深入浅出

设计模式

郭 峰 著

融入设计模式和原则的代码，是最实用的代码

本书帮助您尽量降低代码之间的依赖，做到灵活多变，适应用户需求的不断变化。

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

深入浅出
设计模式

郭 峰 著

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书总结了许多系统软件在设计开发过程中的难点，力图将设计模式的实际应用与实现原理有机结合起来，破解软件开发人员在学习设计模式过程不能透彻理解并灵活运用设计模式的难题。

所有章节都是先通过具体的示例讲解为什么需要使用某个设计模式，然后讲解该模式的实现原理，最后再通过详细的示例或对很多开源框架进行分析，加深读者对设计模式的理解。

本书适用于中、高级软件设计和开发人员，尤其是已经学习过设计模式但没有收获的开发人员，同时也可作为高校相关专业师生和社会培训班的教材。

图书在版编目（CIP）数据

深入浅出设计模式 / 郭峰著. — 北京： 中国铁道出版社，2013.1

ISBN 978-7-113-15265-9

Ⅰ. ①深… Ⅱ. ①郭… Ⅲ. ①程序设计－模式－研究 Ⅳ. ①TP311

中国版本图书馆 CIP 数据核字（2012）第 214461 号

书　　名：深入浅出设计模式
作　　者：郭　峰　著

策划编辑：荆　波　　**读者服务热线：010-63560056**
责任编辑：荆　波　　**特邀编辑：赵树刚**
责任印制：赵星辰

出版发行：中国铁道出版社（北京市西城区右安门西街 8 号　　邮政编码：100054）
印　　刷：北京新魏印刷厂
版　　次：2013 年 1 月第 1 版　　2013 年 1 月第 1 次印刷
开　　本：787mm×1 092mm　1/16　　印张：35.25　　字数：831 千
书　　号：ISBN 978-7-113-15265-9
定　　价：69.80 元

前 言 Contents

作为一名软件开发人员，在日常的软件开发中，如果因为用户需求不断地发生变化，而造成程序代码需要频繁修改，程序缺陷满天飞，项目经常延期，那么学习并掌握设计模式就显得异常重要了。

对于软件开发来说，最痛苦的事情莫过于系统刚上线，用户的新需求或需求变更就马上又提了出来，开发人员经常会抱怨为什么用户不在上线前就把需求说清楚，但用户就是上帝，抱怨归抱怨，程序还得按照用户的要求进行修改。通常在软件开发公司中，针对上面的这些问题，会出台各种规章制度，设立专职的 QA（质量保证）人员，建立各种质量保证体系，甚至通过 CMMI 认证。可采取了这些措施后才发现，新的问题又会出现了，需求人员抱怨需求文档太难写，设计开发人员抱怨需求人员写的文档看不懂，以至于项目比以前延期的时间更长了，程序缺陷更多了，文档到后期都没人看了。

这是目前国内大多数软件开发项目的真实写照，问题出在哪里呢？即便需求人员能够很好地编写出需求文档，难道需求就不会发生变化了吗？其实用户的需求是在不断发生变化的，既然作为软件开发人员不能避免这些变化，那就要正视这些变化，接受变化，尽量地满足用户需求的变化。要达到上述目的，就只有依靠程序设计，因为只有程序设计得更加灵活了，才能更好地适应需求的变化，才能更少地改动代码，从而产生更少的程序缺陷。

程序如何才能设计得更加灵活呢？这就需要本书介绍的一些面向对象的设计原则，以及各种设计模式，依据这些设计原则来进行程序设计和开发，就有可能使程序设计得更加灵活，能够更好地适应用户需求的复杂多变。

如何学习这些面向对象的设计原则和各种设计模式呢？笔者见过太多开发人员天天捧着一本设计模式的书籍，从头读到尾，问他设计模式都有哪些，讲的头头是道，再问他如何用这些设计模式解决系统中遇到的问题，马上就哑口无言了。也有一些开发人员，买了很多设计模式的书籍，却始终不能入门。

如何学习设计模式

那到底要如何学习设计模式呢？其实学习设计模式和学习其他知识一样，遵循自然界的一般规律，就是一个循序渐进的过程，都要从遇到的问题出发，然后解决这些问题，并将解决办法形成一套方法论，再将方法论推向实际应用。因此本书在编排上，每个章节都是首先从为什么要使用某个设计模式入手，介绍如果不使用这个设计模式会遇到哪些问题，使用了设计模式是如何解决这些问题的，然后再介绍设计模式的实现原理，最后再通过一些软件开发过程中的实际应用或一些开源软件中好的做法，最终使读者能够了解设计模式的来龙去脉，并能将其应用到自己的实际开发中。

