计算机软件技术的原理与应用

Computer Software

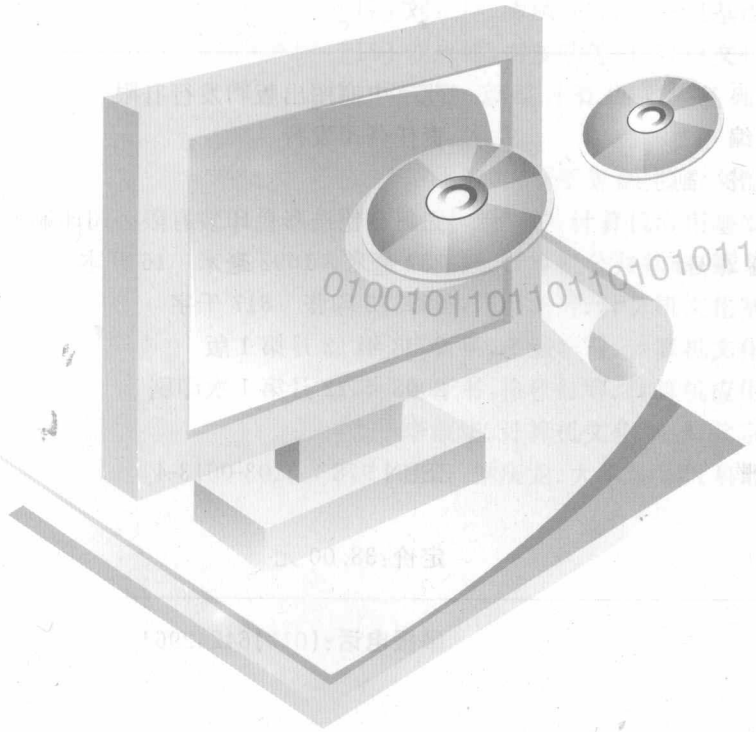
孙博玲　刘天寅　宋建松　主编

CCTP
中国商务出版社
CHINA COMMERCE AND TRADE PRESS

计算机软件技术的原理与应用

Computer Software

孙博玲　刘天寅　宋建松　主　编
畅卫功　牛小梅　王淑敬　李　潜　副主编



CCTP
中国商务出版社
CHINA COMMERCE AND TRADE PRESS

图书在版编目(CIP)数据

计算机软件技术的原理与应用/孙博玲，刘天寅，宋建松主编. —北京：中国商务出版社，2008.12

ISBN 978-7-5103-0013-4

Ⅰ.计… Ⅱ.①孙…②刘…③宋… Ⅲ.软件—技术 Ⅳ.TP31

中国版本图书馆 CIP 数据核字(2008)第 200515 号

计算机软件技术的原理与应用

孙博玲　刘天寅　宋建松　主编

畅卫功　牛小梅　王淑敬　李潜　副主编

中国商务出版社出版

(北京市东城区安定门外大街东后巷 28 号)

邮政编码：100710

电话：010—64269744(编辑室)

010—64266119(发行部)

010—64295501

010—64263201(零售、邮购)

网址：www.cctpress.com

Email：cctp@cctpress.com

北京中商图出版物发行有限责任公司发行

三河市铭浩彩色印装有限公司印刷

787 毫米×1092 毫米　16 开本

25.375 印张　617 千字

2008 年 12 月第 1 版

2008 年 12 月第 1 次印刷

ISBN 978-7-5103-0013-4

定价：38.00 元

前　言

随着计算机技术的深入发展，计算机技术特别是计算机软件技术的应用已经渗透到各个领域，对计算机各种软件技术的掌握也已经不再只是计算机专业人员的事情了。现在，越来越多的软件需要非计算机专业人员来设计与开发，很多系统软件与应用软件由非计算机专业人员来使用，同时日常生活和工作的各个方面也需要应用计算机软件技术方面的知识。因此，普及计算机软件技术已经是大势所趋。

应该说，计算机软件技术的应用不仅在广度上，而且在深度上渗透到了社会的各个层面，因此，对各种非计算机专业的学生及科技人员进行深入的计算机教育已迫在眉睫。他们不仅需要了解计算机软件技术的发展状况、软件工程学和设计技术等方面的知识，而且更需要了解数据结构和算法分析的方法、操作系统的基本功能以及数据库的基本理论等更深层次的计算机软件基础知识。

