

21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

Java面向对象程序设计

Object-Oriented Programming with Java

张亦辉 冯华 胡洁 编著

- 概念与实例相结合，讲解面向对象思想
- 基础与应用相结合，掌握Java基础知识
- 开发与实践相结合，培养程序开发能力



精品系列



人民邮电出版社
POSTS & TELECOM PRESS

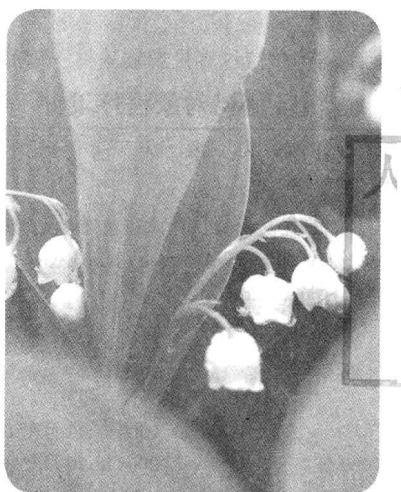
21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

Java面向对象 程序设计

Object-Oriented Programming with Java

张亦辉 冯华 胡洁 编著



精品系列

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 / 张亦辉, 冯华, 胡洁编著.
北京: 人民邮电出版社, 2008.10
21 世纪高等学校计算机规划教材. 精品系列
ISBN 978-7-115-17914-2

I. J… II. ①张…②冯…③胡… 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 (2008) 第 046568 号

内 容 提 要

本书根据 Java 语言程序面向对象的本质特征以及面向对象程序设计课程的基本教学要求, 在详细阐述面向对象程序设计基本理论和方法的基础上, 详细介绍了 Java 语言及其面向对象的基本特性、基本技术。全书共分为 10 章, 首先介绍了面向对象程序设计、Java 语言的基础知识, 而后详细讲述 Java 语言中面向对象思想的实现以及使用, 最后介绍了 Java 图形用户界面、Applet、数据库等相关知识。

书中采用大量的实例进行讲解, 力求通过实例使读者更形象地理解面向对象思想, 快速掌握 Java 编程技术。本书难度适中, 内容由浅入深, 使用性强, 覆盖面广, 条理清晰。每章附有精心编写的实验和习题, 便于读者实践和巩固所学知识。本书可作为普通高等院校 Java 程序设计课程的教材, 也可作为读者的自学用书。

21 世纪高等学校计算机规划教材—精品系列

Java 面向对象程序设计

-
- ◆ 编 著 张亦辉 冯 华 胡 洁
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 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京昌平百善印刷厂印刷
 - ◆ 开本: 787×1092 1/16
印张: 22
字数: 580 千字 2008 年 10 月第 1 版
印数: 1—3 000 册 2008 年 10 月北京第 1 次印刷

ISBN 978-7-115-17914-2/TP

定价: 36.00 元

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

出版者的话

计算机应用能力已经成为社会各行业最重要的工作要求之一，而计算机教材质量的好坏会直接影响人才素质的培养。目前，计算机教材出版市场百花争艳，品种急剧增多，要从林林总总的教材中挑选一本适合课程设置要求、满足教学实际需要的教材，难度越来越大。

人民邮电出版社作为一家以计算机、通信、电子信息类图书与教材出版为主的科技教育类出版社，在计算机教材领域已经出版了多套计算机系列教材。在各套系列教材中涌现出了一批被广大一线授课教师选用、深受广大师生好评的优秀教材。老师们希望我社能有更多的优秀教材集中地呈现在老师和读者面前，为此我社组织了这套“21世纪高等学校计算机规划教材——精品系列”。

“21世纪高等学校计算机规划教材——精品系列”具有下列特点。

(1) 前期调研充分，适合实际教学需要。本套教材主要面向普通本科院校的学生编写，在内容深度、系统结构、案例选择、编写方法等方面进行了深入细致的调研，目的是在教材编写之前充分了解实际教学的需要。

(2) 编写目标明确，读者对象针对性强。每一本教材在编写之前都明确了该教材的读者对象和适用范围，即明确面向的读者是计算机专业、非计算机理工类专业还是文科类专业的学生，尽量符合目前普通高等教学计算机课程的教学计划、教学大纲以及发展趋势。

(3) 精选作者，保证质量。本套教材的作者，既有来自院校的一线授课老师，也有来自IT企业、科研机构等单位的资深技术人员。通过他们的合作使老师丰富的实际教学经验与技术人员丰富的实践工程经验相融合，为广大师生编写出适合目前教学实际需求、满足学校新时期人才培养模式的高质量教材。

(4) 一纲多本，适应面宽。在本套教材中，我们根据目前教学的实际情况，做到“一纲多本”，即根据院校已学课程和后续课程的不同开设情况，为同一科目提供不同类型的教材。

(5) 突出能力培养，适应人才市场要求。本套教材贴近市场对于计算机人才的能力要求，注重理论技术与实际应用的结合，注重实际操作和实践动手能力的培养，为学生快速适应企业实际需求做好准备。

(6) 配套服务完善，共促提高。对于每一本教材，我们在教材出版的同时，都将提供完备的PPT课件，并根据需要提供书中的源程序代码、习题答案、教学大纲等内容，部分教材还将在作者的配合下，提供疑难解答、教学交流等服务。

在本套教材的策划组织过程中，我们获得了来自清华大学、北京大学、人民大学、浙江大学、吉林大学、武汉大学、哈尔滨工业大学、东南大学、四川大学、上海交通大学、西安交通大学、电子科技大学、西安电子科技大学、北京邮电大学、北京林业大学等院校老师的大力支持和帮助，同时获得了来自信息产业部电信研究院、联想、华为、中兴、同方、爱立信、摩托罗拉等企业和科研单位的领导和技术人员的积极配合。在此，人民邮电出版社向他们表示衷心的感谢。

