第5章

软件系统构造方法

面向对象的思维是构造复杂软件系统的基本思维模式。基于面向对象思维的程序设计语言，可以构造复杂及特色化的软件。面向对象的思维及程序设计语言是软件工程专业人才必须掌握的。软件构造能力是软件工程专业毕业生的主要竞争力。目前，软件构造方法学主要涉及面向对象的软件系统构造、基于组件的软件系统构造和面向服务的软件系统构造。

5.1 导学导教

5.1.1 内容导学

本章内容导学图如图 5-1 所示。

图 5-1 软件系统构造方法内容导学图

5.1.2 教学目标

1. 知识目标

掌握面向对象的思维和基本概念、类、对象和消息的基本语法，掌握 UML 的用例图、类图、顺序图、状态图、构件图和部署图表达能力和基本事物，理解函数、类、动态链接库、COM 组件和 Web Service 的特点、适用范围、演化关系以及区别和联系，理解两层架构和三层架构的应用场景和特点。

2. 能力目标

能够运用面向对象的核心概念分析和设计现实软件，能够使用面向对象的基本语言进行基本面向对象程序构造。能够根据实际运用 VS2022 实现动态链接库和 COM 组件和基于 SSM 的管理软件。能够运用 VS2022 和 Eclipse Maven 项目分别实现两层架构和 SSM 架构的软件。能够基于 IIS 运用 VS2022 实现 Web 服务。能够运用 IBM Rational Rose 进行 UML 软件系统建模。

3. 思政目标

能够熟练运用开发工具快速排除语法错误，准确分析、定位和解决逻辑错误，分析可能存在的隐患。

5.2 面向对象的软件构造

5.2.1 面向对象的基本思想与方法

引例："取"与"送"思维的差别。

描述：交通局制定了一个"某道路双向改单向"的通知，要让每一个司机知道，它采用送或取的策略。送，即交通局要把通知送达每个司机的手中。显然，"送"是不可能完成的任务。取，即交通局把该通知发布到一个地点，而由每个司机到该地点获取通知，"取"则可达此目的。

面向过程为中心的构造思维类似于"送"，而面向对象为中心的构造思维类似于"取"。面向对象类似于人们认识现实世界的方法，认为软件系统是由可区分的对象构成的，对象是一个个具有某种功能/行为的个体，这些功能/行为由某种程序来完成。对象具有独立性，当其执行自身的程序时无须其他对象的干预。对象也具有交互性，一个对象可以向其他对象发送消息，请求协助完成某个任务。

1. 对象

对象是面向对象开发方法中的最基本概念，是组成一个系统的最基本元素，系统功能是由若干互相联系的对象来完成的。例如，在高校人力资源管理系统中，教职工是一个对象，他有自己的属性，包括姓名、性别、职称、出生日期等，他也有自己的行为，例如，授课等。人事科也是一个对象，它也有自己的属性（办公地点、人员结构等）和行为（人事调动、档案录入等）。

需要注意的是，有些对象是看得见或摸得着的，但也有一些对象只能被感知得到。例如，一次会议、一堂课，尽管看不到它，也摸不着它，但它仍然是一个对象。

在面向对象的系统开发方法中，找出应用系统中存在的所有对象是非常重要的。但是，在需求分析时所寻找的对象比"能够看得到、摸得到或感知得到的事物"要复杂得多。下面给出了对象的定义。

对象是人们要研究的，能够看得到、摸得到或感知得到的事物，对象既可以是一些简单

的数据,也可以是一些复杂的事件。

对于这个定义需要解释几点。首先,这里所谓的“事物”对象具有多种类别,包括:人、地点、事物和事件。例如,在“人力资源管理软件系统”中定义的对象教辅人员、教师和校领导的类别是教职工,教室、办公室和宿舍的对象类别是地点,工资、职称、职务的对象类别是事物,考核、考勤、工资发放和人员调动的对象类别是事件。

下面来考虑定义中的“数据”。在面向对象方法中,“数据”与面向对象中的属性概念息息相关。

属性是指用来描述事物特征的数据。例如,对于一个教师对象,可以通过下列属性来描述:教师编号,姓名,职称,工作年限,所属院系,家庭电话,办公电话,出生日期,担任职务等。这些属性可以描述很多教师,他们是一类对象,对于其中的单个教师,称之为一个对象实例。

