第 5 章 面向对象软件开发方法

面向对象方法使计算机解决问题的方式符合人类的思维方式，更能直接地描述客观世界，通过增加代码的可重用性、可扩充性和程序自动生成功能来提高软件开发效率，且大大减少软件维护的开销，已经被越来越多的软件开发人员所接受。本章介绍面向对象软件开发方法与技术，包括面向对象的基本特征、基本概念、面向对象分析、面向对象设计、面向对象测试等。

5.1 面向对象基本特征

面向对象的基本思想就是从现实世界中客观存在的事物（对象）出发，构造系统并在系统结构中运用人类的自然思维方式对问题领域内的任务、事情等的抽象，其具体思想概括如下。

（1）客观事物是由对象组成的，对象是在原事物基础上抽象的结果。

（2）对象是由属性和操作组成的，其属性反映了对象的数据信息特征，而操作则用来定义改变对象属性状态的各种操作方式。

（3）对象之间的联系通过消息传递机制来实现。

（4）对象可以按其属性来归类。

（5）对象具有封装的特性，可达到软件（程序和模块）复用的目的。

此外，面向对象方法强调在软件开发过程中面向客观世界或问题域中的事物，采用人类在认识客观世界的过程中普遍运用的思维方法，直观、自然地描述客观世界中的有关事物。面向对象方法的基本特征主要有抽象性、封装性、继承性和多态性。

1. 抽象性

把众多的事物进行归纳、分类是人们在认识客观世界时经常采用的思维方法，“物以类聚，人以群分”就是分类的意思，分类所依据的原则是抽象。**抽象**（abstract）就是忽略事物中与当前目标无关的非本质特征，更充分地注意与当前目标有关的本质特征。从而找出事物的共性，并把具有共性的事物划为一类，得到一个抽象的概念。例如，在设计一个学生成绩管理系统的过程中，考查学生张华这个对象时，就只关心其班级、学号、成绩等，而忽略其身高、体重等信息。因此，抽象性是对事物的抽象概括描述，实现了客观世界向计算机世界的转化。将客观事物抽象成对象及类是比较难的过程，也是面向对象方法的第一步。例如，将学生抽象成对象及类的过程如图 5.1 所示。

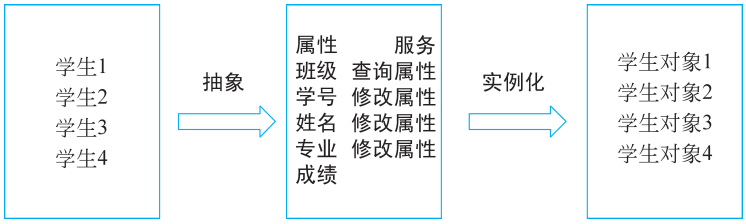

图 5.1　抽象过程示意图

2. 封装性

封装(encapsulation)就是把对象的属性和行为结合成一个独立的单位，并尽可能隐蔽对象的内部细节。图 5.1 中的学生类也反映了封装性。封装有两个含义：一是把对象的全部属性和行为结合在一起，形成一个不可分割的独立单位。对象的属性值(除了公有的属性值)只能由这个对象的行为来读取和修改；二是尽可能隐蔽对象的内部细节，对外形成一道屏障，与外部的联系只能通过外部接口实现。

封装的信息隐蔽作用反映了事物的相对独立性，可以只关心它对外所提供的接口，即能做什么，而不注意其内部细节，即如何提供这些服务。例如，用陶瓷封装起来的一块集成电路芯片，其内部电路是不可见的，而且使用者也不关心它的内部结构，只关心芯片引脚的个数、引脚的电气参数及引脚提供的功能，利用这些引脚，使用者将各种不同的芯片连接起来，就能组装成具有一定功能的模块。

封装的结果使对象以外的部分不能随意存取对象的内部属性，从而有效地避免了外部错误对它的影响，大大减小了查错和排错的难度。另外，当对象内部进行修改时，由于它只通过少量的外部接口对外提供服务，因此同样减小了内部的修改对外部的影响。同时，如果一味地强调封装，那么对象的任何属性都不允许外部直接存取，要增加许多没有其他意义，只负责读或写的行为。这为编程工作增加了负担，增加了运行开销，并且使得程序显得臃肿。为了避免这一点，在语言的具体实现过程中应使对象有不同程度的可见性，进而与客观世界的具体情况相符合。

