面向对象方法与C++新版本

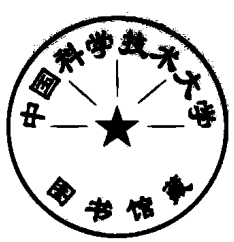
古新生
王拓　王伟

西安交通大学出版社

面向对象方法与C++新版本

古 新 生

王 拓　王 伟



西安交通大学出版社

内 容 简 介

本书以面向对象方法和C++语言为主题，反映了现代软件工程和系统工程的新进展。首先以三分之一篇幅综述面向对象方法及其编程风格，使读者能较快地掌握这两个新领域的主要概念；然后以剩余篇幅详细剖析功能最强的面向对象的编程语言C++的特点与应用，采用和C语言对比的方法，突出C++和C不同的方面，如对象类、导出类、继承性，多形性和虚函数等特色，因而实用性强。

为使读者学习本书时掌握重点，各章末尾均配备了小结、思考题和习题；并列出深入学习的参考文献。

本书可供广大应用计算机的人员、高等院校有关专业师生学习使用，也可作为从事高技术软科学的科研人员、专家教授、工程技术人员、决策人员的重要参考书。

(陕)新登字007号 TS427/13

面向对象方法与C++新版本

古新生

王 拓 王 伟

责任编辑 赵丽平

*

西安交通大学出版社出版

邮政编码:710049

西安交通大学出版社印刷厂印装

陕西省新华书店经销

*

开本 787×1092 1/16 印张 18 字数:429千字

1992年12月第1版 1993年2月第1次印刷

印数:1—10050

ISBN7-5605-0506-6/TP·54 定价:12.00元

序　　言

怀着诚挚与殷切的心情，向广大读者，尤其是从事高技术相关领域研究的朋友们奉献《面向对象方法与C++新版本》一书。我们热切地期望这本书能帮助读者较快地了解被称为是"研究高技术的好方法"之一的面向对象方法以及自80年代以来越来越流行的面向对象编程语言，尤其是深入学习与掌握90年代日益普遍使用的C++。

为什么面向对象方法是研究高技术的一种好方法呢？为什么面向对象语言越来越流行呢？

首先，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；此外，将现实世界的问题转换为计算机所能理解的问题时，其映射量最少。

其次，面向对象方法为高技术研究中大量存在的复杂大型软件工程提供远较传统结构化方法更为优越的新方法；目前已逐渐发展成从面向对象的分析(O-OA)→面向对象的设计(O-OD)→面向对象的编程语言(O-OPL)→面向对象的软件系统(O-OS)这样一套较完整的软件开发方法学。这种方法之所以明显地优于结构化方法主要基于如下的特点：

1) 面向对象方法较结构化方法具有更高的抽象性。尤其具有允许用户定义复杂的数据类型功能，使它能表示传统的结构化方法所不能表示的复杂工程领域、专家规则知识等问题。

2) 完整的封装性保证对象类及对象可作为独立性很强的模块，为大型软件提供可靠的"软件集成"的单元模块。大大提高了软件的可重用性。

3) 继承性为面向对象方法所特有的功能很强的性质。在进行系统分析时，它提供了组织信息分类(共性-个性)的强有力方法；在设计实现阶段，类实例与类、子类及超类的继承机制提供了强大的共享性能，大大简化了类、实例的创建与定义，从而加速系统开发效率，进一步提高了软件的可重用性和可扩充性。

4) 多形性尤其是具有动态联编的多形性使系统更具有灵活性，多形性与继承性相结合更进一步提高了系统的开放性和集成功能。

最后，还要特别指出的是面向对象方法很容易与快速原型方法相结合，这真像"如虎添翼"，进一步使大型复杂的软件找到一条准确与用户交互、共同攻克"难以搞清需求"的难关，大大缩短系统开发周期，并成为降低投资、降低风险、提高软件的质量与可靠性的有效途径。

