

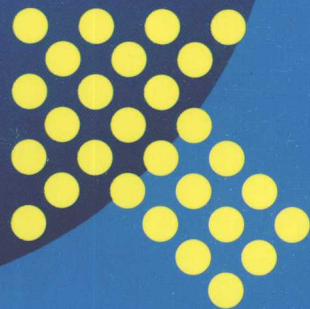
21世纪高等学校规划教材



JAVA YUYAN CHENGXU SHEJI

Java 语言程序设计

王大虎 陈 玮 霍占强 主编



中国电力出版社

<http://jc.cepp.com.cn>

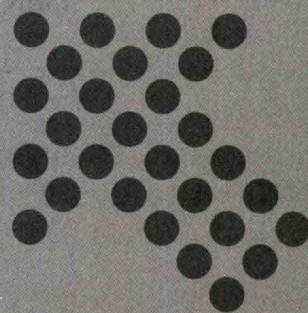
21世纪高等学校规划教材



JAVA YUYAN CHENGXU SHEJI

Java 语言程序设计

主 编 王大虎 陈 玮 霍占强
编 写 罗军伟 刘小燕 吴振江
吴 岩 任多奇 刘 浩
主 审 杜春涛



中国电力出版社

<http://jc.cepp.com.cn>

内 容 提 要

本书为 21 世纪高等学校规划教材。全书共分 11 章, 主要包括绪论、Java 语法基础、类和对象、继承、多态和接口、常用系统类、图形用户界面设计、异常处理、输入输出流、多线程、网络编程、JDBC 等, 每章后附有习题。全书内容丰富, 通俗易懂。

本书可作为普通高等院校本、专科 Java 语言程序设计课程的教学用书, 也可作为计算机等级考试及计算机培训班中相关 java 课程的教材或参考书, 还可作为有关工程技术人员的工作参考书。

图书在版编目 (CIP) 数据

Java 语言程序设计 / 王大虎, 陈玮, 霍占强主编. —北京:
中国电力出版社, 2010.8

21 世纪高等学校规划教材

ISBN 978-7-5123-0720-9

I. ①J… II. ①王… ②陈… ③霍… 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 (2010) 第 149048 号

中国电力出版社出版、发行

(北京三里河路 6 号 100044 <http://jc.cepp.com.cn>)

汇鑫印务有限公司印刷

各地新华书店经售

*

2010 年 8 月第一版 2010 年 8 月北京第一次印刷
787 毫米×1092 毫米 16 开本 18.5 印张 450 千字
定价 29.80 元

敬 告 读 者

本书封面贴有防伪标签, 加热后中心图案消失
本书如有印装质量问题, 我社发行部负责退换

版 权 专 有 翻 印 必 究

前 言

今天,从嵌入式电子设备及海量发行的智能卡、电子商务、电子政务等应用平台的开发到电信运营商、移动终端和 PDA 等系统的开发,都可以看到 Java 程序的应用。Java 已经无处不在,Java 开发人员数量也日益增多。另外,Java 开发者们锐意创新,将 Java 技术应用到各种领域,使得 Java 技术已经成为世界上最卓越的企业应用和移动应用开发平台之一。因此 Java 语言已经成为开发应用软件的主流开发语言之一。

本书的特点如下:

(1) 脉络清晰,易学好用。全书共分十一章,各章的结构安排合理,给读者一个全局的框架。

(2) 案例丰富,剖析透彻。全书共有 150 多个程序示例,均给出到位的分析。

(3) 适用面广,实用性强。从起点上说,本书兼顾了零起点和有语言基础的读者;从目标上说,本书适合 Java 程序设计基础目标和进阶目标的读者。

本书由河南理工大学的王大虎、陈玮和霍占强三位老师合作完成。王大虎负责第 1~第 4 章的编写,陈玮负责第 5~第 7 章的编写,霍占强负责第 8~第 11 章的编写。全书由王大虎统稿。河南省电力公司焦作供电公司的任多奇、刘浩等人也参与了各章的组织、编写工作。

本书在编写过程中得到了河南理工大学计算机学院高岩老师的支持和鼓励,2008 级研究生白鹤原、查小菲以及 2009 级研究生张彤、李林强、刘欢、黄明克等同学帮助查找文献、调试例题和习题程序、排版和校对等,做了大量工作。本书由杜春涛审稿,提出很多宝贵意见,在此一并表示感谢。

