21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

C#面向对象程序设计

OOP With C#

郑宇军 编著

- 深入讲解面向对象程序设计的理论、思想和方法
- 培养学生运用语言和面向对象思维解决实际问题
- 一个“旅行社管理系统”案例贯穿全书

精品系列

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

C#面向对象程序设计

OOP With C#

郑宇军 编著

精品系列

人民邮电出版社

北京

图书在版编目（CIP）数据

C#面向对象程序设计／郑宇军编著. —北京：人民邮电出版社，2009.6
21世纪高等学校计算机规划教材
ISBN 978-7-115-20656-5

Ⅰ. C… Ⅱ. 郑… Ⅲ. C语言－程序设计－高等学校－教材 Ⅳ. TP312

中国版本图书馆CIP数据核字（2009）第048915号

内 容 提 要

本书以面向对象的软件工程思想为主线，细致深入地讲解了C#语言面向对象程序设计的方法和技巧，内容涵盖面向对象的基本概念、基于接口的设计、泛型程序设计方法、Windows 和 Web 应用开发，以及数据库访问技术。全书提供了丰富的示例代码和课后习题，并通过一个贯穿全书的“旅行社管理系统”案例展现了如何运用C#语言和面向对象技术来进行实际软件系统开发。

本书适合作为高等院校计算机及相关专业教材，也可供专业开发人员自学参考。示例源代码和教学课件可在人民邮电出版社教学服务与资源网（http://www.ptpedu.com.cn）上下载。

21世纪高等学校计算机规划教材

C#面向对象程序设计

◆ 编　　著　郑宇军
责任编辑　刘　博
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷
◆ 开本：787×1092　1/16
印张：24.75
字数：648千字　　2009年6月第1版
印数：1－3 000册　　2009年6月北京第1次印刷

ISBN 978-7-115-20656-5/TP

定价：38.00元

读者服务热线：(010)67170985　印装质量热线：(010)67129223

反盗版热线：(010)67171154

出版者的话

计算机应用能力已经成为社会各行业从业人员最重要的工作技能要求之一，而计算机教材质量的好坏会直接影响人才素质的培养。目前，计算机教材出版市场百花争艳，品种急剧增多，要从林林总总的教材中挑选一本适合课程设置要求、满足教学实际需要的教材，难度越来越大。

人民邮电出版社作为一家以计算机、通信、电子信息类图书与教材出版为主的科技教育类出版社，在计算机教材领域已经出版了多套计算机系列教材。在各套系列教材中涌现出了一批被广大一线授课教师选用、深受广大师生好评的优秀教材。老师们希望我社能有更多的优秀教材集中地呈现在老师和读者面前，为此我社组织了这套“21 世纪高等学校计算机规划教材——精品系列”。

本套教材具有下列特点。

（1）前期调研充分，适合实际教学需要。本套教材主要面向普通本科院校的学生编写，在内容深度、系统结构、案例选择、编写方法等方面进行了深入细致的调研，目的是在教材编写之前充分了解实际教学的需要。

（2）编写目标明确，读者对象针对性强。每一本教材在编写之前都明确了该教材的读者对象和适用范围，即明确面向的读者是计算机专业、非计算机理工类专业还是文科类专业的学生，尽量符合目前普通高等教育计算机课程的教学计划、教学大纲以及发展趋势。

（3）精选作者，保证质量。本套教材的作者，既有来自院校的一线授课老师，也有来自 IT 企业、科研机构等单位的资深技术人员。通过他们的合作使老师丰富的实际教学经验与技术人员丰富的实践工程经验相融合，为广大师生编写出适合目前教学实际需求、满足学校新时期人才培养模式的高质量教材。

（4）一纲多本，适应面宽。在本套教材中，我们根据目前教学的实际情况，做到“一纲多本”，即根据院校已学课程和后续课程的不同开设情况，为同一科目提供不同类型的教材。

（5）突出能力培养，适应人才市场要求。本套教材贴近市场对于计算机人才的能力要求，注重理论知识与实际应用的结合，注重实际操作和实践动手能力的培养，为学生快速适应企业实际需求做好准备。

