面向对象分析设计与编程

OOA/OOD/OOP

吴炜煜　编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

面向对象分析设计与编程

（OOA/OOD/OOP）

吴炜煜　编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书简洁而全面地综合论述了面向对象分析、设计技术与编程方法。从面向对象技术的一般原理，到C++(Java)的面向对象语言表达；从面向对象系统分析设计，到编程软件环境的使用，讲解循序渐进，前后贯通，逻辑体系合理，教学方法得当。使学习者能够较快地直接掌握面向对象系统的分析设计方法和具备编程实现的能力。

本书可作为大学本科生教材，也适合于作为自学面向对象程序设计的快捷入门参考书。

书　名：面向对象分析设计与编程(OOA/OOD/OOP)
作　者：吴炜煜
出版者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http:// www. tup. tsinghua. edu. cn
印刷者：北京密云胶印厂
发行者：新华书店总店北京发行所
开　本：787×1092 1/16　印张：17　字数：397 千字
版　次：2000 年 4 月第 1 版　2001 年 8 月第 3 次印刷
书　号：ISBN 7-302-01011-0/TP・373
印　数：10001～16000
定　价：19.50 元

前　言

面向对象技术的理论和应用方法，本质上是一种自然地表示客观世界的思维方法，是超越问题论域的复杂性障碍，实现可计算性的软件设计方法。学习面向对象分析、设计和编程(OOA/OOD/OOP)，不仅能够掌握计算机软件设计的一个重要的方法，而且可以帮助养成良好的研究解决实际问题的工作方法和作风。因此，在清华大学课程设置改革中，我们把原来对研究生开设的该课程，在教学内容和方法上结合本科学生情况进行调整，在大学本科三年级开设“面向对象分析设计与编程”课，为学生参加社会实践和毕业设计作准备。经过教学实践，学生普遍反映这门课程十分解渴，对于能力培养大有裨益。

本教材是根据我们的课堂教学讲义编写的，在内容上，既教授面向对象分析的基本方法，又讲述面向对象的设计和编程实现，形成逻辑合理的一体化系统。在课程实习作业中，我们只提出必须达到的规范要求，而让学生自由选择C++或Java作为工具。学习本课程不要求先学C语言，而是让学生直接以面向对象分析和设计为起点，能够较快地掌握新的程序设计方法。

在本书编写过程中，许多学者参加了工作，其中谢成林、潘鹏、高嵩(MIT)参与了部分讲义整理和程序调试工作，许多学生对本课程的教学内容和方法的完善提供了宝贵建议，在此表示衷心地感谢！

本教材中不当之处，敬请读者不吝指正。

吴炜煜

1999年10月8日于清华园

目　录

第1章　面向对象技术概论

面向对象技术的概念和方法，本质上是一种合理的思维方法，是不依赖于程序设计语言的应用软件开发的基本核心技术。因此，要深刻了解C++语言和Java语言，掌握面向对象编程，首先应该学习面向对象技术的基本要点。越是深入理解面向对象技术的理论和方法，就越能让您在自己的应用领域中最大限度地发挥思维能力和创造本领，就能高屋建瓴地掌握C++和Java面向对象软件设计。

本章目的：

- 了解客观世界的认识抽象与计算机实现的一致性原则
- 理解面向对象技术的基本概念
- 了解面向对象方法的特点和优点
- 初步了解面向对象分析模型及视图
- 了解面向对象设计(OOD)的工作要点
- 对C++的背景和支持面向对象方法的内容有所了解

1.1　引　　论

在人类文明的发展史上，20世纪后半叶，我们地球人写下了多么光辉灿烂的一页，这就是计算机文化的诞生和发展，它比文艺复兴还要更广泛更深入地改变了人们的生活。

计算模型研究取得突破性进展是在20世纪30年代，以阿兰·图灵(A. M. Turing)等人为代表的关于可计算性研究及图灵机概念，深刻地揭示了现代通用数字计算机最核心的内容。图灵机可以动态地描述计算机的整个过程，而且其简洁的构造和运行原理易于为人们所理解。在其后不到10年，即1946年，冯·诺依曼(Von. Neumann)等创造的世界第一台计算机诞生了，现代存储程序式电子数字计算机的基本结构和工作原理也基本确定，随后是在此基础上的不断完善、丰富和提高。现在，人们已在研究新的计算机模型，无止境的创造永远是人类不断追求完美的阶梯。

