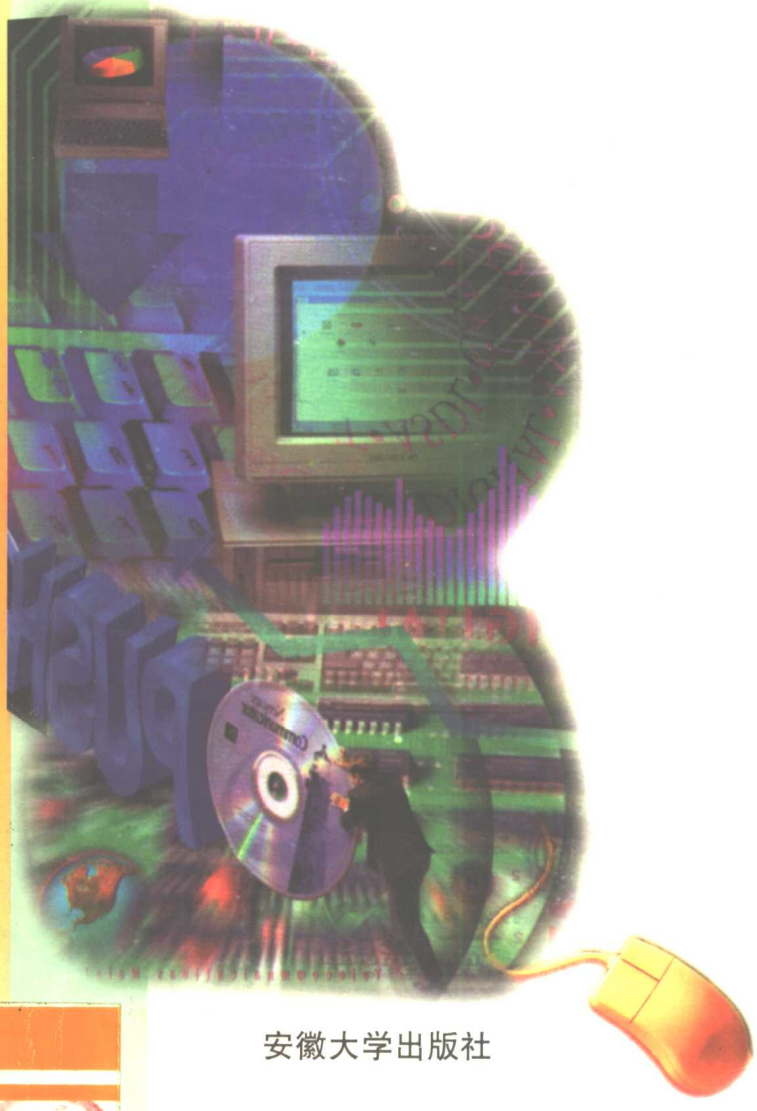
MIANXIANG
DUIXIANG
CHENGXU
SHEJI

面向对象程序设计

⊙王浩 编著



安徽大学出版社

C++ C++ C++

面向对象程序设计

⊙王浩 编著

安徽大学出版社

图书在版编目(CIP)数据

面向对象程序设计/王浩编著．－合肥:安徽大学出版社,1999.6
ISBN 7－81052－213－2

Ⅰ.面… Ⅱ.王… Ⅲ.面向对象语言－程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(1999)第 16811 号

本书已经安徽省高等教育自学考试委员会办公室组织审定,作为全省自学考试统一教材使用。

面向对象程序设计　　王浩　编著

出版发行	安徽大学出版社	印　刷	中国科学技术大学印刷厂
	(合肥市肥西路 3 号 邮码 230039)	照　排	合肥市女娲照排中心
联系电话	总编室 0551－5107719	开　本	787×1092mm　1/16
	发行部 0551－5107784	印　张	16.5
责任编辑	李　虹	字　数	378 千
封面设计	孟献辉	版　次	1999 年 6 月第 1 版
经　销	新华书店	印　次	2000 年 3 月第 2 次印刷