对象实例是包含一定属性值的对象,这些属性描述的是特定的人、地点或事件。例如,每个教师都有特定的值。例如,编号“03061428”,姓名“张三”,职称“讲师”,工作年限“5年”,所属院系“管理学院”。

对象的行为是指对象的动作,对象所具有的功能,可以对其他对象发来的消息做出响应。在面向对象的系统中,对象行为一般表现为对象的方法、操作或服务。

对于系统中事物或对象的理解和描述在结构化方法和面向对象方法中是不同的。在结构化方法中可能简单地认为它是静止的对象,不会考虑它会执行什么动作。但在面向对象的系统开发中,除了具有自己的属性外,还具有很多行为,包括档案维护、人事调动、教职工考核等。

此外,还要特别提一下对象的一个特性——封装。封装又称为信息隐藏,是指对象的属性和行为被隐藏在一个黑箱子里,作为对象的使用者不能也不必知道对象属性及行为的实现细节和步骤,只能通过设计者提供的对象接口来访问对象,使对象的使用者和设计者分开,从而达到对象信息隐藏的目的。其特点如下。

(1) 限定了对象之间通信的方式,即只能通过对象接口通信。

(2) 隐藏和保护了对象的内部实现细节,包括对象的数据和操作方法。对象由属性和行为组成,其属性和行为被封装和打包成一个完全独立的对象,修改或访问对象只能通过对象的行为,从而提高了对象的独立性和可重用性。

2. 类与封装

面向对象方法中另一个概念是类,可以把若干相似的对象抽象成类。类又称为对象类,是具有相同属性和行为的若干对象的抽象,定义了同类对象的属性和功能。对象是类的实例,是实际运行的个体。例如,一个高校有很多教师,他们具有相同的属性,包括编号、姓名、性别、出生日期、职称等,也具有相同的行为,包括授课、考勤、调动等。在 UML 中如何表示一个类呢?图 5-2 给出了类与对象的关系。

对象能够隐藏其内部的各种细节,运用访问控制权限将其对象内部的属性和函数予以隐藏,即“封装”。对象只通过消息与外界进行交互。

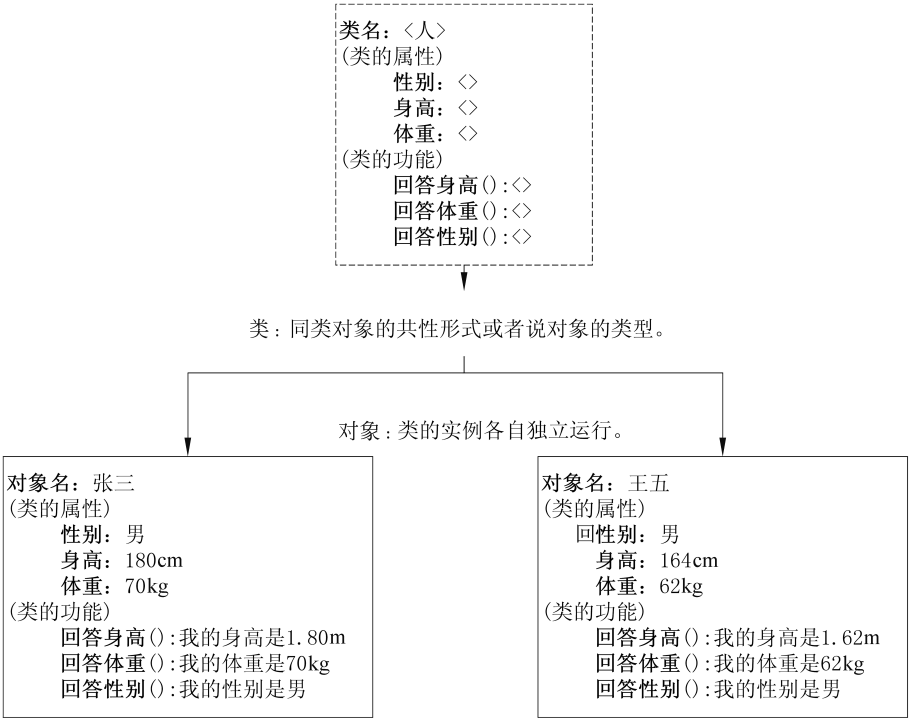

图 5-2　类与对象的关系示例

3. 消息与事件

消息是应用程序内部对象之间传递的内容，如指示、打听、请求、……它是对象之间交互的唯一渠道。消息以“对象.函数()”的方式由一个对象向另一个对象发起调用，被请求对象以执行函数予以响应并返回结果，如图 5-3 所示。