封装机制将对象的使用者与设计者分开，使用者不必知道对象行为实现的细节，只需要用设计者提供的外部接口让对象去做。封装的结果实际上隐蔽了复杂性，并提供了代码重用性，从而降低了软件开发的难度。

3. 继承性

客观事物既有共性，也有特性。如果只考虑事物的共性，而不考虑事物的特性，就不能反映出客观世界中事物之间的层次关系，不能完整地、正确地对客观世界进行抽象描述。运用抽象的原则就是舍弃对象的特性，提取其共性，从而得到适合一个对象集的类。如果在这个类的基础上，再考虑抽象过程中被舍弃的一部分对象的特性，则可形成一个新的类，这个类具有前一个类的全部特征，是前一个类的子集，形成一种层次结构，即继承结构，如图 5.2 所示。

继承(inheritance)是一种联结类与类的层次模型。继承性是指特殊类的对象拥有其一般类的属性和行为。**继承意味着“自动地拥有”，即特殊类中不必重新定义已在一般类中定义过的属性和行为，而它却自动地、隐含地拥有其一般类的属性与行为。**继承允许和鼓励类的重用，提供了一种明确表述共性的方法。一个特殊类既有自己新定义的属性和行为，又有

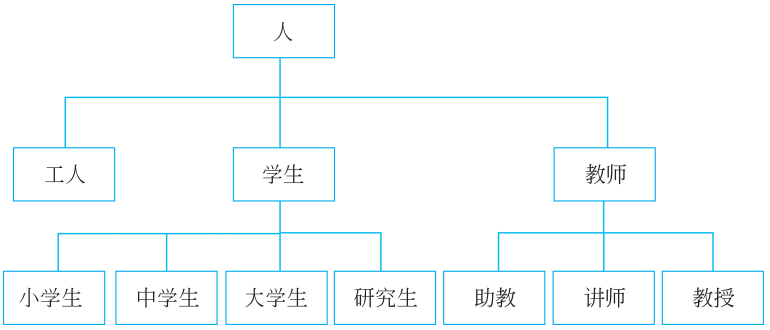

图 5.2 类的继承结构

继承下来的属性和行为。尽管继承下来的属性和行为是隐式的，但无论在概念上还是在实际效果上，都是这个类的属性和行为。当这个特殊类被它更下层的特殊类继承时，它继承来的和自己定义的属性和行为又被下一层的特殊类继承下去。因此，**继承是传递的**，体现了大自然中特殊与一般的关系。

在软件开发过程中，继承性实现了软件模块的可重用性、独立性，缩短了开发周期，提高了软件开发的效率，同时使软件易于维护和修改。这是因为要修改或增加某一属性或行为，只需在相应的类中进行改动，而它的派生类皆自动地、隐含地作了相应的改动。

由此可见，继承是对客观世界的直接反映，通过类的继承，能够实现对问题的深入抽象描述，反映出人类认识问题的发展过程。

4. 多态性

面向对象设计借鉴了客观世界的多态性，体现在不同的对象收到相同的消息时产生多种不同的行为方式。例如，在一般类“几何图形”中定义了一个行为“绘图”，但并不确定执行时到底画一个什么图形。特殊类“椭圆”和“多边形”都继承了几何图形类的绘图行为，但其功能却不同，一个是要画出一个椭圆，另一个是要画出一个多边形。这样一个绘图的消息发出后，椭圆、多边形等类的对象接收到这个消息后各自执行不同的绘图函数。如图 5.3 所示，这就是多态性的表现。

