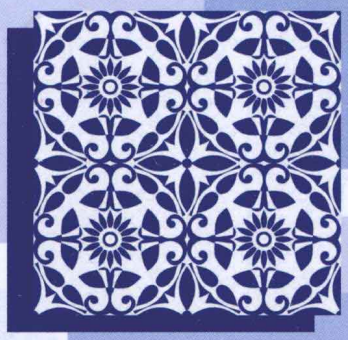
HZ BOOKS
华章教育

重点大学计算机教材

Java语言程序设计

面向对象的设计思想与实践

吴倩 林原 李霞丽 编著



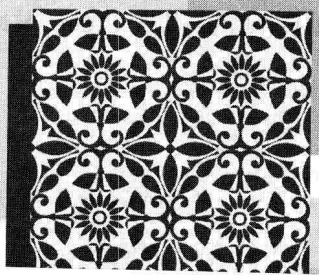
Java Programming Language

机械工业出版社
China Machine Press

重点大学计算机教材

Java语言程序设计
面向对象的设计思想与实践

吴倩 林原 李霞丽 编著



Java Programming Language

机械工业出版社
China Machine Press

本书围绕面向对象的设计思想展开主题，详细讲解了 Java 语言基础语法、面向对象的三大特征、图形用户界面的编程方法、多线程编程、Java 的集合框架和 Java 语言在 Android 平台中的应用等基本理论及实用开发技术。

本书强调面向对象设计思想的重要性，在内容组织上力求从设计理念出发，合理地解释知识点；循序渐进地讲述基本理论，深层次地剖析程序设计方法；在例题与习题的选用与设计上深入浅出，强调其连贯性与实用性，锻炼读者的面向对象程序设计能力。

本书可作为普通高等院校计算机及相关专业的 Java 语言程序设计教材，也可作为 Java 编程爱好者的参考书。

图书在版编目（CIP）数据

Java 语言程序设计：面向对象的设计思想与实践 / 吴倩，林原，李霞丽编著. —北京：机械工业出版社，2012.9
（重点大学计算机教材）

ISBN 978-7-111-39469-3

Ⅰ. J…　Ⅱ. ①吴… ②林… ③李…　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（2012）第 192333 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）
责任编辑：余　洁
北京京师印务有限公司印刷
2012 年 9 月第 1 版第 1 次印刷
185mm×260mm • 16.5 印张
标准书号：ISBN 978-7-111-39469-3
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
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

前　言

Java 语言作为当今最流行的面向对象程序设计语言，随着网络的发展而被广泛应用，并与我们的日常生活息息相关。信息化的发展带动了 Java 在金融、通信、制造、电子政务、移动设备及消费类电子产品等领域日益广泛的应用，尤其是近年来 Android 移动平台的推广，为 Java 语言注入了新的活力，使之成为计算机业界一颗璀璨的明星。

近年来，随着社会对 Java 开发人才的需求日益迫切，广大开发人员学习 Java 语言的兴趣也与日俱增。但是如何选择一本适合自己的教材，从而快速提高 Java 编程水平，对于很多 Java 初学者来说是很重要的。

本书作者具有丰富的教学经验及工业界软件开发经历，本书的设计力求切合实际，尽量缩小计算机专业毕业生与工业界人才需求之间的差距，引导读者从较深层次理解程序设计，少走弯路。本书自始至终渗透着面向对象的编程思想，以 Java 语言为实现方式，强调 Java 语言的精华就在于“面向对象思想”。本书覆盖内容全面，从 Java 语言的基础知识到 Java 类库的应用、数据库程序设计及 Android 平台程序开发，使读者能够从通俗易懂的语言中理解程序设计理念，帮助读者认识到任何一种面向对象程序设计语言的语法和风格可能有所不同，但是其编程思想都是一致的。编程语言的学习不应该局限于表面的语法格式，而须深入理解程序设计语言的本质规律，掌握其精髓思想，这样才能真正学会并运用一门程序语言。

本书特色

本书的特色在于以面向对象设计理念为主线，以 Java 编程语言为载体，阐述面向对象程序设计思想及方法，强调语法知识的学习应以理解编程思想为前提。

本书以当今业内流行的 Eclipse 集成开发环境为开发平台，采用 UML 统一建模语言表述程序分析及设计。全书采用一个贯穿全书的开发实例（电子产品商店管理系统），随着面向对象程序设计理论的深入和 Java 语法知识点的展开，由浅入深逐步完善，扩展实例的规模，最后形成一个具有图形用户界面、实现数据库访问操作、规模适中的应用管理系统。