软件程序是计算机的灵魂。而不带软件的数字电子计算机系统，人们习惯上称其为硬件裸机。软件是相对于机器硬件而言的，是事先编制好的具有特定功能和用途的程序系统及其说明文件的统称。一个计算机系统是由计算机硬件及相应软件一起构成的。

当代软件工程的发展正面临着从传统的结构化范型到面向对象范型的转移，这需要有新的语言、新的系统和新的方法学的支持，面向对象技术就是这种新范型的核心技术。

1.1.1　软件概念的发展

计算机科学发展的每一步几乎都在软件设计和程序设计语言中得到体现。早在美国人进行计算分析机研制试验时，诗人拜伦的独生女儿Ada为之编写了“程序”，而第一个高级

程序设计语言则是在20世纪50年代中期研制的FORTRAN，随后出现的是ALGOL、COBOL。在60年代至少研制了200多个高级语言，其中包括APL、SIMULA67语言等。70年代主要集中于发展命令性语言，最著名的有Pascal、Ada语言。80年代是突出发展作用性语言的时代，如LISP、PROLOG、FFP(函数式程序设计系统)等，推出了典型的面向对象程序设计语言Smalltalk80。现在有影响的语言已超过上百种，这些易于为人们理解、掌握与使用的各种程序设计语言，其表示法、约定与规则的集合以及众多的软件，形成的计算机文化，已超越了计算机的范围，渗透到人们的生活和工作的方方面面。

软件是一个发展的概念，随着软件开发规模的扩大和开发方式的变化，程序设计开始被人们作为一门科学来对待，它研究程序设计和实现的各种性质、规律。经过多年研究，在计算科学中发展了许多程序设计方法和技术，例如，自顶向下和自底向上的程序设计方法、程序推导设计方法、程序变换设计方法、函数式程序设计技术(FP，使用面向问题的说明性语言)、逻辑程序设计技术(LP，把计算过程视为推演过程，使用一定条件下的初始态，经过推理算法和搜索手段进行匹配演算)、面向对象的程序设计技术、程序验证技术、约束程序设计技术、并发程序设计技术，等等。程序设计方法和技术在各个时期的发展不仅直接导致了一大批风格各异的高级语言的诞生，而且对计算机理论、硬件、软件以及计算机应用技术等多方面具有深刻的影响。

1.1.2 软件开发原理的变革

软件工程技术的发展，其目的是提高计算机性能和应用范围，其关键是提高软件质量和生产效率。从汇编语言到高级语言，标志着软件工程技术和软件生产率的一次质的飞跃，促成这次飞跃的技术因素是编译理论和实现方法的完善，使我们实现了从高级源码到机器代码的自动转换。随着应用需求的扩大和变化，软件生产的方式和效率仍然远远跟不上社会发展的需要。

从软件开发原理上看，影响较大的变革有3个：

(1) 20世纪60年代开发的规范化设计，代表性的是瀑布方法：使软件程序设计由个人经验、智慧、技巧、特别定制，逐渐被系统方法所代替，使建立软件系统的过程遵从一系列规范化阶段，包括需求分析、高层设计、详细设计等。这也使人们开始将软件设计工作推进到软件工程时代。

(2) 20世纪70年代末开始的结构化系统分析和程序设计是与冯·诺依曼计算机系统的结构特点一致的，虽然不能直接反映出人类认识问题的过程，但所推广的模块化设计方法是很大的进步。Tom. De Marcod的《结构化分析与系统规格说明》教材提出基于模型的软件工程概念，认为复杂软件系统的创建，首先必须建立系统的书面工作模型。另一个有影响的软件理论是Niklans Wirth提出的“算法＋数据结构＝程序设计”。软件被划分成若干可单独命名和编址的部分，它们被称做模块，模块化使软件能够有效地管理和维护，能够有效地分解和处理复杂问题。在80年代，模块化程序设计方法普遍被人们接受。接下来人们便开始争论模块应当如何建立。有人认为最佳途径是用函数，有人认为每个模块应容纳一个数据结构，有人认为每个模块应做且仅做一件事，有人则认为应以事件来驱动，上述几种观点是当时有代表性的解决方案。

