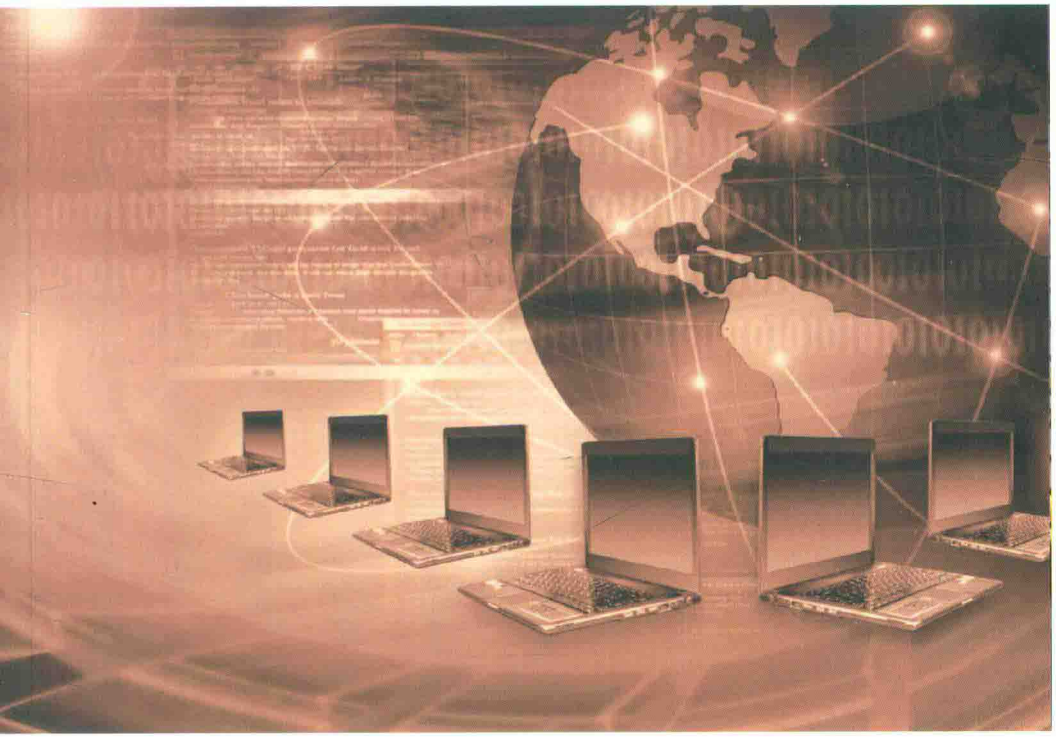
石油高等教育“十三五”规划教材



# 面向对象程序设计教程

李文超　赵新慧　编著

中国石油大学出版社
CHINA UNIVERSITY OF PETROLEUM PRESS


石油高等教育“十三五”规划教材

# 面向对象程序设计教程

李文超　赵新慧　编著

中国石油大学出版社
CHINA UNIVERSITY OF PETROLEUM PRESS

**图书在版编目(CIP)数据**

面向对象程序设计教程 / 李文超，赵新慧编著 . —
东营：中国石油大学出版社，2016. 8
ISBN 978-7-5636-5297-6

Ⅰ. ①面… Ⅱ. ①李… ②赵… Ⅲ. ①面向对象语言
—程序设计—教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2016)第 177188 号

石油高等教育教材出版基金资助出版

书　　名：面向对象程序设计教程
作　　者：李文超　赵新慧

---

**责任编辑**：曹秀丽(电话 0532—86981532)
**封面设计**：青岛友一广告传媒有限公司

---

出 版 者：中国石油大学出版社(山东 东营　邮编 257061)
网　　址：http://www.uppbook.com.cn
电子信箱：shiyoujiaoyu@126.com
排 版 者：青岛友一广告传媒有限公司
印 刷 者：青岛炜瑞印务有限公司
发 行 者：中国石油大学出版社(电话 0532—86981531，86983437)
开　　本：185 mm × 260 mm　印张：20. 75　字数：492 千字
版　　次：2016 年 8 月第 1 版第 1 次印刷
印　　数：1—1 500 册
定　　价：50. 00 元

# PREFACE | 前　言

面向对象程序设计(Object-Oriented Programming, OOP)是一种当前非常流行的程序设计技术。与面向过程的程序设计(Procedure-Oriented Programming, POP)相比,面向对象程序设计更符合人们观察和分析问题的习惯,能够更好地描述现实世界。采用面向对象技术开发的产品具有更易于重用、易于维护、易于修改和易于扩充等优点。

本书采用 C++ 语言作为介绍面向对象程序设计的描述工具。C++ 语言是在 C 语言的基础上引入面向对象机制的一种程序设计语言。因此,C++ 语言具有全面兼容 C 语言并支持面向对象技术的特点。已有的 C 语言程序不加修改或稍加修改就能通过 C++ 编译器进行编译。在面向对象程序设计语言分类中,C++ 语言属于一种混合型面向对象程序的设计语言。