本书内容全面，理论阐述简洁明了，具有较强的可读性，实例生动、完整、连贯性强，力求反映 Java 技术的新成果、新趋势，为读者展示 JDK 5、JDK 6 以及 JDK 7 的最新技术和思想方法，并介绍 Java 在 Android 平台中的简单应用。

本书习题侧重于培养读者自主学习、自行探索、独立解决问题以及团队协作的能力。通过习题，读者不仅可以练习编程，而且须自行设计程序架构，必要时还需要查阅资料以解决问题。本书习题具有连贯性，循序渐进并逐步扩展为一个完善的应用系统。

本书内容

全书分为三个部分：第一部分全面阐述面向对象程序设计思想；第二部分结合实例，以 Java 语言描述面向对象的三大特征及程序设计方法；第三部分介绍 Java 语言类库在输入输出、多线程、图形用户界面、数据库及 Android 无线移动通信平台中的应用。

全书共分为 13 章，内容如下：

第 1 章介绍面向对象的基本概念、面向对象程序设计的三大特征（封装、继承和多态），类的建模及层次结构设计、面向对象程序设计原则。本章还对 Java 语言的特点及集成开发环境 Eclipse 进行了全面介绍。

第 2 章全面介绍 Java 语言基础知识，具体包括 Java 语言基本元素、基本数据类型、引用数据类型、基本数据类型的封装类、运算符、表达式及流程控制。

第 3 章讲述类与对象。从如何设计类入手，到对象的创建及使用方法、static 静态成员的基本特征、方法重载、包的概念、类的访问控制等。除此之外，本章还介绍了基础类库的知识和 Java 的文档生成器。

第 4 章讲述异常处理，包括异常的概念、异常的分类、异常的处理机制原理、自定义异常类。

第 5 章讲述类的重用，包括类的继承和类的组合两种方式的语法实现，还介绍了抽象类和抽象方法、类成员方法的覆盖。

第 6 章讲述接口与多态，从为什么需要接口入手，逐步引导读者了解接口的声明及实现、多态的概念及实现、多态的适用环境。另外，还介绍了内部类的概念及使用方法。

第 7 章讲述 Java 的集合框架及泛型的相关知识。

第 8 章讲述输入输出，包括 I/O 流的概念、I/O 流的分类、文件读写以及对象序列化。

第 9 章讲述 JDBC 访问数据库，主要介绍 JDBC 技术的原理、JDBC API、通过 JDBC 访问数据库、实现与数据库的连接，以及访问数据库的一系列操作。

第 10 章讲述 Java 图形用户界面，具体包括 Java 图形用户界面类库、Swing 组件、Swing 组件的层次结构、Swing GUI 程序、事件处理机制、Eclipse 下的可视化图形界面编程。

第 11 章讲述 Java 多线程，具体包括进程与线程的概念、多线程编程基础、线程的生命周期、线程的常用方法、线程的优先级、多线程的编程方式、死锁等相关问题的处理。

第 12 章讲述 Java applet，具体包括 HTML 与 applet 简介、applet 的工作原理、applet 的创建、HTML 中 applet 标签的使用、applet 的生命周期、applet 在 Web 中的应用。

第 13 章讲述 Android 平台的系统架构、开发环境和应用程序组件、在 Eclipse 环境下的简单开发示例以及 Android API。

本书提供完整的示例程序来讲解基本概念，所有程序都在 Eclipse 3.7 环境下编译并运行通过。

本书作者

本书第 1 章、第 3 ~ 6 章、第 10 章、第 13 章由吴倩编写，第 2 章、第 7 ~ 9 章由李霞丽编写，第 11 章和第 12 章由林原编写，全书由吴倩统稿。另外特别感谢徐鹏、郭莹对本书编写提供的帮助。

致谢

在书稿的编写过程中，机械工业出版社的同志对此书的出版进行了周到的安排并给予了支持，作者也得到了家人、朋友的大力支持，使此书得以在短时间内出版，在此对他们表示真挚的感谢！

