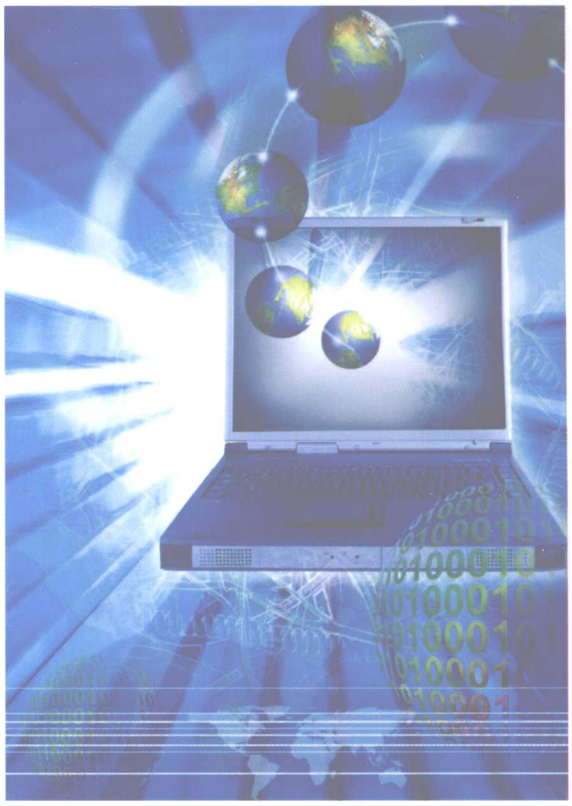
高等学校计算机应用规划教材

Java基础教程

- Java语言概述
- Java语言基础知识
- 类与对象
- 继承与接口
- 数组与字符串
- 异常处理机制
- 输入输出及数据库操作
- 多线程
- 图形用户界面



吴仁群　编著

清华大学出版社

高等学校计算机应用规划教材

Java 基础教程

吴仁群　编著

清华大学出版社
北　京

内 容 简 介

本书是针对 Java 语言初学者编写的基础教程，书中不仅讲解了 Java 程序设计的基础知识，而且提供了大量实用性很强的编程实例。全书共分 9 章：Java 语言概述，Java 语言基础，类和对象，继承和接口，数组和字符串，异常，输入输出和数据库，多线程，图形用户界面设计等。

本书内容实用，结构清晰，实例丰富，可操作性强，可作为高等学校 Java 程序设计课程的教材，也可作为计算机相关专业的培训和自学教材。

图书在版编目(CIP)数据

Java 基础教程/吴仁群　编著. —北京：清华大学出版社，2009.4

(高等学校计算机应用规划教材)

ISBN 978-7-302-19835-2

Ⅰ.J…　Ⅱ.吴…　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(2009)第 045697 号

责任编辑：刘金喜
装帧设计：孔祥丰
责任校对：胡雁翎
责任印制：李红英
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京密云胶印厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：16.75　**字　数**：387 千字
版　　次：2009 年 4 月第 1 版　　**印　　次**：2009 年 4 月第1 次印刷
印　　数：1～5000
定　　价：28.00 元

前　言

Java 语言是目前使用最为广泛的网络编程语言之一，它具有面向对象、与平台无关、安全和多线程等特点。使用 Java 语言不仅可以实现大型企业级的分布式应用系统，还能够为小型的嵌入式设备进行应用程序的开发。Java 平台标准版本(J2SE)是所有 Java 技术的基础，只有掌握 J2SE，才能进一步深入学习 Java 技术。本书内容以 JDK 1.5 为蓝本进行讲解。

