2012年全国电子信息类优秀教材　浙江省“十一五”重点教材建设项目

21世纪高等教育网络工程规划教材

21st Century University Planned Textbooks of Network Engineering

Java面向对象程序设计（第3版）

Java Object-Oriented Programming (3rd Edition)

杨晓燕 李选平◎主编

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

2012年全国电子信息类

21世纪高等教育网络工程规

21st Century University Planned Textbooks of Networ

“十五”重点教材建设项目

Java面向对象程序设计（第3版）

Java Object-Oriented Programming (3rd Edition)

杨晓燕 李选平◎主编

人民邮电出版社

北京

图书在版编目（CIP）数据

Java面向对象程序设计 / 杨晓燕，李选平主编. -- 3版. -- 北京 : 人民邮电出版社，2015.8
21世纪高等教育网络工程规划教材
ISBN 978-7-115-39419-4

Ⅰ. ①J… Ⅱ. ①杨… ②李… 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(2015)第156402号

内 容 提 要

Java 是纯面向对象程序设计语言，具有完全面向对象、简单高效、与平台无关、支持多线程等显著特点，在“互联网+”迅速发展的今天，有非常广阔的应用领域。本书通过“案例分析—知识学习—案例实现”的结构，本着 Java 入门和架构应用并重的原则进行编写，内容主要包括：Java 概述、Java 语言基础、Java 输入/输出、程序流程控制结构和方法、数组、Java 类和对象、类的继承和接口、包和异常、面向对象程序设计的基本原则及设计模式初步、图形处理、图形用户界面、多线程等。

书中示例程序在 JDK 5.0/JDK 6.0 中经过验证，并都给出运行结果。本书注意重要内容的提炼，重点内容重点提示，使读者便于学习和理解。同时为了便于教师教学，主要的程序代码都增加了行号。课后习题分为学习内容积木化的练习和拓展研讨题，提供参考答案。若将本书与其配套实验教材—《Java 面向对象程序设计实践教程（第 3 版）》（杨晓燕　李选平主编）一起使用，效果更好。

本书既可作为 Java 程序设计的教材，也可作为 Java 自学者的入门用书。

◆ 主　　编　杨晓燕　李选平
责任编辑　邹文波
责任印制　沈　蓉　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　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：19　　2015 年 8 月第 3 版
字数：499 千字　　2015 年 8 月河北第 1 次印刷

定价：42.00 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

第 3 版前言

诞生于 1995 年的 Java 语言，已经发展为最主流的面向对象程序设计语言，它简单、高效，与平台无关，是计算机世界的“国际语言”。面向对象技术具有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现代大规模软件开发的要求，成为计算机应用开发领域的主流趋势。不仅如此，Java 的跨平台造就它在 Internet 上无可比拟的应用前景，使得 Java 成为当今互联网和移动互联网领域最流行、最受欢迎的一种程序开发语言。原 Sun 公司总裁兼首席运营官 Jonathan Schwartz 说：“Java 技术正在成为全球网络应用的事实标准，它将大大加快和简化提供移动、消费和企业市场的服务。”现在 Java 平台仍继续为 Java 经济注入活力，驱动全球企业在移动应用和服务器领域的技术创新。

2005 年在 Java 发布 10 周年之际，我们编写了第 1 版 Java 程序设计教程，受到读者好评，连续加印多次；2009 年和 2010 年本书获得宁波市教育局和浙江省教育厅的立项赞助，2012 年出版了第 2 版，本书获得浙江省“十一五”重点教材建设项目以及其他很多赞誉；2015 年重新编写了第 3 版。本次修订在 Java 内容体系和结构上做了微调，使知识点的衔接更合理，教学重点更突出，同时增加了新技术。通过教材案例的引领，突出“学中用”和“用中学”。本书具有以下特色。

- 与其他 Java 图书相比，本书的每一章都由“案例分析”开篇，“案例实现”收尾，案例大小适中，能够快速入门。
- 遵循“案例提出问题——知识学习——案例实现”的体例，为了便于理解，又把综合案例分解为可独立运行的子案例。
- 四化设计：核心知识案例化，抽象概念形象化，复杂问题通过分解尽量简单化，综合知识项目化。
- 基于学生学习实际需要，通过简单程序和图示说明，将内容进行了整合，便于学生交互使用。教材内容循序渐进，程序注重前后衔接和对比，环环相扣。为了便于教师讲解和读者理解，书中对主要程序代码增加了行号。
- 本书不仅注重一般概念和理论的叙述，而且注重系统开发过程中结构和模式的研究。
- 课后练习分 3 部分：习题是积木式内容重建，问题探究是在知识广度和深度上的拓展，OCJP 题为学生打开一扇认证之窗。

本书编写的初衷是重在应用。本书每章后边的习题、问题探究及 OCJP 提供参考答案。

本书出版之际，感谢我的师长姜遇姬教授的指导，感谢我的同事邓芳、刘臻、张梁斌以及我的学生的帮助与支持。

PPT 及相关源码、习题答案参考及实验安排资料可通过编者邮箱 yangxy3225@163.com 索要，也可以在人民邮电出版社教学服务与资源网（www.ptpedu.com.cn）下载。

