第3章

设计模式概述

视频讲解

本章导学

随着面向对象技术的发展和广泛应用，设计模式不再是一个新兴名词，它已逐步成为系统架构人员、设计人员、分析人员以及实现系统的程序员所需掌握的基本技能之一。

设计模式已广泛应用于面向对象系统的设计和开发，成为面向对象技术的一个重要组成部分。当人们在特定的环境下遇到特定类型的问题时，可以采用他人已使用过的一些成功的解决方案，一方面降低了分析、设计和实现的难度；另一方面可以使得系统具有更好的可重用性和灵活性。

本章的重点在于掌握设计模式的定义、基本要素和分类，了解 GoF 23 种设计模式并理解设计模式的优点。

本章的难点在于理解设计模式的基本要素及其每一个要素的作用，掌握设计模式的分类方式以及各类设计模式的异同。

设计模式发展重要等级：★★★☆☆

设计模式定义重要等级：★★★★★

设计模式分类重要等级：★★★★☆

3.1 设计模式的诞生与发展

与很多其他软件工程技术一样，设计模式起源于建筑领域，它是对前人经验的总结，为后人设计与开发基于面向对象的软件提供指导方针和成熟的解决方案。

3.1.1 模式的诞生与定义

模式起源于建筑业而非软件业，模式(Pattern)之父——美国加利福尼亚大学环境结构中心研究所所长 Christopher Alexander 博士用了约 20 年的时间，对舒适住宅和周边环境进行了大量的调查和资料收集工作，发现人们对舒适住宅和城市环境存在一些共同的认同规律。他在其经典著作 *A Pattern Language*：*Towns*，*Buildings*，*Construction*(见图 3-1)中把这些认同规律归纳为 253 个模式，对每一个模式都从 Context(模式可适用的前提条

件)、Theme 或 Problem(在特定条件下要解决的目标问题)、Solution(对目标问题求解过程中各种物理关系的表述)三个侧面进行描述,并给出了从用户需求分析到建筑环境结构设计直至经典实例的过程模型。

图 3-1　Christopher Alexander 及其著作封面

在 Alexander 的另一部经典著作《建筑的永恒之道》中,他提到“每个建筑、每个城市都是由称作模式的一定整体组成的,而且一旦我们以建筑的模式来理解建筑,我们就有了考察它们的方法,这一方法产生了所有的建筑,产生了一个城市的所有相似部分以及所有同类物理结构中的各部分”“每一模式就是一个规则,它描述了它所限定的整体以及你所必须要做的事情”“模式以成千上万次的重复进入世界,因为成千上万的人们共同使用具有这些模式的语言”“在哥特式教堂中,中殿侧面与平行于它的侧廊相连”等。

Alexander 给出了关于模式的经典定义:每个模式都描述了一个在我们的环境中不断出现的问题,然后描述了该问题的解决方案的核心,通过这种方式,我们可以无数次地重用那些已有的解决方案,无须再重复相同的工作。这个定义可以简单地用一句话表示:A pattern is a solution to a problem in a context(模式是在特定环境中解决问题的一种方案)。

在 Alexander 研究模式以前,人们注重研究的是高质量、高效率、低成本的开发方案,而 Alexander 的模式注重的是“什么是最好的、成功的”系统。为了找出最优解决方案,Alexander 用了约 20 年时间对现存物进行比较分析,他的贡献主要体现在两方面:其一是集既往之大成——它概括归纳了迄今为止各种风格建筑师的共同设计规则,给东西方、古代派、现代派建筑设计与城市规划提供了共同的语言和准则;其二是他不仅给出了方法,还给出了最优解决方案。

模式可以应用于不同的领域,建筑领域有建筑模式,桥梁领域也有桥梁模式等。当一个领域逐渐成熟的时候,自然会出现很多模式。因为模式是一种指导,在一个良好的指导下,有助于我们设计一个优良的解决方案,达到事半功倍的效果,而且会得到解决问题的最佳办法。