由于时间仓促及水平有限，书中难免存在不妥之处，恳请广大读者给予批评指正，以便进一步完善（作者电子邮箱：wuqian@muc.edu.cn）。

教学建议

课程内容	教学要求	学时分配
第1章 面向对象程序设计思想	理解面向对象的核心概念 理解面向对象程序设计的三大特征 掌握从现实世界抽象出类的原则 了解 Java 语言的特点 掌握 Eclipse 开发环境 掌握 Java 的程序结构	2
第2章 Java 语言基础知识	掌握 Java 数据类型 理解 Java 语言基本元素、运算符与表达式以及控制语句 重点掌握 Java 数组与其他语言数组的不同 掌握封装数据类型	4
第3章 类与对象	掌握类的创建、对象初始化以及类的使用 掌握数据成员及方法的使用、类的访问控制、static 数据与方法、this 指针、final 修饰符、方法重载 了解 Java 基础类库及相关的应用 理解 String 类和 StringBuffer 类的异同	4
第4章 异常处理	了解异常的概念及分类 重点掌握对异常的处理，学会使用 try、catch、throw 语句处理异常 理解自定义异常类	2
第5章 类的重用	掌握类的继承及实现方式 了解终结类与终结方法 掌握抽象类与抽象方法 掌握类的组合及实现方式	4
第6章 接口与多态	理解使用接口的必要性 掌握接口的基本概念及实现方式 理解多态实现的基本原理 掌握多态的概念及实现方式 理解内部类的概念及应用	4
第7章 对象的集合	理解 Java 集合框架的层次结构 掌握几种基本的集合如 List、Set 和 Map 的使用方法 学会查阅 Java API 集合提供的方法进行排序、查找等操作 掌握集合中元素的遍历方法 理解泛型的概念 掌握泛型在集合中的应用	2
第8章 输入输出	理解 I/O 流的概念及 I/O 流的分类 重点掌握字节流、字符流的文件读写方式 理解标准输入输出 理解对象流的概念 掌握对象序列化的实现方式	2

（续）

课 程 内 容	教 学 要 求	学时分配
第 9 章 JDBC 访问数据库	了解 JDBC 的基本概念及技术原理 掌握 JDBC API 中主要类及接口的使用方法 掌握 JDBC 访问数据库的一系列操作 掌握 Eclipse 下通过 JDBC 访问数据库的操作	4
第 10 章 Java 图形用户界面	理解 Java 图形用户界面容器、组件之间的层次关系 理解 Swing 类库的基本组件 理解几种常用的布局管理器的使用 掌握 Java 的事件处理机制原理及几种实现方法 学会在 Eclipse 环境下使用 Swing Designer 进行可视化 GUI 程序开发	4
第 11 章 多线程	掌握多线程的基本概念 掌握多线程的两种创建方法 了解多线程的生命周期、守护线程及线程优先级 理解线程死锁的基本概念及处理方法 掌握多线程的四种编程方式	4
第 12 章 Java applet	理解 applet 与普通应用程序的区别 了解 HTML 的概念及运用 掌握 applet 的创建方法以及它的生命周期 掌握编写 applet 的方法	2
第 13 章 Java 语言在 Android 平台中的应用	了解 Android 平台的系统性能、开发环境及系统架构 理解 Android 应用程序组件 掌握 Eclipse 下开发 Android 应用程序 理解 Android API	2

说明：本书的理论授课学时建议安排为 36～45 学时，学生上机实验学时可以灵活掌握。不同专业可以根据不同的教学要求和执行计划对教材内容进行适当取舍。

目　录

第 1 章　面向对象程序设计思想

计算机技术日新月异地发展，所应用的程序设计语言已有上百种，要学习的编程新语言也越来越多。历经近二十年的发展，面向对象程序设计语言仍然是当今程序设计的主流，得到了广大程序开发者的青睐。近年来，C++、C#、Java、Objective-C 四大面向对象程序设计语言一直稳居排行榜前几名。

学习面向对象编程最重要的是什么？不是语法，而是思想！其他编程语言也如此，任何一种语言都有其自身的设计理念，精通一门编程语言，需要掌握它的精髓，多问几个为什么，为什么这么设计，目的是什么。只有知其所以然，才能知其然。如果掌握了编程的思想，你将发现理解过程就是小菜一碟，再去学其他的语言，也就万变不离其宗了，无非是学习一种新的表达方式而已。学习一门编程语言首先应以理解其编程思想为前提，然后掌握其基础语法知识，最后学习灵活运用工具和类库写出好程序。

本章我们将围绕面向对象编程语言的设计理念展开探讨面向对象的两个核心概念——类与对象。了解为什么会有类与对象，它们与现实世界的关系以及怎样从现实世界抽象出面向对象的程序。在此基础上分析面向对象的三大特征——封装、继承、多态，这些特征的由来，以及它们给程序设计带来哪些好处，并从理论上阐述面向对象程序设计所遵循的原则。最后介绍面向对象程序设计 Java 语言、Java 语言特性、开发环境、程序特征等相关内容。

