第 2 章　面向对象技术和建模基础

学习目标

本章将向读者详细介绍面向对象方法的基本知识和软件建模的概述。主要内容包括：面向对象的基本概念、面向对象分析、面向对象设计、面向对象编程、软件建模的概念和软件建模的优点等。本章的学习要点包括：

- 面向对象分析；
- 面向对象设计；
- 面向对象编程；
- 软件建模的概念。

学习导航

面向对象建模技术是面向对象技术在软件系统建模中的重要应用，UML 建模的基础是面向对象思想。本章学习导航如图 2-1 所示。

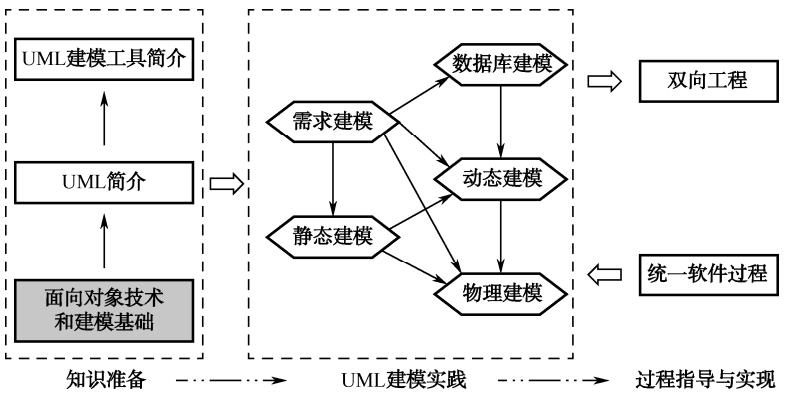

图 2-1　本章学习导航

2.1　面向对象方法

任务 1　了解面向对象软件工程的基本思想和 OOA、OOD 和 OOP 的基本内容。

2.1.1　面向对象方法的基本思想

“对象（Object）”一词，在 19 世纪就由现象学大师胡塞尔提出并定义。胡塞尔认为对象是世界中的物体在人脑中的映象，是人的意识之所以为意识的反映，是作为一种概念而存在的意念的东西，它还包括了人的意愿。例如，当我们认识到一种新的物体，它叫树，于是在我们的意识当

中就形成了树的概念。这个概念会一直存在于我们的思维当中，并不会因为这棵树被砍掉而消失，这个概念就是现实世界当中的物体在我们意识当中的映象。我们对树还可以有我们自己的意愿，我们可以想象着把这棵树砍掉做成桌子、凳子等。所以说，对象就是客观世界中物体在人脑中的映象及人的意向。只要这个对象存在于我们的思维意识当中，我们就可以借此判断同类的东西。例如，当我们看到另外一棵树的时候，并不会因为所见的第一棵树不在了（失去了参照的模板）而不认识其他的树了。

IT 领域中的“面向对象技术”，一般指的是解决信息领域内所遇到问题的方法，特别是应用软件技术来解决问题的方法。如我们经常碰到的面向对象的分析（Object-Oriented Analysis）、面向对象的设计（Object-Oriented Design）和面向对象的编程（Object-Oriented Programming）等。

面向对象方法（Object-Oriented Method）是一种把面向对象的思想应用于软件开发过程中，指导开发活动的系统方法，简称 OO（Object-Oriented）方法。面向对象方法是建立在“对象”概念基础上的方法学。对象是由数据和允许在数据上执行的操作组成的封装体，与客观实体有直接对应关系，一个类定义了具有相似性质的一组对象。而继承性是对具有层次关系的类的属性和操作进行共享的一种方式。所谓面向对象就是基于对象概念，以对象为中心，以类和继承为构造机制，来认识、理解、刻画客观世界和设计、构建相应的软件系统。

面向对象方法作为一种新型的独具优越性的新方法正引起全世界越来越广泛的关注和高度的重视，更是当前计算机界关心的重点。面向对象方法会强烈地影响、推动和促进一系列高技术的发展和多学科的综合。

2.1.2　面向对象方法的发展

面向对象方法起源于面向对象的编程语言。20 世纪 50 年代后期，在用 FORTRAN 语言编写大型程序时，常出现变量名在程序不同部分发生冲突的问题。因此，ALGOL 语言的设计者在 ALGOL60 中采用了以“Begin…End”为标识的程序块，使块内变量名是局部的，以避免它们与程序中块外的同名变量相冲突。这是编程语言中首次提供封装（保护）的尝试。此后程序块结构广泛用于其他高级语言如 PASCAL、ADA 和 C 中。

20 世纪 60 年代中后期，在 ALGOL 语言基础上研制开发了 Simula 语言，Simula 语言将 ALGOL 语言的块结构概念向前发展一步，提出了对象的概念，并使用了类，也支持类继承。20 世纪 70 年代，Smalltalk 语言诞生，它取 Simula 的类为核心概念。Xerox 公司经过对 Smalltalk72、Smalltalk76 持续不断的研究和改进之后，于 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机制。

