普通高等教育
软件工程
“十二五”规划教材

12th Five-Year Plan Textbooks
of Software Engineering

工业和信息化普通高等教育
“十二五”规划教材

软件设计模式

（Java版）

程细柱 ◎ 编著

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

普通高等教育
软件工程 “十二五”规划教材

12th Five-Year Plan Textbook
of Software Engineering

工业和信息化普通高等教育
“十二五”规划教材

软件设计模式

（Java版）

程细柱 ◎ 编著

Software Design Pattern

人民邮电出版社
北京

图书在版编目（CIP）数据

软件设计模式 : Java版 / 程细柱编著. -- 北京 : 人民邮电出版社, 2018.6
普通高等教育软件工程“十二五”规划教材
ISBN 978-7-115-47788-0

Ⅰ. ①软… Ⅱ. ①程… Ⅲ. ①JAVA语言－软件设计－高等学校－教材 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2018)第010096号

内 容 提 要

本书从面向对象程序设计的 7 个基本原则出发，用浅显易懂、可视化的 UML 建模语言逐一介绍 GoF 的 23 种经典设计模式。全书共 9 章，内容包括设计模式基础、创建型模式（共 5 种）、结构型模式（共 7 种）、行为型模式（共 11 种）、设计模式实验指导。前 8 章每章包括教学目标、重点内容、小结和习题等内容，对各模式都介绍了模式的定义与特点、模式的结构与实现、模式的应用实例、模式的应用场景和模式的扩展。第 9 章为上机实验指导，可供读者实践与练习。本书配套有丰富的教学资源供下载，包括本书的课程标准、实验大纲、上机指导、相关案例的源代码、习题答案和电子课件等内容。

本书可作为高等院校计算机科学与技术、软件工程、信息系统与信息管理、电子商务等专业的程序设计类课程的教材，也可作为软件开发者的自学用书。

◆ 编　　著　程细柱
责任编辑　张　斌
责任印制　沈　蓉　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
固安县铭成印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：15.75　　2018 年 6 月第 1 版
字数：423 千字　　2018 年 6 月河北第 1 次印刷

定价：49.80 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前 言

随着软件开发复杂度的增加，软件开发成本变得越来越高。在软件设计中，提高代码的可复用性、可维护性、稳健性、安全性和可读性变得非常重要，GoF 的 23 种设计模式正好解决了其中的主要问题。

现在大多数高等院校的计算机科学与技术专业、软件工程专业都开设了软件设计模式的课程，有些院校的信息管理专业和物联网专业也开设了该课程。但是，目前市场上出现的此类书主要是专著，可作为教材的较少，而且大部分没有提供配套的教辅资源，不太适合作为本专科院校的教学用书。为了满足社会需求，让学生能充分掌握这 23 种设计模式，提高其软件开发能力，有必要编写适用于高校的教材。

本书采用“任务驱动”的教学方法，根据各种设计模式之间的关系和相似点组织教材目录，对每种模式提出产生背景，并用 UML 建模语言分析模式的结构，然后用简单易懂的实例加深学生对该模式的理解。本书的实例都取材于生活，且尽量提供丰富多彩的窗体程序开发，这是其他的教材中难见到的。本书重视编程训练，做到理论与实践相结合，每章包括：教学目标、重点内容、基本概念、基本原理、编程实例、应用场景、习题等多个方面的内容。另外，本书提供丰富的配套教学资源，主要包括本书的课程标准、实验大纲、上机指导、相关案例的源代码、习题答案和电子课件等内容。全书分为 9 章，各章的内容如下。

第 1 章　设计模式基础：主要介绍软件设计模式的产生背景、软件设计模式的定义与基本要素、软件设计模式的分类，以及学习软件设计模式的意义。另外，还介绍了后面各章要用到的 UML 类之间的关系，以及类图的画法。还重点讲解了软件设计必须遵循的 7 种面向对象设计原则。

第 2 章　创建型模式（上）：主要介绍创建型模式的特点和分类，以及单例模式与原型模式的定义与特点、结构与实现、应用场景和模式的扩展，并通过多个应用实例来说明模式的使用方法。

