世纪英才高职高专计算机系列教材

Java 语言程序设计基础

桂　超　赵海廷　主编

张桂刚　包　琼　编著

人 民 邮 电 出 版 社

图书在版编目（CIP）数据

Java 语言程序设计基础/桂超，赵海廷主编；张桂刚，包琼编著．
—北京：人民邮电出版社，2005.5
（世纪英才高职高专计算机系列教材）

ISBN 7-115-13383-2

Ⅰ．J…　Ⅱ．①桂…②赵…③张…④包…　Ⅲ．JAVA 语言—程序设计—高等学校：技术学校—教材　Ⅳ．TP312

中国版本图书馆 CIP 数据核字（2005）第 030232 号

内 容 提 要

本书由浅入深地介绍了 Java 语言的基础知识和编程的特点，旨在培养读者运用面向对象程序设计方法去解决实际问题的能力，是学习面向对象程序设计知识的基础教材。本书实例丰富，能够增强读者对相关内容的理解。同时，本书是对作者多年教学经验的总结，并融入了一定的实际编程技术。

本书内容丰富，系统性强，可作为高等院校 Java 语言程序设计课程的教材，也可供从事软件开发和应用的人员参考。

世纪英才高职高专计算机系列教材

Java 语言程序设计基础

◆ 主　　编　桂　超　赵海廷
编　　著　张桂刚　包　琼
责任编辑　付方明
执行编辑　张　海

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-67129264
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：17.75
字数：440 千字　　2005 年 5 月第 1 版
印数：1 – 5 000 册　　2005 年 5 月北京第 1 次印刷

ISBN 7-115-13383-2/TP・4652

定价：26.00 元

前　言

随着互联网技术的飞速发展，网络已将世界各地的计算机连成了一个可相互传输信息的整体，改变着人类的思维方式和生活方式。Java 语言的产生，对互联网技术的发展又起到了强大的推动作用。Java 语言从 1995 年诞生至今，已逐渐扩展到网络应用的各个领域，其面向对象、平台独立、安全可靠和多线程技术等特点使它深受广大用户的喜爱。目前，基于网络的程序代码绝大部分都采用 Java 语言编写而成。

Java 语言是一种面向对象的程序设计语言，它起源于 C++语言，但比 C++语言简明，且更有针对性。它具有较强的容错能力，使用安全可靠，同时还提供了开发的机制，可最大限度地利用网络，特别适合开发网络应用程序。它的应用程序（Applet）与 HTML 有机地结合，可以借助网络系统进行动态传输各种信息，且不受计算机系统以及运行环境的限制。

作者根据多年的教学积累和软件开发的经验，按照循序渐进、由浅入深的原则，精心编写了本书。在编写过程中，作者充分考虑到了计算机的发展趋势，选取的内容综合性较强、基础性较强，可为读者在 Java 语言程序设计方面进行更深入的学习打下坚实的基础。书中列举的实例都已经通过了测试，均可直接上机运行。

全书共分 14 章。第 1 章介绍了 Java 语言的发展过程、特点、运行环境及开发步骤；第 2 章介绍了 Java 语言的源程序编写与执行方法、HTML 的简要特征、Java 的 Applet 程序及图形用户界面输入/输出基础知识；第 3 章介绍了 Java 语言的数据类型；第 4 章介绍了 Java 语言的运算符和表达式、位运算、强制数据类型转换；第 5 章介绍了 Java 语言的基本语句；第 6 章介绍了 Java 语言的数组、字符串和向量；第 7 章介绍了 Java 语言中方法的建立和使用；第 8 章介绍了类和对象的定义及使用特征；第 9 章介绍了接口和包的概念和使用；第 10 章介绍了异常处理的方法；第 11 章介绍了流的概念、输入/输出及文件的操作等；第 12 章介绍了 Java 语言的图形用户界面设计、AWT 包、swing 包的使用；第 13 章介绍了各种布局管理器的功能和使用；第 14 章介绍了 Java 语言多线程的基本机制、线程的实现及调度技术。

