第二章 面向对象技术和建模基础

主讲教师：徐丙凤

- 面向对象技术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。使用**UML**可以建立易于理解和使用的对象模型，可以方便程序设计者根据模型实现对应的软件。
- 本章主要介绍关于面向对象的基本概念，这些概念也是系统建模的基础。重点理解面向对象的相关概念，掌握面向对象的特征。了解面向对象开发的相关技术和过程。

第二章 面向对象技术和建模基础

- 2.1 面向对象的基本概念
- 2.2 面向对象开发
- 2.3 软件建模概述
- 2.4 小结

2.1面向对象的基本概念

- 在学习面向对象程序设计之前，一般都会学习面向过程的程序设计，例如，使用C面向过程的程序设计语言，面向过程的语言是按流程化的思想来组织的。
- 在这些语言的设计思想中，通常将存放基本数据类型的变量作为程序处理对象、以变量的赋值作为程序的基本操作、以变量值的改变作为程序运行的状态。
- 这种程序设计风格存在着数据抽象简单、信息完全暴露、算法复杂、无法很好地描述客观世界等缺点。在程序设计过程中，为了实现有限度的代码重用，公共代码被组织成为过程或函数。当需要代码重用时，调用已经组织好的过程或函数。在这种应用方式中，如果软件项目庞大，程序的调试和维护将变得异常困难。

- 面向对象的程序设计思想是将数据以及对于这些数据的操作，封装在了一个单独的数据结构中。这种模式更近似于现实世界，在这里，所有的对象都同时拥有属性以及与这些属性相关的行为。
- 对象之间的联系是通过消息来实现的，消息是请求对象执行某一处理或回答某些信息的要求。某个对象在执行相应的处理时，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消息。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和另外的对象联系。所以，一个面向对象程序的执行，就是靠对象间传递消息来完成的。

- 面向对象程序设计是一种新兴的程序设计方法，它使用对象、类、继承、封装、消息等基本概念来进行程序的设计。从现实世界中客观存在的事物（即对象）出发来构造软件系统，并且在系统构造中尽可能运用人类的自然思维方式。
- 开发一个软件是为了解决某些问题，这些问题所涉及的业务范围称作该软件的问题域。其应用领域不仅仅是软件，还有计算机体系结构和人工智能等。使用UML进行系统建模时，首先就必须弄清楚什么是对象，在系统的分析与设计过程中如何利用对象。

2.1.1面向对象的方法

- 在软件开发过程中，客户会不断地提出各种修改要求，即使在软件投入使用后，也常常需要对其做出修改，在用结构化开发的程序中，这种修改往往是很困难的，而且还会因为计划或考虑不周，不但旧错误没有得到彻底改正，又引入了新的错误；另一方面，在过去的程序开发中，代码的重用率很低，使得程序员的效率并不高，为提高软件系统的稳定性、可修改性和可重用性，人们在实践中逐渐创造面向对象方法。
- 面向对象方法的出发点和基本原则是尽可能模拟人类习惯的思考问题的方式，使软件开发的方法与过程尽可能接近人类认识世界、解决问题的方法与过程。由于客观世界的问题都是由客观世界中的实体及实体相互间的关系构成的，因此我们把客观世界中的实体抽象为对象。

面向对象方法具有以下几个要点：

（1）客观世界是由各种对象组成的，任何事物都是对象，复杂的对象可以由比较简单的对象以某种方式组合而成。面向对象的软件系统是由对象组成的，软件中的任何元素都是对象，复杂的软件对象由比较简单的对象组合而成。

（2）把所有对象都划分成各种对象类，每个对象类都定义了一组数据和一组方法，数据用于表示对象的静态属性，是对象的状态信息。因此，每当建立该对象类的一个新实例时，就按照类中对数据的定义为这个新对象生成一组专用的数据，以便描述该对象独特的属性值。

