HZ BOOKS
华章教育

高等院校计算机教材系列

面向对象程序设计
C++版
第2版

钱丽萍 汪立东 张健 编著

机械工业出版社
China Machine Press

高等院校计算机教材系列

面向对象程序设计

C++版

第2版

钱丽萍 汪立东 张健 编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

面向对象程序设计：C++ 版 / 钱丽萍等编著 . —2 版 . —北京：机械工业出版社，2015.10
（高等院校计算机教材系列）

ISBN 978-7-111-51903-4

I. 面… II. 钱… III. C 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（2015）第 252952 号

本书系统地讲解了面向对象程序设计的基本理论和基本方法，阐述了用 C++ 语言实现面向对象基本特性的关键技术。全书利用详实的程序实例，力图使读者在形成面向对象程序设计思维方法的同时，掌握面向对象程序设计语言 C++。

全书分为 12 章，内容包括：面向对象方法学导论、C++ 语言基础一、C++ 语言基础二、封装性、继承性、运算符重载、多态性、模板和 STL、异常处理、输入 / 输出流、Windows 编程初步知识以及综合设计与实现。

本书是在总结作者多年面向对象程序设计类课程（C++）教学经验的基础上编著而成的，各个知识点都密切结合例子展开讲解，并设计了一个贯穿全书各章节内容的实例。为方便读者复习和实践所学知识点，本书还配备有大量相关习题和实验。

本书文字通俗易懂，内容系统全面，既可作为高等院校本科生的面向对象程序设计类教材，也可以作为面向对象程序设计和 C++ 语言自学者的参考用书。

出版发行：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：张梦玲　　责任校对：董纪丽
印　　刷：北京文昌阁彩色印刷有限责任公司　　版　　次：2016 年 1 月第 2 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：16
书　　号：ISBN 978-7-111-51903-4　　定　　价：35.00 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　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

前　言

面向对象的程序设计方法是目前主流的程序设计方法，面向对象语言支持的面向对象特征包括封装（类与对象）、继承与多态。基于这些特征，用面向对象程序设计的方法很容易实现软件工程的重用性、灵活性和扩展性等主要目标。

C++ 语言是由 C 语言演化而来的面向对象的语言，由于 C 语言在程序设计语言中有着重要地位，所以 C++ 语言理所当然地成为一门重要的面向对象的程序设计语言。几乎所有的高校学生都要学习 C 语言，从 C 语言过渡到 C++ 的学习是一个自然的过程，因此大部分高校都将 C++ 语言作为面向对象的第一门学习语言。

通过本门课程的学习，学生应掌握 C++ 语言的语法，但又要避免使本门课程成为单纯的语言类教学。学生要探索如何在语言的学习过程中理解面向对象特征，培养面向对象的思维能力，这也是本书编写的目的。为此，编者在经过多年的教学实践后，搜集、整理并优化有关面向对象课程教学的经验，对本书进行了再版。

再版后，本书由《面向对象程序设计：C++ 版（第 1 版）》的 11 章增加至 12 章，将第 1 版[⊖]的第 2 章“C++ 语言基础”分成了“C++ 语言基础一”和“C++ 语言基础二”。C++ 语言基础一是 C++ 沿袭的 C 语言语法知识的浓缩，学过 C 语言的读者可以略过这章或将本章作为参考；C++ 语言基础二是 C++ 语言在 C 语言基础上新增的语言基础知识点。第 2 版将第 1 版中面向对象分析和设计阶段得到的类及类间关系图换成了 UML 中定义的标准图。另外，第 2 版对 STL 章节进行了更详细的介绍，增加了如何将不同对象存入容器并实现对象操作之类的例题，突出应用型本科教学重点；在综合设计与实现章节增加了综合性实例的开发，突出讲解面向对象基本特征在实际问题中的应用。各章节统一了编程规范，增加了例题，改进了习题和实验，增强了教学中学生不易理解章节的细节描述，书中部分未给出运行结果的例题的运行结果可从 http://hzbook.com 上下载。

由于编者水平所限，书中不妥之处敬请读者批评指正。

在本书的编写过程中，得到各位同仁的关心，并得到北京建筑大学重点教材经费的支持。此外，北京建筑大学计算机系 13 级、14 级以侯一爽为代表的同学们在试用本书的过程中提出了许多宝贵的修改意见，在此一并提出感谢。

