普.通.高.等.学.校
计算机教育“十二五”规划教材

C# 面向对象程序设计

（第2版）

OOP WITH C# (2nd edition)

郑宇军 ◆ 编著

人民邮电出版社
POSTS & TELECOM PRESS

普.通.高.等.学.校
计算机教育“十二五”规划教材

C# 面向对象程序设计

（第2版）

OOP WITH C# (2nd edition)

郑宇军 ◆ 编著

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. -- 2版. -- 北京 : 人民邮电出版社, 2013.7
普通高等学校计算机教育“十二五”规划教材
ISBN 978-7-115-29761-7

Ⅰ. ①C… Ⅱ. ①郑… Ⅲ. ①C语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2012)第284588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WPF窗体界面、文件和数据库访问，以及ASP.NET和Silverlight网站设计，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面向对象技术和C#语言来进行实际软件系统开发。全书提供了丰富的示例代码和课后习题。

本书适合作为高等院校计算机及相关专业的教材，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。

◆ 编　　著　郑宇军
责任编辑　刘　博
责任印制　彭志环　杨林杰

◆ 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　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
大厂聚鑫印刷有限责任公司印刷

◆ 开本：787×1092　1/16
印张：25.25　　2013年7月第2版
字数：697千字　　2013年7月河北第1次印刷

定价：49.80元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

第 2 版前言

自 2009 年 6 月本书第 1 版出版以来，颇受广大读者的欢迎，也有幸被许多老师选为程序设计专业教材。经过 3 年多的时间，编者结合教学实践和软件开发中的经验体会、C#语言的最新升级以及许多读者热情的反馈建议，对原书进行了系统的修订。

这次修订保留了原教材特点，坚持以面向对象的软件工程思想为主线、紧密贴合实际应用需求；同时增加了一些新的知识点，加强了教材内容的新颖性和趣味性，以便进一步提升教学效果。修改的主要方面包括：

（1）以 C# 4.0/4.5 版本为主进行讲解（但语言的核心要素并没有根本变化，新版本的优势主要体现在应用的多样性和开发的方便性上）。

（2）在第 6 章中增加了对 Lambda 表达式的介绍。

（3）将原来的第 7 章和第 13 章压缩合并在一起，在新的第 13 章中对 WPF（Windows Presentation Foundation）进行了较为细致的讲解。

（4）在第 15 章中新增了“LINQ 对象数据查询”一节，使读者能够对 LINQ 数据访问技术有一个初步的了解。

（5）新增了第 16 章“Silverlight 客户端应用程序”，对 Silverlight 富客户端开发技术进行了深入讲解。

本书配套的教学课件和案例程序代码也进行了相应的升级，有关内容可在人民邮电出版社网站上进行下载。

本书第 1 章由王侃执笔，第 2 章和第 15 章由杨军伟执笔，第 3 章～第 14 章和第 16 章由郑宇军执笔，吴晓蓓、凌海风、江勋林、郑艳华、宋琴等也参与了本书的部分文字编写工作。全书由郑宇军统稿。

此次编写工作得到了我校王卫红老师的悉心指导和帮助。在 C#程序设计课程教学实践中，与简琎峰老师和王松老师的经验交流使编者受益良多。同时，感谢我校计算机学院 09 级的徐静、张璐滢、何俊丽、王岳春、顾唯超、诸伊娜等同学，他们在学习开发过程中的创造性给了编者很多启发。

尽管我们做了最大的努力，书中的不足与疏漏之处仍在所难免，恳请广大读者批评指正。我们的 E-mail 地址是：bookzheng@yeah.net（邮件主题请注明“CSharp 程序设计”）。

编者

2012 年 8 月

于浙江工业大学　郁文楼

目 录

第 1 章 面向对象程序设计概述

本章简要地回顾了计算机程序设计语言和软件开发方法的发展历程，并由此引出了面向对象的基本思想、概念和方法。掌握这些基本知识能为深入学习 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，而计算机执行的是机器指令；程序设计语言是人与计算机之间进行交流的工具，它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深入的了解，而且代码晦涩难懂、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主要领域，高级语言已基本上取代了低级语言。

Fortran 语言是第一个被广泛使用的高级语言，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，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，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），上述方法是有效并适用的。

随着信息技术的飞速发展，计算机软件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：

- 功能与数据相分离，使用的建模概念不能直接映射到问题领域中的对象，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；但和传统方法相比，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了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），也包括抽象的逻辑对象（如一个几何图形、一项商业计划等）。对象一般都有自己的属性（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，或者叫做类的实例（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，从而保证其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），而不去管磁条中是怎样存储借书人的有关信息的。

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。发送消息的对象叫做客户（Client），而接收消息的对象叫做服务器（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）：一个对象是另一个对象的组成部分，也叫部分—整体关系，如“轮子”与“汽车”的关系，“学生”和“班级”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，当“学生”对象不存在了（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）：一个对象的类型是另一个对象类型的特例，也叫特殊— 一般关系，其中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方法中另一个重要概念——继承。
- 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关系。

