



工业和信息化普通高等教育“十二五”规划教材立项项目  
21 世纪高等学校计算机规划教材  
21st Century University Planned Textbooks of Computer Science

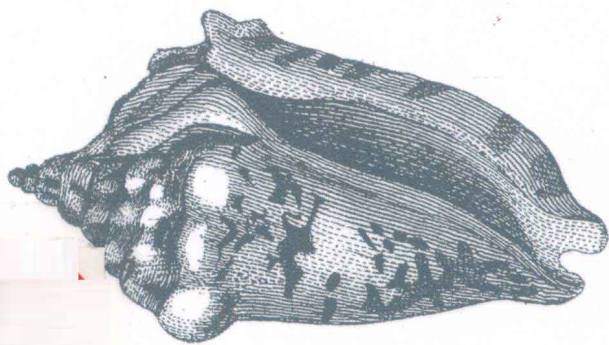
# Java实用编程技术

## Java Programming

王诚 主编

张祖昶 梅霆 汪胡青 编著

- 本书针对Java编程初中级读者，语言力求简单易懂
- 编排内容注重实用性和适用性，强调提高实践技能
- 各章内容相对独立，实例丰富可运行，可按需选学



高校系列



人民邮电出版社  
POSTS & TELECOM PRESS



工业和信息化普通高等教育“十二五”规划教材立项项目

计算机规划教材

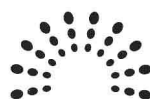
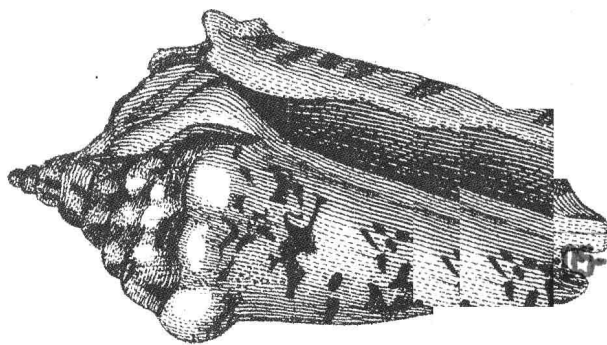
Books of Computer Science

# Java实用编程 技术

Java Programming

王诚 主编

张祖昶 梅霆 汪胡青 编著



高校系列

人民邮电出版社

北京

## 图书在版编目 (CIP) 数据

Java实用编程技术 / 王诚主编. — 北京: 人民邮电出版社, 2012.3  
21世纪高等学校计算机规划教材. 高校系列  
ISBN 978-7-115-27303-1

I. ①J… II. ①王… 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(2012)第003944号

## 内 容 提 要

本书在介绍 Java 编程的相关知识基础上, 以许多实例演示了 Java 编程技术的内容, 特别是 Java 线程、网络编程以及数据库等编程技术。由浅入深, 阐述详尽。

本书在选择编排内容时注重实用性, 强调实践技能的提高, 同时也注重系统性、科学性和学习性。本书重点突出, 层次分明, 理论联系实际, 用典型的案例来演示 Java 编程技术的魅力, 有利于培养工程应用及编程能力, 可以作为高等学校计算机软件及应用、电子信息工程、通信工程、软件工程等相关专业的本科生教材, 也可以供广大 Java 软件工程师学习参考。

工业和信息化部普通高等教育“十二五”规划教材立项项目

21 世纪高等学校计算机规划教材

### Java 实用编程技术

- 
- ◆ 主 编 王 诚  
编 著 张祖昶 梅 霆 汪胡青  
责任编辑 蒋 亮
  - ◆ 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 电子邮件 315@ptpress.com.cn  
网址 <http://www.ptpress.com.cn>  
三河市海波印务有限公司印刷
  - ◆ 开本: 787×1092 1/16  
印张: 14.25 2012 年 3 月第 1 版  
字数: 375 千字 2012 年 3 月河北第 1 次印刷

ISBN 978-7-115-27303-1

定价: 29.80 元

读者服务热线: (010)67170985 印装质量热线: (010)67129223  
反盗版热线: (010)67171154

# 前 言

当前, Internet 正以惊人的速度发展, 计算机技术、软件技术、物联网技术、网络技术、通信技术等 IT 领域新技术不断涌现, 使现在的生活出现了革命性的变化, 特别是物联网技术的兴起, 推动了各行业和部门的智能化、智慧化应用研究。在这些信息技术的应用中, Java 技术凭借其独有的、与平台无关的、与网络紧密结合的特点, 在企业级应用及移动终端的应用编程中发挥越来越重要的作用。

本书主要针对 Java 编程的初中级读者, 特别适用于高校本科生的教学。从最基础的数据类型、表达式、语句讲起, 引导读者进入面向对象的编程环节中; 中间贯穿了线程技术、网络技术、数据库技术等方面的编程知识; 最后结合一个具体的实例来进行应用编程的总结。本书在编写过程中, 力求语言简单易懂, 本着适应性原则, 尽量缩减篇幅, 降低成本。

本书共分为 11 章。第 1 章对 Java 进行简要介绍并讲解 Java 语言的编程基础; 第 2 章重点讲述 Java 面向对象编程技术及相关概念; 第 3 章介绍 Java 异常处理; 第 4 章讲解一个具体实用的 String 类及 StringBuffer 类; 第 5 章介绍输入/输出内容; 第 6 章、第 7 章、第 8 章分别介绍线程编程技术、网络编程技术、数据库编程技术, 这几章内容涉及的编程知识多且面广, 是要重点掌握的; 第 9 章介绍了 Java 图形用户界面, 本章实用性不强, 但可以给读者提供一个整体的界面编程的知识; 第 10 章介绍了 JSP 方面的基础知识, 引导读者继续进行学习与实践, 因为 Java 的优势在于网络应用, 而不是简单的桌面型应用程序, 对于后续的基于网络应用编程的学习, JSP 的内容非常实用。

本书并不要求读者逐章阅读, 每一章内容都是独立的, 可以根据应用需要选择学习。

本书中的实例都是可以运行的, 建议读者在学习过程中, 录入书中的例子以加强学习实践, 以便更好更快地掌握。

本书的先修课程有计算机编程基础、网络技术、数据库技术。如果读者学习过并掌握 C 语言的使用, 则学习本书可能会轻松一些, 因为语法结构是相似的; 如果读者具备面向对象编程的基础, 则更容易掌握 Java 编程技术。

本书适合作为高等院校计算机、信息工程、通信工程等专业课程的教材, 也可以作为软件编程人员的技术参考书。

本书在编写过程中得到了永创教育公司的支持, 也感谢南京邮电大学的刘娇同学在排版、整理资料、校对等方面做的大量工作, 在此一并表示感谢!

