普通高等教育“十二五”规划教材

Java基础与上机实验

吴仁群　编著

内容讲述由浅入深，符合初学者学习习惯
辅以图形或实例，帮助读者理解知识点

包含Java程序设计语言最基础知识

中国水利水电出版社
www.waterpub.com.cn

普通高等教育“十二

Java基础与上机实验

吴仁群　编著

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书共分 9 章，基本涵盖了 Java 程序设计语言最基础的知识，内容包括 Java 语言发展历程、Java 语言的特点、开发平台和开发过程以及如何上机调试程序；Java 语言编程的基础语法知识及上机实验实例；Java 的面向对象技术基础知识及相关上机实验实例；数组和字符串的特点、使用及上机实验实例；Java 语言的异常处理机制等。

本书内容由浅入深，注重理论联系实践，案例丰富，可操作性强，可作为高等学校的教材或教辅使用，也可供相关专业人士参考。

图书在版编目（CIP）数据

Java基础与上机实验 / 吴仁群编著. -- 北京 : 中国水利水电出版社, 2014.8
普通高等教育“十二五”规划教材
ISBN 978-7-5170-2331-9

Ⅰ. ①J… Ⅱ. ①吴… 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188548号

书　　名	普通高等教育“十二五”规划教材 Java 基础与上机实验
作　　者	吴仁群　编著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：sales@waterpub.com.cn 电话：（010）68367658（发行部）
经　　售	北京科水图书销售中心（零售） 电话：（010）88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排　　版	北京零视点图文设计有限公司
印　　刷	北京瑞斯通印务发展有限责任公司
规　　格	184mm×260mm　16 开本　17.5 印张　448 千字
版　　次	2014 年 8 月第 1 版　2014 年 8 月第 1 次印刷
印　　数	0001—1000 册
定　　价	42.00 元

凡购买我社图书，如有缺页、倒页、脱页的，本社发行部负责调换

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、多线程等特点，已被广泛应用于大型企业级分布式应用系统的开发和小型嵌入式设备系统应用程序的开发。当前，中国高等教育毛入学率接近 30%，已进入大众化教育阶段。现在在校大学生全都是 1990 年后出生。如何适应这些新情况培养应用型高级专门人才是众多应用型本科院校必须思考的问题。教材建设在人才培养过程中起着非常重要的作用。

作为一本实践性很强的 Java 语言基础教材，本书具有以下特点：

（1）包含了 Java 程序设计语言最基础的知识，知识点的讲述由浅入深，符合学生学习计算机语言的习惯。

（2）遵循理论知识和实践知识并重的原则，尽量采用图例的方式描述理论知识，并辅以大量的实例来帮助学生理解知识、巩固知识、运用知识。

（3）大部分章节都提供综合性上机实验实例，帮助学生学会综合利用各种知识来解决实际问题。

本书共有 9 章。第 1 章讲述 Java 语言发展历程、Java 语言的特点、开发平台和开发过程以及如何上机调试程序；第 2 章介绍 Java 语言编程的基础语法知识及上机实验实例；第 3 章和第 4 章讲述 Java 的面向对象技术基础知识及相关上机实验实例；第 5 章介绍数组和字符串的特点、使用及上机实验实例；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、数据库操作方法及上机实验实例；第 8 章介绍 Applet 程序的概念、应用及上机实验实例；第 9 章介绍在 Java 语言中如何进行图形用户界面设计、处理功能的实现及上机实验实例。

本书由北京印刷学院吴仁群老师编写，郭峰、刘硕、孙媛媛、蔡春霞、环海、肖志鹏、吴逸伦、包振斌等老师参与了本书部分章节的编写。在编写过程中，也得到了中国水利水电出版社的大力支持。此外，编者还参考了本书参考文献中所列举的图书，在此对参考文献中所列图书的作者及中国水利水电出版社表示深深的感谢。

本书的出版得到学校学科专项（21090113039）资助。

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

2014 年 2 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、与平台无关的并且性能优异的多线程动态语言。

- 了解Java语言的发展历程
- 理解Java语言的特点
- 理解Java虚拟机JVM
- 掌握Java运行平台的安装与使用
- 掌握Java程序开发的过程
- 学会调试简单的Java程序