由于编者水平所限，书中难免还存在一些缺点和错误，希望读者批评指正。

编　者

2015 年 5 月

目　录

第 1 章　Java 概述 …… 1

【案例分析】 …… 1

1.1　Java 的崛起 …… 2

1.2　Java 与 C++、C …… 3

1.2.1　Java 和 C++ …… 3

1.2.2　Java 与 C …… 3

1.3　Java 语言的特点 …… 5

1.4　Java 程序的类型及其不同的编程模式 …… 6

1.5　Java 程序开发过程 …… 7

1.6　Java 开发工具入门 …… 13

1.6.1　JDK 的下载、安装 …… 13

1.6.2　配置环境变量 …… 15

1.6.3　JDK 开发工具简介 …… 17

1.6.4　Java 程序开发步骤小结 …… 17

习题 1 …… 17

问题探究 1 …… 18

第 2 章　Java 语言基础 …… 19

【案例分析】 …… 19

2.1　标识符、关键字和分隔符 …… 19

2.1.1　标识符和关键字 …… 19

2.1.2　分隔符 …… 20

2.2　数据类型 …… 21

2.2.1　基本数据类型 …… 22

2.2.2　变量 …… 23

2.3　运算符与表达式 …… 25

2.3.1　算术运算符 …… 25

2.3.2　赋值运算符 …… 26

2.3.3　关系运算符 …… 26

2.3.4　逻辑运算符 …… 27

2.3.5　条件运算符 …… 28

2.3.6　其他运算符 …… 28

2.3.7　运算符的优先级 …… 28

2.4　字符串 …… 29

2.4.1　创建 String 对象 …… 29

2.4.2　创建 StringBuffer 对象 …… 30

2.5　案例实现 …… 31

习题 2 …… 32

问题探究 2 …… 33

第 3 章　Java 输入/输出 …… 34

【案例分析】 …… 34

3.1　标准输入/输出方法 …… 34

3.1.1　标准输出方法 …… 35

3.1.2　Scanner 键盘输入类 …… 36

3.1.3　read 方法的使用 …… 38

3.2　命令行参数输入法的应用 …… 39

3.3　流式交互输入/输出的应用 …… 40

3.3.1　应用举例 …… 40

3.3.2　Java I/O 基本模型 …… 41

3.4　文件数据的读/写 …… 42

3.5　JOptionPane 对话框输入法 …… 45

3.6　案例实现 …… 46

习题 3 …… 47

问题探究 3 …… 47

第 4 章　程序流程控制结构和方法 …… 49

【案例分析】 …… 49

4.1　语句和程序流程控制结构 …… 49

4.2　选择结构 …… 50

4.2.1　选择语句 …… 50

4.2.2　多选择结构 switch 语句 …… 53

4.3　循环结构 …… 57

4.3.1　3 种循环语句 …… 57

4.3.2　循环程序结构小结 …… 60

4.3.3 循环嵌套和 continue、break 语句 …… 61
4.4 算法设计 …… 63
4.4.1 迭代算法 …… 64
4.4.2 穷举算法 …… 64
4.4.3 递归算法 …… 65
4.5 案例实现 …… 67
习题 4 …… 68
问题探究 4 …… 69

第 5 章 数组 …… 70

【案例分析】 …… 70
5.1 数组的基本概念 …… 70
5.2 一维数组 …… 71
5.2.1 一维数组的声明 …… 71
5.2.2 一维数组内存申请 …… 71
5.2.3 一维数组的初始化 …… 72
5.2.4 测定数组的长度 …… 73
5.2.5 for-each 语句与数组 …… 74
5.3 二维数组 …… 74
5.3.1 认识二维数组 …… 75
5.3.2 二维数组的声明与创建 …… 75
5.3.3 二维数组元素的初始化 …… 76
5.3.4 二维数组的引用 …… 77
5.4 案例实现 …… 79
习题 5 …… 81
问题探究 5 …… 81

第 6 章 Java 类和对象 …… 83

【案例分析】 …… 83
6.1 面向对象编程 …… 83
6.2 类的描述 …… 86
6.2.1 类的定义 …… 86
6.2.2 成员变量的访问控制符 …… 87
6.2.3 成员方法 …… 89
6.2.4 成员变量和局部变量 …… 90
6.2.5 final 变量 …… 91
6.3 对象的创建与使用 …… 91
6.3.1 对象的创建 …… 92
6.3.2 对象的比较 …… 93
6.3.3 对象的使用 …… 94
6.3.4 释放对象 …… 97
6.3.5 Java 变量内存分配 …… 98
6.3.6 匿名对象 …… 99
6.4 类的构造方法 …… 99
6.4.1 构造方法的作用和定义 …… 99
6.4.2 this 引用 …… 101
6.5 static 变量及 static 方法 …… 103
6.5.1 static 变量 …… 104
6.5.2 static 方法 …… 105
6.6 对象初始化过程 …… 108
6.7 成员方法 …… 110
6.7.1 方法调用与参数传递方式 …… 110
6.7.2 方法重载 …… 113
6.7.3 final 最终方法和 abstract 抽象方法 …… 114
6.8 复杂程序解决方案和方法 …… 115
6.9 案例实现 …… 118
习题 6 …… 120
问题探究 6 …… 121