由于 C++ 语言既支持 C 语言的代码风格,又支持面向对象程序设计技术,所以对于初学程序设计的人来说,学习 C++ 语言就可以同时熟悉传统的结构化程序设计和面向对象程序设计两种方法。这是本书选择 C++ 语言作为介绍面向对象程序设计技术的程序设计语言的主要原因。更重要的是,学习面向对象程序设计可以掌握面向对象程序设计思想,使它们在具体的语言实现机制中体现出来。

本书以 C++ 语言作为描述语言,所以它也可以作为学习 C++ 语言的教材。如果受学时的限制或者想以本书作为 C++ 语言的教材,那么面向对象程序设计方法的内容可以作为学生自学和提高的参考,而不必列入授课计划。

目前,国内已有多种 C++ 语言学习教材,但其中大部分是先从 C++ 面向过程编程讲起的,而这些内容实际上是与 C 语言的内容重叠的。为方便读者的学习,本书直接讲述面向对象的内容,更为精炼和实用。

全书按照由浅入深的顺序,共分 12 章。第 1 章是绪论,介绍面向对象程序设计的基本原理和思想;第 2 章介绍从 C 语言到 C++ 语言,讲述 C++ 语言对 C 语言的扩充;第 3 章介绍类与对象,包括类的定义与封装,构造函数、析构函数,对象赋值、复制和对象指针等;第 4 章介绍类与复杂对象,包括堆对象、const 特性、静态成员、友元和类型转换等;第 5 章介绍继承和派生类;第 6 章介绍多态性和虚函数;第 7 章介绍运算符重载,并给出几个典型运算符的重载方法;第 8 章介绍模板,包括类模板、模板类和 STL 模板等;第 9 章介绍 C++

中各种 I/O 流的使用；第 10 章介绍 C++ 中的异常处理；第 11 章介绍面向对象的设计和实现；第 12 章通过一个综合的实例将面向对象知识贯穿起来。其中，第 2～10 章包含上机实训，以便于读者通过理论学习和实践两个环节更好地掌握课程内容，提高编程能力。本书中的所有例题均在 Visual Studio 2010 下调试完成。为便于读者学习，每章开始均配有教学提示，介绍本章中应该重点掌握的内容。

全书主要由辽宁石油化工大学李文超、赵新慧编著，杨妮妮、石元博和王宏亮等也参与了部分章节的编写工作。本书参考和引用了大量的文献资料，在此向相关作者表示衷心的感谢。同时，感谢中国石油大学出版社“石油高等教育教材出版基金”和辽宁石油化工大学教务处的支持。

本书的读者对象是大学本科计算机相关专业的教师和学生，同时也可以作为计算机相关领域技术人员的参考书。

由于作者水平有限，不妥之处在所难免，欢迎读者批评指正。

作　者

2016 年 3 月

CONTENTS | 目 录

# 第1章

# 绪 论

**教学提示**

本章主要介绍传统的结构化程序设计和面向对象程序设计的特点和区别，回顾结构化程序设计方法、面向过程的程序的结构和特点等内容，重点介绍面向对象程序设计的基本概念、原理和方法。其中，面向对象的抽象原则、类、对象、消息、继承、多态等内容是面向对象程序设计的核心概念。正确理解这些概念，了解面向对象程序的构造及其设计语言的特点，对后续章节的学习是非常重要的。

随着计算机硬件技术的不断发展，程序设计方法也在不断地改进和发展。在计算机发展的早期，评价程序好坏的标准是程序指令的多少、运行速度的快慢以及占用内存的多少等。当时的程序员过分追求编程的技巧，往往只将注意力集中在问题求解本身，无暇关注求解的过程，更谈不上考虑程序结构的合理性和可指导性。由于软件开发单纯依赖于程序员的个人经验，缺乏科学的理论和方法做指导，在大型软件的开发过程中出现了复杂程度高、研制周期长和正确性难以保证等三大难题。遇到问题时找不到解决办法致使问题堆积起来，就形成了人们难以控制的局面，最终在20世纪60年代末出现了所谓的“软件危机”。为了克服这一危机，一方面需要对程序设计方法、程序的正确性和软件的可靠性等问题进行系统研究；另一方面，也需要对软件的编制、测试、维护和管理的方法进行研究，这就产生了程序设计方法学。在这些程序设计方法中最值得关注的包括面向过程的结构化设计方法和面向对象的程序设计方法。