1.1 Java语言的特点及相关概念

Java语言的前身是Oak语言。James Gosling项目组在对Oak语言进行小规模改造的基础上于1995年3月推出了Java语言，并于1996年1月发布了包含开发支持库的JDK 1.0版本。1998年12月，Sun公司发布了JDK 1.2版本。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1.7版本。现在Java已经成为一种相当成熟的语言。在这10年的发展中，Java平台吸引了数百万的开发者，在网络计算遍及全球的今天，有几十亿台设备使用了Java技术。

1.1.1 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3个方面。

- Java的风格类似于C++，C++程序员可以很快掌握Java编程技术。
- 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。
- 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和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语言中，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，以便尽早发现可能存在的错误。类型检查帮助用户检查出许多早期开发中出现的错误。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. 解释运行效率高

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。Java用直接解释器1秒钟内可调用300000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++属于源代码级与平台无关，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的平台上运行。Java属于目标代码级，与平台无关，主要靠Java虚拟机（JVM，Java Virtual Machine）来实现。

8. 多线程

Java提供的多线程功能使得在一个程序中可同时执行多个小任务。线程有时也称作小进程，是一个大进程中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。比如上网时都会感觉为调一幅图片而等待是一件很烦恼的事情。而在Java中，可用一个单线程来调一幅图片，同时可以访问HTML中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时方法的发展。它允许程序动态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时所无法实现的。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确定，而是延迟到运行时由解释器确定的，因此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，Java能自动下载所需要的功能程序。

1.1.2 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能执行的代码（我们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与平台无关性（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，它附着在具体的操作系统上，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程序不可缺少的机制。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看做一个虚拟处理器。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。

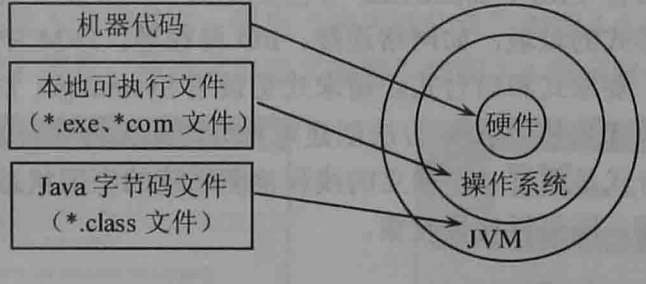

图1-1 计算机硬件、操作系统、JVM与各种可执行程序之间的关系

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。

- 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。类装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。

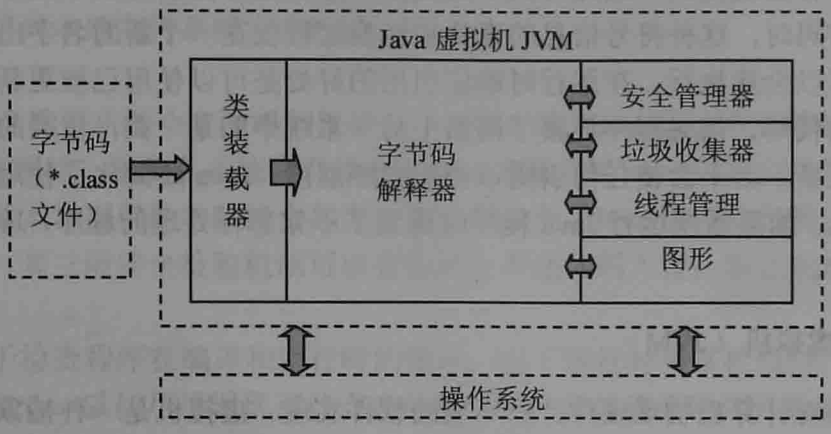

图 1-2　Java 虚拟机体系结构示意图

- 字节码解释器：它是整个 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，最大限度地使用高速缓存以及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、要求式和后台式。请求式是调用 System.gc( )方法请求 JVM 进行垃圾收集的；要求式是使用 new 方法创建对象时，如果内存资源不足，则 JVM 进行垃圾收集；后台式是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。

1.2　Java 程 序 开 发

1.2.1　运行平台

1. 平台简介

Java 运行平台主要分为以下 3 个版本。

- Java SE：Java 标准版或 Java 标准平台。Java SE 提供了标准的 JDK 开发平台。
- Java EE：Java 企业版或 Java 企业平台。