第 7 章 类的继承和接口 …… 124

【案例分析】 …… 124
7.1 类的继承 …… 124
7.1.1 继承的概念 …… 124
7.1.2 创建子类 …… 125
7.1.3 关于父类的构造方法 …… 126
7.2 成员变量的隐藏和成员方法的重构 …… 129
7.3 抽象类 …… 131
7.4 接口 …… 132
7.4.1 接口概述 …… 133
7.4.2 接口的定义 …… 133
7.4.3 实现接口的类定义 …… 133
7.4.4 接口的多态性 …… 136
7.5 泛型 …… 138
7.5.1 泛型的概念和泛型类的声明 …… 138
7.5.2 泛型应用 …… 139
7.5.3 ArrayLsit 泛型数据结构 …… 140
7.6 案例实现 …… 141
习题 7 …… 144

问题探究 7 …… 145

第 8 章 包和异常 …… 146

【案例分析】 …… 146
8.1 包 …… 146
8.1.1 创建包 …… 147
8.1.2 类的包外引用 …… 148
8.1.3 jar 命令打包与引用 …… 150
8.2 异常处理 …… 152
8.2.1 异常的基本概念 …… 152
8.2.2 异常处理机制 …… 155
8.2.3 自定义异常类 …… 159
8.2.4 GUI 应用程序的异常处理 …… 159
8.3 案例实现 …… 161
习题 8 …… 163
问题探究 8 …… 164

第 9 章 面向对象程序设计的基本原则及设计模式初步 …… 165

【案例分析】 …… 165
9.1 UML 类图 …… 165
9.1.1 类的 UML 图 …… 166
9.1.2 UML 接口表示 …… 166
9.1.3 UML 依赖关系 …… 166
9.1.4 UML 关联关系 …… 167
9.1.5 UML 聚合关系 …… 167
9.1.6 UML 组合关系 …… 168
9.1.7 泛化关系 …… 168
9.1.8 实现关系 …… 168
9.2 面向对象程序设计的基本原则 …… 169
9.2.1 发现变化，封装变化 …… 169
9.2.2 单一职责原则和最少知识原则 …… 172
9.2.3 开放-封闭原则 …… 172
9.2.4 子类型能够替换基类型原则 …… 173
9.2.5 合成/聚合复用原则 …… 175
9.3 案例实现 …… 175
习题 9 …… 179
问题探究 9 …… 179

第 10 章 图形处理 …… 181

【案例分析】 …… 181
10.1 Java 坐标系 …… 181
10.2 图形的颜色控制 …… 182
10.3 Graphics 类的基本图形 …… 184
10.3.1 绘制直线和矩形 …… 185
10.3.2 绘制圆弧 …… 187
10.3.3 绘制多边形 …… 188
10.4 案例实现 …… 189
习题 10 …… 190
问题探究 10 …… 190

第 11 章 图形用户界面 …… 191

【案例分析】 …… 191
11.1 图形用户界面概述 …… 192
11.1.1 图形用户界面组件 …… 192
11.1.2 组件分类 …… 192
11.1.3 常用容器类的应用 …… 193
11.2 事件处理 …… 197
11.2.1 基本概念 …… 198
11.2.2 事件处理机制 …… 199
11.2.3 事件处理的实现方式 …… 200
11.2.4 适配器类 …… 204
11.3 一般组件 …… 206
11.3.1 标签 …… 206
11.3.2 按钮 …… 207
11.3.3 文本框 …… 207
11.3.4 文本区 …… 208
11.3.5 列表框组件 …… 209
11.3.6 滚动窗格 …… 211
11.3.7 复选框和单选按钮 …… 212
11.3.8 滑动条 …… 214
11.3.9 多事件处理示例 …… 216
11.4 菜单与对话框 …… 218
11.4.1 创建菜单 …… 218
11.4.2 弹出式菜单 …… 221
11.4.3 对话框 …… 223
11.5 布局管理器 …… 228
11.5.1 顺序布局——FlowLayout …… 228

11.5.2 边界布局——BorderLayout …… 228
11.5.3 网格布局——GridLayout …… 229
11.5.4 卡片布局——CardLayout …… 230
11.5.5 手工布局 …… 231
11.6 JApplet 的使用 …… 232
11.7 Java 事件类方法列表 …… 233
11.8 案例实现 …… 234
习题 11 …… 238
问题探究 11 …… 241

第 12 章 多线程 …… 242

【案例分析】 …… 242
12.1 多线程概述 …… 242
12.1.1 基本概念 …… 243
12.1.2 线程的状态与生命周期 …… 244
12.1.3 线程的调度与优先级 …… 246
12.2 创建和运行线程 …… 247
12.2.1 利用 Thread 类创建线程 …… 247
12.2.2 用 Runnable 接口创建线程 …… 249
12.3 线程间的数据共享 …… 250
12.4 多线程的同步控制 …… 253
12.4.1 线程同步相关概念 …… 253
12.4.2 synchronized 应用 …… 255
12.4.3 synchronized 的进一步说明 …… 257
12.5 案例实现 …… 258
习题 12 …… 260
问题探究 12 …… 261

第 13 章 综合案例——聊天通信 …… 263

13.1 界面及源代码 …… 263
13.2 应用程序框架分解 …… 267
13.2.1 Socket 连接的建立 …… 267
13.2.2 基于 TCP 的 Socket 数据通信架构 …… 269
13.2.3 图形用户界面与事件处理界面设计 …… 272
13.3 知识点 …… 274
13.3.1 网络通信的层次 …… 274
13.3.2 通信端口 …… 275
13.3.3 Java 网络编程中主要使用的类和可能产生的异常 …… 275
13.3.4 Socket 通信模式 …… 275
13.3.5 Socket 类和 ServerSocket 类的构造方法及常用方法 …… 276
13.3.6 API 系统中 DataInputStream 和 DataOutputStream 的应用 …… 277
13.3.7 多线程处理机制 …… 279
习题 13 …… 280