本书前三章介绍面向对象方法：第1章综述此方法，重点在阐明面向对象方法之特色；第2章试图适当扩充视野，使读者了解除了面向对象的编程语言之外，尚有正在研制和发展的面向对象的系统分析、设计与实现。第3章就面向对象方法的核心、成熟的面向对象编程语言进行介绍，这章作为承上启下之接口。第4章至第8章则详细而深入介绍日益流行的面向对象编程语言C++新版本，突出介绍C++的面向对象方面的特性与编程风格：第4章侧重交代C++对C所作的扩充，第5章介绍极其重要的对象类与对象，第6章介绍类层次中的导出类(子类)及其继承性，第7章讨论多形性与虚函数，第8章以C++如何进行输入输出为例，具体说明C++十分有用的类库机制。最后一章通过举例详细说明如何用C++编程，使读者全面地领略面向对象的编程风格。

在编写本书时，力图体现如下特色：

1） 求实 本书作为我们从事面向对象方法研究达五年多的结晶。力争所写内容都经过我们的理解、消化与吸收，尤其是C＋＋编程举例，90％的例子都经过上机调试与运行。做到决不生硬翻译语言手册，避免用晦涩难懂的语句，力求深入浅出，说理清楚。

2） 创新 在编写此书时，我们努力收集可能收集到的国外最新书刊和国内外最新发展动态。在大量阅读最新资料的基础上，进行理解与分析，力图反映国内外在这个领域上研究的最新动态与水平。

3） 易懂 考虑到国内尚未有全面介绍面向对象方法的书，为了使读者尽快学习与掌握此方法，我们力求做到：

- 叙述思路清晰，层次分明，逻辑性强；
- 为便于理解，理论部分注意图文并茂；编程语言C＋＋部分对关键概念必举实例；
- 每章有总结、思考题或习题以及参考资料；
- 书稿写成后，经过一次对研究生的系统教学实践，证明书稿易于为学生所接受。

本书有幸得到国际面向对象委员会成员、著名的计算机科学专家蔡希尧教授的认真、细致的审阅，万分感激他所提出的宝贵意见。

本书由古新生、王拓、王伟三人密切合作写成。其中古新生撰写第1～3章并负责全书主编；王伟撰写第4～6章；王拓完成第7～9章。由于水平所限，再加上面向对象方法发展异常迅猛，肯定有许多不足之处，诚恳希望读者指正。

从编写到出版本书绝非我们三人所能完成，我们衷心感谢西安交大外教中心为我们提供了最新国外书籍；研究生李智芳、胡正辉、陈敬、周鹏飞在上机实践与学习本课程时对书稿所提出的意见；更感谢大量不知名的为此书出版进行过辛勤劳动的同志们。

本书责任编辑赵丽平同志进行了大量细致和认真负责的文字加工工作，对她的辛勤劳动表示崇高的敬意。

编 著 者

1992年元月

目　　录

第1章　面向对象方法导论

一种新颖、具有独特的优越性的新方法正引起全世界越来越强烈的关注和高度的重视，它就是被誉为“研究高技术的好方法”[9]的面向对象方法(Object-Oriented Paradigm，简称为O-O方法)。近十多年来，在对O-O方法如火如荼的研究热潮中，许多专家与学者预言：正如70年代以来结构化方法对计算机技术应用所产生的巨大影响和促进一样，80年代以来O-O方法将从O-O认识(建模)方法论，O-O的系统分析与设计方法，O-O编程风格等领域强烈地影响、推动和促进着一系列高技术的发展和多学科的综合。

为什么O-O方法具有如此强大的功能和迷人的魅力呢？本章试图综述O-O方法来回答。首先介绍O-O方法的由来与发展，再详细说明O-O方法的重要基本概念，最后阐明O-O方法的独到之处。

1.1　面向对象方法的由来与发展

面向对象方法起源于面向对象的编程语言(简称为O-OPL)。随着计算机应用日益普及，编程人员致力于研究和开发各种各样的高级语言，研究编程的构造、规范以及基本原理，以便更有效地编写各种应用程序，同时，也力求使编写、调试好的程序便于运行和维护。