由于时间仓促和编者的知识所限, 书中难免存在错误与不足, 欢迎读者批评指正。作者联系方式: wangc@njupt.edu.cn。

编 著 者

2011 年 11 月

## 第 1 章 Java 语言概述

### 及编程基础 .....1

|                             |    |
|-----------------------------|----|
| 1.1 Java 语言概述 .....         | 1  |
| 1.1.1 Java 语言发展 .....       | 1  |
| 1.1.2 Java 语言的特点 .....      | 2  |
| 1.1.3 Java 平台 .....         | 3  |
| 1.2 Java 语言开发环境 .....       | 4  |
| 1.2.1 Jdk 下载及安装 .....       | 4  |
| 1.2.2 Java 开发环境设置 .....     | 6  |
| 1.2.3 Java 开发过程 .....       | 7  |
| 1.2.4 Eclipse 开发工具及使用 ..... | 8  |
| 1.3 Java 语言编程基础 .....       | 10 |
| 1.3.1 Java 标识符 .....        | 10 |
| 1.3.2 Java 关键字 .....        | 11 |
| 1.3.3 Java 基本数据类型 .....     | 11 |
| 1.3.4 运算符 .....             | 12 |
| 1.3.5 程序控制语句 .....          | 15 |
| 思考与习题 .....                 | 18 |

## 第 2 章 Java 面向对象技术基础 .....19

|                        |    |
|------------------------|----|
| 2.1 类及对象 .....         | 19 |
| 2.1.1 面向对象基本概念 .....   | 19 |
| 2.1.2 类的定义 .....       | 20 |
| 2.1.3 对象的创建和使用 .....   | 21 |
| 2.1.4 成员方法及其修饰符 .....  | 22 |
| 2.1.5 构造方法 .....       | 23 |
| 2.1.6 方法重载 .....       | 24 |
| 2.1.7 成员变量 .....       | 26 |
| 2.2 继承与多态性 .....       | 26 |
| 2.2.1 继承 .....         | 26 |
| 2.2.2 多态与方法重写 .....    | 29 |
| 2.2.3 this 关键字 .....   | 30 |
| 2.2.4 super 关键字 .....  | 31 |
| 2.2.5 static 关键字 ..... | 32 |

|                     |    |
|---------------------|----|
| 2.2.6 final 类 ..... | 33 |
|---------------------|----|

|                             |    |
|-----------------------------|----|
| 2.2.7 abstract 类和抽象方法 ..... | 34 |
|-----------------------------|----|

|             |    |
|-------------|----|
| 2.3 包 ..... | 35 |
|-------------|----|

|                  |    |
|------------------|----|
| 2.3.1 包的概念 ..... | 35 |
|------------------|----|

|                 |    |
|-----------------|----|
| 2.3.2 引入包 ..... | 36 |
|-----------------|----|

|                  |    |
|------------------|----|
| 2.3.3 访问保护 ..... | 36 |
|------------------|----|

|                  |    |
|------------------|----|
| 2.3.4 包的编译 ..... | 37 |
|------------------|----|

|              |    |
|--------------|----|
| 2.4 接口 ..... | 37 |
|--------------|----|

|                  |    |
|------------------|----|
| 2.4.1 接口定义 ..... | 37 |
|------------------|----|

|                  |    |
|------------------|----|
| 2.4.2 扩展接口 ..... | 39 |
|------------------|----|

|                  |    |
|------------------|----|
| 2.5 UML 初步 ..... | 40 |
|------------------|----|

|             |    |
|-------------|----|
| 思考与习题 ..... | 42 |
|-------------|----|

## 第 3 章 Java 异常处理 .....43

|                |    |
|----------------|----|
| 3.1 异常概述 ..... | 43 |
|----------------|----|

|                |    |
|----------------|----|
| 3.2 异常处理 ..... | 45 |
|----------------|----|

|                   |    |
|-------------------|----|
| 3.2.1 异常处理类 ..... | 45 |
|-------------------|----|

|                  |    |
|------------------|----|
| 3.2.2 异常捕获 ..... | 45 |
|------------------|----|

|                     |    |
|---------------------|----|
| 3.2.3 声明及抛出异常 ..... | 50 |
|---------------------|----|

|                     |    |
|---------------------|----|
| 3.3 创建用户自定义异常 ..... | 52 |
|---------------------|----|

|             |    |
|-------------|----|
| 思考与习题 ..... | 53 |
|-------------|----|

## 第 4 章 String 类及 StringBuffer 类 .....54

|                    |    |
|--------------------|----|
| 4.1 String 类 ..... | 54 |
|--------------------|----|

|                    |    |
|--------------------|----|
| 4.1.1 字符串初始化 ..... | 54 |
|--------------------|----|

|                 |    |
|-----------------|----|
| 4.1.2 取字符 ..... | 56 |
|-----------------|----|

|                  |    |
|------------------|----|
| 4.1.3 比较操作 ..... | 57 |
|------------------|----|

|                    |    |
|--------------------|----|
| 4.1.4 字符串的连接 ..... | 58 |
|--------------------|----|

|                   |    |
|-------------------|----|
| 4.1.5 字符串转换 ..... | 59 |
|-------------------|----|

|                    |    |
|--------------------|----|
| 4.1.6 字符串的定位 ..... | 60 |
|--------------------|----|

|                    |    |
|--------------------|----|
| 4.1.7 字符串的修改 ..... | 61 |
|--------------------|----|

|                          |    |
|--------------------------|----|
| 4.2 StringBuffer 类 ..... | 62 |
|--------------------------|----|

|                              |    |
|------------------------------|----|
| 4.2.1 StringBuffer 类介绍 ..... | 62 |
|------------------------------|----|