作为一本基于 J2SE 的 Java 语言基础教材，本书具有以下特点：

(1) 本书内容的讲述由浅入深，符合初学者的计算机语言学习习惯。

(2) 本书在讲述每个知识点时，都辅以图形或具体实例，使读者能够从具体应用中掌握知识，能够很容易地将所学的知识应用于实践。

(3) 本书每章后面附有习题，读者可通过做习题，巩固并掌握所学知识。

本书共有 9 章及两个附录。第 1 章讲述 Java 语言发展历程、Java 语言的特点以及开发平台和开发过程。第 2 章介绍 Java 语言编程的基础语法知识。第 3 章和第 4 章讲述 Java 的面向对象技术，体现了 Java 作为一种纯粹的面向对象编程语言的编程特点。第 5 章介绍了数组和字符串的特点及使用。第 6 章介绍 Java 语言的异常处理机制。第 7 章介绍 Java 语言中输入输出流和数据库操作方法。第 8 章介绍 Java 语言多线程的含义、特点及实现。第 9 章介绍 Java 语言中如何进行图形用户界面设计及处理功能的实现。附录 A 收集了 Java 语言常见命令。附录 B 提供了一个大型的实例，供读者学完本书后进行综合训练模拟。

本书由吴仁群编写，系“北京市属市管高等学校人才强教计划资助项目”。在编写过程中，编者参考了本书《参考文献》所列举的图书，得到了清华大学出版社的大力支持，在此对《参考文献》中图书的作者及清华大学出版社表示深深的感谢。

由于时间仓促，书中难免存在一些不足之处，敬请读者批评指正。

编　者

2009 年 1 月

目　录

第1章　Java语言概述

Java 语言是目前使用最为广泛的编程语言之一，是一种简单、面向对象、分布式、解释、健壮、安全、与平台无关的性能优异的多线程动态语言。

本章的学习目标：

- 了解 Java 语言的发展里程
- 理解 Java 语言的特点
- 理解平台无关性
- 理解 Java 虚拟机 JVM
- 了解 Java 与 C/C++的关系
- 掌握 Java 运行平台
- 掌握 Java 程序开发
- 学会使用 Java 开发工具箱
- 了解 JDK 1.5 编译器的新规定

1.1　Java 语言的发展里程

Java 语言的前身是 Oak 语言，于 1991 年推出，但仅限于 Sun MicroSystems 公司内部使用。1995 年 Oak 语言改名为“Java”，并正式向公众推出，主要贡献者是 James Gosling。

Java 1.2 版本是 Java 语言发展过程中的一个关键阶段，从此，Sun 公司将 Java 更名为 Java 2。经过十年的发展，Java 语言已经发展到 1.6 版本。

Java 的发展得益于 Internet 和 Web 的出现，Internet 上有各种不同的计算机，它们可能使用完全不同的操作系统和 CPU 芯片，但希望运行相同的程序。Java 的出现标志着真正的分布式系统的到来。

当今，Java 技术已经渗透到了世界的每个角落——小到生活中的电话、烤面包机，大到汽车，都有它的身影。权威调查显示，目前全球已有超过 60%的软件开发人员使用 Java 语言。当今全球已经拥有超过 300 万使用它作为开发语言的程序员，超过 2.67 亿部支持 Java 的电话，以及超过 3 亿的 Java 卡在世界各地被配置。

1.2　Java 语言的特点

作为一种面向对象且与平台无关的多线程动态语言，Java 具有以下特点。

1. 语法简单

Java语言的简单性主要体现在以下三个方面：

(1) Java的风格类似于C++，C++程序员可以很快掌握Java编程技术；

(2) Java摒弃了C++中容易引发程序错误的地方，如指针和内存管理；

(3) Java提供了丰富的类库。

2. 面向对象

面向对象编程是一种先进的编程思想，更加容易解决复杂的问题。面向对象可以说是Java最重要的特性。Java语言的设计完全是面向对象的，它不支持类似C语言那样的面向过程的程序设计技术。Java支持静态和动态风格的代码继承及重用。单从面向对象的特性来看，Java类似于SmallTalk，但其他特性，尤其是适用于分布式计算环境的特性远远超越了SmallTalk。

3. 分布式

Java从诞生起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，便于开发网上应用系统。因此，Java应用程序可凭借URL打开并访问网络上的对象，其访问方式与访问本地文件系统几乎完全相同。

4. 安全性

Java的安全性可从两个方面得到保证。一方面，在Java语言里，像指针和释放内存等C++中的功能被删除，避免了非法内存操作。另一方面，当Java用来创建浏览器时，语言功能和一些浏览器本身提供的功能结合起来，使它更安全。Java语言在机器上执行前，要经过很多次的测试。其三级安全检验机制可以有效防止非法代码入侵，阻止对内存的越权访问。