对于 Java 语言的学习，我们建议在实践中分为三个阶段。第一阶段是初步了解和掌握 Java 语言的基本数据类型、语句及系统提供的资源，资源是指 Java 的各种包及 Windows 常用的 API 等；第二阶段是编写较基本的程序，融会贯通；第三阶段则是将各个部分的知识点综合汇集后，针对某一个实际问题，编写规模较大的、综合性较强的、具有一定实用价值的程序。按照这三个步骤进行实践学习，将有助于掌握 Java 语言程序设计的精髓，极大地提高编程能力。

本书内容丰富，系统性强，可作为高等院校 Java 语言程序设计课程的教材，也可供从事软件开发和应用的人员参考。

由于作者水平有限，书中难免存在错误之处，恳请广大专家、读者批评指正。

编　者

目　　录

第 1 章　Java 发展概述 …… 1
1.1　程序设计语言的发展 …… 1
1.2　Java 语言 …… 1
1.2.1　Java 语言的发展历史 …… 1
1.2.2　Java 语言的特点 …… 2
1.3　Java 的开发与运行环境 …… 4
1.3.1　J2SDK 的安装 …… 4
1.3.2　J2SDK 的设置 …… 5
1.4　开发与运行 Java 程序的步骤 …… 5
1.4.1　选择编辑工具 …… 5
1.4.2　编译与运行 Java 程序 …… 5
第 2 章　Java 语言程序和 HTML 简介 …… 7
2.1　Java 语言的 Application 程序 …… 7
2.1.1　源代码的编辑 …… 7
2.1.2　字节码文件的生成 …… 8
2.1.3　字节码文件的解释执行 …… 9
2.2　HTML 简介 …… 10
2.2.1　HTML 的基本要素 …… 12
2.2.2　HTML 文件的结构 …… 13
2.2.3　在 HTML 文件中加入图像、链接和声音 …… 13
2.2.4　Applet 标记 …… 15
2.2.5　常用的 HTML 标记 …… 15
2.2.6　HTML 文件的编辑 …… 16
2.3　Java 语言的 Applet 程序 …… 17
2.3.1　源代码的编辑 …… 17
2.3.2　代码的嵌入 …… 18
2.3.3　Applet 程序的运行 …… 18
2.4　Java 语言字符界面的输入输出 …… 20
2.5　Java 语言图形界面的输入输出 …… 21
2.5.1　Java Applet 图形界面的输入输出 …… 22
2.5.2　Java Application 图形界面的输入输出 …… 23
第 3 章　Java 语言的数据类型 …… 26
3.1　Java 的标识符 …… 26
3.2　Java 的关键字 …… 26
3.3　Java 的常量和变量 …… 28
3.3.1　常量和符号常量 …… 28

3.3.2 变量 30
3.3.3 变量作用域 30
3.4 Java 的基本数据类型 30
3.4.1 整型 30
3.4.2 实型 31
3.4.3 字符型 32
3.4.4 布尔型 32
3.5 变量的初始化 32
3.6 简单程序举例 33
第 4 章 Java 语言的运算符和表达式 37
4.1 赋值、算术运算符及其表达式 37
4.1.1 赋值运算符及其表达式 37
4.1.2 算术运算符及其表达式 38
4.2 关系、逻辑运算符及其表达式 40
4.2.1 关系运算符及其表达式 40
4.2.2 逻辑运算符及其表达式 42
4.3 位运算符和表达式 43
4.3.1 计算机内数据的表示 43
4.3.2 位运算符及其表达式 44
4.4 其他运算符 47
4.5 Java 语言运算符的优先级和结合性 48
4.6 数据类型的转换 49
4.6.1 数据类型的自动转换 49
4.6.2 数据类型的强制转换 49
第 5 章 Java 语言的基本语句 51
5.1 Java 语言的 3 种基本结构 51
5.2 顺序结构语句 51
5.2.1 变量声明语句和表达式语句 51
5.2.2 复合语句和分程序 52
5.3 选择结构语句 53
5.3.1 条件运算符 53
5.3.2 if～else 语句 54
5.3.3 switch～case 语句 60
5.4 循环结构语句 62
5.4.1 while 语句 62
5.4.2 for 语句 63
5.4.3 do～while 语句 64
5.4.4 循环嵌套 65
5.5 转移控制语句 66
5.5.1 break 语句 66