本书配有 PowerPoint 演示文稿和例题源代码，所有例题可以在 Visual C++ 环境和 Dev C++ 环境下运行。

感谢阅读本书的读者！

编　者

2015 年 11 月

⊖ 《面向对象程序设计：C++ 版》(第 1 版)，由机械工业出版社出版。——编辑注

目　录

第 1 章　面向对象方法学导论

面向过程程序设计方法与面向对象程序设计方法在本质上就不同。面向过程程序设计方法与计算机的工作过程是完全吻合的，而面向对象程序设计方法与人类思维习惯相吻合。本章首先用一个简单的计算圆面积的例子对比说明用面向过程程序设计方法和面向对象程序设计方法解题的不同，然后着重介绍了面向对象程序设计方法的基本概念、面向对象分析和设计的过程。

1.1　面向过程的程序设计方法

1.1.1　计算机的工作原理

通用计算机由五大部件组成：控制器、运算器、存储器、输入设备和输出设备。这五大部件间的联系如图 1-1 所示。

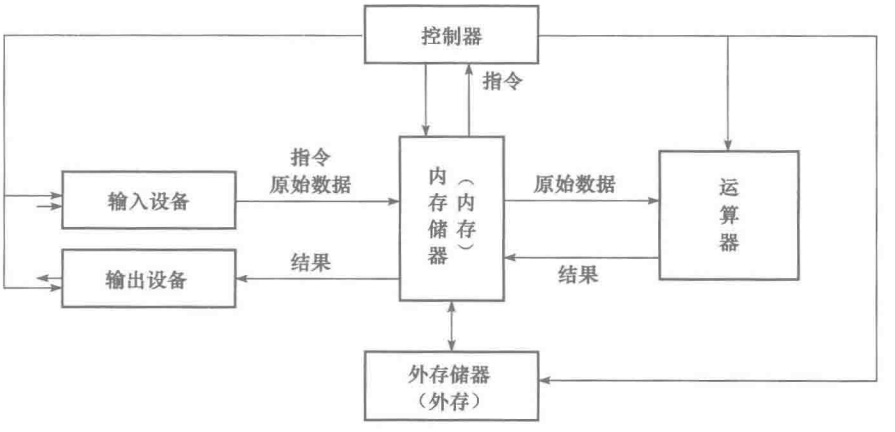

图 1-1　通用计算机的组成

通用计算机的五大部件在指令的控制下，协同工作，共同完成特定的计算任务，其工作过程如下：指令和原始数据从输入设备输入到计算机内存中，运算器从内存中读取原始数据，控制器从内存中读取指令，以控制各个部分协调一致地完成运算，并将结果存于内存中（外存储器用于扩展内存的容量及作为持久存储），输出设备从内存中取出结果并输出。通用计算机的这种架构是由数学家冯·诺依曼（Von Neumann）提出的，它的主要工作机理是“存储程序并逐条执行”。

输入计算机的信息分成两大类：数据（Data）和程序（Program）。所谓“数据”指的是被处理的对象，而“程序”是指示计算机如何工作（处理数据）的一连串指令，即用于控制计算机完成特定任务的指令序列。

1.1.2 面向过程程序设计方法

计算机的工作过程可以归结为：输入→运算→输出，其中最重要的部分是如何进行运算（解题）。计算机在程序的控制下解题，解题的方法称为算法，将算法用计算机语言实现就得到了程序。面向过程程序设计的方法与计算机的工作过程是完全一致的，即首先要明确程序的功能，程序设计的重点是如何设计算法和实现算法。在面向过程程序设计中，通常可以采用流程图来描述算法。

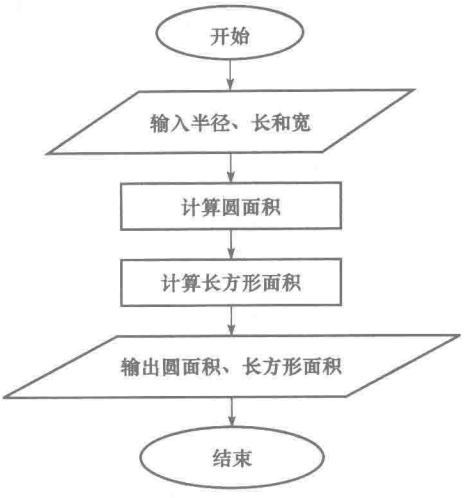

图 1-2 计算圆和长方形面积的流程图

【例 1-1】 计算圆和长方形的面积。

设 r 是圆的半径，π 是圆周率的近似值（取 3.14），则圆面积 s_1 的计算公式为 $s_1=\pi \times r \times r$；设长方形的长为 l，宽为 w，则长方形面积 s_2 的计算公式为 $s_2=l \times w$。计算圆和长方形面积的算法可以用图 1-2 中的流程图来描述。

