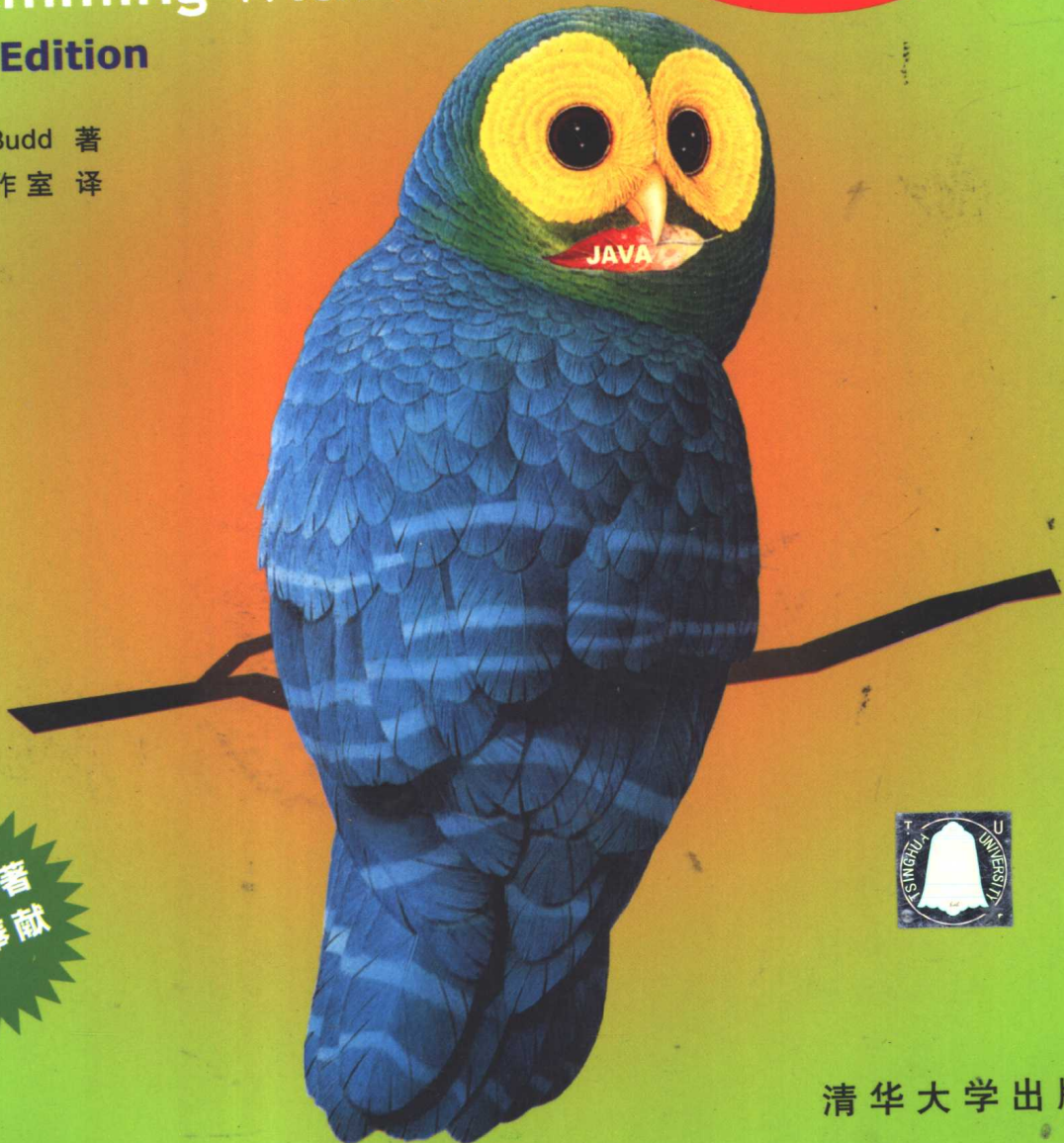


面向对象 (修订版) JAVA 编程思想



Understanding
Object-Oriented
Programming With JAVA
Updated Edition

[美] Timothy Budd 著
三联四方工作室 译



名家名著
最新奉献



清华大学出版社

面向对象 JAVA 编程思想

(修订版)

[美] Timothy Budd 著
三联四方工作室 译

清华大学出版社

(京) 新登字 158 号

内 容 简 介

名家名著最新版, 指导您深入领会面向对象 Java 编程的精髓。示例教学, 使您深入浅出地了解 Java 2 工作原理。本书面向有经验的程序员, 提升其面向对象编程技能, 解释 Java 语言的工作原理。本书是理解 Java 和面向对象编程基本原理的高级工具书, 适用于所有编程人员, 亦可作为高校相关专业参考书。

Understanding Object-Oriented Programming with JAVA, Updated Edition

Timothy Budd

Copyright © 2000 by Addison Wesley Longman, Inc.