- Java ME：Java 微型版或 Java 小型平台。

提示：自 JDK 6.0 开始，Java 的 3 个应用平台称为 Java SE、Java EE 与 Java ME（之前的旧名称是 J2SE、J2EE、J2ME）。

学习 Java 通常从 Java SE 6.0 开始，因此，本书基于 Java SE 6.0 来介绍 Java 的相关知识，所有程序均在 JDK 6.0 版本下调试通过。

2. 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 运行环境变量包括 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:\java\jdk1.6 时，设置如下所示。

```
JAVA_HOME= C: \java \jdk1.6
```

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，设置如下所示。

```
CLASSPATH=C: \java \jdk1.6\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 中输入的执行命令。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，设置如下所示。

```
PATH=…; C: \java \jdk1.6\bin
```

3. JDK1.6 版本的安装

JDK1.6 的安装步骤如下：

（1）从 http://www.oracle.com/technetwork/java/javase/downloads 网站下载 JDK 6.0（程序名如 jdk-6u2q- windows-i586.exe），然后安装该程序。

（2）双击该文件进入安装状态，弹出“许可证协议”如图 1-3 所示对话框。

（3）单击【接受】按钮，弹出“自定义安装”如图 1-4 所示对话框。

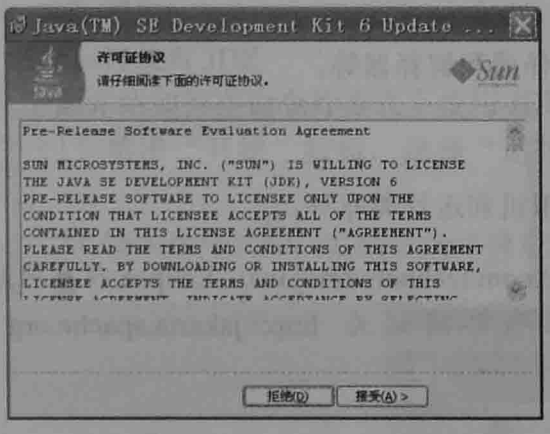

图 1-3 “许可证协议”对话框

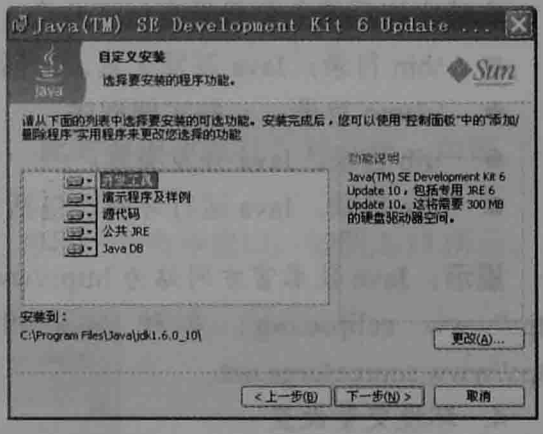

图 1-4 “自定义安装”对话框

（4）单击【更改】按钮，将安装路径修改为 C:\java\jdk1.6，弹出“修改安装路径”对

话框，如图 1-5 所示。

（5）单击【下一步】按钮，“目标文件夹”对话框，如图 1-6 所示。

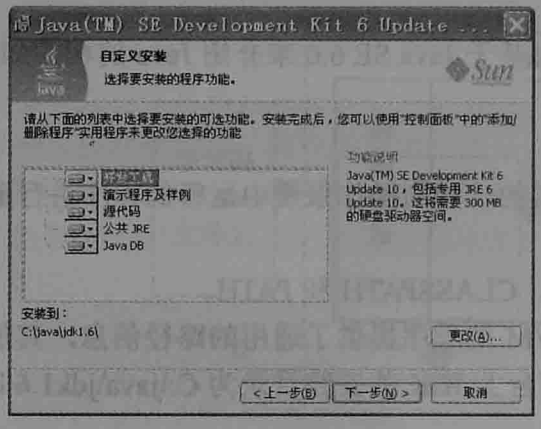

图 1-5　“修改安装路径”对话框

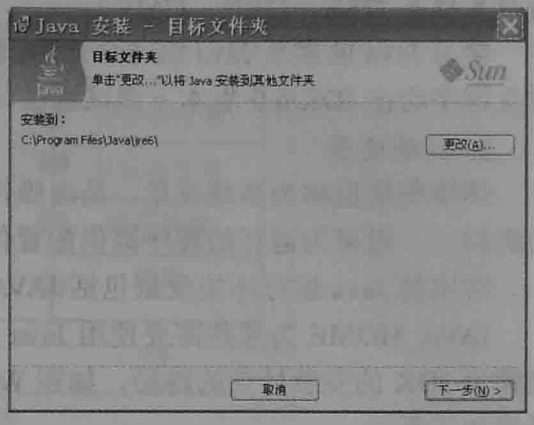