附录 部分习题参考答案 …… 281

第 1 章 Java 概述

为什么在 C/C++语言之后，Java 语言在业界得到快速普及和欢迎，而掌握 Java 语言会为就业和岗位提供很多优势？目前，中国有 200 多万 Java 程序员的空缺，全球拥有 450 万 Java 开发者。近年来，不论是计算机语言类的需求还是薪酬方面，Java 都一直占有领先的优势。让我们循序渐进，走进 Java，学习 Java，掌握 Java。

本章主要内容：

- Java 与 C、C++
- Java 语言的特点
- Java 程序的类型及其不同的编程模式
- Java 开发工具

【案例分析】

使用面向对象方法，描述现实世界中的一个实体——售报亭，如图 1.1 所示。

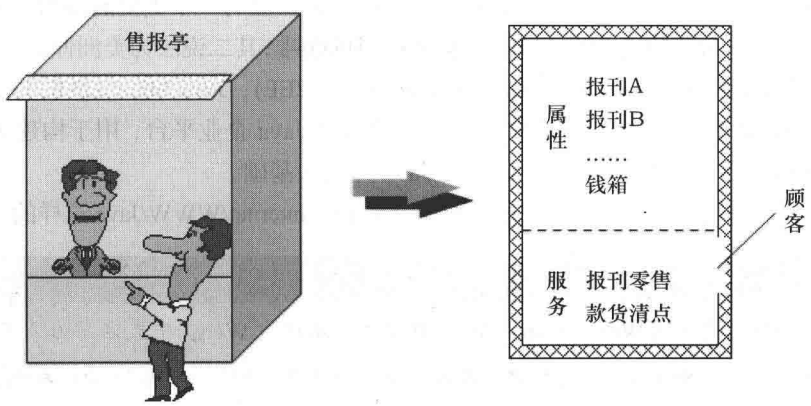

图 1.1　售报亭的对象封装

面向对象方法解决问题的思路：是从现实世界中的客观对象（如人和事物）入手，尽量运用人类的自然思维方式来构造软件系统。Java 就是这样的程序设计思路。

在面向对象方法中，把一切都看成是对象。把对象的属性和服务操作结合成一个独立的系统单位，其属性与操作刻画了事物的性质和行为，并尽可能隐蔽对象的内部细节。向外部只是提供

接口。软件对象是数据和方法的封装体。如图 1.1 所示，属性对应软件对象的数据，服务对应软件对象的方法。

在面向对象系统中，无论系统的构成成分，还是通过这些成分之间的关系而体现的系统结构，都可直接地映射问题域。这使得运用面向对象方法有利于正确理解问题域及系统责任。

1.1 Java 的崛起

1991 年，美国 Sun Microsystems 公司（简称 Sun 公司）启动了名为“Green Project”的研究项目，研究解决家用电器的智能通信和控制问题。开发小组最初构想以当时颇为流行的 C++语言开发此项智能软件。后来，由于 C++语言本身的复杂性、安全性以及平台移植方面的障碍与问题，项目组最后决定另辟蹊径，最终他们基于 C++重新开发了一套新的语言系统——Java 语言。

Java 语言的始创者是 Sun 公司的 James Gosling。起先，他根据办公室窗外的一棵橡树（Oak），将其命名为 Oak 语言。申请注册时，因为命名冲突问题，后来将其改名为 Java。

Oak 语言可以称得上是一种精巧而安全的语言。然而，一开始 Sun 公司就遭遇了“智能化家用电器”市场的萧条和冷遇。同时，Sun 公司以它投标一个自认为乐观的交互式电视项目时，也未能成功。在这种情况下，Oak 语言似乎生不逢时，Green 项目几乎走入了绝境。

1993 年万维网空前流行起来，Java 的发展可谓绝处逢生，其转向了网络应用领域。

Java 语言具有平台无关性，使得 Java 程序适应了 Internet 上多样化的服务器站点环境，Java 程序既可以在 Windows 平台运行，也可以在 UNIX、Linux 等平台上运行，造就了 Sun 公司宣传的“Write Once, Run Anywhere（一次编写，随处运行）”的优势。

1995 年 5 月，Sun 公司正式对外发布了 Java 语言，并随着互联网的飞速发展，逐渐确定了自己网络编程语言的地位。1995 年当年 Java 就被美国著名杂志《PC Magazine》评为十大优秀科技产品之一，1996 年 1 月 JDK1.0 发布。

之所以命名为 Java 语言，有两种说法：其一，印度尼西亚有一个重要的岛屿——爪哇岛，盛产咖啡，开发人员起名 Java 寓意为世人端上一杯热腾腾的咖啡；其二说法为美洲俚语——咖啡之意。

目前 Java 平台主要包括 Java SE（J2SE）、Java EE（J2EE）、Java ME（J2ME）。Java SE 称为 Java 标准版或 Java 标准平台；Java EE 称为 Java 企业版或 Java 企业平台，用于构建企业级的服务应用；Java ME 称为移动开发平台。Java SE 是 Java 开发的基础。

