高等学校软件工程系列教材

软件设计模式与体系结构

主编　孙玉山　刘旭东　　副主编　黄俊恒　夏勇　朱东杰
主审　王宇颖

高等教育出版社
HIGHER EDUCATION PRESS

高等学校软件工程系列教材

软件设计模式与体系结构

Ruanjian Sheji Moshi yu Tixi Jiegou

主　编　孙玉山　刘旭东
副主编　黄俊恒　夏　勇　朱东杰
主　审　王宇颖

高等教育出版社·北京
HIGHER EDUCATION PRESS　BEIJING

内容提要

本书针对软件体系结构理论较为抽象，而在校学生往往无实践经验的特点，将高层的软件体系结构和低层的软件设计模式结合起来，并通过精心设计的实例，引导学生掌握本课程的相关内容。

全书分为上、下两篇，共7章。上篇为软件设计模式，包括第1～4章，分别为软件设计模式概述、创建型软件设计模式、结构型软件设计模式和行为型软件设计模式；下篇为软件体系结构，包括第5～7章，分别为软件体系结构概述、经典软件体系结构和基于网络的软件体系结构。为便于读者理解与掌握相应的理论知识，书中给出了大量的设计实例和趣味性较强的课下设计－编程作业，每个实例和作业都试图解释使用相应设计模式或者体系结构进行设计的要点所在，且均由Java语言实现，完整代码见书后所附光盘。

本书可作为普通高等学校计算机科学与技术、软件工程等专业本科高年级或研究生相关课程教材，也可供软件工程师参考使用。

图书在版编目(CIP)数据

软件设计模式与体系结构/孙玉山，刘旭东主编. --北京：高等教育出版社，2013.5

ISBN 978-7-04-037202-1

Ⅰ. ①软… Ⅱ. ①孙… ②刘… Ⅲ. ①软件设计－高等学校－教材 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2013)第068460号

策划编辑 倪文慧　　责任编辑 倪文慧　　封面设计 于文燕　　版式设计 王艳红

插图绘制 尹　莉　　责任校对 刘娟娟　　责任印制 赵义民

出版发行 高等教育出版社
社　　址 北京市西城区德外大街4号
邮政编码 100120
印　　刷 北京东君印刷有限公司
开　　本 850mm×1168mm　1/16
印　　张 20.25
字　　数 450千字
购书热线 010-58581118

咨询电话 400-810-0598
网　　址 http://www.hep.edu.cn
　　http://www.hep.com.cn
网上订购 http://www.landraco.com
　　http://www.landraco.com.cn
版　　次 2013年5月第1版
印　　次 2013年5月第1次印刷
定　　价 38.00元(含光盘)

物 料 号 37202-00

前　言

早期的软件规模小、功能简单，人们主要关注的是其内部的数据结构和算法。然而随着需求的复杂化、多样化，软件规模变得越来越大，人们逐渐开始认识到，软件体系结构（亦称软件架构）的设计相对于算法和数据结构的选择重要得多。根据美国有线电视新闻网的调查，软件架构师是2010年美国最热门的职业。在我国，软件架构师也是炙手可热的职业。而要成为优秀的架构师就必须掌握软件架构和设计模式等相关的知识和技能。

对软件体系结构和软件设计模式的研究主要来自于建筑设计领域的启发。软件体系结构是软件的高层结构，是有关软件整体结构与组件的抽象描述，用于指导大型软件系统的设计。可以这样说，软件体系结构设计的好坏决定了整个软件开发项目的成败。优秀的软件体系结构，应该在可扩展性、可复用性、可维护性、可靠性、安全性、可伸缩性等诸多质量属性中作完美的折中。

与软件体系结构不同，软件设计模式主要用于软件的局部设计。设计模式是面向对象设计中反复出现的某类问题的成功解决方案，设计者可以直接利用该模式进行同类问题设计，而不必重新构思设计方案。设计模式是设计复用的重要手段。

本书主要面向大学计算机科学与技术、软件工程等专业高年级本科生及研究生。针对软件体系结构较为抽象，而在校学生往往无实践经验的特点，本书特意将高层的软件体系结构和低层的软件设计模式结合起来，并通过精心设计的实例，引导学生掌握课程相关内容，这正是本书的特色所在。

