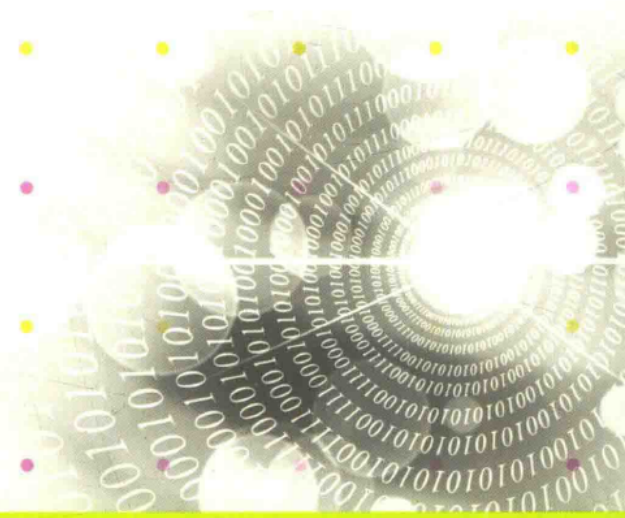
应用型本科信息大类专业"十三五"规划教材



软件建模技术与应用

主编　王智超　曾辉　姜东洋

华中科技大学出版社
http://www.hustp.com

应用型本科信息大类专业"十三五"规划教材

软件建模技术与应用

主　编　王智超　曾　辉　姜东洋
副主编　胡　婧　郭　瑞　王莉莉

华中科技大学出版社
http://www.hustp.com
中国·武汉

内容简介

本书系统地介绍了软件建模的基础理论知识和实用技术方法。其中，基础理论以统一建模语言 UML 为核心，深入浅出地描述了在面向对象的软件开发过程中，如何使用 UML 标准构建系统生命周期中的各种常用模型；实用技术方法则结合业界广泛使用的 UML 开发工具 Rational Rose，并配以具体的软件系统案例进行了详细介绍，使读者能够轻松理解并快速掌握软件建模的技术方法。此外，每章后还附有操作练习题，着重培养读者的动手能力，使其在练习过程中能快速提高实际应用水平。

为了方便教学，本书还配有教学课件等教学资源包，任课教师和学生可以登录“我们爱读书”网（www.ibook4us.com）免费注册并浏览，任课教师可以发邮件至 hustpeiit@163.com 索取。

本书结构合理，语言简练易懂，适合作为高等院校计算机类相关专业的教材或教学参考书，也可以作为软件设计与开发人员的参考资料和相关培训教材。

图书在版编目(CIP)数据

软件建模技术与应用/王智超，曾辉，姜东洋主编. —武汉：华中科技大学出版社，2019.2
应用型本科信息大类专业“十三五”规划教材
ISBN 978-7-5680-4368-7

Ⅰ.①软… Ⅱ.①王… ②曾… ③姜… Ⅲ.①软件设计-高等学校-教材 Ⅳ.①TP311.1

中国版本图书馆 CIP 数据核字(2018)第 134419 号

软件建模技术与应用
Ruanjian Jianmo Jishu yu Yingyong

王智超 曾 辉 姜东洋 主编

策划编辑：康 序
责任编辑：康 序
责任监印：朱 玢
出版发行：华中科技大学出版社(中国·武汉) 电话：(027)81321913
武汉市东湖新技术开发区华工科技园 邮编：430223
录 排：武汉正风天下文化发展有限公司
印 刷：武汉华工鑫宏印务有限公司
开 本：787mm×1092mm 1/16
印 张：15.5
字 数：405 千字
版 次：2019 年 2 月第 1 版第 1 次印刷
定 价：38.00 元

本书若有印装质量问题，请向出版社营销中心调换
全国免费服务热线：400-6679-118 竭诚为您服务
版权所有 侵权必究

前言 PREFACE

统一建模语言(UML)是一种通用的可视化建模语言,适用于各种软件开发方法、软件生命周期的各个阶段、软件的各种应用领域以及各种软件开发工具。它是一种旨在统一过去建模技术的经验,吸收当今软件开发的实践经验从而形成一种标准的方法。UML 包括语义概念、表示法和指导规范,它提供了静态、动态、系统环境及组织结构的模型,为交互式的可视化建模工具所支持,支持现今大部分面向对象的开发过程,其目的是简化和强化现有面向对象的开发方法。Rational Rose 是目前广泛使用的面向对象可视化建模工具之一,可用于对系统的建模、设计与编码,还可对已有的系统实施逆向工程,实现代码模型转换,以便更好地开发与维护系统。UML 与 Rational Rose 的有机结合,在开发大型面向对象的应用中发挥着巨大的作用。

当前对于软件建模的教材需求层次多、范围广,因此需要有适应不同需求特色的教材。鉴于此,编者在实际教学经验的基础上,编写了本书。本书在内容的编排上注重实用性,在强调基本知识理解与基本技能训练的同时,更注重对读者创新能力的培养。