我们相信，“21世纪高等学校计算机规划教材——精品系列”一定能够为我国高等院校计算机课程教学做出应有的贡献。同时，对于工作欠缺和不妥之处，欢迎老师和读者提出宝贵的意见和建议。

前言

面向对象程序设计已经成为软件编程技术中一项非常关键的技术。相比过程化程序设计技术,面向对象程序设计中的继承、封装、多态等特性更接近于人的语言和思维,从而更容易理解和使用。与此同时,面向对象程序设计更加符合现代软件大规模开发的需求,有利于软件复用。

Java 语言是面向对象程序设计语言中的代表,相比 C++,Java 语言更全面地体现了面向对象的思想。Java 语言诞生于 1995 年,短短数年,Java 语言已经遍布软件编程的各个领域。随着 Internet 的飞速发展,Web 得到广泛的应用,而 Java 语言在 Web 应用方面所表现出的强大特性,使得 Java 语言成为 Web 开发的主流技术。

由于 Java 语言具有简单易学、面向对象、使用范围广等特征,因此,非常适合作为普通高等院校程序设计课程,尤其是面向对象程序设计课程。本书采用循序渐进、由浅入深、概念与例子相结合的编写方式,对内容的安排、例程的选择、习题的编写都进行了严格控制,确保难度适中,更贴近于实用。

在学习本教材之前,读者应具有基本的计算机操作基础,但不必具有编程基础。掌握一门语言最好的方式就是实践,本书的着眼点是将基础的理论知识讲解和实践应用相结合,使读者在理解面向对象的思想,快速掌握 Java 编程技术。

全书共分 10 章,在大多数章节中,首先对相关的基础知识进行介绍,然后重点讲解相关的实例。其中,第 1 章对面向对象程序设计和 Java 语言进行简要介绍。第 2 章介绍了 Java 语言的基本语法。第 3 章~第 5 章是本书的重点,详细讲述 Java 语言的面向对象特性,包括 Java 语言中类、对象、继承、多态、接口和内部类等重要概念及其应用实践。第 6 章介绍 Java 中的输入输出以及异常机制。第 7 章讲述如何利用 Java 编写图形用户界面。第 8 章讲述 Applet 的使用,包含如何编写 Applet 以及如何在浏览器中运行 Applet。第 9 章在简要讲述 TCP/IP、UDP、Socket 协议的基础上,介绍如何利用 Java 语言编写网络应用。第 10 章为 Java 的高级应用,包含 java 的多线程技术、JSP 和 Servlet 及数据库技术。

本书在每章之后附有习题和上机指导,供读者练习实践以检验学习效果。

本书由张亦辉、冯华和胡洁共同编写完成。同时参与本书修改、整理以及例程编写调试的还有陈晨、郭军军、符滔滔、王征、张军、罗栋、宋昕、王妍、王建、徐宁、郭瑞、杨晓强等。在此,编者对以上人员致以诚挚的谢意!本书中所有例程和相关代码已调试通过,并根据本书内容制作了电子课件,供老师教学时参考使用。

最后感谢读者选择本书,由于时间仓促和作者的水平有限,书中错误和不妥之处在所难免,敬请批评指正。

编 者

2008 年 4 月

目 录

第 1 章 Java 语言概述	1
1.1 面向对象程序设计	1
1.1.1 面向对象程序设计思想的诞生	1
1.1.2 面向对象与面向过程的对比	2
1.1.3 面向对象技术的背景和特点	5
1.2 Java 概述	5
1.2.1 Java 的起源和发展	6
1.2.2 Java 特性	6
1.2.3 Java 体系结构	7
1.3 Java 运行机制与 JVM	7
1.3.1 JVM 的体系结构	8
1.3.2 JVM 的运行机制	8
1.4 Java 类库	9
1.5 安装 Java 开发工具	11
1.5.1 下载 JDK	11
1.5.2 安装 JDK	12
1.5.3 设置 Java 运行环境	14
1.6 使用命令行	15
1.7 使用集成开发环境	17
1.7.1 使用 JCreator	17
1.7.2 使用 Eclipse	19
1.8 第一个 Java 程序：整数相加	23
1.8.1 开发源代码	24
1.8.2 编译运行	24
小结	25
习题	25
上机指导	25
实验一：编译 Java 程序	25
第 2 章 Java 语言基础	27
2.1 数据类型	27
2.1.1 整型	27
2.1.2 浮点型	28
2.1.3 char 型	29
2.1.4 boolean 型	29
2.1.5 基本数据类型值间的转换	30
2.2 变量	31
2.2.1 变量声明	31
2.2.2 变量名和变量类型	32
2.2.3 变量的初始化	32
2.2.4 final 变量	32
2.3 运算符	33
2.3.1 算术运算符	33
2.3.2 关系和逻辑运算符	35
2.3.3 位运算符	36
2.3.4 赋值运算符	36
2.3.5 其他运算符	37
2.4 表达式和语句	38
2.4.1 表达式	38
2.4.2 语句	39
2.5 控制结构	40
2.5.1 条件语句	40
2.5.2 循环语句	43
2.5.3 跳转语句	45
2.6 字符串	46
2.6.1 String 类型	46
2.6.2 StringBuffer 类型	50
2.7 数组	53
2.7.1 数组的声明与创建	53
2.7.2 数组的初始化	53
2.7.3 数组的常用操作	56
2.8 命名规范	59
2.8.1 标识符命名规则	59
2.8.2 Java 中提倡的命名习惯	60
2.9 注释	60
2.9.1 单行注释	60
2.9.2 区域注释	60
2.9.3 文档注释	61
小结	62
习题	62
上机指导	63