5. 健壮性

Java致力于检查程序在编译和运行时的错误。除了运行时异常检查外，Java提供了广泛的编译时异常检查，以便尽早地发现可能存在的错误。类型检查帮助用户检查出许多开发早期出现的错误。Java自己操纵内存减少了内存出错的可能性。Java还实现了真数组，避免了覆盖数据的可能。这项功能大大缩短了开发Java应用程序的周期。Java提供Null指针检测数组边界及检测异常出口字节代码校验。同时，在Java中对象的创建机制(只能用new操作符)和自动垃圾收集机制大大减少了因内存管理不当引发的错误。

6. 解释运行，高效率

Java解释器(运行系统)能直接运行目标代码指令。Java程序经编译器编译，生成的字节码经过精心设计，并进行了优化，因此运行速度较快，克服了以往解释性语言运行效率低的缺点。Sun用直接解释器一秒钟内可调用300 000个过程。翻译目标代码的速度与C/C++没什么区别。

7. 与平台无关

Java 编译器将 Java 程序编译成二进制代码，即字节码。字节码有统一的格式，不依赖于具体的硬件环境。

平台无关类型包括源代码级和目标代码级两种类型。C 和 C++属于源代码级平台无关，意味着用它编写的应用程序不用修改只需重新编译就可以在不同平台上运行。Java 属于目标代码级平台无关，主要靠 Java 虚拟机 JVM 来实现(Java Virtual Machine)。

8. 多线程

Java 提供的多线程功能使得在一个程序里可同时执行多个小任务。线程，有时也称小进程，是一个大进程里分出来的小的独立的进程。由于 Java 实现了多线程技术，所以比 C 和 C++更健壮。多线程带来的更大的好处是更好的交互性能和实时控制性能。当然实时控制性能还取决于系统本身(UNIX、Windows、Macintosh 等)，在开发难易程度和性能上都比单线程要好。任何用过当前浏览器的人，都感觉为调一幅图片而等待是一件很烦恼的事情。在 Java 里，可用一个单线程来调一幅图片，同时可以访问 HTML 里的其他信息而不必等待它。

9. 动态性

Java 的动态特性是其面向对象设计方法的发展。它允许程序动态地装入运行过程中所需要的类，这是 C++语言进行面向对象程序设计所无法实现的。在 C++程序设计过程中，每当在类中增加一个实例变量或一种成员函数后，引用该类的所有子类都必须重新编译，否则将导致程序崩溃。Java 从如下几方面采取措施来解决这个问题。Java 编译器不是将对实例变量和成员函数的引用编译为数值引用，而是将符号引用信息在字节码中保存下来传递给解释器，再由解释器在完成动态连接类后，将符号引用信息转换为数值偏移量。这样，一个在存储器中生成的对象不是在编译过程中决定，而是延迟到运行时由解释器确定的。这样，对类中的变量和方法进行更新时就不至于影响现存的代码。解释执行字节码时，这种符号信息的查找和转换过程仅在一个新的名字出现时才进行一次，随后代码便可以全速执行。在运行时确定引用的好处是可以使用已被更新的类，而不必担心会影响原有的代码。如果程序连接了网络中另一系统中的某一类，该类的所有者也可以自由地对该类进行更新，而不会使任何引用该类的程序崩溃。Java 还简化了使用一个升级的或全新的协议的方法。如果系统运行 Java 程序时遇到了不知怎样处理的程序，没关系，Java 能自动下载所需要的功能程序。

1.3　平台无关性

如前所述，Java 属于目标代码级平台无关语言类型，主要靠 Java 虚拟机 JVM 来实现。

对高级语言的翻译方式有解释和编译两种，解释方式就是一边翻译一边运行，而编译方式则是一次性翻译好，生成目标程序。移植性涉及目标程序在不同平台上运行。解决移植性的方法有以下两种方式。

方式 1　到一台机器上将源程序重新编译成适合该台机器的机器代码，此时的高级语言源程序相当于逻辑程序模型，而编译出来的目标程序相当于物理模型，逻辑模型可以适合于任何机器，即与机器无关。