全书共分 15 章,各章的具体内容安排如下。

● 第 1 章　简要介绍面向对象技术,包括面向对象的基本概念、面向对象分析、面向对象设计和面向对象建模等基本知识。

● 第 2 章　简要介绍 UML 统一建模语言,包括 UML 的起源与发展历史、UML 的定义、UML 的特点、UML 的作用、UML 视图和 UML 机制等。

● 第 3 章　简要介绍软件建模工具 Rational Rose,包括 Rational Rose 的起源与发展、Rational Rose 的功能特点、Rational Rose 的运行环境、Rational Rose 的安装和 Rational Rose 的基本操作。

● 第 4 章　具体介绍了 UML 的使用过程(如 Rational 统一过程等)。包括软件工程过程定义、UML 过程的基础、传统的面向对象过程、Rational 统一过程和过程工具等。

● 第 5 章　具体介绍用例图,包括用例图的基本概念、用例图的组成元素、用例描述说明、Rational Rose 创建用例图方法和用例图建模案例分析等。

● 第 6 章　具体介绍类图与对象图,包括类图与对象图的基本概念、类图与对象图的组成元素、Rational Rose 创建类图与对象图方法和类图与对象图建模

案例分析等。

● 第 7 章　具体介绍序列图，包括序列图的基本概念、序列图的组成元素、Rational Rose 创建序列图方法和序列图建模案例分析等。

● 第 8 章　具体介绍协作图，包括协作图的基本概念、协作图的组成元素、Rational Rose 创建协作图方法和协作图建模案例分析等。

● 第 9 章　具体介绍状态图，包括状态图的基本概念、状态图的组成元素、Rational Rose 创建状态图方法和状态图建模案例分析等。

● 第 10 章　具体介绍活动图，包括活动图的基本概念、活动图的组成元素、Rational Rose 创建活动图方法和活动图建模案例分析等。

● 第 11 章　具体介绍包图，包括包图的基本概念、包图的组成元素、Rational Rose 创建包图方法和包图建模案例分析等。

● 第 12 章　具体介绍构件图，包括构件图的基本概念、构件图的组成元素、Rational Rose 创建构件图方法和构件图建模案例分析等。

● 第 13 章　具体介绍部署图，包括部署图的基本概念、部署图的组成元素、Rational Rose 创建部署图方法和部署图建模案例分析等。

● 第 14 章　具体介绍 UML 双向工程，包括双向工程基本概念、正向工程、逆向工程和 Rational Rose 双向工程实施等。

● 第 15 章　以具体案例为基础，详细描述完整的软件系统建模过程。

本书的主要特点如下。

(1) 内容全面细致，具有系统性。书中内容既包括面向对象理论介绍，又全面介绍了 UML 的基础知识，特别是对 Rational Rose 支持的图和模型元素进行了详细的讲解，同时给出了相关 Rational Rose 的具体操作。全书集理论、操作于一体。

(2) 案例讲解深入透彻。书中使用了一个具体的 BBS 论坛系统的建模案例，将其贯穿于各个 UML 模型的章节，每一章都力图给出建模时详细的分析过程，而非泛泛的建模结果，让读者在学习的过程中知道如何做以及为什么这样做，有助于读者边学习、边思考和边实践。

(3) 图文并茂，通俗易懂。本书在介绍每个章节、知识点、案例以及 Rational Rose 的使用时配有大量的图表，有助于读者更加直观地理解 UML 的理论知识，掌握 Rational Rose 的使用技巧。

本书适合作为高等学校计算机类专业的本科教材，也可作为 UML 建模人员的参考资料和相关培训教材。

本书由武昌理工学院王智超、曾辉，辽宁机电职业技术学院姜东洋担任主编；由武汉晴川学院胡婧、湖北文理学院理工学院郭瑞、大连工业大学艺术与信息工程学院王莉莉担任副主编。全书由王智超审核并统稿。编写过程中还得到了武昌理工学院信息工程学院许多老师的支持和帮助，并给出了许多好的建议，在此编者对他们表示衷心的感谢。

为了方便教学，本书还配有教学课件等教学资源包，任课教师和学生可以登录“我们爱读书”网(www.ibook4us.com)免费注册并浏览，任课教师可以发邮件至 hustpeiit@163.com 索取。

在本书的编写过程中，借鉴了许多相关的现行教材，在此谨表示衷心的感谢。由于作者水平有限，虽对本书进行反复的审核，但书中难免有错误和不足之处，希望读者给予批评指正，多提宝贵意见。

编　者

2019 年 1 月

此为试读，需要完整PDF请访问：www.ertongbook.com

目录 CONTENTS

第 1 章　面向对象基础 …… (1)

1.1　面向对象的概念 …… (1)

1.2　面向对象与面向过程的区别 …… (6)

1.3　面向对象分析 …… (8)

1.4　面向对象设计 …… (10)

1.5　面向对象软件建模 …… (12)

习题 1 …… (13)

第 2 章　UML 统一建模语言 …… (15)

2.1　UML 简介 …… (15)

2.2　UML 模型 …… (20)

2.3　UML 机制 …… (32)

2.4　UML 未来发展目标 …… (35)

习题 2 …… (36)

第 3 章　Rational Rose 软件建模工具 …… (38)

