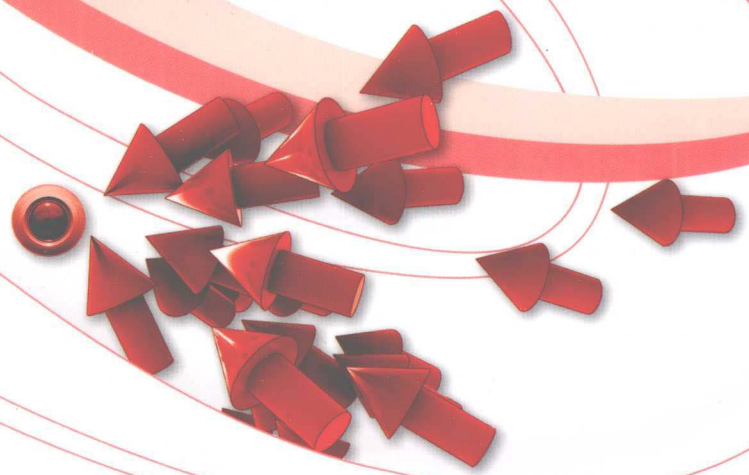
北京市高等教育精品教材立项项目



高等学校计算机规划教材

面向对象程序设计综合实践

骆力明　徐　敏　谭小慧　张汉煜　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

北京市高等教育精品教材立项项目

高等学校计算机规划教材

面向对象程序设计综合实践

骆力明　徐　敏

谭小慧　张汉煜

编著

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书是针对那些学习了C++语言并了解面向对象程序设计的基本方法的学习者，锻炼其使用面向对象的思想和方法对实际问题进行需求分析，根据需求分析结果完成程序的总体设计，在总体设计的基础上实现详细设计和编码调试，并对所设计的程序进行必要的测试。从而达到提高其程序设计的综合能力和训练其工程化软件开发的初级技能的目标。

本书第1章介绍了面向对象方法开发系统的基本步骤及每个步骤的大体工作，目的是让学生在学习完面向对象编程后，从软件工程角度建立软件开发的总体印象。第2章和第3章讲解了后续实验用到的一些技术。第4章通过“简单人事管理系统”的实践让学生具体掌握实验的实现步骤和方法。第5章描述了两个实验的要求及其实验开展建议。第6章和第7章给出了两个实验实现的参考解决方案，供学生参考。

本书可作为高等院校计算机相关专业的面向对象课程的实践课教材，也可作为希望尽快掌握面向对象开发技术和过程的技术人员的参考书。

图书在版编目(CIP)数据

面向对象程序设计综合实践 / 骆力明等编著. —北京：电子工业出版社，2011.1
(高等学校计算机规划教材)
ISBN 978-7-121-12581-2

I. ①面… II. ①骆… III. ①面向对象语言－程序设计－高等学校－教材②C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(2010)第247350号

策划编辑：冯小贝
责任编辑：李秦华
印　　刷：北京市李史山胶印厂
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编　100036
开　　本：787×1092　1/16　印张：15　字数：384千字
印　　次：2011年1月第1次印刷
印　　数：4000册　　定价：29.80元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010)88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：(010)88258888。

前　言

现代教育理论告诉我们，知识不是由教师灌输而得到的，而是在教师的指导和帮助下，由学生通过与周围学习环境进行交互，主动思考、主动探索而得到的。

毋庸置疑，作为信息类的本科学生，程序设计能力是应当具有的基本专业能力，对于软件工程、计算机科学与技术等专业更是应当重点学习和掌握的。但是，软件类课程是实践性很强的课程，仅仅通过阅读教科书或听课是不可能完全掌握的，学习程序设计，最重要环节就是实践。

学生在学习完C语言程序设计、数据结构和面向对象程序设计(C++)等课程后。为了使学生能使用面向对象的思想和方法对实际问题进行需求分析，根据需求分析结果完成程序的总体设计，并对所设计的程序进行测试。达到培养计算机软件本科学生程序设计的综合能力和训练工程化软件开发的初级技能的目标，我们编写了本书。