计算圆和长方形面积的程序标准 C 语言实现如下：

```
#include <stdio.h>
int main()
{
   float r, l, w;
   double s1, s2;
   printf("Input r, l, w:");
   scanf("%f%f%f", &r, &l, &w);                    //输入
   s1=3.14*r*r;                                    //运算
   s2=l*w;                                         //运算
   printf("The area of circle is:%f\n",s1);        //输出
   printf("The area of rectangle is:%f\n",s2);     //输出
   return 0;
}
```

面向过程程序设计方法实际上是从计算机处理问题的观点来进行程序设计工作：输入→运算→输出。因为计算机的工作过程是一步一步进行的，为了完成指定功能，所以必须告诉它详细的解题步骤。面向过程程序设计者需要从计算机的角度，将解题步骤用计算机语言描述出来，因此程序设计者需要变更习惯的思维方法，以贴近计算机的内部工作机理。

在面向过程程序设计中，一种普遍采用的优化方法是使用结构化程序设计方法。结构化程序设计方法的设计思路是自顶向下，逐步求精。在问题求解过程中，先进行整体规划，将一个复杂的问题按功能分解成一个个的子问题，然后对每个子问题按功能再进行细化，依此进行，直到不需要细分为止。具体程序实现时，每个子功能对应一个模块，模块间尽量相对独立，但可通过调用关系或全局变量而有机地联系起来。所有的模块都由顺序、分支和循环三种基本结构组成。在 C 语言中，每一个子模块对应设计成一个函数，各个函数及函数间的调用关系组成了程序。因此，C 语言程序员非常注重函数的编写。

【例 1-2】 例 1-1 的模块化程序设计。

根据结构化程序设计的设计思路，先对问题进行分解。例 1-1 的问题可以分成两个子问

题，即计算圆面积和计算长方形面积，而计算圆面积和计算长方形面积的工作又可以分解为三个子问题来解决，即输入、计算、输出。下面给出计算圆面积和长方形面积的程序。

```
#include <stdio.h>
void input_r(float *r){                          //输入
  scanf("%f",r);
}
double circle_s(float r){                        //计算
   return 3.14*r*r;
}
void output_circle(double s){                    //输出
  printf("The area of circle is:%f\n", s);
}
void input_lw(float *l,float *w){                //输入
  scanf("%f%f",l,w);
}
double rectangle_s(float l, float w){            //计算
   return l*w;
}
void output_rectangle(double s){                 //输出
   printf("The area of rectangle is:%f\n", s);
}
int main(){
   float r, l, w;
   float cs, rs;
   input_r(&r);
   cs=circle_s(r);
   output_circle(cs);
   input_lw(&l,&w);
   rs=rectangle_s(l, w);
   output_rectangle(rs);
   return 0;
}
```

该程序由 7 个子模块组成：输入半径子模块、计算圆面积子模块、输出圆面积子模块、输入长宽子模块、计算长方形面积子模块、输出长方形面积子模块以及主函数。程序运行从主函数 main() 开始，分别调用输入、计算、输出模块。

面向过程程序设计方法所设计的程序架构可以用图 1-3 来概括。

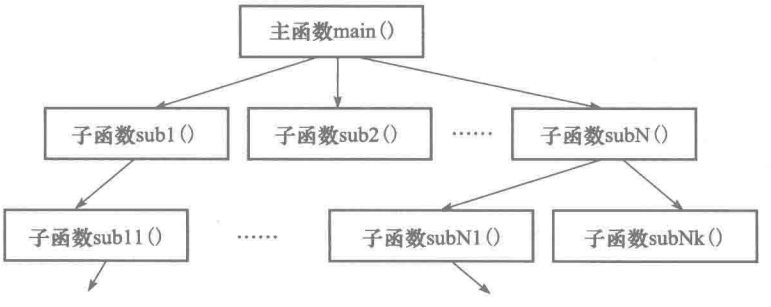

图 1-3　面向过程程序设计方法所设计的程序架构示意

结合图 1-3，可以看出：

1）面向过程程序设计方法一般适宜采用自上而下的设计方法，即先设计出主函数

main()，从整体上概括出整个应用程序需要实现的功能。在此基础上，将每项功能进一步分割成相对较小的功能模块，在实现上对应于一系列子函数，而主函数 main() 的主要任务就是完成对这些子函数的有机调用。各子函数又按同样的细化方法进一步细分。