1.1　类和对象

生活中我们常说，人以类聚，物以群分。如果问：“这个世界由什么组成？”按照生物学的标准，世界由动物、植物等组成，细分下去，动物可分为脊椎动物、无脊椎动物等，脊椎动物又分为兽类、鱼类、爬行类……当然，按照其他标准分类，还可以得出其他不同的种类，即无论怎样划分，包罗万象的世界都是由不同类别的事物所构成。

什么是“类”？比如人类，如何给人类下定义？首先让我们来看看人类具有的一些共同特征。人都有身高、体重、年龄、血型等一些属性。根据《现代汉语词典》里对人类的定义，“人是能制造工具并能熟练使用工具进行劳动的高等动物”。所以，区分人类与其他类型动物的关键就是这些特征，凡具有制造工具、熟练使用工具的高等动物就是人，具体包括人的属性（名词）以及对外界呈现出的一些行为（动词）。每一种类都会展现出它独特的属性和行为特征。猴子不是人类，为什么？猴子不具备独立思考能力、不会制造工具、不会熟练使用工具、不具备人的行为。我们给现实生活中的“类”下一个定义，类即具有相同特征及行为的一种群体。

“人类”仅仅是一个抽象的概念，不是实际存在的实体，真正的实体是所有具备“人类”这个群体特征的每一个具体的人，也就是“人类”这个类的对象，我们每一个人都是一个“人类”的对象。正是诸多不同种类的对象构成了现实世界。

由此可见，类描述了一组有相同特性（属性）和相同行为（方法）的对象。类与对象是面向对象思想的两个核心概念。

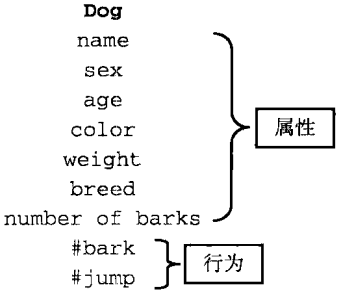
面向对象是一种思维方式，其思想符合人们认识现实世界的思维方式。程序开发者从现实生活的角度出发，以符合人的思维方式解决编程问题，将现实世界用程序语言描述出来，这种方式

使程序设计非常直观、容易理解，容易被人接受，这便是面向对象程序设计语言得以流行的直接原因。

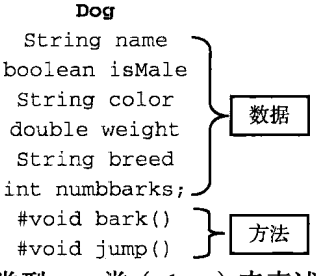
如何把现实生活中的类用编程语言表示出来呢？首先得认识到，计算机语言与我们人类语言是无法比拟的，英语有主语、宾语、动词、形容词、副词等，更不用说我们博大精深的中文还有成语、诗歌等丰富的表现形式，可以尽情地描绘整个大千世界。而计算机语言就不同了，开发者编程如同写文章，用其提供的有限的几种基本数据类型，比如整型、字符串型、字符型、浮点型等，就得表示出事物的属性。写出好代码的关键就看如何把想做的事用这些有限的数据类型组织并运用起来。

在程序中，通过定义若干数据类型的变量来表示对象的属性，比如，年龄用整型表示、性别用字符型表示。面向对象的编程语言的优点在于允许把基本的数据类型组合起来，创建一个适合自己的自定义数据类型——“类”，这种类不仅可以放置属性，而且可以将对象们共有的行为（在Java中称为方法，在C/C++中称为函数）也放置其中。

我们举个例子来说明如何用编程语言模拟真实世界。现实世界中的狗，都属于Dog这个类，具有名字（name）、年龄（age）、毛色（color）、重量（weight）、品种（breed）等名词性特征，也有一些动词性行为比如嗥叫（bark）、跳跃（jump）等。现实世界中的Dog类描述如下：



在面向对象程序中，可以将对象的特征用各种数据类型组合在一起，行为用方法表述出来，函数在Java中被称为方法，将这两种表述组合在一起，程序中的类描述如下：



如果用Java提供的一种抽象的数据类型——类（class）来表述，可以定义Dog类如下：

```
class Dog{
    String name;
    boolean isMale;
    String  color
    double weight;
    String  breed;
    int numbbarks;
    void bark() {……// 编写具体的方法实现部分}
    void jump(){……// 编写具体的方法实现部分}
}
```

这样就构造出一个能表示出Dog共同属性与行为特征的类。所有的狗都具备以上的特征，也就是Dog类的一个对象，任何一只狗都具备Dog类的属性，并表现出Dog 类的行为。