(3) 在20世纪80年代，在软件开发中各种概念和方法积累的基础上，对于如何超越程

序复杂性障碍，如何在计算机系统中自然地表示客观世界，人们拿起了思维科学中面向对象技术作为武器，采取基于客观世界的对象模型的软件开发方法，按问题论域(problem domain)设计程序模块，它不是以函数过程、数据结构为中心，而是以对象代表问题解的中心环节，它使计算机程序的分析、设计和实现的过程和方法，改变了过去的脱节和跳跃状态，使人们对复杂系统的认识过程与系统的程序设计实现过程，尽可能地一致。经验证明，对任何软件系统而言，其中最稳定的成分是对应的问题论域。与功能相比，一个问题论域中的对象一般总能保持相对稳定性，因而以面向对象构造的软件系统主体结构也具有较好的稳定性和可重用性。因此，采用"对象＋消息"的程序设计模式，具有满足软件工程发展需要的更多优势。

面向对象设计方法追求的是现实问题空间与软件系统解空间的近似和直接模拟。从哲学上讲，现实世界空间中的基本问题是物质和意识，映射到面向对象系统解空间就是具体事物(对象)和抽象概念(类)。面向对象技术的封装、继承、多态性等不仅支持软件复用，而且使软件维护工作可靠有效，可实现软件系统的柔性制造。特别是随着 Internet/Intranet 的发展，网络分布计算的应用需求日益增长，面向对象技术为网络分布计算提供了基础性核心技术支持。

1.1.3 面向对象语言的三个里程碑

创始的面向对象语言，现在公认是 20 世纪 60 年代的 Simula 67 语言。虽然它是一种通用的仿真建模语言，但所使用的对象概念和方法，给人们启示了软件设计新思维。它的对象代表仿真中的一个实体(例如，一座楼房、一个工号、一项工程)，在仿真过程中，对象之间可以某种方式进行通信。它使用类(class)的概念，用它作为单元(unit)描述相似的一组对象的结构和行为，并支持类的层次组织和继承，允许共享结构和行为。故有人把类看做数据抽象之父。类是面向对象程序设计技术和语言的一个主要特征与设施。

面向对象程序语言发展的主要里程碑是 Smalltalk 语言，它完整地体现并进一步丰富了面向对象的概念。1980 年美国 Xerox Palo Alto 研究中心推出了 Smalltalk 后，相继开发了配套工具环境，使之走向实用，其缺点是人们需要从头学习一门全新的语言。在 20 世纪 80 年代中期，面向对象语言已形成几大类别：一类是纯面向对象的语言，如 Smalltalk 和 Eiffel；另一类是混合型的面向对象语言，如 C＋＋和 Objective C；还有一类是与人工智能语言结合形成的，如 LOOPS、Flavors 和 CLOS；适合网络应用的有 Java 语言。

对流行的语言进行面向对象的扩充，曾经向社会推出过许多种类的版本，而主要的成功代表是 C＋＋，它是一个混合型语言，既支持传统的面向过程程序设计方法，又支持面向对象的程序设计方法，有广泛的应用基础和丰富的开发环境支持，因而使面向对象程序设计能够得到很快的普及。Java 语言是 Sun 公司 1995 年推出的一个适用于分布网络环境的面向对象语言，它采用了与 C＋＋语法基本一致的形式，并将 C＋＋中与面向对象无关的部分去掉，其语义是纯面向对象的。Java 是使应用程序独立于异构网络上的多种平台，能解释或编译执行、连接简单、支持语言级的多线程。总之，Java 语言环境使应用变得可移植、高安全性和高性能。

应当指出的是，面向对象语言对程序设计的主要影响并不在于它的语法特征，而在于它所提供的自然问题求解的机制和结构。面向对象编程(OOP)将计算过程看做是分类过程加

状态变换过程，即将系统逐步划分为相互联系的多个对象并建立这些对象的联系，以引发状态转换，实现系统计算任务。因此，要理解面向对象语言，应首先理解面向对象技术的基本原理和基本思想，然后再学习此类语言。实际上，如果我们能够深刻理解面向对象技术的原理和方法，即使不用面向对象的语言或系统，也能实现OOP。面向对象语言所起的作用就是给用户提供一些支持面向对象程序设计的环境和管理工具，特别重要的是提供了对象概念和特性。