全书共分 10 章。第 1 章以结构化程序设计和面向对象的程序设计为重点，讨论了计算机设计程序的一般方法。第 2 章和第 3 章则是数据结构与算法分析，这是作为程序设计过程中必不可少的基础知识。第 4 章则以 Visual Basic 可视化程序设计方法为例，讲解了程序设计的一些语言基础、结构设计以及过程与函数设计等内容。接下来的 5 至 9 章，则是涵盖了操作系统、数据库技术、软件工程、网络技术以及多媒体技术等软件技术的方方面面。第 10 章重点探讨了信息安全的重要性以及信息安全中常用的一些技术和方法，如数据备份、防火墙技术、密码技术等。而计算机病毒及计算机犯罪的危害也是不可小视的。书中也作了相应分析。

总之，本书系统地介绍了计算机软件方面的知识，内容涵盖软件技术的各个方面。作者根据自己多年的研究和教学经历，尽量用通俗、简洁的语言来描述与计算机软件相关的原理、方法和应用。本书内容丰富、结构合理、循序渐进、实用性强，便于教学和自学，既可作为高校学生学习计算机软件用书，也可作为广大从事计算机应用工作的科技人员的参考书。

全书由孙博玲、刘天寅、宋建松担任主编，畅卫功、牛小梅、王淑敬、李潜担任副主编，并由孙博玲、刘天寅、宋建松负责统稿。其具体分工如下：

第 5 章，第 7 章第 1～5 节与第 7～9 节，第 9 章第 4～7 节：孙博玲（哈尔滨学院）；

第 1 章第 3～5 节，第 2 章第 1～2 节与第 4～6 节，第 9 章第 3 节：刘天寅（内蒙古科技大学包头师范学院信息科学与技术学院）；

第 2 章第 3 节，第 4 章，第 6 章第 4～5 节：宋建松（长治学院）；

第 1 章第 1～2 节，第 7 章第 6 节，第 8 章：畅卫功（天津科技大学）；

第 1 章第 6 节，第 3 章，第 10 章第 6 节：牛小梅（黄淮学院）；

第 6 章第 2 节，第 10 章第 1～5 节：王淑敬（天津开发区职业技术学院）；

第 6 章第 1 节，第 3 节与第 6 节，第 9 章第 1～2 节：李潜（天津中医药大学）。

本书在编写过程中参阅了大量的著作文献及相关资料，在此对这些资料的作者表达诚挚的谢意。

由于作者水平有限，书中难免有错误或不妥之处，恳请读者批评指正。

编者

2008.12

目　录

第1章 程序设计方法

数据是原始事实的数字记录，它本身并没有什么意义，而信息是经过处理后的数据，信息具有实际含义。数据转化为信息的过程就是数据处理，也称其为信息处理。数据处理的基本模型是：输入数据→数据处理→输出信息。

实际的计算机系统中，数据由键盘、光电阅读器等输入设备进行输入；信息则由显示器、打印机等输出设备进行输出；数据处理则是由计算机程序来进行的，因此，要处理数据，必须编写程序或购买现成的程序。

1.1 计算机程序

程序是计算机任何动作的驱动力，程序是计算机为完成某一任务所必须执行的一系列指令。没有程序，计算机就像原地待命的士兵，什么也不做。

为了理解程序对于计算机多么重要，举例说明。当孑身一人来到一个陌生的城市，可能首先想到去买一张交通图，以便能按图索骥找到目的地。计算机对于程序的需要正像对于地图的需要，只有这样，才能引导计算机一步步地到达目的地，实现目标。程序中的指令正是人们要求计算机一步步地完成任务的一个个小的命令。

1.1.1 程序的存储

从存储的角度来讲，程序是保存在磁盘上的指令序列。比如，用字处理软件编辑处理文档时，首先启动该软件，启动软件就是将该软件包含的主要程序从磁盘调入内存，并执行主要程序中的一部分指令，然后运用另一部分指令来完成文档处理工作。运行一个程序，首先必须将其调入内存，这是由于计算机的 CPU 无法直接执行磁盘上的程序。

除了少量存在于 ROM 等存储器中的特殊程序之外，程序大多以文件的形式存在于磁盘等外存储器上，但存在于磁盘上的文件并不都是程序，有的可能是字处理文档文件、图形图像文件、声音和视频多媒体文件等。

程序的输出通常也是以文件的形式保存到磁盘上，但它们是数据文件或文档文件，它们是程序的执行结果；程序文件则是指令的集合。程序与其处理结果的关系正像菜谱与做好的菜的关系，做好一道菜之后，还可以按照菜谱再做另一道菜。运行某一程序对一些数据进行处理有了结果之后，还可以再次运行该程序对另一些数据进行处理，得到另一些结果。