3.1　Rational Rose 的起源与发展 …… (38)

3.2　Rational Rose 的功能特点 …… (39)

3.3　Rational Rose 运行环境 …… (40)

3.4　Rational Rose 的安装过程 …… (40)

3.5　Rational Rose 操作介绍 …… (46)

习题 3 …… (60)

第 4 章　UML 使用过程 …… (61)

4.1　软件工程过程 …… (61)

4.2 UML 过程基础 …… (64)
4.3 传统的面向对象过程 …… (68)
4.4 Rational 统一过程 …… (70)
4.5 过程工具 …… (73)
习题 4 …… (75)
第 5 章 用例图 …… (76)
5.1 用例图概述 …… (76)
5.2 用例图的组成元素 …… (78)
5.3 使用 Rational Rose 建立用例图的方法 …… (85)
5.4 用例图建模案例分析 …… (91)
习题 5 …… (96)
第 6 章 类图与对象图 …… (98)
6.1 类图概述 …… (98)
6.2 类图的组成元素 …… (99)
6.3 对象图 …… (107)
6.4 使用 Rational Rose 建立类图的方法 …… (108)
6.5 类图建模案例分析 …… (114)
习题 6 …… (119)
第 7 章 序列图 …… (121)
7.1 序列图概述 …… (121)
7.2 序列图组成元素 …… (122)
7.3 序列图中的项目相关概念 …… (126)
7.4 使用 Rational Rose 建立序列图的方法 …… (127)
7.5 序列图建模案例分析 …… (131)
习题 7 …… (133)
第 8 章 协作图 …… (135)
8.1 协作图概述 …… (135)
8.2 协作图组成元素 …… (136)
8.3 序列图与协作图的比较 …… (137)
8.4 使用 Rational Rose 建立协作图的方法 …… (138)
8.5 协作图建模案例分析 …… (140)
习题 8 …… (142)

第 9 章　状态图 …… (143)
9.1　状态图概述 …… (143)
9.2　状态图的组成元素 …… (144)
9.3　状态的类型 …… (149)
9.4　使用 Rational Rose 建立状态图的方法 …… (150)
9.5　状态图建模案例分析 …… (154)
习题 9 …… (155)
第 10 章　活动图 …… (157)
10.1　活动图概述 …… (157)
10.2　活动图组成元素 …… (159)
10.3　活动的类型 …… (162)
10.4　使用 Rational Rose 建立活动图的方法 …… (163)
10.5　活动图建模案例分析 …… (168)
习题 10 …… (169)
第 11 章　包图 …… (171)
11.1　包图概述 …… (171)
11.2　包图的组成元素 …… (172)
11.3　包的嵌套 …… (176)
11.4　使用 Rational Rose 建立包图的方法 …… (176)
11.5　包图建模案例分析 …… (177)
习题 11 …… (179)
第 12 章　构件图 …… (180)
12.1　构件图概述 …… (180)
12.2　构件图的组成元素 …… (180)
12.3　使用 Rational Rose 建立构件图的方法 …… (182)
12.4　构件图建模案例分析 …… (186)
习题 12 …… (187)
第 13 章　部署图 …… (189)
13.1　部署图概述 …… (189)
13.2　部署图组成元素 …… (190)
13.3　使用 Rational Rose 建立部署图的方法 …… (190)
13.4　部署图建模案例分析 …… (193)
习题 13 …… (194)

第 14 章　双向工程 …… (196)

14.1　双向工程概述 …… (196)

14.2　双向工程案例实现 …… (201)

习题 14 …… (205)

第 15 章　项目案例综合实践 …… (206)

15.1　BBS 论坛系统 …… (206)

15.2　基于 Web 的求职招聘系统 …… (226)

参考文献 …… (239)

第1章 面向对象基础

20世纪60年代以来，面向对象的方法在计算机领域得到了广泛应用，如在程序设计中的面向对象程序设计、在人工智能中的面向对象知识表示、在数据库中的面向对象数据库、在人机界面中的面向对象图形用户界面、在计算机体系结构中的面向对象结构体系等。面向对象技术现在已经逐渐取代了传统的技术，成为当今计算机软件工程学中的主要开发技术，随着面向对象技术的不断发展，越来越多的软件开发人员加入到了它的阵营之中。面向对象的开发方法是一种将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，它是建立在对象概念基础上的方法。面对对象技术是软件技术的一次革命，在软件开发史上具有里程碑的意义。面向对象的基本思想是：对问题空间进行自然分割，以符合人的思维方式去建立问题域模型，以便对客观实体进行结构模拟和行为模拟，从而尽可能直接地描述现实世界，构造出模块化的、可重用的、可扩展的软件产品，同时限定软件的复杂性和减少软件维护的代价。

1.1 面向对象的概念

面向对象概念是在20世纪60年代由使用SIMULA语言的人开始提出的，于20世纪70年代初成为Xerox PARC开发的Smalltalk的重要组成部分。20世纪80年代面向对象方法与技术日益受到计算机领域的专家、研究和工程技术人员的重视，之后相继出现了一系列描述能力强、执行效率高的面向对象编程语言，标志着面向对象技术开始走向实用。20世纪90年代人们对面向对象的研究不再局限于编程，而是从系统分析和系统设计阶段就开始采用面向对象方法，这标志着面向对象思想已经发展成一种完整的方法论和系统化的技术体系，下面介绍一些面向对象的基本概念。