5.5.2 continue 语句 …… 68
第 6 章 数组、字符串和向量 …… 73
6.1 一维数组 …… 73
6.1.1 一维数组的声明 …… 73
6.1.2 创建一维数组 …… 73
6.1.3 一维数组的初始化 …… 74
6.1.4 一维数组应用举例 …… 74
6.2 多维数组 …… 75
6.2.1 多维数组声明 …… 75
6.2.2 创建多维数组 …… 75
6.2.3 多维数组的初始化 …… 76
6.2.4 多维数组应用举例 …… 76
6.3 字符数组 …… 77
6.3.1 字符数组声明 …… 78
6.3.2 创建字符数组 …… 78
6.3.3 字符数组初始化及其举例 …… 78
6.4 不变字符串 String 类 …… 80
6.4.1 String 类的构造方法 …… 80
6.4.2 字符串的常用方法 …… 82
6.4.3 字符串与子字符串的操作 …… 82
6.4.4 toString 方法及其他方法 …… 84
6.5 可变字符串 StringBuffer 类 …… 84
6.5.1 StringBuffer 类的构造方法 …… 84
6.5.2 StringBuffer 类的常用方法 …… 85
6.6 向量 …… 86
6.6.1 向量的声明 …… 86
6.6.2 Vector 类的常用方法 …… 87
第 7 章 Java 语言的方法 …… 89
7.1 return 语句 …… 89
7.2 Java 语言方法的定义、返值和调用 …… 90
7.2.1 Java 语言方法的修饰符 …… 90
7.2.2 Java 语言方法的定义 …… 91
7.2.3 Java 语言方法的调用 …… 93
7.2.4 Java 语言方法的返值 …… 94
7.3 方法之间的数据传递 …… 95
7.3.1 方法间的数值传递 …… 96
7.3.2 方法间的引用传递 …… 96
7.4 Java 语言方法的递归调用 …… 98
7.5 Java 语言方法的命令行参数 …… 103
第 8 章 类的声明和对象的实例化 …… 105

8.1 Java 的类 105
8.1.1 类的定义 105
8.1.2 类的修饰符 106
8.1.3 类的类体 107
8.1.4 类的构造方法 108
8.2 类的成员变量 109
8.2.1 类成员变量的声明 109
8.2.2 类成员变量的修饰 111
8.3 类的成员方法 113
8.3.1 成员方法的设计 113
8.3.2 成员方法的声明和修饰 115
8.3.3 方法体 117
8.3.4 消息的传递 118
8.4 Java 对象的实例化 119
8.4.1 创建对象 119
8.4.2 使用对象 119
8.4.3 清除对象 120
8.5 Java 类的继承 120
8.5.1 继承的概念 120
8.5.2 继承的实现 121
第 9 章 接口和包 125
9.1 抽象类和方法 125
9.1.1 定义抽象类 125
9.1.2 抽象类的实现 126
9.2 接口 128
9.2.1 接口的概念 128
9.2.2 定义接口 128
9.2.3 接口的特点和实现 129
9.3 包 134
9.3.1 包的概念 134
9.3.2 包的定义 135
9.3.3 存放的位置 135
9.3.4 包的引用 136
9.3.5 将多个独立的类放入同一个包中 136
第 10 章 异常处理 141
10.1 Java 的异常处理机制 142
10.2 异常处理方法 143
10.2.1 try…catch…finally 结构 144
10.2.2 抛出异常 149
10.2.3 自定义异常 152