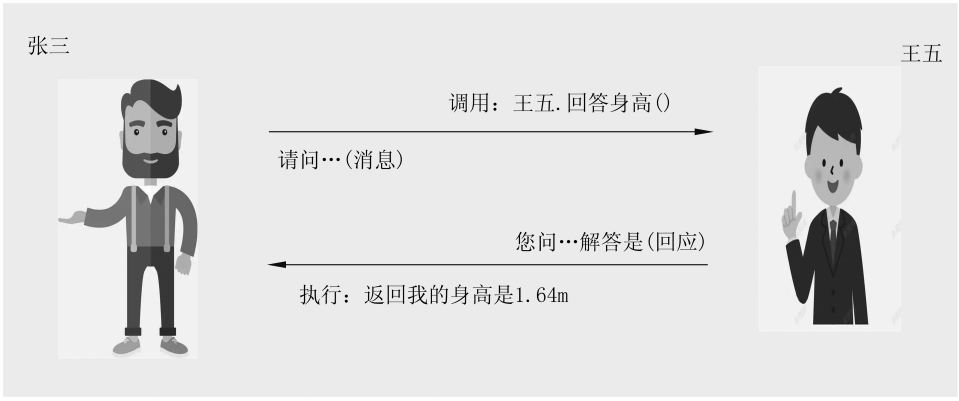

图 5-3　消息调用与回应示例

事件是应用程序外部产生的能够激活对象功能的一种特殊消息。当事件发生后给所涉及对象发送一个消息，对象便可执行相应的功能，简称为“事件驱动—消息处理”。应用程序外部典型的事件有鼠标事件、键盘事件、网络通信事件等。例如，张三老师在上课过程中，手

机铃响或者有人敲门,就属于师生正常上课的外部事件,上课过程中老师的提问则属于消息。

例 1 请用类、对象、消息和事件描述张三老师的一次上课行为。

解答:如图 5-4 所示。

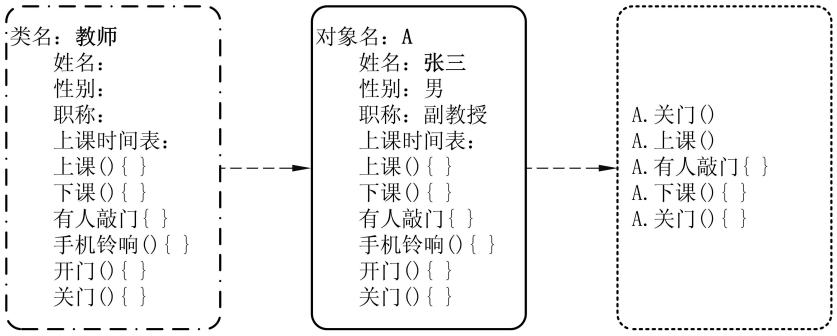

图 5-4 类、对象、消息和事件的应用示例

4. 继承与多态

继承是指在定义一个对象类时,其属性和方法可以重用于另一个对象类,也可以将其属性和方法传递给另一个对象类。例如,高校人力资源管理系统中的教师和管理人员分别是一个类,但两个类之间又非常相似,既有相似的属性(如编号、姓名、出生日期等),也有类似的行为(考勤等)。其实,在高校中教师和管理人员都属于教职工,因此,可以定义一个教职工类,把他们共同的属性和行为抽象出来并赋予教职工类,在定义教师和管理人员两个类时就可以直接重用教职工类的属性和行为,而不用再重复定义,在这个过程中充分体现了继承的概念及其优越性。

面向对象程序设计的三大原则是封装、继承和多态。继承是子类自动共享父类的数据和方法的机制,它是由类的派生功能体现的。继承具有传递性,使得一个类可以继承另一个类的属性和方法,这样通过抽象出共同的属性和方法组建新的类,便于代码的重用。而多态是指不同类型的对象接收相同的消息时产生不同的行为,这里的消息主要是对类成员函数的调用,而不同的行为是指类成员函数的不同实现。当对象接收到发送给它的消息时,根据该对象所属的类,动态选用在该类中定义的实现算法。