本书重点从以下角度组织内容

- 从建立面向对象分析、设计和编程方法的统一性出发，恰当地融入统一的建模语言——UML的概念，并借助于UML使学生了解软件建模的思想和方法。
- 以一个具体的、比较复杂的应用程序为依托仔细介绍面向对象软件开发的方法，并对开发过程中相应的技术难点和实现过程重点讲解。
- 根据所讲解的应用程序设计出若干个含有不同技术难点的实践题目。对每个选题的技术难点都给出了详细的提示和讲解。

本书的主要特点

- 以软件开发的规范分解案例，通过案例讲解每一过程的知识要求和工程规范。
- 选择具有实际锻炼价值的案例，突出重点知识的掌握和应用。
- 强调具体案例与思想方法的关系，在讲解案例的同时抽象出解决问题的方式方法。
- 实验题目强调一定的代码量，使学生完成从实验级代码量到项目开发的过渡。

本书可以作为本科生学习面向程序设计知识的实践类课程教材。使用本书，可以将整个课程分两个过程完成，第一个过程以教师为主体，教师和学生一起完成一个应用程序的所有开发过程，并在开发过程讲解相应的技术难点和实现过程。第二个过程以学生为主体，学生根据自己的兴趣和特长选择相应的题目分组独立完成整个应用程序的开发过程。教师一边指导学生开展项目，一边监督学生开展项目的进度和表现。

该书作为我校精品课程《程序设计综合实践》的教材使用了近5届学生，受到学生的普遍欢迎。

该书内容新颖，条理清楚，重点突出，实用性强，说理透彻，有比较强的实践意义，能满足信息类相关专业面向对象程序设计综合实践课程的教学要求。

本书是由骆力明、徐敏、谭小慧、张汉煜编著，但是本书的内容是由整个课程组老师们在长期教学和科研的工作中不断积累的基础上形成的，这里对所有对本书的编写有过帮助的同仁表示衷心感谢。

对于书中的疏漏和不足之处，恳请读者提出宝贵意见。

编著者

2010-11-19

目　录

第 1 章　面向对象软件开发方法

1.1　概述

计算机发展早期软件开发的个体化特点，造成了许多错误的认识和做法，表现为忽视软件需求和软件分析的重要性，认为软件开发就是写程序并使之运行。随着软件规模的增大，计算机软件开发和维护过程中遇到了一系列严重问题，出现了软件危机。

在软件开发工作中，以下几个观念常常被忽略。

① 软件的生命周期(参见图1.1)大体分为规划、分析、设计、编码、维护等一系列阶段组成。

在开发过程中，每个阶段都可以直接和间接地回馈到前面的阶段。整个软件生命周期是一个迭代、渐增的开发过程，这种迭代过程不仅贯穿整个软件生命周期，并且表　　现在每个阶段中，特别是在分析(全局分析、局部分析)和设计(全局设计、局部设计)　　阶段。

② 程序只是完整产品的一个组成部分，Boehm 对软件的定义是："软件是程序、数据和开发、使用和维护程序所需要的所有文档。"

③ 在软件开发不同阶段进行错误修改所付出的代价是很不相同的，在后期引入一个变动比在早期引入变动，代价高达 2～3 个数量级，所以做好软件定义时期的工作是降低软件成本，提高软件质量的关键。

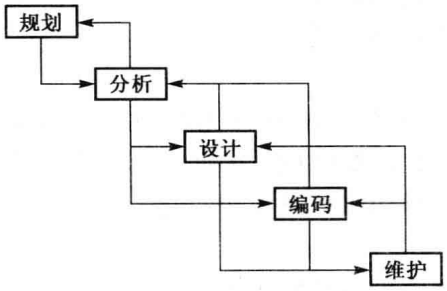


图 1.1　软件生命周期

④ 软件维护是极端艰巨复杂的工作，需要花费极大的代价，统计数据表明，软件维护费用占软件总费用的 55%～70%，所以要采取措施提高软件的可维护性，减少维护费用。

为了解决软件危机，必须要用现代工程的概念、原理、技术和方法进行软件开发、管理和维护。

软件开发不是某种个体劳动的神秘技巧，而应该是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目。另外，在软件开发的每个阶段都有许多烦琐重复的工作需要做，应当学会使用软件辅助工具，在适当的软件工具辅助下，开发人员可以把工作做得既快又好。