Original English language edition published by Addison Wesley Longman, Inc.

All right reserved.

No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher. For sale in the People's Republic of China Only.

本书中文简体版由 Addison Wesley Longman, Inc. 授权清华大学出版社出版发行, 未经出版者书面许可, 不得以任何方式复制或抄袭本书的任何部分。

北京市版权局著作权合同登记号: 图字 01-2002-3040 号

版权所有, 翻印必究。

本书封面贴有清华大学出版社激光防伪标签, 无标签者不得销售。

书 名: 面向对象 JAVA 编程思想 (修订版)
作 者: [美] Timothy Budd
译 者: 三联四方工作室
责任编辑: 尤晓东
出 版 者: 清华大学出版社 (北京清华大学学研大厦, 邮编 100084)
<http://www.tup.tsinghua.edu.cn>
印 刷 者: 世界知识印刷厂
发 行 者: 新华书店总店北京发行所
开 本: 787×960 1/16 印张: 23.5 字数: 481 千字
版 次: 2002 年 8 月第 1 版 2002 年 8 月第 1 次印刷
书 号: ISBN 7-302-05759-1/TP·3406
印 数: 0001~5000
定 价: 45.00 元

前 言

市面上有许多关于 Java 的书籍，它们教您怎样使用 Java 语言。但是，能够告诉您为什么要这么用的，却是寥寥无几。

许多书都能帮助您学习 Java 的编程技巧，但除了编程语法之外，能够指导您了解更深层问题的书却少之又少。本书的目标，是让您更全面、更完整地理解 Java 背后的一些基本原理，而不仅仅是该语言的使用技巧。

这些原理和实践将贯穿于全书，并且通过 Java 标准库中的一系列例子进行说明。例如，您在本书将学习到 AWT 中出现的许多设计模式，继承标准类中的各种用途，以及为什么会有多达 22 种不同类型的输入 / 输出文件流，等等。在此，您将知道为什么标准库中缺乏一个有序的容器，但这并不是一种简单的疏忽，而是对 Java 语言的基础结构及深奥属性的一种反映。

总之，本书并不是一本关于 Java 语言的参考手册，而是用于理解 Java 原理的一个有效工具。

本书的结构

本书划分为 5 个部分。

第 I 部分是对 Java 核心思想的全面概括。这些编程思想与语言本身无关。该部分首先介绍一些重要的面向对象概念（类、封装、行为及职责等），在第 II 部分再深入探讨这些概念。第 1 章指出，针对一个问题的解决方案，可在一系列代理之间通过交互进行构建。第 2 章叙述了 Java 的开发简史。如果作为教材使用，教师可自行决定是否跳过第 2 章，但对设计进行讨论的第 3 章是无论如何都不该忽略的。事实上，即使在没有正式开始学习 Java 编程语言的具体细节时，我也强烈建议学生使用 CRC 卡片至少进行一次练习，而且多多益善，书中提供了这样的一个例子。

第 II 部分通过几个标准的示范程序（如果采用传统的说法，也称为范例）正式开始 Java 的学习，这些例子将指导学生循序渐进地学习 Java 语言的各个方面。针对各个特定的应用程序，必要时还会适时地引入相应的新特性。注意这并不是对 Java 语言的完全、系统性的介绍，而只是提供了一系列经过精心挑选的例子，对本书其他部分将要讨论的一些具体机

11/5 8/1/07

制进行铺垫。

第III部分讨论了继承的问题，这是当学生了解了类和对象之后，必须掌握的下一个重要的面向对象概念。继承技术尽管看起来极其浅显，但却隐藏了许多细微之处，粗心的程序员往往会在这些小地方出错。为一种编程语言引入了继承的概念后，它往往会对语言的其他方面都造成不同程度的影响。考虑到这方面原因，对那些熟悉了传统的、非面向对象语言的读者来说，应该特别关注这一部分的内容。

第IV部分讨论了多态性，在学习这个概念时，往往会比学习继承时更容易栽跟斗。借助这一机制，面向对象技术的许多强大功能和适用性都可以得到明显的反映，所以多态性在Java中呈现出多种多样的形式。这一部分用许多例子进行了精彩示范。

第V部分将帮助学生们进一步地理解Java世界。本部分的内容有助于学生理解那些面向对象特征并不明显的一些特性。之所以把这些内容同本书其他部分分开，是为了保证前面部分一气呵成的叙述风格。但请读者注意，它们放在最后讲述并不表明特别难。根据教学的安排，这些内容可以全部省略，也可以同本书前半部的内容一起讲授。