第 3 章　创建型模式（下）：主要介绍工厂方法模式、抽象工厂模式、建造者模式等 3 种创建型模式的定义、特点、结构与实现，并通过应用实例介绍了这 3 种创建型模式的实现方法，最后分析了它们的应用场景和扩展方向。

第 4 章　结构型模式（上）：主要介绍结构型模式的特点和分类，以及代理模式、适配器模式、桥接模式的定义、特点、结构、实现方法与扩展方向，并通过多个应用实例来说明这 3 种设计模式的应用场景和使用方法。

第 5 章　结构型模式（下）：主要介绍装饰模式、外观模式、享元模式、组合模式的定义、特点、结构、实现方法与扩展方向，并通过多个应用实例来说明这 4 种设计模式的应用场景和使用方法。

第 6 章　行为型模式（上）：主要介绍行为型模式的特点和分类，以及模板方法模式、策略模式、命令模式的定义、特点、结构、实现方法与扩展方向，并通过多个应用实例来说明这 3 种设计模式的应用场景和使用方法。

第 7 章　行为型模式（中）：主要介绍职责链模式、状态模式、观察者模式、中介者模式的定义、特点、结构、实现方法与扩展方向，并通过多个应用实例来说明这 4 种设计模式的应用场景和使用方法。

第 8 章　行为型模式（下）：主要介绍迭代器模式、访问者模式、备忘录模式、解释器模式的定义、特点、结构、实现方法与扩展方向，并通过多个应用实例来说明这 4 种设计模式的应用场景和使用方法。

第 9 章　设计模式实验指导：主要介绍类的基本概念和类之间关系，在 UMLet 中绘制类图的基本方法，以及创建型、结构型和行为型等 3 类设计模式的工作原理，并以工厂方法（Factory Method）模式、代理（Proxy）模式和观察者（Observer）模式为例介绍其相关类图的画法，以及应用相关设计模式开发应用程序的基本方法。每个实验都介绍了其实验目的、工作原理、实验内容、实验要求和实验步骤。

本书由程细柱编写，虽然在编写过程中倾注了大量心血，但书中难免存在疏漏和不足之处，恳请广大读者批评指正，本人不胜感谢。编者 E-mail：cxz973@qq.com。另外，本书免费提供的电子教案和源代码等相关教学资源，可从人邮教育网站（www.ryjiaoyu.com）下载。

编　者

2018 年 2 月

目 录

第1章　设计模式基础

📖本章教学目标：

- 了解软件设计模式的产生背景；
- 掌握软件设计模式的概念、意义和基本要素；
- 明白 GoF 的 23 种设计模式的分类与特点；
- 理解 UML 类之间的关系，并学会类图的画法；
- 正确理解面向对象的 7 种设计原则。

📖本章重点内容：

- GoF 的 23 种设计模式的分类与特点；
- UML 中的类之间的关系；
- UML 中的类图的画法；
- 面向对象的 7 种设计原则。

1.1　软件设计模式概述

本节是后面各章学习的基础，从整体上介绍软件设计模式的概念与特点、软件设计模式的基本要素，以及 GoF 的 23 种设计模式简介。

1.1.1　软件设计模式的产生背景

“设计模式”这个术语最初并不是出现在软件设计中，而是被用于建筑领域的设计中。1977 年，美国著名建筑大师、加利福尼亚大学伯克利分校环境结构中心主任克里斯托夫·亚历山大（Christopher Alexander）在他的著作《建筑模式语言：城镇、建筑、构造》（*A Pattern Language:Towns Building Construction*）中描述了一些常见的建筑设计问题，并提出了 253 种关于对城镇、邻里、住宅、花园和房间等进行设计的基本模式。1979 年他的另一部经典著作《建筑的永恒之道》（*The Timeless Way of Building*）进一步强化了设计模式的思想，为后来的建筑设计指明了方向。