第 11 章　输入和输出 …… 156
11.1　输入输出类库 …… 156
11.2　标准输入输出 …… 160
11.3　文件操作 …… 161
第 12 章　图形用户界面 GUI …… 172
12.1　AWT 包 …… 172
12.1.1　屏幕坐标体系 …… 173
12.1.2　建立窗口的 Frame 类 …… 175
12.1.3　使用基本组件 …… 177
12.1.4　AWT 图形用户界面的深入学习 …… 189
12.2　Java 语言中的事件处理 …… 194
12.2.1　事件处理机制 …… 194
12.2.2　可用的事件监听者和它们处理的事件类型 …… 195
12.2.3　事件及其响应 …… 196
12.2.4　在 Java 中的事件处理方式 …… 199
12.3　swing 包 …… 203
12.3.1　Swing 的层次结构及具体组件 …… 204
12.3.2　创建 JFrame 窗口 …… 205
12.3.3　窗口事件 …… 206
12.3.4　swing 包中常用的组件 …… 208
第 13 章　高级用户界面 GUI 设计 …… 217
13.1　布局管理器 …… 217
13.1.1　布局管理器的概念 …… 217
13.1.2　Border 布局管理器 …… 218
13.1.3　Flow 布局管理器 …… 220
13.1.4　Card 布局管理器 …… 221
13.1.5　Grid 布局管理器 …… 223
13.1.6　Box 布局管理器 …… 224
13.2　键盘和鼠标事件的处理 …… 225
13.2.1　键盘事件 …… 225
13.2.2　鼠标事件 …… 226
13.3　菜单设计 …… 228
13.4　对话框设计 …… 251
13.4.1　JoptionPane …… 251
13.4.2　Dialog …… 252
13.5　窗口 …… 254
13.5.1　JPanel 容器 …… 254
13.5.2　JscrollPane …… 255
第 14 章　多线程技术 …… 257
14.1　多线程的基本概念 …… 257

14.1.1 多线程 257
14.1.2 Windows 平台上线程的运行机制 257
14.2 线程的状态 259
14.2.1 线程的生命周期 259
14.2.2 线程类 260
14.3 线程体及其构造 261
14.3.1 线程体 261
14.3.2 采用直接继承构造线程体 261
14.3.3 采用实现 Runnable 接口构造线程体 262
14.4 线程同步控制 263
14.4.1 synchronized 关键字 263
14.4.2 wait( )、notify( )/notifyall( ) 265
14.4.3 同步控制的信号量 266
14.4.4 线程同步的示例 267
参考资料 274

第 1 章　Java 发展概述

1.1　程序设计语言的发展

计算机能够直接理解的语言是由 0 和 1 组成的机器语言。虽然机器语言执行起来比较快捷，但书写起来却十分烦琐，而且直观性差，难以阅读，也难以移植。所以，为了提高编程的工作效率，人们采用助记符的方式，先将机器指令用特定符号表示，然后进行书写，从而产生了汇编语言。

虽然汇编语言在书写和可读性方面有了很大的进步，但是因为它与具体类型的计算机 CPU 指令集密切相关，所以缺乏通用性和可移植性。

在程序设计中，与所用的计算机类型无关而与日常生活中的书写习惯非常相近的计算机语言被称为高级语言。20 世纪 50 年代初期，第一个高级语言 FORTRAN 被开发出来，并用于科学数值计算。迄今为止，计算机领域已出现了数千种高级语言，但常用的高级语言却不超过 50 种。

高级语言一般具有专长性，即某种高级语言往往在某些方面功能强大，而在其他方面的功能则相对较弱。如 FORTRAN 语言擅长数值计算；C 语言擅长系统软件和应用软件的开发；PASCAL 语言作为教学的示范语言，便于学生了解结构化编程；Java 语言则适用于网络系统的开发和应用。

按照语言的描述结构特征，可将高级语言分类如下：

- 过程语言：FORTRAN、PL/1、PASCAL、C 等。
- 函数语言：LISP、ML 等。
- 面向对象语言：Java、C++、ADA 等。

面向对象的程序设计语言符合人们对客观世界的认识与思维方式，具有比面向过程语言更多的优点，是当今软件开发领域采用的主流程序设计语言。

1.2　Java 语言

1.2.1　Java 语言的发展历史

Java 语言是面向对象的计算机程序设计语言，它既可以编写嵌入在 Web 网页中运行的程

序（Java Applet 程序），也可以编写独立运行的程序（Java Application 程序），是当前非常流行的编程语言。

Java 语言是 Sun 公司在 20 世纪 90 年代初开发的，最初只是作为小型家用电器的编程语言，用于解决诸如电烤箱和 PDA 等小型电子设备的控制和通信问题，当时命名为 Oak。由于这些智能家用电器的市场需求没有预期的高，因此 Sun 公司就放弃了该项计划。就在 Oak 即将夭折之时，Internet 发展迅猛，Oak 的研究人员发现 Oak 非常适合用于 Internet 方面的软件开发。于是，他们在 Oak 的基础上，编写了一个能与 Oak 配合使用的浏览器——Hot Java。实践表明，Oak 在 Internet 方面应用得很好，但是由于 Oak 已被其他产品先行注册，所以 Oak 开发小组就以他们经常饮用的咖啡“ Java ”命名这个软件系统，于是便产生了 Java 语言。