|                             |    |
|-----------------------------|----|
| 4.2.2 StringBuffer 方法 ..... | 62 |
|-----------------------------|----|

|                        |     |                          |     |
|------------------------|-----|--------------------------|-----|
| 思考与习题                  | 65  | 思考与习题                    | 110 |
| <b>第 5 章 输入/输出</b>     | 66  | <b>第 8 章 Java 数据库编程</b>  | 111 |
| 5.1 输入/输出基础            | 66  | 8.1 JDBC                 | 111 |
| 5.1.1 流的概念             | 66  | 8.1.1 JDBC 概念            | 111 |
| 5.1.2 控制台输入/输出         | 68  | 8.1.2 系统编程模型             | 111 |
| 5.1.3 文件类              | 70  | 8.1.3 JDBC 接口            | 112 |
| 5.2 字节流                | 72  | 8.1.4 JDBC 驱动            | 112 |
| 5.3 字符流                | 75  | 8.1.5 java.sql 包         | 114 |
| 思考与习题                  | 78  | 8.2 数据库编程                | 116 |
| <b>第 6 章 线程</b>        | 79  | 8.2.1 SQL 简介             | 116 |
| 6.1 线程概念               | 79  | 8.2.2 编程模型               | 117 |
| 6.1.1 Java 线程模型        | 79  | 8.2.3 图书借阅管理系统实例         | 118 |
| 6.1.2 主线程              | 81  | 思考与习题                    | 122 |
| 6.1.3 创建线程             | 82  | <b>第 9 章 Java 图形用户界面</b> | 123 |
| 6.1.4 线程优先级            | 86  | 9.1 概述                   | 123 |
| 6.2 线程同步               | 88  | 9.1.1 图形用户界面             | 123 |
| 6.2.1 什么是线程同步          | 88  | 9.1.2 组件                 | 123 |
| 6.2.2 线程通信             | 91  | 9.2 Swing 组件             | 125 |
| 6.2.3 死锁               | 95  | 9.2.1 文本组件               | 125 |
| 思考与习题                  | 96  | 9.2.2 按钮组件               | 126 |
| <b>第 7 章 Java 网络编程</b> | 97  | 9.2.3 列表组件               | 126 |
| 7.1 网络编程基础             | 97  | 9.2.4 窗口及面板组件            | 127 |
| 7.1.1 TCP              | 97  | 9.2.5 实例                 | 127 |
| 7.1.2 UDP              | 97  | 9.3 布局管理器                | 129 |
| 7.1.3 端口               | 97  | 9.3.1 布局管理器简介            | 129 |
| 7.1.4 套接字              | 97  | 9.3.2 FlowLayout 顺序布局    | 129 |
| 7.1.5 客户/服务器模式         | 98  | 9.3.3 BorderLayout 边框布局  | 130 |
| 7.1.6 Java 和网络         | 98  | 9.3.4 GridLayout 网格布局    | 130 |
| 7.1.7 InetAddress 类    | 99  | 9.3.5 应用布局管理器的实例         | 131 |
| 7.1.8 URL              | 100 | 9.4 事件                   | 133 |
| 7.2 基于 TCP 的编程         | 102 | 9.4.1 事件处理机制             | 133 |
| 7.2.1 编程模型             | 102 | 9.4.2 事件类                | 134 |
| 7.2.2 实例一              | 104 | 9.4.3 事件源                | 140 |
| 7.2.3 实例二              | 105 | 9.4.4 监听器                | 140 |
| 7.3 基于 UDP 的编程         | 108 | 9.4.5 事件处理实现             | 142 |
| 7.3.1 编程模型             | 108 | 9.4.6 适配器类               | 146 |
| 7.3.2 实例               | 109 | 9.5 界面编程实例               | 147 |
|                        |     | 思考与习题                    | 171 |

|                             |     |
|-----------------------------|-----|
| <b>第 10 章 JSP 技术</b> .....  | 172 |
| 10.1 概述 .....               | 172 |
| 10.1.1 JSP 技术概述 .....       | 172 |
| 10.1.2 JSP 开发环境 .....       | 173 |
| 10.2 JSP 语法 .....           | 174 |
| 10.2.1 JSP 元素 .....         | 175 |
| 10.2.2 JSP 指令 .....         | 175 |
| 10.3 JSP 内置对象 .....         | 176 |
| 10.3.1 page 对象 .....        | 176 |
| 10.3.2 config 对象 .....      | 176 |
| 10.3.3 request 对象 .....     | 177 |
| 10.3.4 response 对象 .....    | 177 |
| 10.3.5 out 对象 .....         | 177 |
| 10.3.6 session 对象 .....     | 178 |
| 10.3.7 application 对象 ..... | 178 |
| 10.3.8 pageContext 对象 ..... | 178 |
| 10.3.9 exception 对象 .....   | 179 |
| 10.4 JSP 编程实例 .....         | 179 |
| 思考与习题 .....                 | 184 |
| <b>第 11 章 实例开发</b> .....    | 185 |
| 11.1 试卷生成系统 .....           | 185 |
| 11.1.1 程序概述 .....           | 185 |
| 11.1.2 文件说明 .....           | 185 |
| 11.1.3 功能结构图 .....          | 185 |
| 11.1.4 主要界面 .....           | 186 |
| 11.1.5 程序代码清单 .....         | 187 |
| 11.2 基于 JSP 技术的登录注册系统 ..... | 208 |
| <b>参考文献</b> .....           | 222 |

# 第 1 章

## Java 语言概述及编程基础

### 1.1 Java 语言概述

#### 1.1.1 Java 语言发展

1991 年 4 月，Sun 公司的 James Gosling 领导的绿色计划（Green Project）开始着力发展一种分布式系统结构，使其能够在各种消费性电子产品上运行。他们研发了一种新的语言，该语言以 C 和 C++为基础，James 根据他在 Sun 公司办公室外的一棵橡树，而称其为 Oak 语言。后来发现已有一种称为 Oak 的计算机语言。当一些 Sun 公司的员工到当地一家咖啡店时，有人提议将该语言命名为 Java，从而使 Java 这个名字一直延续至今。