1987 年，肯特·贝克（Kent Beck）和沃德·坎宁安（Ward Cunningham）首先将克里斯托夫·亚历山大的模式思想应用在 Smalltalk 中的图形用户接口的生成中，但没有引起软件界的关注。直到 1990 年，软件工程界才开始研讨设计模式的话题，后来召开了多次关于设计模式的研讨会。1995 年，艾瑞克·伽马（Erich Gamma）、理查德·海尔姆（Richard Helm）、拉尔夫·约翰森（Ralph Johnson）、约翰·威利斯迪斯

（John Vlissides）等 4 位作者合作出版了《设计模式：可复用面向对象软件的基础》（*Design Patterns: Elements of Reusable Object-Oriented Software*）一书，在此书中收录了 23 个设计模式，这是设计模式领域里程碑的事件，导致了软件设计模式的突破。这 4 位作者在软件开发领域里也以他们的“四人组”（Gang of Four，GoF）匿名著称。直到今天，狭义的设计模式还是这本书中所介绍的 23 种经典设计模式。

1.1.2 软件设计模式的概念与意义

有关软件设计模式的定义很多，有些从模式的特点来说明，有些从模式的作用来说明。本书给出的定义是大多数学者公认的，从以下两个方面来说明。

1. 软件设计模式的概念

软件设计模式（Software Design Pattern），又称设计模式，是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。它描述了在软件设计过程中的一些不断重复发生的问题，以及该问题的解决方案。也就是说，它是解决特定问题的一系列套路，是前辈们的代码设计经验的总结，具有一定的普遍性，可以反复使用。其目的是为了提高代码的可重用性、代码的可读性和代码的可靠性。

2. 学习设计模式的意义

设计模式的本质是面向对象设计原则的实际运用，是对类的封装性、继承性和多态性以及类的关联关系和组合关系的充分理解。正确使用设计模式具有以下优点。

（1）可以提高程序员的思维能力、编程能力和设计能力。

（2）使程序设计更加标准化、代码编制更加工程化，使软件开发效率大大提高，从而缩短软件的开发周期。

（3）使设计的代码可重用性高、可读性强、可靠性高、灵活性好、可维护性强。

当然，软件设计模式只是一个引导。在具体的软件开发中，必须根据设计的应用系统的特点和要求来恰当选择。对于简单的程序开发，可能写一个简单的算法要比引入某种设计模式更加容易。但对大项目的开发或者框架设计，用设计模式来组织代码显然更好。

1.1.3 软件设计模式的基本要素

软件设计模式使人们可以更加简单方便地复用成功的设计和体系结构，它通常包含以下几个基本要素：模式名称、别名、动机、问题、解决方案、效果、结构、模式角色、合作关系、实现方法、适用性、已知应用、例程、模式扩展和相关模式等，其中最关键的元素包括以下 4 个主要部分。

1. 模式名称

每一个模式都有自己的名字，通常用一两个词来描述，可以根据模式的问题、特点、解决方案、功能和效果来命名。模式名称（Pattern Name）有助于我们理解和记忆该模式，也方便我们来讨论自己的设计。

2. 问题

问题（Problem）描述了该模式的应用环境，即何时使用该模式。它解释了设计问题和问题存在的前因后果，以及必须满足的一系列先决条件。

3. 解决方案

模式问题的解决方案（Solution）包括设计的组成成分、它们之间的相互关系及各自的职责和协

作方式。因为模式就像一个模板，可应用于多种不同场合，所以解决方案并不描述一个特定而具体的设计或实现，而是提供设计问题的抽象描述和怎样用一个具有一般意义的元素组合（类或对象的组合）来解决这个问题。

4. 效果

描述了模式的应用效果以及使用该模式应该权衡的问题，即模式的优缺点。主要是对时间和空间的衡量，以及该模式对系统的灵活性、扩充性、可移植性的影响，也考虑其实现问题。显式地列出这些效果（Consequence）对理解和评价这些模式有很大的帮助。