本书的源码

本书中所有示例的源码都可通过匿名FTP下载获得，服务器是 *ftp.cs.orst.edu*，全部源码都放在 */pub/budd/java* 目录下。该目录还包含了其他一些内容，比如纠错表。另外，也可通过万维网在我的个人主页上下载这些东西，网址是 *http://www.cs.orst.edu/~budd/*。如果您还想索取更多的资料，请致函 *budd@cs.orst.edu*，或寄信给 Professor Timothy A. Budd, Department of Computer Science, Oregon State University, Corvallis, Oregon, 97331。

目 录

第 I 部分 理解面向对象的世界观	
第 1 章 面向对象的思想	1
1.1 观察世界的一种方式	1
1.1.1 代理和团体	2
1.1.2 消息与方法	2
1.1.3 责任	3
1.1.4 类和实例	4
1.1.5 类层次结构——继承	4
1.1.6 方法绑定、覆盖和异常	6
1.1.7 面向对象概念的总结	7
1.2 作为仿真的计算	7
1.3 本章小结	9
参考信息	9
思考题	10
练习题	11
第 2 章 面向对象编程简史	12
2.1 Java 的历史	13
2.2 客户端计算	14
2.2.1 字节码解释器和适时编译器	15
2.2.2 安全性问题	15
2.2.3 接口规范	16
2.3 Java 白皮书的描述	16
2.3.1 Java 是简单的	16
2.3.2 Java 是面向对象的	17
2.3.3 Java 是网络通用的	17
2.3.4 Java 是解释型的	17
2.3.5 Java 是健壮的	18
2.3.6 Java 是安全的	18
2.3.7 Java 是结构中性的	18
2.3.8 Java 是可移植的	19
2.3.9 Java 是高性能的	19
2.3.10 Java 是多线程的	19
2.3.11 Java 是动态的	19
2.4 本章小结	19
思考题	20
练习题	20
第 3 章 面向对象的设计	21
3.1 责任意味着互不干涉	21
3.2 小规模编程与大规模编程	22
3.3 为何要从行为开始	22
3.4 RDD 案例分析	23
3.4.1 交互式智能厨房助手 (IIKH)	23
3.4.2 处理组件	24
3.4.3 标识组件	24
3.5 CRC 卡——记录责任	25
3.5.1 给组件一个物理表示	25
3.5.2 “是什么” / “是谁” 循环	25
3.5.3 文档	26
3.6 组件和行为	27
3.6.1 被延迟的决策	27
3.6.2 为变化做准备	28
3.6.3 继续下一场景	28
3.6.4 交互图	30
3.7 软件组件	31
3.7.1 行为和状态	31
3.7.2 实例和类	31
3.7.3 耦合与聚合	32
3.7.4 接口和实现: Parnas 准则	32
3.8 形式化接口	33
3.9 设计组件的表现	35
3.10 实现组件	35
3.11 集成组件	36
3.12 维护与改进	36
3.13 本章小结	37
思考题	37

练习题	38	7.1 游戏的第一个版本	83
第II部分 理解范例		7.1.1 集合类	85
第4章 一个范例	39	7.1.2 鼠标监听器	86
4.1 程序结构	39	7.1.3 多线程执行	88
4.2 与 Java 环境的连接	42	7.1.4 异常处理	89
4.3 类型	43	7.2 增加标靶: 继承和接口	90
4.4 访问修饰符	44	7.2.1 PinBallTarget 接口	90
4.5 终身修饰符	46	7.2.2 往弹球游戏中增加标签	95
4.6 本章小结	46	7.3 弹球游戏的构建工具:	
参考信息	47	鼠标事件的再思考	100
思考题	47	7.4 本章小结	102
练习题	48	参考信息	102
第5章 球的世界	50	思考题	102
5.1 数据字段	52	练习题	103
5.2 构造器	53	第III部分 理解继承	
5.3 继承	57	第8章 理解继承	105
5.4 Java 的图形模型	58	8.1 有关继承的直观描述	105
5.5 Ball 类	59	8.2 Object 基类	106
5.6 同一个类的多个对象	60	8.3 子类化、子类型和可替换性	106
5.7 本章小结	62	8.4 继承的形式	108
参考信息	63	8.4.1 特殊继承	108
思考题	63	8.4.2 规范继承	109
练习题	64	8.4.3 构造继承	110
第6章 加农炮游戏	65	8.4.4 扩展继承	112
6.1 初级版加农炮游戏	66	8.4.5 限制继承	113
6.1.1 具有重力特征的球	71	8.4.6 合并继承	113
6.1.2 Integer 与 int	72	8.4.7 继承方式的总结	114
6.2 增加用户交互性	72	8.5 修饰符和继承	114
6.2.1 内部类	75	8.6 多人参与的编程活动	115
6.2.2 接口	76	8.7 继承的好处	116
6.2.3 Java 事件模型	77	8.7.1 软件可重用性	116
6.2.4 窗口布局	78	8.7.2 增加的可靠性	116
6.3 本章小结	79	8.7.3 代码共享	116
参考信息	80	8.7.4 接口的一致性	116
思考题	80	8.7.5 软件组件	117
练习题	81	8.7.6 快速原型化	117
第7章 弹球游戏的构建工具	82	8.7.7 多态性和框架	117