## 1.1 面向过程的程序设计方法

### 1.1.1 结构化程序设计方法

1965年，E.W.Dijikstra提出了结构化程序设计(Structured Programming)的概念，这是软件发展的一个重要里程碑。在这种思想的影响之下，很快涌现出了一些优秀的程序设计语言，例如PASCAL，C和Ada等。结构化程序设计方法是为了使程序有良好的结构并易

读、易懂而推出的一种科学的程序设计方法。它强调程序结构的规范性，主张采用自顶向下、逐步求精的程序设计方法。结构化程序设计方法是20世纪七八十年代最流行的程序设计方法，人们据此开发出许多有实用价值的软件，解决了许多实际问题。结构化程序设计的设计思想和核心问题包括自顶向下、逐步求精，模块化和语句结构化等。

1）自顶向下、逐步求精

“自顶向下，逐步求精”是Niklaus Wirth提出的设计策略。按照这种策略，一个复杂的问题可以分解为由若干个子问题（模块）组成的层次结构。在最高的抽象层次上，可以使用问题所处环境的语言概括地描述问题的解法，然后对各个层次的过程细节和数据细节逐层细化，直到用程序设计语言的语句能够实现为止，从而最后确立整个体系结构。

根据自顶向下的观点，遇到一个问题后不要试图一下子就触及问题解法的细节，而应当先从问题的全局出发，进行宏观需求分析，确定“做什么”以及怎样把问题分解为几个子问题或子功能。然后在子问题一级描述算法，这是对全局算法的细化。这种细化过程可能要逐步做下去，直到细化的模块能知道具体“怎么做了”，即能写出相应的程序为止。逐步细化的基本思想是把一个复杂问题的求解过程分阶段进行，把每个阶段处理的问题都控制在人们容易解决的范围内。在进行下一层的细化前，对本阶段的正确性进行检查，及时修改存在的问题。

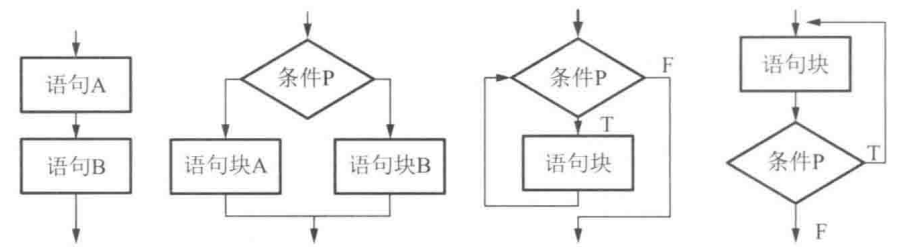
例如，开发一个高校学生教材购销系统，最初的抽象是把整个系统作为一个整体，研究它和周围外部实体之间的数据交换；接下来的下层抽象是把该系统分为学生教材销售和学生教材采购两个子系统；最后的抽象是把学生教材销售子系统分为有效性审查、账目处理、开发票、开领书单等部分，而把学生教材采购子系统分为缺书登记、进书通知等部分。当细化到一定程度时，就可以使用程序设计语言的语句实现了。

2）模块化

结构化程序的结构是将分析设计阶段的每个子问题或功能模块通过相应的子程序（函数或过程）的代码来实现。程序的主体是子程序层次库，它与功能模块的抽象层次相对应，编码原则使得程序流程简洁、清晰且可读性增强。模块之间通过接口传递信息，每一个模块完成一个单一的功能。模块的基本特征是“相对独立，功能单一”。模块的划分尽可能做到高内聚、低耦合。模块化的设计思想在一定程度上有助于程序代码的复用。

3）语句结构化

结构化程序设计的每一个模块在实现的时候只使用三种基本的控制结构，它们分别是顺序结构、分支结构和循环结构，如图1-1所示。这三种结构的共同特征是：它们都只有一个入口和一个出口。




（a）顺序结构　（b）分支结构　（c）当型循环结构　（d）直到型循环结构

图1-1　结构化程序控制结构

1966年，G. Jacobini 和 C. bohm 从理论上证明，任何程序都可以利用顺序结构、分支结构和循环结构表示出来。结构化程序设计限制 Goto 语句的使用。图灵奖获得者 E. W. Dijkstra 在 1968 年指出：Goto 语句是有害的，它造成了程序结构的混乱，高级语言程序设计应取消 Goto 语句。从此程序设计开始由讲究效率向讲究良好的结构转变。

### 1.1.2 面向过程的程序结构

在面向过程的程序设计中，过程（或函数）是程序的基本构造块。结构化程序在每一个过程中使用顺序、循环、分支三种基本控制结构。面向过程的程序结构具有树状或网状的形式，其中的节点是过程（或函数），它们之间的关系是调用，即一个过程调用其他过程。调用通过过程的接口进行，而接口是按照需求事先定义的。用过程性语言编写的程序结构如图 1-2 所示，其中每个矩形表示一个过程。