1.1.4　GoF 的 23 种设计模式简介

设计模式有两种分类方法，即根据模式的目的来分和根据模式的作用的范围来分。

1. 根据目的来分

根据模式是用来完成什么工作来划分，这种方式可分为创建型模式、结构型模式和行为型模式 3 种。

（1）创建型模式：用于描述“怎样创建对象”，它的主要特点是“将对象的创建与使用分离”。GoF 中提供了单例、原型、工厂方法、抽象工厂、建造者等 5 种创建型模式。

（2）结构型模式：用于描述如何将类或对象按某种布局组成更大的结构，GoF 中提供了代理、适配器、桥接、装饰、外观、享元、组合等 7 种结构型模式。

（3）行为型模式：用于描述类或对象之间怎样相互协作共同完成单个对象都无法单独完成的任务，以及怎样分配职责。GoF 中提供了模板方法、策略、命令、职责链、状态、观察者、中介者、迭代器、访问者、备忘录、解释器等 11 种行为型模式。

2. 根据作用范围来分

根据模式是主要用于类上还是主要用于对象上来分，这种方式可分为类模式和对象模式两种。

（1）类模式：用于处理类与子类之间的关系，这些关系通过继承来建立，是静态的，在编译时刻便确定下来了。GoF 中的工厂方法、（类）适配器、模板方法、解释器属于该模式。

（2）对象模式：用于处理对象之间的关系，这些关系可以通过组合或聚合来实现，在运行时刻是可以变化的，更具动态性。GoF 中除了以上 4 种，其他的都是对象模式。

表 1.1 介绍了这 23 种设计模式的分类。

表 1.1　GoF 的 23 种设计模式的分类表

范围\目的	创建型模式	结构型模式	行为型模式
类模式	工厂方法	（类）适配器	模板方法、解释器
对象模式	单例 原型 抽象工厂 建造者	代理 （对象）适配器 桥接 装饰 外观 享元 组合	策略 命令 职责链 状态 观察者 中介者 迭代器 访问者 备忘录

3. GoF 的 23 种设计模式的功能

前面说明了 GoF 的 23 种设计模式的分类，现在对各个模式的功能进行介绍。

（1）单例（Singleton）模式：某个类只能生成一个实例，该类提供了一个全局访问点供外部获取该实例，其拓展是有限多例模式。

（2）原型（Prototype）模式：将一个对象作为原型，通过对其进行复制而克隆出多个和原型类似的新实例。

（3）工厂方法（Factory Method）模式：定义一个用于创建产品的接口，由子类决定生产什么产品。

（4）抽象工厂（Abstract Factory）模式：提供一个创建产品族的接口，其每个子类可以生产一系列相关的产品。

（5）建造者（Builder）模式：将一个复杂对象分解成多个相对简单的部分，然后根据不同需要分别创建它们，最后构建成该复杂对象。

（6）代理（Proxy）模式：为某对象提供一种代理以控制对该对象的访问。即客户端通过代理间接地访问该对象，从而限制、增强或修改该对象的一些特性。

（7）适配器（Adapter）模式：将一个类的接口转换成客户希望的另外一个接口，使得原本由于接口不兼容而不能一起工作的那些类能一起工作。

（8）桥接（Bridge）模式：将抽象与实现分离，使它们可以独立变化。它是用组合关系代替继承关系来实现，从而降低了抽象和实现这两个可变维度的耦合度。

（9）装饰（Decorator）模式：动态的给对象增加一些职责，即增加其额外的功能。

（10）外观（Facade）模式：为多个复杂的子系统提供一个一致的接口，使这些子系统更加容易被访问。

（11）享元（Flyweight）模式：运用共享技术来有效地支持大量细粒度对象的复用。

（12）组合（Composite）模式：将对象组合成树状层次结构，使用户对单个对象和组合对象具有一致的访问性。

（13）模板方法（Template Method）模式：定义一个操作中的算法骨架，而将算法的一些步骤延迟到子类中，使得子类可以不改变该算法结构的情况下重定义该算法的某些特定步骤。