1.1.1　编程中的问题

50年代后期，在编写Fortran的大型程序时出现了变量名在不同的程序部分发生冲突的问题。为此，Algol语言的设计者决定采用“阻挡”(barriers)来隔开程序段中的变量名。这样就产生了Algol 60以“Begin…End”为标识的程序**块**(block)。由于程序块内的变量名是**局部**的，它们的使用就不会与程序中其他块的同名变量相冲突。这是在编程语言中首次提供**保护**(Protection)或**封装**(Encapsulation)的尝试。现在，程序块结构已广泛用于C，Pascal，Ada等各种语言中。

1.1.2　面向对象编程语言的产生

60年代中后期，Simula-67语言的设计者(Dahl和Nygaard在1966年、以及Myhrbang在1970年)采用了Algol的程序块概念，并加以推进，提出了**“对象”**(Object)的概念。虽然Simula源出于Algol，但它主要用于仿真；Simula的对象具有“自身独立存在”并能在仿真过程中以一定的含义彼此通信的特点。这是在编程语言中开始使用**数据封装**(data encapsulation)。遗憾的是，由于Simula编译器价格昂贵(60年代销售为2万美元)，使它无法在市场上得到推广。

1.1.3　面向对象编程语言的发展

70年代，随着对管理大型程序的迫切需要的增长，许多语言设计者追求实现“数据抽象”的概念。由Xerox Paloalt公司经过对Smalltalk72，74，76连续不断的研究、改进之后，终于在1980年推出商品化的Smalltalk-80。它在系统设计中强调对象概念的统一，引入对象、对象类、方法、实例等概念和术语，采用动态联编和单继承性机制。它还建立以O-O编程语言为核心，集各种软件开发工具为一体，建立O-O计算环境，配有很强的图形功能和多窗口用户界面。因

此，Smalltalk 的商品化标志了面向对象的编程语言已建立较完整的概念和理论，并推向实用。Smalltalk 作为一种新的、纯粹的面向对象编程语言，同时又体现和发展了面向对象方法的许多重要的概念，对面向对象方法学的形成和发展起了重大的作用。正是通过 Smalltalk-80 的研制与推广应用，使人们注意到面向对象方法所具有的模块化、信息封装与隐藏、抽象性、继承性、多形性等独特之处，这些优异特性为解决大型软件管理、提高软件可靠性、可重用性、可扩充性和可维护性提供了有效的手段与途径。

与 Smalltalk 语言研制的同时，有关抽象数据类型的基础理论和数学研究亦大量研究与发表（如 Goguen，Thatcher，Wegner 及 Wright 等人在 1975 年、Guttag 于 1977 年发表的文章均属此类研究）。它为 O-O 方法研究提供了初步的理论。

1.1.4 高技术研究有力地促进面向对象方法的发展

近 10 年来，由于一系列高技术项目的研究，如智能计算机、计算机集成制造系统（CIMS）、计算机辅助系统工程（CASE）、办公信息系统（OIS）等项目的研究，迫切要求进一步改进系统研究的方法、提高软件研究的质量。由于传统的结构化方法已很难满足这些复杂大系统的要求，因而强烈地期望出现更先进、功能更强的新方法。应用的需求促使 O-O 方法迅猛地向前发展：从面向对象的编程语言进一步迈向 O-O 认知方法学、O-O 系统分析方法学和 O-O 系统设计方法学。采用 O-O 方法已建立了商用的面向对象数据库管理系统和面向对象的信息（包括数据与知识在内）管理系统原型。

1986 年举行了首届“面向对象编程、系统、语言和应用（O-OPSLA）”国际会议，其后每年都举行一次。进一步标志 O-O 方法的研究已普及到全世界；此外，在软件工程、知识工程、数据库技术、人工智能等领域的国际学术会议与刊物上，也专门开辟了 O-O 方法专题讨论和刊登 O-O 方法的最新论著。

1.1.5 面向对象方法当前的研究动态

当前，在研究 O-O 方法的热潮中，有如下的主要研究领域：

1. 第五代计算机的研究

日本将 O-O 方法的设计思想和编程风格引入到世人瞩目的第五代计算机研究（也称智能计算机）计划中，如 PSI 机的系统程序设计语言 ESP 引入了 O-O 编程风格，PIM 的知识程序设计系统 MA DALA 中也引入 O-O 方法。因为 O-O 方法可将知识片看作对象，并为相关知识的模块化提供方便，所以在知识工程领域也越来越受到重视。