限于编者水平,书中难免存在欠妥之处,恳请读者和同仁批评指正。

编 者

2010 年 7 月

目 录

前言

第 1 章 绪论	1
1.1 Java 语言的发展史	1
1.2 Java 的工作原理——Java 虚拟机	3
1.3 Java 语言的特点	3
1.4 Java 的运行环境	5
1.5 最简单的 Java 程序	7
1.6 Java 的应用范围	10
小结	10
习题	10
第 2 章 Java 语法基础	12
2.1 Java 中的标识符和保留字	12
2.2 数据类型简介	12
2.3 常量和变量	13
2.4 运算符	16
2.5 控制语句	24
2.6 数组	32
小结	37
习题	38
第 3 章 类和对象	42
3.1 类和对象的基本概念	42
3.2 类的实例化	45
3.3 类的组织	51
3.4 方法	55
3.5 构造函数	62
小结	64
习题	65
第 4 章 继承、多态和接口	69
4.1 类的继承和多态基本概念	69
4.2 类的继承	71
4.3 覆盖	77
4.4 方法重载	82
4.5 抽象类和最终类	84
4.6 接口	86

小结	89
习题	90
第5章 常用系统类	92
5.1 Java 语言的类库简介	92
5.2 Java 语言常用类	93
小结	105
习题	105
第6章 图形用户界面 (GUI) 设计	106
6.1 GUI 概述	106
6.2 框架窗口	109
6.3 布局管理器	110
6.4 事件驱动程序设计	114
6.5 GUI 设计	121
小结	146
习题	146
第7章 异常处理	147
7.1 异常处理概述	147
7.2 异常和异常类型	148
7.3 异常处理	149
7.4 创建自定义异常	153
小结	154
习题	154
第8章 输入输出流	156
8.1 java.io.File 类	156
8.2 Java IO 原理	161
8.3 FileInputStream 和 FileOutputStream 类	165
8.4 FileReader 和 FileWriter 类	169
8.5 BufferedReader 和 BufferedWriter 类	171
8.6 DataInputStream 和 DataOutputStream 类	175
8.7 ObjectInputStream 和 ObjectOutputStream 类	178
小结	187
习题	187
第9章 多线程	188
9.1 线程的基本概念	188
9.2 线程的生命周期	195
9.3 线程的调度和优先级	199
9.4 线程的同步	200
9.5 Timer 类	209
小结	211

习题	211
第 10 章 网络编程	212
10.1 网络编程的基本概念	212
10.2 Java 网络类和接口	215
10.3 InetAddress 类	216
10.4 URL 和 URLConnection 类	217
10.5 Socket 套接字	224
10.6 Datagram 套接字	228
10.7 综合实例——聊天室程序	231
小结	243
习题	243
第 11 章 JDBC	244
11.1 JDBC 简介	244
11.2 JDBC 类和接口	244
11.3 JDBC 操作 SQL	250
11.4 JDBC 基本实例	255
11.5 JDBC 综合实例	264
小结	287
习题	287
参考文献	288

第 1 章 绪 论

Java 是目前应用软件开发时使用的主流语言之一,是 1995 年以后伴随着互联网的应用而蓬勃发展起来的。

学习目标:

- 熟悉 Java 语言的发展历史。
- 熟练掌握 Java 应用程序开发环境和 Java 程序工作原理。
- 学会简单 Java 应用程序和小应用程序的调试过程。
- 了解 Java 语言的应用领域。

1.1 Java 语言的发展史

Java 自 1996 年正式发布以来,经历了产生、成长和壮大的阶段,现在已经成为 IT 领域里的主流编程语言之一。Java 起源于 Sun 公司的一个叫“Green”的项目,目的是开发嵌入家用电器的分布式软件系统,使电器更加智能化。Green 项目一开始准备采用 C++ 语言,但是考虑到 C++ 语言太复杂,于是决定基于 C++ 语言开发一种新的 Oak 语言(即 Java 的前身)。Oak 是一种适用于网络编程的精巧而安全的语言,它保留了 C++ 语言的许多语法,但去除了明确的资源引用、指针算法与操作符重载等潜在的危险特性。并且 Oak 语言具有与硬件无关的特性,制造商只需要更改芯片,就可以将烤面包机上的程序代码移植到微波炉上或其他电器上,而不必改变软件,这就大大降低了开发成本。当 Oak 语言成熟时,全球 Internet 也在迅速发展。Sun 公司的开发小组认识到 Oak 非常适合于 Internet 编程。1994 年,他们完成了一个用 Oak 语言编写的早期的 Web 浏览器,称为 WebRunner,后改名为 HotJava,展示了 Oak 作为 Internet 开发工具的能力。1995 年, Oak 语言更名为 Java 语言(以下简称为 Java)。Java 的取名有一个趣闻。据说,有一天,几位 Java 成员组的会员正在讨论给这个新的语言取什么名字,当时他们正在咖啡馆喝着 Java(爪哇)咖啡。有一个人灵机一动说,就叫 Java 怎样,这得到了其他人的赞赏。于是,Java 这个名字就这样传开了。1996 年, Sun 公司发布 JDK 1.0, 计算机产业的各大公司(包括 IBM、Apple、DEC、Adobe、Silicon Graphics、HP、Oracle、Toshiba 和 Microsoft 等)相继从 Sun 公司购买了 Java 技术许可证,开发相应的产品。1998 年, Sun 公司发布了 JDK 1.2(从这个版本开始的 Java 技术都称为 Java 2)。Java 2 不仅兼容于智能卡和小型消费类设备,还兼容于大型服务器系统,它使软件开发商、服务提供商和设备制造商更加容易抢占市场机遇。这一开发工具极大地简化了编程人员编制企业级 Web 应用的工作,把一次编程到处使用的语言应用到服务器领域。