2）由于面向过程程序设计方法需要在一开始就全面地、自上而下地设计整个应用程序的架构，所以要求程序设计者对问题域有全面的了解。对于简单的问题，面向过程的程序设计方法易于理解和掌握，因为人类在思维上可以很容易地逐步细化这些简单的问题。而对于复杂的问题，由于程序设计者往往并非此问题领域的专家，所以很难在较短时间内理解问题的本质并掌握解决方法。

3）由于面向过程程序设计方法一般是专门针对特定问题及其流程而设计的，并且各模块间存在着复杂的依赖和关联关系，所以在解决新问题时，以前对问题的理解在此很难适用，即很难复用以前已经设计完成的软件。

程序设计语言的发展大致经历了五代，其中第一代机器语言（二进制代码）、第二代汇编语言（符号代码）、第三代高级语言（符号代码）的发展演化具有比较明确的界限，用它们编写的程序在本质上是相同的，都是按照机器的工作过程来编写的，只不过程序的描述语句越来越接近人类的思维，通用性越来越强。而自第三代语言开始，第四代面向对象语言、数据库语言、非过程语言等与第五代智能化语言、自然语言等则没有严格的分类标准界限，它们往往呈交叉发展趋势。

可以认为，机器语言是对计算机的抽象，汇编语言是对机器语言的抽象，随后的高级语言是对汇编语言的抽象。这些高级语言较汇编语言有了巨大的进步，但这仍是一种初级的抽象，仍然要求程序设计是从计算机的角度，而不是从待解决的问题的角度来思考。程序设计者必须在机器模型与待解决的问题模型之间建立关联，处理这种映射所带来的压力以及编程语言在这方面的不足，这使得程序编写、维护困难。面向对象的方法则更进了一步，它为程序设计者提供了能在问题空间表述各种元素的工具，因此允许从问题的角度，而不是从计算机的角度来描述问题。

1.2 面向对象程序设计方法

所谓对象就是可以控制和操作的实体。现实世界中，对象无所不在，包括人、车、动物、植物、建筑物等。人类习惯的思维方式是面向对象的思维方式，即从对象的角度出发来处理问题，用对象分解代替了功能分解。人们通过对象来思考问题，对于每一个问题，首先分解出来的是问题域中一个个有关的对象，其中每个对象是相对独立的，各自承担完成任务所需要的技能，同时对象间又存在相互作用，彼此影响。如对于学校的管理，首先浮现在人们脑海中的就是该学校的各个管理部门和各种工作人员。至于各个部门，都有自己的管理方式。对于外部人员来说，一般只需要知道各部门的职责，其内部的具体管理细节我们并不用知道更多。这里的各个部门、各种人员就是对象。再比如，上理发店理发，我们只要找到理发店的一个师傅，告诉他剪头发，师傅自然会将头发理好。我们个人不需要去关心剪发的过程和方式，如是先剪前面还是先剪后面、如何操作剪刀等。这里的理发师也是对象。

面向对象的思想起源于 20 世纪 60 年代的 Simula 语言，并在 Smalltalk 语言中得到完善和标准化。面向对象程序设计方法是一种自下而上的程序设计方法，它不像面向过程程序设计方法那样，需要在一开始就要用主函数 main() 概括出整个程序，它往往是从问题的一部分着手，一点一点地构建出整个程序。面向对象设计以数据为中心，同一种类对象抽象出来的

类作为表现数据的工具，成为划分程序的基本单位。

概括起来，面向对象的思想和方法有以下几个重要特点：

1. 客观世界由对象组成

面向对象方法学认为：客观世界由各种各样的对象组成，任何客观的事物或实体都是对象，复杂的对象可以由简单的对象组成。现实生活中的对象一方面有自己的属性，如汽车有长度、颜色、速度、行驶方向、耗油量等，另一方面有自己的行为（也称为方法或操作），如启动、刹车、鸣笛等。

2. 对象抽象为类

人们通过研究对象的属性，观察对象的行为，从而了解对象。每个对象都有许多属性，忽略那些与当前问题无关的特征，抽取其本质特征，并将具有共性的对象归并为一个类（class），从而可知对象是类中的一个具体实体，类是对象共性的抽象。例如，人可以作为一个类，它是世界上所有实体人（如张三、李四等）的抽象，而实体人（张三、李四等）则是人这个类中的一个个对象。