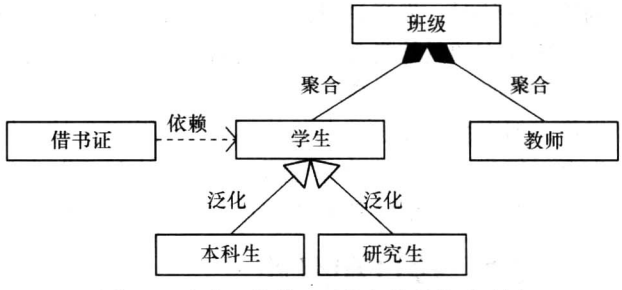

图 1-1　聚合、依赖、泛化和关联关系示例

上述 4 种关系的简单示例如图 1-1 所示。

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，这叫做继承（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、两栖动物等，这种划分可以持续很多层。在分类过程中，低级别的类型通常继承了高级别类型的基本特征。图 1-2 所示为这一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。

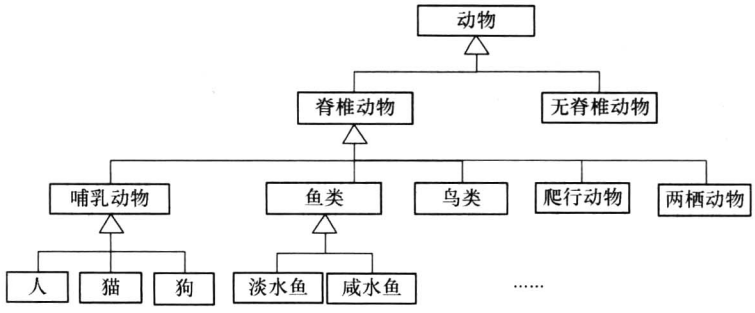

图 1-2　动物的继承关系图

继承具有可传递性。例如“学生”类从“人”类继承，“研究生”类再从“学生”类继承，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，对其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，发送消息的对象只需要确定接收消息的对象能够执行指定的操作，而并不一定要知道接收方的具体类型；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”接口，并规定其中包括“图书目录查询”、“借书”和“还书”这三项功能。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”接口，它就至少要为“图书目录查询”、“借书”和“还书”这三项操作提供具体的实现机制。

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“图书借阅”接口发布后，我们不能简单地试图为该接口增加一项“图书复印”功能，否则很多已支持该接口的类型就

会出错；更合理的方式是定义一个新的“图书复印”接口，那些具有复印能力的类型可以声明支持这个新接口，而不具备复印功能的类型则保持不变。

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”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 UML（Unified Modeling Language），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的建模语言，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。

1.4 案例研究——旅行社管理系统的分析与设计

旅行社管理系统是为旅行社提供业务支持的信息系统。旅行社设计旅游线路，并推出各种类型的旅行团。游客可查询旅游信息并报名参加旅行团，旅行社根据报名情况组织发团。通过简要的分析，可知系统的基本功能需求包括：

（1）旅行社设计旅游线路，每条旅游线路包含若干个旅游景点。

（2）旅行社基于旅游线路定制旅行团。

（3）游客上网查询旅游景点、线路和旅行团信息。

（4）游客报名参加旅行团，旅行社业务员负责接受或拒绝报名申请。

（5）满足条件后，旅行社为旅行团安排导游并发团。

在上述分析的基础上，可以找到一系列可能的对象，并将其抽象到不同的类。最为基本的类

有以下几个：

- 景点类 Scene，具有景点名称、位置、种类、票价等属性。
- 旅游线路类 Line，具有线路名称、景点集合、用时等属性。
- 旅行团类 Tour，具有旅游线路、时间、等级、人数、价格、导游、游客等属性。
- 游客类 Customer：具有游客姓名、身份证号、性别、生日等属性，还可以执行报名和取消报名等操作。

景点所在的位置应通过所属的省市来描述，那么需要定义 Province 和 City 类，它们和 Scene 类之间是聚合的关系。为了加强客户关系管理，旅行社还积极发展游客成为注册会员。为此可定义 Customer 的派生类 Member，它在 Customer 的基础上增加了用户名、密码、会员积分等属性。

一个成熟的旅行社通常会有多个固定的组团方案，如每天一次的北京一日游、每周一次的四川三日游等。那么可以把这些方案抽象为一个 Package 类，由该类维护旅游线路、等级、价格等信息，这样基于同一方案的多个旅行团就可以共享这些信息，而每个旅行团对象只需要关心自己的时间、导游、游客等独立信息。图 1-3 所示为这些类之间的基本关联关系。