1.2 软件生命周期各阶段的基本任务

软件生命周期是指一个软件项目被提出并着手实施开始，到该软件报废或停止使用为止。软件生命周期由软件定义、软件开发和运行维护(也称为软件维护) 三个时期组成，每个时期又进一步划分成若干个阶段：

- 问题定义
- 可行性研究
- 需求分析
- 总体设计
- 详细设计
- 编码和单元测试
- 综合测试
- 运行与维护

(1) 问题定义

确定“要解决的问题是什么?”通过调研，写出关于问题性质、工程目标和工程规模的书面报告，并得到客户的确认。

(2) 可行性研究

确定对于问题定义阶段所确定的问题是否有行得通的解决方法。研究并论证软件系统的可行性，对方案进行选择并形成可行性分析报告。

(3) 需求分析

这个阶段的任务主要是确定所完成的系统必须具备哪些功能，并用正式文档准确地记录结果，这个文档通常称为系统规格说明书。

(4) 总体设计(概要设计)

软件设计的一条基本原理就是，程序应该模块化，也就是说，一个程序应该由若干个规模适中的模块按合理的层次结构组织而成。因此，总体设计的主要任务就是设计程序的体系结构，也就是确定程序由哪些模块组成及模块间的关系。

(5) 详细设计(模块设计)

将解法具体化，确定应该怎样具体地实现这个系统。主要工作是模块详细设计。模块详细设计包括：模块的详细功能、算法、数据结构、模块间的接口等设计，拟订模块测试方案。详细设计的工作体现在详细规格说明书。

(6) 编码和单元测试

根据模块详细规格说明书，把详细设计的结果翻译成用选定的语言书写的程序。还要对模块程序进行测试，验证模块功能及接口与详细设计文档的一致性，并形成单元测试报告。

(7) 综合测试

通过各种类型的测试(及相应的调试)使软件达到预定的要求。

- 集成测试：根据设计的软件结构，把经过单元测试检验的模块按某种选定的策略装配起来，在装配过程中对程序进行必要的测试。

- 验收测试：按照规格说明书的规定，由用户对目标系统进行验收。
- 现场测试或平行运行：平行运行就是同时运行新开发出来的系统和将被它取代的旧系统，以便比较新旧两个系统的处理结果。

用正式的文档资料把测试计划、详细测试方案以及实际测试结果保存下来，作为软件配置的一个组成部分。

(8) 运行与维护

维护阶段的关键任务是，通过各种必要的维护活动使系统持久地满足用户的需要。

- 改正性维护：诊断和改正在使用过程中发现的软件错误。
- 适应性维护：修改软件以适应环境的变化。
- 完善性维护：根据用户的要求改进或扩充软件使它更完善。
- 预防性维护：修改软件为将来的维护活动预先做准备。

每项维护活动都应该经过提出维护要求(或报告问题)，分析维护要求，提出维护方案，审批维护方案，确定维护计划，修改软件设计，修改程序，测试程序，复查验收等一系列步骤。

使用面向对象方法解决问题的过程可以大体划分为面向对象分析(Object Oriented Analysis，OOA)、面向对象设计(Object Oriented Design，OOD)、面向对象编程(Object Oriented Programming，OOP)和面向对象测试(Object Oriented Test，OOT)等步骤。

面向对象分析的主要作用是明确使用程序的用户、用户可以进行的操作，以及数据的输入、输出和储存，并且用标准化的面向对象模型规范地表述这些内容，最后形成面向对象分析模型，即 OOA 模型。在分析问题时，要抽取所有需要的对象实体，然后确定这些对象的状态和行为，以及它们之间的相互关系。一般来说，解决一个问题会涉及多个对象，所以这些对象之间的关系一定要明确，从而反映出整个程序的功能和状态。