ISBN 7－81052－213－2/TP·21　　**定价 21.30 元**

内容简介

面向对象程序设计是当前程序设计的主流，它使得计算机问题解更符合人的思维活动。本书主要以没有学过C语言，而直接进入C ++学习的读者为对象，从面向对象软件开发的基本原理和基本步骤入手，全面介绍了C ++语言的程序设计的基本方法，重点是强调面向对象的程序设计方法及学习方法。

全书共十一章，第一章主要讨论了面向对象技术的基本概念，面向对象分析和面向对象设计的基本方法，以及面向对象语言，以使读者对面向对象方法有一较全面的了解；第二章介绍C ++的基本程序设计技术；第三章介绍C+ +的构造数据类型，如数组、结构、联合等；第四章介绍C ++函数；第五章介绍指针及引用；第六章介绍C ++对类和对象的支持；第七章介绍类层次和继承性；第八章介绍多态性和虚函数；第九章介绍模板的概念和应用；第十章讨论C ++程序中异常处理；第十一章介绍C ++的输入输出。

本书内容深入浅出，概念清楚，重点突出，所举示例通俗易懂，并全部经上机调试通过。本书既适合于大专院校师生、各类计算机培训班师生的使用，也可作为广大计算机爱好者的自学教材。

前　言

面向对象方法与现实世界存在着自然而直接的对应关系，采用它为现实世界建模时，能更充分地描述与表达现实世界。面向对象技术在软件工程领域已逐渐发展成从面向对象分析、面向对象设计、面向对象编程语言、面向对象软件系统这样一套较完整的软件开发方法学。它已逐渐取代结构化方法，成为90年代乃至下个世纪主流的软件开发方法。其实，面向对象方法中有关模块化、数据抽象和信息隐蔽等概念也是继承了70年代软件工程的成果，面向对象方法的贡献是在"对象"这一更高层次上进行了抽象。

面向对象程序设计是面向对象方法在程序语言编码方面的具体应用，它吸收了结构化程序设计的优点，并引进了一些全新的、强有力的概念，从而开创了程序设计的新天地。

面向对象语言分为纯面向对象语言和混合型语言。纯面向对象语言如smalltalk，它摒弃了传统语言的特征，强调对象概念的单纯性，能完全体现面向对象的程序设计思想，但正因为它们与传统程序设计方法差异较大，因而对于广大的使用传统程序设计语言的程序员而言较难掌握，另外这些语言往往执行效率不高，在某些应用领域受到限制。

C ++语言是一种混合型语言，它是在广为使用的C语言基础上发展而来的，它既具有独特的面向对象特征，又保留传统的高效结构化语言C的主要特征。C ++提供给程序员强有力的面向对象软件开发能力，同时又不失去内存运行效率，可以在任何计算机系统上产生高质量的软件产品。

本书共分十一章，第一章着重介绍了面向对象技术的基本概念，面向对象分析和面向对象设计的基本方法，以及面向对象语言，以使读者对面向对象方法有一较全面的了解，并在后面的学习中，始终用面向对象的观点来观察、思考和求解问题。第二章到第五章介绍的C ++支持结构化程序设计的基本机制，它也是后面章节的基础。第六章到第十一章全面介绍了C ++的面向对象机制，它是本书的重点。

学习本书不需要事先学习C语言，实际上没有传统程序设计方法的束缚，面向对象的思想更容易领会。本书例题均已调试通过，可在Turbo C ++或Borland C ++集成环境下运行。

由于时间仓促，加之作者水平有限，本书不妥之处再所难免，恳请读者予以批评指正。

作者

1999年4月

目　录

第一章　面向对象技术导论

面向对象(Object Orientation)技术是90年代软件技术的主流。面向对象的观点,首先是在程序设计语言中得到表达与实现,并迅速发展到系统分析与系统设计中。本章首先介绍面向对象技术的形成与发展,然后详细说明了面向对象的基本概念,阐述了面向对象的系统分析和设计的基本原理和方法,最后简要介绍了面向对象程序设计语言C++的发展过程和特点。