实验一：基本数据类型的定义及转换·····	63	3.10.1 Date 类·····	105
实验二：使用程序控制结构·····	63	3.10.2 GregorianCalendar 类·····	107
实验三：String 的使用·····	64	小结·····	110
实验四：数组的使用·····	65	习题·····	110
第 3 章 类与对象 ·····	66	上机指导·····	110
3.1 面向对象程序设计概述·····	66	实验一：类的定义·····	111
3.1.1 面向对象术语·····	66	实验二：成员变量的使用·····	111
3.1.2 面向对象程序设计方法的优点·····	67	实验三：编写更复杂的类·····	111
3.2 面向对象与 UML 建模·····	68	实验四：静态成员的创建·····	112
3.2.1 为什么需要建模·····	68	第 4 章 继承与多态 ·····	114
3.2.2 UML 建模语言·····	68	4.1 继承概述·····	114
3.2.3 UML 的面向对象分析设计·····	69	4.1.1 超类、子类·····	114
3.3 Java 语言与面向对象特性·····	70	4.1.2 继承层次·····	114
3.4 类的定义和对象的创建·····	71	4.2 Java 中的继承·····	115
3.4.1 类的基本结构·····	71	4.2.1 派生子类·····	115
3.4.2 类之间的关系·····	71	4.2.2 继承规则·····	116
3.4.3 构造函数·····	74	4.2.3 方法的继承与覆盖·····	119
3.4.4 类成员·····	76	4.2.4 this 与 super·····	121
3.4.5 对象的创建·····	79	4.3 强制类型转换·····	124
3.5 方法·····	79	4.4 动态绑定·····	127
3.5.1 方法的定义·····	79	4.5 终止继承：Final 类和 Final 方法·····	128
3.5.2 方法的重载·····	81	4.5.1 Final 类·····	128
3.5.3 递归·····	85	4.5.2 Final 方法·····	129
3.6 静态成员·····	87	4.6 抽象类·····	130
3.6.1 静态方法和静态变量·····	87	4.6.1 抽象类·····	131
3.6.2 静态变量和常量·····	88	4.6.2 抽象的方法·····	132
3.6.3 静态成员的访问·····	89	4.7 多态·····	134
3.6.4 main()方法·····	92	4.8 所有类的超类：Object 类·····	136
3.6.5 Factory 方法·····	93	小结·····	139
3.7 包·····	95	习题·····	139
3.7.1 包的定义·····	95	上机指导·····	140
3.7.2 类的导入·····	96	实验一：抽象类的定义及调用·····	140
3.7.3 静态导入·····	98	实验二：使用多态·····	141
3.8 成员的访问控制·····	99	实验三：使用 Object 类·····	141
3.8.1 公共类型：public·····	100	实验四：构造函数的继承·····	142
3.8.2 私有类型：private·····	100	实验五：对象引用的多态·····	143
3.8.3 默认类型：default·····	101	第 5 章 接口与内部类 ·····	145
3.8.4 保护类型：protected·····	102	5.1 接口的特性·····	145
3.9 封装·····	102	5.2 接口的定义·····	146
3.10 利用系统已有的类·····	105		

5.3 接口的使用	147	6.5.2 异常的层次结构	199
5.3.1 接口实现的基本语法	147	6.5.3 自定义异常	202
5.3.2 接口中方法的实现与使用	147	小结	205
5.4 接口与抽象类	149	习题	205
5.5 接口与回调	151	上机指导	205
5.6 内部类	152	实验一: I/O 流的使用	206
5.6.1 内部类概述	153	实验二: 使用异常处理	206
5.6.2 内部类语法规则	153	实验三: 处理流的使用	206
5.6.3 局部内部类	156	实验四: 自定义异常处理	208
5.6.4 匿名内部类	159		
5.6.5 静态内部类	160	第 7 章 图形用户界面的实现	209
5.6.6 关于内部类的讨论	162	7.1 图形用户界面概述	209
小结	162	7.2 Swing 与 AWT	210
习题	162	7.2.1 Swing 与 AWT 之间的关系	210
上机指导	163	7.2.2 关于 Swing 与 AWT 控件的混用	211
实验一: 接口的创建	163	7.3 事件处理	212
实验二: 内部类的创建	163	7.3.1 事件的层次结构	213
实验三: 创建多个接口	164	7.3.2 窗体事件	214
实验四: 接口和继承的混合使用	165	7.3.3 鼠标事件	214
第 6 章 输入/输出和异常处理	166	7.3.4 事件适配器	216
6.1 I/O 流	166	7.4 创建图形用户界面	216
6.1.1 流的层次	166	7.4.1 窗体	216
6.1.2 输入流和输出流	167	7.4.2 面板	218
6.1.3 字节流和字符流	170	7.4.3 标签	218
6.1.4 随机存取文件流	173	7.4.4 按钮	221
6.2 I/O 流的使用	174	7.5 布局管理	223
6.2.1 标准的 I/O 流	174	7.5.1 流布局	223
6.2.2 基本的 I/O 流	180	7.5.2 网格布局	225
6.2.3 过滤流	182	7.5.3 卡片布局	227
6.2.4 文件随机读写	183	7.6 选择控件	229
6.2.5 流的分割	185	7.6.1 控件概述	229
6.3 对象的序列化	185	7.6.2 文本框	230
6.3.1 存储对象	186	7.6.3 文本区	233
6.3.2 对象的序列化	186	7.6.4 单选按钮、复选框	236
6.3.3 对象序列化中的一些问题	187	7.7 菜单和工具栏	239
6.4 文件管理	188	7.7.1 菜单	239
6.4.1 File 类简介	188	7.7.2 工具栏	243
6.4.2 使用 File 类	190	7.8 对话框	245
6.5 异常处理	191	7.9 图形文本绘制	248
6.5.1 异常处理概述	191	7.9.1 画布	248
		7.9.2 画笔	249