这就是面向对象编程为我们提供的模拟世界的解决方案。面向对象的编程思想是以类来划分、以对象来考虑问题，这与传统的面向过程编程以实现的过程来划分是不同的。

举一个简单例子来说明面向对象与面向过程编程的区别。写一个程序来实现用枪射击 10 只鸭子，如果用 C 语言来实现，首先考虑的是实现步骤：举起枪，瞄准，扳机，发射，射击鸭子 1、鸭子 2……直到鸭子 10，考虑问题的出发点是射击鸭子的过程。如果用面向对象语言来实现就不一样了，首先划分类，鸭子属于鸭子类，枪属于枪类，枪类的对象就是任何一把枪，包括猎枪、手枪、气枪等，这些枪都具备瞄准、扳机、发射功能，我们将挑选其中的一杆猎枪，即“枪”这个类中的一个对象来完成射击任务，需要做的就是用“猎枪”对象来执行一系列行为。如果换成另一个对象——手枪，没有必要重新再写一遍瞄准、扳机、发射的功能，直接执行行为就可以了。如果射击的目标变成老虎，“枪”类的代码仍然可以使用。可见，与面向过程编程相比，面向对象这种思维方式显然更符合认识现实世界的思维方式，而且代码简洁、可重用性强，这也是面向对象得以流行的主要原因。

1.2 面向对象程序设计的三大特征

学习程序设计语言的关键在于学习其编程思想。这一点，从 Bruce Eckel 的《Thinking in Java》可以体会到。那面向对象语言的编程思想体现在哪儿呢？怎样真正地“Think in Java”，而不仅仅是“Programming in Java”？起点就在于其三大特征——封装、继承、多态。

有人认为采用封装、继承、多态语法写出一个 Java 程序，就完全掌握了面向对象语言？那只能说学到了皮毛，仅仅会用 Java 语法写程序而已。事实上，封装、继承、多态是一种设计理念，一种程序艺术，与程序语言没有关系。如果真正透彻地理解了这三大特征的真谛，即使不用 C++、Java、Objective-C 等面向对象的语言，用 C 语言也能写出面向对象的程序。在面对一个项目时，有经验的开发者可以立刻在脑海里构思出怎样从软件角度设计它，类与类之间怎样关联，怎样用封装、继承、多态等机制勾画出程序的基本架构蓝图，以及优化程序架构。至于一些细节问题，如采用什么语言，C++还是 Java，从语法上怎样来实现，那都是次要的，毕竟语言只不过是程序设计思想的一种表现形式而已。当然，要达到这种集设计与编程为一体的境界，还需要脚踏实地，稳扎稳打，从编程中逐步提高，从实践中成长。

我们将把这三大特征的设计理念作为一条主线，一步一步贯穿到后面的“类的重用”、“接口与多态”章节中，以实例的方式逐步详细地讲解这三大特征在程序设计中是如何体现的，该怎样去运用它们。

现在我们先简单解释一下这三大特征。

1.2.1 封装

封装，即将对象的数据和基于数据的操作封装成一个独立性很强的模块。封装是一种信息隐蔽技术，使得用户只能见到对象的外特性，而隐蔽对象的内特性。封装的目的就是将对象的使用者和所有者分开，使用者不必知道对象的内部细节，只需通过对象所有者提供的通道来访问对象。

通俗地解释就是，把对象不需要对外提供的数据隐藏起来，对外形成一道屏障。数据如何隐藏，藏哪儿？藏在类里，基于数据的操作也隐藏在类里。这样一来细节都隐藏起来，那如果外界想对封装的数据进行访问怎么办呢，那就得依靠对象定义的公共接口了，这种公共接口就如同与外界沟通的一个桥梁。以下我们举例说明封装的概念。如果你是“学生”这个类的一个对象，具有姓名、性别、学号、英语成绩、C 语言成绩、高数成绩等数据，一般来说，这些都是你的私有信息，别人是无权知道的。如果班长要统计全班成绩并排名又如何操作呢？这时就需要得到你的允许，即把成绩

公开。可以定义一个公有的方法，比如 getGrade()，通过这个方法访问你的某些私有数据，如成绩，班长就可以得到他需要的成绩，当然，他不需要的信息仍然封装在类里，没有必要公开。

又如另一个对象封装的例子，如图 1-1 所示，公司里的一个部门就是一个对象，三个对象有其封装好的数据及对数据的操作（方法）。以财务部为例，财务数据不能轻易公开，属于隐蔽信息，如果销售部门想看财务数据，一般是不允许的，但是如果经上级批准，也可以通过公有的接口方法“管理财务”操作来实现。图 1-1 中椭圆形表示数据（属性），矩形表示操作（方法）。