1.1 面向对象技术的形成和发展

1.1.1 程序设计方法的变迁

计算机程序设计是一门年轻的学科,第1台可编程的计算机诞生在40多年前,在这40多年内,程序设计方法经历了很大的发展,由面向机器变为面向程序员。

1.线性程序设计

最初的计算机程序是用二进制代码(机器语言)书写的。随着计算机技术的发展,高级语言开始使用,它代替了用位(bit)和字节(byte)的考虑方法,具有很好的语言表达能力,主要用于科学计算和商业事务处理,如Fortran,Cobol,Algol,Basic等语言。

随着计算机计算机性能的提高和软件规模的不断扩大,这些以语句序列作为程序的线性程序设计语言的局限性愈加暴露。在实际应用中需要重复使用已有的程序代码,这在线性程序设计中是很困难的。线性程序趋于运行在一个长序列中,这使得它们的逻辑难于理解,这些程序由跳转来控制,而又常常缺乏程序如何跳转和为什么跳转的明显标志。另外,线性语言所有数据项都是全局的,它可以被程序中的任何部分修改。这些都给线性程序的管理和维护带来很多问题,甚至陷入"软件危机"之中。

2.结构化程序设计

为了克服软件危机,提高程序的可靠性和可维护性,60年代末提出了结构化程序设计方法。结构化程序是通过把程序的主要功能分开,然后变成程序中函数(过程)的基本片断来建立,程序从语句序列转向将程序作为相互作用的模块集合。结构化程序设计语言自70年代以来获得广泛应用,其中,Pascal,C语言是这类语言的代表,而早期的语言如Basic,Fortran也支持结构化程序结构。结构化语言的共同特点是都支持结构化的程序设计原理。按照这一原理,程序中的任何逻辑问题均可用"顺序"、"选择"和"循环"3种基本结构加以描述。

结构化程序设计引入了一个很重要的概念——抽象。抽象可定义为忽略事物或系统的

细节而集中注意力于其本质的方面。在结构化程序中,知道一个执行特定任务的给定过程已经足够了,该任务如何被执行并不重要。而该过程是可靠的,可以在不必知道它如何正确地完成其功能的情况下使用。即任何一个完成明确定义功能的操作都可以被使用者当作单个的实体看待,尽管这个操作实际上可能由一系列更低级的操作来完成,这被称作功能性抽象(或称过程抽象),它是结构化程序设计的基石。

随着程序复杂程度的增加,对其处理的基本数据类型的依赖也增加。程序中的数据结构与在其上进行的操作同样重要。而且随着程序大小的增长,这一点变得更为明显。在一个结构化程序中的许多过程中要处理数据类型。当那些数据类型发生变化时,程序中对那些数据类型起作用的每个地方都必须加以修改。这在包含成千上万行程序、几百个函数的程序中可能是十分花费时间的,而且可能经常遇到挫折。

当多个程序员组成一组进行一个应用程序的开发时,结构化程序设计的进一步弱点就暴露出来了。在一个结构化程序中,每个程序员都被赋以建立一套特定的函数和数据类型的任务。因为不同的程序员处理相对于共享数据类型的不同函数,一个程序员对数据项作的修改必定在小组其他人的工作中反映出来。由于结构化程序更易于在小组的情况下使用,组中成员之内的错误交流可能导致很花费时间的重写。

3. 数据抽象

数据抽象是对数据而言,而功能抽象是对操作而言。所谓数据抽象,就是定义了数据类型和施加于该类型对象的操作,并限定了对象的值只能通过使用这些操作修改和观察。用数据抽象,就可以在不必关心其具体实现细节的情况下使用数据结构和数据项。例如,浮点数在所有程序设计语言中都被抽象。当给浮点数赋值时,不必关心其确切的二进制表示。为了将浮点数相乘不必知道二进制乘法是如何进行的。重要的是浮点数以正确的和可理解的方式进行工作。

数据抽象使你可以不必再关心不重要的细节了。如果程序员必须知晓程序功能的每一个细小方面,则只能写出很少一点程序。在所有程序设计语言中象浮点数这样复杂的组成元素都有数据抽象,如数组、结构等。而面向对象程序设计语言可以让你定义你自己的抽象数据类型。