1.1.1 对象

对象含义广泛，难以精确定义，在不同的场合有着不同的含义。一般来说，任何事物均可看成对象。任何事物均有各自的自然属性和行为，当考察其某些属性与行为并进行研究时，它便成为有意义的对象。采用面向对象方法进行软件开发时，需要区分三种不同含义的对象：客观对象、问题对象和计算机对象。客观对象是现实世界中存在的实体；问题对象是客观对象在问题域中的抽象，用于根据需要完成某些行为；计算机对象是问题对象在计算机系统中的表示，它是数据和操作的封装体。三种对象间的关系如图1-1所示。

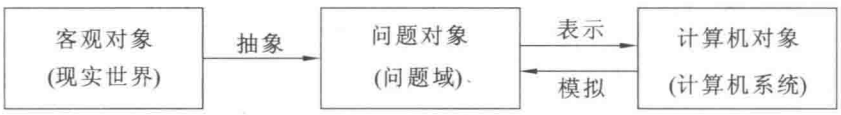


图1-1 三种对象间的关系

对象的表示应包括属性与行为(如数据与操作)，并且对象之间并非彼此孤立，可以通过通信来进行交互。因此，计算机对象可以表示为一个三元组：对象=(接口，数据，操作)，即对象是面向对象系统中运行时刻的基本成分，它是数据和操作的封装体，其中还包括与其他

对象进行通信的设施。

我们可以从以下几个不同的角度来考察对象的概念。

● 首先，从宏观上看，对象是客观对象在计算机中的表示。计算机对象是用来模拟现实世界中的客观对象的，模拟的重点是其中的信息处理和传递。客观对象自身的信息处理过程是由对象内部状态的变化来模拟的，客观对象间的信息传递则是由对象间的通信来模拟的。

● 其次，从微观上看，对象是由能对外通信的数据及其上的操作组成的封装体。对象由一组数据及其上的操作组成，并且能接收其他对象发送的消息，以及向其他对象发送消息。如果把模块间的相互过程调用也看成一种模块间的通信，就可以把对象近似地理解成模块的运行实例。不过，二者并不完全相同，模块中的数据可以移出，从而对这部分数据不加隐蔽，而对象中的数据则全是私有的。

● 最后，从形式描述上看，对象是具有输入和输出的有限自动机。对象是一个通信自动机，即以其他自动机的输出作为输入，以自己的输出作为其他自动机的输入。这种自动机的输入/输出方式反映了对象的通信能力；而把对象看成某种自动机则反映了对象自身具有独立的计算能力，体现为状态及状态转换机制。

上述对象概念的描述中，可以归纳出对象的特点，具体如下。

(1) 自治性：对象的自治性是指对象具有一定的独立计算能力。给定一些输入，经过状态转换，对象能产生输出，说明它具有计算能力。对象自身的状态变化是不直接受外界干预的，外界只有通过发送的消息对它产生影响，从这个意义上说，对象具有自治性。

(2) 封闭性：对象的封闭性是指对象具有信息隐蔽的能力。具体说来，外界不直接修改对象的状态，只有通过向该对象发送消息来对它施加影响。对象隐蔽了其中的数据及操作的实现方法，对外可见的只是该对象所提供的操作(即能接收和处理的信息)。

(3) 通信性：对象的通信性是指对象具有与其他对象通信的能力，也就是对象能接收其他对象发来的消息，同时也能向其他对象发送消息。通信性反映了不同对象间的联系，通过这种联系，若干对象可协同完成某项任务。

(4) 被动性：对象的被动性是指对象的存在和状态转换都是由来自外界的某种刺激引发的。对象的存在可以认为是由外界决定的，而对象的状态转换则是在它接收到某种消息后产生的，尽管这种转换实际是由其自身进行的。

(5) 暂存性：对象的暂存性有两层含义：一是指对象的存在是可以动态地引发的，而不是必须在计算的一开始就存在；二是指对象随时可以消亡，而不是必须存在到计算结束。虽然可以在计算过程中自始至终保存某些对象，但从对象的本质或作用来说，它具有暂存性。

上面五个性质分别刻画了对象的不同方面的特点。自治性、封闭性和通信性刻画的是对象的能力；被动性刻画的是对象的活动；暂存性刻画的是对象的生存特性；自治性反映了对象独立计算的能力；封闭性和通信性则说明对象是既封闭又开放的相对独立体。

1.1.2 类

类是具有相同属性和操作的一组对象的组合，也就是说，抽象模型中的类描述了一组相似对象的共同特征，为属于该类的全部对象提供了统一的抽象描述。例如，名为“学生”的类被用于描述为被学生管理系统管理的学生对象。