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、信息封装与隐蔽、抽象性、继承性、多样性等独特之处，这些优异特性为研制大型软件、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和途径。

自 20 世纪 80 年代以来，人们将面向对象的基本概念和运行机制运用到其他领域，获得了一系列相应领域的面向对象的技术。面向对象方法已被广泛应用于程序设计语言、形式定义、设计方法学、操作系统、分布式系统、人工智能、实时系统、数据库、人机接口、计算机体系结构以及并发工程、综合集成工程等，在许多领域的应用都得到了很大的发展。1986 年在美国举行了首届“面向对象编程、系统、语言和应用（OOPSLA'86）”国际会议，使面向对象受到世人瞩目，其后每年都举行一次，进一步标志面向对象方法的研究已普及到全世界。

2.2 面向对象的基本概念与特征

使用计算机解决问题时需要利用程序设计语言对问题求解加以描述（编程），而软件是问题求解的一种表述形式。显然，假如软件能直接表现人求解问题的思维路径（求解问题的方法），那么软件不仅容易被人理解，而且易于维护和修改，从而会保证软件的可靠性和可维护性，并能提高公共问题域中的软件模块和模块重用的可靠性。面向对象的概念和机制可以使人们按照常规的思维方式来建立问题域的模型，设计出尽可能自然地表现求解方法的软件。

2.2.1 面向对象的基本概念

1. 对象

对象是要研究的任何事物。一本书、一个人、一件商品、一家图书馆、一家极其复杂的自动化工厂、一架航天飞机都可看作对象，它不仅能表示有形的实体，也能表示无形的（抽象的）规则、计划或事件。对象由数据（描述事物的属性）和作用于数据的操作（体现事物的行为）构成一个独立整体。从程序设计者来看，对象是一个程序模块；从用户来看，对象为他们提供所希望的行为。

2. 类

类是对象的模板，即类是对一组有相同数据和相同操作的对象的定义，一个类所包含的方法和数据描述一组对象的共同属性和行为。类是在对象之上的抽象，对象则是类的具体化，是类的实例。类可有其子类，形成类层次结构。

3. 消息

消息是对象之间进行通信的一种规格说明。它一般由三部分组成：接收消息的对象、消息名及实际变元。

2.2.2 面向对象的主要特征

1. 封装性

封装是一种信息隐蔽技术，它体现于类的说明，是对象的重要特性。封装使数据和加工该数据的方法（函数）封装为一个整体，以实现独立性很强的模块，使得用户只能见到对象的外特性（对象能接收哪些消息，具有哪些处理能力），而对象的内特性（保存内部状态的私有数据和实现加工能力的算法）对用户是隐蔽的。封装的目的在于把对象的设计者和对象的使用者分开，使用者不需要知道行为实现的细节，只需通过设计者提供的消息来访问该对象。

2. 继承性

继承性是子类自动共享父类数据和方法的机制，它由类的派生功能体现。一个类直接继承其他类的全部描述，同时可修改和扩充。

继承具有传递性，继承分为单继承（一个子类只有一个父类）和多重继承（一个类有多个父类）。类的对象是各自封闭的，如果没有继承性机制，则类对象中数据、方法就会出现大量重复。继承不仅支持系统的可重用性，而且还促进系统的可扩充性。

3. 多态性

对象根据所接收的消息会产生行动，同一消息为不同的对象接收时可产生完全不同的行动，这种现象称为多态性。利用多态性用户可发送一个通用的信息，而将所有的实现细节都留给接收消息的对象自行决定。例如，Print 消息被发送给图表时调用的打印方法与将同样的 Print 消息发送给正文文件而调用的打印方法会完全不同。多态性的实现受到继承性的支持，利用类继承的层次关系，把具有通用功能的协议存放在类层次中尽可能高的地方，而将实现这一功能的不同方法置于较低层次，这样，在这些低层次上生成的对象就能给通用消息以不同的响应。在面向对象编程语言中可通过在派生类中重定义基类函数（定义为重载函数或虚函数）来实现多态性。

综上所述，在面向对象方法中，对象和消息传递分别表现事物及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类的继承性。通过封装能将对象的定义和对象的实现分开，通过继承能体现类与类之间的关系，以及由此带来的动态联编和实体的多态性，从而构成了面向对象的基本特征。

4. 面向对象方法的优越性

面向对象方法用于系统开发有如下优越性。

（1）强调从现实世界中客观存在的事物（对象）出发来认识问题域和构造系统，这就大大降低了系统开发者对问题域的理解难度，从而使系统能更准确地反映问题域。

（2）运用人类日常的思维方法和原则（体现于面向对象方法的抽象、分类、继承、封装、消息通信等基本原则）进行系统开发，有益于发挥人类的思维能力，并有效地控制了系统复杂性。