1.1.2 算法与程序的关系

只有先提出问题，才能解决问题。计算机程序设计首先从问题的描述开始。它是算法的基础，而算法则是程序的基础。

无论用计算机解决哪一方面的问题，我们都必须设法用数学方法来描述或模拟这些实际

问题，把对实际问题的可行解决方案归结为计算机能够执行的若干步骤，然后再把这些步骤用一组计算机指令进行描述，形成所谓的程序，最后交给计算机去执行。

1. 计数和累加

在数学课上，当写了 Num＝Num＋1 这样的式子，那肯定是错误的，任何数不可能等于自身加 1。然而，在程序中，数学课中的等号“＝”，将被称做赋值号。赋值号“＝”的含义是：获取等号右边的所有符号，计算它们的值，并将结果放在等号左边的变量中。

一个特别有意义的赋值式子是诸如 Num＝Num＋1 这样的式子，事实证明它非常有用。因为它用在循环结构中，能够统计出循环执行的次数。因为每当循环一次，该变量的值就会加 1，故把该式子中的变量称为计数器变量。

累加器类似于计数器，如同样的变量名出现在赋值号的两边。不同的是，累加器通常给变量增加的值不是 1，而是一个数，比如要求若干个学生的总分，可以使用 Total＝Total＋Score 这样的语句，其中 Score 是逐个输入的某个学生的考分，最后所得的 Total 记录下来的就是总分。

2. 交换变量的值

排序算法中一般都用到交换变量的值，也就是对换变量，如图 1-1 所示，交换的结果是原来变量 1 中的值现在存储在变量 2 中，而原来变量 2 中的值却存储在变量 1 中。

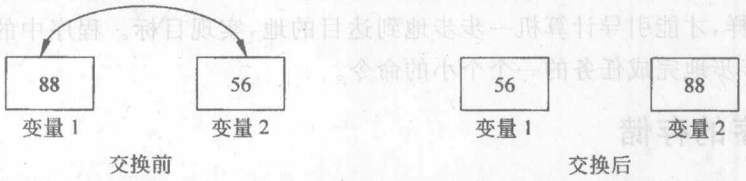


图 1-1　交换变量的值

假如使用下列两条语句，将变量 Variable1、Variable2 分别赋予了具体的值。

Variable1＝88

Variable2＝56

要对它们进行交换，如果使用下面的语句，那就大错特错了。

Variable1＝Variable2

Variable2＝Variable1

这是因为第一句将变量 Variable2 赋予 Variable1，用 Variable2 的值覆盖了原来 Variable1 的值，这时两个变量的值一样了，第二句已是多余的了。

交换变量的值的正确方法是，使用一个中间变量，也称为临时变量，因为一旦交换完成，这个变量也就没有用了。语句表示为：

Temp＝Variable1

Variable1＝Variable2

Variable2＝Temp

1.2 结构化程序设计

程序结构包括程序的3大控制结构和子程序。程序的3大控制结构是顺序结构、选择结构和循环结构。下面以最简单的Visual Basic语句来举例说明这3种结构和子程序。

1.2.1 顺序结构

顺序结构，就是计算机按照程序中语句的自然顺序依次执行。第一条语句先执行，接着第二条……一直到程序结束。如下面一个Visual Basic程序片段：

```
Debug. Print"This is the first line. "
Debug. Print"This is the second line. "
```

这两行程序的功能分别是，打印该语句引号中的内容。它的执行情况如图1-2所示。

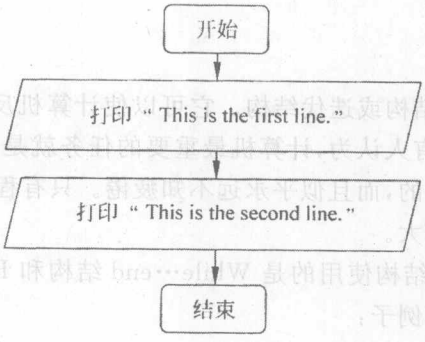


图1-2 顺序结构

顺序结构是3种控制结构中最简单的一种。由于程序大多需要处理判断问题和重复执行一些语句，因此，不可能只由顺序执行的逻辑结构组成。

1.2.2 选择结构

选择结构也称为分支结构，程序执行到选择结构时，必须根据判断的结果作出选择，因此选择结构中的语句只有其中之一被执行，有所选择必定有所放弃。在Visual Basic中，选择结构使用的是If…Then…Else…End If语句。举例说明如下：

```
Number=INPUTBOX("Enter a number from 1 to 10:")
If Number>10 Then
Debug. Print"That number is greater than 10. "
Else
Debug. Print"That number is 10 or less. "
End If
```