图 1-6　“目标文件夹”对话框

（6）单击【更改】按钮，将安装路径变为 C:\java\jre6，弹出如图 1-7 所示的对话框。

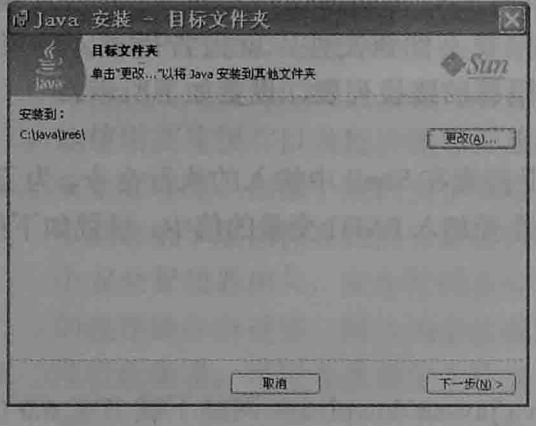

图 1-7　“JRE 路径设置”对话框

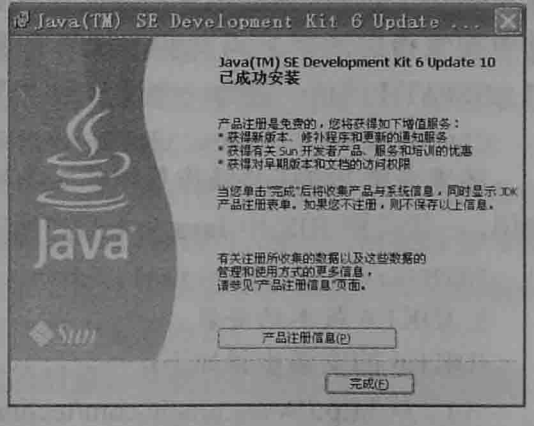

图 1-8　“安装完成”对话框

（7）单击【下一步】按钮，过一两分钟会弹出如图 1-8 所示的对话框。

（8）单击【完成】按钮，至此安装完毕。

安装完毕后的主要目录有以下几个。

- \bin 目录：Java 开发工具，包括 Java 编译器和解释器等。
- \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 虚拟机和运行类库等。

提示：Java 技术官方网站为 http://www.oracle.com/technetwork/java；Eclipse 项目网站为 http://www. eclipse.org；各种 Java 相关开源项目网站为 http://jakarta.apache.org 和 http://www.sourceforge.net。

4. 环境变量设置

（1）设置环境变量 JAVA_HOME。在 Windows 2000 和 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”菜单项。

3）在弹出的窗口中，选择“高级”选项。

4）在弹出的窗口中，选择“环境变量”选项。

此时可以设置 JAVA_HOME 变量，结果如图 1-9 所示。

（2）设置环境变量 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 和 Windows XP 中设置 PATH 的步骤同前，结果如图 1-10 所示。

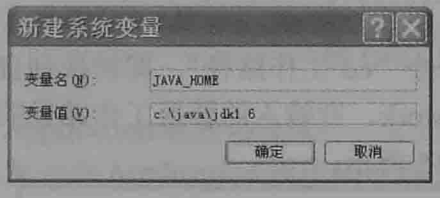

图 1-9 设置环境变量 JAVA_HOME

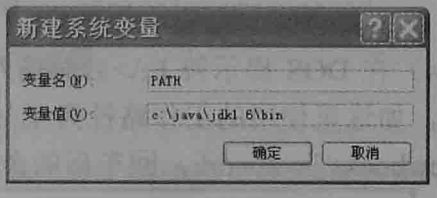

图 1-10 设置环境变量 PATH

（3）设置变量 CLASSPATH。在 Windows 2000 和 Windows XP 中设置 CLASSPATH 的方法同前，结果如图 1-11 所示。

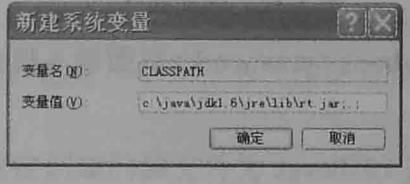

图 1-11 设置变量 CLASSPATH

（4）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，如下所示。

