21世纪高职高专计算机专业教材

董云铮　陈　千　梅慧平　编著

Java技术应用

清华大学出版社
http://www.tup.com.cn
北京交通大学出版社
http://press.bjtu.edu.cn

21 世纪高职高专计算机专业教材

Java 技术应用

董云铮　陈　千　梅惠平　编著

清 华 大 学 出 版 社
北京交通大学出版社
·北京·

内 容 简 介

本书立足于满足广大初学者入门和提高及 Java 编程 IT 国际认证考试的需要，系统全面地介绍了 Java 语言程序设计的基本概念和实际应用。教材内容由浅入深，由例子引出概念，便于读者更好地理解和掌握。全书共分 9 章，可以分为两大部分，第一部分是第 1 章～第 7 章，主要介绍 Java 基础知识、面向对象的程序设计、图形用户界面及 Applet 程序设计，这些内容可以满足应用 Java 开发交互式网页的基本要求；第二部分是第 8 章～第 9 章，介绍多线程编程和网络编程方面的知识，体现了 Java 特有的其他可重用类的使用。

本书以实用为基本出发点，除包含 Java 语言的基本概念外，也强调 Java 具体的应用。书中的所有实例均通过编译运行。本书重点突出，详略得当，适合作为高等职业技术学院、成人高校、各类培训班的教材或参考书，也适合广大 Java 语言爱好者或相关从业人员自学之用。

图书在版编目（CIP）数据

Java 技术应用 / 董云铮，陈千，梅惠平编著. —北京：清华大学出版社；北京交通大学出版社，2007.4
（21 世纪高职高专计算机专业教材）
ISBN 978-7-81082-939-7

Ⅰ. J… Ⅱ. ①董… ②陈… ③梅… 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（2007）第 005888 号

责任编辑：谭文芳　特邀编辑：王芳芳
出版发行：清 华 大 学 出 版 社　邮编：100084　电话：010-62776969
北京交通大学出版社　邮编：100044　电话：010-51686414
印 刷 者：北京东光印刷厂
经　　销：全国新华书店
开　　本：185×260　印张：12.75　字数：320 千字
版　　次：2007 年 4 月第 1 版　2007 年 4 月第 1 次印刷
书　　号：ISBN 978-7-81082-939-7/ TP・331
印　　数：1～4 000 册　定价：19.00 元

前　言

Java 是 1995 年 6 月由 Sun Microsystems 公司推出的一种革命化语言，具有易用性、平台无关性、易移植性等诸多特征，因此它在短时间内得到了迅速的发展和广泛的应用。较之其他语言，Java 语言的程序具有开发费用更小，工作效率更高，拥有更为优良的用户界面和更强大的开发工具，网上数据编程体现更充分等优点，因此 Java 被很多人看作是目前最有发展前景的语言之一。

为满足广大计算机编程初学者入门和提高的要求，以及满足高职高专、成人高校教学之用，我们组织编写了这本教材。本书的编写按照由浅入深、循序渐进的原则进行组织。书中引入大量的例子，在例子中渗透相关的理论知识，便于读者更好地理解和掌握。相关的难点被分解到例子中，降低了学习台阶，同时使用通俗易懂的语言和丰富的例题解释清楚复杂的概念，避免一开始就引入过多的技术术语和介绍过多的 Java 特点，以减少编程初学者理解的困难。书中例子的选择，遵循典型、详细的原则，以培养初学者的编程思想和兴趣，逐步提高编程能力。

为了能让初学者更好的掌握和理解 Java 编程技术，本书一改传统的编排方式，采用的内容编排顺序是：从实践到理论，从具体到抽象，从个别到一般，从零散到系统。同时，考虑到学生在学习过程中的接受心理，本书采用先提出问题，再介绍解决问题的方法，最后归纳出一般规律或概念。教学实践证明这种方法是行之有效的，减少了初学者在学习上的困难。这种内容编排方式非常适合高职院校的学生和入门初学者。