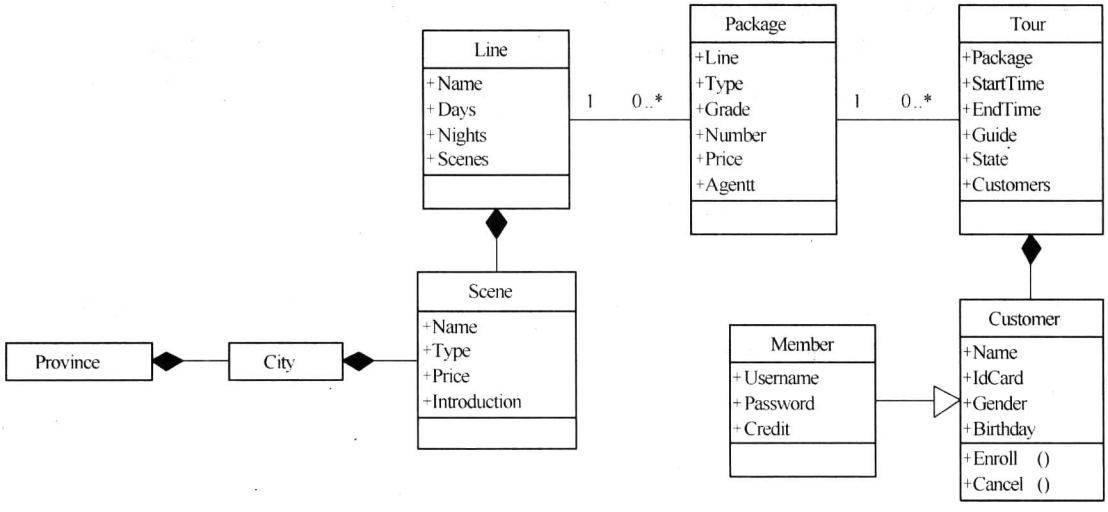

图 1-3　Scene、Line、Package、Tour、Customer 等主要业务类的关系图

系统还需要提供旅行社的内部管理功能，那么可定义旅行社类 TravelAgency，它具有名称、总经理、法人代表、通讯地址、电话等属性。为了对旅行社的职员进行管理，可将旅行社的所有职员抽象为一个 Staff 类，它具有姓名、身份证号、性别、年龄、学历等属性，不同类型的职员可定义为 Staff 的不同派生类，主要包括：

- 导游类 Guide，具有导游证号、导游证有效期等属性。
- 业务员类 Agent，负责处理游客报名、组团管理等操作。
- 主管类 Director，除处理旅行社业务外，还负责管理一般职员。
- 经理类 Manager，基本属性和 Director 相同，还可执行雇佣和解雇其他职员等操作。

另一些岗位（如前台、保安等）在系统中没有专门的功能，那么可作为一般 Staff 对待。系统所关心的职员类型的基本继承关系如图 1-4 所示。

上述类型都是旅行社管理系统的业务类，它们都不可避免地要使用到大量的通用数据类型，如整数、字符串、日期时间等，这大都可以在程序设计语言的类型系统中找到。系统所使用的用户界面中也会用到大量的控件类，如窗体、按钮、网页、图片等，这也主要由商业开发工具来提

供。当然，随着开发过程的深入，还可能发现更多需要的类型，如酒店、机场、车站等，这里可将它们先记录为候选类。

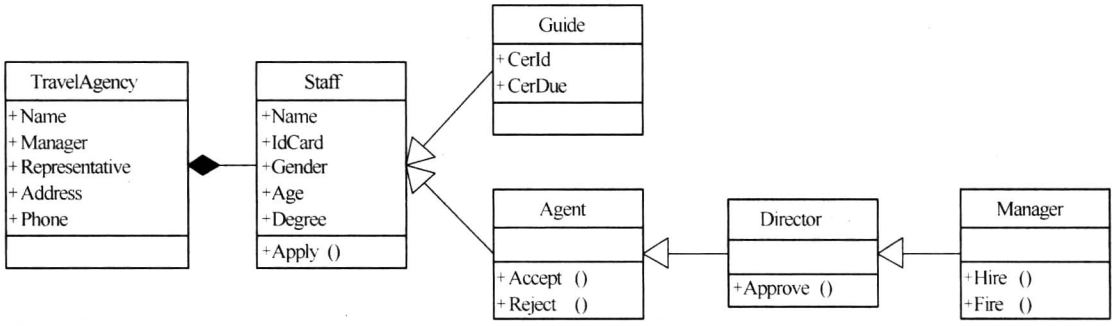

图 1-4　旅行社及其职员类的关系图

从设计角度出发，整个旅行社管理系统可划分为以下 3 部分。

- 业务类库：管理和维护上述基本业务类。
- 旅行社网站：供游客进行旅游资讯查询、旅行团网上报名等操作。
- 旅行社内部管理子系统：供旅行社职员进行线路、旅行团、游客等信息管理工作。

以上是旅行社管理系统的初步分析和设计结果，其具体内容将在本书后续章节中不断细化和完善。

1.5　小　　结

本章介绍了面向对象程序设计的基本概念，它将客观世界中的事物建模为软件对象，具有相同属性和操作的一组对象可抽象为类，类之间的关联和继承等关系是面向对象软件系统设计的关键。

1.6　习　　题

1. 简述对象和类的概念，并说说类在软件系统设计中的重要性。
2. 面向对象技术中的继承是指什么？试举例说明你在日常生活中看到的继承例子。
3. 将图 1-3 和图 1-4 合并成为一个更为完整的关系图。
4. 查阅有关统一建模语言 UML 的资料，初步了解 UML 与面向对象技术的关系。