3.1.2 软件模式

1990年,软件工程界开始关注Christopher Alexander等在这一住宅、公共建筑与城市规划领域的重大突破,最早将该模式的思想引入软件工程方法学的是以“四人组(Gang of Four,GoF,分别是Erich Gamma、Richard Helm、Ralph Johnson和John Vlissides)”自称的四位著名软件工程学者,他们在1994年归纳发表了23种在软件开发中使用频率较高的设计模式,旨在用模式来统一沟通面向对象方法在分析、设计和实现间的鸿沟。

GoF将模式的概念引入软件工程领域,这标志着软件模式的诞生。软件模式是将模式的一般概念应用于软件开发领域,即软件开发的总体指导思路或参照样板。软件模式并非仅限于设计模式,还包括架构模式、分析模式和过程模式等,实际上,在软件生存期的每一个阶段都存在着一些被认同的模式。

软件模式可以认为是对软件开发这一特定“问题”的“解法”的某种统一表示,它和Alexander所描述的模式定义完全相同,即软件模式等于一定条件下出现的问题以及解法。软件模式的基础结构由4个部分构成:问题描述、前提条件(环境或约束条件)、解法和效果,如图3-2所示。

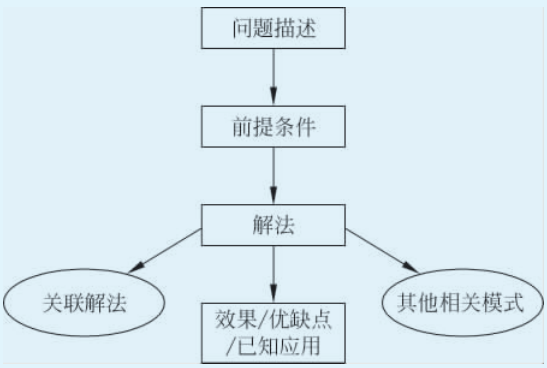

图3-2 软件模式基本结构

软件模式与具体的应用领域无关,在模式发现过程中需要遵循大三律(Rule of Three),即只有经过三个以上不同类型(或不同领域)的系统的校验,一个解决方案才能从候选模式升格为模式。

3.1.3 设计模式的发展

在软件模式领域,目前研究最为深入的是设计模式。下面是软件设计模式的一个简单发展史:

(1) 1987年,Kent Beck和Ward Cunningham借鉴Alexander的模式思想在程序开发中开始应用一些模式,并且在1987年的OOPSLA(Object-Oriented Programming, Systems, Languages & Applications,面向对象编程、系统、语言和应用大会)会议上发表了他们的成果,不过,他们的研究在当时并没有引起热潮。

(2) 1990年,OOPSLA与ECOOP(European Conference on Object-Oriented

Programming,欧洲面向对象编程大会)在加拿大的渥太华联合举办,在由 Bruce Anderson 主持的 Architectural Handbook 研讨会中,Erich Gamma 和 Richard Helm 等人开始讨论有关模式的话题。“四人组”正式成立,并开始着手进行设计模式的分类整理工作。

(3) 在 1991 年的 OOPSLA 中,Bruce Anderson 主持了首次针对设计模式的研讨会,Gamma 和 Johnson 等人再次就设计模式展开讨论。同年,Erich Gamma 完成了他在瑞士苏黎世大学的博士论文,其论文题目为 *Object-Oriented Software Development based on ET++: Design Patterns, Class Library, Tools*,Peter Coad 和 James Coplien 等也开始进行有关模式的研究。

(4) 在 1992 年的 OOPSLA 上,Anderson 再度主持研讨会,模式已经逐渐成为人们讨论的话题。在研讨会中,伊利诺伊大学教授 Ralph Johnson 发表了模式与应用框架关系的论文 *Documenting Framework Using Patterns*,同年 Peter Coad 在国际权威计算机期刊 *Communications of ACM* 上发表文章 *Object-oriented patterns*,该文包含了与 OOAD 相关的 7 个模式。