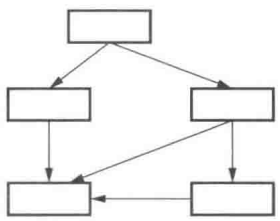


图 1-2 过程性程序结构示意图

显然，过程或函数是结构化程序设计中模块化思想的具体体现。而树状结构反映了自顶向下、逐步分解的层次关系。有人用公式：

程序 = 过程 + 过程调用

来表达结构化程序设计的特点是有道理的。下面是一个过程调用的示例。

【例 1-1】函数与函数调用的示例。

```
float func1 (float X, float Y) {
    float Z;
    Z = X + Y/2;
    return Z;
}
main (  ){
    float a, b, c;
    scanf ("% f, % f", & a, & b);
    c = func1 (a, b);
    printf ("% f", c);
}
```

## 1.2 面向对象的程序设计方法

面向对象程序设计方法是在面向过程程序设计方法的基础上发展起来的。

### 1.2.1 面向对象程序设计方法的产生

早期的计算机主要用于科学计算，为了完成计算，就必须设计一个算法或者解决问题的过程。但是随着计算机硬件的不断发展，计算机的性能越来越强，应用的领域也越来越广泛。随着软件规模的不断增长，面向过程的程序设计方法逐渐暴露出它的不足和缺陷。

（1）面向过程的程序设计在解决问题时是以代码为中心，用计算机观点进行程序设计。这种开发软件的方法和过程与人类认识世界、解决问题时采用的方法和过程明显不同。

（2）面向过程的程序设计是基于数据类型和过程的定义，它把数据和处理数据的过程

分离为相互独立的实体。当数据结构发生变化的时候，所有相关的处理过程都要进行相应的修改。另外，这种程序设计方法缺乏对数据和代码的保护机制。如图 1-3 所示，程序中的全局变量可以被这个过程访问，也可以被另一个过程访问。如果一个外部过程能够轻易地读写只能被特定过程读写的数据和过程，就会给程序设计带来不安定的因素。

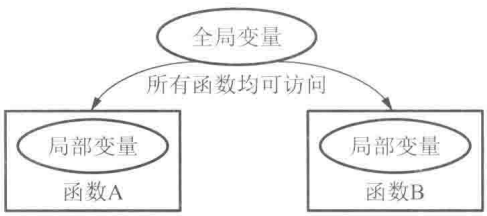



图 1-3 函数通过全局变量对彼此的影响

（3）面向过程的程序设计方法代码只能提供有限的重用性，可维护性差。每当程序员开发一个新系统时都要从零开始，针对具体的问题做大量重复而又烦琐的工作。即使要复用以前项目中的代码，也只能进行简单的拷贝。这种代码复用机制很难适应越来越大、越来越复杂的软件的开发。

### 1.2.2 基本概念

面向对象程序设计是将数据与对数据的操作封装在一起，使之成为一个不可分割的整体，同时将具有相同特征的对象抽象成一种新的数据类型——类。面向对象的基本组成单位就是类。程序在运行时由类生成对象，对象之间通过发送消息进行通信，相互协作完成相应的功能。对象是面向对象程序的核心。现实世界中万物皆对象，都具有各自的熟悉感，都对外界呈现各自的方法。程序中对象都具有标识、属性和方法，通过一个或多个变量来保存其状态，通过方法实现它的方法。将属性及方法相同或相似的对象归为一类可以看成是对对象的抽象，代表了此类对象所具有的共有属性和方法。在面向对象的程序设计中，每一个对象都属于某个特定的类。

1）对象

对象是面向对象程序设计时的基本单元。对象是对客观世界中实际存在的某种事物的抽象，即是描述客观事物的一个实体。对象既可以是一个有形的具体事物，例如一个人、一本书、一张桌子等，也可以是无形的、抽象的事件，例如一首歌、一场比赛、一门课程等。

对象是研究问题和分析问题的出发点。面向对象语言利用对象名、属性和行为三个要素来描述对象。对象名用来标识一个具体的对象；属性是用来描述对象静态特征的一个数据项，代表对象的状态；行为是用来描述对象动态特征和行为的一个操作，代表对象对外提供的功能。对象由类描述，并通过类实例化得到。以下是一个电灯对象的例子。

| 对象的属性： | 属性名 | 属性值 |
| --- | --- | --- |
| | 功率 | 20 W |
| | 状态 | 关 |

对象的行为：开灯、关灯、读取瓦数。

2）类

类是对现实世界中一组具有相同属性和行为的对象的抽象。例如客厅的吊灯、书房的

台灯和床边的落地灯，它们都是不同的电灯对象，但它们都具有相同的特征，都有功率和状态等属性，都能进行开灯和关灯等行为。将所有灯都共有的属性和行为抽象出来，就构成了一个电灯类。图 1-4 所示为从所有电灯对象中抽象出来的电灯类。