（6）配套服务完善。对于每一本教材，我们在教材出版的同时，都将提供完备的 PPT 课件，并根据需要提供书中的源程序代码、习题答案、教学大纲等内容，部分教材还将在作者的配合下，提供疑难解答、教学交流等服务。

在本套教材的策划组织过程中，我们获得了来自清华大学、北京大学、中国人民大学、浙江大学、吉林大学、武汉大学、哈尔滨工业大学、东南大学、四川大学、上海交通大学、西安交通大学、电子科技大学、西安电子科技大学、北京邮电大学、北京林业大学等院校老师的大力支持和帮助，同时获得了来自信息产业部电信研究院、联想、华为、中兴、同方、爱立信、摩托罗拉等企业和科研单位的领导或技术人员的积极配合。在此，向他们表示衷心的感谢。

我们相信，“21 世纪高等学校计算机规划教材——精品系列”一定能够为我国高等院校计算机教学做出应有的贡献。同时，对于工作欠缺和不妥之处，欢迎老师和读者提出宝贵的意见和建议。

前　言

从20世纪90年代开始，面向对象技术逐渐成为计算机软件开发的主流技术，Java、C++等支持面向对象的程序设计语言得到了广泛的应用，制造、能源、金融、电信、教育等各个行业和政府部门中都有大量面向对象的软件系统在发挥着至关重要的作用。

为了提高面向对象开发特别是企业级应用开发的生产率，Microsoft在2000年推出了C#语言，它具有简单易学、类型安全的特点，支持Windows、Web和数据库应用程序开发。在不到10年的时间，C#语言得到了快速发展和普及，特别是在.NET平台上已成为占据绝对优势地位的开发语言。

毋庸置疑，C#对于初学者来说是比较容易上手的；但要想成为一名专业的C#开发人员，还是应当脚踏实地地学好语言中的每个基本概念，再逐步学习和理解更为深奥的知识，并在日常练习和实际应用开发中贯彻所学的思想和方法，不断提高分析和解决问题的能力。我也经常这样勉励学生：要使你在听完客户的需求描述之后，头脑中就能浮现出目标软件的基本框架、关键问题的解决思路，还有那些已经烂熟于心的代码段。

本书以面向对象的软件工程思想为主线，采用对象先行的教学方式，循序渐进地讲解了C#语言面向对象程序设计的基本概念和方法，使学生掌握.NET应用软件开发的基本技术，逐步养成抽象的编程思维和良好的编程风格，为成长为一名高层次的计算机软件专业人才打下夯实的基础。

全书共15章，从知识组织的框架考虑可分为以下几个部分。

第1章介绍了面向对象程序设计的基本概念，使学生初步掌握面向对象的思维方式、指导原则和开发过程。

第2章和第3章介绍了C#程序的基本结构和语法基础，使学生快速入门，能够使用Visual Studio开发简单的应用程序。

第4～6章逐步深入地讲解了C#面向对象程序设计的基本要素，使学生能够充分理解并应用C#语言来实现软件的抽象、封装、继承和多态性。

第7～9章依次介绍了使用C#进行Windows程序界面设计、文件I/O操作以及程序异常处理，帮助学生掌握一般商业应用软件开发的基本技术。

第10章介绍了C#接口程序设计技术，第11章和第12章介绍了泛型程序设计方法，它们是当代面向对象软件技术的重要扩展，也是面向对象软件开发人员加强专业素质的重要方面。

第13章介绍了Windows程序设计的高级主题，第14章和第15章则分别介绍了使用C#进行ASP.NET应用程序和数据库应用程序开发的基础知识。学习完这些内容后，学生应初步具有一名现代软件开发人员的基本能力和素质。

书中每一章都对学习的重点和难点进行了总结，并提供了课后习题来帮助学生温习和提高。

本书还专门以一个“旅行社管理系统”的设计开发作为贯穿始终的学习案例，生动形象地展现了如何运用C#语言和面向对象技术来解决实际系统开发中遇到的问题，使得理论知识讲解更加贴近实际应用需求，特别是使学生对“什么是真正面向对象的软件系统”有一个更为清晰而完整的认识。