1.2 面向对象的基本概念

为了帮助读者理解面向对象的设计思想和方法，本节着重介绍面向对象的基本概念。在此同时，对相应的面向对象的技术方法也顺便做些说明和解释。

1.2.1 对象、类、消息

面向对象技术是基于对象(object)概念的。下面，我们首先从3个层次上，即社会语言、思维科学和面向对象技术，介绍对象概念。

在现代汉语词典中，对象是行动或思考时作为目标的人或事物。在Merrian Webster's Collegiate Dictionary中，object的解释是Something mental or physical toward which thought, beeling or action is directed。对象一词的使用本来是源远流长的，从亚里士多德到笛卡尔的哲学论著，都多次使用了"对象"一词，表示以客观对象为中心来看待哲学问题。

在思维科学中，对象是客观世界中具有可区分性的、能够唯一标识的逻辑单元。对象所代表的本体可能是一个物理存在，也可能是一个概念存在，例如，一粒米、一个人、一所学校、一个工程、一架飞机，等等。

"面向对象"这个术语的使用是计算机科学发展的需要，是一个技术名词，具有其特定的技术含义。从面向对象的观点来看，现实世界是由各式各样独立的、异步的、并发的实体对象所组成，每个对象都有各自的内部状态和运动规律，不同对象之间或某类对象之间的相互联系和作用，就构成了各式不同的系统。

面向对象方法是基于客观世界的对象模型化的软件开发方法。在面向对象程序设计中，所谓对象，是一个属性(数据)集及其操作(行为)的封装体。作为计算机模拟真实世界的抽象，一个对象就是一个实际问题论域、一个物理的实体或逻辑的实体。在计算机程序中，可视为一个"基本程序模块"，因为它包含了数据结构和所提供的相关操作功能。

对象的属性是指描述对象的数据，可以是系统或用户定义的数据类型，也可以是一个抽象的数据类型。对象属性值的集合称为对象的状态(state)。

对象的行为是定义在对象属性上的一组操作方法(method)的集合。方法是响应消息而完成的算法，表示对象内部实现的细节，对象的方法集合体现了对象的行为能力。

对象的属性和行为是对象定义的组成要素，有人把它们统称为对象的特性。无论对象是有形的或是抽象的、简单的或是复杂的，一般具有以下特征：

(1) 具有一个状态，由与其相关联的属性值集合所表征。

(2) 具有唯一标识名，可以区别于其他对象。

(3) 有一组操作方法，每个操作决定对象的一种行为。

(4) 对象的状态只能被自身的行为所改变。

(5) 对象的操作包括自操作(施于自身)和它操作(施于其他对象)。

(6) 对象之间以消息传递的方式进行通信。

(7) 一个对象的成员仍可以是一个对象。

在上述列举的特征中,进一步严格地讲,前3条是对象的基本特征,后4条是属于特性的进一步定义说明。

在面向对象系统中,人们并不是逐个描述各个具体的对象,而是将注意力集中于具有相同特性的一类对象,抽象出这样一类对象的共同的结构和行为,进行一般描述,从而避免数据的冗余。"物以类聚",分类、类比、类型、同类,等等,是人们归纳客观事物的方法。下面我们介绍类的概念和作用。

类(class)是对象的抽象及描述,是具有共同属性和操作的多个对象的相似特性的统一描述体。类也是对象,是一种集合对象,我们称之为对象类(object class),简称为类,以有别于基本的实例对象(object instance)。

在类的描述中,每个类要有一个名字,要表示一组对象的共同特征,还必须给出一个生成对象实例的具体方法。类中的每个对象都是该类的对象实例,也就是说,系统运行时通过类定义属性初始化可以生成该类的对象实例。实例对象是自描述数据结构,每个对象都保存其自己的内部状态,一个类的各个实例对象都能理解该所属类发来的消息。

对象类与对象实例的关系是如此地密切,然而又有区别,如"白马非马"之论。一匹白马是马的一个实例,一匹白马是一个有血有肉的动物,而马是一个抽象概念。若进一步抽象,我们看到,名为"白云"的和名为"白雪"的白马是白马类的实例对象,从这个意义上讲,"白马"成了类。

类提供了完整的解决特定问题的能力,因为类描述了数据结构(对象属性)、算法(方法)和外部接口(消息协议)。例如,在图形处理过程中,一般需要程序控制的画笔,如钢笔类等,要描述画笔的共性结构和信息:其属性数据包括墨水颜色、笔头粗细(线型)、起始位置等;其操作方法包括同一消息接口的多态性,对不同笔执行不同的算法,不同方向移动笔、不同图素的起、落笔及其消息响应描述等。