现在，在计算机领域中从来没有出现过像现在所发生的 Internet/WWW/Java 这样的“火爆”情况。

延伸阅读：万维网和因特网

WWW 是环球信息网（World Wide Web）的缩写，简称为 Web，中文名字为“万维网”。万维网包括 WWW 服务器和 WWW 浏览器。万维网是一个资源空间，由“统一资源标识符”（URL）标识。这些资源通过超文本传输协议（Hypertext Transfer Protocol）传送给使用者，而后者通过单击链接来获得资源。

因特网（Internet）是当前全球最大的、开放的、由众多网络相互连接而成的计算机网络。万维网常常被视作因特网的同义词，其实万维网是依赖因特网运行的一项服务。万维网是基于因特网的，万维网被广泛应用于因特网之上。

1.2　Java 与 C++、C

随着程序规模的不断扩大，在 20 世纪 60 年代末期出现了软件危机，当时的程序设计范型都无法克服错误随着代码的扩大而级数般扩大的问题，这个时候出现了一种新的程序设计范型——面向对象程序设计。

1.2.1　Java 和 C++

Sun 公司的 Java 开发小组汲取了 C++的精华，并将其组合到 Java 中，舍弃了 C++中低效率和不便于程序员使用的特性。Java 小组也创造了一些新的特性，给予 Java 开发基于 Internet 的应用程序时所必须的动态性。

Java 的目的并不是改进 C++并最终取代 C++。C++和 Java 这两种语言是设计用来解决不同问题的。Java 是用来设计必须共存于不同机器的应用程序——常常是基于 Internet 的基础之上。相反，C++用来开发在一台特定机器上运行的程序，尽管 C++程序被重新编译后也能够在其他机器上运行。

Java 语言的许多基本结构与 C++是相似的，有时甚至是相同的。例如，Java 是一种面向对象编程语言，它用类来创建对象的实例。类具有数据成员和方法成员，这和 C++中的类是相似的。

但是 Java 没有指针，在 C/C++编程语言中指针是一个基石。在 C++中正确使用指针能使程序富有效率，但是指针难以掌握，如果使用不当会导致运行错误。

Java 带有自动的垃圾收集器，这是在 C/C++中没有的功能。垃圾收集器是一个常规程序，收集程序中不再使用的内存，程序员不必编写代码来释放之前使用的内存。

在不同的平台上使用 C/C++程序使系统会对每种数据类型依平台的不同分配不同的字节数。而在 Java 中，会为各种数据类型分配合理的固定位数，在每种平台上都不改变，这样便保证了 Java 的平台无关性。

C++支持多重继承，一个类可以有多个父类，这种方式使 C++中的类可以使用多个父类的属性和方法，但结构特别混乱。而在 Java 中，一个类只能有一个父类，但是可以实现多个接口，这样既达到多重继承的目的，又保证了结构比多重继承更加清晰。

除此之外，与 C++不同，Java 不支持结构和联合，不支持宏定义，不支持头文件，不支持友元，这大大保证了 Java 程序的安全性。

1.2.2　Java 与 C

C 语言为面向过程的程序设计语言。面向过程程序设计语言在程序设计过程中都倾向于面向行为。C 语言中，程序设计的单元是函数。C 编程人员着重于编写函数。执行同一任务的一系列动作构成函数，一系列函数再构成程序。这种语言的主要问题是程序中的数据和操作分离，不能够有效地组成与自然界中的具体事物紧密对应的程序成分。

Java 是纯面向对象的程序设计语言，Java 语言中程序设计的单元是类，从类中创建一个一个实例对象。Java 编程人员着重创建用户自定义的类。每一个类均可包含数据属性和若干操作数据的函数。一个类的函数部分称为方法。C 和 Java 编程与执行过程区别如下。

Windows 下 C 语言开发过程如图 1.2 所示。C 语言程序在执行之前需要把程序编译成机器语言文件，程序执行效率高，依赖专门的编译器，跨平台性差一些。

图 1.2　Windows 下 C 语言开发过程

Java 语言开发过程如图 1.3 所示。

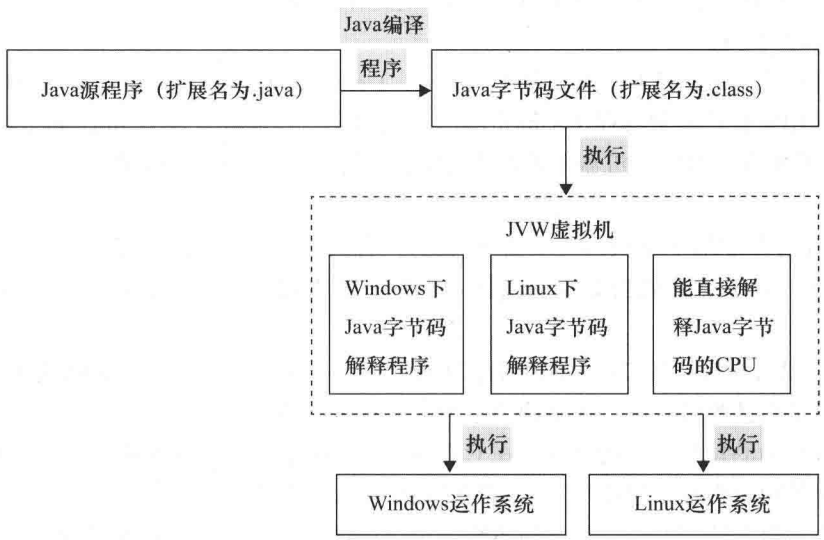

图 1.3　Java 语言开发过程

从图 1.2 和图 1.3 的比较中可以看出，Java 源程序编译后生成的字节码文件就相当于 C 源程序编译后 Windows 上的 EXE 可执行文件，JVM(Java Virtual Machine)虚拟机的作用类似 Windows 操作系统。在 Windows 上运行的是 EXE 文件，在 JVM 上运行的是 Java 字节码文件，即编译后生成的后缀为.class 文件。