类的定义包含以下要素。

● 定义该类对象的数据结构(属性的名称和类型)。

此为试读，需要完整PDF请访问：www.ertongbook.com

● 类的对象在系统中所需要执行的各种操作，如对数据库的操作。

类是对象集合的再抽象，类与对象的关系如同用模具浇注出来的铸件一样，类是创建软件对象的模板——一种模型。类给出了属于该类的全部对象的抽象定义，而对象是符合这种定义的一个实体。

类的用途有如下两点。

● 在内存中开辟一个数据区，存储新对象的属性。

● 把一系列行为和对象关联起来。

一个对象又被称为类的一个实例，也称为实体化。术语“实体化”是指对象在类声明的基础上创建的过程。例如，我们声明了一个“学生”类，可以在这个基础上创建一个“姓名是李明的学生”的对象。

类的确定和划分没有统一的标准和方法，基本上依赖于设计人员的经验、技巧以及对实际项目中问题的把握。通常的标准是“寻求共性、抓住特性”，即在一个大的系统环境中，寻求事物的共性，将具有共性的事物用一个类进行表述。在用具体的程序实现时，具体到某一个对象，要抓住对象的特性。确定一个类的方法通常包含以下几个方面。

(1) 确定系统的范围，如学生管理系统，需要确定一下与学生管理相关的内容。

(2) 在系统范围内寻找对象，该对象通常具有一个或多个类似的事物。例如，在学生管理中，某院系有一个名叫李明的学生，而另一个院系名叫王鑫的学生是和李明类似的，都是学生。

(3) 将对象抽象成为一个类，按照上面类的定义，确定类的数据和操作。

在面向对象程序设计中，类和对象的确定非常重要，是软件开发的第一步，软件开发中类和对象的确定直接影响到软件的质量。如果划分得当，对于软件的维护与扩充以及体现软件的重用性等方面，都非常重要。

1.1.3 消息

对象是一个相对独立的具有一定计算能力的自治体，对象之间不是彼此孤立而是互相通信的，面向对象程序的执行体现为一组相互通信的对象的活动。那么面向对象程序是如何实施计算(运行)的呢？计算是由一组地位等同的称为对象的计算机制合作完成的，其合作方式是通信(即相互交换信息)，这种对象与对象之间所互相传递的信息称为消息。消息可以表示计算任务，也可以表示计算结果。

在面向对象计算中，每一项计算任务都表示为一个消息，实施计算任务的若干相关联的对象组成一个面向对象系统。提交计算任务即由任务提交者(如系统外对象)向承担计算任务的面向对象系统中的某个对象发送表示该计算任务的消息。计算的实施过程是面向对象系统接收到该消息后所产生的状态变化过程，计算的结果通过面向对象系统中的对象向任务提交者回送。

消息一般由如下三个部分组成。

(1) 接收消息的对象。

(2) 接收对象应采用的方法。

(3) 方法所需要的参数。

计算任务通常先由某一对象“受理”(该对象接收到某种消息)，然后通过对象间的通信，计算任务就分散到各个有关对象中，最后再由某些对象给出结果(通过发送消息)。发送消息的对象称为发送者，接收消息的对象称为接收者。消息中包含发送者的要求，它告诉接收

者需要完成哪些处理，但并不指示接收者如何完成这些处理。消息完全由接收者解析，接收者独立决定采用什么方式完成所需处理。一个对象能够接收不同形式、不同内容的多个消息，相同形式的消息可以发往不同的对象，不同的对象对于形式相同的消息可以有不同的解析，并作出不同的反应。对于传来的消息，对象可以返回相应的应答消息。

对象可以动态的创建，创建后即可以活动。对象在不同时刻可处于不同的状态，对象的活动是指对象状态的改变，它是由对象所接收的消息引发的。对象一经创建，就能接收消息，并向其他对象发送消息。对象接收到消息后，可能出现以下情况：①自身状态改变；②创建新对象；③向其他对象发送消息。

从对象之间的消息通信机制可反映出面向对象计算具有如下特性。

(1) 协同性　协同性表现在计算是由若干对象共同协作完成的。虽然计算任务可能首先由面向对象系统中的某个特定对象“受理”(即接收到表示该任务的消息)，但往往并不是由该对象独立完成的，而是通过对象间的通信被分解到其他有关对象中，由这些对象共同完成，对象间的这种协同性使计算具有分布性。

(2) 动态性　动态性表现在计算过程中对象依通信关系组成的结构会动态地改变，新对象会不断创建，旧对象也会不断消亡。面向对象系统最初由若干初始对象组成，一旦外界向这些初始对象发送了表示计算任务的消息，面向对象系统即活动起来直接给出计算结果。在此过程中，面向对象系统的组成因创建新对象而不断改变。

(3) 封闭性　封闭性表现在计算是由一组相对封闭的对象完成的。从外部来观察一个对象，它只是一个能接收和发送消息的机制，其内部的状态及其如何变化对外并不直接可见，外界只有通过给它发送消息才能对它产生影响。对象承担计算的能力完全通过它能接收和处理的消息体现。