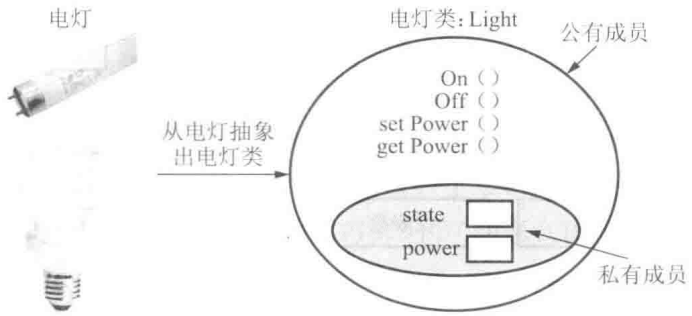



图 1-4 电灯类

类是面向对象程序设计中非常重要的一个概念，它为属于该类的全部对象提供统一的抽象描述。类是生成对象的模板，对象是类的实例。在面向对象程序设计语言里，类是一种数据类型，对象是这种数据类型的实例。类里的属性一般是私有成员，只能在类内使用，不能在类外使用。类里的方法（表示行为）一般是公有成员，可供类外调用。一个类的不同对象一般具有不同的属性值。例如，客厅的吊灯和书房的台灯，它们的属性值是不同的，如图 1-5 所示。

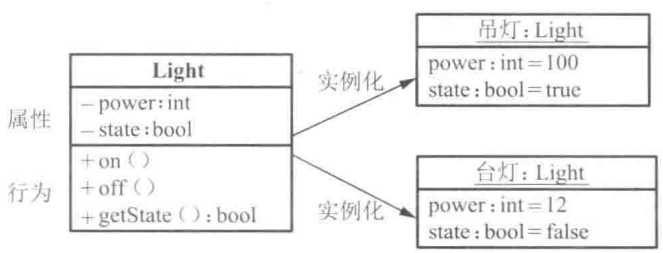



图 1-5 类与对象的关系

类可以再分成不同的层次。为类分层的标准之一是包含关系，用包含关系把类分成多个层次，并分别用父类和子类来称呼有直接包含关系的类。分了层次的类之间就建立了一个层次结构。

类之间除了包含关系外，还有关联、聚合和泛化三种关系。这些关系是把作为面向对象程序构成基本单元的类连接起来以便形成系统的黏合剂。确定系统中含有的类，再确定各个类之间的各种关系，就是构造一个面向对象程序的两大主要任务。

关联关系（Association）用来描述类之间的一种特定联系。例如，有一个学生类（Students），还有一个课程类（Courses），学生可以选修课程，这样在学生类和课程类之间就存在一个“学生选修课程”的关联关系。

聚合关系（Clustering）用来描述整体和部分之间的联系。例如，车轮、底盘、发动机、仪表盘都是汽车的组成部分。在程序设计中，可能有汽车类、车轮类、底盘类、发动机类和仪表盘类，其中汽车类与其他各类之间存在聚合关系，汽车类是标识整体的类，而其他各类是标识部分的类。聚合关系如图 1-6 所示。

泛化关系（Generalization）用来描述一般和特殊的联系。例如，如果称为人员的类具有

一般人员共同属性和行为，那么教师、教授、经理都是特殊的人员，于是在人员类(Person)和教师类(Teacher)之间，教师类和教授类(Professor)之间，人员类和经理类(Manager)之间都是泛化关系，如图 1-7 所示。

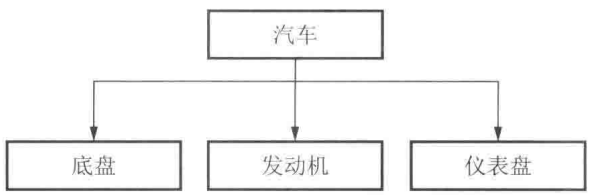



图 1-6 聚合关系的示例

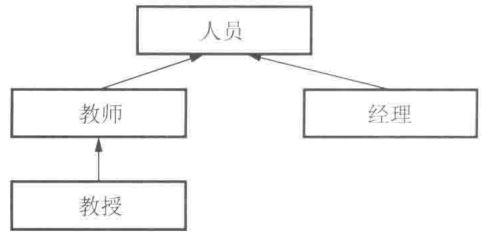



图 1-7 泛化关系的示例

3）消息

在面向对象程序设计中，对象之间的联系是通过消息传递来完成的。一个消息就是一个对象对另一个对象发出的“请求”或“命令”。接收者收到消息后，调用有关的方法，执行相应的操作。

对于不同的程序设计语言，消息可能存在细微的差别。一般地，一条消息中应该包含消息的请求者信息、要求接收者执行什么操作的信息(但不说明应该如何去做)。消息中含有的关于操作的信息引发接收者的处理，相当于一次过程调用。