5.2.2 面向对象的程序设计语言

1. 类的定义

假设属性和方法外界可以访问,以 C++ 语法为例,类(对象的结构)的定义如下。

```
class 类名{
    public:
        类型: 属性名 1;
        类型: 属性名 2;
        …
        类型: 属性名 n;
```

```
        类型 函数名 1(){函数体 1}
        类型 函数名 2(){函数体 2}
        …
        类型 函数名 n(){函数体 n}
}
```

例 2 二维平面点和矩形类的示例定义。

```
class Point                                          //点
{
    public:
        int x,y;                                     //平面坐标
        void Set(int a, int b){
            x=a, y=b;
        }
}
class Rectangle                                      //矩形
{
    public:
        Point Center;                                //矩形中心点
        int Length, Width;                           //矩形的长和宽
        void SetCenter(int a, int b){
            Center.Set(a,b);
        }
        int SetLength(int L){Length=L;}
        int SetWidth(int W){Width=W;}
        int Area(){return Length * Width;}
        int Girth(){return 2 * (Length+Width);}
}
```

2. 对象的创建与运行

对象在运行时创建,每个状态属性都被分配内存空间。对象的方法占用内存情况分为如下两种。

(1) 所有对象共享一份类的程序代码内存空间。

(2) 每个对象的方法占有一份独立的程序代码内存空间。

例 3 例 2 中的类实例化后对象的创建与运行的示例程序。

```
void main(){
    Rectangle myRect;              //创建 Rectangle 类的对象 myRect
    myRect.SetCenter(100,100);     //执行 myRect 对象的 SetCenter()方法
    myRect.SetLength(100);         //执行 myRect 对象的 SetLength()方法
    myRect.SetWidth(50);           //执行 myRect 对象的 SetWidth()方法
    printf("The Rectangle Area is %d", myRectArea());
    //执行 myRect 对象的 myRectArea()方法后将其值打印输出
}
```

3. 面向过程与面向对象程序设计语言的比较

面向过程的程序设计语言构成要素有变量与常量、表达式、语句与函数。程序是具有先

后次序或调用关系的函数集合,函数是若干先后次序的语句集合。面向过程构造程序的方法可由粗到细,也可由细到粗。由粗到细的路线是为控制复杂性,先以函数来代替琐碎的细节,着重考虑函数之间的关系以及如何解决问题。然后再去考虑其中的每一个函数。而函数的处理同样采取这种思路。反之,也可由细到粗,上一层的函数依据下层函数来编写,确认正确后再转至更上层问题处理。一般先编写基础性的函数并确认其正确后,再处理上一层次的问题。

面向对象程序设计语言的基本构成要素是类与对象、消息与事件、函数或者方法。对象是类的实例化,应用程序接收外部事件后向特定对象发送消息,对象收到请求后运用相应的函数进行消息处理。一个应用程序由若干对象构成,应用程序向不同对象发送不同的消息,不同对象收到不同的消息后进行消息处理。一个对象可以与另一个对象通过消息进行交互,从而协作完成任务。

5.2.3 统一建模语言

统一建模语言或标准建模语言(Unified Modeling Language,UML)始于 1997 年一个 OMG 标准,它是一个支持模型化和软件系统开发的图形化语言,为软件开发的所有阶段提供模型化和可视化支持,能进行需求分析、软件设计甚至软件构造和配置。面向对象的分析与设计方法的发展在 20 世纪 80 年代末~20 世纪 90 年代中出现了一个高潮,UML 是这个高潮的产物。它不仅统一了 Booch、Rumbaugh 和 Jacobson 的表示方法,而且对其做了进一步的发展,并最终统一为大众所接受的标准建模语言。

UML 是一种可视化的建模语言,具有严密的语法、语义规范,其所定义的图形符号简单、直观,使得软件建模简洁明了,容易掌握使用,同时提供了便于不同人之间有效地共享和交流设计结果的机制。UML 实现了面向对象的可视化建模,软件模型独立于开发过程和程序设计语言。UML 适用于各种规模的系统开发,能促进软件复用,方便地集成已有的系统并有效减少开发中的各种风险。