1995 年 5 月，Sun 公司正式发布了 Java。由于在 Internet 上存在着巨大商业利益，Java 语言的出现引起了商界的极大兴趣。作为专为商业用途设计的程序设计语言，Java 语言伴随着 Internet 的迅猛发展而发展，逐渐成为了重要的 Internet 编程语言。Java 系统提供了强大的图形、图像、动画、音频、视频和多线程功能以及网络交互能力，使得 Java 语言在设计交互式多媒体网页和网络应用程序等方面大显身手，成为当今推广最为迅速的计算机程序设计语言。

1.2.2 Java 语言的特点

Java 语言是一种体系结构中立的、多线程的动态面向对象程序设计语言，它具有简单、健壮、安全、可移植和高性能等特点。

1．简单性

Java 语言的编程风格非常接近于 C++语言，但是要比 C++语言简单得多。Java 语言抛弃了 C++语言中一些不常用、难以理解或是容易引起混淆的成分，增加了自动垃圾搜集功能，用于回收不再使用的内存区域，简化了内存的管理，避免了内存分配混乱引发的问题。同时，Java 语言的基本系统（Java 解释器和编译器）比较小，不足 250KB。

2．面向对象

面向对象技术大大提高了软件开发中代码的重用效率，增强了可靠性。Java 语言继承了 C++语言中面向对象技术的核心，更具有动态解决问题的能力。

3．分布性

Java 语言提供了大量的系统模块，支持基于 TCP/IP 协议的编程，使得用 Java 语言建立网络连接比用 C/C++语言容易得多。

4．健壮性

Java 语言在编译和运行程序时，提供了早期的编译检查和后期的动态检查，能够消除很多错误。

5．安全性

Java 语言主要使用于网络和分布环境，但是它不支持指针操作，这就迫使系统对内存的

访问都必须通过对象的实例变量进行，从而杜绝了非法存取数据和破坏关键对象属性的可能，关死了计算机病毒进入系统的大门。

6．体系结构中立

Java 语言源程序（扩展名为.java）的执行与一般的高级语言有所不同，它要先被编译，产生一种称为字节码（Byte Codes）的中间编码文件（扩展名为.class），然后再由解释器对字节码文件解释生成机器码，才可以在计算机上运行。将这个过程用图形表述如图 1-1 所示。

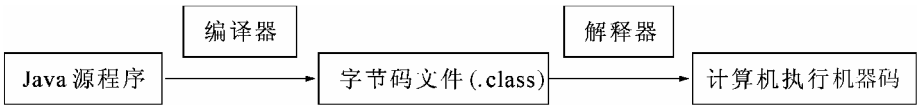

图 1-1　Java 源程序的执行过程

可以运行 Java 字节码的程序就叫作 Java 虚拟机（Java Virtual Machine），例如浏览器和 Java 的开发工具等均属于 Java 虚拟机的一部分。不管什么类型的计算机，也不管计算机上安装的是什么操作系统，只要操作平台上安装有 Java 虚拟机，就可以运行已经编写好的字节码文件。Java 的源程序在不同环境中的运行过程如图 1-2 所示。

