



高等院校规划教材  
计算机基础教育系列

# Java 面向对象程序设计

黄 斐 编著



机械工业出版社  
CHINA MACHINE PRESS

高等院校规划教材·计算机基础教育系列

# Java 面向对象程序设计

黄 斐 编著

机 械 工 业 出 版 社

Java 是一种十分流行的编程语言。本书根据 Java 程序设计的特点而编写,注重提高读者运用 Java 语言和面向对象技术解决实际问题的能力。本书先介绍 Java 基本语法和常用算法,在此基础上重点讨论了 Java 类和对象,再结合程序实例,介绍面向对象技术原理。

全书共分为 12 章,内容包括:概述、程序设计基础、程序流程控制、类的实现、实例对象、类的继承、接口、包和异常、小应用程序和绘图、线程、动画及交互技术、图形用户界面程序设计、输入/输出流控制、网络程序设计等。

本书可作为高等院校计算机及相关专业的程序设计课程的入门教材,也可作为程序设计公共基础课的教材。书中提供了大量经过调试运行的实例,便于初学者学习和上机实践。本书也适用于职业教育或从事实际软件开发的读者学习使用。

## 图书在版编目(CIP)数据

Java 面向对象程序设计/黄斐编著. —北京:机械工业出版社,2007.7  
(高等院校规划教材·计算机基础教育系列)  
ISBN 978-7-111-21834-0

I. J... II. 黄... III. Java 语言—程序设计—高等学校—教材 IV. TP312

中国版本图书馆 CIP 数据核字(2007)第 102877 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

策 划: 赵 慧

责任编辑: 祝 伟

责任印制: 杨 曦

三河市宏达印刷有限公司印刷

2007 年 7 月第 1 版·第 1 次印刷

184mm×260mm·19.75 印张·488 千字

0001—5000 册

标准书号: ISBN 978-7-111-21834-0

定价: 30.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

销售服务热线电话:(010)68326294

购书热线电话:(010)88379639 88379641 88379643

编辑热线电话:(010)88379739

封面无防伪标均为盗版

# 出版说明

计算机技术的发展极大地促进了现代科学技术的发展，明显地加快了社会发展的进程。因此，各国都非常重视计算机教育。

近年来，随着我国信息化建设的全面推进和高等教育的蓬勃发展，高等院校的计算机教育模式也在不断改革，计算机学科的课程体系和教学内容趋于更加科学和合理，计算机教材建设逐渐成熟。在“十五”期间，机械工业出版社组织出版了大量计算机教材，包括“21世纪高等院校计算机教材系列”、“21世纪重点大学规划教材”、“高等院校计算机科学与技术‘十五’规划教材”、“21世纪高等院校应用型规划教材”等，均取得了可喜成果，其中多个品种的教材被评为国家级、省部级的精品教材。

为了进一步满足计算机教育的需求，机械工业出版社策划开发了“高等院校规划教材”。这套教材是在总结我社以往计算机教材出版经验的基础上策划的，同时借鉴了其他出版社同类教材的优点，对我社已有的计算机教材资源进行整合，旨在大幅提高教材质量。我们邀请多所高校的计算机专家、教师及教务部门针对此次计算机教材建设进行了充分的研讨，达成了许多共识，并由此形成了“高等院校规划教材”的体系架构与编写原则，以保证本套教材与各高等院校的办学层次、学科设置和人才培养模式等相匹配，满足其计算机教学的需要。

本套教材包括计算机科学与技术、软件工程、网络工程、信息管理与信息系统、计算机应用技术以及计算机基础教育等系列。其中，计算机科学与技术系列、软件工程系列、网络工程系列和信息管理与信息系统系列是针对高校相应专业方向的课程设置而组织编写的，体系完整，讲解透彻，计算机应用技术系列是针对计算机应用类课程而组织编写的，着重培养学生利用计算机技术解决实际问题的能力；计算机基础教育系列是为大学公共基础课层面的计算机基础教学而设计的，采用通俗易懂的方法讲解计算机的基础理论、常用技术及应用。

本套教材的内容源自致力于教学与科研一线的骨干教师与资深专家的实践经验和研究成果，融合了先进的教学理念，涵盖了计算机领域的核心理论和最新的应用技术，真正在教材体系、内容和方法上做到了创新。另外，本套教材根据实际需要配有电子教案、实验指导或多媒体光盘等教学资源，实现了教材的“立体化”建设。本套教材将随着计算机技术的进步和计算机应用领域的扩展而及时改版，并及时吸纳新兴课程和特色课程的教材。我们将努力把这套教材打造成为国家级或省部级精品教材，为高等院校的计算机教育提供更好的服务。