方式 2　将高级语言源程序编译成一种与机器无关的中间代码(如 Java 语言的字节码)，该中间代码程序不能被操作系统直接执行，需要由解释器来解释和执行，这种方法实际上是编译和解释的结合，称为伪编译，在每台机器上安装解释程序扩展了这种机器的执行系统，被扩展了指令的机器就可以直接执行以中间代码形式存在的程序。

Java 语言采用方式 2，将由解释程序扩展了的指令系统的机器称为 Java 虚拟机，简称 JVM。

1.4　Java 虚拟机 JVM

虚拟机是一种对计算机物理硬件计算环境的软件实现。虚拟机是一种抽象机器，内部包含一个解释器(Interpreter)，可以将其他高级语言编译为虚拟机的解释器可以执行的代码(称这种代码为中间语言 Intermediate Language)，实现高级语言程序的可移植性平台无关性(System Independence)，无论是运行在嵌入式设备还是多个处理器的服务器上，虚拟机都执行相同的指令，所使用的支持库也具有标准的 API 和完全相同或相似的行为。

Java 虚拟机是一种抽象机器，它附着在具体操作系统上，本身具有一套虚拟机器指令，并有自己的栈、寄存器等运行 Java 程序不可少的机制。编译后的 Java 程序指令并不直接在硬件系统 CPU 上执行，而是在 JVM 上执行。在 JVM 上有一个 Java 解释器用来解释 Java 编译器编译后的程序。任何一台机器只要配备了解释器，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。

JVM 是编译后的 Java 程序和硬件系统之间的接口，程序员可以把 JVM 看作一个虚拟处理器。它不仅解释执行编译后的 Java 指令，而且还进行安全检查，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 程序的安全检查引擎，如图 1-1 所示。

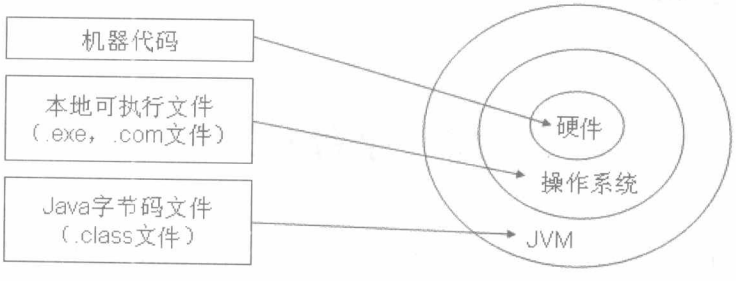


图 1-1　计算机硬件、操作系统、JVM 与各种可执行程序之间的关系

JVM 由多个组件构成，包括类装载器(Class Loader)、字节码解释器(Bytecode Interpreter)、安全管理器(Security Manager)、垃圾收集器(Garbage Collector)、线程管理(Thread Management)及图形(Graphics)，如图 1-2 所示。