4. 面向对象程序设计

面向对象的风范是在结构化程序设计概念和数据抽象的基础上建立起来的。基本的变化是面向对象的程序是围绕被操作的数据而设计的,而不是围绕操作本身。计算机执行的工作被称为数据处理。数据和动作在一个基本的水平上连接,每一个都要求另一个有目的。面向对象程序使这个关系更为明显。

面向对象程序设计把数据结构和操作结合起来,而这正是我们如何考虑世界的方法。我们通常把给定的对象类型和一组特定的动作结合起来。例如,我们知道汽车有轮子,可以移动,可以通过方向盘来改变方向。类似地,我们知道树是有木质茎和叶子的植物。因此我们自然认为用汽车做的事不能用树来做,反之亦如此。我们不能操纵树的方向;而给汽车浇水,也不会使汽车长大。面向对象程序设计指定其数据类型的性质和行为,这使我们可以确切地知道各种数据类型的作用及如何使用。

在面向对象程序中可以在既相似又有区别的数据类型之间建立关系。人们自然地将事物分类,我们常常把新概念与已有概念联系,而且可以基于事物之间的关系上进行演绎。通

常我们用一个树状结构来概念化世界，后继的细节层建立在前面的一般原则之上。面向对象程序以同样自然的方法工作，使新的数据/操作框架建立在已有的框架基础上，在增加新功能的同时结合基础框架的功能。

面向对象程序设计不是要摒弃结构化程序设计方法，相反，它是在吸收结构化程序设计优点的基础上，引进了一些全新的、强有力的概念，从而开创了程序设计的新天地。面向对象程序设计方法把可重用性视为软件开发的中心问题，通过装配可重复使用的软构件来生产软件。

1.1.2 面向对象技术的应用和发展

本世纪 70 年代末，面向对象方法学的一些基本概念已在系统工程领域内萌发出来，如对于系统中的某个模块或构件可表示为问题空间的一个对象或一类对象。到了 80 年代，面向对象的程序设计方法得到了很快的发展，并显示出其强大的生命力。因而使得面向对象技术在系统工程、计算机、人工智能等领域得到了广泛的应用。90 年代以来，随着面向对象技术的不断成熟，其应用向更高的层次、更广泛的领域发展。

如同结构化方法是从结构化程序设计语言发展而来，面向对象方法也是由面向对象程序设计语言发展而来。程序设计语言的发展过程是抽象程度不断演变和提高的过程。1967 年第 1 个具有面向对象特征的 SIMULA 语言问世标志着面向对象方法的诞生。80 年代初 Xerox PARL 推出 Smalltalk－80，使得面向对象引起了广泛的注意。而由 Bell 实验室设计的 C＋＋，使得 OOPL 走向大众化。然而，OO 方法从程序设计语言开始，并不意味着它只是一种程序设计方法，甚至是某种程序设计语言，它的意义要深远的多。

在 OOPL 蓬勃发展的 80 年代，人们基于以往已提出的有关信息隐蔽和抽象数据类型等概念，以及由 Moudula－2，Ada 和 Smalltalk 等语言所奠定的基础，再加上客观需求的推动，逐步地发展和建立起较为完整的面向对象的软件系统的概念和机制。而将这些基本概念和运行机制应用到其它各个领域，就得到了一系列相应领域的面向对象的技术和应用：

（1）面向对象分析（OOA——Object Oriented Analysis）。系统分析过程是对一个问题空间的研究，OOA 模型描述了表示某个特定应用领域中的对象，以及各种各样的结构关系和通信关系。

（2）面向对象设计（OOD——Object Oriented Design）。从 OOA 到 OOD 是一个逐渐扩充模型的过程。在分析阶段，把系统分解成实体和关系，而设计则是解决这些实体和关系如何实现。

（3）面向对象语言（OOL——Object Oriented Language）。在这种语言中，可以把数据和处理数据的过程结合为一个对象。对象既可以象数据一样被处理，又可以象过程一样描述处理的流程和细节。

（4）面向对象数据库（OODB——Object Oriented Data Base）。把对象作为存取和检索的单位，把传统数据库的数据定义语言和数据操作语言融为一体，这种概念也是构成语义数据库和知识库的基础。