7.9.3 文本	251	9.1.1 TCP/IP、UDP	295
7.9.4 字体	252	9.1.2 Socket 套接字	296
7.10 图像处理	255	9.2 Java 网络通信机制	297
7.11 综合示例: 围棋程序	258	9.3 URL 通信	298
小结	268	9.3.1 URL 的创建	299
习题	268	9.3.2 解析 URL	299
上机指导	268	9.3.3 获取数据	300
实验一: 使用按钮	268	9.4 InetAddress 类	301
实验二: 使用 Graphics 类绘图	269	9.5 Socket 套接字	303
实验三: 用户注册界面	269	9.5.1 ServerSocket 类	303
实验四: 编写计算器程序	271	9.5.2 Socket 类	304
第 8 章 Applet 应用程序	274	9.5.3 组播套接字	307
8.1 Applet 基础	274	9.6 综合示例: 聊天室程序	310
8.1.1 查看 Applet	274	小结	315
8.1.2 Applet 与浏览器	275	习题	315
8.1.3 显示 Applet	276	上机指导	316
8.1.4 Applet 生命周期	276	实验一: 创建 URL 连接	316
8.2 Applet 类 API	277	实验二: 获得 URL 中的数据	316
8.3 Applet 的 HTML 标记和属性	278	第 10 章 高级应用	317
8.3.1 定位属性	279	10.1 线程	317
8.3.2 编码属性	279	10.1.1 Java 中的线程模型	317
8.4 创建 Applet	280	10.1.2 线程的创建	319
8.4.1 简单 Applet	280	10.1.3 线程的同步	321
8.4.2 向 Applet 传递参数	282	10.1.4 线程的调度	323
8.5 Applet 与 Application	284	10.1.5 线程的其他方法	325
8.6 Applet 弹出窗口	287	10.2 Servlet 和 JSP 技术	328
8.7 Applet 安全	288	10.2.1 JSP 概述	328
8.7.1 Applet 安全控制	288	10.2.2 JSP 语法	329
8.7.2 Applet 沙箱	289	10.2.3 JSP 与 JavaBean	332
8.8 实例研究: 显示动画	289	10.2.4 Servlet 技术	334
8.8.1 动画原理及重新绘制	290	10.3 数据库技术	337
8.8.2 Timer 类简介	290	10.3.1 SQL 基础	337
小结	292	10.3.2 JDBC 层次结构	338
习题	292	10.3.3 加载数据库驱动	340
上机指导	293	10.3.4 基本数据库访问	341
实验一: 创建 Applet	293	小结	343
实验二: 在 Applet 中显示图像界面	293	习题	343
实验三: 显示 Applet 传递的参数	294	上机指导	343
第 9 章 网络通信	295	实验一: 创建多线程	343
9.1 网络通信概述	295	实验二: 使用 JSP	344

第 1 章

Java 语言概述

Java 是 Sun Microsystem 公司研究开发的一种新型的程序设计语言。在众多高级语言中，Java 语言脱颖而出。它不仅成为最为流行的计算机语言之一，而且形成一种专门的技术。

1.1 面向对象程序设计

面向过程和面向对象是两种主要的程序设计理念。面向过程是早期程序设计的主要方式，近些年来，面向对象逐渐成为程序设计的主要方式。面向对象的编程思想由来已久，但真正意义上的纯面向对象编程语言目前只有 Java。

本节将对面向对象的基础知识进行简单的介绍，主要包括面向对象程序设计思想的诞生、面向对象与面向过程程序设计思想的对比、面向对象技术的背景和特点等三方面的内容。

1.1.1 面向对象程序设计思想的诞生

随着软件复杂度的提高，以及 Internet 的迅猛发展，原先面向过程的软件开发方式已经很难满足软件开发的需要。针对日趋复杂的软件需求挑战，面向对象的软件开发模式诞生了。目前作为针对软件危机的最佳对策，面向对象（Object Oriented，OO）技术已经引起人们的普遍关注。许多编程语言都推出了面向对象的新版本，一些软件开发合同甚至也指明了必须使用基于 OO 的技术和语言。下面简要列出了 OO 技术的发展历程。

（1）诸如“对象”和“对象的属性”这样的概念，可以一直追溯到 20 世纪 50 年代初，首先出现于关于人工智能的早期著作中。然而，OO 的实际发展开始于 1966 年，当时开发了具有当时更高级抽象机制的 Simula 语言。

（2）Simula 语言提供了比子程序更高一级的抽象和封装，并且为仿真一个实际问题，引入了数据抽象和类的概念。后来一些科学家吸取了 Simula 类的概念，开发出了 Smalltalk 语言。

（3）几乎在同时，“面向对象”这一术语被正式确定。在 Smalltalk 中一切都是对象——即某个类的实例。最初的 Smalltalk 世界中，对象与名词紧紧相连。

（4）Smalltalk 语言还影响了 20 世纪 80 年代早期和中期的很多面向对象语言，如 Objective-C（1986 年）、C++（1986 年）、Self（1987 年）、Eiffel（1987 年）、Flavors（1986 年）。同时，面向对象的应用领域也被进一步拓宽，对象不再仅仅与名词相联系，还包括事件和过程。