（3）按照子类与父类的关系，把若干个对象类组成一个层次结构的系统。

（4）对象彼此之间仅能通过传递消息进行联系。

2.1.2 对象

- 对象（Object）是面向对象的基本构造单元，是系统中用来描述客观事物的一个实体，一个对象由一组属性和对属性进行操作的一组方法组成。
- 在面向对象的系统中，对象是一个封装数据属性和操作行为的实体。数据描述了对象的状态，操作指的是操作私有数据，改变对象的状态。当其他对象向本对象发出消息，本对象响应时，其操作才得以实现，在对象内的操作通常叫做方法。

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。如汽车、书、计算机、桌子、鼠标、机器人等等，都属于有形的实体，也是最易于识别的对象。

（2）作用：指人或组织，如医生、教师、员工、学生、公司、部门等所起的作用。

（3）事件：在特定时间所发生的事，如飞行、事故、中断、开会等。

（4）性能说明：制造厂或企业，往往对产品的性能进行全面的说明，如车厂对车辆的性能说明，往往要列出型号及各种性能指标等。

对象具有如下特征：

(1) 模块性

模块性指的是对象是一个独立存在的实体。从外部可以了解它的功能，其内部细节是“隐蔽”的，不受外界干扰，对象之间的相互依赖性很小。因此，模块性体现了抽象和信息的隐蔽。它使得一个复杂的软件系统可以通过定义一组相对独立的模块来完成，这些独立模块之间只需交换一些为了完成系统功能所必须交换的信息就行。当模块内部的实现发生变化而导致必须要修改程序时，只要对外接口操作的功能不变，就不会给软件系统带来影响。

(2) 继承

继承是利用已有的定义作为基础来建立新的定义，而不必重复定义它们。例如，汽车具有“车型”、“颜色”和“出厂日期”等属性，其子类吉普车、轿车及卡车都继承了这些属性。

(3)动态连接性

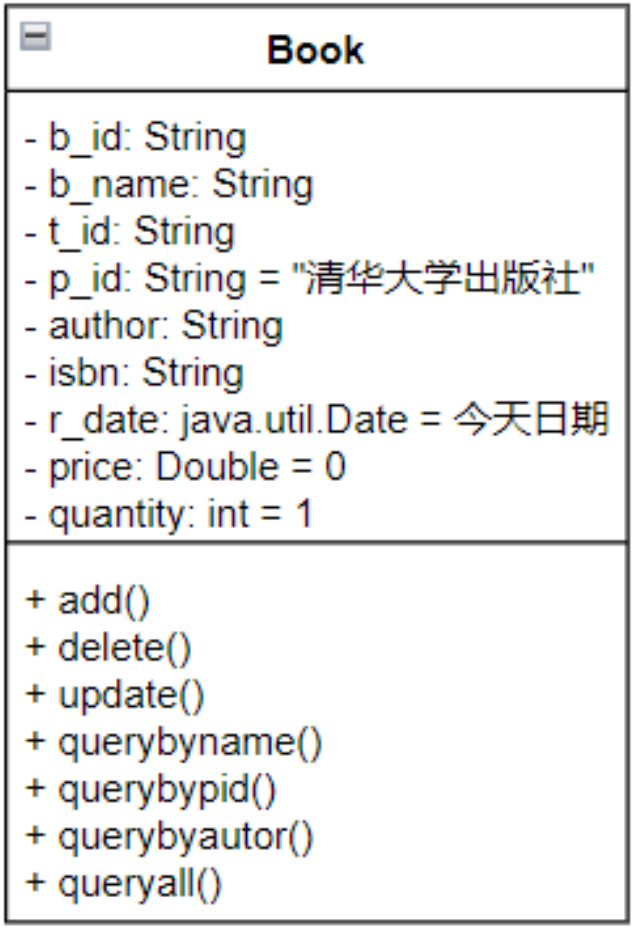
各个对象之间是通过传递消息来建立连接的。消息传递机制是面向对象语言的共同特性，其含义是将一条发送给一个对象的消息与包含该消息方法的对象联接起来，使得增加新的数据类型不需要改变现有的代码。