（5）面向对象的智能程序设计（Object Oriented Intelligent Programming）。把知识系统中的智能实体作为对象，一个对象所具有的知识是该对象的静态属性，一个对象所具有的知识处理方法则是该对象的智能行为的体现。

(6) 面向对象的体系结构(Object Oriented Architecture)。能支持对象的描述、存储和处理的计算机体系结构。一些新的计算机体系结构,如某些多处理系统、神经网络计算机及连接机制等,均试图支持面向对象的程序设计和软件系统的概念。

面向对象技术自产生以来已经得到了很大的发展,并且已在计算机科学、信息科学和系统科学中得到了有效的应用,显示出其强大的生命力。面向对象方法学的最基本点是尽可能地模拟人的思维方式,并且和方法学相适应的面向对象技术也提供了这样的可能性:即可以随着对某个系统的需求逐步具体的过程而逐步地设计和实现这个系统。

未来,面向对象技术将会在更深、更广、更高的方向上取得进展:

(1) 更深的方向。如面向对象技术的理论基础和形式化描述;用面向对象技术的概念设计操作系统。

(2) 更广的方向。如面向对象的知识表示;面向对象的仿真系统;面向对象的多媒体系统;面向对象的虚拟现实系统。

(3) 更高的方向。如从思维科学的高度来丰富面向对象方法学的本质属性,突破现有的面向对象技术的一些局限、研究统一的面向对象的范式(Paradigm)等。

1.2 面向对象技术的基本概念

面向对象技术是分析问题和解决问题的新方法。其基本出发点就是尽可能按照人类认识世界的方法和思维方式来分析和解决问题。客观世界是由许多具体的事物或事件、抽象的概念、规则等组成。因此,我们将任何感兴趣或要加以研究的事、物、概念都称为对象。面向对象的方法正是以对象作为最基本的元素,它也是分析问题、解决问题的核心。用计算机解决问题时需要用程序设计语言对问题的求解加以描述,实质上软件是问题求解的一种表述形式。显然如果软件能够直接地表现求解问题的方法,则软件不仅易于被人理解,而且易于维护和修改,从而提高了软件的可靠性和可维护性。

在面向对象的设计方法中,对象和消息传递分别是表现事物和事物间相互联系的概念。类和继承是适应人们一般思维方式的描述范式。方法是允许作用于该类对象上的各种操作。这种对象、类、消息和方法的程序设计范式的基本点在于对象的封装性和继承性。通过封装能将对象的定义和对象的实现分开,通过继承性能体现类与类之间的关系,以及由此带来的动态连编和实体的多态性,从而构成了面向对象的基本特征。下面将对这些概念和特征作进一步介绍。

1.2.1 对象(Object)

对象是面向对象技术的核心。目前对对象的基本定义尚未统一,但以下几点则是公认的:

1. 对象是人们要进行研究的任何事物

对象不仅能表示结构化的数据,而且也能表示抽象的事件、规则以及复杂的工程实体。这是结构化方法所不能做到的。

客观世界的问题都是由客观世界的实体及实体之间的相互关系构成的,我们把客观世界的实体称之为问题空间的对象。一本书可以是一个对象、一家图书馆也是一个对象。可

见，世界上的各个事物都是由各种“对象”组成的，任何事物都是对象，是某个对象类的一个元素。复杂的对象可由相对比较简单的对象以某种方法组成。

本质上，我们用计算机解题是借助某种语言规定对计算机实体施加某种动作，以动作的结果去映射解，我们把计算机实体称之为解空间的对象。

2. 对象实现了数据和操作的结合

对象的状态用数据来描述，对象还应有操作，用以改变对象的状态。对象将数据及对数据的操作封装在一起，使对象具有较强的独立性和自主性。

从动态的观点来看，对象的操作就是对象的行为。问题空间对象的行为是千变万化、丰富多采的，而解空间对象的行为则是死板的。因此，只有借助于复杂的算法才能操纵解空间对象而得到解，这就是所谓的“语义断层”。传统的程序设计语言限制了程序员定义解空间对象。而面向对象语言提供了“对象”概念，将数据和操作封装在一起，程序员就可以自己定义解空间对象，并与问题空间对象相对应，减少了语义断层。