(5) 1993 年,Kent Beck 和 Grady Booch 赞助了第一次关于设计模式的会议,这次会议邀请了 Richard Helm、Ralph Johnson、Ward Cunningham、James Coplien 等人参加,会议在美国中部科罗拉多(Colorado)州的落基山(Rocky Mountain)下举行,共同讨论如何将 Alexander 的模式思想与 OO(面向对象技术)结合起来。他们决定以 Gamma 的研究成果为基础继续努力研究下去,这个设计模式研究组织发展成为著名的 Hillside Group(山边小组)研究组。

(6) 1994 年,由 Hillside Group 发起,在美国伊利诺伊州(Illinois)的 Allerton Park 召开了第一届关于面向对象模式的世界性会议,名为 PLoP(Pattern Languages of Programs,编程语言模式会议),简称 PLoP'94。

(7) 1995 年,PLoP'95 仍在伊利诺伊州的 Allerton Park 举行,共有 70 多人参加,论文题目比前一年更加多样化,包括 Web 界面模式等,其论文由 John Vlissides 等人负责编辑成书并发行上市。同年发生了设计模式领域里程碑性的事件,“四人组”出版了 *Design Patterns: Elements of Reusable Object-Oriented Software*(《设计模式:可复用面向对象软件的基础》)一书,该书成为 1995 年最抢手的面向对象书籍,也成为设计模式的经典书籍。该书的出版也意味着设计模式正式成为软件工程领域一个重要的研究分支。

(8) 从 1995 年至今,设计模式在软件开发中得以广泛应用,在 Sun 的 Java SE/Java EE 平台和 Microsoft 公司的.NET 平台设计中就应用了大量的设计模式,同时也诞生了越来越多的与设计模式相关的书籍和网站,设计模式也作为一门独立的课程或作为软件体系结构等课程的重要组成部分出现在国内外研究生和大学教育的课堂上。

在设计模式领域,狭义的设计模式就是指 GoF 的《设计模式:可复用面向对象软件的基础》一书中包含的 23 种经典设计模式,不过设计模式不仅仅只有这 23 种,随着软件开发技术的发展,越来越多的新模式不断诞生并得以广泛应用。本书将主要围绕 GoF 23 种模式进行讲解。

3.2 设计模式的定义与分类

设计模式的出现可以让我们站在前人的肩膀上,通过一些成熟的设计方案来指导新项目的开发和设计,更加方便地复用成功的设计和体系结构。

3.2.1 设计模式的定义

设计模式(Design Pattern)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,使用设计模式是为了可重用代码、让代码更容易被他人理解、提高代码的可靠性。

3.2.2 设计模式的基本要素

设计模式一般有如下几个基本要素:模式名称、问题、目的、解决方案、效果、实例代码和相关设计模式,其中的关键元素包括以下四个方面:

1. 模式名称

模式名称(Pattern name)通过一两个词来描述模式的问题、解决方案和效果,以便更好地理解模式并方便开发人员之间的交流,绝大多数模式都是根据其功能或模式结构来命名的。在学习设计模式时,首先应该准确记忆该模式的中英文模式名,在已有的类库中,很多使用了设计模式的类名通常包含了所使用的设计模式的模式名称,如果一个类类名为XXXAdapter,则该类是一个适配器类,在设计时使用了适配器模式,如果一个类类名为XXXFactory,则该类是一个工厂类,它一定包含了一个工厂方法用于返回一个类的实例对象。

2. 问题

问题(Problem)描述了应该在何时使用模式,它包含了设计中存在的问题以及问题存在的原因。这些问题有些是一些特定的设计问题,如怎样使用对象封装状态或者使用对象表示算法等,也可能是系统中存在不灵活的类或对象结构,导致系统可维护性较差。有时候,在模式的问题描述部分可能会包含使用该模式时必须满足的一系列先决条件。如在使用桥接模式时系统中的类必须存在两个独立变化的维度,在使用组合模式时系统中必须存在整体和部分的层次结构等。在对问题进行描述的同时实际上就确定了模式所对应的使用环境以及模式的使用动机。