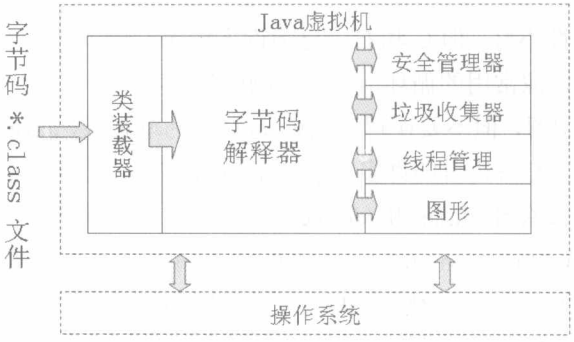


图 1-2　Java 虚拟机体系结构示意图

- 类装载器：负责加载(load)类的字节码文件，并完成类的链接和初始化工作。类加载器首先将要加载的类名转换为类的字节码文件名，并在环境变量 CLASSPATH 指定的每个目录中搜索该文件，把字节码文件读入缓冲区。其次将类转换为 JVM 内部的数据结构，并使用校验器检查类的合法性。如果类是第一次被加载，则对类中的静态数据进行初始化。加载类中所引用的其他类，把类中的某些方法编译为本地代码。
- 字节码解释器：是整个 JVM 的核心组件，负责解释执行由类装载器加载的字节码文件中的字节码指令集合，并通过 Java 运行时环境(JRE)由底层的操作系统实现操作。通过使用汇编语言编写解释器、重组指令流提高处理器的吞吐量，最大程度地使用高速缓存以及最大程度地使用寄存器等措施来优化字节码解释器。
- 安全管理器：根据一定的安全策略对 JVM 中指令的执行进行控制，主要包括那些可能影响下层操作系统的安全性或者完整性的 Java 服务调用，每个类装载器都与某个安全管理器相关，安全管理器负责保护系统不受由加载器载入系统的类企图执行的违法操作所侵害。默认的类转载器使用信任型安全管理器。
- 垃圾收集器：垃圾收集器用于检测不再使用的对象，并将它们所占用的内存回收。Java 语言并不是第一个使用垃圾收集技术的语言。垃圾收集是一种成熟的技术，早期的面向对象语言 LISP、SmallTalk 等已经提供了垃圾收集机制。理想的垃圾收集应该回收所有形式的垃圾，如网络连接、I/O 路径等。JVM 中垃圾收集的启动方式可分为请求式、要求式和后台。请求式是调用 System.gc()方法请求 JVM 进行垃圾收集的。要求式使用 new 方法创建对象时，如果内存资源不足，则 JVM 进行垃圾收集。后台式通过一个独立的线程检测系统的空闲状态，如果发现系统空闲了多个指令周期，则进行垃圾收集。

1.5　Java 与 C/C++之关系

尽管 C++安全性不好，但 C 和 C++已为许多程序设计者所接受，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。Java 设计成 C++形式，让大家很容易学习，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。Java 去掉了 C++语言的许多功能，让 Java 的语言功能很精炼，并增加了一些很有用的功能，Java 中没有#include 和#define 等预处理功能，用 import 语句来包含其他类和包；Java 中没有 structure，union 及 typedef；Java 中没有不属于类成员的函数，没有指针和多重继承，Java 只支持单重继承；Java 中禁用 goto，但 goto 还是保留的关键字；Java 中没有操作符重载；Java 中没有全局变量，可以在类中定义公用、静态的数据成员实现相同功能。

总之，Java 语言和 C++语言各有各的优势，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。

1.6　Java 运行平台

1. 三种平台简介

Java(JDK 1.5)运行平台主要分为下列 3 个版本。

- J2SE：Java 标准版或 Java 标准平台。J2SE 提供了标准的 JDK 开发平台。
- J2EE：Java 企业版或 Java 企业平台。
- J2ME：Java 微型版或 Java 小型平台。

提示：

JDK(Java Development Kit)即 Java 开发工具箱。

2. Java JDK 简介

表 1-1 给出了 JDK 版本、发布日期、版本说明及主要内容。

表 1-1　JDK 版本简介

JDK 版本	发布日期	版本说明及主要内容
JDK 1.0a2	1995.05	正式对外发布
JDK 1.0.2	1996.01	标准的 I/O 库、网络库、applet、文件 I/O 以及基本的窗口库等
JDK 1.1	1997.02	内部类、新的事件处理模型、RMI、JavaBean，JDBC、串行化、国际化、日历类以及性能改进等

(续表)

JDK 版本	发布日期	版本说明及主要内容
JDK 1.2	1998.12	浮点运算改进、String GUI 库、集合、Java 2D 图形、可访问性 (可视化 GUI 支持)、引用对象以及性能改进等
JDK 1.3	2000.05	性能改进、CORBA 兼容性、Java 音频支持以及 JNDI 等
JDK 1.4	2001.12	断言语句、64 位地址空间(Solaris)、新的 I/O 库、模式匹配、鼠标滑轮支持、IPv6、XML、WebStart 支持、性能改进及命名为 Merlin 的代码等
JDK 1.5	2003.07	通用性(代码模块)、允许一些运算符重载以及命名为 Tiger 的代码等