3. 对象应具有唯一的识别功能

对象应具有对象标识符(Object Iedntify—OID)，OID 值可用来唯一而且永久地表示对象。

在一个完全的面向对象系统中，在建立对象时，系统授予新对象以唯一的 OID，它将与这个对象永久结合在一起，不管这个对象的状态和结构经历了怎样的改变。这如同一个人的身份证号码一样，一旦办证则永不改变，尽管人的年龄、学历、工作、住址、甚至名字都可以改变，而这些变化不影响它的身份证号码。不过在面向对象系统中，对象被清除后，OID 的值可被“回收”，再由系统分配给新创建的对象。此外，对象还应有对象名，对象名代表了数据和操作的一体化。

4. 对象必须参与一个或一个以上的对象类，因而成为类的实例

在大多数面向对象系统中，类反过来也可看作是对象，从而实现了类和对象的统一。为了理解对象作为类实例，现以飞机类的实例来说明。显然，每一架飞机都是一个具体的对象，如波音 747、空中客车等。如果将各种各样的飞机抽取它们共同的特性，例如，凡是飞机都能在空中飞行，具有改变飞行方向、控制飞行高度和飞行速度的操作特性；凡是飞机都有机名、机型、飞行高度、飞行速度等数据，用以描述飞机的结构特性；飞机还有严格的飞行安全约束规则，如飞行条件、起飞和降落的条件等。因此，可将各种各样的飞机抽象为“飞机”类，而每架飞机自然就可以作为这个类中的一个实例了。

1.2.2 消息(Messages)和方法(Methods)

如何要求对象完成一定的处理工作？对象之间如何进行联系？所有这一切只能通过消息来实现。消息是用来请求对象执行某一处理或回答某些信息的要求；消息统一了数据流和控制流；某一对象在执行相应的处理时，如果需要，它可以通过传递消息请求其他对象完成某些处理工作或回答某些信息；其他对象在执行所要求的处理活动时，同样可以通过传递消息与别的对象联系。因此，程序的执行是靠在对象间传递消息来完成的。

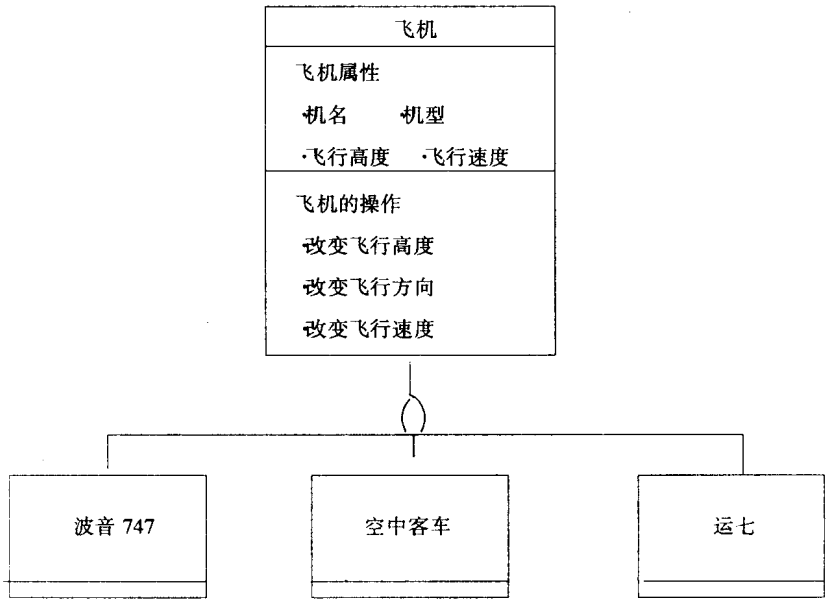


图 1.1 “飞机”类的实例——对象

发送消息的对象称为发送者,接受消息的对象称为接受者。消息中只包含发送者的要求,它告诉接受者需要完成哪些处理,但并不指示接受者应该如何完成这些处理。消息完全由接受者解释,接受者独立决定采用什么方式完成所需要的处理。一个对象可以接受不同形式、不同内容的多个消息;相同形式的消息可以送往不同的对象;不同的对象对于形式相同的消息可以有不同的解释,从而做出不同的反应。对于传来的消息,对象可以返回相应的回答信息,但这种返回不是必须的,这与子程序的调用/返回有着明显的不同。