全书共分 9 章，每一章均配有内容提要和习题。全书大致可以分为两大部分，第一部分是第 1 章～第 7 章，主要介绍 Java 基础知识、面向对象的程序设计、图形用户界面及 Applet 程序设计。这些内容可以满足应用 Java 开发交互式网页的基本要求。第二部分是第 8 章～第 9 章，介绍多线程编程和网络编程方面的知识，体现了 Java 特色的其他可重用类的使用。其中第 1、2、3、6、7 章由董云铮编写，第 4、5 章由梅惠平编写，第 8、9 章由陈千编写。

由于时间仓促，加之作者水平有限，书中难免存在疏漏之处，敬请专家和读者不吝指正。

编　者

2007 年 1 月

目　录

第 1 章　Java 概述 …… 1
1.1　Java 的发展和特点 …… 1
1.2　Java 程序的工作机制 …… 2
1.3　Java 应用程序和 Java 小应用程序 …… 2
1.4　Java SDK …… 3
1.4.1　Java SDK 的安装 …… 3
1.4.2　Java SDK 的使用 …… 4
1.5　Java 程序举例 …… 5
小结 …… 10
习题 …… 10
第 2 章　Java 语言基础 …… 11
2.1　标识符、保留字和分隔符 …… 11
2.1.1　标识符 …… 11
2.1.2　保留字 …… 11
2.1.3　分隔符 …… 12
2.2　数据类型 …… 12
2.2.1　变量和常量 …… 12
2.2.2　基本类型 …… 13
2.3　运算符及表达式 …… 16
2.3.1　表达式 …… 17
2.3.2　运算符 …… 17
2.4　数据类型转换 …… 21
2.4.1　自动类型转换 …… 21
2.4.2　强制类型转换 …… 22
2.5　数组 …… 22
2.5.1　一维数组 …… 22
2.5.2　多维数组 …… 25
2.6　流程控制 …… 28
2.6.1　选择语句 …… 28
2.6.2　循环语句 …… 31
小结 …… 35
习题 …… 35
第 3 章　面向对象程序设计 …… 36

3.1 面向对象的理论基础 ……36
3.2 类与对象的基本概念 ……36
3.2.1 对象 ……36
3.2.2 类 ……36
3.3 Java 的类 ……37
3.3.1 简单的例子 ……37
3.3.2 Java 类 ……39
3.4 对象的创建与使用 ……41
3.4.1 对象说明 ……41
3.4.2 对象的实例化和初始化 ……42
3.5 方法说明 ……42
3.5.1 方法首部说明 ……43
3.5.2 方法体 ……45
3.5.3 构造方法 ……46
3.5.4 this 关键字 ……48
3.5.5 方法的调用 ……48
3.6 继承 ……51
3.6.1 类继承的实现 ……51
3.6.2 抽象类和抽象方法 ……55
3.6.3 最终类 ……56
3.7 接口 ……56
3.7.1 接口的概念 ……56
3.7.2 接口的说明 ……56
3.7.3 接口的使用 ……58
3.8 包 ……58
3.8.1 包的创建 ……59
3.8.2 包的使用 ……59
小结 ……60
习题 ……60
第 4 章 异常处理 ……62
4.1 异常类介绍 ……62
4.1.1 异常的概念 ……62
4.1.2 异常的类层次 ……63
4.2 异常处理 ……63
4.2.1 try…catch…finally 语句 ……63
4.2.2 多异常的处理举例 ……65
4.3 自定义异常 ……67
4.3.1 自定义异常类设计 ……67
4.3.2 抛出异常 ……67