2.1.3 类

- 一个类定义了一组大体上相似的对象。一个类所包含的方法和数据描述一组对象的共同行为和属性。例如，窗口、车轮、玻璃等都是类的例子。类是在对象之上的抽象，有了类以后，对象则是类的具体化，是类的实例。类可以有子类和父类，形成层次结构。
- 类是对事物的抽象，它不是个体对象，而是描述一些对象的完整集合。
  - 例如，你可以把CPU看作是对象类，它具有CPU的共同属性，如：主频、指令集、Cache容量、运算位数、功率等。也可以考虑CPU的某个具体实例，如“Intel的P4处理器”。

- 类是静态的，类的语义和类之间的关系在程序执行前就已经定义好了，而对象是动态的，对象是在程序执行时被创建和删除的。

如图所示的类中类的名字是图书Book。该类有9个属性（b_id,b_name,t_id,p_id,author,isbn,r_date,price,quantity），7个方法（add,delete,update,querybyname,querybypid,querybyauthor,queryall）。



2.1.4 封装

- 封装（Encapsulation）就是把一个对象的方法和属性组合成一个独立的单位，并尽可能隐蔽对象的属性、方法和实现细节的过程，仅仅将接口进行对外公开。包含两个含义：
  - 把对象的全部属性和方法结合在一起，形成一个不可分割的独立单位。对象的属性（除了公有访问的属性）只能由这个对象的方法来存取。
  - 信息隐蔽，即尽可能隐蔽了对象的内部实现细节，与外部的联系只能通过外部接口来实现。

2.1.4 封装

- 在访问类的时候，根据其封装的特点，对外访问时提供了4种访问控制级别：

（1）**public**：公有访问。最高一级的访问，所有的类都可以访问。

（2）**protected**：受保护的。只有同一个包中的类或者子类可以进行公开访问。

（3）**private**：私有访问。最低一级的访问，只能在对象的内部访问，不对外公开。

（4）**default**：默认的。属于当前目录（包）下的类都可以访问。

- 封装的最大优点是：

（1）方便了使用者对类和对象的操作，并降低了使用者错误修改其属性的可能性。

（2）体现了系统之间的松散耦合关系并提高了系统的独立性。

（3）提高了程序的复用性。

（4）针对大型的开发系统，降低了开发风险。如果整个系统开发失败。一些相对独立的子系统仍然存在可用价值。

2.1.5 继承

- 继承是一种由已有的类创建新类的机制。利用继承，我们可以先创建一个拥有共有属性的一般类，根据该一般类再创建具有特殊属性的新类，新类继承一般类的状态和行为，并根据需要增加它自己的新的状态和行为。
- 继承（Inheritance）是一种一般类与特殊类的层次模型。继承性是指特殊类的对象具有其一般类的属性和方法，在其之上又增加了自己的特殊属性和方法。继承意味着在特殊类中不用重新定义在一般类中已经定义过的属性和方法，特殊类可以自动地、隐含地拥有其一般类的属性与方法。

- 继承的过程，就是从一般到特殊的过程。继承性提供了父类和子类之间共享数据和方法的一种机制。继承表示的是类之间的一种关系，在定义和实现一个类的时候，可以通过一个已经存在的类来创建新类，把这个已经存在的类作为父类，将其所定义的内容作为自己的内容的一部分、并加入一些新的内容。