面向对象软件的开发生命周期是由分析、设计、演化和维护四个阶段组成的,每个阶段都可以反馈,是一个迭代、渐增的开发过程。这种迭代渐增过程不仅贯穿整个软件生命周期,并且表现在每个阶段中特别是分析(全局分析、局部设计)和设计(全局设计、局部设计)阶段。这与传统的程序设计所遵循的分析、设计、编码、调试和维护五个阶段组成的瀑布式生命周期有着很大不同,更符合随着人的认识逐步深化,软件分析、设计逐步求精的规律,更有利于软件的编码、调试和维护、扩展。

1. UML 的主要特点

首先,UML 是 Booch、OMT 和 OOSE 等方法基本概念的拓展与延伸。

其次,UML 吸取了面向对象技术领域中其他流派的长处。UML 符号表示考虑了各种方法的图形表示,删掉了大量易引起混乱的、多余的和极少使用的符号,也添加了一些新符号。因此,在 UML 中汇入了面向对象领域中的众多思想。这些思想并不是 UML 的开发者们发明的,而是开发者们依据最优秀的 OO 方法和丰富的计算机科学实践经验综合提炼而成的。

最后,UML 在演变过程中还提出了一些新的概念。在 UML 标准中新加了模板

(Stereotypes)、职责(Responsibilities)、扩展机制(Extensibility mechanisms)、线程(Threads)、过程(Processes)、分布式(Distribution)、并发(Concurrency)、模式(Patterns)、合作(Collaborations)、活动图(Activity diagram)等新概念,并清晰地区分类型(Type)、类(Class)和实例(Instance)、细化(Refinement)、接口(Interfaces)和组件(Components)等概念。因此可以认为,UML是一种先进实用的标准建模语言,但其中某些概念尚待实践来验证,UML也必然存在一个进化过程。

2. 应用领域

UML的目标是以面向对象图的方式来描述任何类型的系统,具有很宽的应用领域。其中最常用的是建立软件系统的模型,但它同样可以用于描述非软件领域的系统,如机械系统、企业机构或业务过程,以及处理复杂数据的软件系统、具有实时要求的工业系统或工业过程等。总之,UML是一个通用的标准建模语言,可以对任何具有静态结构和动态行为的系统进行建模。

此外,UML适用于系统开发过程中从需求规格描述到系统完成后测试的不同阶段。在需求分析阶段,可以用用例来捕获用户需求。通过用例建模,描述对系统感兴趣的外部角色及其对系统(用例)的功能要求。分析阶段主要关心问题域中的主要概念(如抽象、类和对象等)和机制,需要识别这些类以及它们相互间的关系,并用UML类图来描述。为实现用例,类之间需要协作,这可以用UML动态模型来描述。在分析阶段,只对问题域的对象(现实世界的概念)建模,而不考虑定义软件系统中技术细节的类(如处理用户接口、数据库、通信和并行性等问题的类)。这些技术细节将在设计阶段引入,因此设计阶段为构造阶段提供更详细的规格说明。

编程(构造)是一个独立的阶段,其任务是用面向对象编程语言将来自设计阶段的类转换成实际的代码。在用UML建立分析和设计模型时,应尽量避免考虑把模型转换成某种特定的编程语言。因为在早期阶段,模型仅仅是理解和分析系统结构的工具,过早考虑编码问题十分不利于建立简单正确的模型。UML模型还可作为测试阶段的依据。系统通常需要经过单元测试、集成测试、系统测试和验收测试。不同的测试小组使用不同的UML图作为测试依据:单元测试使用类图和类规格说明;集成测试使用部件图和合作图;系统测试使用用例图来验证系统的行为;验收测试由用户进行,以验证系统测试的结果是否满足在分析阶段确定的需求。

软件的概念模型能帮助开发人员认识所设计系统的需求,便于用户和开发人员充分交流并达成一致;能管理需求分析、设计阶段的繁杂信息;能准确描述需求分析、设计的结果,从而成为编码、调试和维护的依据。使用UML进行建模可以提高程序员在OO软件开发的每个阶段的工作效率——从记录新的问题领域中心概念的一些最初想法,到组织开发人员与领域专家进行交流,直到最终软件产品的图形文档记录等。这些优点主要表现在直观、易于理解问题领域和发现设计中的错误,特别是那些有关对象间关系的错误;便于准确地从模型到实际应用编码的转换;用于系统的调试和出问题时检错的依据。