4.3.3 throws 抛出异常 ……68
4.3.4 throw 抛出异常 ……69
4.4 实训：异常使用 ……72
小结 ……76
习题 ……77
第 5 章 Java 小应用程序 Applet ……78
5.1 Applet 程序开发步骤 ……78
5.2 Applet 的几种常用方法 ……81
5.3 Applet 类 ……83
5.3.1 应用文字 ……83
5.3.2 颜色控制 ……85
5.3.3 绘制图形 ……87
5.3.4 绘制图像 ……89
5.3.5 播放声音 ……92
5.4 Applet 的事件及其处理 ……94
5.4.1 处理鼠标事件 ……94
5.4.2 处理键盘事件 ……96
5.4.3 按钮事件处理 ……98
5.5 实训：Applet 的应用 …… 100
5.5.1 在 HTML 文件中给 Applet 提供参数 …… 100
5.5.2 Applet 代码中读取 Applet 参数值 …… 100
5.5.3 访问 WWW 资源 …… 101
5.5.4 远程浏览 Applet …… 103
5.5.5 Applet 的综合实例运用 …… 103
小结 …… 109
习题 …… 110
第 6 章 流式输入输出 …… 111
6.1 输入输出基本概念 …… 111
6.1.1 输入输出设备与文件 …… 111
6.1.2 流的概念 …… 111
6.2 面向字节的输入输出流 …… 112
6.2.1 面向字节的输入流 …… 112
6.2.2 面向字节的输出流 …… 116
6.3 面向字符的输入输出流 …… 118
6.3.1 面向字符的输入流 …… 118
6.3.2 面向字符的输出流 …… 120
小结 …… 121
习题 …… 122
第 7 章 图形用户界面及事件处理 …… 123

7.1 AWT 概述 …… 123
7.1.1 AWT 的基本组件 …… 123
7.1.2 AWT 中的容器组件 …… 124
7.2 AWT 基本组件和容器组件 …… 124
7.2.1 按钮 …… 124
7.2.2 标签 …… 125
7.2.3 文本框和多行文本框 …… 126
7.2.4 复选框 …… 129
7.2.5 单选框 …… 130
7.2.6 下拉式列表 …… 131
7.2.7 列表 …… 132
7.2.8 面板 …… 133
7.2.9 框架 …… 133
7.2.10 对话框 …… 134
7.2.11 菜单 …… 134
7.3 布局管理器 …… 136
7.3.1 FlowLayout 管理器 …… 136
7.3.2 BorderLayout 管理器 …… 138
7.3.3 GridLayout 管理器 …… 139
7.3.4 CardLayout 管理器 …… 141
7.4 委任事件模式 …… 143
7.5 ActionListener 及 ActionEvent …… 143
7.6 TextListener 及 TextEvent …… 154
7.7 ItemListener 及 ItemEvent …… 156
小结 …… 159
习题 …… 159
第 8 章 多线程编程 …… 160
8.1 多线程机制介绍 …… 160
8.2 创建线程 …… 161
8.2.1 主线程 …… 161
8.2.2 通过继承 Thread 类创建线程 …… 162
8.2.3 通过实现 Runnable 接口创建线程 …… 163
8.3 线程同步 …… 164
8.3.1 线程同步问题 …… 164
8.3.2 同步代码块 …… 166
8.3.3 同步方法 …… 168
8.4 线程通信 …… 169
8.5 实训：线程应用 …… 172
小结 …… 173

习题 …… 173
第 9 章　网络编程 …… 174
9.1　网络编程的基础知识 …… 174
9.1.1　IP 地址和端口号 …… 174
9.1.2　UDP 和 TCP …… 174
9.1.3　Socket …… 175
9.2　使用 TCP 协议编写网络程序 …… 175
9.2.1　ServerSocket 类 …… 176
9.2.2　Scoket 类 …… 176
9.2.3　InetAddress 类 …… 177
9.2.3　简单的 TCP 服务器端程序 …… 178
9.2.4　多线程的 TCP 服务器程序 …… 180
9.2.5　TCP 客户端程序 …… 181
9.3　使用 UDP 协议编写网络程序 …… 182
9.3.1　DatagramSocket …… 182
9.3.2　DatagramPacket …… 183
9.3.3　简单的 UDP 发送和接收程序 …… 183
9.4　实训：网络编程应用 …… 185
小结 …… 190
习题 …… 190
参考文献 …… 191

第 1 章　Java 概述

本章要点

- ☑ Java 的安全性
- ☑ Java 的可移植性

1.1　Java 的发展和特点

1990 年，美国 Sun 公司的 James Gosling、Bill Joe 等人，为在电视、控制烤箱等家用消费类电子产品上进行交互式操作而开发了一种与平台无关、可靠性强、小而灵活的编程语言，但当时并没有引起人们的注意。直到 1994 年下半年，Internet 的迅猛发展，全球信息网 WWW 的快速增长，人们才发现 Java 这种中性平台及其可靠性强的语言恰恰就是 www 需要的语言。Java 的开发人员对 Java 进行了一系列的改进，融合了 C 和 C++等语言的优点，形成了现在这套与众不同的面向对象程序设计语言。