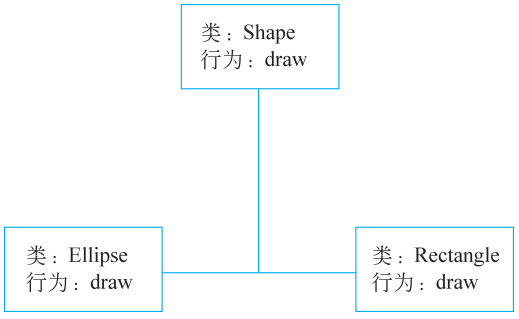

图 5.3 多态性示意图

具体来说，**多态性**(polymorphism)是指类中同一函数名对应多个具有相似功能的不同函数，可以使用相同的调用方式来调用这些具有不同功能的同名函数。

继承性和多态性的结合，可以生成一系列虽类似但独一无二的对象。由于继承性，这些对象共享许多相似的特征；由于多态性，针对相同的消息，不同对象可以有独特的表现方式，

实现特性化的设计。

5.2　面向对象基本概念

面向对象技术是一种以对象为基础，以事件或消息来驱动对象执行处理的程序设计技术。它以数据为中心而不是以功能为中心来描述系统，数据相对于功能而言具有更强的稳定性。它将数据和对数据的操作封装在一起，作为一个整体来处理，采用数据抽象和信息隐蔽技术，将这个整体抽象成一种新的数据类型——类，并且考虑类间联系和类重用性。另外，面向对象程序的控制流程由运行时各种事件的实际发生来触发，而不再由预定顺序来决定，更符合实际。事件驱动程序执行围绕消息的产生与处理，靠消息循环机制来实现。

例如，用面向对象技术来解决学生管理方面的问题。重点应该放在学生上，要了解在管理工作中，学生的主要属性，要对学生做些什么操作等，并把他们作为一个整体来对待，形成一个类，称为学生类。作为其实例，可以建立许多具体的学生，而每一个具体的学生就是学生类的一个对象。学生类中的数据和操作可以提供给相应的应用程序共享，还可以在学生类的基础上派生出大学生类、中学生类或小学生类等，实现代码的高度重用。

1. 对象

与人们认识客观世界的规律一样，面向对象技术认为客观世界是由各种各样的对象组成的，每种对象都有各自的内部状态和运动规律，不同对象间的相互作用和联系就构成了各种不同的系统，构成了客观世界。在面向对象程序中，客观世界被描绘成一系列完全自治、封装的对象，这些对象通过外部接口访问其他对象。可见，对象是组成一个系统的基本逻辑单元，是一个有组织形式的含有信息的实体。

对象(object)由**属性**(attribute)和**行为**(action)两部分组成。对象只有在具有属性和行为的情况下才有意义，属性是用来描述对象静态特征的一个数据项，行为是用来描述对象动态特征的一个操作。对象是包含客观事物特征的抽象实体，是属性和行为的封装体，在程序设计领域，可以用“对象＝数据＋作用于这些数据上的操作”这一公式来表达。

类(class)是具有相同属性和行为的一组对象的集合，它为属于该类的全部对象提供了统一的抽象描述，其内部包括属性和行为两个主要部分，**类是对象集合的再抽象。**

类与对象的关系如同一个模具与用这个模具铸造出来的铸件之间的关系。类给出了属于该类的全部对象的抽象定义，而对象则是符合这种定义的一个实体。所以，**一个对象又称为类的一个实例**(instance)。

2. 消息与事件

消息(message)是描述事件发生的信息，**事件**(event)由多个消息组成。消息是对象之间发出的行为请求。封装使对象成为一个相对独立的实体，而消息机制为它们提供了一个相互间动态联系的途径，使它们的行为能互相配合，构成一个有机的运行系统。

对象通过对外提供的行为在系统中发挥自己的作用，当系统中的其他对象请求该对象执行某个行为时，就向这个对象发送一个消息，这个对象就响应这个请求，完成指定的行为。

程序的执行取决于事件发生的顺序，由顺序产生的消息驱动，不必预先确定消息产生的顺序，更符合客观世界的实际。

5.3 面向对象方法

面向对象方法主要包括面向对象分析(OOA)和面向对象设计(OOD)等建模过程。当完成建模后,一般使用面向对象测试(OOT)来验证面向对象设计的正确性与完备性。

5.3.1 面向对象分析

面向对象的分析,就是运用面向对象方法进行系统分析。OOA 所强调的是在系统调查资料的基础上,针对面向对象(OO)方法所需要的素材进行的归类分析和整理。

OOA 过程并不是从考虑对象开始,而是从理解系统的使用方式开始,如果系统是人机交互的,则考虑被人使用的方式;如果是设计过程控制的,则考虑被机器使用的方式;如果是系统协调和控制应用,则考虑被其他程序使用的方式。