不过，在当时市场不成熟的情况下，他们的项目没有获得成功。市场对智能型电子装置需求的上升率并不像 Sun 公司所期盼的那样快，更糟的是 Sun 公司参加竞争的一个重要的销售合同被另一公司得去了。此时，Green 项目几乎处于被取消的境地。但很幸运的是，1993 年万维网（WWW）疯狂地流行起来，由于 Internet 的迅猛发展和 WWW 的快速增长，第一个全球信息网络浏览器 Mosaic 诞生了。此时，工业界非常急迫地需求一种适合在网络异构环境下使用的语言。James Gosling 决定改变绿色计划的发展方向，对 Oak 进行了小规模的改造。就这样给该项目重新注入了生机，1995 年，Oak 语言更名为 Java 语言。Java 的诞生标志着互联网时代的开始。

Sun 公司于 1995 年 5 月在一个重要会议上正式发布了 Java。这样的事通常不会引起广泛的注意，但是由于万维网的商业利益，Java 立即引起了商业界的极大兴趣。目前，Java 被广泛应用于创建具有动态的、交互内容的 Web 网页，开发大规模企业应用程序，增强万维网服务的功能，向消费者的设备提供应用程序。

1998 年是 Java 迅猛发展的一年。在 1998 年 12 月 4 日，Sun 发布了 Java 历史上最重要的一个版本：JDK1.2。这个版本的发布标志着 Java 已经进入 Java2 时代。这个时期也是 Java 飞速发展的时期。在这一年中 Sun 发布了 JSP/Servlet、EJB 规范以及将 Java 分成了 J2EE、J2SE 和 J2ME。

2000 年 5 月 8 日，Sun 对 JDK 1.2 进行了重大升级，推出了 JDK 1.3。2002 年 2 月 13 日，Sun 公司发布了 JDK 历史上最为成熟的 JDK 1.4 版本。2004 年 10 月，Sun 发布了期待已久的 JDK 1.5 版本，同时将 JDK 1.5 改名为 J2SE 5.0。

Java 问世以来，其技术发展非常快，在计算机、移动电话、家用电器等领域都得到了广泛的应用。



## 1.1.2 Java 语言的特点

Java 语言的前身是在 C++ 的基础上开发的，它继承了 C、C++ 语言的优点，增加了一些实用的功能，使 Java 语言更加精练。Java 摒弃了 C、C++ 语言的缺点，去掉了 C、C++ 语言中的指针运算、结构体定义、手工释放内存等容易引起错误的功能和特征，增强了安全性，也使其更容易被接受和学习。Java 是独立于平台，面向 Internet 的分布式编程语言。

Java 是一种简单的、面向对象的、分布式的、解释执行的、健壮的、安全的、结构中立的、可移植的、高效率的、多线程的和动态的语言。

### 1. 简单

Java 是一种纯面向对象的语言，它通过提供最基本的方法来完成指定的任务，只需理解一些基本的概念，就可以用它编写出适合于各种情况的应用程序。Java 略去了指针、运算符重载、多重继承等内容，并且通过实现无用信息自动回收，大大简化了程序设计者的内存管理工作。同时，Java 很小，基本的解释器以及类支持大约仅为 40kB。

### 2. 面向对象

Java 是一种纯面向对象的语言，Java 的核心是面向对象编程。事实上，所有的 Java 程序都是面向对象的，这一点与 C++ 语言不同，因为在那里可以选择是否面向对象编程。Java 程序面向对象的设计思路不同于 C 语言基于过程的程序设计思路。面向对象程序设计，具备更好地模拟现实世界环境的能力和可重用性。它将待解决的现实问题转换成一组分离的程序对象，这些对象彼此之间可以进行交互。一个对象包含了对应实体应有的信息以及访问和改变这些信息的方法，重点放在数据（即对象）和对象的接口上。通过这种设计方式，所设计出来的程序更易于改进、扩展、维护和重用。Java 语言只支持类之间的单继承，但支持接口之间的多继承，并支持类与接口间的实现机制。

### 3. 分布式

Java 是一种分布式的语言。传统的基于 C/S（客户端/服务器）架构的程序，客户端向服务器提出服务请求，服务器将程序执行结果返回，所以，服务器负荷较重。Java 采用 Java 虚拟机架构，可将许多工作直接交由终端处理，因此，数据可以被分布式处理。此外，Java 类库的运用，大大减轻了网络传输的负荷。Java 类库包含了支持 HTTP、FTP 等基于 TCP/IP 的子库。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。Java 的网络功能强大且易于使用，特别是远程方法调用使得分布式对象之间可以互相通信。

### 4. 高效解释执行

Java 是高效解释执行的语言。高级语言程序必须转换为机器语言程序才能在计算机上执行。Java 程序在编译时并不直接编译成特定的机器语言程序，而是编译成与系统无关的“字节码（bytecode）”，由 Java 虚拟机（Java Virtual Machine, JVM）来执行。JVM 使得 Java 程序可以“一次编译，随处运行”。任何系统只有安装了 Java 虚拟机后，才可以执行 Java 程序。JVM 能直接在任何机器上执行，为字节码提供运行环境。

### 5. 健壮性

Java 是健壮的语言。Java 不需要指针就可以构造其他语言中需要指针构造的数据结构，即不会存取“坏的”指针而造成内存分配、内存泄露等错误。在传统的编程环境下，内存管理是一项困难、乏味的工作。例如，在 C 或 C++ 语言中，必须手工分配、释放所有的动态内存。如果忘记释放原来分配的内存，或是释放了其他程序正在使用的内存时，系统就会出错。同时，在传统的编程环境下，对异常情况必须用既繁琐又难理解的一大堆指令来进行处理。Java 通过自行管理内

存分配和释放的方法,从根本上消除了有关内存的问题。Java 提供垃圾收集器,可自动收集闲置对象占用的内存。Java 提供面向对象的异常处理机制来解决异常处理的问题。

## 6. 安全

Java 是安全的网络编程语言,使用 Java 可以构建防病毒、防篡改的系统。Java 提供了一系列的安全机制以防恶意代码攻击,确保系统安全,如禁止运行时堆栈溢出、禁止在自己处理空间外破坏内存。Java 的安全机制分为多级,包括 Java 语言本身的安全性设计以及严格的编译检查、运行检查和网络接口级的安全检查。