（3）对象的概念贯穿于开发过程的始终，使各个开发阶段的系统成分有良好的对应，从而显著地提高了系统的开发效率与质量，并大大降低系统维护的难度。

（4）对象概念的一致性，使参与系统开发的各类人员在开发的各阶段具有共同语言，有效地改善了人员之间的交流和协作。

（5）对象的相对稳定性和对易变因素隔离，增强了系统的应变能力。

（6）对象类之间的继承关系和对象的相对独立性，对软件复用提供了强有力的支持。

2.3　面向对象分析

当我们遵照面向对象方法学的思想进行软件系统开发时，首先要进行面向对象的分析（Object Oriented Analysis，OOA），其任务是了解问题域所涉及的对象、对象间的关系和作用。然后构造问题的对象模型，力争该模型能真实地反映出所要解决的“实质问题”。在这一过程中，抽象是最本质、最重要的方法。针对不同的问题性质选择不同的抽象层次，过简或过繁都会影响到对问题本质属性的了解和解决。

面向对象的分析方法是在一个系统的开发过程中进行系统业务调查，按照面向对象的思想来分析问题。面向对象分析与结构化分析有较大的区别，面向对象分析所强调的是在系统调查资料的基础上，针对面向对象方法所需要的素材进行归类分析和整理，而不是对管理业务现状和方法的分析。

2.3.1 处理复杂问题的原则

用面向对象分析方法对所调查结果进行分析处理时，一般依据以下几项原则。

1. **抽象**（abstraction）

抽象是指为了某一分析目的而集中精力研究对象的某一性质，它可以忽略其他与此目的无关的部分。在使用这一概念时，我们承认客观世界的复杂性，也知道事物包括多个细节，但此时并不打算去完整地考虑它。抽象是我们科学地研究和处理复杂问题的重要方法。抽象机制被用在数据分析方面，称为数据抽象。数据抽象是 OOA 的核心。数据抽象把一组数据对象以及作用在其上的操作组成一个程序实体。使得外部只知道它是如何做和如何表示的。在应用数据抽象原理时，系统分析人员必须确定对象的属性以及处理这些属性的方法，并借助于方法获得属性。在 OOA 中属性和方法被认为是不可分割的整体。抽象机制有时也被用在对过程的分解方面，称为过程抽象。恰当的过程抽象可以对复杂过程的分解和确定，以及描述对象发挥积极的作用。

2. **封装**（encapsulation）

封装即信息隐蔽，是指在确定系统的某一部分内容时，应考虑到其他部分的信息及联系都在这一部分的内部进行，外部各部分之间的信息联系应尽可能少。

3. **继承**（inheritance）

继承是指能直接获得已有的性质和特征而不必重复定义它们。OOA 可以一次性地指定对象的公共属性和方法，然后再特化和扩展这些属性及方法为特殊情况，这样可大大地减轻在系统实现过程中的重复劳动。在共有属性的基础之上，继承者也可以定义自己独有的特性。

4. **相关**（association）

相关是指把某一时刻或相同环境下发生的事物联系在一起。

5. **消息通信**（communication with message）

消息通信是指在对象之间互相传递信息的通信方式。

6. **组织方法**（method of organization）

在分析和认识世界时，可综合采用如下三种组织方法：

- 特定对象与其属性之间的区别；
- 整体对象与相应组成部分对象之间的区别；
- 不同对象类的构成及其区别。

7. **比例**（scale）

比例是一种运用整体与部分原则，辅助处理复杂问题的方法。

8. **行为范畴**（categories of behavior）

行为范畴是针对被分析对象而言的，它们主要包括：

- 基于直接原因的行为；
- 时变性行为；
- 功能查询性行为。

2.3.2　OOA 方法的基本步骤

在用 OOA 具体地分析一个事物时，大致遵循如下五个基本步骤。

1. 确定对象和类

这里所说的对象是对数据及其处理方式的抽象，它反映了系统保存和处理现实世界中某些事物信息的能力。类是多个对象的共同属性和方法集合的描述，它包括如何在一个类中建立一个新对象的描述。

2. 确定结构

结构是指问题域的复杂性和连接关系。类成员结构反映了泛化-特化关系，整体-部分结构反映整体和局部之间的关系。

3. 确定主题

主题是指事物的总体概貌和总体分析模型。

4. 确定属性

属性就是数据元素，可用来描述对象或分类结构的实例，可在图中给出，并在对象的存储中指定。

5. 确定方法

方法是在收到消息后必须进行的一些处理操作。对于每个对象和结构来说，用来增加、修改、删除和选择一个方法本身都是隐含的，而有些则是显示的。

2.4　面向对象设计

使用面向对象方法的第二步就是进行面向对象的设计（Object Oriented Analysis，OOD），即设计软件的对象模型。根据所应用的面向对象软件开发环境的功能强弱不等，在对问题对象模型分析的基础上，可能要对它进行一定的改造，但应以最少改变原问题域的对象模型为原则。然后就在软件系统内设计各个对象、对象间的关系（如层次关系、继承关系等）、对象间的通信方式（如消息模式）等。