请求者发送消息，接收者响应消息是面向对象程序工作的基本方式。因此可以说，消息是驱动面向对象程序运转的源泉。在面向对象程序系统中，消息分为两类：私有消息和公有消息。私有消息是对象发给自身的消息，通常表现在对象的行为在执行过程中调用了一个自身私有的行为操作。私有消息对外是不公开的。公有消息是外部对象发送给这个对象的消息。

### 1.2.3 面向对象程序设计的基本特征

1）抽象性

抽象是人类认识问题、解决问题的基本手段。由于客观世界是复杂的，解决任何问题时，降低问题的复杂程度是首要的任务，而抽象是降低问题复杂度的最佳途径。抽象是面向对象程序设计的要素。利用抽象，从众多事物中抽取出共同的、本质的特征，忽略次要的和非本质的特征。例如考虑电灯这个事物时，位于不同位置的 5 盏灯是 5 个对象，这 5 个对象拥有瓦数和状态这些共同的属性(只不过取值不同而已)，都具有开灯、关灯和获取瓦数的行为。在这种情况下，就可以将这 5 盏灯抽象为电灯类，每盏灯都是电灯类的不同对象。

一般来说，对一个事物的抽象应该包括两个方面：数据抽象和行为抽象(也称功能抽象)。数据抽象描述某一类对象的属性和状态，也就是一类对象区别于另一类对象的特征。例如，建立一个电灯类时，电灯会有如下特征：瓦数、电灯当前状态等，这些特征就将成为类的属性。行为抽象描述的是某一类对象的共同行为或功能。例如对于电灯可以进行开灯、关灯和查看电灯瓦数等操作，定义电灯类时就可以将这些操作抽象为类的方法。

2）封装性

封装(Encapsulation)就是把类(对象)的属性和行为结合成一个独立的单位，并尽可能

隐蔽类(对象)的内部细节。封装有两个含义:一是把类(对象)的全部属性和行为结合在一起,形成一个不可分割的独立单位。对象的属性值(除了公有的属性值)只能由这个对象的行为来读取和修改。二是尽可能隐蔽类(对象)的内部细节,对外形成一道屏障,与外部的联系只能通过外部接口实现。例如,一台袖珍收音机,它的部件封装在一个盒子内,外部所见的只是一些开关和旋钮。如果我们希望收听某个电台的节目,只需在相应的开关和旋钮上进行相应的操作就行了,不必知道它的内部构造。而且,如果我们不动任何开关和旋钮,原则上收音机的状态就不会改变,也就是说,它的状态改变需要外界的激发,并且这种改变是它内部机制执行的结果。封装的信息隐蔽作用反映了事物的相对独立性,可以只关心它对外所提供的接口(即能做什么),而不注意其内部细节(即怎么提供这些服务)。图 1-8 给出了电灯类对数据的封装和隐藏。

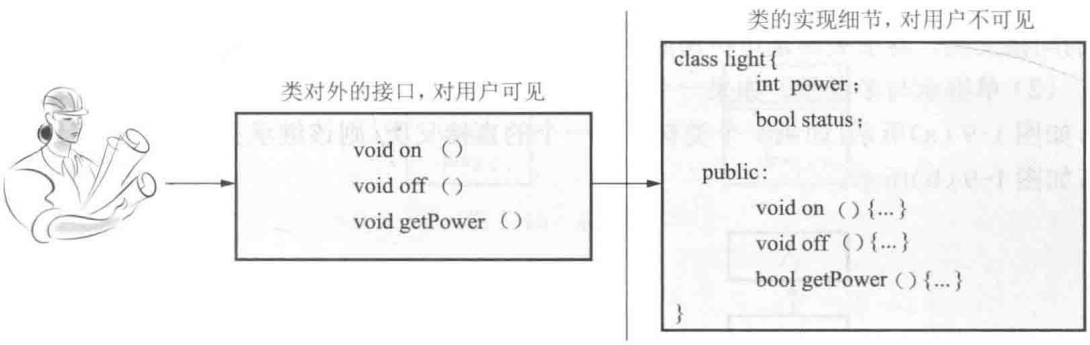



图 1-8 电灯类的数据封装和隐藏

封装的结果,一方面使对象以外的部分不能随意存取对象的内部属性,从而有效地避免了外部错误对它的影响,大大减小了查错和排错的难度;另一方面,当对象内部进行修改时,由于它只通过少量的外部接口对外提供服务,因此同样减小了内部的修改对外部的影响。同时,如果一味地强调封装,则对象的任何属性都不允许外部直接存取,要增加许多没有其他意义而只负责读/写的行为。这为编程工作增加了负担和运行开销,并且使程序显得臃肿。为了避免这一点,在语言的具体实现过程中应使对象内部有不同程度的可见性,进而与客观世界的具体情况相符合。