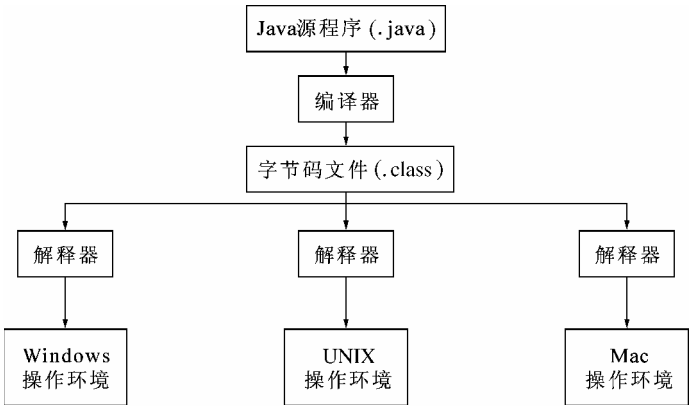

图 1-2　Java 源程序在不同操作环境中的运行过程

7．可移植性

Java 语言的结构中立性是构成程序的可移植性的基础。很多计算机语言的基本数据类型长度都有平台依赖性，而 Java 则采用固定长度。例如，整数类型（int）的长度固定为 32 位，双精度类型（double）的长度固定为 64 位。Java 语言中的类库也提供了可移植的接口，例如，类库中有一个抽象类 Window，它适合于 UNIX、WindowsNT／95 和 Macintosh。Java 系统本身也是可移植的。Java 编译器是用 Java 语言编写而成的，Java 解释器是用 ANSI C 语言编写而成的，它们都有良好的可移植性。

8．解释性

Java 语言是解释执行的。在程序运行时，字节码被直接翻译成本地机器指令，之间没有

存储。模块的连接是步进和多线程的，因此执行速度很快。

9．高性能

Java 语言在设计字节码时已经把机器码的翻译问题考虑进去了，所以，实际翻译过程非常简单。尽管字节码翻译执行的速度已经很快了，但是，在运行某些程序时，字节码仍将被快速翻译成当前 CPU 的指令，这在某种程度上相当于将最终机器指令的产生放在了动态加载器中进行，从而降低了实际翻译的速度。在 Sun Microsystems SparcStation 10 计算机上进行的一项 30 万个方法的调用试验，证明了将解释型字节码翻译成机器码的速度与 C/C++差不多。

10．多线程

在现实世界里，每时每刻都有许多事情在同时发生。多线程的概念和这种情况类似，也就是让计算机同时运行多个程序段。编写一个能同时处理多个任务的多线程程序要比编写一个单线程程序困难得多。Java 提供了一套复杂的线程同步化机制，程序员可以方便地使用这种机制设计的方法，编写出健壮的多线程程序。

11．动态性

在许多方面，Java 比 C/C++更加动态化，它被设计成能够适应环境变化的语言。C++属于编译加载，不同版本的 C++和类库的升级通常会使得软件开发商重新编译他们的程序。而 Java 属于运行加载，Java 的类库可以自由添加方法和属性且不会影响到用户程序。因此，Java 语言的动态性使它可以更好地适应不断变化的执行环境。

1.3 Java 的开发与运行环境

开发 Java 程序，一是需要书写 Java 源程序的编辑工具，二是需要将 Java 源程序编译成字节码和将字节码解释为机器码的执行工具，这二者被称为 Java 的开发与运行环境。

Sun 公司推出的 J2SDK（Java2 Software Development Kit）是一个非常广泛应用的软件包，可以从网上免费下载。其他软件如 Borland 公司的 Jbuilder、IBM 公司的 Visual Age for Java 以及 Symantec 公司的 Visual cafe 等，使用得也比较多。

1.3.1 J2SDK 的安装

J2SDK 也被简称为 JDK，是目前进行 Java 开发的重要工具，较新的版本为 JDK1.4，用户可从“http://Java.sun.com/”免费下载。可下载的 JDK 有 3 个版本供用户选择，即标准版（STANDARD EDITION）、企业版（ENTERPRISE EDITION）和小型家电版（MICRO EDITION）。

下载的 JDK 文件是自解压文件，正确下载后，可以直接执行该文件。用户只需要依据屏幕提示逐步进行，即可完成安装。

1.3.2 J2SDK 的设置

对于 UNIX（Solaris 系统）用户，下载后的 JDK 文件名为 jdk1.1.3_solaris2_sparc.bin，应当安装在/usr/local 目录下面，在 shell 窗口中执行如下命令：

```
%chmod  a+x  jdk1.1.3_solaris2_sparc.bin
%./jdk1.1.3_solaris_sparc.bin
```

执行后，生成名为 jdk1.1.3 的目录，然后修改.cshrc 文件：

```
set  Path=($ Path/usr/local/jdk1.1.3/bin)
```