Java 的原名叫 Oak（橡树），但在申请注册商标时，发现 Oak 已经被人使用了。在想了一系列名字后，最终，使用了提议者在喝一杯 Java 咖啡时无意提到的 Java。

Java（JDK 1.0）正式发表于 1995 年 5 月。Java 的“Write Once, Run Anywhere”口号使得 Java 一出现就引起了广泛的注意，用 Java 开发的软件可以不用修改或重新编译而直接应用于任何计算机上。Java 语言的众多优点使得它逐渐成为 Internet 上受欢迎的开发与编程语言。Java 的诞生对传统的计算机模型提出了新的挑战。

Java 是一门迅速发展的网络编程语言，它是一种新的计算概念。

首先，作为一种程序设计语言，它简单、面向对象、分布式，解释执行不依赖机器的结构，具有可移植性、安全性，并且是多线程的、动态的，具有很高的性能。

其次，Java 最大限度地利用了网络。被称为 Java 小程序（Applet）的 Java 程序是动态、安全、跨平台的网络应用程序，可在网络上运行而不受 CPU（中央处理器）和操作系统环境的限制。将 Java Applet 嵌入 HTML 语言中，通过 Web 页发布到 Internet 上。网络用户访问服务器的 Applet 时，这些 Applet 从网络上进行传输，然后在支持 Java 的浏览器中运行。由于 Java 语言的安全机制，用户一旦载入 Applet，就可以放心地生成多媒体的用户界面或完成复杂的计算而不必担心病毒的入侵。虽然 Applet 可以和图像、声音、动画等一样从网络上下载，但它并不同于这些多媒体文件格式，它可以接收用户的输入，动态的进行改变，而不仅仅是动画的显示和声音的播放。

另外，Java 还提供了丰富的类库，以满足网络化、多线程、面向对象系统的需要，使程序设计者可以很方便地建立自己的系统。Java 语言包提供包括字符串处理、多线程处理、异

常处理、数学函数处理等多种支持，可以用它简单的实现 Java 程序的运行平台。Java 实用程序包提供包括哈希表、堆栈、可变数组、时间和日期等支持。Java 输入输出包用统一的“流”模型来实现所有格式的输入输出。抽象图形用户接口包实现了不同平台计算机的图形用户接口组件，包括窗口、菜单、滚动条、对话框等，使 Java 可以移植到不同平台的机器中去。

1995 年是微软发展到颠峰的一年，Windows 95 风光进军市场，尽管如此，比尔·盖茨仍然敏锐地注意到 Java，并给予了这样的评价：“Java 是很长时间以来最优秀的程序设计语言。”基于此，微软于 1996 年 3 月申请并获得了 Java 许可证。微软对于 Java 的热情关注，大大提高了人们对 Java 的兴趣和信心，加快了 Java 的普及使用。

1.2 Java 程序的工作机制

Java 语言具有高安全性、高移植性等优点，这与它的工作机制有关。下面介绍 Java 的工作机制，这将更有助于理解 Java 语言的特点。

对于运行在 Internet 上的网络应用程序，需要有良好的可移植性。因为 Internet 是由各种各样不同类型的终端、服务器和计算机等硬件设备组成的，而且在这些设备上运行的软件系统也是多种多样的，所以 Internet 上的网络应用程序应该具有在各种不同的软硬件平台上均可正常工作的能力。以前的各种优秀语言都无法圆满解决这个问题，而 Java 的工作机制使得它具有了这样的能力。

Java 的工作机制是这样的：编程人员首先编写好源代码，然后经编译生成一种二进制的中间码，称为字节码（byte code），最后再通过运行与操作系统平台环境相对应的、一种称为 Java 解释器的运行机构来执行编译生成的字节码。虽然不同的平台环境需要有各自相应的解释器，但是任何一个平台上的解释器，对于一段 Java 程序的字节码来说却是相同的，因为它们对 Java 字节码呈现出完全相同的面貌。也就是说，Java 的运行机制可以利用解释器来隐藏网络上平台环境的差异性。由此可见，Java 实现了二进制代码级的可移植性，在网络上实现了跨平台的特性。