面向对象设计是将在面向对象分析步骤中创建的 OOA 模型加以扩展并得到面向对象设计步骤中的 OOD 模型。面向对象设计在 OOA 模型的基础上引入界面管理、任务管理和数据管理三部分的内容，进一步扩充 OOA 模型。界面管理负责整个系统的人机对话界面的设计，任务管理负责处理整个程序资源管理功能的工作及设置客户与服务器之间的接口，数据管理负责设计程序与数据库的交换方式。面向对象设计还需要明确每个类方法的参数、返回值、功能等，以及各类之间的相容性和一致性的验证，对各个类、类内成员的访问权限的严格合理性的验证，也包括验证对象类的功能是否符合用户的需求。

面向对象编程就是具体的程序编写阶段，其主要过程是先选择一种合适的面向对象编程语言，再用选定的语言编写程序实现设计步骤中对各个对象的详尽描述，然后将编写好的各个类根据其关系集成为整个程序，最后通过各种实例测试找出程序的漏洞并改善程序，最终完成整个软件的开发。

1.3　面向对象分析

1.3.1　确定客户需要什么

人们通常认为确定客户需要什么是一件非常容易的事，只需向客户询问就能轻松获得。其实它远比我们想象的困难，原因主要有三个。

① 首先是客户说不清需求。有些客户对需求只有模糊不清的感觉，自然不能很好地用语言描述出来；有些客户心里非常清楚想要什么，但却表达不清楚。少数客户本身就懂得软件开发，能把需求说得清清楚楚，这样的需求分析将会非常轻松、愉快，但这种可能性很小。

② 其次是需求自身不断变化。有些客户对目标系统的预期一天一个想法，一拍脑袋就变一个主意，这是软件开发人员最头疼的事情，但是也只能面对现实寻找策略应对这种情况。分析人员在进行需求分析时就要注意以下两点：

- 尽可能地分析清楚哪些是稳定的需求，哪些是易变的需求。以便在进行系统设计时，将软件的核心建筑在稳定的需求上。
- 在合同中一定要说清楚“做什么”和“不做什么”。如果合同不清晰明确，日后会引起很多纠纷。

③ 最后就是需求分析人员和用户交流中产生误解。

分析人员和目标系统客户，一方是计算机专业人员，另一方是系统应用领域专家。虽然彼此都对对方的领域有所了解，但离专业人员还是存在一定的距离，这样他们在沟通交流中就不可避免地可能会产生误解。

由于需求分析存在众多困难，所以对软件需求分析人员的要求是非常高的。他们通常都是资深的计算机专家，同时具备丰富的业务领域知识和良好的沟通技能。

1.3.2 需求阶段概述

需求阶段的第一步是理解应用领域，也就是目标系统应用的特定环境，例如银行、证券公司、学校、政府等。一旦开发团队充分了解应用领域，就可以实施目标系统的系统建模，一种很主流的建模方法就是使用统一建模语言 UML 来描述目标系统的业务逻辑。通过模型，开发团队可以和目标系统的用户进行充分交流以确定客户的业务需求，确定客户的最终需求是一个反复迭代的过程，经过多次沟通、理解、修正才能比较客观地确定客户对系统的真实需求。

1.3.3 理解应用域

为了有效地挖掘出客户的真实需求，技术人员必须熟悉目标系统的应用领域。如果不了解系统的业务领域，很难向客户提出有意义的问题，所以不可能成功完成系统的需求分析。理解应用域的方法就是亲临目标系统将来应用的真实环境，去了解目标系统将完成哪些业务，这些业务手工完成的流程是怎样的，目前的业务流程有哪些优点和不足，等等，只有完全弄明白了这些问题，才有可能与客户进行充分且有效的交流。

1.3.4 用例建模

使用 UML 中的“用例图”描述拟建软件与外部环境之间的关系。一个用例表示一个外部角色 Actor(例如，用户或其他外部软件环境)与拟建软件之间一个单独交互。将所有的用例集中在一起，就可以描述一个 OO 软件的总体功能需求。例如，一个网上拍卖系统的拍卖过程用例图(见图1.2)。