## 7. 结构中立

Java 是结构中立的语言。Java 的设计目标是要支持网络应用。Java 编译器会产生一种具备结构中立性的对象文件格式,即 Java 字节码文件。精心设计的 Java 字节码不仅很容易在任何机器上解释执行,还可以快速翻成本机的代码。

## 8. 可移植性

结构中立是确保程序可移植的必要条件,此外,还需要很多其他条件的配合。Java 在可移植性方面做了许多工作。Java 通过定义独立于平台的基本数据类型及其运算,使数据得以在任何硬件平台上保持一致。例如,Java 中的 int 类型永远是 32 位的整数,这样就消除了代码移植时的主要问题。

## 9. 高效率

Java 是高效率的语言。每一次的版本更新,Java 在性能上均做出了改进。在历经数个版本变更后,Java 已经拥有与 C/C++ 同样甚至更好的运行性能。当 JVM 解释执行 Java 程序时,Java 实时编译器(Just-In-Time, JIT)会将字节码译成目标平台对应的机器语言的指令代码,并将结果进行缓存,因为 JVM 能够直接使用 JIT 编译技术将经过精心设计的字节码转换成高性能的本机代码,所示 Java 可以在非常低档的 CPU 上顺畅运行。

## 10. 多线程

Java 是支持多线程的语言。多线程可以带来更好的交换响应和实时行为。多线程是一种应用程序设计方法。线程是从大进程里分出来的、小的、独立的进程,使得在一个程序里可同时执行多个小任务。多线程带来的好处是具有更好的交互性能和实时控制性能,但采用传统的程序设计语言(如 C/C++)实现多线程非常困难。Java 实现了多线程技术,提供了一些简便地实现多线程的方法,并拥有一套高复杂性的同步机制。

## 11. 动态

Java 语言具有动态特性,能够适应不断发展的环境。Java 的动态特性是其面向对象设计方法的扩展,库中可以自由地添加新方法和实例变量,允许程序动态地装入运行过程中所需的类,Java 将符号引用信息在字节码中保存后传递给解释器,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据偏移量。存储器生成的对象不在编译过程中确定,而是延迟到运行时由解释器确定。

# 1.1.3 Java 平台

Java 推出了 3 个领域的应用平台:标准版 Java 2 Platform Standard Edition (Java SE)、企业版 Java 2 Platform Enterprise Edition (Java EE) 和微型版 Java 2 Platform Micro Edition (Java ME)。

Java SE 是各应用平台的基础。Java SE 可以分为 4 个主要部分:JVM、JRE、JDK 和 Java 语言。为了能运行 Java 程序,平台上必须安装有 Java 虚拟机(Java Virtual Machine, JVM)。JVM 包含在 Java 运行环境(Java SE Runtime Environment, JRE)中,所以,为了要运行 Java 程序,必须取得 JRE 并进行安装。如果要开发 Java 程序,则必须取得 JDK (Java SE Development Kits) 并进行安装。

JDK 包括了 JRE 以及开发过程中所需要的一些工具程序,如 Javac、Java、Appletviewer 等。Java 语言只是 Java SE 的一部分。此外,Java 提供了庞大且功能强大的 API(Application Programming Interface)类库,可以使用这些 API 作为基础进行程序开发,而无须重复开发功能相同的组件。

Java EE 以 Java SE 为基础,定义了一系列的服务、API、协议等,适用于开发分布式、多层式 (Multitiered)、以组件为基础、以 Web 为基础的应用程序。整个 Java EE 的体系是相当庞大的,比较为人熟悉的技术有 JSP、Servlet、Enterprise JavaBeans (EJB)、Java Remote Method Invocation (RMI) 等。

Java ME 是 Java 平台版本中最小的一个,是作为小型数字设备上开发及部署应用程序的平台,如消费型电子产品、嵌入式系统等。

## 1.2 Java 语言开发环境

### 1.2.1 Jdk 下载及安装

JDK 是 Java 开发工具包。JDK 中包括 Java 编译器 (javac)、打包工具 (jar)、文档生成器 (javadoc)、查错工具 (jdb),以及完整的 JRE (Java Runtime Environment, Java 运行环境)。

JDK 一般有 3 种版本,如表 1.1 所示。

表 1.1

JDK 版本

| 名 称          | 说 明                                                          |
|--------------|--------------------------------------------------------------|
| SE (Java SE) | Standard Edition, 标准版, 主要用于开发 Java 桌面应用程序                    |
| EE (Java EE) | Enterprise Edition, 企业版, 使用这种 JDK 开发 Java EE 应用程序, 用于 Web 方面 |
| ME (Java ME) | Micro Edition, 微型版, 主要用于移动设备、嵌入式设备上的 Java 应用程序               |

在开发 Java 程序之前,先要在本机上安装 Java 程序开发工具包 JDK。安装好后的 JDK 具有 bin、demo、jre、lib、src 等子目录。在 Sun 公司的网站上有免费的 JDK 可供下载。下载及安装步骤如下。

(1) 打开浏览器,在地址栏输入 Sun 公司的网址: <http://java.sun.com/javase/downloads/index.jsp>, 进入下载页面,在该页面中选择合适的 Java 版本,如图 1.1 所示。