（5）到了 20 世纪 90 年代，随着 Internet 的迅猛发展，Sun 公司于 1995 年推出了纯面向对象的 Java 语言，自此之后，OO 技术在开发中越来越占主导地位。



Smalltalk 被认为是第一个真正面向对象的语言, 现在国外还有一些开发人员在使用, 对该语言感兴趣的同学可以参阅其他相关资料做更详细的了解。

1.1.2 面向对象与面向过程的对比

传统的过程化程序设计通过设计一系列的过程 (即算法) 来求解问题。这些过程一旦被确定, 下一步就要开始寻找存储数据的方式, 即“程序=算法+数据结构”。其中, 算法是第一位, 而数据结构是第二位。而面向对象的程序设计 (Object Oriented Programming, OOP) 调换了这个次序, 将数据放在第一位, 然后再考虑操纵数据的算法。

在 OOP 中, 程序被看作是相互协作的对象集合, 每个对象都是某个类的实例, 所有类构成一个通过继承关系相联系的层次结构。面向对象的语言通常具有以下特征。

- 对象生成功能
- 消息传递机制
- 类和遗传机制

这些概念可以并且也已经其他编程语言中单独出现, 但只有在面向对象语言中, 它们才共同出现, 并以一种独特的合作方式互相协作, 互相补充。实际上, 软件开发的过程就是人们使用各种计算机语言将自身关心的现实世界 (问题域) 映射到计算机世界的过程, 这个过程通常如图 1-1 所示。

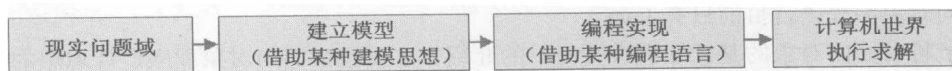


图 1-1 问题域映射过程

面向对象方法的软件开发主要经历如下 3 个阶段。

- 面向对象的系统分析 (Object-Oriented System Analysis, OOA)。系统分析的主要任务是通过用户需求进行分析确定系统的整体功能, 即系统要做什么。
- 面向对象的系统设计 (Object-Oriented System Design, OOD)。系统设计的核心是确定系统应怎样做。
- 面向对象的编程 (Object-Oriented Programming, OOP)。

OOA 强调直接针对要开发的系统、客观存在的各种事物建立 OOA 模型。系统中有哪些值得考虑的事物, OOA 模型就有哪些对象; 即客观世界与面向对象存在着——对应的映射关系。面向对象分析方法用属性描述事物的静态 (状态) 特性; 用方法描述事物的动态行为。OOA 分析方法的核心思想是利用面向对象的概念和方法为软件分析建造模型, 从而将用户需求逐步细化、完整、精确。OOA 分析方法的大致步骤为: 识别对象、属性及外部服务, 识别类及其结构, 定义对象之间的消息传递等。

OOD 是对 OOA 产生的结果增添实际的计算机系统中所需的细节, 如人机交互、任务管理和数据管理的细节。从 OOA 到 OOD 是一个模型扩充过程。OOD 包括两种: 一是将 OOA 模型直接引入而不必转换, 只作细节修正与补充; 二是针对具体实现中的人机界面、数据存储、任务管理等因素运用面向对象的方法进行模型扩充。OOD 可以分为四部分: 问题空间部分的设计 (Problem Domain Component, PDC)、人机交互部分的设计 (Human Interface Component, HIC)、任务管理的设计 (Task Management Component, TMC)、数据部分的设计 (Data Management

Component, DMC)。

OOP 是与面向过程编程完全相反的思考模式,利用面向对象的特性,把问题中的所有数据及操作过程,一一封装成独立的对象,而对象之间的关系即是对象彼此之间如何传递消息(Message)。就像日常生活中每个人之间的关系,如师生关系、亲子关系、朋友关系等。即,面向对象是以人类观察世界的思维方式。OOP 的目标在于创建软件重用代码,具备更好地模拟现实世界环境的能力。

本小节将分别介绍面向对象与面向过程的编程模式,进而帮助读者更加清楚地了解两者之间的区别。

1. 面向过程的编程模式

在这种编程模式中,数据和函数(过程)是分开的,即开发人员看到的是函数或过程的集合以及单独的一批数据。程序的处理过程为:参数输入→函数/过程代码→结果输出,其编程模式如图 1-2 所示。

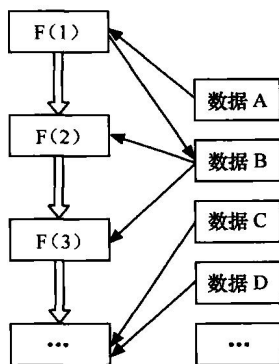


图 1-2 面向过程的编程模式

图 1-2 中的“F(1)”、“F(2)”、“F(3)”表示函数(过程),细线箭头表示函数(过程)对数据的访问。

从图 1-2 可以看出,对于软件维护人员来说,无论是函数还是数据结构的改动,都会使整个程序受到干扰,进而可能引发软件系统的崩溃,可以说是“牵一发而动全身”,程序的维护和扩展几乎难以进行。

通过前面的分析还可以看出,在面向过程的编程模式中,开发人员分析了问题之后,得到一个面向过程的模型,其过程如图 1-3 所示。

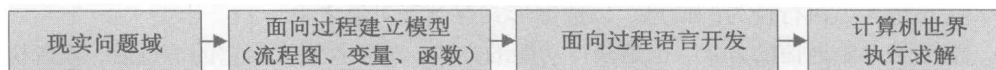


图 1-3 面向过程的基本开发过程

2. 面向对象的编程模式

在这种模式中,函数和它需要存取的数据封装在称为对象的单元中,对象之间的数据访问是间接的,是通过接口进行的。这里所说的接口是指为其他代码调用提供的一套访问方法,即代码的 API,其编程模式如图 1-4 所示。