总之,UML适用于以面向对象技术来描述任何类型的系统,而且适用于系统开发的不同阶段,从需求规格描述直至系统完成后的测试和维护。

3. UML 分析工具

下面简单介绍 UML 主要建模图以及它们的用途,并着重给出类图和状态图的图例及简单建模示例。

1) 用例图

用例表明了一个参与者与计算机系统交互来完成业务活动。用例是一种高层的描述,它可能包含完成这个用例的所有步骤。用例图刻画软件系统的功能,一般用于需求分析,它以图形的形式描述了系统或外部系统与用户之间的交互行为,说明了谁将使用这个系统,用户在系统中可以做什么,以及用户将以什么样的方式与系统交互。用例图是软件开发者和用户进行交流与沟通的非常有效的工具,帮助需求分析员正确提炼出系统的具体需求。此外,它还附有用例描述,具体描述每个用例中详细的交互细节。

用例图是把应满足用户需求的基本功能(集)聚合起来表示的强大工具。对于正在构造的新系统,用例描述该系统应该做什么;对于已构造完毕的系统,用例则反映了系统能够完成什么样的功能。构建用例图通过系统开发者与系统的客户(或最终使用者)共同协商完成,他们要反复讨论需求的规格说明,达成共识,明确系统的基本功能,为后面阶段的工作打下基础。

构成用例图的图元是用例、参与者。用例用于描述系统的功能,也就是从外部用户的角度观察系统应具备哪些功能,帮助分析人员理解系统的行为,它是对系统功能的宏观描述。一个完整的系统中通常包含若干个用例,每个用例具体说明应完成的功能,代表系统的所有基本功能(集)。参与者是与系统进行交互的外部实体,它可以是系统用户,也可以是其他系统或硬件设备,总之,凡是需要与系统交互的任何东西都可以称作参与者。

2) 类图

类图描述软件系统的静态结构,给出系统中各个类的结构,包括系统中各个类的组成,以及这些类间存在的各种关联关系。类图表达类和类之间的关系,如图 5-5 所示。类由一个矩形被一条分割线分成上下两个部分表示,上部给出类的名称,下部给出其属性和方法(函数名)。类之间的关系有继承、组合、聚合和关联关系 4 种。

继承关系由一个空心三角箭头连接线连接两个类,箭头指向父类。组合关系和聚合关系分别由一个黑色实心和黑色空心菱形箭头连接线连接两个类,箭头指向聚合/组合类。关联关系由一条无箭头的连接线连接关联类的双方,连接线两端加上关联关系 $m:n$、$1:m$、$m:1$ 或者 $1:1$,如图 5-5 中的类 5 和类 1 之间的关系所示。

聚合、组合和关联的区别在于聚合关系是“has-a”关系。组合关系是“contains-a”关系。聚合关系表示整体与部分的关系比较弱,部分事物的对象与整体事物的对象的生存期无关,一旦删除了被聚合的部分对象不一定删除了代表整体事物的对象,也就是整体和部分关系可以分开。组合中一旦删除了组合对象,同时也就删除了代表部分事物的对象,即组合的整体和部分不可分开。聚合是关联关系的一种特例,强调了关联关系的 $1:m$ 或 $m:1$关系,1 代表整体,m 代表部分。