1999 年, Sun 公司把 Java 2 技术分成 J2SE、J2EE 和 J2ME。其中 J2SE 就是指从 1.2 版本开始的 JDK,它为创建和运行 Java 程序提供了最基本的环境。J2EE 和 J2ME 建立在 J2SE 的基础上, J2EE 为分布式的企业应用提供开发和运行环境, J2ME 为嵌入式应用(比如运行

在手机里的 Java 程序) 提供开发和运行环境。在进入 21 世纪以来, 随着 Web 技术成为展示和操作数据的事实标准, 企业利用 J2EE 平台对原来分散的子系统进行整合。尽管应用整合可以通过多种手段来实现, 但 J2EE 在出现后, 因其天生具备良好的开放性和可扩展性, 使之在应用整合和开发的过程中发挥了愈来愈显著的优势。J2EE 逐渐成为开发商创建电子商务应用的事实标准。Java 的公用规范 (Publicly Available Specification, PAS) 在 1997 年被国际标准化组织 (ISO) 认定, 这是 ISO 第一次破例接受一个具有商业色彩的公司作为公用规范 PAS 的提交者。总之, 面向对象的 Java 语言具备一次编程、任何地方均可运行的能力, 这使其成为服务提供商和系统集成商用以支持多种操作系统和硬件平台的首选解决方案。Java 作为软件开发的一种革命性的技术, 其地位已被确定。如今, Java 技术已被列为当今世界信息技术的主流之一。

Java 语言产生于 C++ 语言之后, 是完全的面向对象编程语言, 充分吸取了 C++ 语言的优点, 采用了程序员所熟悉的 C 语言和 C++ 语言的许多语法, 同时又去掉了 C 语言中指针、内存申请和释放等影响程序健壮性的部分, 可以说 Java 语言是站在 C++ 语言这个“巨人的肩膀上”前进的。Java 的发展历史见表 1-1。

表 1-1

Java 的发展历史

时 间	描 述
1991	Sun 公司进军消费电子产品 (IA) 市场
1991.4	Sun 成立 “Green” 小组, 以 C++ 为基础开发新的程序设计语言, 并将其命名为 Oak
1995	Green 小组升格为 First Person 公司, 他们将 Oak 的技术转移至 Web 上, 并把 Oak 改名为 Java
1993~1994	Web 在 Internet 上开始流行, 致使 Java 得以迅速发展并成功
1995.5	Sun 公司正式发布 Java 与 HotJava 产品
1995.10	Netscape 与 Sun 合作, 在 Netscape Navigator 中支持 Java
1995.12	微软公司 IE 加入支持 Java 的行列
1996.2	Java Beta 测试版结束, Java 1.0 版正式诞生
1997.2	Java 发展至 1.1 版, Java 的第一个开发包 JDK (Java Development Kit) 发布
1998	JDK 1.1 下载量超过 200 万次, JDK 1.2 (称 Java2) 发布, JFC/Swing 技术发布, JFC/Swing 被下载了 50 多万次
1999.7	Java 发展至 1.2 版, Java 被分成 J2SE、J2EE 和 J2ME, JSP/Servlet 技术诞生
2000.9	Java 发展至 1.3 版
2001.7	Java 发展至 1.4 版, Nokia 公司宣布到 2003 年将出售 1 亿部支持 Java 的手机, J2EE 1.3 发布
2003	5.5 亿台桌面上运行 Java 程序, 75% 的开发人员将 Java 作为首要开发工具
2004	J2SE 1.5 发布, 这是 Java 在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	JavaOne 大会召开, Sun 公司公开 Java SE6。此时, Java 的各种版本被更名, 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
2009.4	sun 公司因为经营不善被迫以 74 亿美元被 Oracle 公司收购