8.7.8 信息隐藏	117
8.8 继承的代价	118
8.8.1 执行速度	118
8.8.2 程序的大小	118
8.8.3 消息传递的开销	118
8.8.4 程序的复杂性	119
8.9 本章小结	119
思考题	119
练习题	120
第9章 案例分析: Solitaire	121
9.1 Card类	121
9.2 游戏	125
9.3 纸牌叠——动作继承	126
9.3.1 花色牌叠	129
9.3.2 发牌叠	130
9.3.3 Discard 纸牌叠	131
9.3.4 TablePile 纸牌叠	132
9.4 应用程序类	135
9.5 玩多态游戏	137
9.6 构建一个更完整的游戏	138
9.7 本章小结	138
思考题	139
练习题	139
第10章 软件复用机制	141
10.1 可替换性	141
10.1.1 “是一个”和“有一个”规则	142
10.1.2 代码继承和行为继承	143
10.2 合成与继承的描述	144
10.2.1 使用合成	145
10.2.2 使用继承	146
10.3 合成与继承的比较	147
10.4 继承与合成的结合	148
10.5 软件复用的新形式	150
10.5.1 动态合成	150
10.5.2 内部类的继承	152
10.5.3 未命名类	152
10.6 本章小结	153
思考题	154
练习题	154

第11章 继承的外延	155
11.1 多态变量	155
11.2 内存布局	157
11.3 赋值	159
11.3.1 克隆	160
11.3.2 赋值形式的参数	163
11.4 相等性测试	163
11.5 垃圾回收	166
11.6 本章小结	167
思考题	167
练习题	168

第IV部分 理解多态性

第12章 多态性	169
12.1 多态性种类	169
12.2 多态变量	170
12.3 重载	170
12.3.1 现实生活中的重载消息	171
12.3.2 重载和强制	171
12.3.3 独立类的重载	172
12.3.4 参数重载	172
12.4 覆盖	173
12.5 抽象方法	175
12.6 纯多态性	175
12.7 效率和多态性	176
12.8 本章小结	177
参考信息	177
思考题	178
练习题	178
第13章 AWT	179
13.1 AWT 类层次结构	179
13.2 布局管理器	182
13.3 用户界面组件	184
13.3.1 标签	185
13.3.2 按钮	185
13.3.3 画布	187
13.3.4 滚动条	187
13.3.5 文本组件	188

13.3.6 复选框	188
13.3.7 复选框组、下拉式列表和列表	189
13.4 面板	192
13.5 案例学习: 颜色显示	194
13.6 对话框	197
13.7 菜单栏	200
13.8 本章小结	202
思考题	202
练习题	203
第 14 章 输入输出流	204
14.1 流、Reader 及 Writer	204
14.2 输入流	205
14.2.1 物理输入流	206
14.2.2 虚拟输入流	206
14.3 流记号器	209
14.4 输出流	210
14.5 对象序列化	213
14.6 管道输入输出	215
14.7 Reader 和 Writer	219
14.8 本章小结	222
思考题	222
练习题	223
第 15 章 设计模式	224
15.1 适配器	224
15.2 组合	225
15.3 策略	226
15.4 观察者	227
15.5 享元	228
15.6 抽象工厂	229
15.7 工厂方法	229
15.8 迭代器	230
15.9 装饰器(过滤器或包装器)	231
15.10 代理	232
15.11 桥接	232
15.12 本章小结	233
参考信息	233
思考题	233
练习题	233

第 V 部分 理解 Java 世界

第 16 章 异常处理	235
16.1 传递到 catch 块的信息	237
16.2 捕捉多个错误	237
16.3 finally 子句	238
16.4 终止模型或恢复模型	238
16.5 标准类库中抛出的异常	239
16.6 抛出异常	240
16.7 传递异常	241
16.8 本章小结	242
思考题	242
练习题	242
第 17 章 实用工具类	243
17.1 Point 类	243
17.2 Dimension 类	243
17.3 Date 类	244
17.4 Math 类	246
17.5 Random 类	247
17.6 Toolkit 类	248
17.7 System 类	249
17.8 字符串及其相关类	249
17.8.1 String 类中的操作	250
17.8.2 字符串缓冲器	253
17.8.3 字符串记号器	254
17.8.4 解析字符串值	255
17.9 本章小结	256
思考题	256
第 18 章 理解 Java 中的图形	257
18.1 颜色	257
18.2 长方形	257
18.3 字体	261
18.3.1 FontMetrics 类	262
18.3.2 Font 示例程序	263
18.4 图像	266
18.5 图形上下文环境	268
18.6 一个简单的绘图程序	269
18.7 本章小结	273
思考题	273