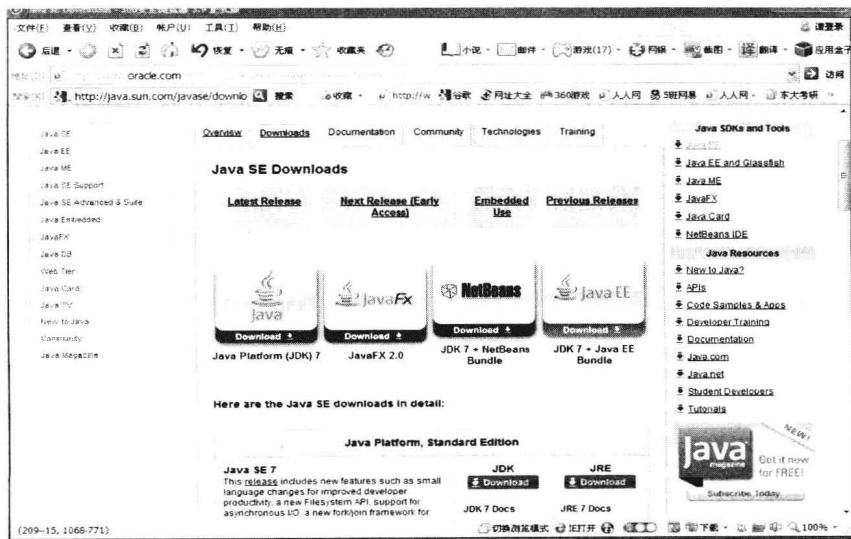


图 1.1 下载页面

(2) 在图 1.2 所示的页面中, 显示的是不同平台下的 JDK 安装包, 如 Windows、Linux、Solaris。找到适合于自己计算机平台的 JDK 版本, 这里下载 jdk-7-windows-i586.exe 可执行程序。需要注意的是, 在下载 JDK 工具包之前, 要选择接受协议。



图 1.2 选择安装包

(3) 下载好后双击图标, 弹出安装向导对话框, 在这里设置 JDK 的安装路径, 可以自行更改路径也可以使用默认路径, 如图 1.3 所示。

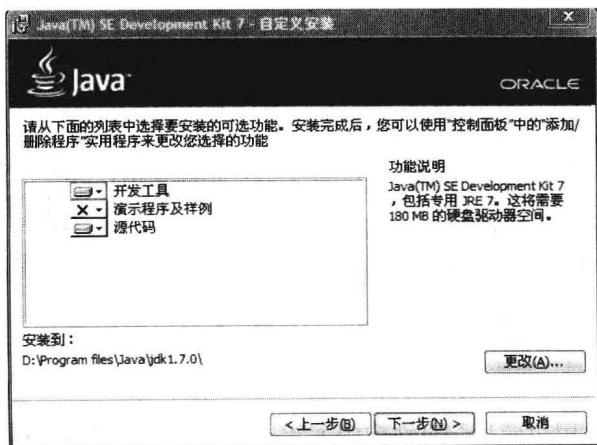


图 1.3 选择安装路径

(4) 单击“下一步”按钮, 开始安装 JDK, 如图 1.4 所示。

(5) 在安装过程中会出现安装 Java 运行环境的对话框, 选择默认设置不做改变, 单击“下一步”按钮, 继续安装, 安装成功后会出现相应的提示信息。

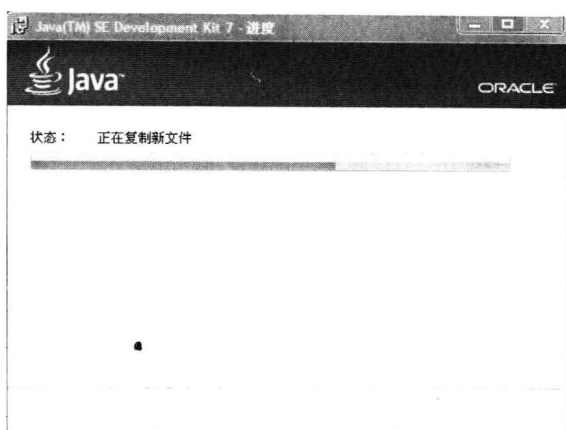


图 1.4 安装进度

## 1.2.2 Java 开发环境设置

JDK 安装后，需要设置开发的环境变量，具体操作与步骤如下。

(1) 在“我的电脑”上单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，弹出“系统属性”对话框，选择“高级”选项卡，如图 1.5 所示。

(2) 单击“环境变量”按钮，在打开的“环境变量”对话框中，单击“新建”按钮，如图 1.6 所示。

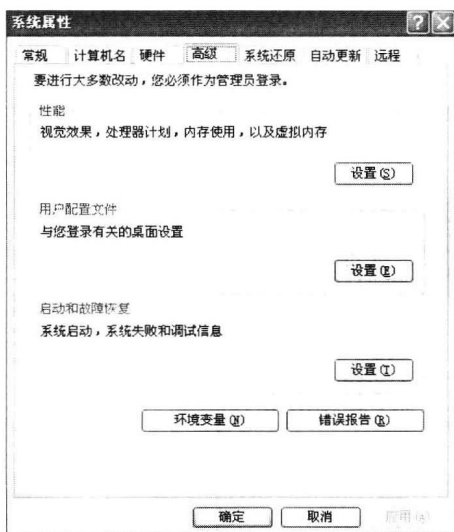


图 1.5 “系统属性”对话框

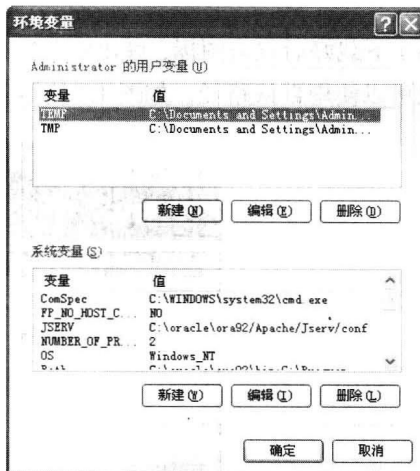


图 1.6 “环境变量”对话框

(3) 在弹出的“新建用户变量”对话框中创建一个新的系统变量名“JAVA\_HOME”，其值为 JDK 的安装路径。单击“确定”按钮保存，如图 1.7 所示。

(4) 在图 1-6 中选中“Path”变量名，单击“编辑”按钮，在弹出的“编辑系统变量”对话框中设置变量值为“D:\jdk1.7.0\bin”。单击“确定”按钮保存，如图 1.8 所示。