Windows 执行 EXE 可执行文件的过程，就是从 EXE 文件中取出一条条的计算机指令，交给 CPU 去解释执行。字节码并不是机器指令，它不和特定的平台相关，不能被任何平台直接识别、执行。字节码是可以被 Java 虚拟机识别、执行的代码，即 Java 虚拟机负责解释、运行字节码，将字节码翻译成所在平台的机器码，并让当前平台运行该机器码。可见，只要能实现特定平台下的解释器程序，Java 字节码就能通过解释器程序在该平台上运行，这是 Java 跨平台的根本。

Java 兼顾解释性与编译性语言的特点，Java 源文件转换成 class 字节码文件过程是编译型的，class 字节码文件在操作系统上运行的过程则是解释型的，Java 虚拟机充当了解释器的作用，C/C++ 都是编译型的语言，运行速度较快。

延伸阅读：程序设计语言发展脉络

计算机程序设计语言的发展是一个不断演化的过程，从最开始的机器语言到汇编语言再到各种结构化高级语言，最后发展到面向对象程序设计语言。

机器语言是第一代计算机语言，是最原始的编程语言，用二进制代码（0 或 1）书写，能被机器直接识别。二进制是计算机语言的基础。在计算机发展初期，软件工程师们只能用晦涩的机器语言来编写程序。**汇编语言**将一个特定指令的二进制串机器指令映射为简洁的英文助记符。例如，用“ADD”代表加法，用“MOV”代表数据传递等。它是比机器语言“高层”的符号语言。**高级语言**是采用命令或语句的语言，屏蔽了机器的细节问题，提高了语言的抽象层次，正如我们正在学习的 Java。

1.3　Java 语言的特点

Java 语言是由 C++语言发展而来的，是一种彻底的面向对象的程序设计语言。作为一种纯面向对象的程序设计语言，它非常适合大型软件的开发，同时它去掉了 C++语言的一些容易引起错误的特性。例如，Java 语言没有指针，避免了使用指针直接访问物理寄存器带来的风险，Java 取消了运算符重载，以及 Java 语言用接口代替了 C++语言中容易引起混乱的多重继承机制等。Java 语言具有如下特点。

1. 面向对象

对象是程序的基本单元和构件。在面向对象的程序语言中，对象是类的实例，而类则是描述对象的模板。类是具有相同属性和服务的一组对象的抽象、一般描述。抽象是事物的泛化，抽象的目的是提取重要的特征而忽略不重要的细节。对象是现实世界中某一个实际存在的事物，软件对象是数据和方法的封装体。类与对象的关系，如同一个模具与用这个模具铸造出来的铸件之间的关系；如同自行车图纸和自行车的关系。

封装是面向对象的一个重要原则。它有两个含义：第一个含义是，把对象的全部属性和全部服务结合在一起，形成一个不可分割的独立单位（即对象）；第二个含义也称作“信息隐蔽”，即尽可能隐蔽对象的内部细节，对外形成一个边界（或者形成一道屏障），只保留有限的对外接口使之与外部发生联系。这主要是指对象的外部不能直接的存取对象的属性，只能通过几个允许外部使用的服务（或称方法）与对象发生联系。

2. 跨平台

这里所指的平台是由操作系统（OS）和处理器（CPU）所构成的。跨平台或与平台无关是指应用程序不因操作系统、处理器的变化而导致程序无法运行或出现运行错误。

用 Java 语言编写的程序，经过 Java 编译器编译后生成 Java 语言特有的字节码（Bytecode），而不生成特定的 CPU 机器代码。字节码是一种中间码，它比机器码更抽象，跨平台性能更好。Java 字节代码运行在 Java 虚拟机 JVM（Java 语言解释器）上。Java 语言借助 Java 虚拟机，首先对 Java 编译后生成的字节码进行解释，虚拟机底层的运行系统把字节代码转化成实际的硬件调用，再执行它。

JVM 是一种抽象机器，它附着在具体操作系统之上，本身具有一套虚拟机器指令，并有自己的栈、寄存器等。JVM 类似一个小巧而高效的 CPU，是一个应用程序仿真的软件实现。JVM 通常不是在硬件上实现（目前，SUN 公司已经设计实现了 Java 芯片，主要使用在网络计算机上。

另外，Java 芯片的出现也会使 Java 更容易嵌入到家用电器中）。Java 虚拟机是与硬件、软件平台有关的，它使 Java 语言程序在某一特定硬件、软件平台环境中直接运行目标代码指令。Java 编译出来的字节码与平台无关，这一点正是网络传输所需要的。

Java 主要靠 JVM 在目标代码级实现平台无关性。JVM 是 Java 平台无关的基础，Java 源代码先经过 Java 编译器生成 Java 虚拟机的字节码，再经过 Java 解释器将字节码转换成实际系统平台上的机器码，然后真正执行。任何一台机器只要配备了 Java 解释器，就可以运行字节码，而不管这种字节码是在何种平台上生成的。另外，Java 采用基于 IEEE 标准的数据类型。通过 JVM 保证数据类型的一致性，也就确保了 Java 的平台无关性。