对于本套教材的组织出版工作，希望计算机教育界的专家和老师能提出宝贵的意见和建议。衷心感谢计算机教育工作者和广大读者的支持与帮助！

机械工业出版社

# 前 言

Java 是一种非常流行的面向对象程序设计语言。面向对象技术提倡的模拟现实世界的思维方式、数据与操作相捆绑的程序风格符合了现代大规模软件开发的要求和潮流，而成为计算机应用开发领域的主流趋势。Java 是伴随互联网发展起来的编程语言，由于具有平台无关特性、安全机制、高可靠性和内嵌的网络支持等优点，Java 已经成为当前编写网络应用的首选工具之一。

本书以 Java 语言为基础，介绍面向对象程序设计的基本原理和方法。应用 Java 语言进行程序设计的关键是要掌握其特定的思维方式和方法。本书先介绍 Java 基本语法和常用算法，然后在此基础上重点讨论了 Java 类和对象。考虑到程序设计课程的特点，本书在讲述语法规则的同时，给出相关的程序实例。

Java 的功能非常强大，因此所涉及到的概念、规则等要比其他计算机语言复杂一些，使不少读者感到学习困难。本书是 Java 程序设计的基础教材，不仅适合初学者，同样也适合具有一定程序设计基础的读者。本书具有以下特点：

- (1) 力求文字流畅、通俗易懂，使读者循序渐进。
- (2) 对于重要概念和技术要点，配有类图、文字说明及程序实例。
- (3) 书中所有的程序实例都在 Windows XP 支持下的 JDK6 上调试通过。
- (4) 提供配套的实验指导书和教学实验网站。

本书由黄斐编著。书中不妥之处，敬请广大读者指正。

本书为读者免费提供电子教案，可到 <http://www.cmpbook.com> 网站下载。

编 者

# 目 录

出版说明

前言

|                          |    |
|--------------------------|----|
| 第 1 章 概述 .....           | 1  |
| 1.1 Java 程序设计概述 .....    | 1  |
| 1.1.1 Java 开放式平台 .....   | 1  |
| 1.1.2 Java 语言的特点 .....   | 2  |
| 1.1.3 面向对象要素 .....       | 3  |
| 1.2 Java 程序简介 .....      | 4  |
| 1.2.1 简单 Java 程序 .....   | 4  |
| 1.2.2 程序的基本要求 .....      | 5  |
| 1.2.3 Java 程序开发步骤 .....  | 6  |
| 1.3 编辑和运行 Java 程序 .....  | 7  |
| 1.3.1 Java 程序运行环境 .....  | 7  |
| 1.3.2 Java 程序编辑工具 .....  | 8  |
| 1.3.3 运行 Applet 程序 ..... | 11 |
| 习题 1 .....               | 13 |
| 第 2 章 程序设计基础 .....       | 14 |
| 2.1 基本数据类型 .....         | 14 |
| 2.2 常量和变量 .....          | 16 |
| 2.2.1 常量 .....           | 17 |
| 2.2.2 变量 .....           | 18 |
| 2.3 基本运算符和表达式 .....      | 19 |
| 2.3.1 基本运算符 .....        | 20 |
| 2.3.2 表达式 .....          | 25 |
| 2.3.3 表达式语句 .....        | 28 |
| 2.4 常用数学方法 .....         | 31 |
| 2.4.1 数学方法简介 .....       | 31 |
| 2.4.2 算术运算方法 .....       | 32 |
| 2.4.3 转换方法 .....         | 32 |
| 2.4.4 最大和最小方法 .....      | 33 |
| 2.5 基本输入输出方法 .....       | 33 |
| 2.5.1 标准输入输出 .....       | 34 |
| 2.5.2 自定义输入法 .....       | 37 |
| 2.5.3 对话框输入输出 .....      | 39 |
| 习题 2 .....               | 42 |
| 第 3 章 程序流程控制 .....       | 45 |
| 3.1 选择结构 .....           | 45 |
| 3.1.1 if 语句 .....        | 45 |