3. 类与类之间存在继承关系

就好比人按性别可以分为男人、女人，男人按年龄大小可以分为老年人、中年人、儿童等，人与人之间存在着一种层次关系，男人、女人是人，作为共性的人，具有人的属性和行为；同时男人、女人还有区别于共性人的特有属性和行为。一个类可以具有另一个类的所有属性和行为，还可以有自己特有的新特性（新的属性或新的行为），这种类与类之间的关系称为继承和派生关系，如图 1-4 所示。

在类的层次结构中，处于上层的类称为父类或基类，处于下层的类称为子类或派生类。

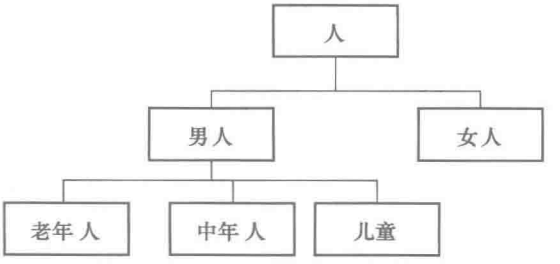

图 1-4 类的层次结构示例

4. 对象之间通过传递消息而彼此联系

客观世界中各个对象之间不是孤立的，它们之间通过传递消息而互相联系。

对于面向对象的程序设计，程序员注重的是类的设计和编写，即问题域中涉及几个类，各个类之间的关系如何，每个类包含哪些数据和方法，而不再是为求解问题而设计功能模块。面向对象程序设计方法中，在着手编写代码之前，先要进行面向对象的分析和设计，分析和设计的过程可以用图形化语言，如统一建模语言 UML（Unified Modeling Language）来表示。

【例 1-3】 利用面向对象的思想求解圆和长方形的面积。

首先利用面向对象的思想来分析这个问题，这里存在两个类：圆和长方形。不同大小的圆、不同大小和形状的长方形各自都是对象。它们有许多属性（数据）：大小、颜色、位置等，还有许多方法，如以不同颜色画图形、移动图形、计算周长、计算面积等。这里与解决本问题有关的数据是大小（可以分别用圆的半径、长方形的长和宽来度量），与解题有关的方法是计算面积。另外，由于圆和长方形都是图形，具有图形的一般特征，如标识图形大小的边（半径或长与宽等），可以计算图形面积、周长等，因此它们又是图形这个类的子类。

面向对象程序设计中，类的属性称为数据成员，表现为类中的变量；类的方法称为成员函数，表现为类中定义的函数。

图形类是从各种不同图形抽象出来的有关图形共性特征的类，称为抽象类，设其类名为

Shape，两个成员函数为 GetMessage() 和 Area_Output()，这是两个公用接口，其实现部分在派生类中，其中，GetMessage() 用于输入数据成员的值，Area_Output() 用于输出图形的面积。Shape 有两个子类：Circle（圆类）和 Rectangle（长方形类）。Circle 类另有数据成员 r 和 s，表示圆的半径和面积，其成员函数 GetMessage() 用于输入半径 r 的值，Area_Output() 用于输出圆的面积。Rectangle 类另有数据成员 l 和 w 以及 s，分别表示长方形的长和宽以及面积，其成员函数 GetMessage() 用于输入长 l 和宽 w 的值，Area_Output() 用于输出长方形的面积。经过分析，可以建立图 1-5 中的对象模型。