3. 安全性

Java 将重点用于网络/分布式运算环境，确保建立无病毒且不会被侵入的系统。内存分配及布局由 Java 运行系统决定，字节码验证可以轻松构建出防病毒、防黑客的系统。

Java 最初设计目的是应用于电子类消费产品，要求有较高的可靠性。Java 虽然源于 C++，但它消除了很多 C++不可靠因素，可以防止很多编程错误。首先，Java 是强类型的语言，要求显式的方法声明，保证了编译器可以发现方法调用错误，保证程序更加可靠；其次，Java 不支持指针，杜绝了内存的非法访问；第三，Java 的自动单元收集防止了内存丢失等动态内存分配的问题；第四，Java 解释器运行时实施检查，可以发现数组和字符串访问的越界；最后，Java 提供了异常处理机制，便于程序即时发现运行错误。

由于 Java 主要用于网络应用程序开发，因此对安全性有着较高的要求。如果没有安全保证，用户从网络下载程序执行就非常危险。

4. 多线程

线程是操作系统的一种概念，被称为轻量级进程，是比传统进程更小的、可并发执行的单位。C 和 C++采用单线程体系结构，Java 提供了多线程支持。

一个线程是一个程序内部的顺序控制流。在 DOS 环境下我们只能同时运行一个程序，也就是程序只有一条顺序控制流。当一部分程序因为某种原因不能执行下去的时候，整个程序就停止在那里，其他的操作就不能执行。进程的特点是每个进程都有独立的代码和数据空间，进程切换的开销大。线程是轻量的进程，同一类线程共享代码和数据空间，每个线程有独立的运行栈和程序计数器，线程切换的开销小。通常，线程之间的切换是非常迅速的，使人们觉得好像所有的线程都是在同时执行似的。但是在系统内部来看，线程仍是串行执行的，只不过由于操作系统可以快速、自动地进行切换，从而给人一种并发执行的感觉。

多进程指在操作系统中，能同时运行多个任务（程序）。多线程是在同一应用程序中，有多个顺序流同时执行。如果多进程指在操作系统中我们可以同时运行多个任务（程序）的话，那么，多线程就是指在同一应用程序（进程）中我们也可以有多个任务（顺序流）同时执行。多线程的优点是具有更好的交互性以及实时行为。

1.4 Java 程序的类型及其不同的编程模式

用 Java 书写的程序有两种类型：Java 应用程序 （Java Application）和 Java 小应用程序（Java Applet）。

Java 应用程序必须得到 Java 虚拟机的支持才能够运行。Java 小应用程序则需要客户端浏览器

的支持。Java 小应用程序运行之前必须先将其嵌入 HTML 文件的<applet> 和</applet>标记中。当用户浏览该 HTML 页面时，Java 小应用程序将从服务器端下载到客户端，进而被执行。

Application 的基本编程模式：

```
class 用户自定义的类名  // 定义类
{
    public static void main(String args[ ] )   //定义 main( )方法
     {
        方法体
      }
}
```

Applet 的基本编程模式：

```
import java.awt.Graphics;  //引入 java.awt 系统包中的 Graphics 类
import java.applet.Applet;  //引入 java.applet 系统包中的 Applet 类
class  用户自定义的类名  extends Applet  //定义类
{
  public void paint(Graphics g)  //调用 Applet 类的 paint( ) 方法
   {
     方法体
    }
}
```

Applet 需要的 HTML 文件的最小集的格式为：

```
<HTML>
<applet code=类名.class   width= 宽度   height=高度>
</THML>
```

HTML 标记包含在尖括号内，并且总是成对出现，前面加斜杠表明标记结束。用<HTML>和</THML>来标记 HTML 文件的开始和结束，用<applet>和</applet>标记 applet 的开始和结束。必须把以.class 结尾的字节码文件名嵌入到 HTML 文件中。HTML 文件应和字节码文件放在同一目录下。另外，HTML 对字符大小写是不敏感的，参数值可加引号也可不加。

综上所述，Applet 和 Application 是 Java 程序的两种基本类型，从源代码的角度来看，Applet 和 Application 有两个基本的不同点。

① 一个 Applet 类必须定义一个从 Applet 类派生的类，Application 则没有这个必要。

② 一个 Application 必须定义一个包含 main 的方法，以控制它的执行，即程序的入口。而 Applet 不会用到 main 方法，它的执行是由 Applet 类中的几个系统方法来控制的。两者共同之处是：编程语法是完全一样的。

1.5　Java 程序开发过程

1. 开发过程简介

要编写和运行第一个 Java 程序，需要有文本编辑器和 Java 开发平台。可以使用 DOS 操作系统提供的 Edit 或记事本作为文本编辑器，用 J2SDK(Java 2 Software Development Kit) 作为开发平台，J2SDK 往往习惯称为 JDK。也可以在安装好 JDK 之后，下载 TextPad 作为编辑和运行平台。通常，初学者使用 Windows 环境中的记事本作为创建源文件的文件编辑器。