- 继承性分为单重继承和多重继承两类。

单重继承：指的是一个子类只有一个父类。派生类仅从一个基类继承属性和方法，形成简单的树状层次结构。

多重继承：指的是一个子类可以有多个父类。派生类可以同时继承多个基类的成员。

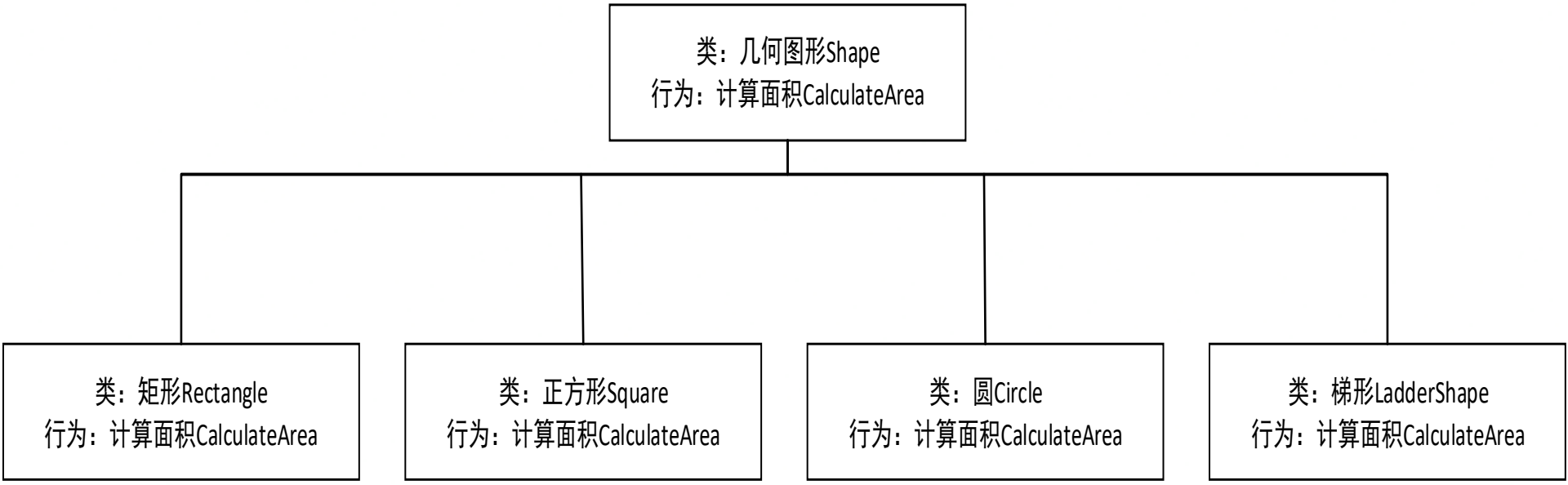
- 在软件开发过程中，继承性体现的是软件模块的可重用性和独立性，可以缩短软件的开发周期，提高软件的开发效率，并为日后的维护和修改软件提供了方便。因为如果要修改某个模块的功能，只需在相应的类中进行一些变动，而它派生的所有类都自动地、隐含地作了相应的改动。
- 继承真实地反映了客观世界中事物的层次关系，通过类的继承，能够实现对问题的深入抽象描述，反映出事物的发展过程，继承性是面向对象程序设计语言不同于其他语言的最主要的特点。

2.1.6 多态

- 对象的多态是由封装和继承引出的面向对象程序设计语言的另一特征。主要体现在两个方面：方法重载时实现的静态多态和方法重写时实现的动态多态。
- 多态性（**Polymorphism**）是指类中同一函数名对应多个功能相似的不同函数，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，这些同名的函数可以是参数的个数或是类型不同，但是函数名相同，当进行调用的时候，根据所传的数据选定相应的函数，从而去执行不同的功能。
- 在面向对象程序设计中通过继承性和多态性的结合，可以生成许多类似但是功能却各不相同的对象。根据继承性的特点，这些对象共享一些相似的特征，并显出自己的特性；根据多态性，针对相同的消息，不同对象可以具有特殊的表现形式，实现个性化的设计。