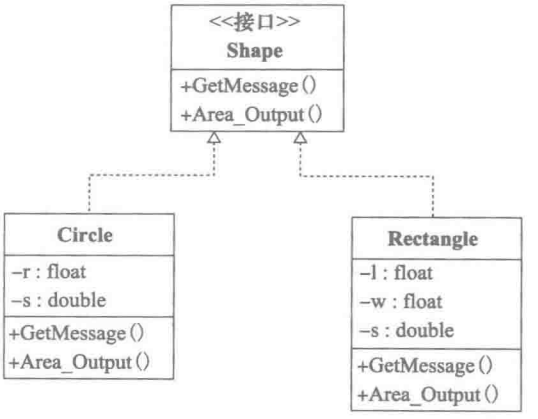

图 1-5 图形的对象模型

根据此模型，可以编写如下的 C++ 代码：

```
#include <iostream>
using namespace std;
class Shape{
  public:
   virtual void GetMessage()=0;
   virtual void Area_Output()=0;
};
class Circle: public Shape{
   float r;                                  //半径
   double s;                                 //面积
public:
   void GetMessage(){
      cout<<"Please input r of circle:"<<endl;
      cin>>r;
   }
   void Area_Output(){
      s=3.14*r*r;
      cout<<"the area of circle is:"<<s<<endl;
   }
};
class Rectangle: public Shape{
   float l, w;                               //长和宽
   double s;                                 //面积
public:
   void GetMessage(){
      cout<<"Please input l of rectangle:"<<endl;
      cin>>l;
      cout<<"Please input w of rectangle:"<<endl;
      cin>>w;
   }
   void Area_Output(){
      s=l*w;
      cout<<"the area of rectangle is:"<<s<<endl;
   }
};
int main(){
   Shape *p;
```

```
    p=new Circle;
    p->GetMessage();
    p->Area_Output();
    p=new Rectangle;
    p->GetMessage();
    p->Area_Output();
    return 0;
}
```

由于 Shape 类在未确定是何图形时，其 GetMessage() 和 Area_Output() 都没有实质性操作，所以将它们的实现部分设为 0。但从程序框架上可以看出，面向对象程序是由类（Shape、Circle 和 Rectangle）组成，各个类中有数据（r、l、w）和方法（GetMessage() 和 Area_Output()），类间具有层次关系，对象间通过传递消息彼此联系（p->GetMessage() 和 p->Area_Output()）。

面向对象的程序由类和对象组成，复杂的程序由比较简单的类和对象组合而成。每个类集中了自己的属性（数据）和方法（操作），并且对于同一个类的不同对象，具体的数据和操作也可能是不同的。如上，所有有关圆的属性和操作封装在 Circle 类中，而所有有关长方形的属性和操作封装在 Rectangle 中。

由于类描述的是一组具有相似特征的属性和行为的对象，因此类实际上也是一种数据类型。例如，整型、浮点型、字符型等 C 语言的基本类型，实际上也有自己的属性和行为，如存储字长、取值范围、算术运算等。这些基本的数据类型之所以有别于类的概念，是因为数据类型是为了表示计算机的存储而设计的，而类是由程序设计者根据问题的需求而自行定义的。

在学习“数据结构”这门课程时，关于程序有一种经典的说法，即：

程序 = 数据结构 + 算法

这一说法对面向过程的程序设计方法是适用的，但对于面向对象的程序设计方法而言，如下表述则更为合适：

程序 = 对象 + 消息

1.3 面向对象方法的基本概念

本节系统地阐述面向对象方法的一些基本概念和理论。

1.3.1 对象、类、实例

现实世界是由各种各样的实体（事物、对象）所组成的，每种对象都有自己的内部状态和运动规律，不同对象间的相互联系和相互作用就构成了各种不同的系统，进而构成整个客观世界。对象是客观世界中的实体，它既可以是具体的物理实体，如公司、汽车、张三等，也可以是一个抽象的事或物，如法律、计划、存款等。对于问题域中的对象，只需考虑与问题本身有关的部分。

对象具有如下特征：

1）有一个名字：称为对象名，用来区别于其他的对象；

2）有一组属性：对象的性质称为对象的属性，一般可以用数据来表示，所有的属性都有值；

3）有一组方法：对象的功能或行为称为方法，一般用一组操作来描述；

4）有一组接口：除施加于对象内部的操作外，对象提供了一组公有操作，用于与外界接口，从而可以与其他对象建立关系。

类是对象抽象的结果，是对具有相同属性和相同操作的一个或一组相似对象的抽象，对象是类的具体化。组成类的对象又称为类的实例，如张三、李四、王五等都是具体的人，他们具有人的共性，所以可以将人抽象为一个类，而张三等则为人这个类的实例。

1.3.2 消息传递

面向对象程序设计方法隐藏了某一方法的具体执行步骤，取而代之的是通过消息传递机制传送消息给它。对象之间相互联系的唯一途径是消息传递。消息是对象之间相互请求、相互协作的途径，是要求对象执行其中某个操作的规程说明。发送者发送消息，接收者通过调用相应的方法对消息做出响应。这个过程不断重复，系统不停地运转，最终得到相应的结果。