2. 新一代操作系统的研究

采用 O-O 方法来组织新一代操作系统具有如下优点：

1） 采用对象来描述操作系统所需要设计、管理的各类资源信息，如文件、打印机、处理机、各类外设等更为自然。

2） 引入 O-O 方法来处理操作系统的诸多事务，如命名、同步、保护、管理等，会更易实现、更便于维护。

3） O-O 方法对于多机、并发控制可提供有力的支持，并能得当地管理网络，使其更丰富和协调。

3. 多学科的综合研究

当前，人工智能、数据库、程序设计语言研究有汇合的趋势。例如，在研究新一代数据库系统（或称智能数据库系统）中，能否采用人工智能思想与 O-O 方法建立描述功能更强的数据模

型？能否将数据库语言与编程语言融为一体？为了实现多学科的综合，O-O 方法是一个很有希望的汇聚点。事实上，目前已涌现出不少面向对象的应用系统产品。

4. 新一代面向对象的硬件系统的研究

要支持采用 O-O 方法设计和实现的软件系统的运行，必须建立更理想的能支持 O-O 方法的硬件环境。目前采用松耦合(分布主存)结构的多处理机系统更接近于 O-O 方法的思想。因为松耦合的对象可占用不同的处理机，使多处理机都能处于忙碌状态从而充分发挥每个处理机的效能。

作为最新出现的神经网络计算机的体系结构与 O-O 方法的体系结构具有惊人的类似，并能相互支持与配合：一个神经元就是一个小粒度的对象；神经元的连接机制与 O-O 方法的消息传送也有着天然的联系：一次连接可以看作一次消息的发送。可以预料，将 O-O 方法与神经网络研究相互结合，必然可以开发出功能更强、更迷人的新一代计算机硬件系统。

1.2 面向对象方法的重要术语和基本概念

面向对象方法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、规则等组成的。因此，我们将任何感兴趣或要加以研究的事、物、概念都统称为**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。由此可见，O-O 方法很自然地符合人的认识规律。计算机实现的对象与真实世界具有一对一的关系，不必作任何转换。这样就使 O-O 方法更易于为人们所理解、接受和掌握。下面介绍 O-O 方法常用的重要术语及其基本概念。

1.2.1 对象(Object)

对象是 O-O 方法的核心。然而，在不同的研究领域、不同的学者对“对象”的概念解释各异。尽管目前尚未有统一的定义，但可以抽取出如下的共同认识：

1. 对象是人们要进行研究(感兴趣)的**任何事物**。

从最简单的整数到极其复杂的自动化工厂、航天飞机等都可看作对象。对象不仅能表示具体的实体，也能表示抽象的规则、计划或事件。主要有如下的对象类型：

1) 有形的实体。指一切看得见、摸得着的实物。如计算机、机床、机器人、工件等等，都属于有形的的实体，也是最易于识别的对象。

2) 作用。指人或组织所起的作用。如医生、教师、职工、公司、部门等。

3) 事件。在特定时间所发生的事。如飞行、演出、事故、开会等。

4) 性能说明。制造厂或企业，往往对产品的性能加以全面说明。如机床厂对机床的性能说明、往往要列出机床名、型号以及机床的各种性能指标等等。

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。因此，对象具有很强的表达能力和描述功能。

2. 对象实现了数据与操作的结合

对象具有**状态**，通常用**数据**来描述；对象还应当有**操作**，用以改变对象的状态。从动态的观点来看，对象及其操作就是对象的**行为**。现实世界中，对象的行为是千变万化、丰富多采的；而采用传统的结构化方法在计算机上解题(解决现实世界中的问题)的方法则极其刻板，往往共