- 图 1-4 中每一个圆表示一个对象,细线箭头表示对象间的消息(调用)。

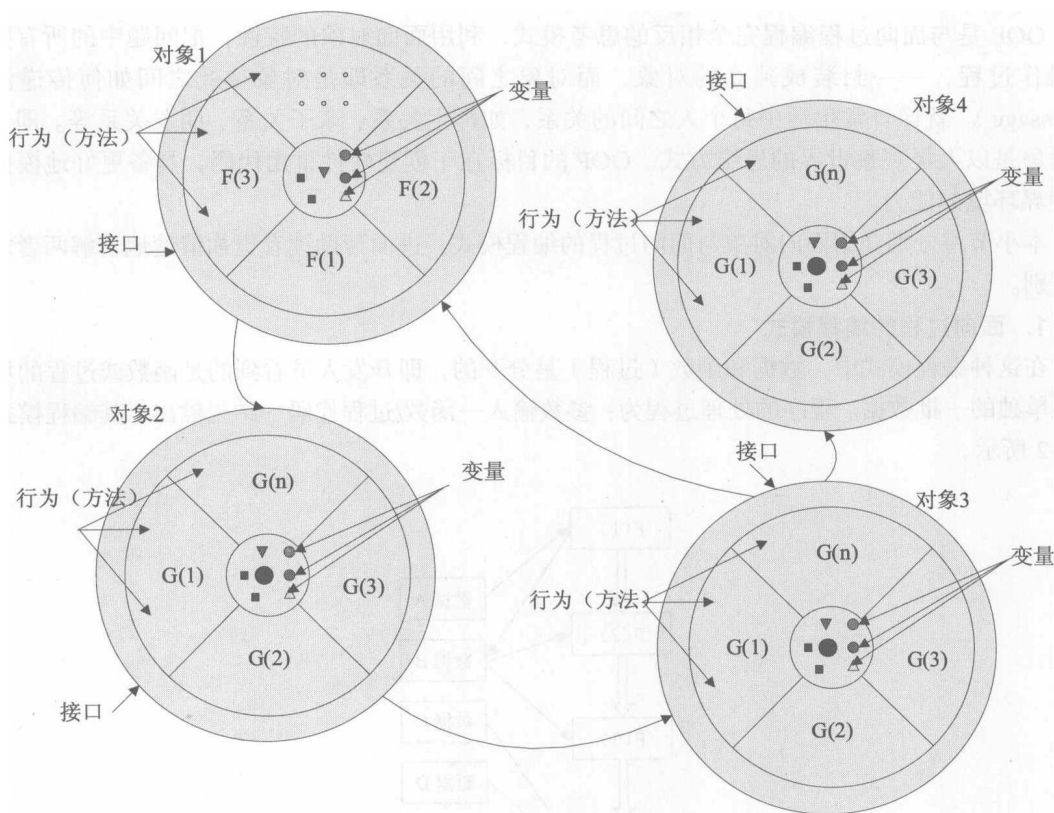


图 1-4 编程模式

● 通常将对象比作鸡蛋，蛋黄是数据，蛋清是访问数据的函数，蛋壳代表接口（即那些公开或公布的方法和属性）。

● 蛋壳接口隐匿了函数和数据结构的实现，当数据结构和内部函数变化时，这种变化被限制在内部的局部范围内。由于接口的相对稳定性，使得这种内部变化的影响不会波及到其他对象，除非蛋壳破裂（即接口发生变化）。

从上面的分析可以看出，通过使用面向对象的开发模式，可以解决面向过程中“牵一发而动全身”的问题，大大提高了程序的灵活性和可维护性。

在面向对象的编程模式中，程序的功能是通过对象间的通信获得的。对象被定义为一个封装了状态（数据）和行为（操作）的实体。

- 状态包含了执行行为的信息，以数据形式存储于对象之中。
- 消息是对象通信的方式，也是获得功能的方式。对象收到发给它的消息后，或者执行一个内部操作，或者再去调用其他对象的操作。

在面向对象编程模式中，开发人员先得到一个面向对象的模型，其中常见的词语是类、对象、方法、消息等，其基本过程如图 1-5 所示。

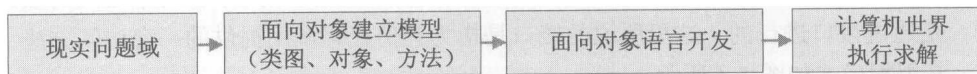


图 1-5 面向对象的基本开发过程

从上面的对比可以看出，面向过程更接近于计算机世界的物理实现，而面向对象的思想则更符合人们的认知习惯。面向对象作为一种开发模式，为软件开发的整个过程（从分析设计到编码）提供了一套完整的解决方案。

1.1.3 面向对象技术的背景和特点

OO 是一种方法，一种思想，同时又是一种技术。OO 力求更客观自然地描述现实世界，使分析、设计和实现系统的方法同认识客观世界的过程尽可能一致。

对象是客观世界中的事物在人脑中的映像，这种映像通过对同一类对象的抽象反映成为人的意识，并作为一种概念而存在。例如，当人们认识到一种新的事物——苹果，于是人们的意识当中就形成了苹果的概念。这个概念会一直存在于人们的思维当中，并不会因为这个苹果被吃掉而消失。