本书分为上、下两篇，共7章。上篇为软件设计模式，包括第1～4章；下篇为软件体系结构，包括第5～7章。书中给出了大量的设计实例，每个实例都试图解释使用相关设计模式或者体系结构进行设计的要点所在。书中的实例均由Java语言实现，详见本书附带光盘。此外，全书还精心设计了许多趣味性较强的课下设计－编程作业，这些作业已给出了部分设计，学生只要在原有的基础上添加一个类或者一组类即可实现完整的设计。这些实例和课下设计－编程作业正是本书的精髓，如果没有这些实战性的练习，大量晦涩难懂的抽象概念将会使本课程的教学效果大打折扣。

建议本书按40学时讲授，教师可以有选择地讲授8～10个设计模式与部分或全部的软件体系结构方面的内容。建议将课程成绩分为两部分：设计－编程作业部分和期末考试部分。其中，设计－编程作业部分占40%或者50%，可以安排6～8个设计题目；期末考试部分占60%或者50%。

本书上篇第2～4章由孙玉山教授撰写，下篇由孙玉山教授给出初稿轮廓。刘旭东老师除独立撰写第1章、第5章及第7章7.5节以外，还对下篇补充了大量的内容。黄俊恒、夏勇教授分

别参与了上、下篇的编撰、修改与校对。朱东杰老师除了参与正文的增删、修改、校对以外，还负责书中所有练习题与课后作业的源代码的组织、验证和注释工作。

王宇颖教授对本书的撰写提出了原则性、架构性的建议，并且几次审阅了书稿，提出了很多宝贵的具体修改意见。张廷斌教授对7.1节的撰写提出了宝贵的建议，在此一并表示感谢。

北京大学软件学院的郁莲博士认真审阅了本书，并提出了中肯的意见与建议，在此向她表示衷心的感谢。

本书是作者对8年来相关课程教学内容的凝练，可作为高年级本科生或研究生教材使用，也可为已经在软件公司就职的软件工程师们提供借鉴。总之，作者希望本书成为一本很好的入门教材，使学习软件设计模式和软件体系结构成为一件比较轻松的事情。对于本书的不足之处，也希望读者能够及时指正（作者电子邮箱 mikesun725@ yahoo. com. cn），作者将不胜感激。

作者

2013年4月

目　录

上篇　软件设计模式

下篇 软件体系结构

上篇　软件设计模式

第1章　软件设计模式概述

1.1　软件设计模式的由来

设计模式(Design Pattern)的思想来源于建筑学领域。著名建筑学家 Christopher Alexander 和他的同事们在1977年出版的《A Pattern Language:Towns, Buildings, Construction》一书中,以模式(Pattern)这一术语来概括建筑学中常见的设计问题及其解决方案:"每个模式都描述了在环境中反复出现的(某类)问题,并以一种适当的方式描述该问题的核心解决方案,以使该方案可以千百遍地被重复使用。"可见,模式的核心思想是进行设计复用。

计算机科学家 Kent Beck 等人在1987年 OOPSLA 国际会议上介绍了他们在利用 Smalltalk 语言进行用户界面开发时所总结的一些模式,首次将模式的思想引入到面向对象程序设计领域,从此在软件工程领域开启了一场"模式"运动。

1994年,被人称为"四人帮"(Gang of Four,GOF)的 Erich Gamma、Richard Helm、Ralph Johnson 和 John Vissides 等人所著的《Design Patterns:Elements of Reusable Object - Oriented Software》成为软件工程中关于设计模式的标志性著作。在该书中,Erich Gamma 等人认为"设计模式描述了定制化的相互通信的对象与类,以解决特定环境中的通用设计问题。"并总结了面向程序设计中3类23种经典设计模式。

随着模式运动的发展,很多研究者从不同领域对设计模式展开了深入研究。

Frank Buschmann 等人所著的《Pattern - Oriented Software Architecture》系列(共4卷)是关于设计模式的另一套经典著作,主要讨论了分布式计算领域中的各种设计模式,特别是在第四卷中总结了13类114种设计模式。

在企业应用软件领域,Martin Fowler 在《Patterns of Enterprise Application Architecture》一书中,针对企业应用软件的特点,总结了10类41种设计模式。

Thomas Erl 在《SOA Design Patterns》中总结了面向服务架构中的设计模式,共5类85种。

除上述著作外,还有一类著作从实现角度介绍设计模式在不同编程语言中的应用。目前,几乎每种主流的程序设计语言都有一本甚至多本关于设计模式应用的著作,这里不再一一介绍。