(4) 自治性　自治性表现在计算是由一组自治的对象完成的。对象在接收了消息后，如何处理该消息(即自身状态如何改变，需创建哪些新对象，以及向其他对象发送什么消息)，完全由该对象自身决定。在面向对象计算中，数据与其上的操作地位同等，二者紧密耦合在一起形成对象，亦即数据及其上的操作构成对象。因此，在面向对象计算中，数据与其上的操作之间的联系处于首要地位，何时对哪些数据实行何种操作完全由相应数据所在对象所接收到的消息及该对象自身决定。由于对象的封装性和隐蔽性，对象的消息仅作用于对象的接口，通过接口进一步影响和改变对象状态。

1.1.4　封装

封装就是将对象的状态和行为捆绑在一起的机制，使对象形成一个独立的整体，并且尽可能地隐藏对象的内部细节。封装有两方面的含义：一是把对象的全部状态和行为结合在一起，形成一个不可分割的整体，对象的私有属性只能够由对象的行为来修改和读取；二是尽可能隐蔽对象的内容细节，与外界的联系只能够通过外部接口来实现。

封装的信息屏蔽作用反映了事物的相对独立性，我们可以只关心它对外所提供的接口，即能够提供什么样的服务，而不用去关注其内部的细节问题。例如，使用手机，我们关注的通常是这个手机能实现什么功能，而不太会去关心这个手机是怎么一步步制造出来的。

封装的结果使对象以外的部分不能随意更改对象的内部属性和状态，如果需要更改对象内部的属性和状态，则需要通过公共访问控制器来进行。通过公共访问控制器来限制对象的私有属性，有以下优点。

- 避免对封装数据的未授权访问。

● 当对象为维护一些信息，并且这些信息比较重要，不能够随便向外界传递的时候，只需要将这些信息属性设置为私有的即可。

● 帮助保护数据的完整性。

● 当对象的属性设置为公共访问的时候，代码可以不经过对象所属类希望遵循的业务流程而去修改对象的值，对象很容易失去对其数据的控制。我们可以通过访问控制器来修改私有属性的值，并且在赋值或取值的时候检查属性值的正确与否。

● 当类的私有方法必须修改时，限制了对整个应用程序内的影响。

当对象采用一个公共的属性去暴露的时候，我们知道，甚至修改一下这个公共属性的名称、程序都需要修改这个公共属性被调用的地方。但是，通过私有的方式就能够缩小其影响的范围，将程序的影响范围缩小到一个类中。

例如，房子就是一个类的实例，室内装饰和陈设只能由室内的居住者欣赏和使用，如果没有四周墙壁的遮挡，室内的所有活动在外人面前将一览无余。由于有了封装，房屋内的所有陈设都可以随意改变且不影响他人，然而，如果没有门窗，即使它的空间再宽阔，也没有实用的价值。房屋的门窗，就是封装对象暴露在外的属性和方法，专供人进出，以及空气流通之用。

但是在实际项目中，如果一味地的强调封装，对象的任何属性都不允许外部直接读取，反而会增加许多无意义的操作，为编程增加负担。为避免这一点，在语言的具体使用过程中，应该根据需要和具体情况，来决定对象属性的可见性。

1.1.5 继承

对于客观事物的认知，既应当看到其共性，也应当看到其特性。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，从而不能完整、正确地对客观世界进行抽象的描述。如果说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的话，那么在这个类的基础上，再重新考虑抽象过程中被舍弃的那一部分对象的特性，则可以形成一个新的类，这个类具有前一个类的全部特征，是前一个类的子集，从而形成一种层次结构，即继承结构。以动物为例，可以分为哺乳动物、爬行动物、两栖动物和鸟类等，通过抽象的方式实现一个动物类以后，可以通过继承的方式分别实现哺乳动物、爬行动物、两栖动物和鸟类等类，并且这些类包含动物的特性，如图 1-2 所示就展示了这样一个继承的结构。

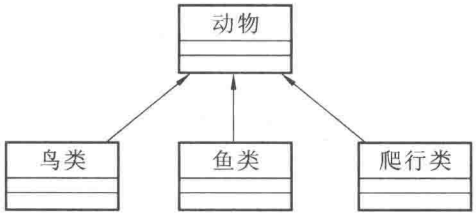


图 1-2 动物类继承结构

继承是一种连接类与类之间的层次模型。继承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”，即在特殊类中不必对已经在一般类中定义过的属性和行为进行重新定义，而是自动地、隐含地拥有其一般类的属性和行为。继承对类的重用性提供了一种明确表述共性的方法，即一个特殊类既有自己定义的属性和行为，又有继承下来的属性和行为。尽管继承下来的属性和行为在特殊类中是隐式的，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。继承是传递的，当这个特殊类被它更下层的特殊类继承的时候，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。有时把一般类称为基类，把特殊类称为派生类。

继承在面向对象软件开发过程中，有其强有力和独特的一面，通过继承可以实现以下几种功能。

● 使派生类能够比不使用继承直接进行描述的类更加简洁。派生类只需要描述那些与基类不相同的、特殊的地方，且把这些添加到类中然后继承就可以了。如果不使用继承而去直接描述，需要将基类的属性和行为全部描述一遍。