|              |             |           |
|--------------|-------------|-----------|
| 3.1.2        | if-else 语句  | 46        |
| 3.1.3        | if 语句嵌套     | 46        |
| 3.1.4        | 条件运算符       | 47        |
| 3.1.5        | switch 语句   | 48        |
| 3.2          | 循环结构        | 51        |
| 3.2.1        | while 语句    | 51        |
| 3.2.2        | do-while 语句 | 52        |
| 3.2.3        | for 语句      | 53        |
| 3.2.4        | 循环嵌套        | 56        |
| 3.3          | 跳转结构        | 56        |
| 3.3.1        | break 语句    | 56        |
| 3.3.2        | continue 语句 | 58        |
| 3.4          | 控制结构程序示例    | 59        |
| 3.5          | 数组          | 63        |
| 3.5.1        | 一维数组        | 63        |
| 3.5.2        | 多维数组        | 67        |
| 习题 3         |             | 71        |
| <b>第 4 章</b> | <b>类的实现</b> | <b>75</b> |
| 4.1          | 类的描述        | 75        |
| 4.2          | 成员变量        | 78        |
| 4.2.1        | 成员变量定义      | 78        |
| 4.2.2        | 成员变量的访问权限   | 79        |
| 4.2.3        | 静态变量        | 81        |
| 4.2.4        | final 变量    | 82        |
| 4.3          | 成员方法        | 82        |
| 4.3.1        | 定义方法        | 82        |
| 4.3.2        | 实参与形参       | 85        |
| 4.3.3        | 变量的作用域      | 87        |
| 4.3.4        | 递归调用方法      | 88        |
| 4.3.5        | 重载方法        | 89        |
| 习题 4         |             | 91        |
| <b>第 5 章</b> | <b>实例对象</b> | <b>93</b> |
| 5.1          | 对象的创建       | 93        |
| 5.1.1        | 说明对象        | 93        |
| 5.1.2        | 实例化对象       | 94        |
| 5.1.3        | 初始化对象       | 94        |
| 5.2          | 对象的使用       | 96        |
| 5.2.1        | 引用对象中的变量    | 96        |
| 5.2.2        | 调用对象的方法     | 96        |
| 5.2.3        | 释放对象        | 97        |
| 5.3          | 对象的访问       | 97        |

|                             |            |
|-----------------------------|------------|
| 5.3.1 对象的私有成员 .....         | 97         |
| 5.3.2 方法的对象参数 .....         | 99         |
| 5.3.3 方法的对象返回值 .....        | 103        |
| 5.3.4 数组元素对象 .....          | 105        |
| 5.3.5 类的实例对象 .....          | 107        |
| 5.3.6 程序设计实例 .....          | 108        |
| 5.4 使用 this .....           | 111        |
| 5.4.1 成员变量使用 this .....     | 111        |
| 5.4.2 构造方法使用 this .....     | 111        |
| 5.4.3 成员方法使用 this .....     | 112        |
| 习题 5 .....                  | 114        |
| <b>第 6 章 类的继承 .....</b>     | <b>115</b> |
| 6.1 子类的创建 .....             | 115        |
| 6.1.1 方法和变量的继承 .....        | 115        |
| 6.1.2 类成员的隐藏和重构 .....       | 116        |
| 6.1.3 方法的动态调用 .....         | 119        |
| 6.2 子类对象的使用 .....           | 120        |
| 6.2.1 子类对象与父类对象 .....       | 120        |
| 6.2.2 对象引用的类型转换 .....       | 121        |
| 6.2.3 子类对象方法的实参 .....       | 122        |
| 6.3 抽象类 .....               | 123        |
| 习题 6 .....                  | 126        |
| <b>第 7 章 接口、包和异常 .....</b>  | <b>127</b> |
| 7.1 接口 .....                | 127        |
| 7.1.1 接口定义 .....            | 127        |
| 7.1.2 接口的实现 .....           | 128        |
| 7.1.3 接口作为类型 .....          | 129        |
| 7.1.4 接口继承 .....            | 131        |
| 7.1.5 接口中的变量 .....          | 133        |
| 7.2 包 .....                 | 134        |
| 7.2.1 语句 package .....      | 135        |
| 7.2.2 包的使用 .....            | 137        |
| 7.2.3 标识符的作用域 .....         | 139        |
| 7.3 异常处理 .....              | 141        |
| 7.3.1 异常的分类 .....           | 141        |
| 7.3.2 异常的抛出 .....           | 143        |
| 7.3.3 异常处理机制 .....          | 144        |
| 习题 7 .....                  | 150        |
| <b>第 8 章 小应用程序和绘图 .....</b> | <b>151</b> |
| 8.1 Applet 原理与方法 .....      | 151        |
| 8.1.1 Applet 基本工作原理 .....   | 151        |