```
set CLASSPATH =C:\java\jdk1.6\jre\lib\rt.jar;.;
```

（5）仅安装 JRE。如果只想运行 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 http://www.oracle.com 网站免费下载 Java 的 JRE。

5. 如何使用 JDK

下面介绍如何在命令行方式下使用 JDK。

（1）单击“开始”按钮，选择“运行”菜单，此时弹出“运行”对话框，如图 1-12 所示。

（2）输入命令 cmd，之后单击“确定”按钮，弹出一个命令窗口，如图 1-13 所示。

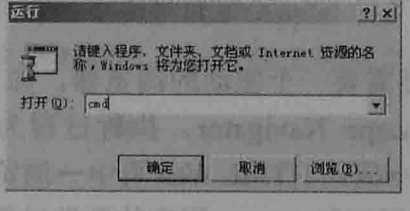

图 1-12 “运行”对话框

```
C:\Documents and Settings\wu>
```

图 1-13　命令窗口

（3）在 DOS 提示符后面（注：提示符内容视机器而定，这里为 C:\Documents and Setting \wu）输入工作路径所在硬盘的盘符（如 e:）并回车，弹出如图 1-14 所示。

```
C:\Documents and Settings\wu>e:
E:\>
```

图 1-14　命令窗口

（4）在 DOS 提示符 E:\>后面输入转换路径的命令“cd 工作路径”，即转换到自己的工作路径，如这里使用的工作路径为 E:\newbooks\java\work，则输入的转换工作路径的命令为 cd E:\newbooks\java\work，回车后的命令窗口如图 1-15 所示。

```
C:\Documents and Settings\wu>e:
E:\>cd e:\newbooks\java\work
E:\newbooks\java\work>
```

图 1-15　命令窗口

（5）在 DOS 提示符 E:\newbooks\java\work>后面输入批命令 setpath 来设置系统执行文件的位置。

setpath.bat 的内容如下：

```
set  PATH="%PATH%";c:\java\jdk1.6\bin;
set  JAVA_HOME=c:\java\jdk1.6
set  CLASSPATH=c:\java\jdk1.6\jre\lib\rt.jar;.;e:\wu\lib;;e:
\newbooks\java\work;
```

说明：

- 如果已经设置好环境变量 JAVA_HOME、PATH 和 CLASSPATH，则没有必要执行 setpath。
- 由于许多学生是在公共机房学习，一般不让修改系统信息，因此，建议执行 setpath 来临时设置这些环境变量。

1.2.2　Java 程序开发过程

利用 Java 可以开发 Application 程序和 Applet 程序。

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

Applet 程序在网页上运行还需要一个驱动的浏览器，如 Sun 的 HotJava、Microsoft 的 Internet Explorer、网景的 Netscape Navigator。执行过程为：编写好 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 1-16 显示了 Application 程序和 Applet 程序的开发过程。

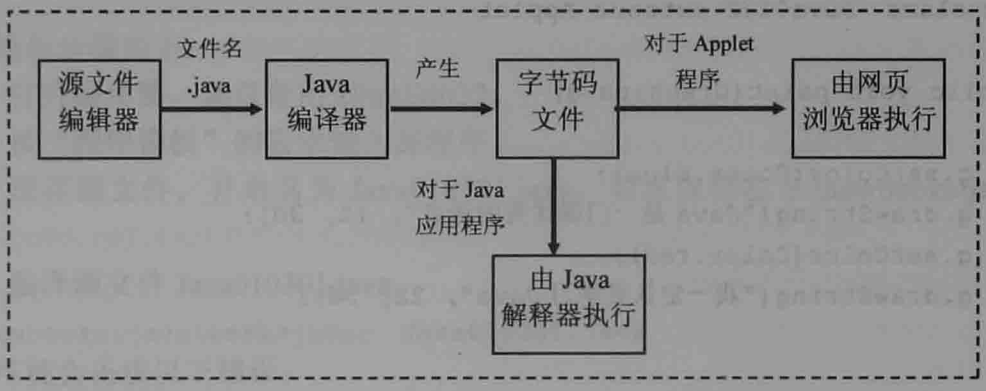

图 1-16 Java 程序开发过程示意图

1. Application 程序的开发

开发一个 Application 程序需经过 3 个步骤：编写源文件、编译源文件生成字节码和加载运行字节码。

（1）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。一个 Java 程序的源文件由一个或多个书写形式互相独立的类组成。在这些类中，最多只能有一个类是 public 类。

对 Application 程序而言，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，它是一个字符串类型的数组（注意 String 的第一个字母是大写的）。

（2）编译 Java 源程序。假定创建的源文件为 Java0101.java，此时使用编译器（javac.exe）对其进行编译。