1.2　软件设计模式的概念及意义

软件设计模式是对软件设计经验的总结,是对软件设计中反复出现的设计问题的成功解决

方案的描述。为了记录这些成功的设计经验并方便以后使用，软件设计模式通常包含4个基本要素：模式名称、问题、解决方案以及效果。

模式名称实际上就是一个帮助记忆的名称，是用于软件设计的技术术语，有助于设计者之间的交流。

问题描述了设计者所面临的设计场景，用于告诉设计者应该在什么情况下使用该模式。

解决方案描述了设计的细节，通常会给出方案的原理图示（例如UML的类图、序列图等，也可能是一些示意图）及相关文字说明，如果可能，还会给出一些代码示例，以便对解决方案的深入理解。

效果描述了设计方案的优势和劣势，这些效果通常面向软件的质量属性，例如可扩展性、可复用性等。

软件设计模式的重要意义在于设计复用。设计模式可以使设计者更加方便地借鉴或直接使用已经过证实的成功设计方案，而不必花费时间进行重复设计。一些设计模式甚至提供了显式的类图设计及代码实例，为设计的文档化及软件的开发提供了直接的支持。总之，设计模式可以帮助设计者更快、更好地完成软件系统的设计工作。

1.3 软件设计模式与软件体系结构

软件体系结构是对系统的高层设计，是从一个较高的层次来考虑组成系统的构件、构件之间的连接关系，以及系统需满足的约束等。设计模式可以用于软件体系结构的设计，以实现体系结构级的设计复用。用于软件体系结构的设计模式通常称为架构模式（Architectural Pattern）或体系结构风格（Architectural Style）。

在《Pattern - Oriented Software Architecture》第一卷中，Buschmann等人根据问题的规模或抽象层次将软件设计模式分为三个层次：架构模式（Architectural Patterns）、设计模式（Design Patterns）和习惯用法（Idioms）。

（1）架构模式

架构模式是一种高层模式，用于描述系统级的结构组成、相互关系及相关约束。对架构模式的选择是最基本的设计决策，将决定系统的基本架构，并决定后续的设计及开发活动。

例如，模型 - 视图 - 控制器（Model - View - Controller，MVC）即是一种架构模式，该模式给出了一种交互式系统的架构设计，该模式的主要思想是实现业务逻辑、用户界面的分离。

（2）设计模式

设计模式是中层模式，是针对系统局部设计问题给出的解决方案。一般情况下，人们所讲的设计模式都是指中层模式。设计模式的选择对系统的基本架构没有影响，但在实现架构模式时，则可能采用多种设计模式。

例如，在实现MVC架构模式时，采用的主要设计模式有观察者模式、组合模式和策略模式等（本书后续章节将对这些模式进行详细介绍）。

(3) 习惯用法

架构模式和设计模式被认为是与具体编程语言无关的,而习惯用法则通常被认为是与具体编程语言相关的一种低层模式。习惯用法给出的解决方案通常与具体编程语言的某种语法机制相关。

下面用一个设计实例来解释软件体系结构与软件设计模式之间的关系。假如要设计一个锅炉温度显示软件,要求该软件可根据需要按摄氏或华氏显示温度值,此外,该软件还需按一定的时间间隔将锅炉温度保存到数据库(Database)中。

图 1.1 锅炉温度显示的软件层次架构设计

首先,该软件可以使用层次架构模式进行设计,可分为三个层次,分别为显示层、应用层与数据持久层,如图 1.1 所示。

在进一步的设计中,可以利用观察者模式实现显示层与应用层之间的交互,其中将观察者类 CelsiusGUI 和 FahrenheitGUI 放在显示层,而将被观察者类 Boiler 放在应用层,如图 1.2 所示。