这个概念就是现实世界中的事物在人们意识中的抽象。只要这个对象存在于人们的思维意识当中，人们就可以藉此判断同类的东西。下面将详细介绍面向对象技术的背景和特点。

1. 面向对象技术的背景

客观世界是由许多不同种类的对象构成的，每一个对象都有自己的运动规律和内部状态，不同对象之间相互联系、相互作用。例如，一辆自行车具有的状态是轮胎个数、车身颜色、车灯个数、车速等；具有的行为是加速、刹车、减速等。所有东西都是对象，可将对象理解为一种变量，对象保存着数据，并且可对自身进行操作。

面向对象技术是一种从组织结构上模拟客观世界的方法，它从组成客观世界的对象着眼，通过抽象，将对象映射到计算机系统中，又通过模拟对象之间的相互作用、相互联系来模拟客观世界，描述客观世界的运动规律。面向对象技术以基本对象模型为单位，将对象内部处理细节封装在模型内部，并且重视对象模块间的接口联系和对象与外部环境间的联系，能层次清晰地表示对象模型。

2. 面向对象技术的特点

通过前面的讨论，可以看出对象有以下几个特点。

- 某类对象是对现实世界具有共同特性的某类事物的抽象。
- 对象蕴涵许多信息，可以用一组属性来表征。
- 对象内部含有数据和对数据的操作。
- 对象之间是相互关联和作用的。

面向对象技术，正是利用对现实世界中对象的抽象和对象之间相互关联和作用的描述来对现实世界进行模拟，并且使其映射到目标系统中。所以面向对象的主要特点概括为：抽象性、继承性、封装性和多态性，本章后面的小节将详细讲解。

从编程开发的角度来看，所谓对象就是协调数据存储以及作用于数据之上操作的独立实体。用户可以通过定义一个对象集合以及它们之间的相互作用来创建一个面向对象的程序，许多对象协同工作来定义一个完成用户需要的软件系统。

1.2 Java 概述

本节首先对 Java 语言进行简述，包含 Java 的起源和发展，Java 的特性和 Java 的体系结构，使读者对 Java 有一个初步的认识。

1.2.1 Java 的起源和发展

Java 起源于 1994 年,美国 Sun Microsystem 的 Patrick Nawghton、Jame Gosling 和 Mike Sheridan 等人组成的开发小组,开始了代号为 Green 的项目的研制,其目标是研制一种开发家用电器的逻辑控制系统,产品名称为 Oak。1995 年 1 月, Oak 被更名为 Java。这个名字来自于印度尼西亚的一个盛产咖啡的小岛的名字,小岛的中文名叫爪哇。正是因为许多程序设计师从钟爱的热腾腾的香浓咖啡中得到灵感,因而热腾腾的香浓咖啡也就成为 Java 语言的标志。

1996 年初, Sun 公司发布了 Java 1.0,一直到 2006 年 Java 已经发展到了 Java 6.0,不但进一步提升了性能,还加入了对各种脚本语言嵌入式开发的支持等特性,大大提升了 Java 的开发能力和易用性。1995 年 5 月 23 日 Java 正式公布,随着 Java 技术的不断发展,Java 也在各个领域得到了广泛的应用。

1.2.2 Java 特性

随着 Internet 的飞速发展,人们的学习、工作、科研、商业和生活方式随之发生了巨大变化。人们不仅需要具有声音、图像和动画等多媒体信息的 Web 页面,以及实时视频、多用户网络游戏等,而且要求能向用户提供更好的实时交互性,并具有平台无关性。Java 的出现,使人们看到了解决以上问题的希望。Java 之所以能解决这些问题是与其自身的特点分不开的。Java 具有如下特点。

1. 简单性

由于 Java 语言与 C 语言类似,只要学过 C 语言的人,就可以很容易地掌握。而 Java 语言本身编写容易,语法简单,稍有程序设计经验的人,很快就能上手。

2. 平台无关性

Java 执行环境与使用平台无关,可以方便地将 Java 部署到任何不同平台的机器上。同时,Java 的类库封装了不同平台上的实现,为其提供统一的接口,这使得同样的类库可以在不同的平台上使用。这也就意味着用 Java 开发的应用可以“一次开发,随处运行”。

3. 分布式

Java 在网络方面的强大易用是其他任何语言无法比拟的,可以说 Java 是面向网络的语言。通过其提供的类库可以方便地处理各种网络协议,方便地进行传统的套接字网络开发,如 RMI、CORBA、Web 服务等现在流行的网络开发。

4. 健壮性

Java 在编译和运行时,都会对程序可能出现的问题进行检查,并将出错信息报告给程序员。同时,其提供垃圾收集机制来自动管理内存,避免了程序员无心的错误和恶意的攻击。

5. 安全性

在安全性方面,Java 表现得非常出色,从一开始,Java 就被设计为有防范各种病毒、袭击的能力。例如,一切对内存的访问都必须通过对象的实例引用来实现;禁止破坏自己处理空间之外的内存;禁止运行时堆栈溢出;未经授权禁止读写文件等。

6. 浏览器应用

Applet 是只能运行在 Web 浏览器里的小程序。Applet 作为 Web 页面的一部分自动下载,就像图片一样自动下载,程序员只需要创建一个简单的程序,就能让它自动地运行于任何机器,只要这台机器装上了内置有 Java 解释器的浏览器就行了。

1.2.3 Java 体系结构

虽然 Java 常用作开发各种应用程序的编程语言，但是作为编程语言只是 Java 的众多用途之一，而真正形成 Java 众多优点（包括平台独立性）的是其底层架构。完整的 Java 体系结构实际上是由 4 个组件组合而成：Java 编程语言、Java 类文件格式、Java API 和 JVM。

其中，JVM 是 Java Virtual Machine（Java 虚拟机）的缩写，它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来实现的。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。因此，使用 Java 开发时，本质就是用 Java 编程语言编写代码，然后将代码编译为 Java 类文件，接着在 JVM 中执行类文件。