使用本书作为高等院校程序设计课程的教材时，计算机专业以及信息管理、电子商务、工业工程等非计算机专业可分别参考下面的建议学时来进行教学和上机实践安排。

章　节	计算机软件及相关专业	非计算机专业
第1章	2	2
第2章	2+1	1+1
第3章	4+5	3+4
第4章	4+4	4+3
第5章	3+3	2+3
第6章	2+2	2+1
第7章	3+4	2+4
第8章	2+4	2+3
第9章	2+1	1+1
第10章	3+2	2+2
第11章	6+4	4+2
第12章	2+2	2+1
第13章	2+3	2+2
第14章	5+6	4+6
第15章	3+4	3+3
合计	45+45	36+36

本书主要基于C#2.0版本进行讲解，个别章节也简要介绍了C#3.0的相关特性。书中的示例程序以及“旅行社管理系统”案例均可在.NET Framework 2.0上编译运行，不过我们推荐使用Visual C# 2005 Express、Visual Studio 2005或更高版本作为开发工具。源代码及教学课件等相关配套资源可登录人民邮电出版社教学服务与资源网（http://www.ptpedu.com.cn）下载。

在本书的编写和教学实践过程中，杨军伟、郑艳华、王侃、宋琴等参与了部分文字整理和程序调试的工作。在此谨对每一位关心和支持本书编写工作的人表示衷心的谢意。因时间仓促，书中的不足与疏漏之处在所难免，恳请广大读者不吝指教，以使本书能够日臻完善。有关问题、意见和建议可发送邮件（主题请注明“CSharp程序设计”）到bookzheng@yeah.net。

编　者

2009年2月

目 录

第 1 章 面向对象程序设计概述

本章将简要地回顾一下计算机程序设计语言和软件开发方法的发展历程，并由此引出面向对象的基本思想、概念和方法。掌握这些基本知识将为深入学习 C#面向对象程序设计打下良好的基础。

1.1 计算机程序设计语言

人类使用自然语言，而计算机最终执行的是机器指令；作为人和计算机之间进行交流的工具，程序设计语言定义了一套代码规则，程序设计人员遵循这些规则所编写出来的程序可被翻译成计算机能够“理解”的形式。

程序设计语言可以分为低级语言和高级语言。低级语言包括机器语言和汇编语言，使用它们进行编程需要对机器结构有较多的了解，编写的代码晦涩难懂，不利于人们的理解和交流。高级语言则更加接近自然语言，比较符合人们的思维方式，因此大大提高了程序设计的效率，并使得人们通过阅读程序文本来理解计算过程成为可能。高级语言程序在计算机上有两种处理方式：一是由专门的解释程序来直接解释执行高级语言代码，二是由专门的编译程序将其翻译为低级语言代码而后执行。目前在程序设计的各个领域中，高级语言已基本上取代了低级语言。

Fortran 语言是第一个被大规模推广使用的高级语言，其程序由一个主程序和若干个子程序组成，通过将不同的功能分配到独立的子程序中，能够有效地实现程序的模块化。20 世纪七八十年代非常流行的 Pascal 语言则提供了丰富的数据类型和强有力的控制结构，使用它能够方便地编写结构化的应用程序，有效避免了滥用 goto 语句所带来的危害；其程序结构中的一个模块就是一个过程，因此也被称为面向过程的语言。当然，最为流行的结构化程序设计语言莫过于 C 语言，它兼顾了诸多高级语言的特点，具有丰富的数据结构和控制结构，同时还提供了指针和地址等低级操作的能力，因此既适合于开发应用程序，又适合于开发系统程序，此外，它还有良好的可移植性，成为程序设计语言诞生以来最为成功的范例之一。

简而言之，结构化程序设计采用自顶向下、分而治之的方法，对目标系统进行功能抽象和逐步分解，直至每个功能模块都能以一个过程或函数来实现为止。这样就将复杂系统划分为一系列易于控制和处理的软件模块，其特点是结构良好，条理清晰，功能明确。对于需求稳定、算法密集型的领域（如科学计算领域），上述方法是有效和适用的。