|                                |            |
|--------------------------------|------------|
| 8.1.2 Applet 类的主要方法 .....      | 152        |
| 8.2 Applet 与 HTML 文件配合使用 ..... | 155        |
| 8.2.1 HTML 中的 Applet 标记 .....  | 155        |
| 8.2.2 向 Applet 传递参数 .....      | 157        |
| 8.3 Applet 绘图基础 .....          | 159        |
| 8.3.1 使用 paint 方法 .....        | 159        |
| 8.3.2 绘制字符串方法 .....            | 160        |
| 8.3.3 其他绘图方法 .....             | 164        |
| 习题 8 .....                     | 169        |
| <b>第 9 章 线程、动画及交互技术 .....</b>  | <b>172</b> |
| 9.1 线程概述 .....                 | 172        |
| 9.1.1 线程状态 .....               | 172        |
| 9.1.2 线程调度 .....               | 173        |
| 9.1.3 线程同步 .....               | 175        |
| 9.2 动画制作 .....                 | 176        |
| 9.2.1 图像绘制 .....               | 176        |
| 9.2.2 图像运动 .....               | 183        |
| 9.3 鼠标事件处理 .....               | 190        |
| 9.3.1 事件接口描述 .....             | 191        |
| 9.3.2 鼠标点击画圆 .....             | 192        |
| 9.3.3 鼠标状态处理 .....             | 194        |
| 习题 9 .....                     | 197        |
| <b>第 10 章 图形用户界面程序设计 .....</b> | <b>198</b> |
| 10.1 Java 图形用户界面概述 .....       | 198        |
| 10.1.1 容器与基本组件 .....           | 198        |
| 10.1.2 Java GUI 的组成 .....      | 201        |
| 10.1.3 设计图形用户界面 .....          | 201        |
| 10.2 GUI 程序设计常用技术 .....        | 203        |
| 10.2.1 属性、方法和事件 .....          | 203        |
| 10.2.2 创建 GUI 应用程序 .....       | 204        |
| 10.2.3 事件处理机制 .....            | 205        |
| 10.3 GUI 常用组件 .....            | 211        |
| 10.3.1 标签和文本框 .....            | 211        |
| 10.3.2 命令按钮 .....              | 215        |
| 10.3.3 复选框和单选按钮组 .....         | 217        |
| 10.3.4 组合框和列表框 .....           | 222        |
| 10.4 布局管理和容器 .....             | 228        |
| 10.4.1 布局管理 .....              | 228        |
| 10.4.2 容器组件 .....              | 235        |
| 习题 10 .....                    | 249        |

|                                             |     |
|---------------------------------------------|-----|
| <b>第 11 章 输入/输出流控制</b>                      | 253 |
| 11.1 输入/输出包                                 | 253 |
| 11.1.1 I/O 流                                | 253 |
| 11.1.2 文件管理                                 | 256 |
| 11.2 节点流                                    | 260 |
| 11.2.1 FileReader 类                         | 260 |
| 11.2.2 FileWriter 类                         | 262 |
| 11.2.3 FileInputStream 和 FileOutputStream 类 | 264 |
| 11.3 过滤流                                    | 267 |
| 11.3.1 使用 I/O 缓冲区                           | 267 |
| 11.3.2 基本类型数据传输                             | 270 |
| 11.3.3 PrintWriter 类                        | 273 |
| 11.4 标准输入/输出流                               | 274 |
| 11.4.1 标准输入流 in                             | 275 |
| 11.4.2 标准输出流 out                            | 275 |
| 11.4.3 标准错误输出流 err                          | 276 |
| 11.4.4 System 类常用方法                         | 276 |
| 习题 11                                       | 278 |
| <b>第 12 章 网络程序设计</b>                        | 280 |
| 12.1 网络程序设计基础                               | 280 |
| 12.1.1 TCP/IP 编程简介                          | 280 |
| 12.1.2 Java 网络编程机制                          | 281 |
| 12.1.3 Socket 程序设计实例                        | 283 |
| 12.2 数据库接口                                  | 290 |
| 12.2.1 JDBC 简介                              | 290 |
| 12.2.2 JDBC-ODBC 桥接器                        | 291 |
| 12.2.3 JDBC 程序实例                            | 292 |
| 12.3 JSP 程序设计                               | 294 |
| 12.3.1 安装和配置 JDK                            | 295 |
| 12.3.2 安装和配置 Tomcat                         | 296 |
| 12.3.3 JSP 程序构成                             | 299 |
| 习题 12                                       | 303 |
| <b>附录</b>                                   | 304 |

# 第1章 概述

Java 是一门发展很快的计算机语言,现在许多大型项目开始逐渐考虑使用 Java。Java 以其独特的可跨平台运行和开发速度快、维护方便等优点受到许多开发者的青睐。本章介绍了 Java 的起源,发展的历史及其特点;Java 程序的基本结构;面向对象程序设计的基本概念;上机操作过程。

## 1.1 Java 程序设计概述

1995 年 5 月 23 日,SUN 公司正式对外发布了 Java 语言规范,此后经过多年发展,现在 Java 已经真正成长为严格的、主流的程序设计语言。

### 1.1.1 Java 开放式平台

Java 开放式平台包括 J2EE、J2SE 和 J2ME。随着这三大平台的迅速推进,Java 的应用领域也在不断拓展。在 2005 年的 SUN ONE 大会上,SUN 公司把 J2SE、J2EE 和 J2ME 改为 Java SE、Java EE 和 Java ME。下面简要介绍这三大平台。