面向对象的设计方法是面向对象方法中一个中间过渡环节，其主要作用是对面向对象分析的结果作进一步的规范化整理，以便能够被面向对象编程直接接受。在 OOD 的设计过程中，要开展的主要工作如下。

1. 对象定义规格的求精

对于 OOA 所抽象出来的对象和类以及汇集的分析文档，OOD 需要有一个根据设计要求整理和求精的过程，使之更能符合 OOP 的需要。这个整理和求精过程主要有两个方面：一是要根据面向对象的概念模型整理分析所确定的对象结构、属性、方法等内容，改正错误的内容，删去不必要和重复的内容等；二是进行分类整理，以便于下一步数据库设计和程序处理模块设计的需要。整理的方法主要是进行归类，即对类、对象、属性、方法、结构和主题进行归类。

2. 数据模型和数据库设计

数据模型的设计需要确定类和对象属性的内容、消息连接的方式、系统访问、数据模型的方法等。最后每个对象实例的数据都必须落实到面向对象的库结构模型中。

3. 优化设计

OOD 的优化设计过程是从另一个角度对分析结果和处理业务过程的整理归纳，优化包括对象和结构的优化、抽象、集成。对象和结构的模块化表示 OOD 提供了一种范式，这种范式支持对类和结构的模块化。这种模块符合一般模块化所要求的所有特点，如信息隐蔽性好，内部聚合度强和模块之间耦合度弱等。集成化使得单个构件有机地结合在一起，相互支持。

2.5 面向对象实现

最后阶段是面向对象的实现（Object Oriented Implementation，OOI），即指软件功能的编码实现，主要工作为面向对象的编程（Object Oriented Programming，OOP）。它包括：每个对象的内部功能的实现，确立对象哪一些处理能力应在哪些类中进行描述，确定并实现系统的界面、输出的形式及其他控制机理等，总之是实现在 OOD 阶段所规定的各个对象所应完成的任务。

面向对象编程的基本步骤如下。

（1）分析确定在问题空间和解空间出现的全部对象及其属性。

（2）确定应施加于每个对象的操作，即对象固有的处理能力。

（3）分析对象间的联系，确定对象彼此间传递的消息。

（4）设计对象的消息模式，消息模式和处理能力共同构成对象的外部特性。

（5）分析各个对象的外部特性，将具有相同外部特性的对象归为一类，从而确定所需要的类。

（6）确定类间的继承关系，将各对象的公共性质放在较上层的类中描述，通过继承来共享对公共性质的描述。

（7）设计每个类关于对象外部特性的描述。

（8）设计每个类的内部实现（数据结构和方法）。

（9）创建所需的对象（类的实例），实现对象间应有的联系（发消息）。

使用 OOP 的优点是使人们的编程与实际的世界更加接近，所有的对象被赋予属性和方法，结果编程就更加富有人性化。但 OOP 也存在缺点：由于面向更高的逻辑抽象层，在实现的时候，不得不做出性能上的牺牲。

2.6 面向对象方法的内涵

面向对象方法的作用和意义绝不只局限于编程技术，它是一种新的程序设计范型（面向对象程序设计范型），是信息系统开发的新方法论（面向对象方法学），是正在兴起的新技术（面向对象技术）。

1. 面向对象程序设计范型

程序设计范型（以下简称“程设范型”）具体指的是程序设计的体裁，正如文学上有小说、诗歌、散文等体裁，程序设计体裁是用程序设计语言表达各种概念和各种结构的一套设施。目前，程设范型分为：过程式程设范型、函数式程设范型，此外还有进程式程设范型、事件程设范型和

类型系统程设范型。每一程设范型都有多种程序设计语言支持（如 FORTRAN、PASCAL、C 均体现过程式程设范型，用来进行面向过程的程序设计），而某些语言兼备多种范型（如 Lisp 属过程与函数混合范型，C++则是进程与面向对象混合范型的语言）。

过程式程设范型是流行最广泛的程序设计范型（人们平常所使用的程序设计语言大多属于此类型），这一程设范型的中心点是设计过程，所以程序设计时首先要决定的是问题解所需要的过程，然后设计过程的算法。这类范型的语言必须提供设施给过程（函数）传送变元和返回的值，如何区分不同种类的过程（函数）、如何传送变元是这类程序设计中关心的主要问题。

面向对象程设范型是在以上范型上发展起来的，它的关键在于加入了类及其继承性，用类表示通用特性，子类继承父类的特性，并可加入新的特性。对象以类为样板被创建。所以在面向对象程设范型中，首要的任务是决定所需要的类，每个类应设置足够的操作，并利用继承机制共享共同的特性。