1.2 Java 的工作原理——Java 虚拟机

Java 虚拟机 (Java Virtual Machine, JVM) 是软件模拟的计算机,可以在任何处理器上(无论是在计算机上还是在其他电子设备上)安全并且兼容地执行保存在.class 文件中的字节码。Java 虚拟机的“机器码”保存在.class 文件中,有时也可以称之为字节码文件。Java 程序的跨平台主要是指字节码文件可以在任何具有 Java 虚拟机的计算机或者电子设备上运行,Java 虚拟机中的 Java 解释器负责将字节码文件解释成为特定的机器码进行运行。Java 源程序需要通过编译器编译成为.class 文件(字节码文件),Java 程序的编译和执行过程如图 1-1 所示。

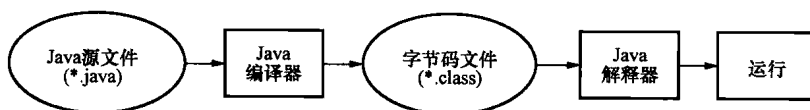


图 1-1 Java 程序的编译和执行过程

但是,Java 虚拟机的建立需要针对不同的软硬件平台做专门的实现,既要考虑处理器的型号,也要考虑操作系统的种类。目前在 SPARC 结构、X86 结构、MIPS 和 PPC 等嵌入式处理芯片上、在 UNIX、Linux、windows 和部分实时操作系统上都有 Java 虚拟机的实现,如图 1-2 所示,即 JVM 的工作方式。

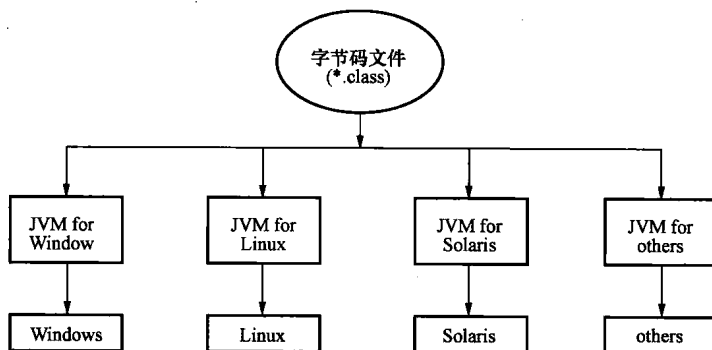


图 1-2 JVM 工作方式

1.3 Java 语言的特点

(1) 简单、面向对象

Java 的简单首先体现在精简的系统上,力图用最小的系统实现足够多的功能;对硬件的要求不高,在小型的计算机上便可以良好地运行。和所有的新一代程序设计语言一样,Java 也采用了面向对象技术并更加彻底,所有的 Java 程序和 applet 程序均是对象,封装性实现了模块化和信息隐藏,继承性实现了代码的复用,用户可以建立自己的类库。而且 Java 采用的是相对简单的面向对象技术,去掉了运算符重载、多继承的复杂概念,而采用了单一继承、类强制转换、多线程、引用(非指针)等方式。无用内存自动回收机制也使得程

程序员不必费心管理内存，使程序设计更加简单，同时大大减少了出错的可能。Java 语言采用了 C 语言中的大部分语法，熟悉 C 语言的程序员会发现 Java 语言在语法上与 C 语言极其相似。

(2) 鲁棒并且安全

Java 语言在编译及运行程序时，都要进行严格的检查。作为一种强制类型语言，Java 在编译和连接时都进行大量的类型检查，防止不匹配问题的发生。如果引用一个非法类型或执行一个非法类型操作，Java 将在解释时指出该错误。在 Java 程序中不能采用地址计算的方法通过指针访问内存单元，大大减少了错误发生的可能性；而且 Java 的数组并非用指针实现，这样就可以在检查中避免数组越界的发生。无用内存自动回收机制也增加了 Java 的鲁棒性（鲁棒性即健全性）。作为网络语言，Java 必须提供足够的安全保障，并且要防止病毒的侵袭。Java 在运行应用程序时，严格检查其访问数据的权限，比如不允许网络上的应用程序修改本地的数据。下载到用户计算机中的字节代码在其被执行前要经过一个核实工具，一旦字节代码被核实，便由 Java 解释器来执行，该解释器通过阻止对内存的直接访问来进一步提高 Java 的安全性。同时 Java 极高的鲁棒性也增强了 Java 的安全性。