#### 1. Java SE

Java SE 是 Java 标准版的英文缩写,主要针对桌面软件的开发。目前,桌面软件开发有三种图形界面构架,分别是 AWT、Swing 和 SWT。

AWT 的历史比较悠久,早在 JDK1.0 的时候就已经有了。AWT 的原理是找出各种平台下图形控件的交集,然后用 AWT 分别调用各个平台下控件的本地代码来实现图形界面。由于其控件是每个操作系统的交集,因此控件种类较少,功能也不强。目前普遍使用的是 Swing 和 SWT。

Swing 是 SUN 公司提供的,其实现原理是取所有平台下图形控件集合,用 Java 虚拟机模拟这些控件。Swing 的图形控件种类比较丰富,功能也比较强大,而且是用纯 Java 实现的,所以跨平台性很好。但由于 Swing 的控件采用 Java 虚拟机模拟,因此运行速度比较慢。

IBM 推出的 SWT 折中了 AWT 和 Swing 技术方案。如果本地平台中拥有指定的控件,那么 SWT 就直接调用本地控件;如果本地平台没有,那么 SWT 会仿照 Swing 来模拟该控件的功能。

#### 2. Java EE

Java EE 是 Java 企业版的英文缩写,其技术基础是 Java SE。Java EE 不仅巩固了 Java SE 中的许多优点(例如“编写一次、随处运行”的特性,方便存取数据库的 JDBC API、CORBA 技术以及能够在 Internet 应用中保护数据的安全模式等),还提供了对 EJB、Java Servlets API、JSP 以及 XML 技术的全面支持。其最终目的就是成为一个能够使企业开发者大幅缩短投放市场时间的体系结构。

Java EE 体系结构提供中间层集成框架以满足用户需求。它通过提供统一的开发平台降

低开发多层应用的成本和复杂性,对现有应用系统提供强有力的支持,增强了安全机制。

### 3. Java ME

Java ME 是 Java 微型版的英文缩写。也许 Java 的设计初衷就是想按照 Java ME 的路线来发展。当初 SUN 公司开发 Java 的目的是为了在各种复杂的电子设备中提供一个通用的技术,使同样的软件可以在不同的电子设备中顺利运行。也许是“有心栽花花不开,无心插柳柳成荫”,Java 在互联网上大显身手,先是在 Applet 中展现其魅力,后又在 Java EE 的 Web Service 中大红大紫。但 Java ME 一直不温不火,很长一段时间,Java ME 都只被当作开发手机游戏的工具,其潜在能力还没有被完全挖掘出来。不过,随着网络的不断发展,机顶盒、手机等智能设备的硬件条件也越来越好,在这些领域中 Java ME 应该会大显身手。

#### 1.1.2 Java 语言的特点

Java 具有简单、灵活、功能强大的优点。采用 Java 可以进行面向对象程序设计,可以进行对象的事件描述、处理和综合,能够进行可视化软件开发,实现动态画面的设计,可以进行交互操作设计,如选择交互、定向交互、控制流程等。Java 可以与数据库连接,用 SQL 语句实现数据库操作。Java 应用程序具有十分快速的实时性、精减性、安全性和完全的开放性。Java 的应用领域广阔,例如企业信息化、电子商务、电子政务、科学计算,软件工程等,还可以把 Java 芯片嵌入到移动电话、电视机、洗衣机等家用电器设备。总的来说,Java 具有面向对象、高度安全、跨平台、多线程和交互性的特点。

##### 1. 面向对象

Java 语言是在面向对象程序设计语言 C++ 的基础上发展形成的,并扩充了 C++ 中语言的对象模型,除了 Java 语言的基本数据类型以外,其他都是以对象来表示的。同时还删除了 C++ 语言的某些复杂的特性,Java 中没有结构联合,抛弃了多重继承,不提供运算重载,取消了指针和内存分配等。这样使 Java 应用程序更易编制和掌握。

##### 2. 高度安全

作为一种网络编程语言,其安全性是起决定作用的。当用户使用浏览器在互联网上漫游时,Java 小应用程序便会随时传到用户的计算机上,而且所传输的不是静态的数据文件,而是可执行代码。

Java 的安全性是通过多个层次上的措施得到保证的。首先,从内存管理上加强安全。对程序员来说,内存分配是透明的,即程序员没有对内存的分配权,这样减少了许多内存出错的可能。另一方面,在 Java 中编译器也没有对内存布局的确定权,Java 把内存分配权交给运行平台。此外,类装载器也为安全性提供了保障措施。类装载器在装载字节码时,将本地的类组成一个统一的空间,而将外来的类组成另一个空间,这样就为本地类建立了一个安全屏障。

##### 3. 跨平台

用 Java 语言写的程序经 Java 编译生成字节码后可以在任何符合 Java 规范的 Java 虚拟机上运行,与操作系统无关,即用 Java 写的各种程序可在 Windows、Unix 等系统平台上运行。