封装性的引入为程序员在使用面向对象技术编程过程带来了以下便利。

(1) 封装性使得两个对象在协作时仅仅依靠彼此提供的接口进行,而不必关心内部实现的细节,提高了软件复用。

(2) 封装性降低类与类之间的耦合度,功能的内聚使得软件更易于维护。

(3) 封装性增强了信息的隐藏性和安全性。

当然,为了保证封装性,有时不得不增加更多的方法用于对数据的存取。

3) 继承

继承(Inheritance)是一种联结类与类的层次模型。继承性是指特殊类的对象拥有其一般类的属性和行为。继承意味着“自动地拥有”,即特殊类中不必重新定义已在一般类中定义过的属性和行为,而它却自动地、隐含地拥有其一般类的属性与行为。继承允许和鼓励类的重用,提供了一种明确表述共性的方法。一个特殊类既有自己新定义的属性和行为,又有继承下来的属性和行为。继承下来的属性和行为尽管是隐式的,但无论在概念上还是

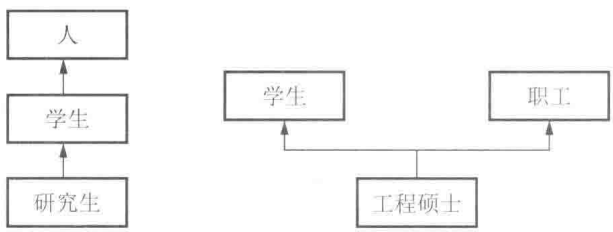
在实际效果上都是这个类的属性和行为。当这个特殊类又被它更下层的特殊类继承时，它继承来的和自己定义的属性和行为又被下一层的特殊类继承下去。因此，继承是传递的，体现了大自然中特殊与一般的关系。

在软件开发过程中，继承性实现了软件模块的可重用性、独立性，缩短了开发周期，提高了软件开发的效率，同时使软件易于维护和修改。这是因为要修改或增加某一属性或行为，只需在相应的类中进行改动，而它派生的所有类都自动地、隐含地做了相应的改动。由此可见，继承是对客观世界的直接反映，通过类的继承能够实现对问题的深入抽象描述，反映出人类认识问题的发展过程。

继承关系可以分为两种形式：

（1）直接继承和间接继承。如果类 C 继承于类 B，则称类 B 为类 C 的直接父类，类 C 为类 B 的直接子类；如果同时类 B 直接继承于类 A，则称类 C 间接继承于类 A，类 A 为类 C 的间接父类。继承关系是可传递的。

（2）单继承与多继承。如果一个类只有一个直接父类，则该继承关系被称为单继承关系，如图 1-9（a）所示；如果一个类有多于一个的直接父类，则该继承关系被称为多继承关系，如图 1-9（b）所示。




（a）单继承　　（b）多继承

图 1-9　继承关系

4）多态

多态是面向对象设计的另一个重要特征。当一个对象发出消息时，由于接收对象的类型可能不同，所以它们可能做出不同的反应。这样，一个消息可以产生不同的响应效果，这种现象称为多态性。简单地说，多态就是“单个接口，多种实现”。利用多态性用户可以发送一个通用的消息，把实现的细节留给接受者决定，于是用同一个消息可以调用不同的方法。

考虑一个乐谱编辑器中包含了休止符（Staff）、全音音符（WholeNote）和半音音符（HalfNote）等音符，它们都有共同的性质，例如一个屏幕位置，一个可以被画在五线谱上的形状，可以被鼠标拖拽到其他位置，如图 1-10 所示。它们对共同的消息绘制（draw）就会有不同的响应。例如，全音音符对象接收到 draw 消息后，会在五线谱当前的位置上绘制一个全音符号；半音音符对象在接收到同样的消息后，在五线谱上绘制出的是一个半音符号。显然，绘制全音音符和绘制半音音符需要不同的实现，但是它们可以被同一条消息 draw 引发，这就是多态性。

多态性是面向对象程序设计方法的精髓，也是面向对象程序设计方法最有特色的特征之一。它增加了程序设计的灵活性，是面向对象程序设计实现代码重用的重要特征。多态性把程序设计从关注如何实现一个行为转到关注对象提供什么抽象行为这一观点上来。

面向对象程序设计通过继承和重载两种机制实现多态。继承性和多态性的结合可以生成一系列虽类似但独一无二的对象。由于继承性，这些对象共享许多相似的特征；由于多态性，针对相同的消息，不同对象可以有独特的表现方式，实现特性化的设计。