3. 解决方案

解决方案(Solution)描述了设计模式的组成成分,以及这些组成成分之间的相互关系,各自的职责和协作方式。模式是一个通用的模板,它们可以应用于各种不同的场合,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(类或对象组合)来解决这个问题。在学习设计模式时,解决方案通过类图和核心代码来加以说明,对于每一个设计模式,必须掌握其类图,理解类图中每一个角色的意义以及它们之间的关系,同时需要掌握实现该模式的一些核心代码,以便于在实际开发中合理应用设计模式。

4. 效果

效果(Consequences) 描述了模式应用的效果以及在使用模式时应权衡的问题。效果主要包含模式的优缺点分析,我们应该知道,没有一个解决方案是百分之百完美的,在使用设计模式时需要进行合理的评价和选择。一个模式在某些方面具有优点的同时可能在另一方面存在缺陷,因此需要综合考虑模式的效果。在评价效果时,我们通过结合上一章所学的面向对象设计原则来进行分析,如判断一个模式是否符合单一职责原则,是否符合开闭原则等。

除了上述的四个基本要素,完整的设计模式描述中通常还包含该模式的别名(其他名称)、模式的分类(模式所属类别)、模式的适用性(在什么情况下可以使用该设计模式)、模式角色(即模式参与者,模式中的类和对象以及它们之间的职责)、模式实例(通过实例来进一步加深对模式的理解)、模式应用(在已有系统中该模式的使用)、模式扩展(该模式的一些改进、与之相关的其他模式及其他扩展知识)等。

本书将按照以下次序来介绍设计模式:

(1) 模式动机与定义:通过一些简单问题引出模式,了解该模式可以解决的问题,并对模式进行准确的定义(包括中文定义和英文定义)。

(2) 模式结构与分析:模式结构图(类图)及角色分析,理解该模式解决方案的构成以及成分的关系,并结合示例代码或实例对模式结构和角色进行进一步说明。

(3) 模式实例与解析:通过一或两个实例对模式进行深入学习,了解如何在实际开发中应用该模式。在本书中,大部分模式都提供了两个实例,一个来源于现实生活,方便对模式的理解;另一个来源于软件开发。

(4) 模式效果与应用:对每一个模式的优缺点进行分析,学会识别模式的适用场景,了解在已有系统中模式的使用情况。

(5) 模式扩展:模式的一些改进方案,包括模式功能的增强和简化,与其他模式的联用以及模式的变异,还包括与该模式相关的其他扩展知识。

3.2.3　设计模式的分类

设计模式一般有以下两种分类方式。

(1) 根据其目的(模式是用来做什么的)可分为创建型(Creational)、结构型(Structural)和行为型(Behavioral)三种:

① 创建型模式主要用于创建对象,GoF 提供了 5 种创建型模式,分别是工厂方法模式(Factory Method)、抽象工厂模式(Abstract Factory)、建造者模式(Builder)、原型模式(Prototype)和单例模式(Singleton);

② 结构型模式主要用于处理类或对象的组合,GoF 提供了 7 种结构型模式,分别是适配器模式(Adapter)、桥接模式(Bridge)、组合模式(Composite)、装饰模式(Decorator)、外观模式(Facade)、享元模式(Flyweight)和代理模式(Proxy);

③ 行为型模式主要用于描述对类或对象怎样交互和怎样分配职责,GoF 提供了 11 种行为型模式,分别是职责链模式(Chain of Responsibility)、命令模式(Command)、解释器模式(Interpreter)、迭代器模式(Iterator)、中介者模式(Mediator)、备忘录模式(Memento)、观

察者模式(Observer)、状态模式(State)、策略模式(Strategy)、模板方法模式(Template Method)和访问者模式(Visitor)。

(2) 根据范围,即模式主要是用于处理类之间关系还是处理对象之间的关系,可分为类模式和对象模式两种:

① 类模式处理类和子类之间的关系,这些关系通过继承建立,在编译时刻就被确定下来,是属于静态的。

② 对象模式处理对象间的关系,这些关系在运行时刻变化,更具动态性。

根据"合成复用原则",在系统设计时,我们应该尽量用关联关系来取代继承关系,因此大部分模式都属于对象模式,纯的类模式很少。

3.3 GoF 设计模式简介

在 GoF 的经典著作《设计模式:可复用面向对象软件的基础》一书中一共描述了 23 种设计模式,这 23 种模式分别如表 3-1 所示。

表 3-1 GoF 23 种模式一览表

范围\目的	创建型模式	结构型模式	行为型模式
类模式	工厂方法模式	(类)适配器模式	解释器模式 模板方法模式
对象模式	抽象工厂模式 建造者模式 原型模式 单例模式	(对象)适配器模式 桥接模式 组合模式 装饰模式 外观模式 享元模式 代理模式	职责链模式 命令模式 迭代器模式 中介者模式 备忘录模式 观察者模式 状态模式 策略模式 访问者模式

下面简单对 GoF 23 种设计模式进行说明,如表 3-2 所示。

表 3-2 GoF 23 种模式简要说明

模式类别	模式名称	模式说明
创建型模式 (Creational Patterns)	抽象工厂模式 (Abstract Factory)	提供了一个创建一系列相关或相互依赖对象的接口,而无须指定它们具体的类
	建造者模式 (Builder)	将一个复杂对象的构建与它的表示分离,使得同样的构建过程可以创建不同的表示
	工厂方法模式 (Factory Method)	将类的实例化操作延迟到子类中完成,即由子类来决定究竟应该实例化(创建)哪一个类
	原型模式 (Prototype)	通过给出一个原型对象来指明所要创建的对象的类型,然后通过复制这个原型对象的办法创建出更多同类型的对象
	单例模式 (Singleton)	确保在系统中某一个类只有一个实例,而且自行实例化并向整个系统提供这个实例

续表

模式类别	模式名称	模式说明
结构型模式（Structural Patterns）	适配器模式（Adapter）	将一个接口转换成客户希望的另一个接口，从而使接口不兼容的那些类可以一起工作
	桥接模式（Bridge）	将抽象部分与它的实现部分分离，使它们都可以独立地变化
	组合模式（Composite）	通过组合多个对象形成树形结构以表示“整体-部分”的结构层次，对单个对象（即叶子对象）和组合对象（即容器对象）的使用具有一致性
	装饰模式（Decorator）	动态地给一个对象增加一些额外的职责
	外观模式（Facade）	为复杂子系统提供一个统一的入口
	享元模式（Flyweight）	通过运用共享技术有效地支持大量细粒度对象的复用
	代理模式（Proxy）	给某一个对象提供一个代理，并由代理对象控制对原对象的引用
行为型模式（Behavioral Patterns）	职责链模式（Chain of Responsibility）	避免请求发送者与接收者耦合在一起，让多个对象都有可能接收请求，将这些对象连接成一条链，并且沿着这条链传递请求，直到有对象处理它为止
	命令模式（Command）	将一个请求封装为一个对象，从而使得请求调用者和请求接收者解耦
	解释器模式（Interpreter）	描述如何为语言定义一个文法，如何在该语言中表示一个句子，以及如何解释这些句子
	迭代器模式（Iterator）	提供了一种方法来访问聚合对象，而不用暴露这个对象的内部表示
	中介者模式（Mediator）	通过一个中介对象来封装一系列的对象交互，使得各对象不需要显式地相互引用，从而使其耦合松散，而且可以独立地改变它们之间的交互
	备忘录模式（Memento）	在不破坏封装的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态，这样可以在以后将对象恢复到原先保存的状态
	观察者模式（Observer）	定义了对象间的一种一对多依赖关系，使得每当一个对象状态发生改变时，其相关依赖对象皆得到通知并被自动更新
	状态模式（State）	允许一个对象在其内部状态改变时改变它的行为
	策略模式（Strategy）	定义一系列算法，并将每一个算法封装在一个类中，并让它们可以相互替换，策略模式让算法独立于使用它的客户而变化
	模板方法模式（Template Method）	定义一个操作中算法的骨架，而将一些步骤延迟到子类中
	访问者模式（Visitor）	表示一个作用于某对象结构中的各元素的操作，它使得用户可以在不改变各元素的类的前提下定义作用于这些元素的新操作