练习题	274	21.5 应用程序和 applet 的结合	319
第 19 章 集合类	275	21.6 本章小结	320
19.1 元素类型和基本数据类型包装器	275	思考题	320
19.2 枚举器	276	练习题	321
19.3 数组	277	第 22 章 网络编程	322
19.4 向量	278	22.1 地址、端口和套接字	323
19.4.1 把向量当作数组	280	22.2 一个简单的客户机/服务器程序	324
19.4.2 把向量当作堆栈	280	22.3 多客户机	327
19.4.3 把向量当作队列	280	22.4 通过网络传递对象	332
19.4.4 把向量当作集合	281	22.5 提供更强大的功能	334
19.4.5 把向量作为列表	282	22.6 本章小结	335
19.5 堆栈集合	282	思考题	335
19.6 BitSet 集合	283	练习题	336
19.6.1 示例程序：质数过滤器	284	第 23 章 Java 1.2 的新特性	337
19.7 字典接口和哈希表集合	285	23.1 集合类	337
19.7.1 示例程序：词索引	286	23.2 Swing 用户界面组件	338
19.7.2 属性集	288	23.3 改进了 Graphics 类	338
19.8 为什么没有有序集合	288	23.4 国际化	338
19.9 构建自己的容器	291	23.5 Java Beans	338
19.10 本章小结	293	23.6 声音	339
思考题	294	23.7 数据库	339
练习题	294	23.8 远程方法调用	339
第 20 章 多线程	296	23.9 Servlet	339
20.1 创建线程	296	23.10 本章小结	340
20.2 案例分析：俄罗斯方块游戏	300	参考信息	340
20.2.1 俄罗斯方块游戏类	301		
20.2.2 PieceMover 线程	305		
20.2.3 Piece 类	309		
20.3 本章小结	312		
参考信息	312		
思考题	312		
练习题	313		
第 21 章 Applet 和 Web 编程	314		
21.1 Applet 和 HTML	314		
21.2 安全性问题	315		
21.3 Applet 和应用程序	315		
21.4 使用 applet 来获取资源	317		
21.4.1 统一资源定位符	317		
21.4.2 下载一个新的页面	319		
		附 录	
		附录 A Java 语法	341
		A.1 程序结构	341
		A.1.1 import 声明	341
		A.1.2 类声明	341
		A.1.3 接口声明	342
		A.1.4 方法声明	343
		A.1.5 构造器	344
		A.1.6 数据字段声明	344
		A.2 语 句	345
		A.2.1 声明语句	345
		A.2.2 赋值语句	345

A.2.3 过程调用	346	A.3.1 元表达式	349
A.2.4 if 语句	346	A.3.2 变量	350
A.2.5 switch 语句	347	A.3.3 数据字段访问和方法访问	351
A.2.6 while 语句	347	A.3.4 运算符	351
A.2.7 for 语句	347	A.3.5 对象的创建	352
A.2.8 return 语句	348	A.3.6 数组	352
A.2.9 throw 语句	348	附录 B Java API 中的包	353
A.2.10 try 语句	348	术语表	354
A.3 表达式	349		

第 I 部分 理解面向对象的世界观

第 1 章 面向对象的思想

这是一本关于面向对象编程的书。更具体地说，本书介绍了在 Java 编程语言环境中的面向对象的基本思想。10 多年来，面向对象程序设计一直是热门话题，但只是在最近，Java 语言才被公认为是面向对象思想的最好体现。本书将帮助您去理解 Java。本书并不打算写成 Java 语言的参考手册，这方面的书在市面上已经有很多了。对于一门语言的学习，不应该只限于知道它的语法，更应该知道为什么这门语言会被设计成这样、为什么它是以这样的方式来处理某些问题的，或者说，Java 程序为什么会是这个样子的。本书要解释的正是这些“为什么”问题。

面向对象编程经常被认为是一种新的编程范例 (paradigm)。范例一词的原意是指例子或模型。例如，给出某个例句的目的是为了帮助我们记忆某个外语动词的用法。更一般的说法是，模型也是一个例子，帮助您理解事物的原理。比如，牛顿的物理模型解释了为什么苹果会落到地面上。在计算机科学中，范例用于解释组成计算机程序的各要素是如何组织到一起，以及它们之间是如何相互起作用的。正因为如此，要理解 Java 首先就必须理解面向对象的世界观。