OOA 的关键是识别出问题域内的对象,并分析它们相互间的关系,最终建立起问题域的简洁、精确、可理解的正确模型。在用面向对象观点建立起的 3 种模型中,对象模型是最基本、最重要、最核心的模型。

OOA 的基本任务是:运用面向对象方法,对问题域和系统责任进行分析和理解,对其中的事物和它们之间的关系产生正确的认识,找出描述问题域及系统责任所需要的类及对象,定义这些类和对象的属性与服务,以及它们之间所形成的结构、静态联系和动态联系。最终目的是产生一个符合用户需求,并能直接反映问题域和系统责任的 OOA 模型及其详细说明。

OOA 的对象是对问题域中事物的完整映射,包括事物的数据特征(属性)和行为特征(服务)。OOA 的结构与连接如实地反映了问题域中事物间的各种关系。OOA 强调从问题域中的实际事物以及与系统责任有关的概念出发来构造系统模型。OOA 要求系统各个单元成分之间接口尽可能少,当需求不断变化时,OOA 把系统中最易变化的因素隔离起来,把需求变化所引起的影响局部化。

5.3.2 OOA 主要原则

1. 抽象

抽象是从许多事物中舍弃个别的、非本质的特征,抽取共同的、本质性的特征。抽象是形成概念的必须手段。

抽象原则有两方面的意义:第一,尽管问题域中的事物是很复杂的,但是分析师并不需要了解和描述它们的一切,只需要分析研究其中与系统目标有关的事物及其本质性特征。第二,通过舍弃个体事物在细节上的差异,抽取其共同特征而得到一批事物的抽象概念。

抽象是面向对象方法中使用最为广泛的原则。抽象原则包括过程抽象和数据抽象两方面。

过程抽象是指任何一个完成确定功能的操作序列,其使用者都可以把它看作一个单一的实体,尽管实际上它可能是由一系列更低级的操作完成的。

数据抽象是根据施加于数据之上的操作来定义数据类型,并限定数据的值只能由这些操作来修改和观察。数据抽象是 OOA 的核心原则。它强调把数据(属性)和操作(服务)结

合为一个不可分的系统单位(即对象),对象的外部只需要知道它做什么,而不必知道它如何做。

2. 封装

封装是把对象的属性和服务结合为一个不可分的系统单位,并尽可能隐蔽对象的内部细节。

3. 继承

特殊类的对象拥有的其一般类的全部属性与服务,称为特殊类对一般类的继承。在OOA中运用继承原则,就是在每个由一般类和特殊类形成的一般与特殊结构中,把一般类的对象实例和所有特殊类的对象实例都共同具有的属性和服务,一次性地在一般类中进行显式的定义。在特殊类中不再重复地定义一般类中已定义的东西,但是在语义上,特殊类却自动地、隐含地拥有它的一般类(以及所有更上层的一般类)中定义的全部属性和服务。继承原则可以使系统模型比较简洁,也比较清晰。

4. 分类

分类就是把具有相同属性和服务的对象划分为一类,用类作为这些对象的抽象描述。分类原则实际上是抽象原则运用于对象描述时的一种表现形式。

5. 聚合

聚合又称为组装,把一个复杂的事物看成若干比较简单的事物的组装体,从而简化对复杂事物的描述。OOA中运用聚合时要区分事物的整体和它的组成部分,形成一个整体-部分结构,以清晰地表达它们之间的关系。

6. 关联

关联是人类思考问题时经常运用的思维方法,通过一个事物联想到另外的事物。能使人发生联想的原因是事物之间确实存在着某些联系。OOA中运用关联原则就是在系统模型中表示对象之间的静态联系,这种联系信息是系统责任所需要的。

7. 消息通信

消息通信原则要求对象之间只能通过消息进行通信,而不允许在对象之外直接地存取对象内部的属性。通过消息进行通信是由于封装原则而引起的。在OOA中要求用消息连接表示出对象之间的动态联系。