分析如图1-3所示的程序流程图，更容易通过上述Visual Basic程序片段来理解分支结构是如何执行的。

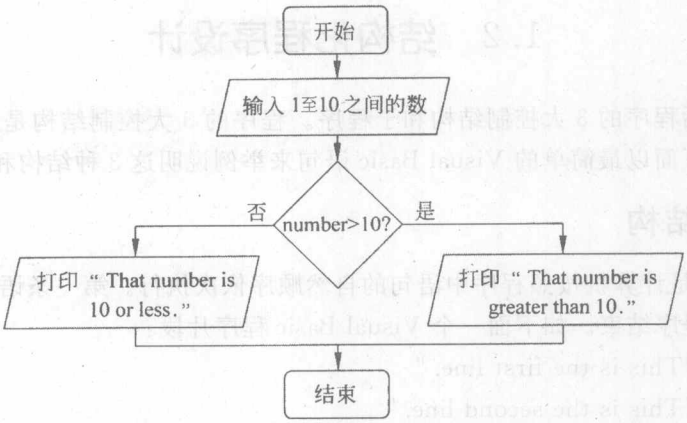


图 1-3　选择结构

1.2.3　循环结构

循环结构又称重复控制结构或迭代结构。它可以使计算机反复多次执行同一段程序，直到循环条件不满足时结束。有人认为，计算机最重要的任务就是循环。成千上万次地重复执行某些操作，是计算机最擅长的，而且似乎永远不知疲倦。只有程序中恰当地采用了循环控制结构之后，才可能变得功能强大。

在 Visual Basic 中，循环结构使用的是 While…end 结构和 For…Next 结构。同样，举一个简单的 BASIC 程序片段的例子：

```
For N=1 TO 3
Debug. Print"There's no place like home. "
Next N
End
```

如图 1-4 所示的程序流程图，描述了上述 Visual Basic 程序是如何执行循环结构的。

图 1-4　循环结构

为了对循环结构有一个更深刻的理解，下面将程序的执行过程详细说明如下：

如图 1-5 所示，用一个标有“N 框”的矩形框来代表记录循环次数的内存变量，用另一个标有“显示框”的矩形框来代表计算机屏幕，用来显示结果。随着循环一次次地进行，这两个框中的内容也不断改变。

①首先，执行语句 For N=1 TO 3，把 N 赋值为 1。N 框中的值为 1。

②执行语句 Debug. Print" There's no place like home. "。显示框中显示" There's no place like home. "。

③语句 Next 使程序的执行回到语句 For N=1 TO 3。因为这是第二次执行这条语句，所以 N 的值自动加 1，N 框中的值由 1 改为 2。

④测试 N 的值是否大于 3。这是因为语句 For N=1 TO 3 表示只有

当 N 的值不大于 3 时，才能继续循环下去。此时，N=2，继续执行。

⑤在显示框中再显示“There's no place like home.”。

⑥继续执行，又遇到 Next 语句，回到 For 语句。

⑦把 N 框中的值改为 3，并判断此时 N 的值并不大于 3，再次进入循环。

③在显示框中再显示“ There's no place like home.”。

⑨继续执行，又遇到 Next 语句，回到 For 语句。

⑩把 N 框中的值改为 4，并判断此时 N 的值已大于 3，跳出循环，循环结束。

⑩执行语句 End，程序运行结束。

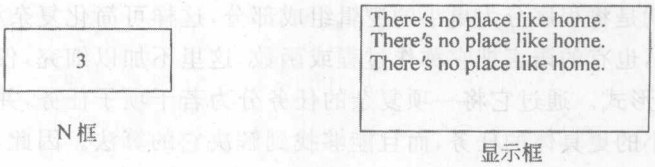


图 1-5 N 框和显示框

如果程序需要不止一次重复一个循环，比如需要打印乘法口诀表，可以使用嵌套循环。与其他语句一样，循环语句也可以嵌套在另一个循环结构之中。如图 1-6(左)所示，显示了一个 Visual Basic 的二重循环轮廓图，图中的外层循环执行了 9 次，内层循环在外层循环的每一次完成了一个轮回，即 9 次，因此内层循环代码总共执行了 81 次。