享操作而无特殊性，只能采用极其复杂的算法才能操纵对象而获得解，这就是所谓“语义断层”。面向对象方法则试图尽可能自然而灵活地模拟现实世界，向着减少语义断层的方向努力。其中关键的一点就是改变传统方法中将数据与操作（亦称函数或过程）相分离的做法，实现了将数据与操作封装在**对象**的统一体中；O-O 方法还提供一种机制，使得对象内的私有数据有其私有的操作，从而可灵活地专门描述对象的独特行为，在某些问题上可直接模拟现实世界中的对象，这样研制出的软件易于理解与维护。由于对象封装了数据及对数据所进行的操作，从而使对象具有较强的**独立**性和**自治**性，其内部状态不受或很少受外界的影响。这就是说，它具有很好的模块化的特点，可为软件重用奠定坚实的基础。

3. 对象应具有唯一识别的功能

在对象建立时，由系统授予新对象以唯一的对象标识符（Object Identify，简称 OID），OID 值可用来唯一而且永久地标识对象。也就是说，在对象的整个生命期中，它的 OID 值不会改变；不同的对象不可能取相同的 OID 值；对象清除后，OID 值可被“回收”，再由系统分配给新创建对象。此外，对象还应有对象名，对象名代表了数据与操作的一体化。

4. 对象必须参与一个或一个以上的对象类

在面向对象的方法中，对象应当参与对象类（有关对象类的概念在 1.2.2 中介绍）并成为类的**实例**。因此，在 O-O 方法中，对象与类实例是同义词。

在大多数 O-O 方法中，类反过来也可看作是对象，从而实现了类与对象的统一。由于类实例总是由类来创建，因此，可将创建类的类称为**元类**（Metaclass），这时，类就可作为元类的实例。

为了理解对象作为类实例，现以图 1.1“飞机”类的实例来加以说明。显然，每一架飞机都是一个具体的对象，如波音 747，图 154 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（而不是陆地或海洋）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、起飞与降落的条件等。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作为这个类中的一个实例了。

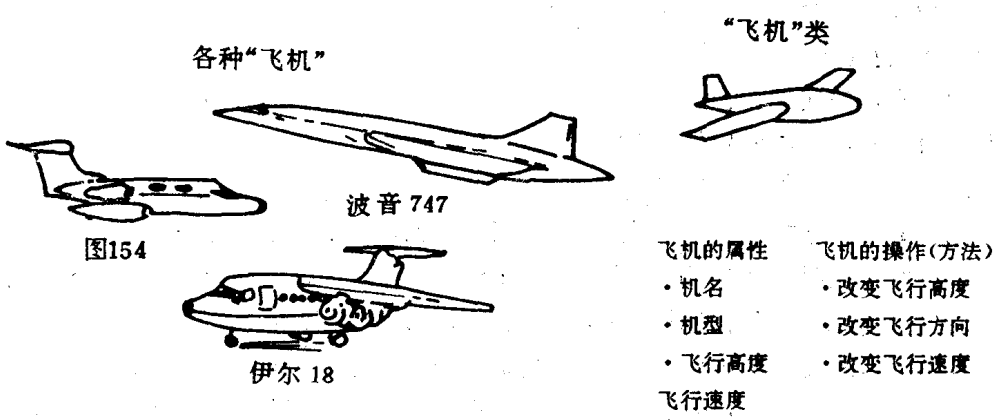


图 1.1 “飞机”类的实例——对象

1.2.2 对象类（Object Class）

面向对象方法的另一重要术语就是对象类，或简称为类。

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，这组对象集合就称为类。具体对类进行定义时，最低限度应包括如下内容：