8. 粒度控制

一般来讲,人在面对一个复杂的问题域时,不可能在同一时刻既能纵观全局,又能洞察秋毫。因此需要控制自己的视野:考虑全局时,注意其大的组成部分,暂时不需要详察每一部分的具体的细节;考虑某部分的细节时则暂时撇开其余的部分。

9. 行为分析

现实世界中事物的行为是复杂的。由大量的事物所构成的问题域中各种行为往往相互依赖、相互交织。确定行为的归属和作用范围,OOA以对象为单位分析系统中的各种行为,并用对象的服务加以表示,服务只作用于它所在的对象自身的属性,并通过消息连接描述对象服务之间的依赖关系。

5.3.3　面向对象设计模型

面向对象设计得到的模型包含对象的3个要素,即静态结构(对象模型)、交互次序(动

态模型)和数据变换(功能模型)。

根据解决的问题不同,这 3 个子模型的重要程度也不同。几乎解决任何一个问题,都需要从客观世界实体及实体间相互关系抽象出极有价值的对象模型;当问题涉及交互作用和时序时(如用户界面及过程控制等),动态模型是重要的;解决运算量很大的问题(如高级语言编译、科学与工程计算等),则涉及重要的功能模型。动态模型和功能模型中都包含了对象模型中的操作(即服务或方法)。

1. 对象模型

对象模型描述的是现实世界中对象的静态结构,即对象的标识、对象的属性、对象的操作和对象之间的关系。

对象模型对用例模型进行分析,把系统分解成互相协作的分析类,通过类图、对象图描述对象、对象属性或对象间关系,是系统的静态模型,为动态模型和功能模型提供了不可缺少的框架,是动态模型和功能模型赖以活动的基础。

2. 动态模型

动态模型描述了对象中与时间和操作次序有关的各种因素,它关心的是对象的状态是如何变化的,这些变化是如何控制的。动态模型可以用状态图表示,每个对象有它自己的一个状态图,其中的"结点"表示对象在不同时刻的状态,"边"表示状态之间的变化。动态模型描述了系统必须实现的操作。

3. 功能模型

功能模型描述了系统内值的变化,以及通过值的变化表现出来的系统功能、映射、约束和功能依赖的条件。功能模型只考虑系统"做"什么而不考虑系统何时"做"和如何"做"。功能模型说明了系统是如何响应外部事件的。

5.3.4 面向对象建模过程

面向对象建模大体上按照如下顺序进行:建立功能模型、建立对象模型、建立动态模型、定义服务。

1. 建立功能模型

功能模型从功能角度描述对象属性值的变化和相关的函数操作,表明了系统中数据之间的依赖关系以及有关的数据处理功能,它由一组数据流图组成。其中的处理功能可以用 IPO 图、伪码等多种方式进一步描述。

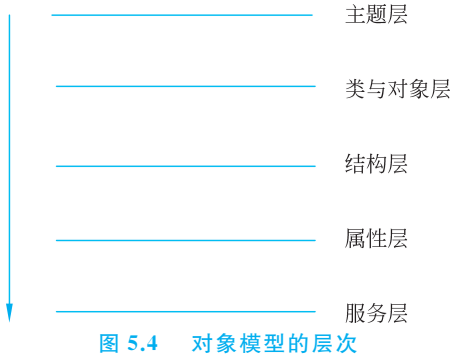

图 5.4 对象模型的层次

建立功能模型首先要画出顶层数据流图,然后对顶层图进行分解,详细描述系统加工、数据变换等,最后描述图中每个处理的功能。

2. 建立对象模型

复杂问题(大型系统)的对象模型由下述 5 个层次组成:主题层(也称为范畴层)、类与对象层、结构层、属性层和服务层,如图 5.4 所示。

建立对象模型典型的工作步骤是:先确定类与对象,对于大型复杂的问题还要识别结构,识别主题,然后定义属性,以进一步描述它们;接下来

利用适当的关系进一步合并和组织类。

1）确定类与对象

类与对象是在问题域中客观存在的，系统分析师的主要任务就是通过分析找出这些类与对象。首先，找出所有候选的类与对象；其次，从候选的类与对象中筛选掉不正确的或不必要的项。