学习 Java 必须从 J2SE 开始，因此，本书基于 J2SE 来学习 Java，所有程序均在 JDK 1.5 版本下调试通过。

3. 环境变量

环境变量也称为系统变量，是由操作系统提供的一种与操作系统中运行的程序进行通信的机制，一般可为运行的程序提供配置信息。

常用的 Java 环境变量包括 JAVA_HOME、CLASSPATH 和 PATH。

JAVA_HOME 为那些需要使用 Java 命令和 JVM 的程序提供了通用的路径信息，其值应设置为 JDK 的安装目录的路径，如在 Windows 平台上 JDK 的安装目录为“C:\jdk1.5”，则

```
JAVA_HOME= C:\jdk1.5
```

CLASSPATH 用于指明字节码文件的位置。当执行 Java 程序时，执行命令首先把类名转换为字节码文件的路径信息，再在环境变量 CLASSPATH 值的路径列表的每个路径及其子路径中搜索指定的字节码文件，如果在所有路径都找不到该文件，就报告错误。环境变量 CLASSPATH 的值一般为一个以分号“;”作为分隔符的路径列表，如

```
CLASSPATH=c:\jdk1.5\jre\lib\rt.jar;.;
```

环境变量 PATH 是操作系统使用的变量，用于搜索在 Shell 中输入的希望执行的命令。为了便于使用，一般可把 JDK 中 Java 命令程序所在目录的路径加入 PATH 变量的值中，如

```
PATH=…; c:\jdk1.5\bin
```

4. JDK 1.5 版本安装

从 http://java.sun.com 网站下载 jdk-1_5_0_01-windows-i586-p.exe，然后安装该程序。

1) 置环境变量 JAVA_HOME

在 Windows 2000、Windows XP 中设置 JAVA_HOME 的步骤如下：

(1) 鼠标右键单击“我的电脑”。

(2) 选择“属性”菜单子项。

(3) 在出现的窗口中，选择“高级”选项。

(4) 在出现的窗口中，选择“环境变量”选项。

此时可以设置 JAVA_HOME 变量，如图 1-3 所示。

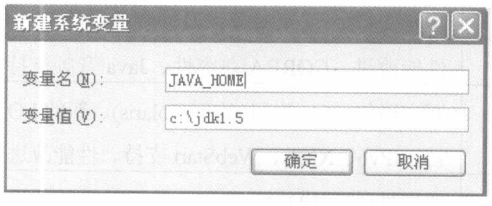


图 1-3　设置 JAVA_HOME 变量

2) 置环境变量 PATH

为了能在任何目录中使用编译器和解释器，应在系统特性中设置 PATH。

在 Windows 2000、Windows XP 中设置 PATH 的步骤同前，结果如图 1-4 所示。

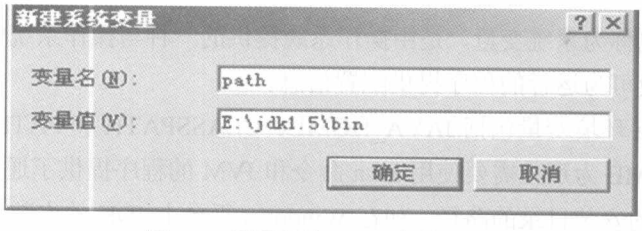


图 1-4　设置环境 PATH 变量

3) 设置变量 CLASSPATH

在 Windows 2000、Windows XP 中设置 CLASSPATH 的方法同前，结果如图 1-5 所示。

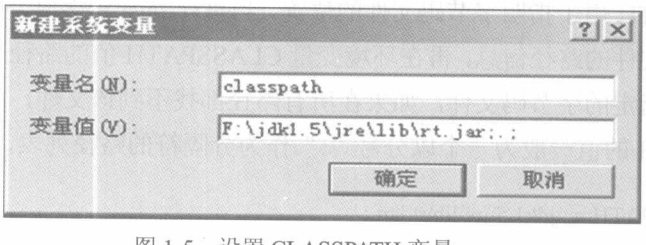


图 1-5　设置 CLASSPATH 变量

4) 在 Windows 9X 系列设置环境变量

编辑 Autoexec.bat 文件，内容如下：

```
SET path="%PATH%";c:\jdk1.5\bin;
SET JAVA_HOME=c:\jdk1.5
set classpath=E:\jdk1.5\jre\lib\rt.jar;.;
```