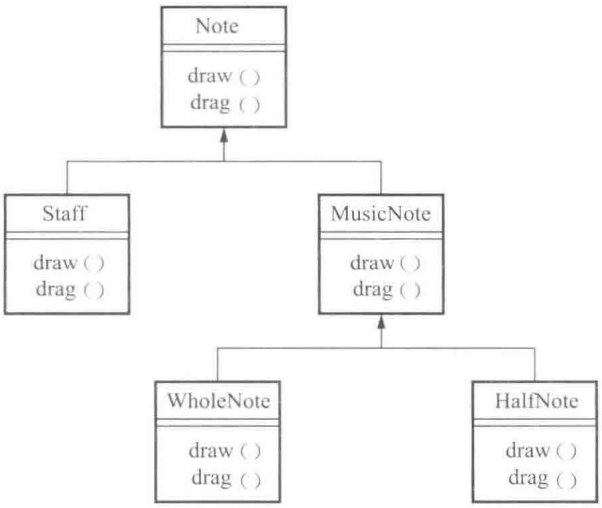



图 1-10 多态性

### 1.2.4 面向对象程序举例

面向过程的程序是由过程或函数构成的，而面向对象的程序一般是由类构成的。我们在设计程序时，首先想到的是这个程序需要哪些类，所以面向对象的程序设计实际上是面向类的程序设计。

由于面向对象的程序设计中，“对象”是一个核心概念，所以先弄清楚“对象”在程序设计过程中如何使用就显得十分重要。其实，在面向对象程序设计中，对象一词的使用在不同场合有不同的含义。在问题域分析阶段，“对象”指的是客观世界中的事物；在程序设计阶段，“对象”指的是一组事物的集合，这个集合在实现阶段被编码为类；在程序运行阶段，“对象”指的是以类为样板、在需要时才创建且在运行时存在的实体，因此这时的“对象”是类的实例。

在面向对象程序中，我们只见到类。在程序中，类要用程序设计语言提供的语句逐一声明和定义，但类是只在程序编译时存在的实体，是支持数据封装的工具。声明类的目的是建立对象。对象实现数据封装，并在程序运行时承担具体的计算任务。对象具有状态，它们由该对象属性的值决定，程序运行的过程在一定意义上是改变对象状态的过程。因此，在程序运行时，只有对象。所以，面向对象程序设计的首要任务是决定系统中应该包含哪些类，以及每个类的结构和它们能产生的对象。下面的例子说明了面向对象程序的这一特征。

【例 1-2】面向对象程序结构的示例。

```
#include<iostream.h>
class ClassA
{
  int i;
  public:
```

```
    ClassA (int x)
{
      i = x;
}
    void show ( ) ;
}
void ClassA: : show ( )
{
  cout<<i>>endl;
}
void main ( )
{
  ClassA obj (10), *p;
  p = & obj;                                    //p 指向对象 obj
  p -> show ( ) ;
  (*p).show ( ) ;
}
```

在这个程序中，语句 class ClassA 下面定义了一个称为 ClassA 的类，它有数据项 i 和函数 show ( )。而在 main ( )函数里定义了一个对象 obj (10)，还有一个指针对象 p。可以看出，主要的工作都是在类的定义里完成的。

类之间的继承、聚合和泛化关系用类构建了一个系统结构，因此确定这些关系是形成系统的关键，也是构造程序系统的关键。定义了一个个单独的类并不构成系统，只有确定了这些类之间的联系才能使之成为系统。有了系统结构，系统各部分才能协同工作，完成希望的任务。

对象交互是面向对象程序设计的关键所在。面向对象程序设计使用消息传递的方式实现对象之间的交互。消息的发送者通过消息提出请求，一个请求是一个操作调用，消息中确定了消息的接收者应该执行的操作，接收者做出响应，响应是通过完成操作调用而实现的。对象所能提供的操作以接口的形式描述，这些操作接口又称为协议。所以，仔细定义接口是程序设计的又一个重要工作。它们的具体实现方法和技术将在后续章节中依次加以介绍。

面向对象的程序看上去是一些类和消息的组合，这和面向过程的程序有显著不同。

## 1.3 面向对象的程序设计语言

1) Simula 67

1967 年挪威科学家 Ole-Johan Dahl 和 Kristen Nygaard 正式发布了 Simula 67 语言。它引入所有后来面向对象程序设计语言所遵循的基础概念(对象、类和继承)，奠定了面向对象程序设计方法的基础。因此，Simula 67 被公认为是最早的面向对象程序设计语言，最初主要用于仿真建模，在计算机上模拟离散事件。

虽然 Simula 67 因为比较难学、难用而没有得到广泛流行，但是它的问世对面向对象概念的形成产生了巨大而深远的影响。在 Simula 67 的影响下产生的面向对象技术迅速传播开来，并在全世界掀起了一股面向对象程序设计的热潮，至今盛行不衰。Simula 67 出现时正值第一次软件危机爆发，面向对象程序设计在软件开发领域引起了巨大的变革，极