随着信息技术的飞速发展，计算机软件也从单纯的科学和工程计算渗透到社会生活的方方面

面，软件的规模也越来越大，复杂性急剧提高，此时结构化方法逐步暴露出诸多问题和缺陷，这主要体现在以下几个方面。

- 功能与数据相分离，使用的建模概念不能直接映射到问题域中的对象，不符合人们对现实世界的认识和思维方式。
- 自顶向下的设计方法限制了软件模块的可复用性，降低了开发效率。
- 当系统需求发生变化时，系统结构往往需要大幅调整，特别是对于较复杂的系统，维护和扩展都会变得非常困难。

为了解决上述问题，一种全新的、强有力的软件开发方法——面向对象（Object-Oriented，OO）的方法应运而生。它是在结构化等传统方法的基础上发展起来的，也使用抽象和模块化等概念。和传统方法相比，其根本性的变化在于：不再将软件系统看成是工作在数据上的一系列过程或函数的集合，而是一系列相互协作而又彼此独立的对象的集合。这不仅更为符合人们的思维习惯，有助于保持问题空间和解空间在结构上的一致性，同时能够有效控制程序的复杂性，提高软件的生产效率。近二十年来，面向对象的方法学得到了迅速发展和广泛应用，Java、C++、C#等面向对象的程序设计语言也成为当今世界上计算机软件的主流开发语言。

1.2 面向对象的基本概念

本节介绍面向对象技术中一些最为基本的概念。

1.2.1 对象

客观世界中的事物都是对象（object），这既包括有形的物理对象（如一个人、一只狗、一本书等），也包括可感知的逻辑实体（如一个几何图形、一种化学元素等），还可以包括概念化的抽象实体（如一项计划、一种管理方式等）。对象通常有自己的属性（attribute），而且能够执行特定的操作（operation）。例如，一个人可以描述为“姓名——张三，身高——170，体重——65”，这里的“姓名”、“身高”、“体重”就是对象的属性，而“张三”、“170”、“65”则是对应的属性值；该对象还可以执行“走路”、“看书”等操作。属性用于描述对象的静态特征，而操作用于描述对象的动态特征。

在面向对象的模型中，软件对象就是对客观世界中对象的抽象描述，是构成软件系统的基本单位。但软件对象不应也不可能描述现实对象的全部信息，而只应包含那些与问题域有关的属性和操作。例如，在一个学籍管理系统中，通常会关心每个“学生”对象的“姓名”、“学号”、“专业”等属性信息，而他们的“发型”、“鞋号”等信息则不属考虑范围。

1.2.2 类

类（class）是指具有相同属性和操作的一组对象的集合；它描述的不是单个对象，而是“一类”对象的共同特征。例如，在学籍管理系统中可以定义“学生”类，而“张三”、“李明”、“王娟”这些学生就是属于该类的对象，或者叫作类的实例（instance）；它们都具有该类的属性和操作，但每个对象的属性值可以各不相同。

类是面向对象技术中最重要的结构，它支持信息隐藏和封装，进而支持对抽象数据类型（Abstract Data Type，ADT）的实现。信息隐藏是指对象的私有信息（包括属性和操作）不能由外

界直接访问，而只能通过该对象公开的操作来间接访问，这有助于提高程序的可靠性和安全性。例如，“学生”类可以有“生日”和“年龄”这两个属性，那么可以把它们都定义为私有的，不允许直接修改；再定义一个根据生日计算年龄的私有操作，以及一个修改生日的公共操作，这样用户就只能通过对象的生日来间接修改其年龄，从而保证年龄和生日的合法性。

类将数据和数据上的操作封装为一个有机的整体，类的用户只关心其提供的服务，而不必了解其内部实现细节。例如，对于“借书证”类的“刷卡”操作，用户可以只关注该操作返回的借书人信息（如借书权限、是否欠费等），而不去关心系统内部究竟是怎样判断磁条和借书人的对应关系的。