需要注意的是,这 23 种设计模式并不是孤立存在的,很多模式彼此之间存在联系,如在访问者模式中操作对象结构中的元素时通常需要使用迭代器模式,在解释器模式中定义终结符表达式和非终结符表达式时可以使用组合模式;此外,还可以通过组合两个或者多个模式来设计同一个系统,在充分发挥每一个模式优势的同时使它们可以协同工作,完成一些更复杂的设计工作。

3.4 设计模式的优点

设计模式是从许多优秀的软件系统中总结出的成功的、能够实现可维护性复用的设计方案,使用这些方案将避免我们做一些重复性的工作,而且可以设计出高质量的软件系统。具体来说,设计模式的主要优点如下:

(1) 设计模式融合了众多专家的经验,并以一种标准的形式供广大开发人员所用,它提供了一套通用的设计词汇和一种通用的语言以方便开发人员之间沟通和交流,使得设计方案更加通俗易懂。对于使用不同编程语言的开发和设计人员可以通过设计模式来交流系统设计方案,每一个模式都对应一个标准的解决方案,设计模式可以降低开发人员理解系统的复杂度。

(2) 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,将已证实的技术表述成设计模式也会使新系统开发者更加容易理解其设计思路。设计模式使得重用成功的设计更加容易,并避免那些导致不可重用的设计方案。

(3) 设计模式使得设计方案更加灵活,且易于修改。在很多设计模式中广泛使用了开闭原则、依赖倒转原则、迪米特法则等面向对象设计原则,使得系统具有较好的可维护性,真正实现可维护性的复用。在软件开发中合理使用设计模式,可以使得系统中的一些组成部分在其他系统中得以重用,而且在此基础上进行二次开发很方便。正因为设计模式具有该优点,因此在 JDK 类库、.NET Framework SDK、Struts、Spring 等类库和框架的设计中大量使用了设计模式。

(4) 设计模式的使用将提高软件系统的开发效率和软件质量,且在一定程度上节约设计成本。设计模式是一些通过多次实践得以证明的行之有效的解决方案,这些解决方案通常是针对某一类问题最佳的设计方案,因此可以帮助设计人员构造优秀的软件系统,并可直接重用这些设计经验,节省系统设计成本。

(5) 设计模式有助于初学者更深入地理解面向对象思想,一方面可以帮助初学者更加方便地阅读和学习现有类库(如 JDK)与其他系统中的源代码;另一方面还可以提高软件的设计水平和代码质量。

3.5 本章小结

(1) 模式是在特定环境中解决问题的一种方案。

(2) GoF (Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides)最先将模式的概念引入软件工程领域,他们归纳发表了 23 种在软件开发中使用频率较高的设计模式,

旨在用模式来统一沟通面向对象方法在分析、设计和实现间的鸿沟。

（3）软件模式是将模式的一般概念应用于软件开发领域，即软件开发的总体指导思路或参照样板。软件模式可以认为是对软件开发这一特定“问题”的“解法”的某种统一表示，即软件模式等于一定条件下出现的问题以及解法。

（4）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，使用设计模式是为了可重用代码、让代码更容易被他人理解、提高代码的可靠性。

（5）设计模式一般有如下几个基本要素：模式名称、问题、目的、解决方案、效果、实例代码和相关设计模式，其中的关键元素包括模式名称、问题、解决方案和效果。

（6）设计模式根据其目的可分为创建型、结构型和行为型三种；根据范围可分为类模式和对象模式两种。

（7）设计模式是从许多优秀的软件系统中总结出的成功的、能够实现可维护性复用的设计方案，使用这些方案将避免我们做一些重复性的工作，而且可以设计出高质量的软件系统。

思考与练习

1. 什么是设计模式？它包含哪些基本要素？
2. 设计模式如何分类？每一类设计模式各有何特点？
3. 设计模式具有哪些优点？
4. 请查阅相关资料，了解在 JDK 类库设计中使用了哪些设计模式，在何处使用了何种模式？至少列举两个。
5. 除了设计模式之外，目前有不少人在从事“反模式”的研究，请查阅相关资料，了解何谓“反模式”以及研究“反模式”的意义。