Java API 是预先编写的代码，并按相似主题分成多个包。现在，Java API 主要分为以下 3 大平台。

- Java SE（Java Standard Edition）：该平台中包含核心 Java 类和 GUI 类。
- Java EE（Java Enterprise Edition）：该平台中包含开发 Web 应用程序所需的类和接口，有 Servlet、Java Server Pages 以及 Enterprise JavaBean 类等。
- Java ME（Java Micro Edition）：该平台体现了 Java 的传统优势，为消费类产品提供了一个已优化的运行时环境，用于传呼机、手机或汽车导航系统等。

在 Java SE 5.0 以前，这 3 个平台分别称为 J2SE（Java 2 Standard Edition）、J2EE（Java 2 Enterprise Edition）、J2ME（Java 2 Micro Edition）。

图 1-6 所示为 Java 不同功能模块之间的相互关系，以及它们与应用程序、操作系统之间的关系。

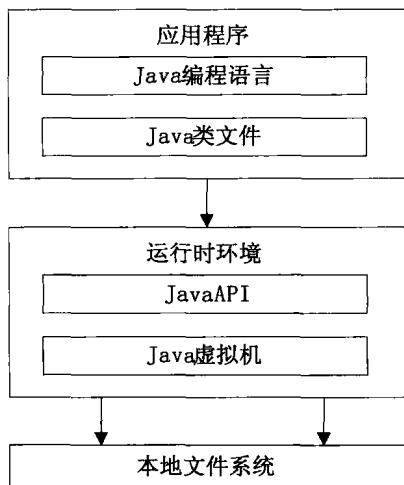


图 1-6 Java 功能模块及模块间关系图

从图 1-6 中可以看出，JVM 与核心类共同构成了 Java 平台，也称为 Java 运行时环境（Java Runtime Environment, JRE）。JRE 是 Java 体系结构的核心，上层是应用程序包括 Java 语言源程序和类文件，底层是异构的操作系统平台。JRE 可以建立在任意操作系统上。

1.3 Java 运行机制与 JVM

在 1.2 节中讲到，在 Java 体系结构中，JVM 处在核心的位置，是程序与底层操作系统和硬件无关的关键。本节将从 JVM 的体系结构和运行过程这两个方面来对 JVM 进行深入介绍。

1.3.1 JVM 的体系结构

Java 语言的一个非常重要的特点就是与平台的无关性，而使用 JVM 是实现这一特点的关键。一般的高级语言如果要在不同的平台上运行，至少需要编译成不同的目标代码。而引入 JVM 后，Java 语言在不同平台上运行时不需要重新编译即可运行。

JVM 屏蔽了与具体平台相关的信息，使得 Java 语言编译程序只需生成在 JVM 上运行的目标代码（字节码），就可以在多种平台上不加修改地运行。JVM 在执行字节码时，把字节码解释成具体平台上的机器指令执行。

在介绍 JVM 体系结构之前首先要明确一个概念，什么是 JVM？JVM 是一个想象中的机器，在实际的计算机上是通过软件模拟来实现。JVM 有自己虚拟的硬件，如处理器、堆栈、寄存器等，还具有相应的指令系统。

每个 JVM 包括方法区、堆、Java 栈、程序计数器和本地方法栈这五个部分，这几个部分和类装载机制与运行引擎机制一起组成的体系结构如图 1-7 所示。

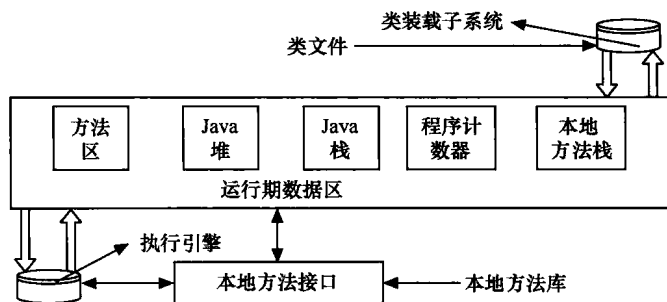


图 1-7 JVM 体系结构

如图 1-7 所示，每一个 JVM 都由一个类装载器子系统（Classloader subsystem）负责加载程序中的类型（类和接口），并赋予唯一的名字，每一个 JVM 都有一个执行引擎（Execution Engine）负责执行被加载类中包含的指令。

程序的执行需要一定的内存空间，存放字节码、被加载类的其他额外信息等。JVM 将这些信息统统保存在数据区（Data areas）中。

数据区中的一部分被整个程序共享，其他部分被单独的线程控制。每一个 JVM 都包含方法区（Method area）和堆（Heap），他们都被整个程序共享。JVM 加载并解析一个类以后，将从类文件中解析出来的信息保存在方法区中。程序执行时创建的对象都保存在堆中。

JVM 不使用寄存器保存计算的中间结果，而是用 Java 堆栈来存放中间结果。这使得 JVM 的指令更紧凑，也更容易在一个没有寄存器的设备上实现 JVM。

JVM 的每个实例都有一个它自己的方法域和堆，运行于 JVM 内的所有线程都共享这些区域。

1.3.2 JVM 的运行机制

JVM 通过调用某个指定类的方法 main 启动，传递给 main 一个字符串数组参数，使指定的类被装载，同时链接该类所使用的其他类型，并且初始化它们。整个过程如图 1-8 所示。

类被装载后，在 main 方法被调用之前，必须对 Java 类与其他类型进行链接然后进行初始化。链接包含 3 个阶段：检验、准备和解析。

（1）检验负责检查被装载的主类的符号和语义。