简而言之，面向对象程设范型具有其他范型所缺乏或不具备的特点，极富生命力，能够适应复杂的大型的软件开发。可以肯定地说，这种新的程设范型必将有力地推动软件开发的新进展。

2. 面向对象方法学

面向对象方法遵循一般的认知方法学的基本概念（有关演绎—从一般到特殊和归纳—从特殊到一般的完整理论和方法体系），并以面向对象方法为基础。

面向对象方法学要点之一：认为客观世界是由各种“对象”所组成的，任何事物都是对象，每一个对象都有自己的运动规律和内部状态，每一个对象都属于某个对象“类”，都是该对象类的一个元素。复杂的对象可以是由相对比较简单的各种对象以某种方式而构成的。不同对象的组合及相互作用就构成了我们要研究、分析和构造的客观系统。

面向对象方法学要点之二：通过类比，发现对象间的相似性，即对象间的共同属性，这就是构成对象类的依据。在“类”、“父类”、“子类”的概念构成对象类的层次关系时，若不加特殊说明，则处在下一层次上的对象可自然地继承位于上一层次上的对象的属性。

面向对象方法学要点之三：认为对已分成类的各个对象，可以通过定义一组“方法”来说明该对象的功能，即允许作用于该对象上的各种操作。对象间的相互联系是通过传递“消息”来完成的，消息就是通知对象去完成一个允许作用于该对象的操作，至于该对象将如何完成这个操作的细节，则是封装在相应的对象类的定义中的，细节对于外界是隐蔽的。

3. 面向对象技术

技术“泛指根据生产实践经验和自然科学原理而发展起来的各种工艺操作方法与技能”；“广义地讲，还包括相应的生产工具和其他物质设备，以及生产的工艺过程或作业程序、方法”。面向对象方法既是程序设计新范型、系统开发的新方法学，作为一门新技术它就有了基本的依据。事实上，面向对象方法可支持种类不同的系统开发，已经或正在许多方面得以应用，因此，可以说面向对象方法是一门新的技术——面向对象技术。

近十多年来，除了面向对象的程序设计以外，面向对象方法已发展应用到整个信息系统领域和一些新兴的工业领域，包括：用户界面（特别是图形用户界面——GUI）、应用集成平台、面向对象数据库（OODB）、分布式系统、网络管理结构、人工智能领域及并发工程、综合集成工程等。人工智能是和计算机密切相关的新领域，在很多方面已经采用面向对象技术，如知识的表示，专家系统的建造、用户界面等。人工智能的软件通常规模较大，用面向对象技术有可能更好地设计

并维护这类程序。

20 世纪 80 年代后期形成的并发工程，其概念要点是在产品开发初期（方案设计阶段）就把结构、工艺、加工、装配、测试、使用、市场等问题同期并行地启动运行，其实现必须有两个基本条件：一是专家群体，二是共享并管理产品信息（将 CAD、CAE、CIN 紧密结合在一起）。显然，这需要面向对象技术的支持。目前，一些公司采用并发工程组织产品的开发，已取得显著效益：波音公司开发巨型 777 运输机，比开发 767 节省了一年半时间；日本把并发工程用于新型号的汽车生产，和美国相比只用了一半的时间。产业界认为它们今后的生存要依靠并发工程，而面向对象技术是促进并发工程发展的重要支持。

综合集成工程是开发大型开放式复杂统的新的工程概念，和并发工程相似，专家群体的组织和共享信息，是支持这一新工程概念的两大支柱。由于开放式大系统包含人的智能活动，建立数学模型非常困难，而面向对象方法能够比较自然地刻画现实世界，容易达到问题空间和程序空间的一致，能够在多种层次上支持复杂系统层次模型的建立，是研究综合集成工程的重要工具。

4. 面向对象方法当前的研究领域

当前，在研究面向对象方法的热潮中，有如下主要研究领域。

（1）智能计算机的研究。因为面向对象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越来越受到重视。面向对象方法的设计思想被引入到智能计算机的研究中。

（2）新一代操作系统的研究。采用面向对象方法来组织设计新一代操作系统具有如下优点：采用对象来描述 OS 所需要设计、管理的各类资源信息，如文件、打印机、处理机等各类设备更为自然；引入面向对象方法来处理面向对象的诸多事务，如命名、同步、保护、管理等，会更易实现、更便于维护；面向对象方法对于多机、并发控制可提供有力的支持，并能恰当地管理网络，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、编程语言的研究有汇合趋势。例如，在研究新一代数据库系统（智能数据库系统）中，能否用人工智能思想与面向对象方法建立描述功能更强的数据模型？能否将数据库语言和编程语言融为一体？为了实现多学科的综合，面向对象方法是一个很有希望的汇聚点。