2.1.6 多态

例如，在一般类“几何图形”中定义了“计算图形面积、周长或绘制图形”等行为，但是这些行为并不具备具体的含义，也就是说还不确定要计算什么几何图形的面积或者是绘制一个什么样的图形。根据一般类再定义特殊类如“矩形”、“正方形”、“圆”和“梯形”等，它们都继承了父类“几何图形”的“计算图形面积、周长或绘制图形”等行为，因此自动具有了“计算面积或绘制图形”的功能，但每个特殊类的功能却不一样，一个是要计算矩形的面积或画出一个矩形，另一个是计算正方形的面积或是要画出一个正方形等功能。这样的计算面积或绘图的消息发出后，矩形、正方形等类的对象接收到这个消息后各自执行不同的计算面积或绘图函数。


类：几何图形Shape
行为：计算面积CalculateArea
类：矩形Rectangle
行为：计算面积CalculateArea
类：正方形Square
行为：计算面积CalculateArea
类：圆Circle
行为：计算面积CalculateArea
类：梯形LadderShape
行为：计算面积CalculateArea

2.1.7 消息

在面向对象程序设计中，对象之间要进行数据的传递，通过消息进行通信的，多个对象之间通过传递消息来请求或提供服务，从而使一个软件具有更强大的功能。

在面向对象的系统中，把“请求”或“命令”抽象成“消息”，当系统中的其他对象请求这个对象执行某个服务时，就将一个消息发送给另一个对象，接收到消息的对象将消息进行解释，然后响应这个请求，完成指定的服务。通常，把发送消息的对象称为发送者，把接收消息的对象称为接收者。通常，一个消息由以下几部分组成：

提供服务的对象名。

服务的标识，即方法名。

输入信息，即实际参数。

响应结果，即返回值或操作结果。

2.2 面向对象开发

如果遵照面向对象方法的思想进行软件系统的开发，过程共分成4个阶段：

- 系统调查和需求分析，分析问题并求解。对用户的开发需求以及要开发的系统所面临的问题进行调查和研究。针对复杂的问题领域，抽象出对象及其属性和方法。这一个阶段通常称之为面向对象分析，即**OOA**。
- 整理问题：对第一阶段的结果进一步抽象、归类整理。对每一部分进行分别的具体的设计，首先是进行类的设计，类的设计可能包含多个层次（利用继承、派生）。然后在这些类的基础之上，提出程序设计的思路和方法，对算法的设计。在设计阶段，不牵扯某一种具体的计算机语言，而是用一种更通用的描述工具进行描述，这个阶段即为面向对象设计**OOD**。
- 程序实现。利用面向对象的程序设计语言，进行系统的具体实现，即面向对象编程**OOP**。
- 系统测试。系统开发好后，在交付用户使用前，必须对程序进行严格的测试。测试的主要目的就是发现程序中的错误，进行改正，使得系统更健壮。面向对象测试时，采用面向对象的方法进行测试，以类作为测试的一个基本单元。这个阶段称为面向对象测试**OOT**。

2.2.1 系统调查和需求分析

- 面向对象的系统调查和需求分析阶段主要是提取系统的需求，也就是要分析出为了满足用户的需求，系统必须“做什么”（系统能提供的功能），而不是“怎么做”（系统如何实现）。

1）分析过程概述

- 在进行系统调查和需求分析阶段，系统分析员要对需求文档进行分析。通过分析可以发现并对需求文档中的歧义性、不一致性进行修正，剔除那些冗余内容，挖掘系统中应该存在的潜在内容，弥补系统中的不足，从而使需求文档更完整和准确。
- 对需求文档进行了分析和整理后，为了给面向对象分析过程提供依据，需要进行需求建模。这时系统分析员根据提取的用户需求，对用户的需求进行深入地理解，识别出问题领域内的对象，并分析对象之间的关系，抽象出目标系统需要完成的任务，这样就可以利用OOA模型准确地表示出来，即用面向对象观点建立对象模型、动态模型和功能模型。
- 经过需求的分析和建模，最后对所获得的需求进行评审。通过用户、领域专家、系统分析员和系统设计人员的评审，并进行反复修改后，最终确定目标系统的需求规格说明。