1.2.3　消息和通信

对象具有自治性和独立性，它们之间通过消息（message）进行通信，这也是对客观世界的形象模拟，此时发送消息的对象叫作客户（client），而接收消息的对象叫作服务器（server）。按照封装原则，对象总是通过公开其某些操作来向外界提供服务；如果某客户要请求其服务，那么就需要向服务器对象发送消息，而且消息的格式必须符合约定要求。消息中至少应指定要请求的服务（操作）名，必要时还应提供输入参数和输出参数。例如，某“学生”对象需要办理借书证，那么就要请求“图书馆”对象的“办理图书证”服务，并在消息中提供自己的姓名、学号等消息；“图书馆”对象检查这些消息合格后，创建一个新的“借书证”对象并返回给该学生。

1.2.4　关系

在很多情况下，单个对象是没有作用的。例如，“轮子”、“车厢”、“发动机”等对象单独放在那里都没有什么意义，而只有将它们组成一个“汽车”对象才能发挥各自的作用。再如，“学生”对象也不能是孤立的，而是要和“班级”、“老师”、“考试”等对象进行交互才能有意义。对象之间的关系可在类级别上进行概括描述，典型的有以下几种。

- 聚合（aggregation）：一个对象由其他对象作为其构成部分，也叫整体-部分关系，如“汽车”与“轮子”、“车厢”的关系，“班级”与“老师”、“学生”的关系等。
- 依赖（dependency）：一个对象对另一个对象存在依赖关系，且对后者的改变可能会影响到前者，如“借书证”对象依赖于某个“学生”对象；“学生”对象不存在了（例如该学生毕业或退学），相应的“借书证”对象也应被销毁。
- 泛化（generalization）：对象所属的类直接的一般-特殊关系，特殊类表示对一般类内涵的进一步细化，如“学生”类可进一步细化为特殊的“本科生”和“研究生”类。从泛化关系可以引出面向对象中另一个极为重要的概念——继承，这将在下一小节详细描述。
- 一般关联（association）：对象之间在物理或逻辑上更为一般的关联关系，主要是指一个对象使用另一个对象的服务，如“老师”和“学生”之间的教学关系。根据语义还可将关联关系分为多元关联和二元关联，二元关联还可进一步细分为一对一关联、一对多关联以及多对多关联等。聚合和依赖有时也被视为特殊的关联。

图 1-1 给出了上述 4 种关系的简单示例。

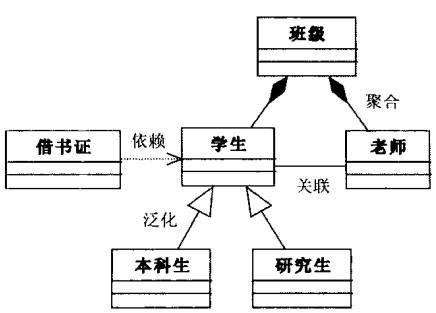

图 1-1　聚合、依赖、泛化和关联关系示例

1.2.5 继承

在泛化关系中，特殊类可自动具有一般类的属性和操作，这叫作继承（inheritance）；而特殊类还可以定义自己的属性和操作，从而对一般类的功能进行扩充。例如，“学生”类可以从“人”这个类中继承，这样就继承了“人”的“姓名”、“身高”等属性，而“学号”、“专业”等则是“学生”类自己特有的属性。在类的继承结构中，一般类也叫作基类或父类，特殊类也叫作派生类或子类。

继承的概念是从生物学中借鉴而来的，它可以具有多层结构。例如，动物可分为脊椎动物和无脊椎动物，脊椎动物又可分为哺乳动物、鱼类、鸟类、爬行动物和两栖动物……在分类过程中，低级别的类型总是继承了高级别类型的基本特征。图 1-2 示意了这样一个简单的动物继承关系。不过，在实际软件系统的建模过程中，继承的层次结构不宜过细、过深，否则会增加理解和修改的难度。