要创建一个 Java 需要 3 个基本步骤。

① 创建带有文件扩展名.java 的源文件。

② 利用 Java 编译器生成文件扩展名为.class 的字节码文件。

③ Application 程序利用 Java 解释器运行该字节码文件，Applet 利用 Java 自带查看器或浏览器运行嵌有字节码文件的 HTML 文件。

保存文件时一定要使用 public 的类名作为文件名，用.java 作为后缀。记事本默认的扩展名是.txt，所以必须修改文件扩展名为.java，可在文件名的开始和扩展名的结尾处加上一对双引号后保存；或者不加双引号，保存类型选择 all files。

Java 编译器是 JDK 中的 javac.exe，将 Java 源程序编译成字节码文件，语法如下：

```
javac  类名.java
```

然后按 Enter 键。如果源程序没有错误，则屏幕上没有输出，否则将显示出错信息。

Java 解释器是 JDK 中的 java.exe，解释和执行 Java 应用程序使用语法：

```
java  类名
```

然后按 Enter 键。Java 的平台无关性就是因为每一种计算机上都安装了一个合适的解释器，将不同计算机上的系统差别隐藏起来，使字节码面对一个相同的运行环境，实现了“编写一次，到处运行”的目标。

随着 JDK 版本的升级，编译器 javac 后边的类名可以不是主类的名称，但是解释执行的时候必须是：主类.class，因为文件名可以是任一类名，但是生成的 class 文件，解释从主类开始。

对于 Applet 程序来说，需要 HTML 文件的配合，使用语法：

```
appletviewer  HTML 文件名.html
```

然后按 Enter 键。字节码文件嵌入 HTML 文件中， appletviewer 为 Applet 查看器（JDK 中的 appletviewer.exe），含有内置 Java 解释器。appletviewer 又称小浏览器，它仅显示相关 Applet 的属性，初学者使用方便。

2. 创建 Java Application 程序示例

编写一个 Java Application 程序，其过程简要描述如下。

首先，用户需要下载和安装 J2SDK（JDK）。这里以 JDK 1_5_0_06 版本为例。暂且把程序源文件放置在 JDK 的 bin 目录之下自己创建的 code 文件夹中。

其次，确定文本编辑器。在本例中使用记事本，以 Windows 2000/XP 为例，从“开始”菜单项中选择“程序”→“附件”→“记事本”。当然，用户也可以选择其他文本编辑器。

【例 1.1】 实现第一个简单的应用程序：打印一行文字。

（1） 在“记事本”中编写如下源程序

```
// 文件名: Welcome.java
public class Welcome {
    public static void main( String args[] )
    {
        System.out.println( "Welcome to Java Programming!" );
    } //结束 main 方法的定义
} //结束类 Welcome 的定义
```

（2）语法说明

程序中的“//”为单行注释符，只对当前行有效，表示该行是注释行。程序人员在程序中加入注释，用于提高程序的可读性，使程序便于阅读和理解。程序执行时注释行会被 Java 编译器忽略。多行注释用“/*”开始，以“*/”结束。

Java 程序由类或类的定义组成，类构成了 Java 程序的基本单元。创建一个类是 Java 程序的首要工作。Java 用关键字 class 标志一个类定义的开始，class 前面的 public 关键字代表该类的访问属性是公共的，表示这个类在所有场合中可使用。一个程序文件中可以声明多个类，但仅允许有一个公共的类。class 后面是该类的类名，在本例中是 Welcome。

Application 中有一个显著标记就是必须定义一个 main()主方法，而且应该按照例 1.1 中所示的那样来定义其修饰符和命令行参数，用关键字说明它是 public，静态的 static，无返回值的 void，主方法的参数是字符串类型 String 的数组 args[]。一个类中可以声明多个方法，Java 应用程序自动从 main 主方法开始运行，通过主方法再调用其他的方法。Java 语言的每条语句都必须用分号结束。

System.out 是标准输出对象，它用于在 Java 应用程序执行的过程中向命令窗口显示字符串和其他类型的信息。方法 System.out.println 在命令窗口中显示一行文字后，会自动将光标位置移到下一行（与在文本编辑器中按 Enter 键类似）。

（3）编译运行程序

源程序编写并保存好之后，接下来准备执行该程序。为此，打开一个命令提示符窗口，用 cd.. 退到根目录，如图 1.4 所示。

图 1.4　“命令提示符”窗口

接着，进入程序所存储的目录（假定应用程序存放在 C:\Program Files\Java\jdk1.5.0_06\bin\code 下），在“命令提示窗口”中键入 dir Welcome.java 命令，显示文件，如图 1.5 所示。

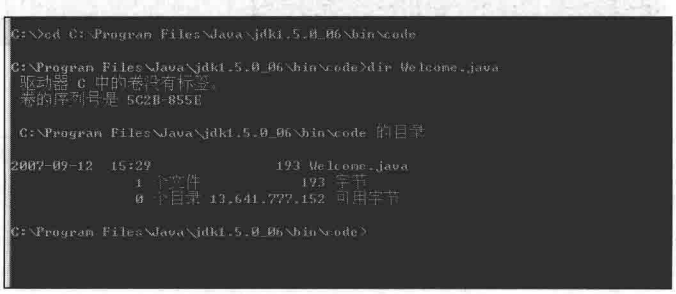

图 1.5　显示 Welcome.java 文件

再在“命令提示符”窗口中键入 javac Welcome.java，如图 1.6 所示。

如果此程序不含语法错误提示，那么，将生成一个 Welcome.class 文件，自动保存在源文件同级目录下，此文件含有表示该程序的 Java 字节码。