Java 采用了一种半编译、半解释的方法,并定义出 Java 虚拟机概念。所谓虚拟机就是一种假设出来但不存在的平台。Java 源代码先经过 Java 编译器产生出 Java 虚拟机的机器码——字节码(Bytecode),再经 Java 解释器将字节码转换成实际系统平台上的机器码,去真正执行。

Java 的跨平台性正是网络传输所需要的。由于支持 Java 的最主要的客户程序(即 WWW 浏览器)内嵌入了 Java 解释器,因此 Java 字节码从服务器端转到客户端后,便会被客户端浏览器解释成当前平台的机器码来运行,而不用担心 Java 的字节码会由于服务器端的平台不同而编译出不同的目标码。

#### 4. 多线程

单线程应用程序,只执行一个线程,一次只能做一件事,在它运行时,其他用户接口不会响应;多线程应用程序,可以同时独立执行多个线程,这些线程可以交流或合作,如 Windows 中的 32 位执行程序,就可称为进程,同时,Windows 运行后,就有多达十几个线程在运行;Java 语言全面地支持多线程机制,并还能实现许多其他语言难以实现的数据同步化过程。

#### 5. 交互性

Java 程序可以使 Internet 上的网页互动起来。Java 动画只是 Java 功能的一个直观反映,实际上 Java 与众不同的关键之处是 Java 小应用程序(Applet)能被传输。Java 的字节码被客户端浏览器解释后便成为通过网络传输的可执行代码,既然是一种可执行程序,就能在客户端与用户之间实现即时性交互。

### 1.1.3 面向对象要素

面向对象程序设计(Object Oriented Programming,简记为 OOP)是目前占据主流地位的一种程序设计技术,它试图用客观世界中描述事物的方法来描述一个程序要解决的问题。Java 是一种面向对象程序设计语言,在 Java 中通过引入类(Class)、对象(Object)、消息(Message)、继承(Inherit)等概念,增加了程序模块的独立性和可扩展性,为软件开发人员和软件产品提供了很多便利。

对象(Object)是 OOP 中最重要的概念之一。简单地说,对象是一个抽象的概念,它是对一个客观实体的描述。它是既有数据又有对数据进行操作的代码的逻辑实体。由此,面向对象程序设计,就是用对象来描述客观世界中所需说明的有关事物。

Java 支持面向对象程序设计的几个要素,即封装性、重载性、继承和派生性、多态性。下面简单介绍这些要素。

#### 1. 封装性

将描述对象的数据与对这些数据进行处理程序代码有机地组成一个整体,形成一个模块,对这些数据与代码的存取权限加以限制后,模块完全独立。对象的这种特性称为封装性。这样就使得描述对象的数据只能通过对象中的程序代码来处理,而其他任何程序代码均不能访问对象中的数据。封装性可以通过定义类来实现,对象是类的一个实例。封装性非常有利于程序的调试和维护。

#### 2. 多态性

在面向对象程序设计中,按一定格式传递的信息称为消息,同一个消息被不同的对象接收时,可以导致不同的行为,这种特性称为多态性。多态性的重要性在于允许一个类体系的不同对象各自以不同的方式响应同一个消息,这就可以实现“同一接口,多种方法”。多态性不仅提高了程序设计的灵活性,而且大大减轻了类体系使用者的记忆负担。

#### 3. 重载性

一个函数名或一个运算符根据不同的对象可以完成不同的功能或运算,这种特性称为重

载性。例如：“+”运算符可以完成两个整数的求和运算，也可以完成两个实数的求和运算，还可以完成两个复数的求和运算，也可以完成两个字符串的拼接。在 Java 中有两种重载，一种是运算符的重载，另一种是函数的重载，例如用相同的函数名 abs() 可以分别求整数、实数和双精度实数的绝对值。重载性为编程提供了极大的便利。

4. 继承和派生性

一个类可以派生出新的类。其中，原来的类称为基类，新的类称为派生类。派生类可以全部或部分地继承基类的数据或程序代码，这种特性称为继承和派生性。派生类又可以作为其他类的基类，派生出新的类。这样一层一层地继承和派生下去，可形成一棵树状的类结构。

利用这种特性，对于类同的问题或只有部分类同的问题，都可以从已定义的类中派生出来，而不需要做重复性的工作。这样，一方面可减少程序设计的错误(利用原来已调试好的类)，另一方面可加快和简化程序设计，提高工作效率。

1.2 Java 程序简介

通常，Java 的开发环境约定为：当文件的扩展名为 Java 时，则为 Java 语言源程序；当文件的扩展名为 class 时，则为字节码程序，可以直接在 Java 虚拟机(JVM)上运行。