（14）策略（Strategy）模式：定义了一系列算法，并将每个算法封装起来，使它们可以相互替换，且算法的改变不会影响使用算法的客户。

（15）命令（Command）模式：将一个请求封装为一个对象，使发出请求的责任和执行请求的责任分割开。

（16）职责链（Chain of Responsibility）模式：把请求从链中的一个对象传到下一个对象，直到请求被响应为止。通过这种方式去除对象之间的耦合。

（17）状态（State）模式：允许一个对象在其内部状态发生改变时改变其行为能力。

（18）观察者（Observer）模式：多个对象间存在一对多关系，当一个对象发生改变时，把这种改变通知给其他多个对象，从而影响其他对象的行为。

（19）中介者（Mediator）模式：定义一个中介对象来简化原有对象之间的交互关系，降低系统中对象间的耦合度，使原有对象之间不必相互了解。

（20）迭代器（Iterator）模式：提供一种方法来顺序访问聚合对象中的一系列数据，而不暴露聚

合对象的内部表示。

（21）访问者（Visitor）模式：在不改变集合元素的前提下，为一个集合中的每个元素提供多种访问方式，即每个元素有多个访问者对象访问。

（22）备忘录（Memento）模式：在不破坏封装性的前提下，获取并保存一个对象的内部状态，以便以后恢复它。

（23）解释器（Interpreter）模式：提供如何定义语言的文法，以及对语言句子的解释方法，即解释器。

必须指出，这23种设计模式不是孤立存在的，很多模式之间存在一定的关联关系，在大的系统开发中常常同时使用多种设计模式，希望读者认真学好它们。

1.2 UML中的类图

1.2.1 统一建模语言简介

统一建模语言（Unified Modeling Language，UML）是用来设计软件蓝图的可视化建模语言，1997年被国际对象管理组织（OMG）采纳为面向对象的建模语言的国际标准。它的特点是简单、统一、图形化、能表达软件设计中的动态与静态信息。它能为软件开发的所有阶段提供模型化和可视化支持。它融入了软件工程领域的新思想、新方法和新技术，使软件设计人员沟通更简明，进一步缩短了设计时间，减少开发成本。它的应用领域很宽，不仅适合于一般系统的开发，而且适合于并行与分布式系统的建模。

UML从目标系统的不同角度出发，定义了用例图、类图、对象图、状态图、活动图、时序图、协作图、构件图、部署图等9种图。本书主要介绍软件设计模式中经常用到的类图，以及类之间的关系。另外，在实验部分将简单介绍UML建模工具的使用方法，当前业界使用最广泛的是Rational Rose。使用Umlet的人也很多，它是一个轻量级的开源UML建模工具，简单实用，常用于小型软件系统的开发与设计。

1.2.2 类、接口和类图

1. 类

类（Class）是指具有相同属性、方法和关系的对象的抽象,它封装了数据和行为，是面向对象程序设计（OOP）的基础，具有封装性、继承性和多态性等三大特性。在UML中，类使用包含类名、属性和操作且带有分隔线的矩形来表示。

（1）类名（Name）是一个字符串，例如，Student。

（2）属性（Attribute）是指类的特性，即类的成员变量。UML按以下格式表示：

［可见性］ 属性名：类型 ［=默认值］

注意 “可见性”表示该属性对类外的元素是否可见，包括公有（Public）、私有（Private）、受保护（Protected）和朋友（Friendly）4种，在类图中分别用符号+、−、#、~表示。

例如：

```
- name: String
```

（3）操作（Operations）是类的任意一个实例对象都可以使用的行为，是类的成员方法。UML 按以下格式表示：

```
[可见性] 名称（参数列表） [: 返回类型]
```

例如：

```
+ display(): void。
```

图 1.1 所示是学生类的 UML 表示。

2. 接口

接口（Interface）是一种特殊的类，它具有类的结构但不可被实例化，只可以被子类实现。它包含抽象操作，但不包含属性。它描述了类或组件对外可见的动作。在 UML 中，接口使用一个带有名称的小圆圈来进行表示。