对于使用 Windows 系统的用户来说，需要对 path 和 classpath 两个环境变量进行正确设置。如果 JDK 的安装目录为 C:\jdk1.4，则需要将 path 设置到 C:\jdk1.4\bin，将 classpath 设置到 C:\jdk1.4\lib\tools.jar 和“ . ”。

具体地说，在 Windows 98/me 中，只需在文件 autoexec.bat 中加入以下两行命令：

```
Set path=%path%;C:\jdk1.4\bin
Set classpath=.;C:\jdk1.4\lib\tools.jar
```

在 Windows 2000 中，设置方法是通过双击“ 控制面板 ”的“ 系统 ”图标，并选择“ 高级 ”，来添加或修改环境变量 path 和 classpath 的。

1.4 开发与运行 Java 程序的步骤

在开发 Java 应用程序时，可先用纯文本编辑器书写 Java 源程序，其文件扩展名为.java，然后再对源程序进行编译，生成扩展名为.class 的字节码文件，最后，用 Java 虚拟机来运行字节码文件。

1.4.1 选择编辑工具

Borland 公司的 Jbuilder 和 IBM 公司的 Visual Age 等开发工具提供了一个集成的开发环境，但 JDK 不含编辑功能，因此，使用 JDK 的用户必须选择一个文本编辑器来书写 Java 源程序。Windows 环境下的写字板、记事本可以用来编写 Java 源程序，另外，Office 中的 Word 软件也可以用来编写大型程序，但保存文件时应注意以纯文本方式存储。

1.4.2 编译与运行 Java 程序

在 Windows 环境下，一般在命令行状态运行 Java 程序。首先在“ 程序 ”菜单中选择“ 附件 ”，然后选择“ 附件 ”中的子菜单项“ 命令提示符 ”即可。

先用编辑器写好 Java 源程序，然后按下列步骤进行。

（1）编译 Java 源程序

用 javac.exe 文件对 Java 源程序进行编译，格式为：

```
javac  文件名.java
```

编译的目的是产生字节码文件。

例如，已经写好一个 Java 源程序 my.java，在编译时，应该使用如下命令：

javac　　my.java

需要注意的是，在编译时不能漏写 Java 源程序的扩展名。

经过上述编译后，将会产生一个字节码文件 my.class。如果在编译过程中出现错误，则应该修改错误后再进行编译，直至成功通过。

（2）对字节码文件解释执行

用 java.exe 文件对 Java 字节码文件进行解释执行，格式为：

java　字节码文件名

例如，生成 my.class 字节码文件后，用 java my 即可运行。

习　　题

1．Java 语言有何特点？

2．Java 编译器产生的文件扩展名是什么？

3．能否用 Office 中的 Word 软件书写 Java 源程序代码？

4．试简述在不同操作系统上进行 Java 开发时，对两个环境参数的设置方法。

5．什么是字节码文件？它可以直接执行吗？

6．Java 的平台是什么？为什么说它可以跨平台？

第 2 章　Java 语言程序和 HTML 简介

2.1　Java 语言的 Application 程序

根据结构和运行环境的不同，Java 语言程序可以分为两类：Java Applet 和 Java Application。Java Applet 是嵌入在由 HTML 编写的 Web 页面中的非独立程序，它需要由 Web 浏览器内部所包含的 Java 解释器来解释运行，而 Java Application 则是一个完整的程序。Java Application 和 Java Applet 适用于不同的场合。

一般而言，高级语言编程需要经过源程序编辑、目标程序编译和连接成可执行文件 3 个阶段。从第 1 章可知，Java 程序要经过源程序编辑、编译生成字节码和解释执行字节码 3 个阶段。

2.1.1　源代码的编辑

Java 源程序是以“.java”为后缀的文本文件，既可以使用各种 Java 集成开发环境中的源代码编辑器来编写，也可以使用其他文本编辑器，如 Turbo C 集成开发环境编辑器、Windows 95/98/2000 中的记事本、Word 软件或 DOS 中的 EDIT 等。下面是一个 Java　Application 程序的例子。

【例 2.1】L0201.java 程序的源代码。

```
import java.io.*;
public class L0201
{   public static void main(String args[ ])
    {   System.out.println("欢迎使用 Java 语言编写程序！");}
}
```