1) 类名；

2) **外部接口**：用于操纵类实例的外部操作。

（外部操作较典型的应包括目标对象和许多变量）

3) 内部表示；

4) 接口的内部实现。

图 1.2 用“公司”作为类，给出对“公司”类的表示。由图可看出，类的表示通常分成两大部分：**类说明**部分和类**实现**部分。

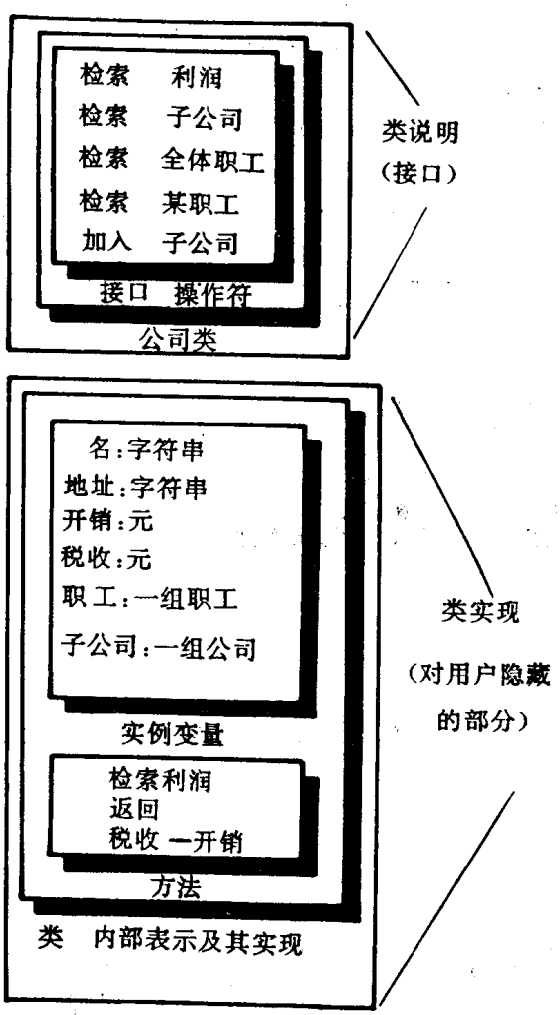


图 1.2 “公司”作为类的表示

2. 类**说明**(Class Specification)

类说明也称为类的外部特性或称外部**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。以“公司”类的操作符为例对其中几个进行解释：

1) 操作符名：检索利润

结　　果：某公司近年的利润

2) 操作符名：检索子公司

结　　果：返回某公司下属的全部子公司名

3) 操作符名：增加子公司

变　　量：新子公司名

结　　果：在某公司的子公司名单中，增加一个新的子公司名

类说明是让用户了解对象类**是什么**和能够**做什么事**，因而也是与用户沟通的“外部接口”或“界面”。

3. 类**实现**(Class Implementation)

类实现是类的内部表示及类说明的**具体实现方法**。也就是详细设计对象类所说明的功能应当如何做，通常由系统开发人员通过编程实现。对用户来说，是不必了解类实现的。也就是类实现是对用户实现**信息隐藏**的。

我们还是以“公司”类为例，说明类实现。

1) “公司”类的内部表示：公司名、地址、支出、税收金、职工等也就是描述公司的属性。

2) “公司”类的内部实现：实现各种操作的详细程序模块。通常，称这些程序模块为“方法”。也就是说，对应于每一个操作符，都应在内部实现中找出具体实现这一操作的程序模块（方法）。关于方法的概念将在下面 1.2.4 详细叙述。

4　对象类概念的精髓

对象类的概念是O-O方法所特有的(结构化方法没有类的概念)重要概念。对象类最鲜明的特色是将数据的**结构**与数据的**操作**都**封装**在对象类中，并实现了类的**外部特性**与**类实现**的隔离，也就是实现了将使用对象、对象类的用户与具体实现对象、对象类的开发者区分开，从而使O-O方法具有良好的**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。

这里还要阐明对象类(class)和类型(type)是不同的概念。类不是类型，类除了具有本身的结构特性、操作外，也可以有**具体的例子**；类型则仅代表数据的抽象描述，即值的集及其操作，但不可能有具体的例子。

类和类型的另一个显著差别在于**静态**与**动态**之别。类型往往是静态的数据抽象描述。而类则具有动态性，也就是可以从类动态地**生成**新的属于该类的对象，隐含了"对象产生器"及"对象存储器"的概念。因此，可以将对象类比拟为一个制造产品的工厂。如图1.3所示。其中，新对象(属于该类的一个实例)可通过调用"新"操作符来建立。因此，若将类比拟为工厂的话，它的输入就是建立新对象的要求。例如，若要在图1.2的例"公司"类中建立这个公司的实例(新对象)，就应输入具体公司中的名，地址等内部变量。采用Smalltalk语言可表达为：

IBM:=New(Company,"IBM","111 Success Street New York")