用例图在软件的建模过程中是必不可少的。

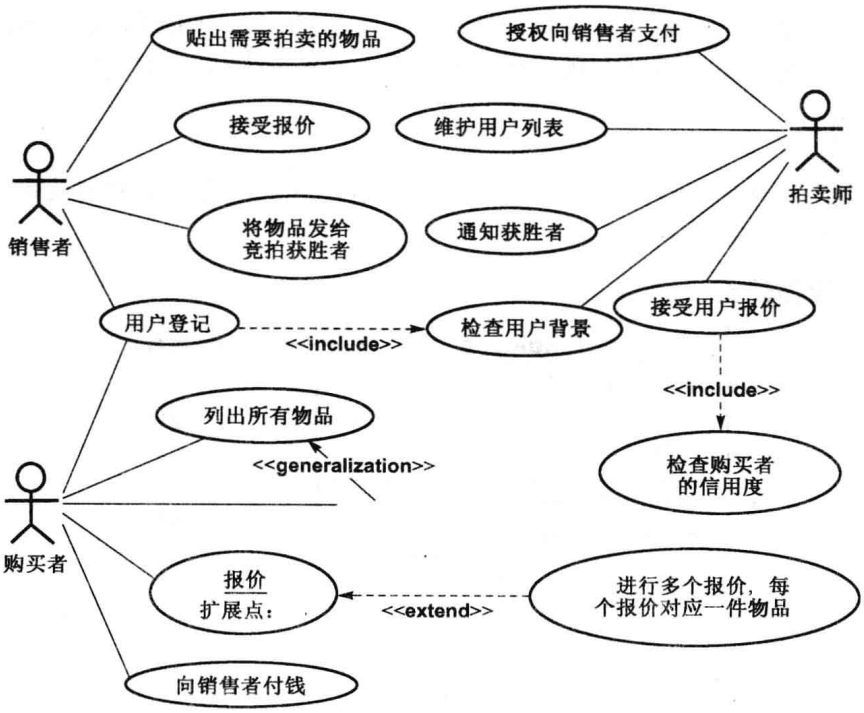


图 1.2　拍卖过程用例图

1.4　面向对象设计

1.4.1　有效应用设计模式

众所周知，人是一种经验性的动物。我们常说：据我所知，据我所了解，据我的经验，等等。生活中很多事情都是依靠自己或他人的经验，软件设计也类同。建筑大师 Christopher Alexander 说过："每一个模式描述了一个在我们周围不断重复发生的问题，以及该问题的解决方案的核心。这样，你就能一次又一次地使用该方案而不必做重复劳动。"他所指的是建筑领域，软件设计模式则是描述软件设计过程中某一类常见问题的一般性的解决方案。1995 年，Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides 四人合著了一本经典巨作《设计模式——可复用面向对象软件的基础》，该书奠定了坚实的面向对象设计模式理论的基础，它介绍了 23 种基本设计模式的结构、特性和使用方法。《设计模式》一书被软件人士当作"模式的圣经"，是所有面向对象分析设计人员必读的书籍。四位作者被大家称为"四人帮"（Gang of Four, GoF）。

由于《设计模式——可复用面向对象软件的基础》一书奠定了设计模式的地位，人们通常所说的设计模式隐含地表示"面向对象设计模式"。但这并不意味"设计模式"就等于"面向对象设计模式"，也不意味着 GoF 23 种模式就表示了所有的"面向对象设计模式"。除了"面向对象设计模式"外，还有其他设计模式。除了 GoF 23 种设计模式外，还有更多的面向对象设计模式。

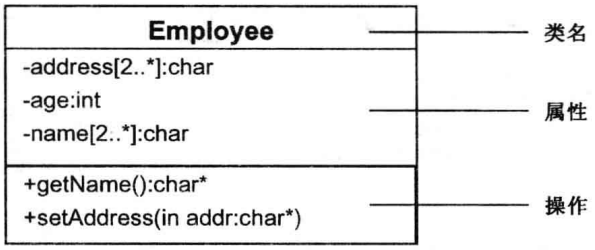
面向对象设计模式精髓是隐藏在背后的三条面向对象设计原则和设计理念：适应需求的变化；针对接口编程，而不要针对实现编程；优先使用聚合，而不是继承。