为了使读者能够知其然还要知其所以然，本书在介绍设计模式之前，先介绍了一些面向对象的设计原则，这些设计原则，都是前人通过多年设计经验总结出来的，核心思想就是强调尽量降低代码之间的依赖，从而

能够灵活多变，适应用户需求的不断变化，这也是“高内聚、低耦合”思想的具体实现。

在读者掌握了面向对象的设计原则和各种设计模式后，本书最后还综合运用各种设计模式实现了一个MVC框架，该框架主要采用了命令模式、模板方法模式、单例模式、享元模式、代理模式等，使得读者对各种设计模式的综合应用能有一个更加深入的了解。

本书正是按照上述思想来安排各章节内容的，先介绍面向对象的设计原则，再讲解各种设计模式的实现方式，最后通过一个综合示例来演示各种设计模式的应用，从而破解软件开发人员在学习设计模式过程不能透彻理解并灵活运用设计模式的难题。

与同类书相比，本书的优势

1．循序渐进

本书按照人类对事物认识的一般规律，每一章都从软件开发中遇到的实际问题出发，介绍如果不使用这个设计模式会遇到哪些问题，使用了此设计模式是如何解决这些问题的，然后再介绍设计模式的实现原理，最后再讲解一些软件开发过程中的实际应用或一些开源软件中好的做法。

2．引人入胜

本书没有生硬地讲解各种设计模式的实现原理，而是通过一个个小示例，讲解设计模式的具体使用方法，并对一些业界公认设计比较好的开源软件进行分析，使读者不但能够掌握这些开源软件的使用，还可以了解它们的实现原理。

3．实战性强

本书不但每一章节都给出了各种应用设计模式的示例，在最后一个章节，更是通过实现一个类似Struts的MVC框架，使读者能够一步一步地运行各种设计模式，设计自己的底层框架。

4．着重基础

本书没有一上来就讲解各种设计模式，而是先讲解了软件开发遇到的难题，以及前人为此所做的努力，并详细讲解了面向对象的设计原则，这是理解设计模式的基础，基础不扎实，设计模式也就掌握得不够牢靠。

本书包括的内容

本书是笔者在多年项目开发过程中的经验总结，第1章从软件开发遇到的问题讲起，简要介绍了面向对象的设计原则，以及设计模式的划分和学习途径。第2章对设计模式需要用到的工具进行了简要介绍。从第3章开始，依次讲解了创建型模式、结构型模式和行为型模式。第27章通过实现一个MVC框架的综合示例，深入讲解了各种模式的应用。

本书每个章节都是先通过具体的示例讲解为什么需要使用某个设计模式，然后讲解该模式的实现原理，最后再通过详细的示例或对很多开源框架的分析，来加深读者对设计模式的理解。

如何阅读本书

笔者强烈建议读者能够按照本书的章节顺序，一章一章地进行阅读，因为笔者按照人类对事物认识的一般规律，精心编写了各章节的内容，每个章节环环相扣，先介绍面向对象的设计原则，再讲解各种设计模式的实现方式，最后通过一个综合示例来演示各种设计模式的应用，目的是破解软件开发人员在学习设计模式过程不能透彻理解并灵活运用设计模式的难题。

当然如果读者对设计模式已经非常熟悉，只是想查找某一个设计模式的具体用法，也可以单独阅读其中一章的内容。

适合阅读本书的读者

本书适用于中、高级软件设计和开发人员，尤其是已经学习过设计模式但没有收获的开发人员，同时也可作为高校相关专业师生和社会培训班的教材。

作　者

2012 年 9 月

读者意见反馈表

亲爱的读者：

感谢您对中国铁道出版社的支持，您的建议是我们不断改进工作的信息来源，您的需求是我们不断开拓创新的基础。为了更好地服务读者，出版更多的精品图书，希望您能在百忙之中抽出时间填写这份意见反馈表发给我们。随书纸制表格请在填好后剪下寄到：北京市西城区右安门西街8号中国铁道出版社综合编辑部 荆波 收（邮编：100054）。或者采用传真（010-63549458）方式发送。此外，读者也可以直接通过电子邮件把意见反馈给我们，E-mail地址是：jb@163.jb18803242@yahoo.com.cn。我们将选出意见中肯的热心读者，赠送本社的其他图书作为奖励。同时，我们将充分考虑您的意见和建议，并尽可能地给您满意的答复。谢谢！