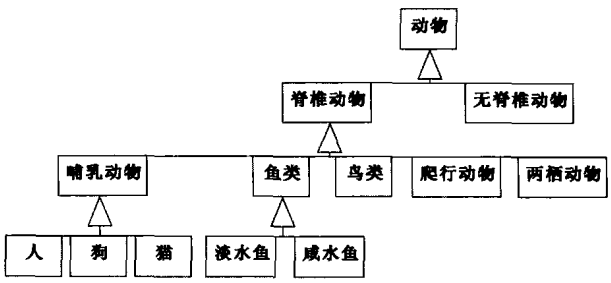

图 1-2 动物的继承关系图

继承具有可传递性。例如，“学生”类从“人”类继承而来，“研究生”类再从“学生”类继承而来，那么“本科生”和“研究生”类也就自动继承了“人”的“姓名”、“身高”等属性。这样派生类就能够享受其各级基类所提供的服务，从而实现高度的可复用性；当基类的某项功能发生变化时，对它的修改会自动反映到各个派生类中，这也提高了软件的可维护性。

自然界中还存在一种多继承的形式，例如，鸭嘴兽既有鸟类的特征，又有哺乳动物的特征，那么可以把它看成是鸟类和哺乳动物共同的派生类；再如，一名在职研究生可能同时具有老师和学生的身份。在面向对象的软件开发中，多继承具有较大的灵活性，但同时也会带来语义冲突、程序结构混乱等问题。目前 C++和 Eiffel 等语言支持多继承，而 Java 和 C#等则不支持，但它们可通过接口等技术来间接地实现多继承的功能。

1.2.6 多态性

多态性（polymorphism）是指同一事物在不同的条件下可以表现出不同的形态。在面向对象的消息通信时，发送消息的对象并不一定要知道接收消息的对象属于哪一类；接收到消息之后，不同类型的对象可以作出不同的解释，执行不同的操作，从而产生不同的结果。例如，学籍管理系统在每学期开始时会要求每个学生对象执行“选课”操作，但系统在发送消息时并不需要区分学生的具体类型是本科生还是研究生，而不同类型的学生会自行确定自己的选课范围，因为“本科生”类和“研究生”类会各自定义不同的“选课”操作。多态性特征能够帮助我们开发灵活且易于修改的程序。

1.2.7 接口和组件

随着软件规模和复杂度的不断增长，现代软件开发越来越强调接口（interface）和组件（component）技术，而接口也已成为面向对象不可或缺的重要元素。组件是指可以单独开发、测试和部署的软件模块，接口则是指对组件服务的抽象描述。一个组件中可以只有一个类，也可以包含多个类。

接口是一种抽象数据类型，它所描述的是功能的“契约”，而不考虑与实现有关的任何因素。例如，可以定义一个名为“图书借阅服务”的接口，并规定其中包括“图书目录查询”、“借书”和“还书”这 3 项功能。一个类如果声明支持某接口，它就必须支持接口契约中规定的全部功能。例如，“图书馆”类要声明支持“图书借阅服务”接口，它就至少应当提供“图书目录查询”、“借书”和“还书”3 项操作。

接口一旦发布，就不应再作修改，否则就会导致所有支持该接口的类型都变得无效。而组件一经发布，也不应取消它已声明支持的接口，而是只能增加新的接口。例如，一个“书店”类已经实现了“图书目录查询”操作，如果该类在升级时增加了“借书”和“还书”两项操作，它就可以声明支持“图书借阅服务”接口。

对于服务的使用方（客户）而言，它既不关心服务提供者的实际类型，也不关心实现服务的具体细节，而只需要根据接口去查询和使用服务即可。例如，读者可以向任何一个声明支持“图书借阅服务”接口的对象发送图书查询请求，并在查到自己所需图书后发送借阅请求，而不必考虑服务的提供者究竟是图书馆、书店还是别的什么机构。接口将功能契约从实现中完全抽象出来，能够有效地实现系统职责分离，同时弥补继承和多态性的功能不足，进而实现良好的系统设计。

1.3　面向对象的开发方法

从 20 世纪 80 年代开始，面向对象技术得到了飞速的发展，业界也涌现了一系列面向对象的软件开发方法学，其中代表性的有 Booch 方法、Wirfs-Brock 的责任驱动设计方法、Rambaugh 的对象模型技术、Coad/Yourdon 分析和设计方法、Jacobson 的面向对象软件工程方法等。为了避免不同符号所引起的混乱，Booch、Rambaugh 和 Jacobson 等共同参与制定了统一建模语言（Unified Modeling Language，UML），采用统一的概念和符号来描述对象模型，支持软件开发的全过程。目前 UML 已成为世界通用的面向对象建模语言，适用于各种开发方法。