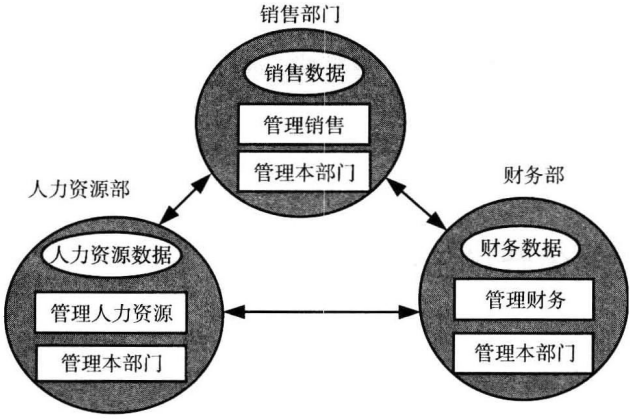


图 1-1 对象的封装示意图

1.2.2 继承

继承，即在现有类的基础上创建新类，并在其中添加新的属性和功能。现有类与新类之间是一种一般性与特殊性的关系，现有类具有该类及其新类的共同特征，而新类还具有一些特殊的特征。

类的封装引出了继承的概念。日常生活中的类通常可以再派生出新类：人类可以分为下列子类，如白种人、黄种人、黑种人；交通工具类可分为下列子类，如轿车、卡车、公共汽车等。面向对象最大的优点就是代码重用，而代码重用的主要方式就是继承（后面我们还会提到代码重用的其他方法）。继承是在封装的基础上实现的，前面提到最好把一个封装好的类放在一个.java 文件里，目的就是便于类的重用，可以直接在这个类的基础上派生出子类。该子类可以使用现有类的所有功能，并在无需重新编写现有类的情况下对这些功能进行扩展。试想你要设计一个关于狗的卡通（Cartoon）程序，首先想到的是定义一个 CartoonDog 类，原先已定义的 Dog 类已经具有了普通狗的特征，就没有必要再从头写代码，可以直接继承 Dog 类，在它的基础上添加一些卡通的特性，生成一个新的 CartoonDog 类，既省时又省力。

Java 语言中通过继承创建的新类称为“子类”或“派生类”，被继承的类称为“基类”、“父类”或“超类”。继承的过程就是从一般到特殊的过程。在继承过程中子类继承了父类的特性，包括方法和变量；子类也可修改继承的方法或增加新的方法，使之更适应特殊的要求。继承使代码可以重用，避免了数据、方法的大量重复定义，保证了系统的可重用性，促进了系统的可扩充性，同时也使程序结构清晰，易于维护，提高了编程效率。

下面我们以“愤怒的小鸟”为例。假如你是“愤怒的小鸟”游戏的开发者，当游戏版本不断更新，如 Angry Bird、Space Angry Bird、Crazy Bird……如果游戏的每一个版本都从头开始开发，就不可避免产生重复劳动、开发时间加长、效率低下，而采用面向对象的设计就是一种比较好的解决方案。

图 1-2 展现了小鸟类的继承关系，设计 Bird 类为一个基类，具有普通鸟的特征，而 Angry Bird、Crazy Bird、Kind Bird 都是在 Bird 的基础上派生出来的子类。它们继承了 Bird 类的所有功能，同时又具备各自的一些新的特性。把原有的 Bird 代码拿过来重用，不失为一种提高效率的好方法。

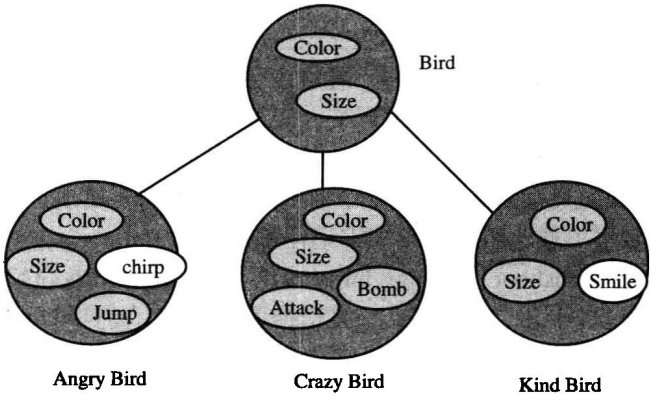


图 1-2 类的继承示意图

1.2.3 多态

多态，即在一个程序中同名的不同方法可以共存，子类的对象可以响应同名的方法，但具体的实现方法不同，完成的功能也不同。

从字面上“多态”这个词不好理解。事实上，“多态”起源于我们现实生活。生活中常提到“打”这个动词，“打”可以组词为“打的”、“打扫”、“打架”……这就是“多态”，同一个“打”的行为有不同的反映。多态就是同一个方法可以用来处理多种不同的情形。在程序中，“多态”可以理解为父类与子类都有的一个同名的方法，针对继承关系下不同对象可以采取不同的实现方式。我们还以“愤怒的小鸟”为例。