图 1.2 所示是图形类接口的 UML 表示。

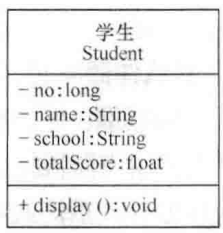

图 1.1 Student 类

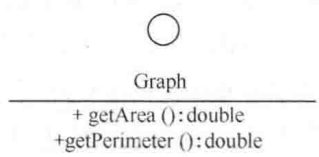

图 1.2 Graph 接口

3. 类图

类图（Class Diagram）是用来显示系统中的类、接口、协作以及它们之间的静态结构和关系的一种静态模型。它主要用于描述软件系统的结构化设计，帮助人们简化对软件系统的理解，它是系统分析与设计阶段的重要产物，也是系统编码与测试的重要模型依据。类图中的类可以通过某种编程语言直接实现。类图在软件系统开发的整个生命周期都是有效的，它是面向对象系统的建模中最常见的图。图 1.3 所示是“计算长方形和圆形的周长与面积”的类图，图形接口有计算面积和周长的抽象方法，长方形和圆形实现这两个方法供访问类调用。

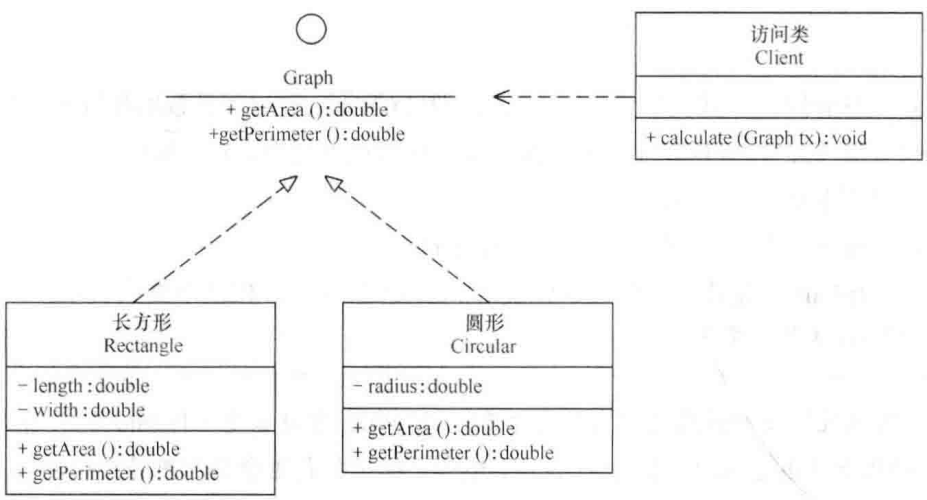

图 1.3 “计算长方形和圆形的周长与面积”的类图

1.2.3　类之间的关系

在软件系统中，类不是孤立存在的，类与类之间存在各种关系。根据类与类之间的耦合度从弱到强排列，UML 中的类图有以下几种关系：依赖关系、关联关系、聚合关系、组合关系、泛化关系和实现关系。其中泛化和实现的耦合度相等，它们是最强的。

1. 依赖关系

依赖（Dependency）关系是一种使用关系，它是对象之间耦合度最弱的一种关联方式，是临时性的关联。在代码中，某个类的方法通过局部变量、方法的参数或者对静态方法的调用来访问另一个类（被依赖类）中的某些方法来完成一些职责。在 UML 类图中，依赖关系使用带箭头的虚线来表示，箭头从使用类指向被依赖的类。图 1.4 所示是人与手机的关系图，人通过手机的语音传送方法打电话。