如果要打印乘法口诀表的话，就可以用这个轮廓图所指示的二重循环，它可打印出 81 个数。如果需要的话，可以将多个循环嵌套在一个外层循环之中。在外层循环完成它的第一次循环之前，必须将内部的两个或多个循环都完全执行一个轮回。当外层循环执行第二次的时候，内部每个循环又将作同样的重复，直到外层循环结束。如图 1-6(右)所示的循环结构中。每一个内层循环代码块都会执行 81 次。

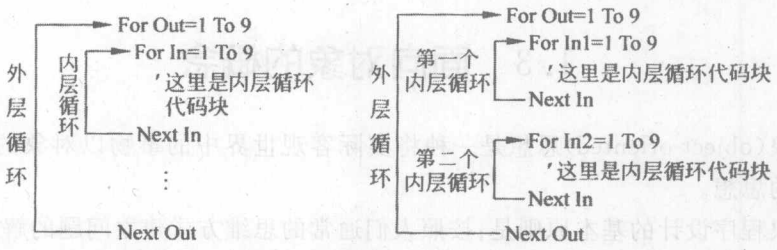


图 1-6 嵌套循环结构

在使用嵌套循环时，一个要注意的问题是，Next 语句与 For 语句成对，并且 Next 总是与离它最近的 For 匹配。如果程序中内部循环的 Next 语句在外部循环的 Next 语句的后面，将会引起语法错误。

```
Dim xOut,xIn as integer
For xOut=1 To 9
```

```
For xIn=1 To 9
If xOut>=xIn Then
Debug. Print"xOut * xIn"
End If
Next xIn
Debug. Print
Next xOut
```

1.2.4 子程序

所谓子程序，就是将程序分为更小的逻辑组成部分，这样可简化复杂程序的编写，这些组成部分就叫子程序，也有的语言把它称作过程或函数，这里不加以细究，仅认为子程序是程序模块化的一种组织形式。通过它将一项复杂的任务分为若干项子任务，并如此一步步地细分下去，直到面对更小的更具体的任务，而且能够找到解决它的算法。因此，用子程序的思想编程有以下一些优点：

(1)子程序将程序分为一个个独立的逻辑单位，对每个单位的调试比起一个大的程序要容易得多。

(2)一个子程序可以稍作修改或不作修改再被其他程序调用，从而避免了书写重复代码，节省了时间和资源。

在 Visual Basic 中，调用子程序使用 GOSUB 语句或 CALL 语句。在被调用的子程序中使用 RETURN 语句结束子程序，返回调用子程序的语句处继续执行程序。

当子程序只有几行长，并且不是被多次调用的时候，可能体会不到使用子程序的优越性。然而，如果长期从事编程工作，自然会想到将常用的算法编成子程序，比如查找算法或排序算法等，这样通过一段时间的积累，编程工作更像搭积木，而不是所有工作都是从零开始。

通过子程序方式来组织的程序，结构非常清晰，即使不能一下子读懂程序，也不会觉得程序杂乱，并且很快会把注意力集中到较难的语句上。

1.3 面向对象的概念

面向对象(object-oriented)思想是一种将实际客观世界中的事物以对象的形式映射到计算机世界中的思想。

面向对象程序设计的基本原则是：按照人们通常的思维方式建立问题的解空间，要求解空间尽可能自然地表现问题空间。为了实现这个原则，必须抽象出组成问题空间的主要事物，建立事物之间相互联系的概念，还必须建立按人们一般思维方式进行描述的准则。

1.3.1 面向对象的基本概念

面向对象的思想建立在对象的概念基础之上，在 20 世纪 60 年代，第一次出现了具有面向对象思想的语言 Simula-67。在 20 世纪 70 年代初，Xerox 公司推出了 Smalltalk 语言，奠定了面向对象程序设计的基础，到了 20 世纪 80 年代，随着 Smaltalk-80 的推出，面向对象的程序设

计进入了实用阶段。从这个时候开始，人们开始关注面向对象分析和设计的研究，并逐步形成了面向对象方法学。在当前的软件开发中，比较流行的面向对象开发语言主要以C++、Java为主，在C++中为了实现对C语言的继承，仍然可以看到一些面向过程语言的痕迹，而Sun公司所推出的Java语言，是一种完完全全的面向对象语言，本节的例子代码均采用Java语言来编写。目前，面向对象分析已经被软件工程界公认为最具有发展潜力的、重要的需求分析方法，面向对象设计虽然没有结构设计和结构设计方法那样流行，但是在不远的将来，随着面向对象语言的逐渐成熟，面向对象的开发方法和面向对象程序设计语言一定将在软件工程中占据主导地位。