根据目标系统的特点和需要，选择合适的模式加以应用，应用模式后最大的优点就是系统可以适应需求的变化，从而具有更好的可复用性、可扩展性和可维护性。

1.4.2 类建模

使用“类图”描述所设计的软件中包含的类，以及这些类之间的静态关系，从而描绘了整个软件的静态组成和结构。

1. 类的一般描述图



(1) 类名

类名即类的名称。

(2) 属性

属性描述的语法：

```
visibility name [N] : type = initialValue {property-string}
```

① visibility：属性的可见性：+ 表示 public； # 表示 protected；– 表示 private。

② name：属性的名称。

③ [N]：属性的多值性：[2..*]表示属性能接受多值；如果属性描述中无此项，则表示该属性只允许接受一个值。

④ type：属性的实现类型(依赖于实现语言的规范)。

⑤ initialValue：属性的初始值。

⑥ property-string：表示属性的一些无法用上述语法描述的特性。例如，一个只读属性(如 C++的 const 成员或 Java 的 final 成员)， 则 property-string 将可以设置为{frozen}。

(3) 操作

操作描述的语法：

```
visibility name (parameter-list) : return-type {operation-string}
```

① visibility：操作的可见性：+ 表示 public；# 表示 protected；– 表示 private。

② name：操作的名称。

③ parameter-list：操作的形参列表，表中允许有多个形参，形参之间由逗号分隔。每个参数的表示语法：kind name : type = defaultValue。

- kind：表示参数的作用分类
  - in 表示参数向操作中传入一个值。
  - out 表示参数从操作中提取一个值。
  - inout 表示参数既可以向操作中传入一个值又可以从操作中提取一个值。
- name：表示形参名。

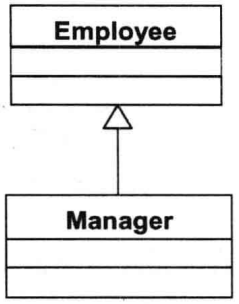
- type：表示形参类型。
- defaultValue：表示形参的默认值。

④ return-type：操作的返回类型(因编译器而异)。

⑤ operation-string：表示操作的一些无法用上述语法描述的特性。例如，一个操作是抽象操作，则 operation-string 可以设置为{abstract}。

2．类的相互关系图

用于描述类之间的静态关系。在关系图中的类图除了类名部分不能省略外，其他两部分都可以根据需要省略。主要的静态关系有：

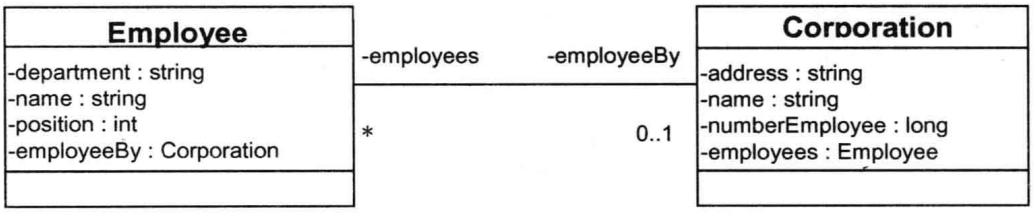


(1)归纳关系(Generalization)

归纳关系表示两个类之间的继承和派生关系，如右图中的类 Employee 是类 Manager 的超类(基类)，而类 Manager 是类 Employee 的子类(派生类)。

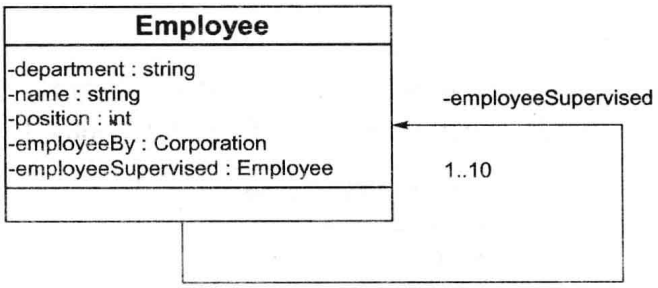
(2)关联关系(Association)