1.1 观察世界的一种方式

为了说明面向对象编程的主要思想，让我们考虑如何处理一个现实世界的问题，然后看看如何让计算机建立更接近技术模型。假设我想送一束鲜花给外地一位名叫 Sally 的朋友。由于距离的关系，我不可能亲自把花送给 Sally。但是，给她送花还是一件相当容易的事：我只要到本地的花商 Flora 处，告诉她 Sally 的地址、花的品种和数量，就可以确保将花迅速并自动地送到 Sally 手中。

1.1.1 代理和团体

可能有点啰嗦，我们还是需要强调解决该问题所用的机制：找到一家合适的代理（即 Flora），告诉她一条消息，该消息中有我的要求，然后由 Flora 负责满足我的要求。Flora 会用某种方法——某种算法或者一组操作——来完成这项任务。我不需要知道她使用哪种方法来满足我的要求；实际上，我也不想知道具体的细节。这些信息通常都是隐藏的。

但是，如果我去深入调查的话，可能会发现，Flora 把我的要求转达给了我朋友所在城市的另一位花商，当然，传达的信息会稍有不同。而这位花商会有一个下属负责送花。花商会给送花人传递另外一条消息，并把花送到指定地址。更早些时候，Sally 所在城市的花商可能已从花卉批发商处购得鲜花。花卉批发商会从鲜花种植园主处获得鲜花，而每一位鲜花种植园主又可能管理着许多园丁。

因此，我们可以看到，正如解决我所面临的问题一样，面向对象问题的解决首先需要许多其他独立个体的帮助（如图 1.1）。没有他们的帮助，我的问题是不可能轻易得到解决的。我们把这一过程总结为以下通用的形式：

一个面向对象的程序是由一个相互作用的代理团体组成，这些代理被称作对象。每一个对象承担一个角色。每一个对象都提供一种服务或者执行一种动作，以便为团体中其他对象服务。

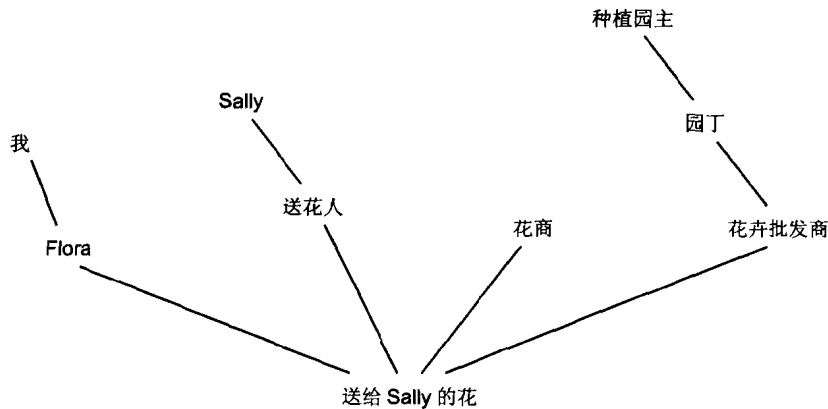


图 1.1 帮助我完成送花过程的代理机构

1.1.2 消息与方法

使问题最终得到完满解决的一连串反应开始于我向 Flora 发出的一个请求。这个请求引出了别的请求，而别的请求又引出了更多的请求，直到最终鲜花送到了朋友的手中。从中

我们可以看到，团体中的成员是通过互相发出请求来相互作用的。因此，我们可以得出解决面向对象问题的第二条原则，通过什么工具发起活动：

在面向对象编程中，激发动作需要向响应该动作的代理（一个对象）传送一条消息。消息中包括对动作的请求，还有完成要求所需的其他信息（参数）。接收者就是消息所发送到的对象。如果接收者接收了消息，它就承担了执行指定动作的责任。作为消息的答复，接收者将执行某个方法，来满足所接收的请求。

我们注意到，在消息传递过程中的一个重要原则，就是信息隐藏（information hiding）——也就是说，发送请求的客户并不需要知道请求是如何被执行的。在消息传递过程中，也隐含了其他所有公认的原则。如果需要完成一个任务，客户首先想到的就是找到能完成任务的人。而有丰富传统编程技术经验的程序员却常常想不到这一点。通常情况下，要克服的一个难点就是程序员的信任。程序员都喜欢自己编写所有的代码，而不去使用别人的服务。面向对象编程的一个重要方面就是可重用组件的开发，而使用可重用组件时重要的第一步就是无条件相信别人编写的软件。