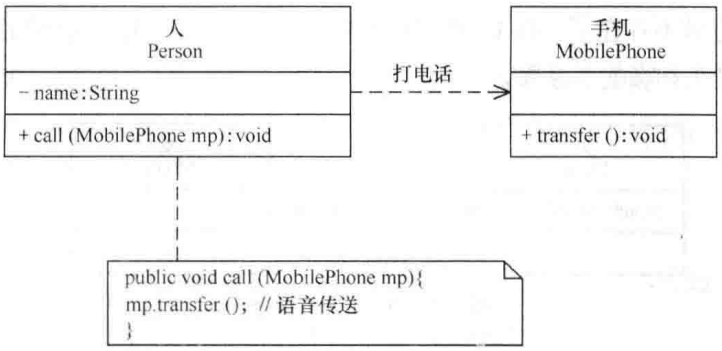

图 1.4　依赖关系的实例

2. 关联关系

关联（Association）关系是对象之间的一种引用关系，用于表示一类对象与另一类对象之间的联系，如老师和学生、师傅和徒弟、丈夫和妻子等。关联关系是类与类之间最常用的一种关系，分为一般关联关系、聚合关系和组合关系。我们先介绍一般关联。关联可以是双向的，也可以是单向的。在 UML 类图中，双向的关联可以用带两个箭头或者没有箭头的实线来表示，单向的关联用带一个箭头的实线来表示，箭头从使用类指向被关联的类。也可以在关联线的两端标注角色名，代表两种不同的角色。在代码中通常将一个类的对象作为另一个类的成员变量来实现关联关系。图 1.5 所示是老师和学生的关系图，每个老师可以教多个学生，每个学生也可向多个老师学，他们是双向关联。

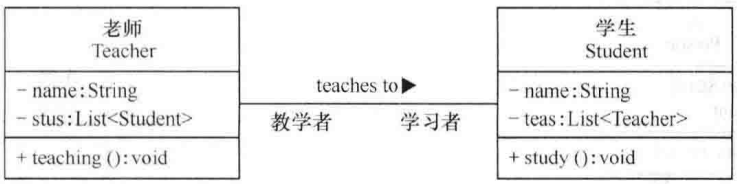

图 1.5　关联关系的实例

3. 聚合关系

聚合（Aggregation）关系是关联关系的一种，是强关联关系，是整体和部分之间的关系，是 has-a 的关系。聚合关系也是通过成员对象来实现的，其中成员对象是整体对象的一部分，但是成员对象可以脱离整体对象而独立存在。例如，学校与老师的关系，学校包含老师，但如果学校停办了，老师依然存在。在 UML 类图中，聚合关系可以用带空心菱形的实线来表示，菱形指向整体。图 1.6 所

示是大学和教师的关系图。

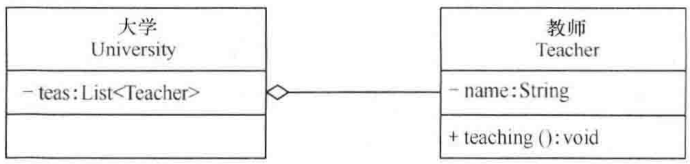

图 1.6　聚合关系的实例

4. 组合关系

组合（Composition）关系也是关联关系的一种，也表示类之间的整体与部分的关系，但它是一种更强烈的聚合关系，是 contains-a 关系。在组合关系中，整体对象可以控制部分对象的生命周期，一旦整体对象不存在，部分对象也将不存在，部分对象不能脱离整体对象而存在。例如，头和嘴的关系，没有了头，嘴也就不存在了。在 UML 类图中，组合关系用带实心菱形的实线来表示，菱形指向整体。图 1.7 所示是头和嘴的关系图。

图 1.7　组合关系的实例

5. 泛化关系

泛化（Generalization）关系是对象之间耦合度最大的一种关系，表示一般与特殊的关系,是父类与子类之间的关系，是一种继承关系，是 is-a 的关系。在 UML 类图中，泛化关系用带空心三角箭头的实线来表示，箭头从子类指向父类。在代码实现时，使用面向对象的继承机制来实现泛化关系。例如，Student 类和 Teacher 类都是 Person 类的子类，其类图如图 1.8 所示。

6. 实现关系

实现（Realization）关系是接口与实现类之间的关系。在这种关系中，类实现了接口，类中的操作实现了接口中所声明的所有的抽象操作。在 UML 类图中，实现关系使用带空心三角箭头的虚线来表示，箭头从实现类指向接口。例如，汽车和船实现了交通工具，其类图如图 1.9 所示。