（4）新一代面向对象的硬件系统的研究。要支持采用面向对象方法设计的软件系统的运行，必须建立更理想的能支持面向对象方法的硬件环境。目前采用松耦合（分布主存）结构的多处理机系统更接近于面向对象方法的思想。最新出现的神经网络计算机的体系结构与面向对象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面向对象方法的消息传送有着天然的联系；一次连接可以看做一次消息的发送。可以预料，将面向对象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。

课堂实践 1

1. 操作要求

（1）应用面向对象方法中的概念对 DVD 和播放 DVD 的情景进行描述。

（2）结合软件开发实践，举例说明 OOA、OOD 和 OOP 的具体任务及相关之间的联系。

（3）举例说明面向对象编程和结构化编程的优缺点。

2. 操作提示

（1）以学习小组为单位分组进行讨论，每小组推荐一名成员进行汇报。
（2）通过上网查阅面向对象方法相关资料进行更为详细的了解。
（3）结合自己的编程实践，进一步理解面向对象的基本思想。

2.7　软件建模概述

任务 2　了解软件模型在开发一个软件系统时的重要作用，理解软件建模的优点。

2.7.1　软件建模的概念

1. 什么是模型

模型是什么？模型是对现实存在的实体的抽象和简化，模型提供了系统的蓝图。模型过滤非本质的细节信息，抽象出问题本质，使问题更容易理解。模型是用某种工具对同类或其他工具的表达方式。它是从某一个建模观点出发，抓住事物最重要的方面而简化或忽略其他方面。在工程、建筑等其他需要创造性的领域中都需要使用模型。建立模型的目的是因为在某些用途中模型使用起来比操纵实物更容易和方便。大家在购买房屋时所看到的楼盘的微缩形状就是对应楼盘的模型。

表达模型的工具要求便于使用。建筑模型可以是图纸上所绘的建筑图，也可以是用厚纸板制作的三维模型，还可以用存于计算机中的有限元方程来表示。一个建筑物的结构模型不仅能够展示这个建筑物的外观，还可以用它来进行工程设计和成本核算。

软件系统的模型用建模语言来表达（如 UML）。这里的模型包含语义信息和表示法，可以采取图形和文字等多种不同形式。可视化建模是使用一些图形符号进行建模，可视化建模的作用如下：它可以捕捉用户的业务过程，可以作为一种很好的交流工具，可以管理系统的复杂性，可以定义软件的架构，还可以增加重用性。本文所提的建模都是指可视化建模。

2. 为什么要建模

需要为软件系统建立模型是因为开发一个具有一定规模和复杂性的软件系统和编写一个简单的程序大不一样。其间的差别，借用 G. Booch 的比喻，如同建造一座大厦和搭一个狗窝的差别。大型的、复杂的软件系统的开发是一项工程，必须按工程学的方法组织软件的生产与管理，必须经过分析、设计、实现、测试、维护等一系列的软件生命周期阶段。这是人们从软件危机中获得的最重要的教益。这一认识促进了软件工程学的诞生。编程仍然是重要的，但是更具有决定意义的是系统建模。只有在分析和设计阶段建立了良好的系统模型，才有可能保证工程的正确实施。正是由于这一原因，许多在编程领域首先出现的新方法和新技术，总是很快地被拓展到软件生命周期的分析与设计阶段。

作曲家会将其大脑中的旋律谱成乐曲，建筑师会将其设计的建筑物画成蓝图，这些乐曲、蓝图就是模型（Model），而建构这些模型的过程就称为建模（Modeling）。软件开发如同音乐谱曲及建筑设计，其过程中也必须将需求、分析、设计、实现、布署等各项工作流程的构想与结果予以呈现，这就是软件系统的建模。

现在的软件越来越大，大多数软件的功能都很复杂，使得软件开发只会变得更加复杂和难以

把握。解决这类复杂问题最有效的方法之一就是分层理论，即将复杂问题分为多个问题逐一解决。软件模型就是对复杂问题进行分层，从而更好地解决问题。这就是为什么要对软件进行建模的原因。有效的软件模型有利于分工与专业化生产，从而节省生产成本。软件建模也是为了降低软件的复杂程度，便于提早看到软件的将来，便于设计人员和开发人员交流使用了建模技术。对于软件人员来说，模型就好像是工程人员的图纸一样重要。只是目前来看，软件模型在软件工程中的重要性还远远没有达到图纸在其他工程中的地位。

在软件系统建模中，抽象是一种处理复杂问题的常用方法。为了建立复杂的软件系统，我们必须抽象出系统的不同视图，使用精确的符号建立模型，验证这些模型是否满足系统的需求，并逐渐添加细节信息把这些模型转变为实现。这样的一个过程就是模型形成的过程，建模是捕捉系统本质的过程，也就是把问题从问题领域转移到解决领域的过程。软件建模是开发优秀软件的一个核心工作，其目的是把要设计的结构和系统的行为联系起来，并对系统的体系结构进行可视化和控制。