1.3.1　面向对象的分析

面向对象的分析（Object-Oriented Analysis，OOA）就是运用面向对象的方法对目标系统进行分析和理解，找出描述问题域和系统责任所需要的对象，定义对象的基本框架（包括对象的属性、操作以及它们之间的关系），最后得到能够满足用户需求的系统分析模型。OOA 主要有以下 5 项任务。

（1）识别问题域中的对象和类。通过对问题域和系统责任的深入分析，尽可能地找出与应用有关的对象和类，并从中筛选出真正有用的对象和类。

（2）确定结构。找出对象和类中存在的各种整体-部分结构和一般-特殊结构，并进一步确定这些结构组合而成的多重结构。

（3）确定主题。如果系统包含大量的对象和类，那么可划分出不同的应用主题域，并按照主题域对分析模型进行分解。

（4）定义属性。识别各个对象的属性，确定其名称、类型和限制，并在此基础上找出对象之间的实例连接。

（5）定义服务。识别各个对象所提供的服务，确定其名称、功能和使用约定，并在此基础上

找出对象之间的消息连接。

OOA 的结果是系统分析说明书，其中包括使用类图和对象图等描述的系统静态模型，使用用例图、活动图和交互图等描述的系统动态模型，以及对象和类的规约描述。模型应尽量与问题域保持一致，而不考虑与目标系统实现有关的因素（如使用的编程语言、数据库平台和操作系统等）。

1.3.2 面向对象的设计

面向对象的设计（Object-Oriented Design，OOD）是以系统分析模型为基础，运用面向对象的方法进行系统设计，解决与系统实现有关的一系列问题，最后得到符合具体实现条件的系统设计模型。OOD 主要有以下 4 项任务。

（1）问题域设计。对问题域中的分析结果作进一步的细化、改进和增补，包括对模型中的对象和类、结构、属性、操作等进行组合和分解，并根据面向对象的设计原则增加必要的新元素类、属性和关系。这部分主要包括以下设计内容：

- 复用设计，即寻找可复用的类和设计模式，提高开发效率和质量；
- 考虑对象和类的共同特征，增加一般类，以建立共同协议；
- 调整继承结构，如减少继承层次、将多继承转换为单继承、调整多态性等；
- 改进性能，如分解复杂类、合并通信频繁的类、减少并发类等；
- 调整关联关系，如将多元关联转换为二元关联、将多对多关联转换为一对多关联等；
- 调整和完善属性，包括确定属性的类型、初始值和可访问性等；
- 构造和优化算法。

（2）用户界面设计。对软件系统的用户进行分析，对用户界面的表现形式和交互方式进行设计。这部分主要包括以下设计内容：

- 用户分类；
- 人机交互场景描述；
- 系统命令的组织；
- 详细的输入和输出设计；
- 在需要时可增加用于人机交互的对象和类，并使用面向对象的方法对其进行设计。

（3）任务管理设计。当系统中存在多任务（进程）并发行为时，需要定义、选择和调整这些任务，从而简化系统的控制结构。这部分主要包括以下设计内容：

- 识别系统任务，包括事件驱动任务、时钟驱动任务、优先任务和关键任务等；
- 确定任务之间的通信机制；
- 任务的协调和控制。

（4）数据管理设计。识别系统需要存储的数据内容和结构，确定对这些数据的访问和管理方法。这部分主要包括以下设计内容：

- 数据的存储方式设计，目前主要有文件系统和数据库系统两种方式；
- 永久性类（persistent class）的存储设计，包括其用于存储管理的属性和操作设计；
- 永久性类之间关系的存储设计。

OOA 和 OOD 之间不强调严格的阶段划分，设计模型是对分析模型的逐步细化，主要是在问题域和系统责任的分析基础上解决各种与实现有关的问题。OOA 阶段一些不能确定的问题可以遗留到 OOD 阶段解决，开发过程中也允许存在反复和迭代。