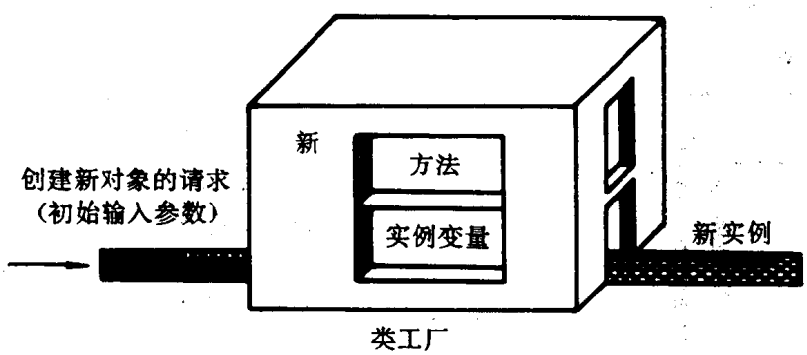


图 1.3　类与工厂的类比

其含义是调用"New"操作符，在Company类中创建新对象IBM公司，IBM公司的地址是"111 Success Street New York".

通过这个操作后，就建立起"公司"类的一个新实例。

5.　**类层次**与**类格**(Class Hierarchy and Class lattice)

对象类之间可能具有层次的结构或格的结构关系。类的层次结构和格结构可用来描述现实世界中"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。例如，图1.4表示小学生、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、工厂、教师、农民又可进一步用"人"加以概括，从而形在类的层次结构。其中，处于上层的对象类"人"称为超类(或称基类)，下层的学生、工厂、教师、农民等对象类就是"人"的"子类"(或称导出类)。同理，小学生、中学生、大学生和研究生可看作是学生的子类，而学生就作为它们的超类。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。如果子类有一个以上的超类，则称为**类格**结构。例如，若研究生子类中，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 故意用粗实线表示研究生子类同时有两个超类的类格结构。

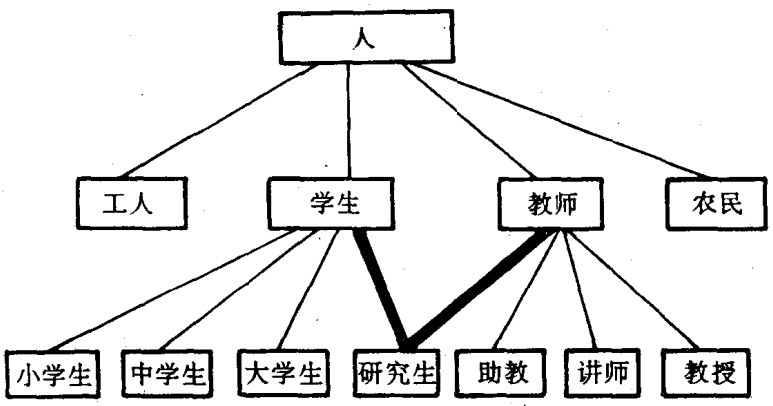


图 1.4 类层次结构

6. 采用类表示知识

1) 面向对象方法与基于框架的知识表示的类同 明斯基(Minsky)在 1975 年提出采用框架(Frame)来表示知识，将知识表示环境中的声明与过程结合于框架内。基于框架系统的基本组成原理是知识包，由框架将许多函数组合起来并使它们在表示知识与知识的推理、演绎中发挥重要的作用。

在框架内所包含的绝大部分概念与 O-O 方法中所包含的概念极其相似，它们之间可用表 1.1 说明其相互的映射关系。

表 1.1 对象类与框架之间的映射关系

O-O 方法	框架表示法
对象类(Class)	框架(Frame)
超类(Super Class)	双亲(Parent)
属性(Attribute)	槽(Slot)
值(Value)	值(Value)
方法(Method)	附加谓词(Attached Predicate)

2) 对象类的定义方式 对象类采用如下的形式定义：

Class 〈类名〉 [:〈超类名〉]
[〈实例变量表〉]
Structure
〈对象的静态结构描述〉
Method
〈对象的操作方法的定义〉
[**Restraint**]
〈限制条件或规则〉