信息隐藏也是使用传统语言编程的一个重要方面。消息传递与过程调用有何不同呢？虽然这两种情况都是收到请求后执行一系列预先定义好的步骤。但却有两点重要的不同：

首先，消息传递必须有指定的接收者：接收者就是消息所发送给的某些对象。而在过程调用中，并没有指定接收者。

其次就是对消息的解释（或者说，用于响应消息的方法）依赖于接收者，会因接收者的不同而不同。比如说，我给我的妻子伊丽莎白发送一条消息，她就会明白，并且产生了预期的效果（把鲜花送到了我朋友处）。然而，伊丽莎白用于满足请求的方法（很可能只是简单地把请求传递给了 Flora）与 Flora 用于响应相同请求的方法不相同。如果我请求我的牙科医生 Kenneth 给我的朋友送束鲜花，他也许不知道该如何解决这个问题。即使他能理解这个请求，也可能会做出错误的诊断。

让我们把话题转回到计算机和编程上。我们曾说过，消息传递和过程调用的区别在于消息传递有一个指定的接收者。不同接收者的解释——指响应消息所执行的方法集合——会有所不同。通常，直到运行时才能最后确定所给消息的特定接收者，因此也只有到那时才能最终确定要调用的方法。我们把这种在运行时才能将消息（函数或过程名）与响应消息的代码段（方法）相绑定的机制称为后期绑定。而在传统的过程调用中，绑定名称和代码段相对来说就很早了（在编译时，或者联接时）。

1.1.3 责任

面向对象编程的一个基本概念就是用术语责任（Responsibilities）来描述行为。我的动

作请求只表达了所希望得到的结果（给朋友送花）。花商 Flora 可以自由选择完成任务的方式，我不会去干涉她的实现细节。

当用责任去描述这个问题时，得到了更高层次的抽象。这允许对象之间具有更大的独立性，在解决复杂的问题时这是一个关键的因素。与对象有关的所有责任集合被称为协议。

传统的程序通常直接作用于数据结构，例如，直接改变数组和记录中的字段。相反，面向对象的程序是请求数据结构（即对象）来执行一项服务。我们可以引用以下著名的绕口令，概括从传统的、结构化的角度看软件与从面向对象角度看软件的差别：

不要问你能为你的数据结构做些什么，而是去问你的数据结构能为你做些什么。

1.1.4 类和实例

尽管我只跟 Flora 打过几次交道，但我走进她的商店，说出我的要求，可以粗略地想像她的行为。我能够做出一定的假设，这是因为我了解花商的一般信息，Flora 作为花商类别中的一个实例，也会符合这些一般的特征。我们可以用术语 Florist 代表所有花商这一类别（或者称为类）。在此我们可以得出面向对象编程的又一条重要原则：

所有的对象都是某个类的实例。对象响应消息所调用的方法由接收者类决定。给定类的所有对象都使用相同的方法来响应相似的消息。

1.1.5 类层次结构——继承

我知道更多有关 Flora 的信息——她不仅是一个花商，她还是一位店主(Shopkeeper)。跟她交易时我需要付钱，而她会给我开收据。对于别的杂货商、百货商以及其他店主，也有这些动作。由于 Florist 是 Shopkeeper 类别的一种特例，它就具有了店主的共同特性，Flora 同样也就具有了店主的这些共同特性。

我是按图 1.2 的类别结构来组织我对 Flora 的认识的。首先 Flora 是一位 Florist(花商)，而 Florist 又是 Shopkeeper(店主)的特例。更进一步，Shopkeeper 同样又是一个 Human(人类)；因此我们可以推断 Flora 很可能是双足动物。而 Human 是 Mammal(哺乳动物，这种动物会照顾年幼者，并且有毛发)，而 Mammal 是一种 Animal(动物，这种物体会呼吸氧气)，Animal 又是一种 Material Object(物质对象，这种东西有重量和体积)。因此，对 Flora 的认识就不仅仅只限于她本身，甚至远远超过所在的类别 Florist。