所购书名： ______________________

个人资料：

姓名：__________ 性别：__________ 年龄：__________ 文化程度：__________

职业：__________ 电话：__________ E-mail：__________

通信地址：______________________ 邮编：__________

您是如何得知本书的：

□书店宣传 □网络宣传 □展会促销 □出版社图书目录 □老师指定 □杂志、报纸等的介绍 □别人推荐

□其他（请指明）______________________

您从何处得到本书的：

□书店 □邮购 □商场、超市等卖场 □图书销售的网站 □培训学校 □其他

影响您购买本书的因素（可多选）：

□内容实用 □价格合理 □装帧设计精美 □带多媒体教学光盘 □优惠促销 □书评广告 □出版社知名度

□作者名气 □工作、生活和学习的需要 □其他

您对本书封面设计的满意程度：

□很满意 □比较满意 □一般 □不满意 □改进建议

您对本书的总体满意程度：

从文字的角度 □很满意 □比较满意 □一般 □不满意

从技术的角度 □很满意 □比较满意 □一般 □不满意

您希望书中图的比例是多少：

□少量的图片辅以大量的文字 □图文比例相当 □大量的图片辅以少量的文字

您希望本书的定价是多少：

本书最令您满意的是：

1.

2.

您在使用本书时遇到哪些困难：

1.

2.

您希望本书在哪些方面进行改进：

1.

2.

您需要购买哪些方面的图书？对我社现有图书有什么好的建议？

您更喜欢阅读哪些类型和层次的计算机书籍（可多选）？

□入门类 □精通类 □综合类 □问答类 □图解类 □查询手册类 □实例教程类

您在学习计算机的过程中有什么困难？

您的其他要求：

目 录 Contents

第 18 章　模板方法模式（Template Method）

第 19 章　中介者模式（Mediator）

第 20 章　访问者模式（Visitor）

第 21 章　职责链模式（Chain of Responsibility）

第 22 章　状态模式（State）

第 27 章　综合应用设计模式实现 MVC 框架

第 1 章　如何学习设计模式

作为一名软件开发人员，在日常的软件开发中，如果因为用户需求不断发生变化，而造成程序代码频繁的修改，程序缺陷满天飞，项目经常延期，那学习并掌握设计模式就显得异常重要了。实际开发中，用户的需求总是在变化，因此开发人员需要掌握设计模式，把遇到的问题化解于无形。

1.1　软件开发遇到的问题

软件开发真正流行起来，也就是最近这 20 年的时间，在软件项目中，通常追求的目标是：高效率、高品质、低成本，但是常常会遇到这样那样的问题，比如：

- 项目常常延期。
- 总是觉得缺少足够的资源。
- 提交给用户的产品 bug 满天飞。
- 员工常常抱怨自己的待遇不公平。
- 不同项目间在效率、质量、客户满意度等方面差异很大。

为什么会造成上述问题呢？通常有以下这些原因：

- 业务越来越复杂，而需求变更却很难管控。
- 技术更新频繁，而人员水平却参差不齐。
- 系统的功能越来越强大，而 bug 却越来越多。
- 文档越来越多，而版本管理混乱。
- 人员越来越多，而流动越来越大。
- 项目越来越多，做事情的方法各不相同。

