的版本中，Java又有不同的安全机制。在最初的JDK 1.0版本中，安全模型是“沙箱”模型，在该模型下，那些从网络上下载的代码只能在一个受限的环境中运行，这个环境像个箱子一样限制了代码能访问的资源。到了JDK 1.1版本，就提出了“签名Applet”的概念。有正确签名的Applet视同本地代码一样，可以使用本地的资源；没有签名的Applet还与前一版本一样，只在沙箱中运行。

在Java 2平台下，安全机制又有较大改善。它允许用户自己设定相关的安全级别。另外，对于应用程序，也采取了和Applet一样的安全策略，程序员可以根据需要对本地代码或是远

程代码进行设定，以保证程序更安全高效地运行。

1.2 基本的Java应用程序介绍

1.2.1 开发环境的安装

要运行一个Java应用程序，必须先安装开发环境。开发环境包括一个编辑软件及开发Java程序必需的JDK工具。编辑软件可以使用你机器上的任何一个文本编辑器，例如Notepad、TextPad、UltraEdit等。也可以安装集成的开发环境，即所谓的IDE。比较著名的IDE有Java WorkShop、Jbuilder（Borland公司）、Visual J++（Microsoft公司）、Visual Age for Java（IBM公司）及eclipse和eclipse的加强版WSAD。但建议读者在熟悉JDK之后再慢慢过渡到使用IDE，毕竟JDK是基础，而使用IDE编程更方便些。

JDK是Sun公司提供的软件包，其中含有编写和运行Java程序的所有工具，常用的工具包括有：

- javac：Java语言编译器，它读入Java源程序，输出结果为Java字节码。
- java：Java字节码解释器，它执行Java编译器生成的字节码文件。这是面向Java程序的一个独立运行系统。
- javaprof：资源分析工具，用于分析Java程序在运行过程中调用了哪些资源，包括类和方法的调用次数和时间，以及各数据类型的内存使用情况等。
- javahC：代码处理工具，用于从Java类调用C++代码。
- javaAppletViewer：小应用程序浏览工具，用于测试并运行Java小应用程序。

JDK还包括Java类库（包括I/O类库、用户界面类库、网络类库等）和HotJava WWW浏览器。其中，HotJava浏览器提供了在WWW环境下运行Java代码的一个运行系统，而且还为WWW开发人员提供了一个Java开发框架。

编写Java程序的机器上一定要先安装JDK，安装过程中要正确设置PATH和CLASSPATH环境变量，这样系统才能找到javac（用于编译的命令）和java（用于执行程序的命令）所在的目录。

我们以Windows环境为例。首先，登录到下列网址：

```
http://java.sun.com/javase/downloads/index.jsp
```

这是Sun公司关于Java的产品页面，从中能够找到最新版本下载。这个页面中提供了多个版本的产品，你可以根据实际需要进行选择，目前最新的版本是JDK6。

以JDK5为例，下载的文件是：jdk-1_5_0_04-nb-4_1-win.exe，这是一个自展开文件。下载成功后，双击该文件即开始安装JDK的过程，如图1-3所示，之后按照向导的提示进行安装即可。

图1-3 JDK安装开始

可以为JDK选择一个安装目录。假设JDK安装在系统的C分区上，安装JDK后产生如图1-4所示的目录结构。

- bin目录：Java开发工具，包括Java编译器、解释器等。
- demo目录：一些实例程序。
- lib目录：Java开发类库。
- jre目录：Java运行环境，包括Java虚拟机、运行类库等。

bin目录下的Java开发工具包括：

- javac：Java编译器，用来将Java程序编译成字节码。
- java：Java解释器，执行已经转换成字节码的Java应用程序。
- jdb：Java调试器，用来调试Java程序。
- javap：反编译，将类文件还原为方法和变量。
- javadoc：文档生成器，用于创建HTML文件。
- appletviwer：Applet解释器，用来解释已经转换成字节码的Java小应用程序。

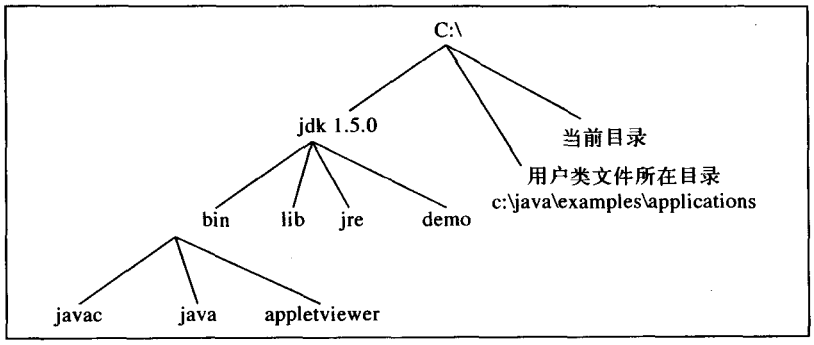

图1-4 JDK目录

接下来，需要设置环境变量，以便系统能自动找到命令所在的目录。

可以先设定Java的安装目录，假定你将JDK安装在C:\Program Files\Java\jdk1.5.0_02下，可用环境变量JAVA_HOME来指定：

```
JAVA_HOME= C:\Program Files\Java\jdk1.5.0_02
```