在例 2.1 程序中，第一行利用 import 语句将 Java 语言定义好的类或包加载到本程序中，它类似于 C 程序的#include 语句。程序第二行中的关键字 class 说明一个类定义的开始。类定义由类头 public class L0201 和类体组成。类体部分的内容由一对大括号括起来，并且在类体内部不能再定义其他的类。任何一个 Java 程序都是由若干个这样的类定义组成的，就像 C 程序都是由若干个函数组成的一样。例 2.1 的 Java 程序中只定义了一个类，类名字是 L0201。在此需要特别指出的是，Java 语言和 C 语言一样对大小写很敏感，因此，class、Class 和 CLASS 在 Java 程序中代表不同的含义。定义类使用关键字 class 作为标志。

类体通常由两部分组成：一部分是域（也称为成员变量），它包括变量、常量和对象数

组等独立的实体；另一部分是方法，也就是 C 语言中的函数或其他语言中的过程，它是代码单元块。这两种组成部分通常被称为类的成员。在例 2.1 中只有一个成员，即 main 方法。本例中的“ public static void main(String args[ ]) ”是 main 方法的方法头，而“ { System.out.println ("欢迎使用 Java 语言编写程序！");} ”则是 main 方法的方法体。main 是 main 方法的名字，String args[ ]是 main 方法的形式参数，public static 是 main 方法的属性修饰符。void 表示 main 方法没有返回值。方法体部分由若干以分号结尾的语句组成，并由一对大括号对括起来。在方法体内不能再定义其他的方法，即方法是不能被嵌套定义的。

main 方法是一个特殊的方法，它是所有的 Java Application 程序执行的入口点，任何一个 Java Application 方法都必须有且只能有一个 main 方法，而且这个 main 方法的头部分必须按照如下的格式书写：

```
public static void main(String args[ ])
```

Java Application 程序将从 main 方法体中的第一个语句开始执行。例 2.1 中的 main 方法体只有一个语句，即：

```
System.out.println("欢迎使用 Java 语言编写程序！");
```

这个语句的作用是把字符串“ 欢迎使用 Java 语言编写程序！”输出到系统的标准输出设备上，一般是系统的显示屏。其中 System 是系统内部定义的一个系统对象；out 是 System 对象中的一个域，也是一个对象；println 是对象 out 的一个方法，其作用是向系统的标准输出设备输出其参数所指定的字符串，并回车换行。

需要特别注意的是，Java 语言程序的文件名必须以其主类名命名，否则在程序编译时将出现错误。

2.1.2 字节码文件的生成

高级语言程序从源代码到目的代码的生成过程称为编译。在 Java 程序中，源代码经过编译所得到的目的代码称为字节码。由于字节码是二进制的，因此编程人员无法直接读懂。而要由 Java 语言的解释器来解释执行。编译字节码需要使用专用的 Java 编译器来编译程序。在集成化的 Java 开发环境中，如 Jbuilder9 和 Visual j++等，只要选择一个菜单命令或单击某一个按钮就可以完成这个编译过程，而在 JDK 这样的命令行开发工具中，则需要运行独立的编译程序。

JDK 有多个版本，一般有 1.02、1.1.x、1.2 和 1.3 等系列，其高级版本可以向低级版本兼容。把 JDK 软件包从网上下载到本地机之后，就可以使用命令行调用专用的 Java 编辑器生成字节码。例 2.1 中的源程序可以使用以下命令使其生成字节码文件：

```
javac  L0201.java
```

如果没有错误发生，系统就会生成对应的字节码文件（L0201.class）；如果有错误发生，则编译器会在屏幕上给出错误所在的源程序的行号和提示信息，这样，用户便可以依据提示修改源程序，然后再进行编译，直到源程序调试通过。

在 C 语言等其他高级语言的编译过程中，通常是一个源代码文件生成一个目的文件，而 Java 程序的编译则是对应源代码文件中定义的每一个类生成一个以这个类名字命名、以.class 为后缀的字节码文件。例 2.2 是定义了两个类的 Java 程序的例子。

【例 2.2】定义两个类的源代码文件 L0202.java。