(3) 结构中立并且可以移植

网络上充满了各种不同类型的机器和操作系统，为使 Java 程序能在网络的任何地方运行，Java 编译器编译生成了与体系结构无关的字节码结构文件。任何种类的计算机，只要在其处理器和操作系统上有 Java 运行环境，字节码文件就可以在该计算机上运行。即使是在单一系统的计算机上，结构中立也有非常大的作用。随着处理器结构的不断发展变化，程序员不得不编写各种版本的程序在不同的处理器上运行，这使得用其他语言开发出能够在所有平台上工作的软件集合是不可能的，而使用 Java 将使同一版本的应用程序可以运行在不同的平台上。

体系结构的中立也使得 Java 系统具有可移植性。Java 运行系统可以移植到不同的处理器和操作系统上，Java 的编译器是由 Java 语言实现的，解释器是由 Java 语言 and 标准 C 语言实现的，因此可以较为方便地进行移植工作。

(4) 高性能

虽然 Java 是解释执行的，但它仍然具有非常高的性能，在一些特定的 CPU 上，Java 字节码可以快速地转换成为机器码进行执行。而且 Java 字节码格式的设计就是针对机器码的转换，实际转换时相当简便，自动的寄存器分配与编译器对字节码的一些优化可使之生成高质量的代码。随着 Java 虚拟机的改进和“即时编译”（just in time）技术的出现使得 Java 的执行速度有了更大的提高。

(5) 解释执行、多线程并且是动态的

为易于实现跨平台，Java 被设计成为解释执行型的语言，字节码本身包含了许多编译时生成的信息，使连接过程更加简单。而多线程使应用程序可以同时进行不同的操作，处理不同的事件。在多线程机制中，不同的线程处理不同的任务，互不干涉，不会由于某一任务处于等待状态而影响了其他任务的执行，这样就可以容易的实现网络上的实时交互操作。Java 在执行过程中，可以动态的加载各种类库，这一特点使之非常适合于网络运行，同时也非常有利于软件的开发，即使是更新类库也不必重新编译使用这一类库的应用程序。

1.4 Java 的运行环境

(1) Java 开发工具 J2SDK

Java 不仅提供了一个丰富的语言和运行环境,而且还提供了一个免费的 Java 软件开发工具集 (Java Development Kits, JDK)。到目前为止, Sun 公司先后发布了 JDK1.0, JDK1.2, JDK1.3, JDK1.4, JDK1.5 等多个主要的 JDK 版本,其最新版本为 JDK1.6。通常所说的 J2SDK,是指与 Java 2 语言相对应的 Java SDK (Java Software Development Kits),它是对较新版本中 JDK 的一个特定称呼。J2SDK (JDK) 包括 Java 的编译器、解释器、调试器等开发工具以及 Java API 类库。编程人员和最终用户可以利用这些工具来开发 Java 程序。其调试工具主要包括 Java 语言编译器 Javac, 用于编译 Java 源程序; Java 字节码解释器 Java, 用于解释运行 Java 程序, 显示程序运行结果; 小应用程序浏览工具 Appletviewer, 用于测试并运行 Java 小程序。

JDK 中主要包括: JRE (Java Run Time Environment), 即 Java 运行时环境; JVM (Java Virtual Machine, Java 虚拟机), 主要作用是进行 Java 程序的运行和维护; Java API (应用程序编程接口), 主要为编程人员提供已经写好的功能, 便于快速开发; Java 编译器 (Javac.exe)、Java 运行时解释器 (Java.exe)、Java 文档化工具 (Javadoc.exe) 及其他工具和资源。

JRE 的三项主要功能: 加载代码, 由类加载器 (Class Loader) 完成; 校验代码, 由字节码校验器 (Bytecode Verifier) 完成; 执行代码, 由运行时解释器 (Runtime Interpreter) 完成。

(2) 安装和设置环境变量

为了建立起 Java 的运行环境,可以到 Oracle 公司的网站(<http://www.oracle.com/us/sun/index.html>)上下载最新的 JDK。建议同时下载其 Java Documentation, 这是 Java 的帮助文档。

以 JDK1.6 版的 J2SDK 为例 (即 JDK6), 运行下载的 “jdk6u20-windows-i586.exe” 文件, 进行 Java 环境的正式安装。默认安装于 C:\Program Files\Java\jdk1.6.0 目录下, 用户可以更改这个默认安装目录, 建议安装路径不要有空格。

接下来需要设置运行环境参数, 以 Windows 系统为例, 主要是设置 path 和 classpath 路径, 以便能够在 Windows 的任何目录下面都能编译和运行 Java 程序。