然后再修改PATH。这个环境变量已经存在，现在只需再增加关于Java的内容，方法是保留原来的PATH的内容，并在其后加上；%JAVA_HOME%\bin。

最后还要增加CLASSPATH环境变量，一般地，这个变量不存在，所以需要新建。设置的值为：

```
CLASSPATH=.;%JAVA_HOME%\lib\tools.jar
```

如果读者使用的是Windows环境，例如Windows NT/2000/XP，则打开控制面板，选择“系统→高级→环境变量”，然后分别添加或是修改上面两项设置即可。如果读者选择自己独特的安装目录，则JAVA_HOME中的设置要做相应的修改。

1.2.2 Java应用程序示例

从程序的运行环境来看，Java程序分为三种。第一种是Java应用程序（Java Application），它通过Java解释器控制，在命令行下使用javac/java命令来运行。第二种是Java小应用程序（Java Applet），或叫Java小程序。Java小程序不能独立运行，它要被嵌入到Web页中，由Java兼容浏览器控制执行。第三种是Servlet，这是Java技术对CGI编程的解决方案，它运行于Web服务器上，作为来自于Web浏览器或其他HTTP客户端的请求和在HTTP服务器上的数据库及其他应用程序之间的中间层程序。从技术分类来看，Java程序分为独立的应用程序和小应用程序两种。本书的前几章先介绍Java应用程序，在第8章介绍Java小程序。除非特别指明，“程序”一词是指应用程序。

和其他任何程序设计语言一样，Java语言也可以创建通常意义下的应用程序。我们先从

编制一个最小的应用程序入手，它的作用是在屏幕上显示字符串“Hello World!”，仅含有最基本的语句。我们以这个应用程序为例来说明Java程序的基本结构及生成过程。图1-5所示的是Java程序的生成步骤。

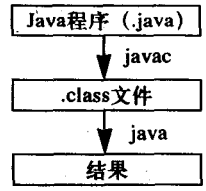

图1-5 程序的生成过程

1. 程序的创建

读者可以使用安装在系统中的任何一个文本编辑器，如Windows系统下的记事本，键入程序1-1，并将文件名取为HelloWorldApp.java。键入程序及程序名时都请注意字符的大小写，因为Java语言对大小写敏感。程序中的类名要与文件名完全一致，本例中是HelloWorldApp，文件的扩展名部分为java，不是txt。如果读者使用的是字处理程序，例如Word，则应注意不要在文件中加入任何排版信息，保证得到的源文件一定是纯文本形式的。

程序1-1 一个基本的Java应用程序

```
1 //
2 // 简单的应用程序HelloWorld
3 //
4 public class HelloWorldApp{
5    public static void main (String args[])
6    {   System.out.println ("Hello World!");
7    }
8 }
```

这些程序行包含了在屏幕上打印“Hello World!”所需的最基本的内容。

2. 程序的编译

源文件是文本形式的文件，这种形式的字符不能被计算机识别。程序要想运行，必须经过由文本文件到机器码文件的转换，而这个工作由编译器来完成。当然编译器不只完成字符编码格式的转换，它还负责检查程序的正确性，这一过程称为编译，编译是程序运行前必须经过的一步。

编译器生成的文件是字节码文件，即类文件，它的格式能被系统识别和运行。所谓的类文件是二进制格式的，它有统一的格式，JVM可以识别类文件并执行它。创建HelloWorldApp.java源文件后，就可以进行编译了。Java的编译程序是javac.exe，编译命令的一般格式是：

```
$[path]javac[-g][-O][-debug][-depend][-nowarn][-verbose][-classpathpath]
[-nowrite][-ddir] filename.java
```

该行中的第一个字符$为系统命令行提示符，这个提示符依系统不同会有所差异。提示符后面的内容才是用户键入的真正命令，其中[path]是可选项。javac命令一般放在系统的$JAVA_HOME\bin目录中，系统配置文件的PATH变量中应包含该目录（安装JDK时已经设置了）。在用户工作目录下使用javac命令时，系统自动到PATH所含的目录中查找这些命令。如果在PATH中没有包含命令所在的路径，则在键入命令时，前面要明确指明路径，也就是要增加[path]选项。

这个命令中常用的选项如下：

- -classpath path：定义javac搜索类的路径。它将覆盖默认的CLASSPATH环境变量的设置。路径是由一个用分号隔开的路径名组成，例如：

```
.;C:\java\doc\classes;C:\tools\java\classes
```

表示编译器遇到一个新类时，它先在当前文件夹中查找它的定义，如果没有，则在当前文件所处目录下其他文件夹中查找它的定义，如果还没有，则继续搜索C:\java\doc\classes目录中的