5) 命令行键入命令

若只是临时使用环境变量，可在 DOS 窗口的命令行输入设置环境变量的命令。例如：

```
set classpath=E:\jdk1.5\jre\lib\rt.jar;.;
```

6) 仅装 JRE

如果只想运行别人的 Java 程序可以只安装 Java 运行环境 JRE，JRE 由 Java 虚拟机、Java 的核心类以及一些支持文件组成。可以登录 Sun 的网站免费下载 Java 的 JRE。

7) 安装完毕后主要目录

\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。

提示：

Sun 公司 Java 技术官方网站: http://java.sun.com; Eclipse 项目网站: http://www.eclipse.org，各种 Java 相关开源项目网站: http://jakarta.apache.org，http://www.sourceforge.net。

1.7　Java 程序开发

利用 Java 可以开发 application 程序和 applets 程序。

Java application 程序类似于传统的 C 和 C++ 程序，不需 WWW 浏览器支持就可以直接运行。执行过程：先由 Java compiler 对源代码进行编译，然后由 Java 解释器(interpreter)解释执行。

Java applet 程序运行在网页上并且需要一个驱动的浏览器，如 Sun 的 HotJava，Microsoft 的 Internet Explorer，网景的 Netscape Navigator。它主要应用于 WWW 的 C/S 环境。执行过程：编写好的 applet—交给 Java compiler—生成可执行的字节码 —放入 HTML Web 页中—浏览器浏览。

图 1-6 显示了 application 程序和 applets 程序的开发过程。

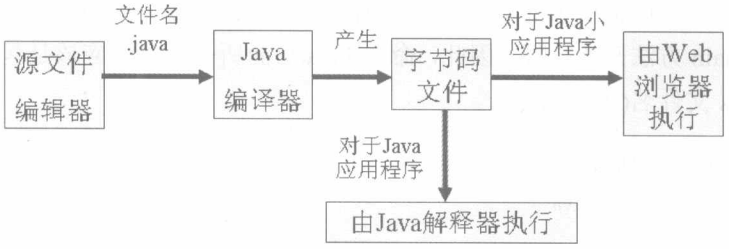


图 1-6　Java 程序开发过程示意图

1. Java 应用程序的开发

开发一个 Java 应用程序需经过三个步骤：编写源文件、编译源文件生成字节码、加载运行字节码。

第一步：编写源文件

可使用任何一个文字编辑器来编写源文件，建议使用 Editplus 或 UltraEdit。Java 是面向对象编程，Java 应用程序的源文件是由若干个书写形式互相独立的类组成，其中必须有一个主类。

一个 Java 应用程序必须有一个类含有 public static void main(String args[ ])方法，args[]是 main 方法的一个参数，是一个字符串类型的数组(注意 String 的第一个字母是大写的)。

【实例 1-1】

```
//程序 1-1
public class Hello {
    public static void main(String args[ ]){
        System.out.println("你好，很高兴学习 Java");
    }
}
```

源文件的命名规则是：Java 应用程序文件名必须和类的名称一致；如果源文件中有多个类，那么只能有一个类是 public 类；如果有一个类是 public 类，那么源文件的名字必须与这个类的名字完全相同，扩展名是 java；如果源文件没有 public 类，那么源文件的名字只要和某个类的名字相同，并且扩展名是 java 就可以了。

因此实例 1-1 的文件名为 Hello.java，以下假定该文件保存在 C:\java01 目录下。

第二步：编译 Java 源程序

当创建了 Hello.java 这个源文件后，就要使用编译器(javac.exe)对其进行编译。

```
C:> cd   java01
C:\java01>javac Hello.java
C:\java01>
```

第三步：运行 Java 源程序

必须通过 Java 虚拟机中的 Java 解释器(java.exe)来解释执行其字节码文件。Java 应用程序总是从 main 方法开始执行。因此，命令 Java 后面所带的参数应该是包含 main 方法的类对应的 class 文件名(不含后缀)。

```
C:\java01>java Hello
你好，很高兴学习 Java
```

当 Java 应用程序中有多个类时，java 命令执行的类名必须是包含 main 方法的类的名字(没有扩展名)。

【实例 1-2】

```
//程序 1-2
public class Tom{
    void show(String s){
```