方法是描述对象对消息的响应。把所有对象分成各种对象类,每个对象类都定义一组所谓的“方法”,它们实际上可视为允许作用于该对象上的各种操作。图 1.2 给出了对象分解图。

消息的形式用消息模式刻画,一个消息模式定义了一类消息,它可以对应内容不同的消息。例如,定义“+ Interger”为一消息模式,而“+3”、“+4”等都属于该消息模式的消息。对于同一消息模式的不同消息,同一个对象所做的解释和处理都是相同的,只是处理结果可能不同。对象固有的处理能力按消息分类,一种消息模式对应对象的一种处理能力。这种处理能力是通过该模式及消息引用表现出来的。所以,只要给出对象的所有消息模式及相应于每一个消息模式的处理能力,也就定义了一个对象的外部属性。

消息应当包含:

(1) 选择器实质上是方法名;

(2) 一个或多个变量

由选择器从目标对象的有关方法中选择合适的方法。当选中相应的方法后,就将消息中的变量连编到方法的参数中。这个过程与传统结构化语言的过程调用相似。但较重大的差异在于面向对象技术的消息中,变量连编既可以在编译时进行(与传统方法相同),也可以

在运行时进行。变量在运行时的连编就称为动态连编或迟后连编，后面将会介绍。

当一个面向对象的程序运行时，一般要做三件事：首先，根据需要创建对象；其次，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（或从用户到对象）；最后，若不在需要该对象时，应删除它并回收它所占用的资源。

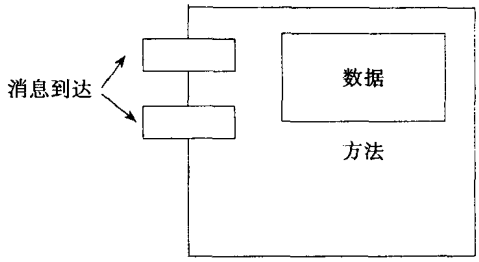


图 1.2 对象分解图

由此可见，面向对象的设计方法放弃了传统语言中控制结构的概念，以往的一切控制结构的功能都可以通过对象及其相互间传递消息来实现。

1.2.3 类(Class)和类层次(Class Hierarchy)

在面向对象程序设计中，"对象"是程序设计的基本单位，相似的对象可以和传统语言中的变量和类型关系一样，归并到一类中。程序员只需要定义一个类对象就可以得到若干个实例对象了。

1. 对象类的定义

将具有相同结构、操作，并遵守相同约束规则的对象聚合成一组，这组对象的集合就称为对象类，简称为类。

具体来说，类由方法和数据组成，它是关于对象性质的描述，包括类说明和内部实现两个方面。见图 1.3。

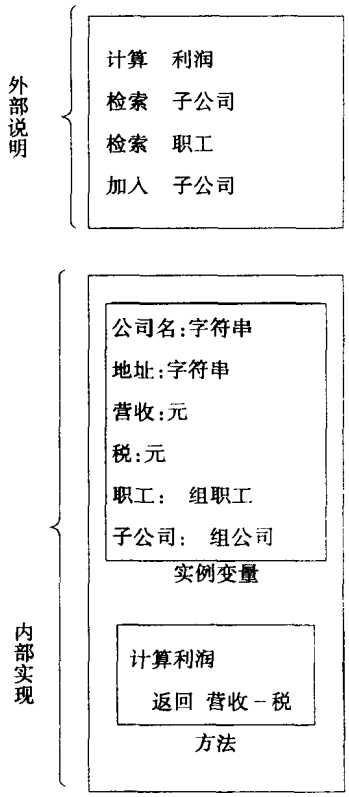


图 1.3 "公司"类的表示

2. 类说明

类说明也称类的外部特性或外部接口。它往往由一组操作符组成，通常这些操作符可以送到目标对象中。例如，图 1.3 所示的"公司"类的操作符有：

(1)计算利润：计算某公司的利润；

(2)检索子公司：检索某公司下属的全部子公司；

(3)检索职工：检索某公司中的职工；

(4)加入子公司:增加一个新的子公司到某公司中;