如图 1-3 所示，三个类都具有一个相同的功能“shoot”，以射向隐藏好的小猪，Angry Bird 和 Crazy Bird 的对象，如红色鸟、小蓝鸟、白鸟，同样都具有“shoot”功能。然而它们发射后执行的方式与效果是截然不同的，小蓝鸟弹出后分离出攻击力更强的三只小鸟，白鸟弹出后会像炸弹一样爆炸，这就是程序中的多态。需要注意的是，多态是以继承为基础的，只有继承关系下的类才具有多态的特性。

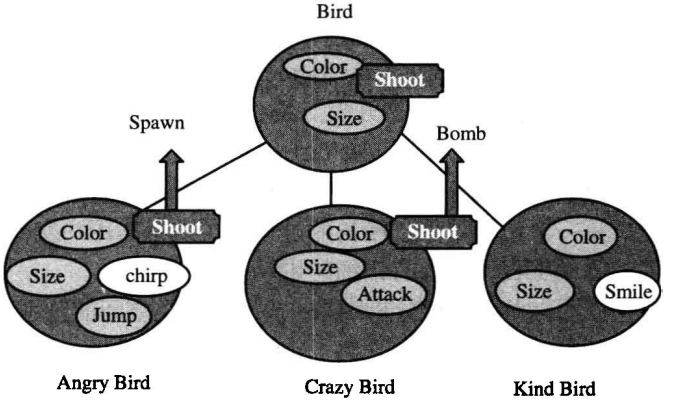


图 1-3 多态示意图

那么，多态的作用是什么呢？我们知道，封装可以隐藏实现细节，使得代码模块化；继承可以扩展已存在的代码模块（类）；它们的目的都是为了代码重用。继承虽然已经扩展了功能，但还不够丰富，多态的引入就是要在继承的基础上进一步实现变异的可能性，增强程序的可扩展性，简单地说，就是一种方法多种实现。

你可能马上会问，如果不用多态，简单地把三个 shoot 重新命名为“Birdshoot”、“Angrybirdshoot”、“Crazybirdshoot”同样也可以达到目的。没错，在继承关系层次简单的情况下多态的优越性是无法体现出来的。我们看看在继承关系复杂的情形下，假设愤怒的小鸟接二连三派生出不同的版本，如 PC 版、季节版、手机版、PC 版 2、季节版 2 等，如果同一个游戏开发小组的成员每人承担其中一个版本的工作，组长负责最后的版本整合，而每个组员都需要实现一个“shoot”的功能。试想如果“shoot”的命名不一样，组长就得牢牢记住每个人的“shoot”方法名，如“Birdshoot”、“Angrybirdshoot”、“Crazybirdshoot”、“Angrybirdshoot2”、“Crazybirdshoot3”……以在他的整合程序里调用。三四个还好说，十几二十个呢，一旦记错了名字呢，而且组员们派生出的类还在不断增加，最终组长就崩溃了。所以，他最大的希望就是大家在开发前定好规矩，都统一命名为“shoot”，然后把“shoot”的定义与具体实现分离开来。无论组里成员怎样去实现，如何扩展他们的程序，与他都没有关系，他只管关心“shoot”这个公共接口，只记住“shoot”这个词，实现的细节不必要考虑，执行时就能够自动调用所有子类的“shoot”功能。同时，组员们只管安心写自己的程序。这时，多态就是最好的解决方案。可见，多态使面向对象语言具有了灵活性、扩展性、代码共享的特征，把继承的优势发挥得淋漓尽致。

1.3 面向对象的程序设计

首先，我们必须知道面向对象的软件开发是一项巨大的工程，也是一门专业学科，仅仅依靠学习语法知识就立即进行代码的编写是无法达到开发要求的。实际上，面向对象的软件开发需要经历一个从需求分析、设计、编程、测试到维护的生命周期。面向对象的思想渗透到软件开发的各个方面，这里简要介绍面向对象软件开发的几个基本流程。

第一阶段是面向对象需求分析（Object Oriented Analysis，OOA），需要系统分析员对用户的需求做出分析和明确的描述。包括从客观存在的事物和它们之间的关系归纳出有关的类及其类之间的关系，并将具有相同属性和行为的对象用一个类来表示。