● 能够重用和扩展现有类库资源。当我们使用已经封装好的类库的时候，如果需要对某个类进行扩展，通过继承的方式很容易实现，而不需要再重新编写，并且扩展一个类的时候并不需要其源代码。

● 使软件易于维护和修改。当需要修改或增加某一属性或行为时，只需要在相应的类中进行改动，而它派生的所有类全部都将自动地、隐含地进行相应地修改。

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，缩短了开发的周期，提高了软件的开发效率，同时使软件易于维护和修改。继承是对客观世界的直接反映，通过类的继承，能够实现对问题深入抽象的描述，也反映出人类认识问题的发展过程。

1.1.6 多态

多态的一般含义是，某一领域中的元素可以有多种解释，程序设计语言中的一名多用即是支持多态的设施。继承机制是面向对象程序设计语言中所特有的另一种支持多态的机制。

在面向对象的软件技术中，多态是指在类继承层次中的类可以共享的一个行为的名字，而不同层次的类却各自按自己的需要实现这个行为。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地选择在该类中定义的行为实现。

在面向对象程序设计语言中，一个多态的对象指引变量可以在不同的时刻，指向不同类的实例。由于多态对象指引变量可以指向多类对象，所以它既有一个静态类型又有多个动态类型。多态对象指引变量的动态类型在程序执行中会经常改变，在强类型的面向对象环境中，运行系统自动为所有多态对象指引变量标记其动态类型。多态对象指引变量的静态类型由程序正文中的变量说明决定，它可以在编译时确定，它规定了运行时刻可接受的有效对象类型的集合，这种规定是通过对系统的继承关系图进行分析得到的。

具体到面向对象程序设计来说，多态性是指在两个或多个属于不同类的对象中，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，不同的对象可以有独特的表现方式，实现个性化的设计。

1.2 面向对象与面向过程的区别

在面向对象程序设计(object oriented programming，OOP)方法出现以前，结构化程序设计占据着主流。结构化程序设计是一种自上而下的设计方法，通常使用一个主函数来概括出整个程序需要做的事情，而主函数是由一系列子函数所组成。对于主函数中的每一个子函数，又都可以被分解为更小的函数。结构化程序设计思想就是将大的程序分解为具有层次结构的若干个模块，每个模块再分解为下一层模块，如此自顶向下、逐步细分，从而把复杂的大模块分解为许多功能单一的小模块。结构化程序设计的特征就是以函数为中心，也

就是以功能为中心来描述系统，用函数来作为划分程序的基本单位，数据在过程式设计中往往处于从属的位置。结构化程序设计的优点是易于理解和掌握，这种模块化、结构化、自顶向下与逐步求精的设计原则，与大多数人的思维和解决问题的方式比较接近。

然而，对于比较复杂的问题或在开发中需求变化比较多的情况下，结构化程序设计往往显得力不从心。这是因为结构化程序设计是自上而下的，这要求设计者在一开始就要对解决的问题有一定的了解。在问题比较复杂时，要做到这一点会比较困难，而当开发中的需求变化时，以前对问题的理解也许会变得不再适用。事实上，开发一个系统的过程往往也是一个对系统不断了解和学习的过程，而结构化程序设计方法忽略了这一点。结构化程序设计方法把密切相关、相互依赖的数据和对数据的操作相互分离，这种实质上的依赖与形式上的分离使得大型程序的编写比较困难，并难于调试和修改。在多人进行协同开发的项目组中，程序员之间很难读懂对方的代码，代码的重用变得十分困难。由于现代应用程序的规模越来越大，对代码的可重用性和易维护性的要求也越来越高，面向对象技术对以上问题提供了很好地支持。

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。它是一种自下而上的程序设计方法，它不像面向过程程序设计那样，一开始就需要使用一个主函数来概括整个程序，面向对象程序设计往往从问题的一部分着手，一点一点地构建出整个程序。面向对象设计是以数据为中心，使用类作为表现数据的工具，类是划分程序的基本单位。而函数在面向对象设计中成了类的接口，以数据为中心而不是以功能为中心来描述系统，相对来说，这样更能使程序具有稳定性。它将数据和对数据的操作封装到一起，这种作为一个整体进行处理并且采用数据抽象和信息隐藏技术最终被抽象成一种新的数据类型——类。类与类之间的联系以及类的重用使得类出现了继承、多态等特性。类的集成度越高，越适合大型应用程序的开发。另外，面向对象程序的控制流程运行是由事件进行驱动的，而不再由预定的顺序进行执行。事件驱动程序的执行围绕消息的产生与处理，靠消息的循环机制来实现。更加重要的是，其可以利用不断成熟的各种框架（如. Net 的. Net Framework 等），在实际的编程过程中迅速地将程序构建起来。面向对象的程序设计方法还能够使程序的结构清晰简单，从而大大提高代码的重用性，有效地减少程序的维护量，提高软件的开发效率。