在一个有效率的面向对象系统中,是没有完全孤立的对象的,对象的相互作用的模式是采用消息传送来进行的。

消息(message)是面向对象系统中实现对象间的通信和请求任务的操作。消息传递是系统构成的基本元素,是程序运行的基本处理活动。一个对象所能接受的消息及其所带的参数,构成该对象的外部接口。对象接受它能识别的消息,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,也可以接受多个对象发来的消息。消息只反映发送者的请求,由于消息的识别、解释取决于接受者,因而同样的消息在不同对象中可解释成相应的行为。

对象间传送的消息一般由三部分组成,即接受对象名、调用操作名和必要的参数。在C++中,一个对象的可能消息集是在对象的类描述中说明,每个消息在类描述中由一个相应的方法给出,即使用函数定义操作。对象的相互作用,用一种类似于客户/服务器的机制把消息发送到给定对象上。换句话说,向对象发送一个消息,就是引用一个方法的过程。实施对象的各种操作,就是访问一个或多个在类对象中定义的方法。

消息协议是一个对象对外提供服务的规定格式说明，外界对象能够并且只能向该对象发送协议中所提供的消息，请求该对象服务。也就是说，请求对象操作的唯一途径是通过协议中提供的消息进行。具体的实现上，是将消息分为公有消息和私有消息，而协议则是一个对象所能接受的所有公有消息的集合。

1.2.2 封装性、继承性和多态性

在上述面向对象的基本概念的基础上，下面将就所有面向对象程序设计都具有的3个共同的特性，进行分析说明，使我们对面向对象的概念和原理能够有进一步的认识和理解。

封装性(encapsulation)是面向对象具有的一个基本特性，其目的是有效地实现信息隐藏原则。这是软件设计模块化、软件复用和软件维护的一个基础。

封装是一种机制，它将某些代码和数据链接起来，形成一个自包含的黑盒子(即产生一个对象)。一般地讲，封装的定义为：

(1) 一个清楚的边界，封装的基本单位是对象；

(2) 一个接口，这个接口描述该对象与其他对象之间的相互作用；

(3) 受保护的内部实现，提供对象的相应的软件功能细节，且实现细节不能在定义该对象的类之外。

面向对象概念的重要意义在于，它提供了较为令人满意的软件构造的封装和组织方法：以类/对象为中心，既满足用户要求的模块原则和标准，又满足代码复用要求。客观世界的问题论域及其具体成分，在面向对象系统中，最终只表现为一系列的类/对象。

对象的组成成员中含有私有部分、保护部分和公有部分，公有部分为私有部分提供了一个可以控制的接口。这就是说，在强调对象的封装性时，也必须允许对象有不同程度的可见性。可见性是指对象的属性和服务允许对象外部存取和引用的程度。

面向对象程序设计技术鼓励人们把一个问题论域分解成几个相互关联的子问题，每个子问题(子类)都是一个自包含对象。一个子类(subclass)可以继承父类的属性和方法，还可以拥有自己的属性和方法，子类也能将其特性传递给自己的下级子类，这种对象的封装、分类层次和继承概念，同人们在对真实世界认识的抽象思维中，运用聚合和概括，是自然地映射、融洽地结合。所以，有些人称面向对象程序设计是使计算机走向“自然主义”，也不是没有根据的。

继承性(inheritance)是面向对象技术中的另一个重要概念和特性，它体现了现实世界中对象之间的独特关系。既然类是对具体对象的抽象，那么就可以有不同级别的抽象，就会形成类的层次关系。若用结点表示类对象，用连接两结点的无向边表示其概括关系，就可用树形图表示类对象的层次关系，其中高层结点称为其下层结点的父类，下层结点称为高层结点的子类。继承关系可分为以下几种：一代或多代继承、单继承(single inheritance)和多继承(multiple inheritance)。子类仅对单个直接父类的继承叫做单继承。子类对多于一个的直接父类的继承叫做多继承。单继承的类层次是树形结构，如图1.1(a)所示。多继承的类层次由树结构扩变为类格(class lattice)，如图1.1(b)所示。图1.2所表达的类与继承的实例说明了上述概念的应用。

就继承风格而言，具有多样性，除上述的单继承、多继承、多代继承外，还有全部继承、部分继承，等等。一般的面向对象系统，在不同程度上支持如下四种类型的继承：