1) Path: 用于指定操作系统的可执行指令的路径, 即要告诉操作系统, Java 编译器和运行器在什么地方可以找到并运行 Java 程序的工具。

2) classpath: Java 虚拟机在运行某个类时会按 classpath 指定的目录顺序去查找这个类, 对于 Windows 2000 或 Windows XP 以上版本操作系统的用户, 使用鼠标右击【我的电脑】, 依次选择【属性】→【高级】→【环境变量】, 打开【环境变量】对话框, 在【系统变量】列表框中, 单击【新建】按钮, 新建环境变量 classpath, 其变量值为 “.; C:\Program Files\Java\jdk1.6.0\lib”; 选择 path 变量, 单击【编辑】按钮, 在 path 变量的变量值后面加上 “;C:\Program Files\Java\jdk1.6.0\bin”。

默认安装会在 C:\Program Files\Java\jdk1.6.0 目录下产生如下内容:

bin 目录: 存放 Java 的编译器、解释器等工具 (可执行文件)。

db 目录: JDK6 附带的一个轻量级的数据库, 名字叫做 Derby。

demo 目录: 存放演示程序。

include 目录：存放用于本地方法的文件。

jre 目录：存放 Java 运行环境文件。

lib 目录：存放 Java 的类库文件。

sample 目录：存放一些范例程序。

src.zip 文件：JDK 提供的类的源代码。

选择【开始】→【运行】命令，在弹出的【运行】对话框中的【打开】下拉列表框中输入 cmd，接着单击【确定】按钮切换到 DOS 状态，直接输入 javac 按 Enter 键，如果能出现如图 1-3 所示的效果（英文版也行），说明配置成功，否则需要重新进行配置。



图 1-3 path 和 classpath 路径配置正确的验证

(3) Java 的编辑、编译和运行

Java 源程序是一种文本文件，可以使用任何的文本编辑器编写，只是要注意存储时的文件名后缀必须是.java。建议读者把所有源程序文件都保存在一个指定的目录下，便于调试和运行。

这里向大家推荐两种编辑器：一是 Windows 的记事本，二是文本编辑工具 UltraEdit 或者 EditPlus。使用 Windows 记事本编辑 Java 源程序文件，存储时先选择*.（所有文件）的文件类型，然后，输入带有.java 后缀的文件名；或者直接以带英文双引号“XXXX.java”的形式输入文件名。UltraEdit 和 EditPlus 是两个非常易用且功能强大的文本编辑工具。编辑时，它们自动地把关键字、常量、变量等不同元素用不同的颜色区分开来，从而有助于减少语法错误。还有其他的 Java 开发工具可供选择，如 JCreator、JBuilder、Eclipse、NetBean 等。

Java 是解释型语言。Java 源程序必须先由 Java 编译器进行编译,生成字节码文件(也称类文件),然后在 Java 解释器的支持下解释运行。Java 编译器是 `javac.exe`,其用法如下:

```
javac filename.java
```

其中 `filename.java` 是 Java 源程序文件的文件名。如果编译器没有返回任何错误信息,则表示编译成功,并在同一目录下生成与类名相同的字节码文件 `filename.class`。如果编译出错,则需查找错误原因,进一步修改源程序,并重新编译。编译成功只能说明程序的语法正确。

Java 源程序经过 `javac` 编译后的字节码文件需要通过解释器 `java.exe` 来执行,其用法如下:

```
java filename
```

其中 `filename` 是编译生成的 Java 字节码文件的文件名,注意不要带后缀名 `.class`。

1.5 最简单的 Java 程序

Java 有两类程序,即 Java 应用程序 (Java application) 和 Java 小程序 (Java applet),前者是在命令行中运行的独立的应用程序,它类似于以往用其他高级语言开发的程序;后者需要嵌入网页在浏览器中执行。

(1) 第一个 Java 应用程序的编写和调试

首先介绍 Java 应用程序的一个简单示例,以此来学习 Java 应用程序的开发流程,并了解程序的基本结构。

【例 1-1】 编写一个应用程序,在屏幕上显示字符串 “Hello, World!”。

第一步,编写源程序。使用记事本编写程序的源代码。将源代码保存为文件 `HelloWorld.java`,并存放在一个指定的目录例如 `C:\test` 中。注意:输入文件名时必须区分大小写。为了叙述方便,我们特地为本程序的源代码行增加上编号(值得注意的是真正的程序是没有行号的)。

```
1  /* HelloWorld.java */
2  public class HelloWorld {
3      public static void main(String args[]) {
4          System.out.println("Hello, World!");
5      }
6  }
```

第二步,编译源程序。在 Win2000 或以上版本中,单击【开始】→【运行】命令,在命令行上输入 “`cmd`” 并按回车键,打开一个 MS-DOS 命令窗口;将当前目录转换到 Java 源程序所在的目录 `C:\test`;输入 “`javac filename.java`” 形式的命令进行程序编译。本例应输入 “`javac HelloWorld.java`”。如果编译正确,就显示界面如图 1-4 所示。