在结构上，面向对象程序设计和结构化程序设计也有很大不同。结构化程序设计首先应该确定的是程序的流程怎么走、函数间的调用关系怎么样，以及函数间的依赖关系是什么。一个主函数依赖于其子函数，子函数又依赖于更小的子函数，而在程序中，越小的函数处理的往往是细节实现，具体的实现又常常变化。这种变化的结果就是程序的核心逻辑依赖于外延的细节，程序中本来应该是比较稳定的核心逻辑，也因为依赖于易变化的部分而变得不稳定起来，一个细节上的小改动也有可能在依赖关系上引发一系列变动。可以说这种依赖关系也是过程式设计不能很好地处理变化的原因之一，而一个合理的依赖关系应该由细节实现依赖于核心逻辑。面向对象程序设计由类的定义和类的使用两部分组成，主程序中定义对象并规定它们之间消息传递的方式，程序中的一切操作都是通过面向对象的发送消息机制来实现的。对象接收到消息后，启动消息处理函数完成相应的操作。

以图书管理系统为例，使用结构化程序设计的方法时，首先需要在主函数中确定图书管理系统要做哪些事情，并使用函数来表示这些事情进行表示，使用一个分支选择程序进行选择，然后将这些函数进行细化实现，并确定调用的流程等。使用面向对象技术来实现图书管理系统时，以学生为例，要了解图书管理系统中学生的主要属性（如学号、院系等）、学生要进

行什么操作(如借书、还书等)等,并且把这些当成一个整体进行对待,形成一个类,即学生类。使用这个类可以创建不同的学生实例,即创建许多具体的学生模型,每个学生拥有不同的学号,都可以在图书馆借书和还书。学生类中的数据和操作都是可以共享的,可以在学生类的基础上派生出专科生类、本科生类、研究生类等,从而实现代码的重用。

类与对象是面向对象程序设计中最基本和最重要的概念,也是创建和使用 UML 图的基础,有必要仔细理解和掌握,并且在学习中不断深化。

1.3 面向对象分析

众所周知,在解决问题之前必须理解索要解决的问题。对问题理解的越透彻,就越容易解决它。当完全、彻底的理解了一个问题的时候,通常就已经将问题解决了一大半。为了更好的理解问题,人们常常采用建立问题模型的方法。所谓模型,就是为了理解事物而对事物进行的一种抽象,是对事物的一种无歧义的书面描述。通常,模型是由一组图示符号和组织这些符号的规则组成,利用它们来定义和描述问题域中的术语和概念。更进一步来说,模型是一种思考工具,利用这种工具可以把知识规范地表示出来。模型可以帮助人们思考问题、定义术语,在选择术语时进行适当的假设,并且有助于保持定义和假设的一致性。

为了开发复杂的软件系统,系统分析员应该从不同角度抽象出目标系统的特性,使用精确的表示方法构造系统的模型,验证模型是否满足用户对目标系统的需求,并在设计过程中逐渐将与实现有关的细节加入模型中,直至最终用程序实现模型。对于那些因过分复杂而不能直接理解的系统,特别需要建立模型,建模的目的主要是减少复杂性。人的大脑每次只能处理一定数量的信息,模型通过将系统的重要部分分解成人的头脑一次能处理的若干个子部分,从而减少系统的复杂程度。在对目标系统进行分析的初始阶段,面对大量模糊的、涉及众多专业领域的、错综复杂的信息,系统分析员往往感到无从下手。模型提供了组织大量信息的一种有效机制,一旦建立起模型之后,这个模型就要经受用户和各个领域专家的严格审查。由于模型的规范化和系统化,比较容易暴露出系统分析员对目标系统认识的片面性和不一致性。通过审查,往往会发现许多错误。发现错误是正常现象,这些错误可以在成为目标系统中的错误之前,就被预定清除掉。通常,用户和领域专家可以通过快速建立的原型亲身体验,从而对系统模型进行更有效的审查。模型常常会经过很多次必要地修改,通过不断改正错误的或不全面的认识,最终使软件开发人员对问题有透彻的理解,从而为后续的开发工作奠定坚实的基础。

使用面向对象方法成功开发软件的关键,同样是对问题域的理解。面向对象方法最基本的原则,是按照人们习惯的思维方式,用面向对象观点建立问题域的模型,开发出尽可能自然地表现求解方法的软件。使用面向对象方法开发软件,通常需要建立三种形式的模型,分别是描述系统数据结构的对象模型,描述系统控制结构的动态模型和描述系统功能的功能模型。这三种模型都涉及数据、控制和操作等共同的概念,只不过每种模型描述的侧重点不同。这三种模型从三个不同但又密切相关的角度模拟目标系统,它们各自从不同侧面反映了系统的实质性内容,综合起来则全面地反映了对目标系统的需求。一个典型的软件系统综合了上述三方面的内容,即它使用数据结构(对象模型),执行操作(动态模型),并且完成数据值的变化(功能模型)。为了全面地理解问题域,对任何大系统来说,上述三种模型都是必不可少的。当然,在不同的应用问题中,这三种模型的相对重要程度会有所不同。但是,用面向对象方法开发软件,在任何情况下,对象模型始终都是最重要、最基本、最核心的。

此为试读,需要完整PDF请访问: www.ertongbook.com