3. 建模的必要性

模型是软件开发之根本，无论软件的大小、涉及的范围，还是建模本身，都是系统化认识所开发软件的一个初步的途径。

在现在软件开发的过程中，必须经历的几个过程是需求分析、系统设计、初步实现、系统实现、系统运行、系统维护。在这几个阶段，迭代式的开发模式让我们每个阶段都经历一次系统建模的洗礼，现在 Rational 公司的 RUP 在系统的开发过程中也约束我们性情的自由发展（软件开发必须遵循某中模式，而不是我们在体现高尚的情操）。

原先的系统建模的形式是初步的、不完善的，随着系统实施向前推进，系统模型必须随之改变，但建模没有跟踪过程，但 RUP 提供一个合理的机制——迭代，可以帮助我们解决系统级建模的所有问题。

迭代式是开发过程的描述，实质就是在各个阶段对模型的描述更新，重新认识系统，并把握系统发展趋向，从而有效地控制开发和系统的架构。

当需求分析进行到一个合理化的阶段时，系统模型就出现了，但是目前几乎所有的企业都在“多快好省”地开发系统，这是一个大忌，有时需求必须在一定的阶段才会暴露出来，所以急于求成不是开发系统的方法。

2.7.2 软件建模的用途

模型有多种用途，主要包括以下几个方面。

（1）精确捕获和表达项目的需求和应用领域中的知识，以使各方面的利益相关者能够理解并达成一致。

建筑物的各种模型能够准确表达出这个建筑物在外观、交通、服务设施、抗风和抗震性能，以及消费和其他需求。各方面的利益相关者则包括建筑设计师、建筑工程师、合同缔约人、各个子项目的缔约人、业主、出租者和市政当局。

软件系统的不同模型可以捕获关于这个软件的应用领域、使用方法、度量手段和构造模式等方面的需求信息。各方面的利益相关者包括软件结构设计师、系统分析员、程序员、项目经理、顾客、投资者、最终用户和使用软件的操作员。在 UML 中要使用各种各样的模型。

（2）进行系统设计。

建筑设计师可以用画在图纸上的模型图、存于计算机中的模型或实际的三维模型使自己的设计结果可视化，并用这些模型来做设计方面的试验。建造、修改一个小型模型比较简单，这使得设计人员不需花费什么代价就可以进行创造和革新。

在编写程序代码以前，软件系统的模型可以帮助软件开发人员方便地研究软件的多种构架和设计方案。在进行详细设计以前，一种好的建模语言可以让设计者对软件的构架有全面的认识。

（3）使具体的设计细节与需求分开。

建筑物的某种模型可以展示出符合顾客要求的外观；另一类模型可以说明建筑物内部的电气线路、管线和通风管道的设置情况。实现这些设置有多种方案。最后确定的建筑模型一定是建筑设计师认为最好的一个设计方案。顾客可以对此方案进行检查验证，但通常顾客对具体的设计细节并不关心，只要能满足他们的需要即可。

软件系统的一类模型可以说明这个系统的外部行为和系统中对应于真实世界的有关信息；另一类模型可以展示系统中的类以及实现系统外部行为特性所需要的内部操作。实现这些行为有多种方法。最后的设计结果对应的模型一定是设计者认为最好的一种。

（4）生成有用的实际产品。

建筑模型可以有多种相关产品，包括建筑材料清单、在各种风速下建筑物的偏斜度、建筑结构中各点的应力水平等。

利用软件系统的模型，可以获得类的声明、过程体、用户界面、数据库、合法使用的说明、配置草案以及与其他单位技术竞争情况的对比说明。

（5）组织、查找、过滤、重获、检查以及编辑大型系统的有关信息。

建筑模型用服务设施来组织信息，如建筑结构、电器、管道、通风设施、装潢等。除非利用计算机存储，否则对这些信息的查找和修改没那么容易。相反，如果整个模型和相关信息均存储在计算机中，则这些工作很容易进行，并且可方便地研究多种设计方案，这些设计方案共享一些公共信息。

软件系统用视图来组织信息，如静态结构视图、状态机视图、交互视图、反映需求的视图等。每一种视图均是针对某一目的从模型中挑选的一部分信息的映射。没有模型管理工具的支持不可能使模型做得任意精确。一个交互视图编辑工具可以用不同的格式表示信息，可以针对特定的目的隐藏暂时不需要的信息并在以后再展示出来，可以对操作进行分组，修改模型元素以及只用一个命令修改一组模型元素等。

（6）经济地研究多种设计过程中的解决方案。

对同一建筑的不同设计方案的利弊在一开始可能不很清楚。例如，建筑物可以采用的不同的子结构彼此之间可能有复杂的相互影响，建筑工程师可能无法对这些做出正确的评价。在实际建造建筑物以前，利用模型可以同时研究多种设计方案并进行相应的成本和风险估算。