而数据持久层可通过 Martin Fowler 介绍的对象 - 关系元数据映射模式进行实现,这里不再详述。

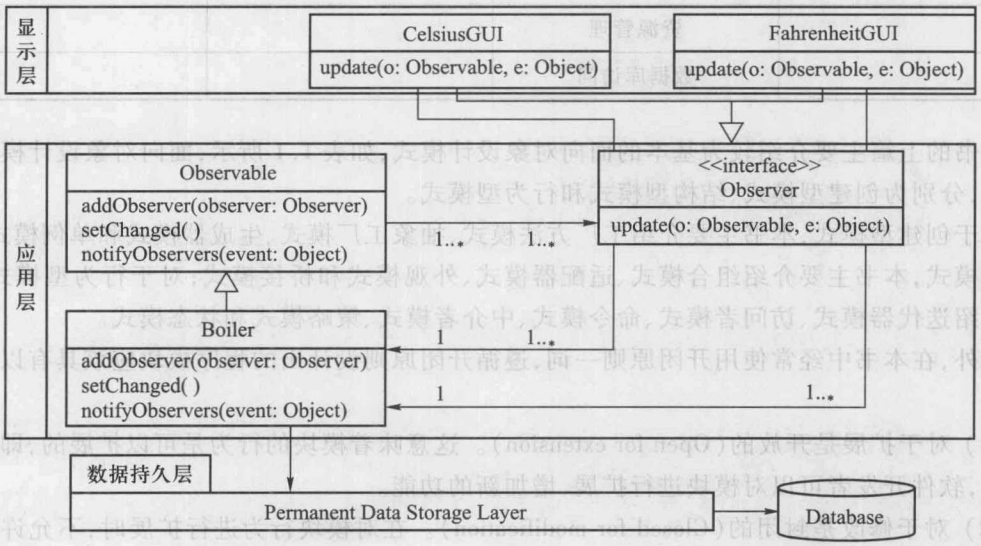

图 1.2 锅炉温度显示软件的层次架构与相关设计模式

1.4 软件设计模式的分类

不同领域总结出的软件设计模式其分类也各不相同,表 1.1 给出了常见的软件设计模式分类。

表 1.1　不同领域的软件设计模式的分类

面向对象	分布式计算	企业应用软件	面向服务的体系结构(SOA)
创建型模式	从混沌到结构	领域逻辑模式	服务设计模式
结构型模式	分布式基础设施	数据源架构模式	服务库设计模式
行为型模式	事件分离与分发	对象 - 关系行为模式	服务组合设计模式
	接口划分	对象 - 关系结构模式	
	组件划分	对象 - 关系元数据映射模式	
	应用控制	Web 表现模式	
	并发	分布模式	
	同步	离线并发模式	
	对象交互	会话状态模式	
	适配与扩展	基本模式	
	模态行为		
	资源管理		
	数据库访问		

本书的上篇主要介绍较为基本的面向对象设计模式，如表 1.1 所示，面向对象设计模式可分为三类，分别为创建型模式、结构型模式和行为型模式。

对于创建型模式，本书主要介绍工厂方法模式、抽象工厂模式、生成器模式和单例模式；对于结构型模式，本书主要介绍组合模式、适配器模式、外观模式和桥接模式；对于行为型模式，本书主要介绍迭代器模式、访问者模式、命令模式、中介者模式、策略模式和状态模式。

此外，在本书中经常使用开闭原则一词，遵循开闭原则设计出的程序模块应该具有以下两个特征：

(1) 对于扩展是开放的(Open for extension)。这意味着模块的行为是可以扩展的，即当需求改变时，软件开发者可以对模块进行扩展，增加新的功能。

(2) 对于修改是封闭的(Closed for modification)。在对模块行为进行扩展时，不允许改动模块中已经存在的类的源代码。

如果一个设计能够同时满足以上两条，则称该设计符合开闭原则。

第2章　创建型软件设计模式

在面向对象编程中，在编写创建对象的代码时，往往需要设置许多条件语句，以便决定在什么条件下，何时、怎样创建某个类的对象。这样，客户类（即这段代码所在的类）将变得比较臃肿，使得对客户类的维护变得困难。因此，有必要将创建对象的责任委托给某个特殊的类。本章要介绍的几个创建型软件设计模式主要就是为了解决上述问题的。

创建型软件设计模式是解决对象创建机制的设计模式。该类设计模式试图根据具体的情况，以适当方式创建对象。

创建型软件设计模式的目标是将一个系统与其对象的创建、组合、表示分离开来，其目的是在哪个对象被创建、谁负责创建对象、怎样创建对象、何时创建对象方面增强灵活性。该模式的主要任务是为客户程序创建对象，而不是由客户程序直接初始化对象。这样，可以大量减少客户程序中对象创建的代码量。

创建型软件设计模式的两个主导思想为，封装了系统使用的具体类的知识，及隐藏这些具体类的实例被创建与结合的细节。

本章将介绍创建型软件设计模式中的简单工厂方法模式、工厂方法模式、抽象工厂模式、生成器模式和单例模式等。

2.1　工厂方法与抽象工厂模式

人类进入工业社会以来，遍布世界各地的各类工厂将生产各类产品的任务从传统的家庭劳动中分离了出来，使得人们实现了更为专业的社会分工。因此，工厂的意义就在于实现了责任分离。本节所述的工厂方法模式在软件设计与开发中所起的作用除了责任分离，还有其他的优点。