2）实例需求文档

- 需求文档也叫需求陈述或问题陈述。对于需要开发的任何一个系统，需求陈述是首要任务。因为系统最终是要由用户使用，而在该过程中，主要是陈述用户的需求，即该系统应该“做什么”， 而不是“怎么做”，即系统要完成的任务是什么，而不是解决问题的方法。
- 在进行需求陈述时，必须要清楚地定义所要解决问题的目标。如果目标模糊，将会影响整个系统分析、设计和实现等后续开发阶段的所有工作。需求质量的好坏直接影响到整个系统的质量，因此，需求陈述是很关键的过程，如果想准确表达用户的要求，在对需求进行陈述时需要分析人员和用户一起研究和讨论。

2.2.2 面向对象分析方法

- 面向对象的分析方法，指的是按照面向对象的概念和方法，在对任务的分析中，根据客观存在的事物以及事物之间的关系，归纳出相关的对象，包括对象的属性、行为以及对象之间的联系，并将具有共同属性和行为的对象用一个类来表示。
- 通过面向对象的分析，建立一个能反映真实工作情况的需求模型。在这个阶段所形成的模型只是一个比较粗略的模型。OOA所强调的是在系统调查资料的基础上，进行的对象的归类分析和整理。
- 使用OOA方法对系统调查和需求分析进行分析处理时，一般要遵循前面讲述的抽象、封装、继承、消息通信等原则。
- 在使用OOA具体地分析一个事物时，一般要分如下几个阶段：

1）识别并筛选对象

- 按照对象的定义，对象应该是实际问题域中有意义的个体或概念实体。对象具有目标软件系统所关心的属性。对象应该以某种方式与系统发生关联，即对象必须与系统中其他有意义的对象进行消息传递，并提供外部服务。
- 通过对用户需求分析文档的分析可以找出所有的名词或名词短语，合并同义词，这些是极有可能成为对象的。除去具有动作含义的名词，这些动词将被描述为对象的操作而不是对象本身。

2）标识对象的属性

- 属性是对问题域中对象性质的一个描述，对象在系统中所有可能的状态就是属性的取值。对象一般具有很多属性，但在分析阶段就要分析出对象的哪些属性是和系统紧密相关的。
- 在问题域中，如何能够识别出对象的哪些属性是有意义？要识别出所关心的潜在属性，需要对问题领域涉及到的知识进行深刻的理解。
- 在识别属性的过程中，对于问题领域中的某个实体，不但要求其取值有意义，而且它本身在系统中必须是独立存在的。这时应该将该实体作为一个对象，而不能作为另一对象的属性。此外，为了保持需求模型的简洁性，一般将省略对象的一些导出属性。例如，“年龄”可通过“出生日期”和系统当前时间导出，因此，不应该将“年龄”作为人的基本属性。

3）识别对象的行为

对象的行为可以简单地理解为对象对外提供的所有的功能。比如说，在面向对象模型中，一个对象要处理另一对象的请求、查询或命令，即响应外部的事件，要完成某项操作，这种操作将改变对象自身的属性值或系统的状态，这些都是对象的行为。当对象受到外部事件的刺激或接收另一个对象传来的消息后，为完成某项操作，响应外部事件，该对象可能又需要向其他对象发送消息。因此，我们可以把整个系统看成是对象之间的相互通信，以及在通信过程中引发的动作。

- 一般可以将对象的行为分为以下三类：

① 对象生命周期中的创建、维护、删除行为。例如，图书管理系统中的图书信息的创建，删除和修改等行为。

② 计算性行为

典型的计算性行为主要包括：利用基本的对象属性值计算出派生出的属性值，以及为了响应其他对象的请求，完成某些数据处理功能，并将结果返回。这类计算性行为往往完成的是数据处理功能，即对象提供的外部的计算性行为。因此，分析人员可以在定义对象的外部行为时，针对其他对象发出的消息请求提取计算性行为。