例 4 用类图的聚合关系表达读者所借阅的图书。

解答:假设读者(Reader)为整体,图书(Book)为部分,类的聚合关系如图 5-6 所示。

例 5 用类图的关联关系表达读者所借阅的图书。

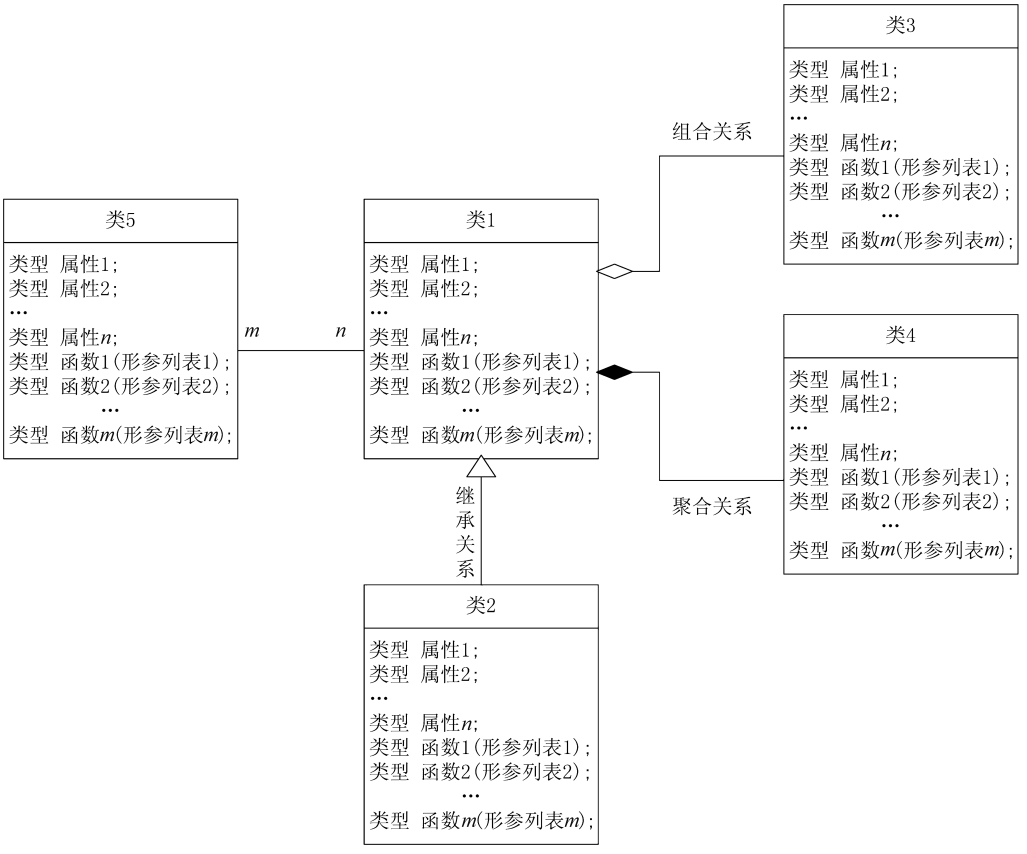

图 5-5 类图图元示例

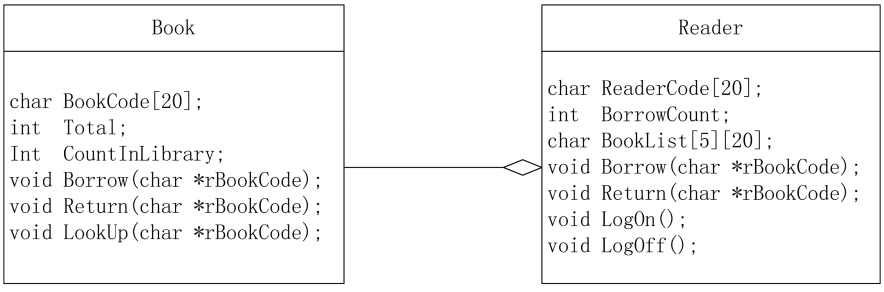

图 5-6 基于两个类相聚合的软件构造

解答：用一个借阅关系类来表达读者和图书的关联，如图 5-7 所示。

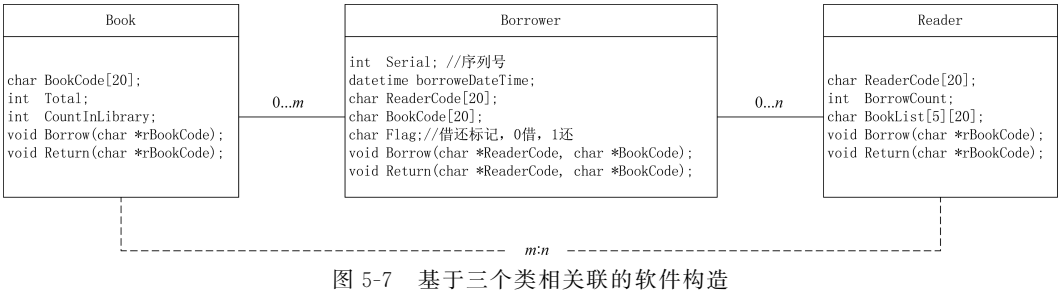

图 5-7 基于三个类相关联的软件构造