Java 的解释器又称为“Java 虚拟机（Java Virtual Machine，JVM）”，是驻留于计算机内存的虚拟计算机或逻辑计算机，实际上是一段负责解释执行 Java 字节码的程序。JVM 能够从字节码流中读取指令并解释指令的含义，每条指令都含有一个特殊的操作码，JVM 能够识别并执行它。从这个意义上说，Java 可以被称为是一种“解释型”的高级语言。高级语言的解释器对程序边解释边执行，执行效率较低。因此，运行 Java 程序比运行 C 或 C++等“编译型”语言程序速度要慢一些，这是 Java 语言的一个不足。随着硬件的发展，这个速度差别正变得越来越小，而网络的飞速发展，这种体系结构中立、平台无关的特性却显得越来越重要。

1.3 Java 应用程序和 Java 小应用程序

Java 可用来生成两类程序：应用程序（Applications）和小应用程序（Applet）。

应用程序是可以在我们的计算机操作系统中运行的程序，从这一方面来说，用 Java 编制的应用程序在某种程度上与使用 C 或 C++编制的应用程序有些类似。在创建应用程序的时候，Java 与其他计算机语言的确没有多大区别。

Java 的重要性就在于它具有编制小应用程序的功能。小应用程序是可以在 Internet 中传输并在兼容 Java 的 Web 浏览器中运行的应用程序。小应用程序实际上就是小型的 Java 程序，能像图像文件、声音文件和视频片段那样通过网络动态下载。它与其他文件的重要差别是，小应用程序是一个职能的程序，能对用户的输入做出反应，并且能动态变化，而不是一遍又一遍地播放同一动画或声音。

人们在网络上下载程序的时候，常会有被病毒感染、被恶意代码控制的危险。Java 在网络应用程序和计算机之间提供了一道防火墙，消除了用户的这些顾虑。

在网络上下载 Java 小应用程序是绝对安全的。Java 实现这种保护功能的方式是将 Java 程序限制在 Java 运行环境中，不允许它访问计算机的其他部分。

1.4　Java SDK

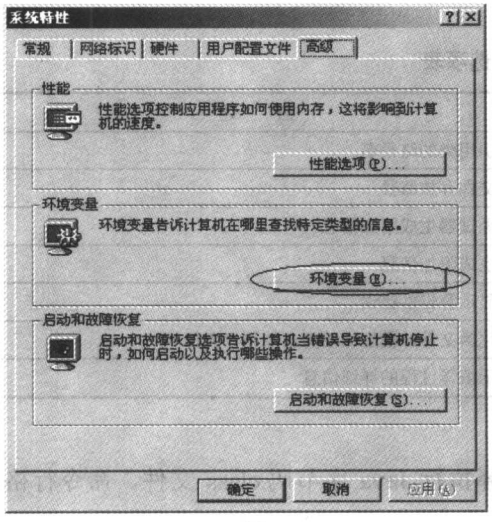
1.4.1　Java SDK 的安装

Java SDK 的意思是 Software Development Kit，即 Java 软件开发工具包，是由 Sun 公司开发的，用户可在网上下载。

以 Windows 系统为例，JDK 的安装与环境配置如下。

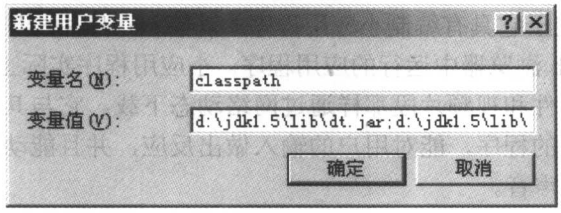
Windows 下，直接运行 exe 文件，安装到一个目录，例如将其安装到 D:\jdk1.5 目录下面。