在一般的软件开发公司中，针对上面介绍的这些原因，通常的做法是出台各种规章制度，设立专职的 QA（质量保证）人员，建立质量保证体系，如图 1.1 所示。

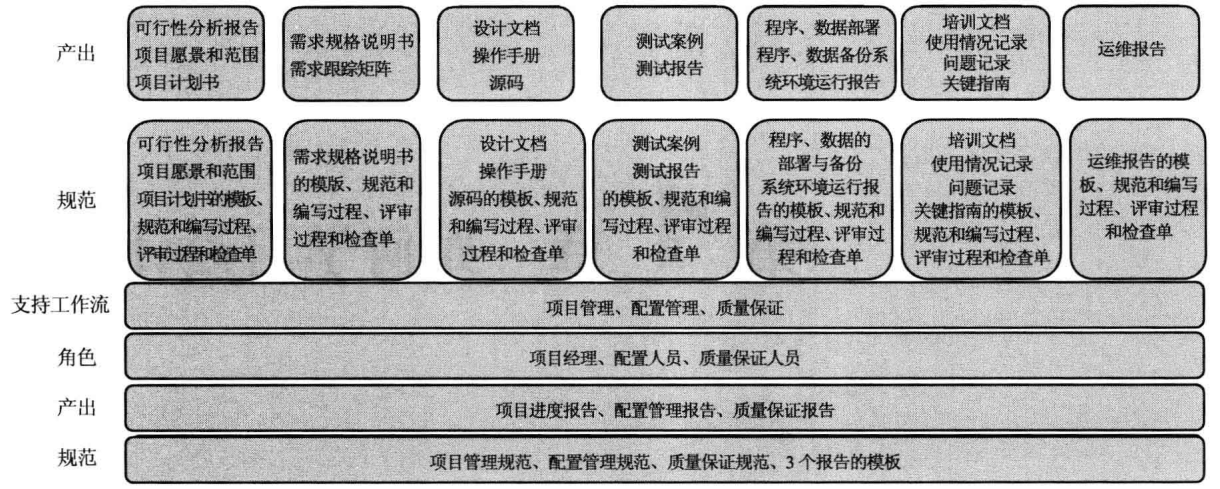

图 1.1　质量保证体系

各种质量保证的制度出台了，也有专人天天检查，可问题更多了，需求人员抱怨需求文档太难写，设计开发人员抱怨需求人员写的文档看不懂，以至于项目比以前延期的时间更长了，程序缺陷更多了，文档到后期都没人看了。这是目前国内大多数软件开发项目的真实写照，问题出在哪里呢？即便需求人员能够很好地编写出需求文档，难道需求就不会发生变化了吗？

1.2　面向对象的设计原则

上一节讲述了目前软件开发存在的问题，该如何解决这些问题？从前面的分析可以看出，用户的需求是在不断发生变化的，既然作为软件开发人员不能拒绝这些变化，那就要正视这些变化，接受变化，尽量地满足用户需求的变化，要达到上述目的，就只有依靠程序设计，因为只有程序设计得更加灵活了，才能更好地适应需求的变化，才能更少地改动代码，从而产生更少的程序缺陷。

程序如何才能设计得更加灵活呢？随着经验的积累，程序设计人员总结出了一些面向对象的设计原则，大体上可以分为：单一职责原则、开闭原则、依赖注入原则、里氏替换原则、迪米特原则、接口分离原则和优先使用组合而不是继承原则。依据这些设计原则来进行程序设计和开发，就有可能使程序设计得更加灵活，能够更好地适应用户需求的复杂多变。下面来详细地讲解一下这些设计原则：

1．单一职责原则

单一职责原则（SRP，Single Responsibility Principle）的意思是说系统中的每一个对象都应该只有一个单独的职责，而所有对象所关注的就是自身职责的完成。每个类应该只有一个职责，对外只能提供一种功能，而引起类变化的原因应该只有一个。

这是软件开发人员经常会违反的一个设计原则，通常为了省事，开发人员会将很多功能集中在一个类里面，从而造成这个类庞大无比，难于维护。

2．开闭原则

开闭原则（OCP，Open for Extension，Closed for Modification Principle）的核心思想就是：一个对象对扩展开放，对修改关闭。它的意思是说对类的改动是通过增加代码进行的，而不是改动现有的代码。通俗地说，软件开发人员一旦写出了可以运行的代码，就不应该去改变它，而是要保证它能一直运行下去。

这也是软件开发人员经常会违反的一个设计原则，为了满足用户的新需求，需要修改或扩充原来的代码，这样造成的后果就是如果这个类中有一个功能要进行修改，就很可能会引起其他功能的变动，从而产生不可预知的后果，在软件开发中，经常会出现为了修改一个程序缺陷而产生新的程序缺陷的问题，就是违法了这一原则造成的。

3．依赖注入原则

依赖注入原则（DIP，Dependence Inversion Principle）的意思是说要依赖于抽象，不要依赖于具体的实现。在软件开发中，所有的类如果需求调用其他的类，就应该调用该类的接口或抽象类，而不是直接调用该类的实现类。

依赖注入原则在 Spring 中得到了充分的应用，这样才能保证 Spring 的灵活性，因此在采用 Spring 开发代码时，要针对接口编程，而不是针对实现编程。

4．里氏替换原则

里氏替换原则（LSP，Liskov Substitution Principle）的意思是说在任何抽象类出现的地方都可以用它的实现类来替代。采用里氏替换原则可以更好地使用继承。

5．迪米特原则

迪米特原则（LOD，Law of Demeter）的意思是说一个对象应当对其他对象尽可能少的了解，从而能够降低各个对象之间的耦合，提高系统的可维护性。

在程序设计中，各个模块之间互相调用时，通常都会提供一个统一的接口来实现，这样其他模块就不需要了解另一个模块的内部实现细节，这样当一个模块内部的实现发生改变时，不会影响其他模块的使用。

6．接口分离原则

接口分离原则（ISP，Interface Segregation Principle）的意思是说不应该强迫客户程序依赖它们不需要使用的方法。

在软件开发中，开发人员经常会将很多对外提供的方法封装到一个类中实现，然后提供给其他模块使用，这就违反了接口分离原则，一个接口应该只提供一种对外功能，不应该把所有的操作都封装到一个接口当中。