因此，一个消息的发送者通常要说明三部分内容：

1）接收消息的对象；

2）消息名；

3）零个或多个变元（消息参数）。

在 C++ 语言中，每个消息在类中由一个相应的方法给出。消息的传递在程序中通过对象调用接口（函数）来实现。例如，“MyCircle.Show(RED);”其中：

1）MyCircle 是接收消息的对象的名字；

2）Show 是消息名；

3）RED 是消息的变元，表示以红色显示圆。

一个对象所能接收的消息及其所带参数构成该对象的外部接口。对象接收它能识别的消息，并按照自己的方式来解释和执行。一个对象可以同时向多个对象发送消息，也可以接收多个对象发来的消息。相同形式的消息可以发送给不同的对象，同一个对象也可以接收不同形式的消息。消息的发送者可以不考虑具体接收者，只是反映发送者的请求。由于消息的识别、解释取决于接收者，对象可以响应消息也可以不响应消息。消息的传递机制如图 1-6 所示。

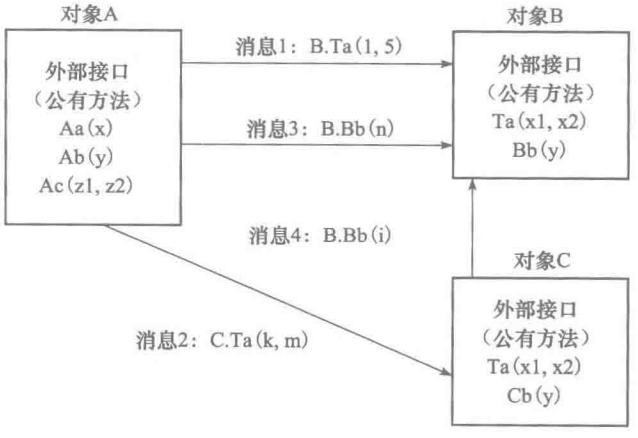

图 1-6 消息的传递机制

1.3.3 类的基本特征：封装、继承和多态

1. 封装

封装是把对象的属性（数据）和操作代码（方法）集中起来放在对象内部，外界通过对象

提供的接口来使用对象，而不需要知道其内部数据结构的细节和实现操作的算法。封装包含两方面的含义：

1）封装是把对象的全部属性和方法结合在一起，形成一个不可分割的独立单位，即对象。

2）封装实现了信息隐藏，即尽量隐藏对象的内部细节，对外形成一个黑盒，只保留有限的对外接口，使之能够与外部发生联系。

从图 1-7 中可以看出，有些数据和方法对外界是不可见的（或称其为私有的）。私有的数据或方法一般由对象的内部操作来改变，外界不必知道对象内部是如何实现的，外界也不可随意对其使用和修改。对象与外界的交流是通过公有操作（外部接口）来实现的。

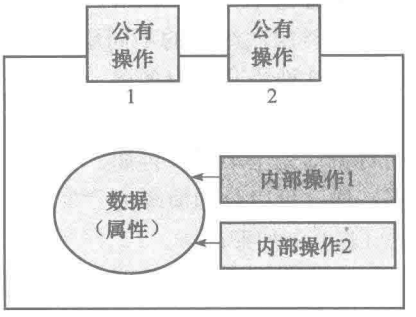

图 1-7 封装图示

例如，录音机、收音机分别都是对象。录音机又有多种，如盘式、卡式、盒式、数字式等，其中卡式录音机具有音量等属性，相关方法包括开盒、播放、停止、倒带、快速前进、调节音量大小、录音、走带方向控制、杂音抑制、立体声通道幅度平衡等方法，对外提供的接口包括开盒、播放、停止、倒带、快速前进、调节音量大小、录音。

在软件设计上，封装原则的要求是：对象以外的部分不能随意存取对象的内部数据（属性）。这一原则既实现了信息隐藏，也有效地避免了外部对它的错误操作。

这很像电视机、录音机、游戏机等，从其外形来看，它们都是各种机械零件被封装在盒子内部。使用这些机器的人并不需要知道机器内部有哪些零件、它们是如何组装的、它们的工作原理又如何。使用者只需要会使用机器提供的几个外部按钮（对应于对象的外部接口），就可以实现自己所需要的功能。将机器零件封装在盒子内部，既可以避免各种人为的损坏，也便于维护和管理。很显然，实现封装的条件包括：

1）有一个清楚的边界：对内的功能和对外的功能可以比较明确地划分开。

2）有确定的接口：用于接收用户发送的消息。

3）受保护的内部实现：内部功能对外是不可见的。

在 C++ 语言中，封装是通过定义类来实现的。

2. 继承

继承是类与类之间一种比较常见的关系，它是指一个类（称为子类）可以直接获得另一个类（称为基类）的性质和特征，而不用重新定义它们。例如，图形与线、弧、多边形及圆之间的关系是继承关系，所有的图形都由线条构成，有颜色、有位置、可以移动等，所以这些公共属性可以放在图形类中，线、弧、多边形及圆可以继承图形的公共属性，而不必重复定义，同时它们也可以各自定义专用的而图形类中未包含的属性。