步骤 1：找出候选的类与对象

对象是对问题域中有意义的事物的抽象，它们既可能是物理实体，也可能是抽象概念，在分析所面临的问题时，可以参照几类常见事物，找出在当前问题域中的候选类与对象。

另一种更简单的分析方法，是所谓的非正式分析。这种分析方法以用自然语言书写的需求陈述为依据，把陈述中的名词作为类与对象的候选者，用形容词作为确定属性的线索，把动词作为服务（操作）候选者。当然，用这种简单方法确定的候选者是非常不准确的，其中往往包含大量不正确的或不必要的事物，还必须经过更进一步的严格筛选。通常，非正式分析是更详细、更精确的正式的面向对象分析的一个很好的开端。

步骤 2：筛选出正确的类与对象

非正式分析仅仅帮助我们找到一些候选的类与对象，接下来应该严格考察候选对象，从中去掉不正确的或不必要的，仅保留确实应该记录其信息或需要其提供服务的那些对象。筛选时主要依据如下标准，删除不正确或不必要的类与对象。

（1）**冗余**（如果两个类表达了同样的信息）。

（2）**无关**（仅需要把与本问题密切相关的类与对象放进目标系统中）。

（3）**笼统**（需求陈述中笼统的、泛指的名词）。

（4）**属性**（在需求陈述中有些名词实际上描述的是其他对象的属性）。

（5）**操作**（正确地决定把某些词作为类还是作为类中定义的操作）。

（6）**实现**（去掉仅和实现有关的候选的类与对象）。

2）确定关联

两个或多个对象之间的相互依赖、相互作用的关系就是关联。分析确定关联，能促使分析师考虑问题域的边缘情况，有助于发现那些尚未被发现的类与对象。

步骤 1：初步确定关联

在需求陈述中使用的描述性动词或动词词组，通常表示关联关系。因此，在初步确定关联时，大多数关联可以通过直接提取需求陈述中的动词词组而得出。通过分析需求陈述，还能发现一些在陈述中隐含的关联。最后，分析师还应该与用户及领域专家讨论问题域实体间的相互依赖、相互作用关系，根据领域知识再进一步补充一些关联。

步骤 2：自顶向下

把现有类细化成更具体的子类，这模拟了人类的演绎思维过程。从应用域中常常能明显看出应该做的自顶向下的具体化工作。例如，带有形容词修饰的名词词组往往暗示了一些具体类。但是，在分析阶段应该避免过度细化。

3）识别结构

结构指的是多种对象的组织方式，用来反映问题空间中的复杂事物和复杂关系。这里的结构包括两种：分类结构与组装结构。分类结构针对的是事物的类别之间的组织关系，组织结构则对应着事物的整体与部件之间的组合关系。

使用分类结构，可以按事物的类别对问题空间进行层次化的划分，体现现实世界中事物的一般性与特殊性。例如，在交通工具、汽车、飞机、火车这几件事物中，具有一般性的是交通工具，其他则是相对特殊化的。因此可以将汽车、飞机、火车这几种事物的共有特征概括在交通工具之中，也就是把对应于这些共有特征的属性和服务放在“交通工具”这种对象之中，而其他需要表示的属性和服务则按其特殊性放在“汽车”“飞机”“火车”这几种对象之中，在结构上，则按这种一般与特殊的关系，将这几种对象划分在两个层次中，如图 5.5 所示。

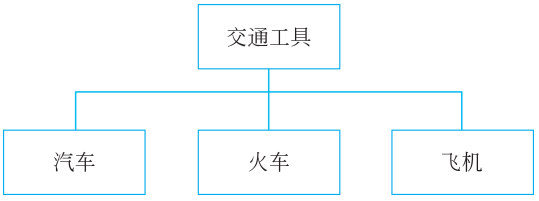

图 5.5　分类结构示例

组织结构表示事物的整体与部件之间的关系。例如，把汽车看成一个整体，那么发动机、变速箱、刹车装置等都是汽车的部件，相对于汽车这个整体就分别是一个局部。

4）识别主题

对一个实际的目标系统，特别是大型系统而言，尽管通过对象和结构的认定对问题空间中的事物进行了抽象和概括，但对象和结构的数目仍然是可观的，因此如果不对数目众多的对象和结构进行进一步的抽象，势必造成对分析结果理解上的混乱，也难以搞清对象、结构之间的关联关系，因此引入主题的概念。