第三步,执行程序。在同样的命令窗口中输入 “`java filename`” 形式的命令执行程序,本例应输入 “`java HelloWorld`”,运行结果如图 1-5 所示。

至此,完成了这个简单程序的开发。查看 `C:\test` 目录,其中应该有两个文件,分别是 `HelloWorld.java` 和 `HelloWorld.class`。

通过本程序,可以看出 Java 应用程序基本框架如下:

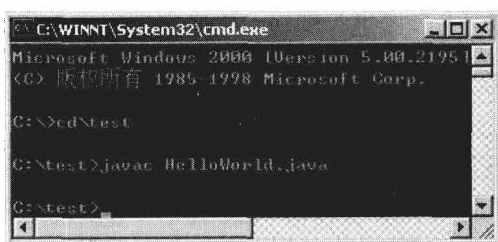


图 1-4 HelloWorld 程序编译结果

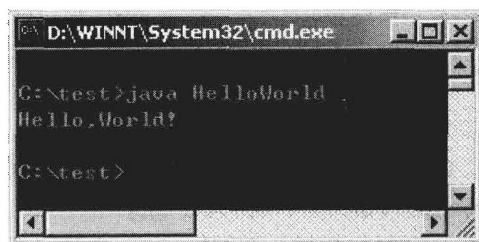


图 1-5 HelloWorld 程序运行结果

```
public class 类名{  
    public static void main ( String args [ ] ){  
        .....           //程序代码  
    }  
    //其他属性和方法  
}
```

第 1 行是注释行。Java 语言主要有三种注释：① “/* 注释内容*/ ” 的格式，可以注释一行或多行文本；② “// 注释内容” 的格式，可以注释一行文本；③ “/** 注释内容*/ ” 的格式，可以注释一行或多行文本，并可用于生成专门的 Javadoc。注释可以放在一行的开头或某个语句之后，为程序增加必要的解释，提高程序的可读性。

第 2 行是类的定义。使用关键字 `class` 定义了一个 `HelloWorld` 类，`class` 前面的 `public` 关键字表示这个类的访问特性是公共的。

Java 语言中的基本程序单位是类，一个程序文件可以定义多个类，但仅允许有一个公共的类。源程序的文件名要与公共类的名称相同（包括大小写），其扩展名为 `.java`。因此，`HelloWorld` 程序的源程序文件名必须是 `HelloWorld.java`。

第 2 行最后到第 6 行的一对花括号，其内部是类体。类体中可以声明类的变量（属性）和类的方法（函数），它们是类的成员。本例中没有声明类的成员变量。其中 `public class HelloWorld` 叫做类头。

第 3 行是类的成员方法的声明，这是一个主方法 `main`。Java 应用程序必须含有一个主方法。`public` 关键字表示这个方法是公共的，可以从其他类中访问；`static` 关键字表示这个方法是静态的，指出这个方法是针对这个类而不是针对类生成的对象；`void` 关键字表示这个方法没有返回值。

一个类可以声明多个方法，但最多只能有一个主方法 `main`。应用程序从 `main` 方法获得入口点开始运行，并通过主方法调用类中的其他方法。

`main` 方法后的小括号中是方法的参数列表，它们是方法内的局部变量，接收从外部向 `main` 方法中传递的参数。它通常写成 `String args[]`，表明所接收的参数是一个名为 `args` 的字符串数组。

第 3 行最后到第 5 行的一对花括号，其内部是方法体。在方法体内部，可以声明方法的局部变量及书写执行语句，实现数据处理功能。

第 4 行是 `main` 方法唯一的一个语句，其作用是在标准输出设备（屏幕）上输出一行字符“Hello, World!”。这是一个字符串，必须用引号括起来。最后的分号是必须的，表明这是一条 Java 语句。

为了实现屏幕输出, 这里使用了系统包 `java.lang` 中的 `System` 类, 该类中有静态成员变量 `out`, 它是一个标准输出流, 主要用于输出用户信息, 通过 `out` 调用标准输出流类中的 `println` 方法。 `println` 方法将作为参数的字符串输出到屏幕并换行。与此相似的方法是 `print`, 不同之处是 `print` 输出内容后不换行。

Java 语言系统以包的形式提供了许多的标准类库, 这些类库是编制 Java 程序的基础。调用类库中的方法之前, 先要使用 `import` 语句导入相应的类库(包), 而系统包 `java.lang` 是 Java 最基本的类库, 由系统自动导入, 因此, Hello World 程序没有使用 `import` 语句。