安装完毕后，Java 源代码就可以运行了，但是它只能在 D:\jdk1.5\bin\下运行，这时候就需要配置环境变量，使得任何地方都可以运行 Java 程序。配置过程如下：从桌面上选择【我的电脑】(右键)，选择【属性】，在跳出对话框中选择【高级】，然后选择【环境变量】，如图 1-1(a)所示。这时候跳出【环境变量】对话框，这时在“Administrator 的用户变量”一栏中选择【新建】，如图 1-1(b)所示。这时会跳出如图 1-1(c)的对话框，输入变量名和变量值。



(a)

(b)



(c)

图 1-1 J2SDK 安装示意图

在变量名中输入 classpath，变量值中输入类库名 D:\jdk1.5\lib\dt.jar;D:\jdk1.5\lib\tools.jar;（注意要输入实际安装路径），然后选择【确定】。再次选择“Administrator 的用户变量”的【新建】，在变量名中输入 path，变量值中输入 D:\jdk1.5\bin，然后选择【确定】。或者直接修改自动批处理文件 Autoexec.bat，利用任何编辑器编辑该文件，然后在其中加入如下的设置语句：

```
set path=%path%;d:\jdk1.5\bin;
set classpath=.;d:\jdk1.5\lib\dt.jar;d:\jdk1.5\lib\tools.jar;
```

在配置完成后，要重新启动计算机，环境变量配置才能生效。

1.4.2 Java SDK 的使用

1. javac

javac.exe 文件是 Java 语言的语言编译器。该编译器读取 Java 程序源代码文件，并将其编译成类文件（一组*.class 文件），类文件中包含有 Java 字节码。调用 javac.exe 的命令行中指定程序源文件时，必须有文件扩展名.java。命令行格式如下：

```
javac [选项] FileName.java
```

其中选项的定义如表 1-1 所示。

表 1-1 javac 选项表

名　称	描　述
-classpath<path>	被引用类的路径名
-d <directory>	类文件存放路径
-g	为调试器生成附加信息
-ng	不生成附加信息
-nowarn	不显示警告性错误
-o	优化类文件，使生成的类文件不包含行号
-verbose	显示编译过程的详细信息

2. java

java.exe 文件是 Java 语言的解释器，用来解释执行 Java 字节码.class 文件。命令行格式如下：

java [选项] ClassName <参数表>

类的字节码在被称为 className.class 的文件中，这个文件是由 javac.exe 编译类文件源代码产生的，所有 Java 字节码文件都有.class 扩展名，扩展名是在编译时自动加上的。ClassName 中必须包含一个 main()方法。一般情况下，命令行中指明字节码文件名即可，文件扩展名.class 在命令中不需要写出。<参数表>是 ClassName 类的参数。

表 1-2　java 选项表

名　称	描　述
-classpath<path>	设置类搜索路径
-cs 或-checksource	检查类文件和源程序之间的一致性
-dpropertyname=value	设置属性值
-debug	允许调用 Java 调试器 jdb
-ms initmem[klm]	设置初始内存空间
-mx maxmem[klm]	设置最大内存空间
-noasyncgc	无用空间搜集程序的自动搜集动作
-noverify	不检验类文件
-oss stacksize[klm]	设置每线程的 Java 代码栈大小
-ss stacksize[klm]	设置每个线程的原始码栈大小
-v,-verbose	显示类装载信息
-verbosegc	显示无用空间搜集程序的运作信息
-verify	检验所装载的全部类文件
-verifremote	检验类装载器所装载的类文件

3. appletviewer

appletviewer.exe 提供了一个 Java Applet 运行环境，在其中可测试 Java Applet。appletviewer 读取包含 Java 小程序的 HTML 文件，并在一个窗口中运行它们。命令行格式如下：

appletviewer [选项] URL

URL 表示由 URL 描述的 HTML 文档，要指出文件的扩展名 html 等。

1.5　Java 程序举例

了解过 JDK 的安装和基本使用方法后，下面介绍两个具体的小程序。

例 1-1　输出信息“Hello World!”的 Java 应用程序。

```
/*一个简单的 Java 应用程序，显示内容：Hello World!
文件名为 Hello.java*/
class Hello{
    //程序开始于 main()
  public static void main(String args[]){
      System.out.println("Hello World!");
```

```
    }
}
```