在面向对象程序设计中，继承是子类自动共享基类中已定义的数据和方法的机制。这种机制增强了系统的灵活性、易维护性和可扩充性。从继承的定义可以看出，继承具有传递性，如果类 C 继承类 B，类 B 继承类 A，则类 C 继承类 A。类 C 除了具有该类所描述的性质外，还具有该类上层全部基类（类 B 和类 A）所描述的一切性质。

一个类也可以从多个类中继承属性与方法，这称为多重继承，例如，一个公司的销售经理既具有销售员的角色，又具有经理的角色，因此销售经理这个类可以继承销售员类和经理类。再如，收音机的例子，收音机又分为调幅（中波）收音机、调频（FM）收音机、调频 / 调

幅两波段收音机、调频立体声/调幅两波段收音机、调频/中波/短波多波段收音机等，其中调幅收音机、调频收音机是收音机的子类，调频立体声收音机是调频收音机的子类，调频/调幅两波段收音机是调频收音机和调幅收音机的共同子类，既继承调频收音机，又继承调幅收音机。另外，收录两用机是收音机和录音机的共同子类。

3. 多态

在类层次结构的不同层级中，相同的消息被不同类的对象接收，可能会产生不同的响应效果，这种情况称为多态。多态的概念在现实生活中到处可见，也具有多种形式的多态。“打”这个字，可用于“打篮球”“打乒乓球”“打酱油”“打人”，虽然都是同一个“打”字，但触发的对象却完全不同。再如，“动物吃东西”这个方法，具体到牛和狼，它们吃的方式和内容都不一样，可以分别演绎为“牛吃草”和“狼吃肉”。再如，水具有固态（冰）、液态（水）和气态（汽）三种形态。

在面向对象程序设计中，多态性依托于继承性。在具有继承关系的类层次结构中，不同层级的类可以共享一个操作，但却可以有不同的实现。当对象接收到一个请求时，它根据其所属的类，动态地选用在该类中定义的操作。

多态性是在对象体系中把设想和实现分开的手段。多态性意味着某种概括的动作可以由特定的方式来实现，把需要设计处理的特定抽象描述留给知道该如何完美地处理它们的对象去实现。多态性可以为程序提供更强的表达能力，如可以使程序中的数学运算（如复数的加减乘除运算）符合常规的数据（整型数据、实型数据）运算规则。多态性也可以使得对不同类型的数据有同样的操作形式，从而实现程序的重用，并通过重用标识符提高程序的可阅读性。

对多态性的概括性描述是：对象具有唯一的静态类型和多个可能的动态类型。多态是面向对象程序设计的精髓之一，它增加了软件系统的灵活性，减少了信息冗余，从而提高了软件的可重用性和可扩充性。

1.4 面向对象的开发过程

面向对象的软件开发过程包括面向对象的分析（Object-Oriented Analysis，OOA）、面向对象的设计（Object-Oriented Design，OOD）和面向对象的实现，即面向对象的编程（Object-Oriented Programming，OOP）。面向对象的方法学大师 Grady Booch 对面向对象的分析、面向对象的设计和面向对象的实现有一段经典的论述，即：

1）面向对象的分析是一种分析方法，它用可在问题域的词汇表中找到的类和对象的观点来审视需求。

2）面向对象的设计是一种设计方法，它包含面向对象的分解过程，以及一种表示方法，用来描写设计中的系统逻辑模型与物理模型，以及静态模型与动态模型。

3）面向对象的实现是一种实现方法，程序被组织成对象的协作集合，每一个对象代表某个类的实例，对象的类是通过继承关系联合在一起的类层次中的所有成员。

在现实生活中，有时为了透彻地理解问题，人们通常采用建立问题模型的方法，即用一组图示符号及其组织规则，将现实具体的东西形式化地表达出来。通过模型抽象出问题的本质，使问题更容易理解，建立问题模型的过程称为建模。统一建模语言（Unified Modeling Language，UML）是一种基于面向对象的可视化建模语言，它采用标准的、易于理解的方式表达出问题，便于用户和开发者进行交流。常用的 UML 建模工具有 Rational Rose、Microsoft Visio 等，比较容易得到而且被广泛使用的是 Microsoft Visio，它是 Office 系列软件之一。