(2) 第一个 Java applet 程序编写和调试

Java 程序的另一种形式是 Java applet, applet 没有 `main()` 方法, 它必须嵌在超文本文件中, 在浏览器中运行。下面这个程序将在浏览器中显示一行字符串。

```
//这是我们的第一个 Java applet, 该程序保存的文件名是 HelloJava.java
import java.awt.Graphics;      //在进行显示输出时, 需要用到类 Graphics 的对象;
import java.applet.Applet;    //Applet 类是所有的 java applet 的父类;
public class HelloJava extends Applet {
                                //程序员给这个类取名为 HelloJava
                                //所有的 applet 程序都是 Applet 类的子类

    public String s;
    public void init() {
        s=new String("Welcome to Java Java"); //生成一个字符串对象
    }
    public void paint(Graphics g){
        g.drawString(s,25,25); //在浏览器中坐标为(25, 25)的位置显示字符串 s
    }
}
```

applet 程序是从方法 `init()` 开始执行的, 在该方法中完成了对字符串 `s` 的初始化工作, 而显示功能是在方法 `paint()` 中执行的。 `paint()` 方法是类 `Applet` 的一个成员方法, 其参数是图形对象 `Graphics g`, 通过调用对象 `g` 的 `drawString()` 方法就可以显示输出。

Java applet 程序也是一个类, 其编译方式与 Java application 完全一样, `HelloJava.java` 程序经过编译以后就生成了 `HelloJava.class` 文件。Java applet 的执行方式与 Java application 完全不同, Java applet 程序必须嵌入到 `html` 文件中才能够执行, 因此必须编写相应的 `html` 文件。下面为 `HelloJava.html` 文件的内容:

```
<html>
  <applet code=HelloJava.class width=250 height=250>
  </applet>
</html>
```

然后可以通过 JDK 所提供的命令 “`appletviewer`”, 在命令行下面执行 Java applet 程序。如果是在 Windows 操作系统中, 就可以在“命令提示符”下敲入“`appletviewer HelloJava.html`”。此时系统会弹出另外一个窗口运行该 applet 程序。

applet 还可以采用另外一种方式运行, 即直接在浏览器中打开 `HelloJava.html` 程序。在主流的浏览器如 IE、Netscape 中都包含有 Java 虚拟机, 负责解释执行 Java applet 程序。

1.6 Java 的应用范围

(1) 早期用于 Web 的 Applet

applet 是嵌入到 HTML 中的小应用程序, 但 Java 语言的全部功能都可以实现, 能解决一些传统编程语言很难解决的问题, 如多线程、网络连接和分布式计算等。

(2) 独立运行的 Application

Java 也是一种高级编程语言, 和其他语言有许多共同之处。

(3) 数字设备领域

目前, Java 的应用正从传统的计算机应用向其他数字设备领域扩展。

(4) Java 应用的发展趋势

可以预见, Java 在智能 Web 服务、移动电子商务、分布计算技术、企业的综合信息化处理、嵌入式 Java 技术等方面必将得到越来越广泛的应用。

小 结

Java 语言是当今流行的网络编程语言, 特别适合于开发网络上的应用程序, 具有面向对象、简单、平台无关、多线程等优秀特性。

Java 程序主要包括 Java 应用程序 (Java application) 和 Java 小程序 (Java applet) 两种。Java 应用程序的开发必须经过编写、编译、运行三个步骤: 使用记事本等文本编辑工具进行程序代码的编写, 使用 Java 开发工具集 JDK 提供的编译器进行编译, 最后使用 Java 解释器解释运行。Java 虚拟机使 Java 应用程序实现了跨平台运行。

限于篇幅, Java 代码的编写规范虽然不在这里特别讲述, 但是这是一个初学者必须重视的问题。从一开始就按照规范来写代码, 对于形成良好的编程习惯非常重要。

习 题

(一) 填空题

- (1) Java 编译器将把 Java 语言编写的源程序编写成_____。
- (2) Java 程序基本上可以分成两类, 即_____和_____。
- (3) Java 程序能在 WWW 浏览器上运行的是_____程序。
- (4) Java 源程序文件和字节码文件的扩展名分别为_____和_____。
- (5) Java 源程序的运行, 至少要经过_____和_____两个阶段。
- (6) 与结构化编程语言不同, Java 是一种完全_____且平台_____的语言。

(二) 选择题

- (1) 注释风格在 Java 中是错误的是 ()。
A. `/**comment**/` B. `/*comment*/` C. `/*comment` D. `//comment`
- (2) Java 是一种 () 语言。
A. 编译型 B. 解释型 C. 混合型 D. 以上都不是