1. 输入程序

对于大多数计算机语言来说，包含程序源代码的文件名是任意的，但 Java 就不是这样。关于 Java，源文件的名字必须和类名一致。在本例中，文件名必须命名为“Hello.java”。输入的时候，我们可以在记事本里输入程序，然后选择【另存为】即可。如图 1-2 所示：

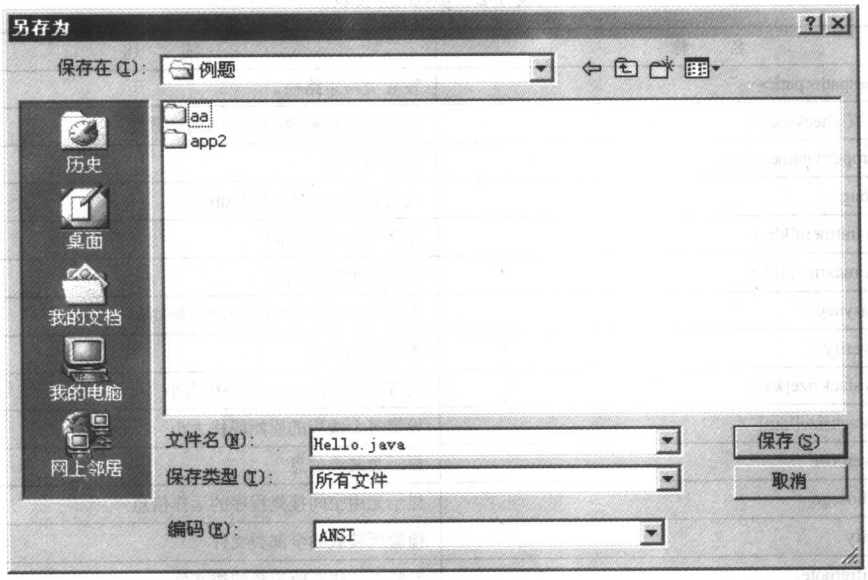


图 1-2 文件保存样式

在 Java 里面是区分大小写的，因此写成 hello.java 和 Hello.java 是完全不同的两个名称，如果例 1-1 的文件命名为前者，则会提示编译出错。虽然文件名必须与类名一致的约定似乎有点专制，但是这个约定有助于用户轻松地维护及组织程序。

2. 编译程序

要编译示例程序 Hello，必须运行编译器程序 javac，并在命令行上指定源程序文件名，格式如下：

```
javac Hello.java
```

编译器 javac 产生了一个名为 Hello.class 的文件，该文件包含程序的字节码。Java 字节码中包含的是 Java 解释程序将要执行的指令码。因此，javac 的输出并不是可以直接运行的代码。要真正运行该程序，你必须使用名叫 Java 的 Java 解释器。具体做法是把类名 Hello 作为一个命令行参数输入，格式如下：

```
Java Hello
```

运行这个程序，将输出如下内容：

```
Hello World!
```

参考效果如图 1-3 所示：

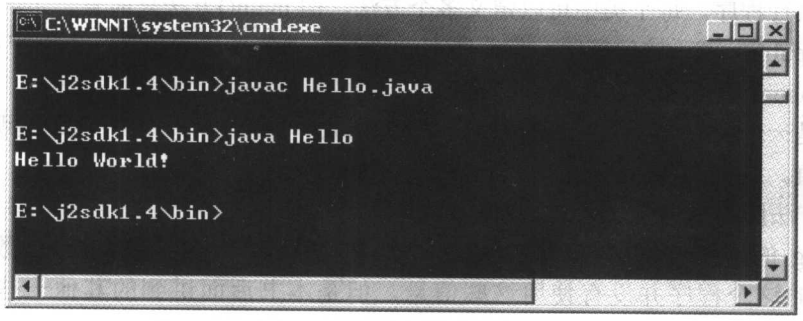


图 1-3　Hello World 运行示例

3. 详细讨论

尽管 Hello.java 很短，但它包括了所有 Java 程序具有的几个关键特性。我们仔细分析一下该程序的每个部分。

程序开始于以下几行：

```
/*一个简单的 Java 应用程序，显示内容：Hello World!
文件名为 Hello.java*/
```

这是一段注释。像大多数其他的编程语言一样，Java 也允许你在源程序文件中加注释。注释中的内容将被编译器忽略。事实上，注释是为了给任何阅读源代码程序的人说明或解释程序的操作的。在本例中，注释对程序进行了说明，并提醒用户该程序的名字应命名为“Hello.java”。当然，在真正的应用中，注释通常用来解释程序的某些部分如何工作或某部分有什么特殊功能。