1.2.1 简单 Java 程序

我们首先通过一个简单的例子来说明 Java 程序的基本结构及其主要特点。

例 1-1：一个简单的 Java 程序。

|                                         |       |
|-----------------------------------------|-------|
| /* 源程序文件名为:ex1_1.java */                | 第 1 行 |
| /* * Java 程序的基本结构 */                    | 第 2 行 |
| import java.lang.System;                | 第 3 行 |
| public class ex1_1{                     | 第 4 行 |
| public static void main(String[] args){ | 第 5 行 |
| int n = 4;                              | 第 6 行 |
| System.out.println(n+"平方的输出是 "+n*n);    | 第 7 行 |
| }                                       | 第 8 行 |
| }                                       | 第 9 行 |

该程序经编译后，运行可执行程序时，在显示器上显示：

4 平方的输出是 16

对这个程序的基本结构和各语句的作用说明如下：

1. 注解或说明信息

在 Java 程序中插入注解信息是为了提高程序的可读性，便于程序的维护。有三种方法进行注解：第一种是用“/\*”和“\*/”把注解信息括起来，这种注解可以出现在程序中的任何位置；第二种是用“/\*\*”和“\*/”把注解信息括起来；第三种是用两个连续的“/”字符(即“//”)，它表示从“//”开始到本行结束均为注解。

## 2. 类文件

第3行使用关键字 `import` 引入了在程序外部定义的类 `System`,它是在包 `java.lang` 中定义的,因而引入时必须指出完整的路径。第4行使用关键字 `class` 来声明将要定义的新类,`ex1_1` 是类的名称。整个类的定义包括代码和数据都在本行的“`{`”和与之对应的第8行上的“`}`”之间。

## 3. 主方法

任何一个 Java 应用程序(Application)均要有一个且只能有一个主方法。一个 Java 程序总是从 `main()` 方法开始执行,而不考虑其在整个程序中的位置。

## 4. 花括号对

`{}` 称为方法括号或语句括号。任一方法体均以“`{`”开始,并以“`}`”结束。注意:花括号要配对使用。

## 5. 语句符号

任何一个 Java 应用程序均由一个或多个方法组成,其中必须有一个主方法 `main()`,其余方法可有可无。任一方法可由若干个语句组成,每个语句均以“`;`”结束。

## 6. 程序的书写规则

对于 Java 的编译器而言,一个语句可以写成若干行,一行内也可以写若干个语句。Java 允许的书写格式非常自由,但是为了便于程序的阅读和相互交流,程序的书写必须符合以下基本规则:

(1) 对齐规则。同一层次的语句必须从同一列开始,同一层次的“`{`”必须与对应的“`}`”在同一列上。

(2) 缩进规则。属于内一层次的语句必须缩进几个字符,通常缩进两个、四个或八个字符的位置。

(3) 任一函数的定义均从第一列开始书写。

## 7. 输入和输出

Java 语言没有专门的输入输出语句,其输入和输出是通过方法来实现的,如 `System.out.println()` 为输出方法。

## 8. 严格区分大小写字母

某些高级语言不区分大小写字母,但 Java 是严格区分大小写字母的,如 `A` 与 `a` 表示两个不同的标识符。在书写程序或编辑程序时,要注意这一点。

# 1.2.2 程序的基本要求

用 Java 语言进行程序设计并解决实际问题时,对设计的程序有以下几个基本要求:

## 1. 正确性

要求程序正确无误,包括两个方面:语法和语义正确;算法描述正确。这是对程序的最基本的要求。

## 2. 可读性和可理解性

程序应该容易被读懂,使人容易理解程序的设计思想和设计方法。这通常包括三个方面:一是程序的结构性好,要求采用结构化的程序设计方法或软件工程的程序设计方法来设计程序;二是在程序中增加注解,说明程序设计的思想和方法;三是程序的书写格式规范。

### 3. 可维护性

程序要易于修改,易于增加新的功能。这要求程序的结构性好,各模块的独立性强。

### 4. 程序的构思好,程序简短,执行速度快

与前三点要求相比,这个要求略高,但也应尽力做到。要设计出高质量的程序,仅学习本课程的知识是不够的,还要学习数据结构、算法设计与分析、软件工程及程序设计方法等方面的知识。希望大家在相关课程的学习中掌握这些知识,并通过大量的程序设计实践,编写出能解决实际问题的高质量的程序。

## 1.2.3 Java 程序开发步骤

对于高级语言编写的源程序而言,编译是指将其翻译成可运行的目标程序,而目标程序作为一个独立的文件可以无数次地运行。编译过程所需要的存储空间较大,编译的时间较长,但目标程序执行速度快,这种方式特别适用于重复使用的程序。

解释是指对高级语言编写的源程序翻译一句执行一句,翻译和运行交叉进行。如果要再运行一次,那就必须重新翻译,重新执行。因此,解释型语言的执行速度远远低于编译型语言。解释型语言适用于需要经常修改程序的情况。