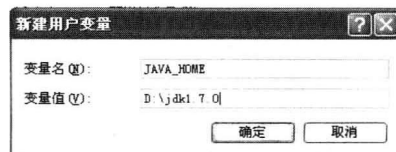


图 1.7 新建用户变量

(5) 新建“CLASSPATH”变量名, 设置变量值为“D:\jdk1.7.0\lib\tools.jar”。单击“确定”按钮保存, 如图 1.9 所示。

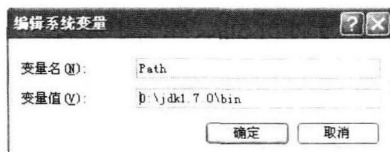


图 1.8 编辑系统变量

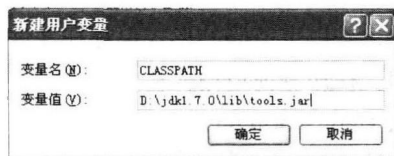


图 1.9 新建用户变量

(6) 验证变量设置的正确性。选择“开始”“→运行”命令, 在运行窗口中输入“cmd”命令, 打开命令行编辑器窗口, 在该窗口中分别输入“javac”(编译器)和“java”(解释器)命令, 并按 Enter 键运行这两条命令。如果可以看到如下的帮助信息, 则说明 JDK 的安装和设置是正确的, 如图 1.10 所示。

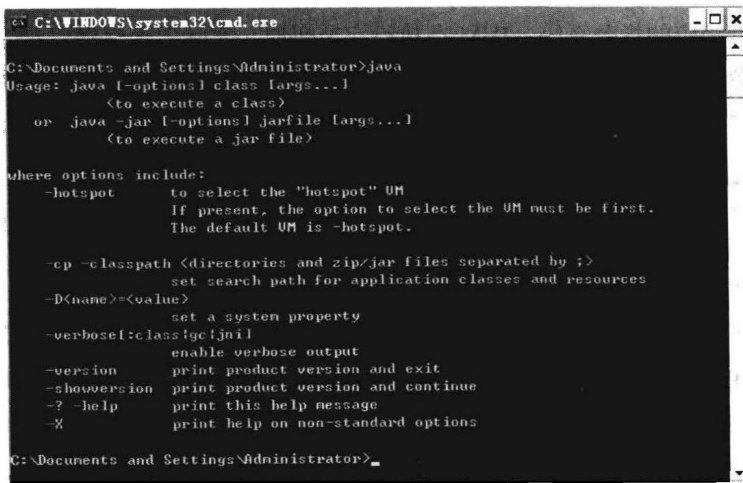


图 1.10 cmd 窗口

### 1.2.3 Java 开发过程

用 Java 语言可以编写两种程序, 一种是应用程序 (Application), 另一种是小应用程序 (Applet)。应用程序是基于桌面型的应用, 可以独立运行, 也可以用在网络、多媒体等开发上。小应用程序是一种特殊的 Java 程序, 不可以独立运行, 是嵌入到 Web 网页中且由带有 Java 插件的浏览器解释运行, 主要用在 Internet 上。

开发应用程序的步骤如下。

(1) 编辑源程序。Java 源程序一般用 Java 作为扩展名, 是一个文本文件, 用 Java 语言写成, 可以用任何文本编辑器创建与编辑。

例如, 打开记事本编辑器, 编辑如下代码:

```
//文件名: HelloWorld.java
public class HelloWorld { //类名要和文件名一致
    public static void main(String args[])
    {
        System.out.println("Hello World!");
    }
}
```

```
    }    //结束 main 方法的定义  
}  
    //结束类 HelloWorld 的定义
```

(2) 编译源程序。使用“javac”编译器,读取 Java 源程序并翻译成 Java 虚拟机能够明白的指令集合,并以字节码的形式保存在文件中。通常,字节码文件以 class 作为扩展名。

例如,在命令窗口输入:

```
javac HelloWorld.java
```

(3) 解释执行,使用“java”解释器,读取字节码,取出指令并且翻译成计算机能执行的代码,完成运行过程。字节码运行的平台是 Java 虚拟机,只要计算机上安装有 Java 虚拟机,不论采用哪种操作系统,硬件配置如何不同,运行的结果都一样。

例如,在命令窗口输入:

```
Java HelloWorld
```

JVM 是一个虚构出来的计算机,它是通过在实际的计算机上仿真模拟各种计算机功能而实现的,是可运行 Java 字节码的虚拟计算机系统。JVM 有自己完善的硬件架构,如处理器、堆栈、寄存器等,还具有相应的指令系统。JVM 在执行字节码时,实际上最终还是把字节码解释成具体平台上的机器指令执行,即在计算机和编译程序之间加入了一台抽象的虚拟的机器。这台虚拟的机器在任何平台上都提供给编译程序一个共同的接口。

## 1.2.4 Eclipse 开发工具及使用

Eclipse 是 IBM 主导开发的 IDE(Integrated Development Environment)。第 1 版 1.0 在 2001 年 11 月发行,随后逐渐受到欢迎。Eclipse 已经成为开放原始码计划(Open Source Project),Eclipse 项目由 Project Management Committee(PMC)所管理,Eclipse 项目分成 3 个子项目:平台(Platform)、开发工具箱(Java Development Toolkit(JDT))、外挂开发环境(-Plug-in Development Environment(PDE))。

可以从 eclipse.org 网站(<http://www.eclipse.org/downloads>)下载,有 4 种版本或建置(build)可供下载。

(1) 发行版(Release builds): 稳定版本,经过完整测试,并具有一致性、定义清楚的功能。

(2) 稳定版(Stable builds): 比 Release build 新一级的版本,经由 Eclipse 开发团队测试,并认定它相当稳定。

(3) 整合版(Integration builds): 具有一定的稳定度,但不保证没问题。

(4) 当日最新版(Nightly builds): 当日最新版可能有问题,或有严重问题。

在 Eclipse 中做任何事之前,都必须新增一个项目,如图 1-11 所示。Eclipse 可透过外挂支持数种项目(如 EJB 或 C/C++),预设支持下列 3 种项目:

(1) Java Project——Java 开发环境;

(2) Plug-in Project——自行开发 plug-in 的环境;

(3) Sample Project——提供操作文件的环境。

### 1. 建立 Java 项目示例

(1) 选择“File”→“New”→“Project”(或是在“Package Explorer”窗口上单击鼠标右键,选择“New”→“Project”选单选项)。

(2) 在“New Project”对话框中,选择“Java Project”(或是展开“Java”的数据夹,选择“Java Project”),如图 1.12 所示。