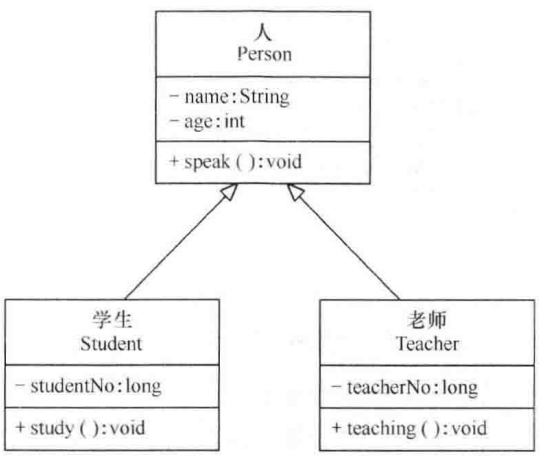

图 1.8　泛化关系的实例

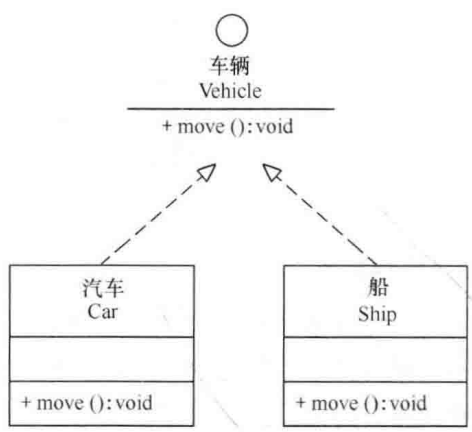

图 1.9　实现关系的实例

1.3　面向对象的设计原则

在软件开发中，为了提高软件系统的可维护性和可复用性，增加软件的可扩展性和灵活性，程序员要尽量根据以下 7 条原则来开发程序，从而提高软件开发效率、节约软件开发成本和维护成本。

1.3.1　开闭原则

1. 开闭原则的定义

开闭原则（Open Closed Principle，OCP）由勃兰特·梅耶（Bertrand Meyer）提出，他在 1988 年的著作《面向对象软件构造》（Object Oriented Software Construction）中提出：软件实体应当对扩展开放，对修改关闭（Software entities should be open for extension,but closed for modification），这就是开闭原则的经典定义。

这里的软件实体包括以下几个部分：①项目中划分出的模块；②类与接口；③方法。开闭原则的含义是：当应用的需求改变时，在不修改软件实体的源代码或者二进制代码的前提下，可以扩展模块的功能，使其满足新的需求。

2. 开闭原则的作用

开闭原则是面向对象程序设计的终极目标，它使软件实体拥有一定的适应性和灵活性的同时具备稳定性和延续性。具体来说，其作用如下。

（1）对软件测试的影响

软件遵守开闭原则的话，软件测试时只需要对扩展的代码进行测试就可以了，因为原有的测试代码仍然能够正常运行。

（2）可以提高代码的可复用性

粒度越小，被复用的可能性就越大；在面向对象的程序设计中，根据原子和抽象编程可以提高代码的可复用性。

（3）可以提高软件的可维护性

遵守开闭原则的软件，其稳定性高和延续性强，从而易于扩展和维护。

3. 开闭原则的实现方法

可以通过“抽象约束、封装变化”来实现开闭原则，即通过接口或者抽象类为软件实体定义一个相对稳定的抽象层，而将相同的可变因素封装在相同的具体实现类中。因为抽象灵活性好，适应性广，只要抽象的合理，可以基本保持软件架构的稳定。而软件中易变的细节可以从抽象派生来的实现类来进行扩展，当软件需要发生变化时，只需要根据需求重新派生一个实现类来扩展就可以了。下面以 Windows 的桌面主题为例介绍开闭原则的应用。

【例 1.1】 Windows 的桌面主题设计。

分析：Windows 的主题是桌面背景图片、窗口颜色和声音等元素的组合。用户可以根据自己的喜爱更换自己的桌面主题，也可以从网上下载新的主题。这些主题有共同的特点，可以为其定义一个抽象类（Abstract Subject），而每个具体的主题（Specific Subject）是其子类。用户窗体可以根据需要选择或者增加新的主题，而不需要修改原代码，所以它是满足开闭原则的，其类图如图 1.10 所示。