在面向对象程序设计中，对象(object)和消息传递(message passing)分别表现事物以及事物之间的相互关系。类(class)和继承(inheritance)是按照人们一般思维方式的描述准则。方法(method)是允许作用于该类对象上的各种操作。这种对象、类、消息(message)和方法的程序设计的基本点在于对象的封装性(encapsulation)和继承性。通过封装能将对象的定义和对象的实现分开，通过继承体现类与类之间的相互关系，以及由此带来的实体的多态性(polymorohism)，从而构成了面向对象的基本特征。下面分别介绍这些概念和特征。

1. 类与对象

在真实世界里，有许多相同“种类”的对象。而这些同“种类”的对象可被归为一个“类”。例如，可将世界上所有的汽车归类为汽车类，所有的动物归为动物类。在面向对象程序设计中，类的定义实质是一种对象类型，它是对具有相同属性和相似行为的对象的一种抽象。例如汽车类有些共同的状态(汽缸排气量、排挡数、颜色、轮胎数等)和行为(换挡、开灯、开冷气等)。

对象是在程序中根据需要动态生成的，一个类可以生成许多状态不同的对象。同一个类的所有对象具有相同的性质，即它们的属性和行为相同。一个对象的内部状态只能由其自身来修改，任何别的对象都不能改变它。因此，同一个类的对象虽然属性相同，但它们可以有不同的状态，这些对象是不同的。

对象是具有某些特殊属性(数据)和行为方式(方法)的实体。可以把现实生活中的任何事物都看作对象。对象可以是有生命的个体，如一个人或一只鸟；对象也可以是无生命的个体，如一辆汽车或一台计算机；对象也可以是一件抽象的概念，如天气的变化或鼠标所产生的事件。

对象有两个特征：属性(property)和行为(behavior)。例如：一个人的属性有性别、年龄、身高、体重等，行为有唱歌、打球、骑摩托车、开汽车等；一只狗有它的属性，如颜色，也有它的行为，如吠叫或跳跃。

在面向对象程序设计中，对象的概念由现实世界对象而来，也可以被看做一组成员变量和相关方法的集合。对象的属性保存在成员变量(variables)或数据字段(data field)里，而行为则借助方法(methods)来实现。对象占据存储空间，一旦给对象分配了存储空间，相应的属性赋了值，就确定了对象的状态，而与每个对象相关的方法定义了该对象的操作。

对象可以被看做一片私有存储空间，其中有数据也有方法。其他对象的方法不能直接操纵该对象的私有数据，只有对象自己的方法才可操纵它。以汽车对象为例，可以定义其属性与方法，如图1-7所示。

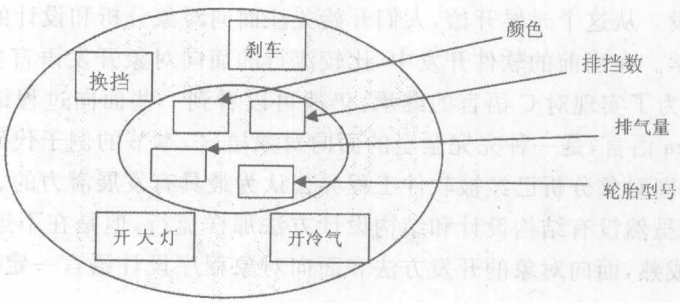


图 1-7　汽车对象

(1)对象

在现实的世界中,每个实体都是对象(object),比如,电视机、汽车、电冰箱等,这些都是现实世界中的对象。这些对象都有自己的属性和可以执行的操作,如电视机有颜色、音量、频道等属性,可以进行开/关电视机、调节音量、切换频道等操作。电视机的这些属性表示了电视机当前所处的状态,通过执行相应的操作可以改变电视机的这些属性,更改电视机的工作状态。电视机如何播放电视节目,内部的电路板、开关如何工作,这些具体的细节用户并不关心,用户只需要知道可以通过一些操作改变电视机的属性,使电视机工作在所希望的状态下。