第二阶段是面向对象设计（Object Oriented Design，OOD），在需求分析的基础上，对每一部分分别进行具体设计，首先是类的设计，可能包括多个层次，利用继承和组合等机制设计出类的层次关系，然后提出程序设计的具体思路和方法。

第三阶段是面向对象编程（Object Oriented Programming，OOP），选用适当的面向对象编程语言，如 C++、C#、Objective-C 或 Java，选用开发工具，设置开发环境进行代码的编写工作。

第四阶段是面向对象测试（Object Oriented Test，OOT），对程序进行严格的测试，这个过程包括单元测试、集成测试及系统测试等。最后还要对程序进行维护管理。

面向对象的需求分析是一个比较繁琐、漫长的过程，主要是与用户交流、沟通、收集信息、整理需求的过程。相关的软件工程课程会有详细的讲解。

本节主要讲述如何根据面向对象需求分析结果，找出类与类之间的联系，用 UML 设计出类的层次结构，这部分设计是编程的基础，只有理解了类设计的主导思想，设计出清晰、合理、扩展性强的类结构，才能编写出好的面向对象程序。在实际软件开发中，开发人员在编写代码之前，进行面向对象设计工作是必不可少的步骤。

1.3.1 类的建模

通常使用 UML（Unified Modeling Language）来设计类的结构，即统一建模语言。它不是编程语言而是为计算机程序建模的一种图形化“语言”，所谓“建模”就是勾画出工程的蓝图，就像盖房子需要首先设计出房子的模型。同样的道理，软件工程“建模”就是在考虑实际的代码细节之前，用 UML 图示将程序结构在很高的层次上表示出来。UML 除了能帮助进行程序的结构设计外，还有助于理解程序的具体工作流程。因为对于大型的程序，仅仅看源代码很难搞清楚其各部分之间的联系，而 UML 提供了一种直观的方法让我们了解程序概貌，并能描述程序的主要部分、它们是如何一起工作的，以及工作的流程是怎样的。事实上，从文档管理、测试到维护，UML 在软件开发的所有阶段都是有用的。而且，从公司项目管理的角度考虑，UML 也是规范项目管理的一种行之有效的好方法。

经常使用的建模工具有 IBM 公司的 Rational Rose、Together、MyEclipse 等。当然用微软公司的 Visio 也能画出图，而且使用比较简单。UML 最重要的部分是 9 种类图。如表 1-1 所示。

表 1-1 UML 的主要类图

图 名	说 明
类图（Class Diagram）	表示类之间的关系
对象图（Object Diagram）	表示特定对象之间的关系
时序图（Sequence Diagram）	表示对象之间在时间上的通信
协作图（Collaboration Diagram）	按照时间和空间顺序表示对象之间的交互和它们之间的关系
状态图（State Diagram）	表示对象的状态和响应
用例图（User Case Diagram）	表示程序用户如何与程序交互
活动图（Activity Diagram）	表示系统元素的活动
组件图（Component Diagram）	表示实现系统的组件的组织
配置图（Deployment Diagram）	表示环境的配置

UML 有两套建模机制：静态建模机制和动态建模机制。静态建模机制包括用例图、类图、对象图、组件图和配置图，用于需求分析阶段，反映了程序的功能需求。动态建模机制包括消息、状态图、时序图、协作图、活动图，反映了程序运行过程中对象的状态，以及它们之间的交互等动态信息。

1.3.2 类的层次结构设计

类的层次结构代表了类与类之间的关系，包括有多少个类、它们之间的关系是什么、又是如何关联的等。比如“愤怒的小鸟”游戏，我们很直观地可以分析出至少有两个类：“小鸟”类和“绿猪”类。“小鸟”类还可以分成几种具有不同功能的子类，如“白鸟”类、“蓝鸟”类、“红鸟”类，而“小鸟”类的行为又影响着“绿猪”类。这就是经过面向对象设计初步得出的类的层次结构。

从程序设计的角度分析，类和类之间的关系包括泛化、依赖、关联、聚合和组合，理解这些类的关系，然后依据这些关系进行类的层次结构设计，有助于优化代码的组织结构。

1. 泛化

泛化关系就是继承，即找出现有的一个类或者若干个类的一些共同的属性或者方法，构造出一个一般类，其他的凡是具有该一般类特征并且还有自身一些特殊特征的类为特殊类。一般类和特殊类之间的关系就是继承。一般类也就是父类，特殊类都是它的子类。这种关系就是通常所说的“is-a”关系。例如，一个 Button 类继承 Control 类，那就是说 a Button is a Control，那么 Control