关联关系表示两个类之间的关联关系，如下图中类 Employee 和类 Corporation 之间的关联关系。



图中 employeeBy 是 Employee 的属性成员，其类型为 Corporation。通过该属性 employeeBy 使两个类关联；0..1 表示关联的多样性，此关联表示一个 Employee 对象不会被超过一个 Corporation 对象所雇用。同样，employees 是 Corporation 的属性成员，其类型为 Employee。*表示一个 Corporation 对象允许有 0 至任意数量的 Employee 对象，即雇用任意多个雇员。上述两端关联关系也可以用两条箭头线分别表示，箭头线的方向是从关联属性所在类指向属性的类型类。

下面的关联关系表示同一类的两个以上不同对象间的关联。它的含义是该雇员可以管理 1～10 个其他员工。

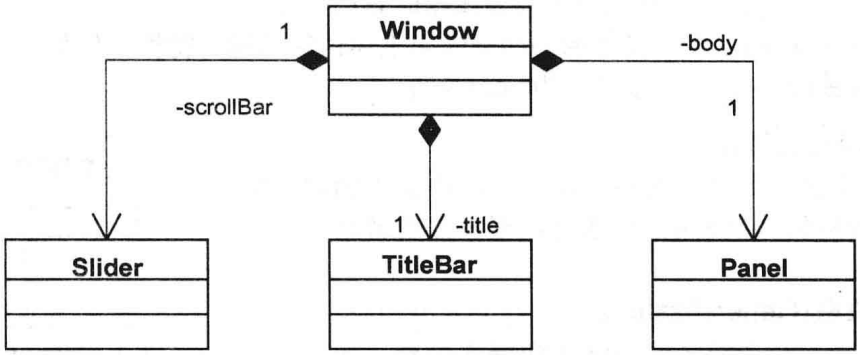


注意，参与一个关联的两个对象常常是独立存在的，即关联关系中的一个对象的存在与否不会影响到所关联的另一个对象的存在。

(3)类之间的聚合和合成关系

表示对象之间的“整体”和“部分”之间的关系，即在整体和部分之间可能存在生命期

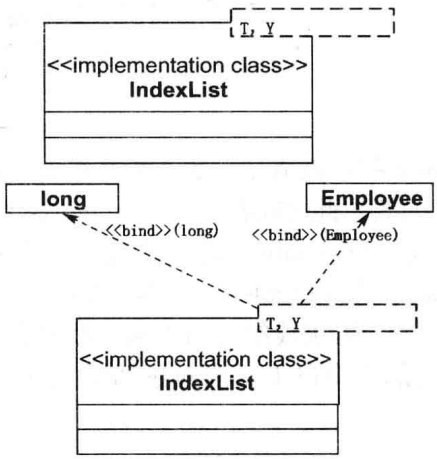
的依赖性。合成关系表示当整体不再存在时，部分同时被销毁的紧密关系。例如，下图中的 Window 和 Slider、TitleBar、Panel 之间的关系。



聚合关系表示当整体不再存在之后，部分还会继续存在的关系。例如，类 Orchestra 和 Performer 之间的关系。当然这种关系也可以通过前面介绍过的关联关系表示。但是，聚合关系可以很形象地说明一些概念。如本例中说明：乐队是由演奏人员所组成的，但乐队的生存期与演奏人员的寿命并不存在紧密依赖的关系。

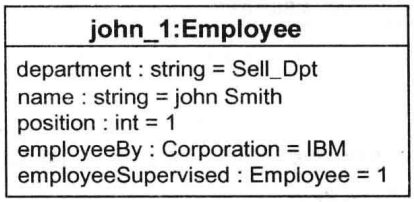
(4) 类模板和类模板的实例化

类模板是具有成员类型参数的类。例如，下图中的类就是一个具有两种成员类型参数 T 和 Y 的类模板，和将该类模板的成员类型实例化后的类。



(5) 类的实例——对象

对象是按照类定义创建的实例，在对象的图形描述中实例被命名，同时类的各个属性被赋予特定的值。例如，下图就是一个用 Employee 类创建的对象 john_1。



显然，在建模的主要工作——类设计中，使用类图是十分必要的，也是十分方便的。

1.4.3　状态图建模

状态图最适合用于显示一个类对象在经历一系列相关用例的过程中，所呈现的不同状态。它们可以帮助我们更好地理解一个 OO 程序中单一对象的生命期行为。状态图主要由三种图示构件组成：