关于一般类别的知识也适用于特殊类别，这个原则被称为**继承**。我们可以这么说，Florist 类继承了 Shopkeeper 类（或者类别）的属性。

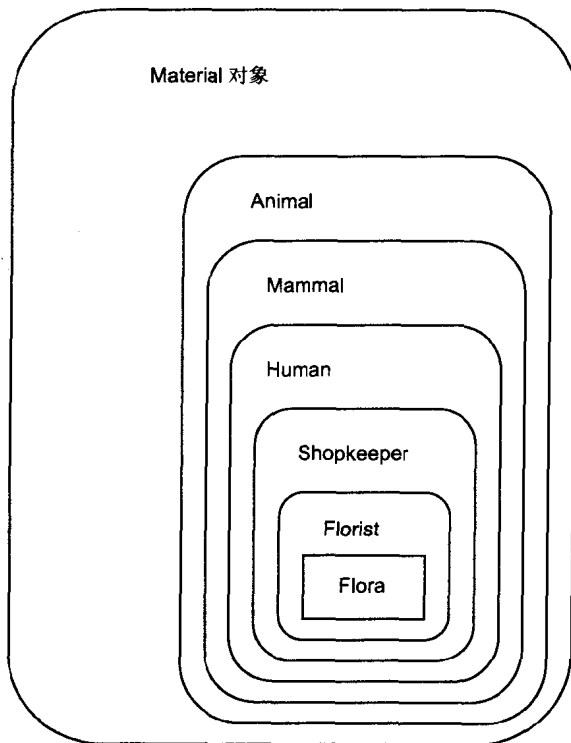


图 1.2 Flora 周围的类别

另一种图形常常用来说明这种关系，特别是在不同的个体有不同的血统时。这种技术把类组织成树状的层次结构，在顶部有一些较抽象的类（比如 Material Object 或者 Animal），下部是一些较具体的类，而最后是一些个体。图 1.3 表示了 Flora 的类层次结构。这个层次结构也包括了我的妻子伊丽莎白（Elizabeth）、我的宠物狗 Flash、动物园里的鸭嘴兽 Plyl 以及送给朋友的鲜花。

由于 Flora 是 Human 类的实例，她就拥有了人类的共同特性，我也可以把这些共同特性用于我的妻子伊丽莎白。我的妻子作为 Mammal 类的实例所拥有的特性也可以用于 Flora。关于 Material Object 的所有成员信息，对于 Flora 和她的花而言都是等同的。我们可以把继承总结为：

可以把类组织成层次继承结构。子类将继承它上部父类的所有属性。抽象父类（如 Mammal）不能有自己直接的实例；它只用于创建子类。

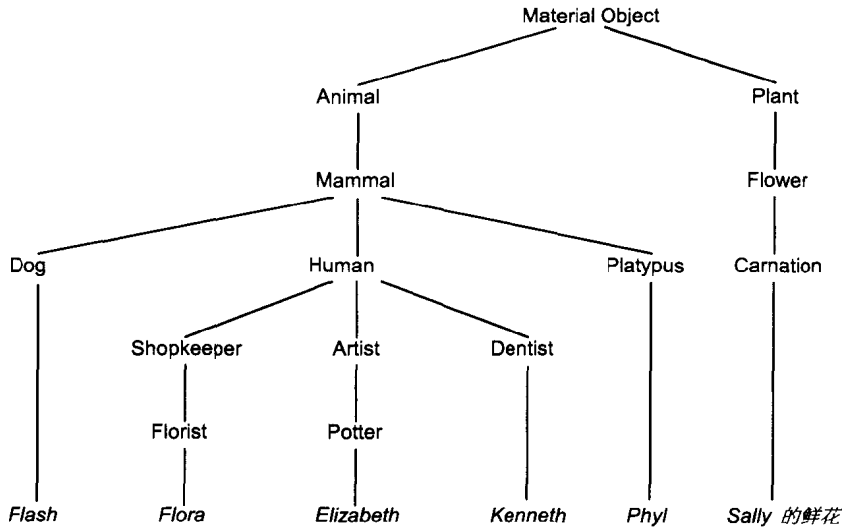


图 1.3 多种物质对象的类层次结构

1.1.6 方法绑定、覆盖和异常

鸭嘴兽 Phyl 给我们简单的组织结构提出了一个问题。我们知道哺乳动物是直接生育后代的，Phyl 当然是 Mammal，但是 Phyl（或者是 Phyl 的伙伴 Plyllis）却是通过孵卵的方式来繁衍后代。为了能够适应这种偏差，我们需要找到一种技术，能把异常溶入一般规则。

我们通过允许子类覆盖（Overriding）从父类中继承的信息来实现这种功能。大多数情况下，实现这种方法需要某个子类方法名称和父类方法名称相同，并且还要有一定的规则，说明如何查找一个方法以便匹配特定的消息：

首先在接收者类中搜寻响应给定消息的方法，如果没有找到合适的方法，就到该类的父类中继续搜寻，这种搜寻在父类链中一直往上持续，直到找到合适的方法，或是父类链已结束。在前一种情况下，将执行找到的方法；若是后一种情况，则返回一个错误信息。如果在类层次更高的地方又找到了相同的方法名称，我们就说所执行的方法覆盖了所继承的行为。

在许多面向对象的编程语言中，如 Java，即使编译器不能决定在运行时该执行哪一个方法，至少可以判断是否有合适的方法，或是返回一个用于诊断的编译时错误消息，而不是返回运行时消息。

我的妻子伊丽莎白与花商 Flora 对我的消息采用不同的响应方法，这正是多态性的一个例子，这个重要的面向对象编程方面将在第 12 章中详细介绍。正如以前已经解释过的，我