为了便于学习，本节将首先通过一个实际的软件设计实例介绍工厂方法的应用场景，然后再分别介绍简单工厂方法模式、工厂方法模式和抽象工厂模式。

2.1.1　工厂方法应用场景

【例2.1】 假如要设计一个汽车保险管理程序。汽车保险分为很多险种，例如人身伤亡（Body Injur）、碰撞（Collision）、驾驶员本身伤亡（Person Injur）、财产损失（Property）、综合险（Com）等。如果一个应用知道它所需要的准确功能，它可以从客户类的主方法中直接初始化类结构体中的某个子类，并且调用该类提供的功能。例如，在准确知道要初始化的类为 Body Injur 时，汽车保险管理程序调用该类的情况如图2.1所示。

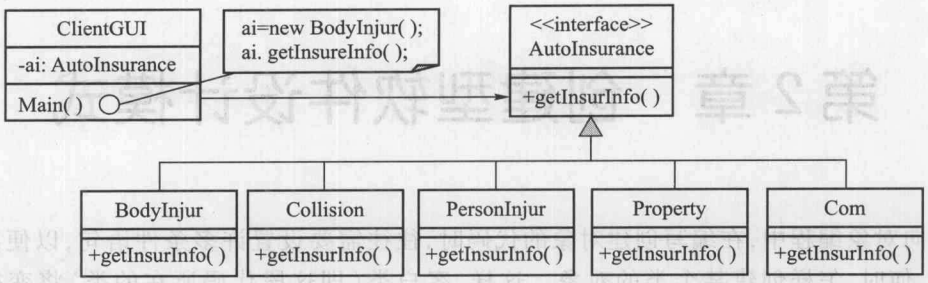

图 2.1　在准确知道要初始化的类为 BodyInjure 的情况

可是,通常一个应用只知道它需要访问一个类的等级结构中的某个类,但是不知道需要准确地访问哪一个类,这是因为合适的类的选择可能依赖于以下几个因素:① 程序的运行状态;② 配置设置;③ 运行时客户的输入。在这种情况下,该应用需要实现从一个类的层次结构中选择合适的类的选择标准,如图 2.2 所示。

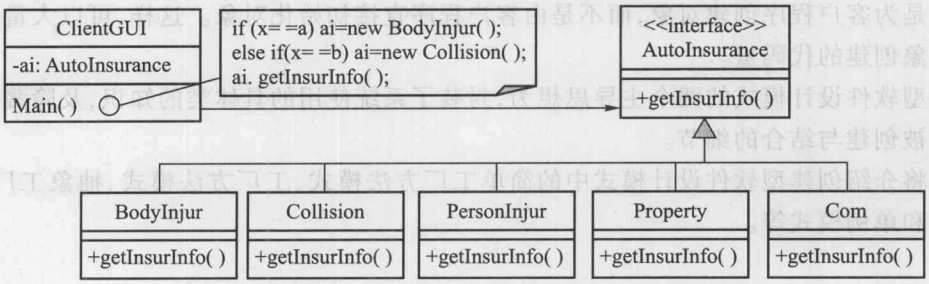

图 2.2　在不知道到底要初始化哪个类的情况

在图 2.2 所示的主方法 main( ) 中,直接用条件语句选择初始哪个类的方法将会使程序比较乱,降低了程序的可读性。一种有效的改善方法是在 ClientGUI 类中单独写一个方法 createObj( ),用于创建对象。该方法封装了所有不同的创建对象的条件,见图 2.3 所示。

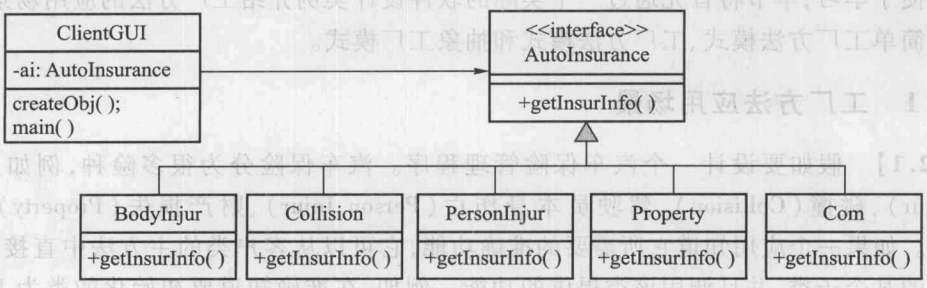

图 2.3　将创建类的对象的责任封装在一个单独的方法中