① 对象状态：对象的一个状态用圆角矩形表示。在该矩形中，状态名称用粗体字显示在矩形最上方；如果还有额外信息，用一条线将这些额外信息与状态名分隔开，以“do:/”为前缀的信息项目表示处于此状态下的一个活动。开始和结束是两个特殊状态。

② 转换路径：对象从一个状态转换到另一个状态用一条箭头线（箭头指向转换后的状态）表示。

③ 状态转换标签：描述引起状态转换的原因，描述语法是

```
Event[Guard]/Action
```

该语法的含义是：事件 Event 的出现将导致以 Guard 为条件的状态转换结果为 TRUE。但在实际转换到新状态之前，动作 Action 必须先执行。由于对象从一个状态到其他状态只能进行一次转换，因此从一个给定状态可能出现的所有转换路径必须相互排斥。注意，转换描述的三个部分（事件、条件和动作）都是可选择的。

图1.3 描述了一次拍卖程序中 Buyer 对象的一个状态图。

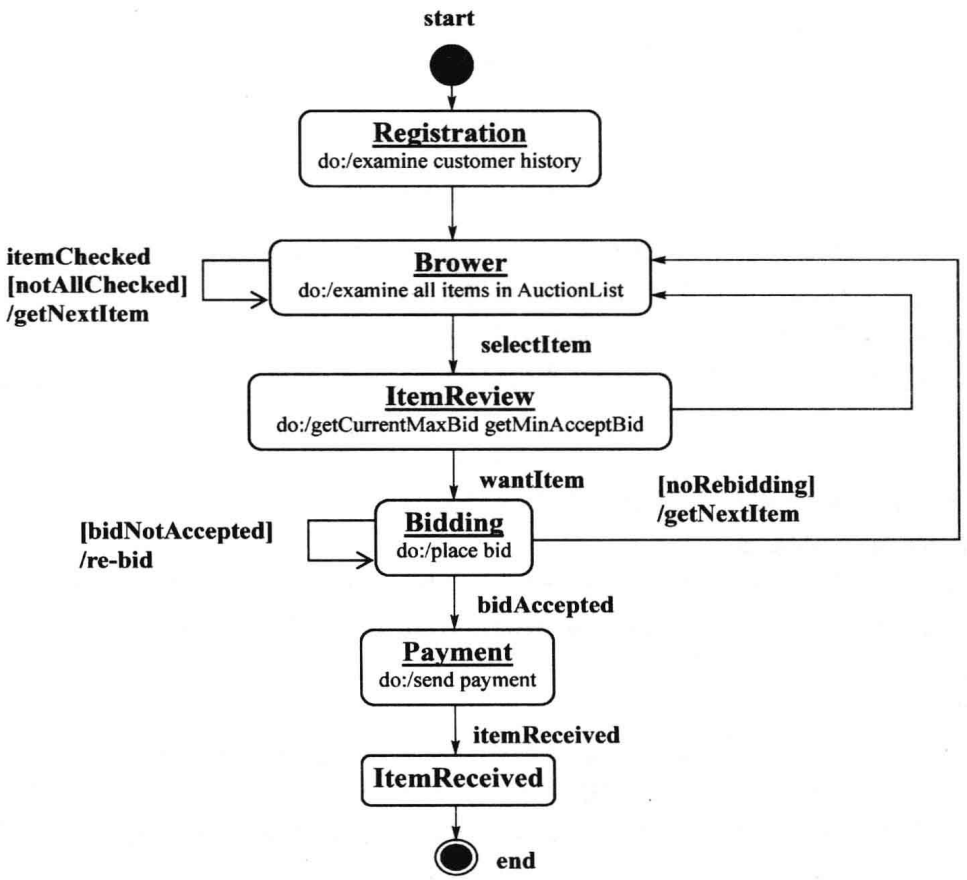


图 1.3　Buyer 对象的一个状态图