Java 是一种半编译半解释的语言。Java 程序的执行过程是:先由 Java 编译器将其编译成字节码,运行时由运行系统上的字节码解释器将其翻译成机器语言。Java 字节码解释器的工作是边翻译边执行,但因为字节码已非常接近于机器码,所以速度仍相当快。而且在一个系统上编写并编译好的 Java 程序字节码可以不加任何变动地放到另一系统上运行。

设计好的 Java 源程序需要经过编译生成可在 Java 虚拟机上执行的程序文件,然后执行并调试程序。Java 程序的开发与执行流程如图 1-1 所示。

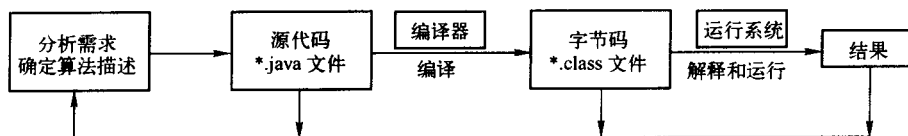


图 1-1 Java 程序的开发与执行流程

Java 程序开发可分为如下四个步骤:

#### 1. 需求分析

根据要解决的问题分析需求,并用合适的方法描述之。

#### 2. 编写源程序

编写 Java 源程序,并利用编辑器将源程序输入到计算机内的某个文件中,文件的扩展名为 java。

#### 3. 编译

编译源程序,生成字节码程序,文件的扩展名为 class。

#### 4. 调试

运行可执行文件,分析运行结果。若结果不正确,则要修改源程序,并重复以上的过程,直到结果正确为止。



1.3 编辑和运行 Java 程序

JDK(Java Development Kit)是 Java 的标准基础开发工具包,也是大多数 Java 开发者遵循和使用的标准开发环境。尽管现在已经出现许多第三方的高级 Java 开发工具,但仍以 JDK 作为其基础。JDK 已成为事实上的工业标准,只有遵循其规定,所开发的 Java 程序才能真正做到跨平台运行。

1.3.1 Java 程序运行环境

JDK 包括一个标准类库和一组建立、测试及建立文档的 Java 实用程序。JDK 的核心是以 Java 应用程序接口(API)给出的一些预定义的类库,开发人员需要用这些类库中的类来访问 Java 语言的功能。

Sun 公司目前已经为各种主流操作系统平台提供了相应的 JDK 版本。根据实际情况,本书只讨论目前常用在 Windows 操作系统平台下的 Java 开发环境。本书使用的 Java 开发包 JDK 可以从 SUN 公司的网站或者其他网站下载。SUN 公司网站的 JDK 下载地址是:<http://java.sun.com>。

如果 JDK 安装程序是 jdk-6-windows-i586.exe,以默认方式运行安装程序完毕,就会发现在指定目录(d:\jdk1.6.0)中安装了对应的 JDK 系统,如图 1-2 所示。在其安装路径下的子目录名称及存放内容如表 1-1 所示。这些子目录的有无与所选择的安装组件有关。

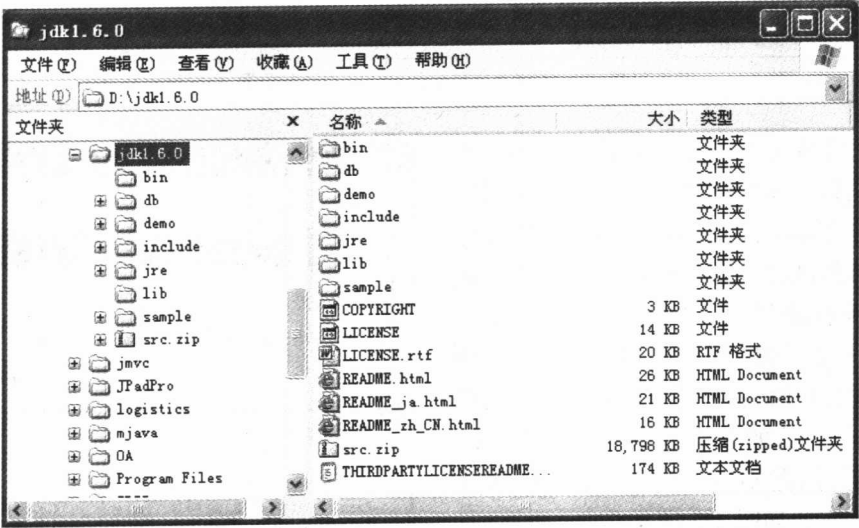


图 1-2 JDK 安装完成后的目录

表 1-1 安装路径下的子目录名称及存放内容

| 目录名称 | 存放内容                                                                        |
|------|-----------------------------------------------------------------------------|
| bin  | 开发工具,可帮助用户开发、执行、调试和保存 Java 程序                                               |
| demo | 演示 applet 和应用程序,带有源代码的 Java 编程示例,包括使用 Swing 和其他 Java 基类以及 Java 平台调试器体系结构的示例 |