在计算机世界中,对象是指具有一组属性及由对这些属性的专用操作组成的封装体。属性通常是一些数据,也可以是其他的对象。例如,电视机是对象,具有生产厂家的属性,而生产厂家同样也可以是对象。对象的属性只能通过该对象所提供的方法来进行存取和更改。对于用户来说,用户只能通过对象所提供的方法来访问和操作对象,对象的内部对用户是不可见的,这就达到了对象的封装性。例如,前面例子中的电视机,普通用户只需要按动电视机的按键,就可以进行频道切换,不需要用户亲自去设计和实现电视机内部如何切换频道,也不允许用户打开电视机的后盖自己进行维护。

(2)类

类(class)是具有相同属性和相同操作的对象的集合,是对这些相同对象的抽象,相当于对象的"模板"。通过这个"模板",可以创造具有相同特征的对象,用类创建对象的过程叫做类的实例化(instantiation)过程,每个创建的对象叫做类的一个实例(instance)。这种关系有点类似于高级程序设计语言中的数据类型和变量的关系。如图 1-8 所示,左图是"电视机"类,其中的"型号"、"厂家"为类的属性,在这里为了简化,没有列出类的操作;右图是"电视机"类的一个实例,"小刘家的彩电"是"电视机"类的一个对象,它具有自己的属性值。

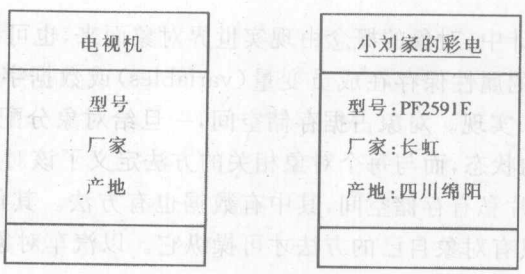


图 1-8　类和类的实例

图 1-8 对应的 Java 代码：

```
public class TV//声明 TV 类(电视机类)
{
private String type;//型号,private 表示属性只能在类内部访问
private String factory;//厂家
private String manu_area;//产地

public TV(string t,String f,String m)//构造函数,创建对象时调用
{
this. type=t;//将 TV 的属性 type 赋值为 t
this. factory=f;//将 TV 的属性 factory 赋值为 f
this. manu_area=m;//将 TV 的属性 manu-area 赋值为 m
}
public static void main(String [] args)//主函数,程序的出发点
{
/ * 新创建的一个对象,表示小刘家的彩电
 * tv_liu 前面的 TV,表示小刘家的彩电 tv_liu 是属于 TV 类的,是一个电视
 * /
TV tv_liu=new TV("PF2591E","长虹","四川绵阳");
}
}
```

2. 消息

单一对象本身并不单独使用，而通常是成为一个包含许多对象的较大型程序的一个组件。对象之间需要进行交互，通过程序中对象的交互，程序可以达成更高级的功能以及更复杂的行为。就如汽车本身并不会产生行为，而是当驾驶员(另一个对象)发动汽车，踩油(交互)后，汽车内部就发生一连串复杂的行为。

对象通过传送消息给其他对象来达到交互及沟通的作用。而消息是用来请求对象执行某一处理或回答信息的要求的。

一个消息由以下 3 个元素组成。

(1)消息的接收者，即消息的目标对象(汽车)。

(2)执行方法的名字(换挡)。

(3)执行方法所要的参数(parameters)(低、高速挡)。

采用消息的处理方式的好处如下：

(1)一个对象的行为是通过它的方法来表达的，所以(除了直接的变量存取外)消息传递支持所有对象间可能的交互。

(2)对象不需要在相同的程序中、相同的机器上送出或接收与其他对象间的交互信息。

当一个面向对象的程序运行时，一般要做以下 3 件事情。

(1)首先需要创建对象。

(2)当程序处理信息或响应来自用户的输入时,要从一个对象传递消息到另一个对象。

(3)若不再需要该对象时,应删除该对象并回收它所占用的存储空间。

消息(message)是对象间进行通信的唯一途径。一个对象通过向另一个对象发送消息来请求这个对象的外部服务。通常,对象的操作主要用于响应外来的消息请求,因此它们也常常被叫做对象的"外部服务"。消息一般包括接收对象名、调用的操作名和该操作所需要的参数。消息仅仅告诉接收对象需要完成的操作,并不包括如何完成该操作,完成该操作的具体细节由接收对象自己决定,消息本身没有任何关于接收对象的内部信息。例如,急救中心接到急救请求电话,急救中心会自己安排急救车开赴急救现场,并在接到病人后安排急救中心的抢救工作。所有工作都不需要急救请求方进行干涉,完全由急救中心来实现。