3. **内部实现**

类的内部实现是类的内部表示及类说明的具体实现方法,通常由系统开发人员通过编程实现。对用户来说,是不必了解内部实现的。即内部实现是对用户实行信息隐蔽的。

在图 1.3 的例子中,"公司"类的内部表示有:公司名、地址、营收、税、职工、子公司等,也就是公司的属性。而对应每一个操作符,都应在内部实现中有一个具体实现这一操作的程序模块,亦即是方法。

类的概念是面向对象所特有的重要概念。其最鲜明的特色是将数据的结构和数据的操作都封装在类中,并实现了类的外部特性与内部实现的隔离,也就是实现了将使用对象、类的用户与具体实现对象、类的开发者区分开,从而使面向对象技术具有良好的模块化特性,进而为复杂大系统的分析、设计、实现提供先进的方法。

4. **类层次**

一个类的上层可以有超类(superclass),下层可以有子类(subclass),形成一种层次结构或格(lattice)结构关系。类的层次结构和格结构可用来描述客观世界中"概括"的抽象关系。通常,越在上层的类越具有普遍性和共性,越在下层的类越细化、专门化。图 1.4 给出了一个类层次的例子,其中小学生、中学生、大学生、研究生可概括为"学生";教授、讲师、助教可概括为"教师";学生、教师、职工又可进一步用"人"加以概括。上层的类"人"称为超类,下层的学生、教师、职工等类就是"人"类的子类。同理,学生是小学生、中学生、大学生、研究生的超类,而它们则是学生的子类。这种结构就是类层次结构,其特点是每个子类只有一个超类。如果一个子类有一个以上的超类,则称为类格结构。例如,在研究生类中,既攻读学位又从事教师工作时,就出现既属于学生又属于教师的情况。

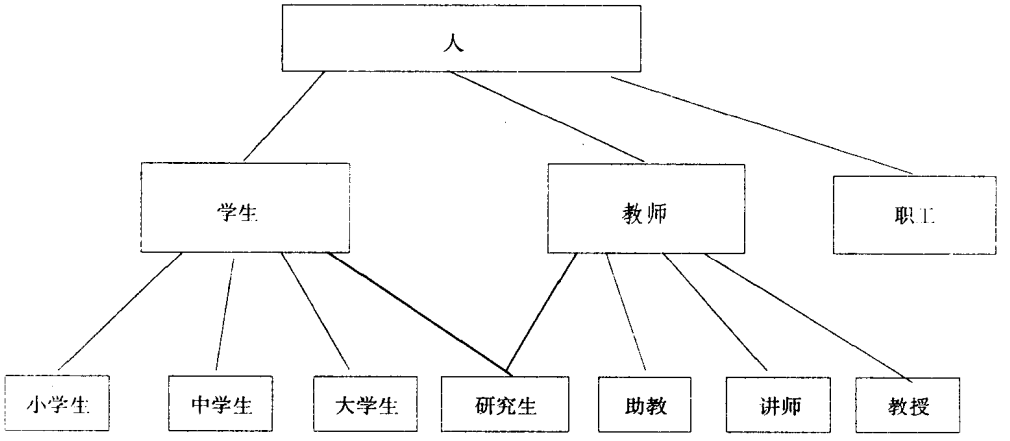


图 1.4 类层次结构

1.2.4 继承性(Inheritance)

继承性是自动地共享类、子类和对象中的方法和数据的机制,它是面向对象技术所独有的。每个对象都是某个类的实例,一个系统中类对象是封闭的。如果没有继承性机制,则类对象中的数据和方法就可能大量重复。类的层次结构的一个重要特点就是继承性,子类可