```
e:\newbooks\java\work>javac  Java0101.java
```

编译完成后，会生成一个名为 Java0101.class 的字节文件。

提示：如果在一个源程序中有多个类定义和接口定义，则在编译时将为每个类生成一个.class 文件（每个接口编译后也生成.class 文件）。

（3）运行 Java 源程序。字节码文件必须通过 Java 虚拟机中的 Java 解释器（java.exe）来解释执行。由于 Application 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）。

```
e:\newbooks\java\work>java  Java0101
```

2. Applet 程序的开发

开发一个 Applet 程序需经过 3 个步骤：编写源文件、编译源文件生成字节码，以及通过浏览器加载运行字节码。

（1）编写源文件。一个 Applet 源文件也是由若干个类组成的，一个 Applet 源文件不再需要 main 方法，但必须有且只有一个类扩展了 Applet 类，即它是 Applet 类的子类（Applet 类是系统提供的类），我们把这个类称作 Applet 源文件的主类。

【例 1-1】

```
//Java0102.java
import java.applet.*;
import java.awt.*;
```

```
public class  Java0102 extends Applet
{
   public void paint(Graphics g)
   {
      g.setColor(Color.blue);
      g.drawString("Java 是一门很优秀的语言", 12, 30);
      g.setColor(Color.red);
      g.drawString("我一定认真学习 Java", 22, 56);
   }
}
```

（2）编译源文件。

```
e:\newbooks\java\work>java  Java0102.java
```

编译成功后，文件夹 E:\newbooks\java\work 下会生成一个 Java0102.class 文件。

（3）运行。Applet 程序由浏览器运行，因此必须编写一个超文本文件（含有 Applet 标记的 Web 页）通知浏览器来运行这个 Applet 程序。

下面是一个最简单的 html 文件（名称由使用者自己确定，这里不妨假定为 Java0102.html），该文件通知浏览器运行 Applet 程序。使用记事本编辑如下。

```
<applet code=Java0102.class  height=100  width=300>
 </applet>
```

现在可以使用浏览器打开文件 Java0102.html 来运行 Applet 程序，运行结果如图 1-17 所示。

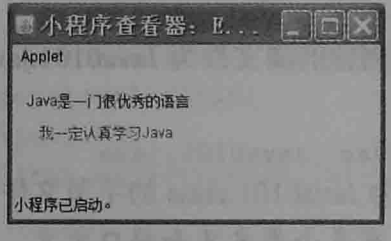

图 1-17　用 IE 打开.html 文件

另外，还可以在 DOS 命令行下使用 appletviewer 来打开网页文件以便执行 Applet 程序。

```
e:\newbooks\java\work> appletviewer  Java0102.html
```

1.3　上　机　实　验

1.3.1　一个简单的 Application 程序

【例 1-2】编写一个简单的 Java 应用程序，该程序在命令行窗口输出文字："你好，很高兴学习 Java"。

```
//Java010301.java
public class Java010301 {
    public static void main(String args[ ]){
        System.out.println("你好, 很高兴学习 Java");
    }
```