通过研究一个大型软件系统的模型可以提出多个实际方案并可以对它们进行相互比较。当然模型不可能做得足够精细，但即使一个粗糙的模型也能够说明在最终设计中所要解决的许多问题。利用模型可以研究多种设计方案，所花费的成本只是实现其中一种方案所花费的成本。

（7）利用模型可以全面把握复杂的系统。

一个关于龙卷风袭击建筑物的工程模型中的龙卷风不可能是真实世界里的龙卷风，仅仅是模型而已。真正的龙卷风不可能呼之即来，并且它会摧毁测量工具。许多快速、激烈的物理过程现

在都可以运用这种物理模型来研究和理解。

一个大型软件系统由于其复杂程度可能无法直接研究，但模型使之成为可能。在不损失细节的情况下，模型可以抽象到一定的层次以使人们能够理解。可以利用计算机对模型进行复杂的分析以找出可能的“问题点”，如时间错误和资源竞争等。在对实物做出改动前，通过模型研究系统内各组成部分之间的依赖关系可以得出这种改动可能会带来哪些影响。

2.7.3 软件建模的优点及误区

1. 模型的好处

- 使用模型便于从整体上、宏观上把握问题，可以更好地解决问题；
- 可以加强人员之间的沟通；
- 可以更早地发现问题或疏漏的地方。模型为代码生成提供依据；
- 模型帮助我们按照实际情况对系统进行可视化；
- 模型允许我们详细说明系统的结构或行为；
- 模型给出了一个指导我们构造系统的模板；
- 模型对我们做出的决策进行文档化。

2. 建模的误区

由于软件建模技术的发展时间并不长，中国软件业中实际应用建模技术也是近几年的事情，这样就必然存在对软件建模认识的误区。下面是一些常见的误区。

误区一：建模＝写文档 。很多开发人员认为建模就是写文档从而放弃了软件建模。许多优秀的软件开发人员不想把时间浪费在这些“无用的”文档上，整天沉溺于编码之中，而制造一些脆弱而劣质的系统。实际上“模型”与“文档”在概念上是风马牛不相及的。我们可以拥有一个不是文档的模型和不是模型的文档。

误区二：建模是在浪费时间。很多比较初级的程序员都这样认为，这主要是因为他们所掌握的软件知识仅局限于如何编写代码，对于软件开发没有一个整体的认识。这是编者在工作中经常见到的一种现象，也是推行软件建模技术的障碍之一。

误区三：从开始阶段就形成一个很完美的模型。建模应该是一个不断迭代的过程，一下子形成一个完美的模型想法是好的，但是很难实现。我们对事物认识的过程总是由浅入深，不断完善的。现在提倡的软件过程都是增量式迭代开发，也就是这个原因。

课堂实践 2

1. 操作要求

（1）结合生活中的实例，举例说明模型的重要作用。

（2）结合软件开发实践，举例说明软件建模的必要性及使用软件模型的特点。

2. 操作提示

（1）以学习小组为单位分组进行讨论，每小组推荐一名成员进行汇报。

（2）通过上网查阅软件建模相关资料进行更为详细的了解。

（3）通过类比方式，理解模型和建模的含义。

习　题

一、填空题

1．_____是对象的模板，即类是对一组有相同数据和相同操作的对象的定义。

2．面向对象的主要特征中，_____是子类自动共享父类数据和方法的机制，它由类的派生功能体现；同一消息为不同的对象接收时可产生完全不同的行动，这种现象称为_____。

3．在_____语言中最先强调了对象概念，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机制。

二、选择题

1．下列不属于面向对象方法的基本特性的是_____。

A．多态性　　B．封装性

C．继承性　　D．抽象性

2．“了解问题域所涉及的对象、对象间的关系和作用，然后构造问题的对象模型”，这是利用面向对象方法学进行软件系统开发过程中_____阶段的任务。

A．OOA　　B．OOD

C．OOI　　D．OOP

3．下列关于软件建模的用途，说法错误的是_____。

A．软件建模可以帮助进行系统设计

B．软件建模可以使具体的设计细节与需求分开

C．通过软件建模可以利用模型全面把握复杂的系统

D．软件建模可以直接生成最终的软件产品

三、简答题

1．举例说明面向对象的主要特征及优越性？

2．举例说明 OOA、OOD、OOP 的主要任务。

3．结合建筑实例，举例说明什么是模型？什么是建模？

课外拓展

1. 操作要求

（1）查阅百度百科（http://baike.baidu.com/）的关于面向对象、模型等相关词条的内容，并对相关词条进行适当的补充。

（2）结合您所使用过的编程语言，比较 C 与 C++/Java/C#语言间的区别。

2. 操作提示

（1）通过网址（http://baike.baidu.com）进入百度百科。

（2）通过学习小组讨论的形式完成本次课外拓展。