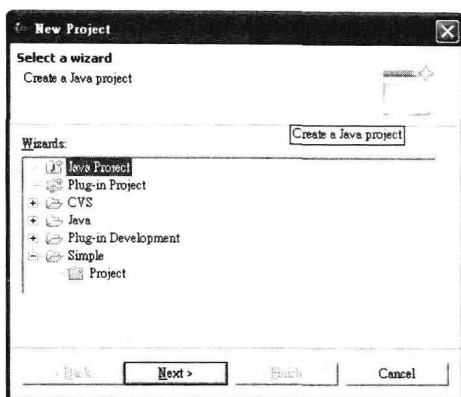


图 1.11 创建新项目

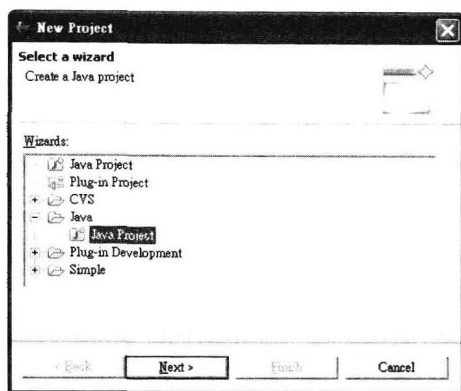


图 1.12 新建 Java 项目

(3) 在“New Java Project”对话框中，输入 Project 的名称，然后单击“Finish”按钮，如图 1.13 所示。

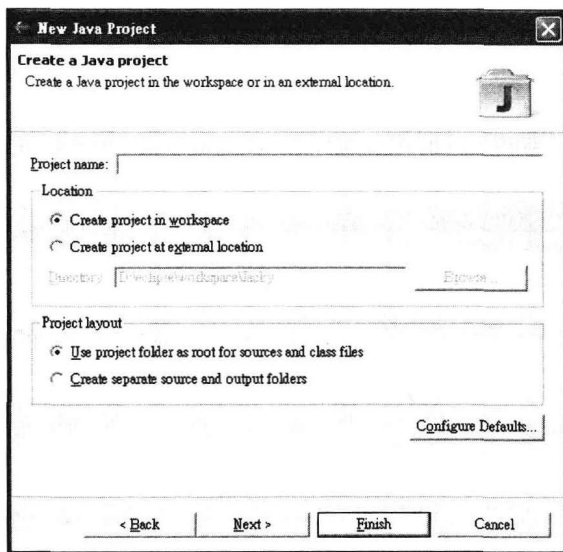


图 1.13 命名 Project name

## 2. 建立 Java 类并运行示例

(1) 选择“File”→“New”→“Class”（或是在“Package Explorer”窗口上单击鼠标右键，选择“New”→“Class”菜单选项）。

(2) 在如图 1.14 所示的“New Java Class”对话框中，“Source Folder”文本框默认值是项目的数据夹，不需要更改。

(3) 在“Package”文本框输入程序套件的名称。

(4) 在“Name”文本框输入 Class Name。

(5) 在“Which method would you like to create?”选项中，如果勾选 `public static void main(String[] args)`，则会生成主方法。

(6) 单击“Finish”按钮，会依套件新增适当的目录结构及 Java 原始文件。

(7) 在“main”方法中输入：`System.out.println(“Hello world!”)`。



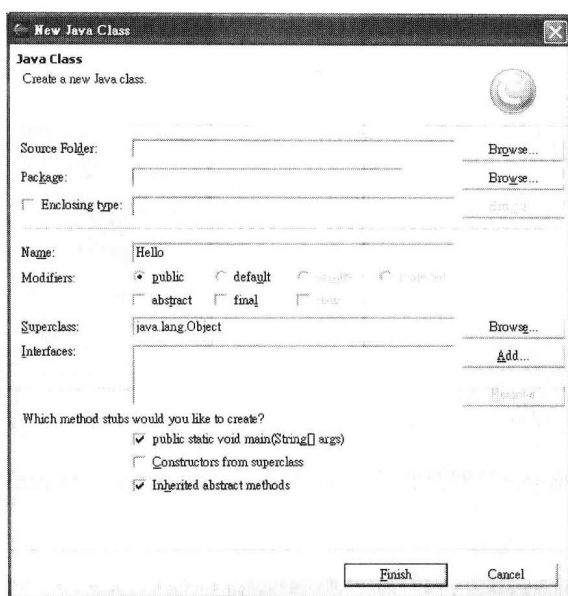


图 1.14 建立 Java 类

(8) 选择“Run”→“Run”开启Run的设定窗口，单击“Run”按钮运行，如图 1.15 所示。

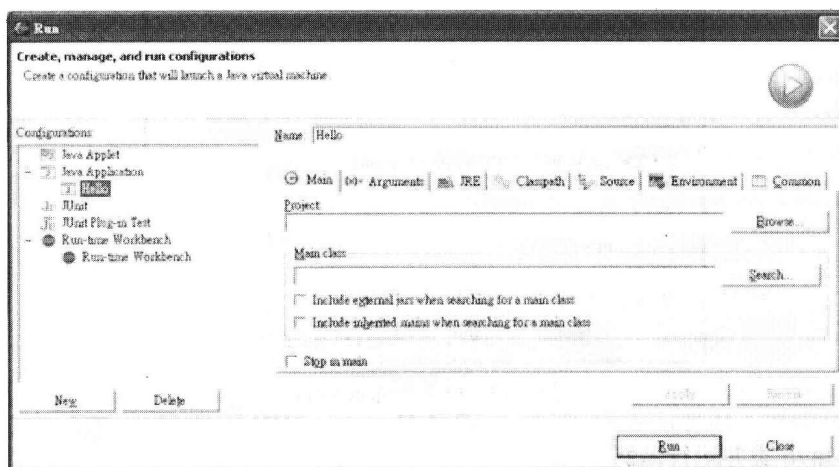


图 1.15 Run 的设定窗口

## 1.3 Java 语言编程基础

### 1.3.1 Java 标识符

在 Java 语言中，用来标识类名、对象名、变量名、方法名、类型名、数组名、文件名的有效字符序列被称为“标识符”。简单地说，标识符就是一个名字。