主题是一种关于模型的抽象机制，它给出了一个分析模型的概貌。也就是通过划分主题，把一个大型、复杂的对象模型分解成几个不同的概念范畴。

主题直观地来看就是一个名词或名词短语，与对象的名字类似，只是抽象的程度不同。识别主题的一般方法是：为每个结构追加一个主题；为每种对象追加一个主题；如果当前的主题的数目超过了 7 个，就对已有的主题进行归并，归并的原则是，当两个主题对应的属性和服务有着较密切的关联时，就将它们归并成一个主题。

5）定义属性

属性是数据元素，用于描述对象或分类结构的实例。

定义一个属性有 3 个基本原则：首先，要确认它对响应对象或分类结构的每个实例都是适用的；其次，对满足第一个条件的属性还要考察其在现实世界中与这种事物的关系是不是足够密切；最后，认定的属性应该是一种相对的原子概念，即不依赖于其他并列属性就可以被理解。

3. 建立动态模型

当问题涉及交互作用和时序时（例如用户界面及过程控制等），建立动态模型则是很重要的。

建立动态模型的第一步，编写典型交互行为的脚本。脚本是指系统在某执行期间内出现的一系列事件。编写脚本的目的，是保证不遗漏重要的交互步骤，它有助于确保整个交互过程的正确性和清晰性。

第二步，从脚本中提取出事件，确定触发每个事件的动作对象以及接受事件的目标

对象。

第三步，排列事件发生的次序，确定每个对象可能有的状态以及状态间的转换关系。

第四步，比较各个对象的状态，检查它们之间的一致性，确保事件之间的匹配。

4. 定义服务

通常在完整地定义每个类中的服务之前，需要先建立起动态模型和功能模型，通过对这两种模型的研究，能够更正确更合理地确定每个类应该提供哪些服务。

正如前面已经指出的那样，“对象”是由描述其属性的数据，及可以对这些数据施加的操作（即服务）封装在一起构成的独立单元。因此，为建立完整的动态模型，既要确定类的属性，又要定义类的服务。在确定类中应有的服务时，既要考虑类实体的常规行为，又要考虑在本系统中特殊需要的服务。

首先，考虑常规行为：在分析阶段可以认为类中定义的每个属性都是可以访问的，即假设在每个类中都定义了读、写该类每个属性的操作。

其次，从动态模型和功能模型中总结出特殊服务。

最后，应该尽量利用继承机制以减少所需定义的服务数目。

总之，面向对象分析大体上按照如下顺序进行：建立功能模型，寻找类与对象，识别结构，识别主题，定义属性，建立动态模型，定义服务。分析不可能严格地按照预定顺序进行，大型、复杂系统的模型需要反复构造多遍才能建成。通常，先构造出模型的子集，然后再逐渐扩充，直到完全、充分地理解了整个问题，最终才能把模型建立起来。

5.4　本章小结

总之，面向对象分析方法使得软件工程师能够通过对对象、属性和操作（作为主要的建模成分）的表示来对问题建模。OOA 中引入了许多面向对象的概念和原则，如对象、属性、服务、继承、封装等，并利用这些概念和原则来分析、认识和理解客观世界，将客观世界中的实体抽象为问题域中的对象，即问题对象，分析客观世界中问题的结构，明确为完成系统功能，对象间应具有的联系和相互作用。

OOA 过程从用例（use case）的定义开始；然后应用类-责任-协作者建模技术来为类及其属性与操作建立文档，它也提供了发生在对象间的协作的初始视图；然后是对象的分类和类层次的创建，子系统可用于封装相关的对象，对象-关系模型提供了对象间如何相互连接的指示，而对象-行为模型指明了个体对象的行为和 OO 系统的整体行为。

习　题　5

1. 什么是面向对象？
2. 面向对象有哪些特征？
3. 简述面向对象与面向过程的优缺点，并举例说明。
4. 面向对象的方法主要包括哪些？
5. 如何面向对象建模？