③ 监视性行为或称响应行为

为了提取对象的响应行为，分析人员需要对对象的主要状态进行定义。对于每一个状态，列出可能的外部事件，预期的反应，并进行适当的精化。例如，“图书”对象的状态可以为借出、库存等，在每一状态可处理的事件及预期反应可以表示为响应行为。

2.2.3 面向对象设计方法

- 面向对象的设计方法（简称为OOD）是面向对象方法中一个中间过渡环节。其主要作用是对OOA分析的结果进行规范化的整理，以便为面向对象程序设计阶段打下基础。在OOD的设计过程中，主要进行如下几个过程：

1）精化对象的定义规格

对于OOA所抽象出来的对象和类以及在分析过程中产生的分析文档，在OOD过程中，根据设计要求对其进行整理和精化，使之更能符合面向对象程序设计的需要。整理和精化的过程主要包括两个方面：一个是根据面向对象的概念模型，整理分析所确定的对象结构、属性和方法等内容，纠正错误的内容，删去不必要和重复的内容等。另一个是进行分类整理，这样便于下一步数据库设计、程序处理模块设计。整理的方法主要是进行归类，即对类和对象、属性、方法和结构等进行归类。

2）数据模型和数据库设计

- 数据模型的设计是对系统中的类和对象的属性、方法等内容的确定，以及消息连接的方式、系统访问数据模型的方法等内容的确定。最后将每个对象实例化数据都映射到面向对象的库结构模型中。

3）优化

- 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。集成化使得单个构件有机地结合在一起，相互支持。

2.3 软件建模概述

模型提供了系统的蓝图，可以包括详细的计划，也可以包括从很高的层次考虑系统的总体计划。模型可以是结构性的，强调系统的组织。它也可以是行为性的，强调系统的动态方面。

建模是为了能够更好地理解正在开发的系统。通过建模，要达到如下4个目的：

模型有助于按照实际情况或按照所需要的样式对系统进行可视化。

模型能够规约系统的结构或行为。

模型给出了指导构造系统的模板。

模型对做出的决策进行文档化。

2.3.1 软件建模的概念

软件建模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。可视化的建模使用一些图形符号进行建模，可视化建模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。

2.3.2 软件建模的用途

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。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，从而使用了软件建模技术。

2.3.2 软件建模的优点

软件建模的优点主要有如下几点：

- 使用模型便于从整体上、宏观上把握问题，以便更好地解决问题。
- 软件建模可以加强软件工作人员之间的沟通。便于提早发现问题。
- 模型为代码生成提供依据，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。给出了一个指导我们构造系统的模板。并对我们做出的决策进行文档化。

2.4小结

- 面向对象程序设计是一种新兴的程序设计方法，或者是一种新的程序设计规范，它使用对象、类、继承、封装、消息等基本概念来进行程序设计。在面向对象方法中需要明确什么是对象、类，以及类的相关特征。
- 对象（**Object**）是面向对象的基本构造单元，是系统中用来描述客观事物的一个实体。一个对象由一组属性和对属性进行操作的一组方法组成。一个类定义了一组大体上相似的对象。一个类所包含的方法和数据描述一组对象的共同行为和属性。

2.4小结

- 随着软件规模的迅速增大，软件人员面临的问题十分复杂，需要考虑的因素很多，需要将软件整个开发过程规范化，明确软件开发过程中每个阶段的任务，在保证前一个阶段的正确性的情况下，再进行下一个阶段的工作。这就是软件工程学需要研究和解决的问题。
- 面向对象的软件工程一般包括面向对象分析、面向对象设计、面向对象编程和面向对象测试。
- 如果设计一个规模大的软件，就要严格按照面向对象软件工程的几个阶段进行开发。