一般的,传统的过程式编程语言是以过程为中心,以算法为驱动的,而面向对象的编程语言则是以对象为中心,以消息为驱动的。用公式来表示,过程式编程语言为

程序=数据结构+算法

面向对象编程语言为

程序=对象+消息

3. 聚合

在现实世界中除了一般到特殊的关系,还存在着整体与部分的关系,这种由部分组成整体的关系叫做聚合(aggregation)关系。例如,汽车是由发动机、变速箱、传动轴等组成的。利用不同的类组合成一个新的类,这里用发动机类、变速箱类、传动轴类以及其他需要的类组成汽车类。

1.3.2 面向对象的基本特征

1. 封装

从汽车对象的表示图里,可以看到对象的核心是由对象的变量所构成的。对象的方法包围此核心,使核心对其他的对象是隐藏的,而将对象的变量包裹在其对象方法的保护性监护之下就称之为封装。封装用来将对其他对象不是重要的细节隐藏起来,就好比开车换挡时,并不需要知道变速箱、齿轮等机械如何运作的,只要知道将挡排到哪里即可。同样在软件程序中,并不需要知道一个类的完整结构如何,只要知道要调用哪一个方法即可。面向对象程序设计将数据成员(data member)和属于此数据的操作方法(operating method)都放在同一个实体(entity)或称对象(object)中,这就是所谓的封装。

封装的用意是避免数据成员被不正当地存取,以达到信息隐藏(information hiding)的效果,避免错误地存取发生。封装相关的变量及方法到一个软件包里,是一个简单但却很友好的想法。此想法对软件开发者提供了两个主要的好处:模块化和信息隐藏。

(1)模块化(modularity)

一个对象的原始文件可以独立地被编写及维护而不影响其他对象,而且对象可以轻易地在系统中来回地传递使用。就好像把车借给朋友,而它仍能正常地运行一样。

(2)信息隐藏(information hiding)

一个对象有一个公开的接口可供其他的对象与之沟通,但对象仍然维持私有的信息及方法,这些信息及方法可以在任何时间被修改,而不影响那些依赖此对象的其他对象。

2. 继承

面向对象程序设计便是以类来定义一个对象的。当要使用一个对象(的变量或方法)时,首先会想到它是属于哪一种类的。不仅对象是以类来定义的,更进一步地,类也可以用其他类来定义。

继承(inheritance)关系是面向对象思想中一般与特殊之间的关系,它模拟的是现实世界中的遗传关系。把一般的类叫做"父类"(superclass),特殊类叫做一般类的"子类",特殊类具有一般类的全部特征。例如汽车是一种交通工具,"汽车"类就是"交通工具"类的子类,"交通工具"是"汽车"类的父类;同时,轿车是一种汽车,"轿车"类就是"汽车"类的子类,"汽车"类又是"轿车"类的父类。这样,根据这种一般到特殊的关系,就可以把整个世界抽象成一个由不同的类所组成的树,如图 1-9 所示。

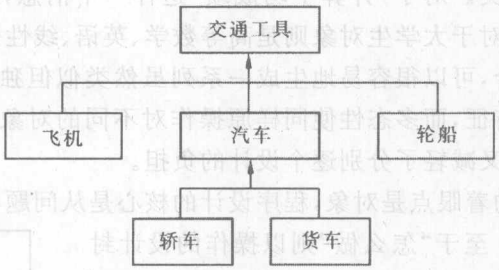


图 1-9 继承关系

从这里可发现"汽车类"是比较通用、概念性的类,故在汽车类中定义了一些通用的属性与行为,如引擎数量、汽缸数、排气量、外观颜色、开大灯、开窗户、转向、加速等,但这些属性与行为在汽车类中可不用实现(implement),而在子类(轿车、出租车、巴士)中实现。如"外观颜色"在汽车类中只定义有这样的属性,而到了出租车类中才实现为"黄色"。这样在超类中只定义一些通用的属性与实现部分的行为,到了子类中才实现细节,称这样的超类为抽象类(abstract class)。在抽象类中只定义一些属性和实现少部分行为。这样其他的程序设计师就可依照他们所要的特定子类进行实现与定义,就像轿车、出租车、巴士都有它们特定的属性与行为,例如颜色、刹车等。

继承的好处如下。

(1)实现代码重用。利用超类程序代码,在编写子类时,只要针对其所需的特别属性与行为来写即可,提高程序编写的效率。

(2)先写出定义好却尚未实现的抽象超类,可使得在设计子类时简化设计过程,只要将定义好的方法填满即可。