Java 支持 3 种类型的注释，例题中开头两行注释称为多行注释。它开始于“/*”，结束于“*/”。它允许包含多行文字，这两个注释符之间的任何内容都将被编译器忽略。

下面我们看下一行代码：

```
class Hello{
```

该行使用关键字 class 声明了一个新类，Hello 是类名，整个类定义都将位于一对花括号（{}）之间，花括号在 Java 中的使用方式与 C 或 C++相同。在 Java 中，所有程序活动都发生在类内，这就是为什么说 Java 程序是面向对象的。

下面一行程序是单行注释：

```
//程序开始于 main()
```

这是 Java 支持的第二种类型的注释。单行注释开始于“//”，在该行的末尾结束。通常情况下，程序员们对于较长的注释使用多行注释，而对于简短的注释则采用单行注释。

下一行代码如下所示：

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

该行开始于 main()方法，正如注释所说，这是程序将要开始执行的第一行。对于一个 Java

应用程序来说，main()方法是必需的，Java 解释器在没有生成任何对象的情况下，以 main()作为入口来执行程序。每个类中可以定义多个方法，但 main()方法只能有一个。

关键字 public 表示访问权限，指明所有的类都可以使用这一方法。本例中，main()必须被定义为 public 类型，因为当程序开始执行时它需要被它的类之外的代码调用。关键字 static 指明该方法是一个类方法（类方法的概念在第 3 章具体介绍）。关键字 void 指明 main()方法不返回任何值。这些概念，在后面的章节中将会进行详细讨论。

main()方法圆括号()中定义的 String args[]是传送给 main()方法的参数，参数名为 args，它是类 String 的一个对象。方法的参数用“类名 参数名”来指定，多个参数间用逗号分隔。本例中的这个程序并没有使用这些信息，但是本书后面讲到的其他一些程序将使用它们。

该行的最后一个字符是“{”，它表示了 main()程序体的开始，一个方法中包含的所有代码都将包括在这对花括号中间。

另外，main()仅是解释器开始工作的地方，一个复杂的程序可能包含几十个类，但这些类仅需要一个 main()方法。当编写 Java 的小应用程序时，根本不用使用 main()方法，Web 浏览器使用另一种不同的方法启动小应用程序。

接下来的代码行如下所示（请注意，它出现在 main()内）：

```
System.out.println("Hello World!");
```

本行代码的作用是在屏幕上输出字符串“Hello World!”，输出结果后面带一个回车，这个输出任务实际上是由内置方法 println()来实现的。在这里，println()的作用是显示传递给它的字符串，当然 println()方法也能用来显示其他类型的信息。该行开始的 System.out 涉及很多复杂的内容。我们后面会作详细介绍，简单地说，System 是一个预定义的可访问系统的类，out 是连接到控制台的输出流。

最后我们注意到该语句的末尾是以一个分号结束的，在 Java 中，所有的语句都以一个分号结束。该程序的其他行没有以分号结束，这是因为从技术上讲，它们并不是程序语句。

最后是两个“}”符号，第一个“}”结束了 main()，而最后一个“}”结束类 Hello 的定义。

对于编程语言来说，变量是一个最为基本的概念。下一个程序将介绍如何声明变量，如何给变量赋值。

例 1-2 变量使用示例。

```
/*
   这是简单变量使用示例
   文件应该命名为 Example.java
 */
class Example{
public static void main(String args[]){
      int num;          //声明变量 num
      num=100;//给变量 num 赋值 100
      System.out.println("这是 num: "+num);
      num=num*2;
```