

设计模式

其实很简单

刘径舟 张玉华 等编著

设计模式很难？NO！设计模式很简单？YES！

抛弃枯燥乏味、晦涩难懂的讲解，以轻松愉悦的方式学习吧！

设定23个场景教学，趣谈23种设计模式，妙趣横生

配备36幅情景插画，提供35幅模式结构图，直观明了

体验模式的精髓，领略设计的优雅，轻松应对项目的考验



清华大学出版社

设计模式 其实很简单

刘径舟 张玉华 等编著



清华大学出版社

北 京

内 容 简 介

本书以提高开发者代码设计能力为指导思想,以 Java 代码整合常见的设计模式为主线,通过讲解面向对象程序开发中最常见的 23 个设计模式典型模块和众多代码案例,详细介绍了面向对象中设计模式的思想 and 运用。

本书共 25 章,分为 5 篇。内容涵盖了模式设计的六大法则:单一职责原则、里氏代换原则、开放封闭原则、依赖倒转原则、合成/聚合复用原则、迪米特法则,同时又介绍了 23 种模式:工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式、结构型模式、桥接模式、适配器模式、装饰模式、组合模式、享元模式、外观模式、代理模式、行为模式、模板方法模式、策略模式、状态模式、观察者模式、备忘录模式、中介者模式、命令模式、访问者模式、职责链模式、迭代器模式、解释器模式等。

本书语言幽默,内容丰富,实例典型,实用性强,适合已经有了开发基础、想要进一步深入学习设计模式的人员阅读,尤其适合有进一步提高代码设计能力需要的开发者阅读。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

设计模式其实很简单/刘径舟,张玉华等编著. —北京:清华大学出版社,2013.7
ISBN 978-7-302-31811-8

I. ①设… II. ①刘… ②张… III. ①软件设计 IV. ①TP311.5

中国版本图书馆 CIP 数据核字(2013)第 063038 号

责任编辑:夏兆彦

封面设计:欧振旭

责任校对:胡伟民

责任印制:刘海龙

出版发行:清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址:北京清华大学学研大厦 A 座 邮 编:100084

社 总 机:010-62770175 邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者:清华大学印刷厂

装 订 者:北京市密云县京文制本装订厂

经 销:全国新华书店

开 本:185mm×260mm 印 张:18.5 字 数:465 千字

版 次:2013 年 7 月第 1 版 印 次:2013 年 7 月第 1 次印刷

印 数:1~3000

定 价:39.80 元

产品编号:049201-01

前言

电商、社交网络、移动互联网……这一个个名词成为近几年来互联网发展的代名词，而在背后推动这种高速发展的就是技术。敏捷开发是互联网企业进行产品开发时常用的一种模式，一个产品从制定需求到上线甚至在一周之内就要完成，毕竟快竞争对手一步才能抓住更多的用户。在这种情况下，一些久经考验的架构模式被广泛使用起来，当确定一个产品的架构方案时，“设计模式”一词被越来越多地提出来，甚至一些公司既定的框架就是某一个设计模式的体现。

那么学习并且灵活使用设计模式就成为了一个开发人员必须经历的过程，而一本结合 Java（大型公司最青睐的语言）语言特点讲述设计模式的图书，则可以让读者快速地了解设计模式。一个设计模式的合理运用可以使产品的开发周期大大缩短，同时大幅度降低产品维护的成本。同时，对于设计模式的深入理解还能帮助开发人员更好地理解面向对象编程的思想，延长一名开发人员的职业生命。

本书便是一本专门针对学习设计模式的“菜鸟”级别的读者的书。本书以提高开发者代码设计能力为指导思想，以 Java 代码整合常见的设计模式为主线，通过讲解面向对象程序开发中最常见的 23 个设计模式典型模块和众多代码案例，详细介绍了面向对象中设计模式思想和运用。

本书有何特色？

1. 语言风趣幽默，通过实际例子循序渐进

一提到软件开发的相关书籍，“晦涩难懂”是大部分人的第一印象。在本就繁重和枯燥的工作之余，大部分程序员都希望可以以一种愉悦的方式来给自己充电。本书使用幽默诙谐的语言加上生动的小故事，把整个设计模式的学习过程串联起来，通过一个个实际的例子，将设计模式的精髓表现出来，大大方便了读者理解本书内容。

2. 详细讲解设计模式原则，追踪设计模式的起源

设计模式并不是凭空产生的，所有设计模式都满足某种软件设计的准则，这些准则并不是某个人强加在软件开发过程中的，而是多年来许许多多经验的积累以及严格的证明。理解这些准则以后，就更加明白设计模式的可贵之处，也为灵活使用这些模式打下了坚实的基础。

3. 用最简单的代码来描述问题

软件开发相关的书籍，实例代码往往非常重要，但是这些代码有时又是一把双刃剑，

帮助读者理解问题的同时，也可能因为过于复杂的逻辑导致读者陷入困惑。本书在描述实例时使用尽量简单的代码，让读者可以一目了然，将更多的精力投入到对于设计模式思想的认知当中。代码是开发人员的最终产出，它们同样可以帮助读者理解文字难以表达的内容。

4. 条目清晰便于查找

设计模式的特点是在实践中逐渐融会贯通，而不是死记硬背记住那些定义。本书不仅仅是一本学习设计模式的书籍，同时也可以作为设计模式的工具书，当需要时可以随时翻阅找到相应的模式以及该模式对应的类图，帮助读者在实际设计时快速地抓住重点。

本书内容安排

第1篇 六大设计原则（第1章）

本篇主要内容包括讲解设计模式中必须掌握的六大设计原则：单一职责原则、里氏代换原则、开放-闭原则、依赖倒转原则、合成/聚合复用原则和迪米特法则。通过本篇的学习，读者可以了解设计模式最初是如何产生的，还会了解在什么情况下出现了 23 种设计模式。

第2篇 创建型模式（第2~6章）

本篇主要内容包括：工厂方法模式、抽象工厂模式、单例模式、原型模式、建造者模式。通过本篇的学习，读者可以掌握创建型模式的工作原理和使用技巧。

第3篇 结构型模式（第7~13章）

本篇主要内容包括：适配器模式、桥接模式、组合模式、享元模式、装饰模式、外观模式、代理模式。通过本篇的学习，读者可以掌握结构型模式的工作原理和使用技巧。

第4篇 行为模式（第14~24章）

本篇主要内容包括：模板方法模式、策略模式、状态模式、观察者模式、备忘录模式、中介者模式、命令模式、访问者模式、职责链模式、迭代器模式、解释器模式。通过本篇的学习，读者可以掌握行为模式的工作原理和使用技巧。

第5篇 模式总结（第25章）

本篇主要内容包括：各种模式之间的关系、设计模式思考图、架构师职责等。通过本篇的学习，读者可以从大体上了解各种模式的交叉使用技巧，也会了解如何用 UML 来实现模式的类图，以及作为架构师应该熟悉的各种技术方向。

适合阅读本书的读者

□ 需要设计模式对自己的开发进行指引的 Java 程序员；

- ❑ 广大互联网开发工程师；
- ❑ 各种开发语言的程序员；
- ❑ 对面向对象编程需要深入了解的程序员；
- ❑ 希望提高项目开发水平的人员；
- ❑ 专业培训机构的学员；
- ❑ 软件开发项目经理。

阅读本书的建议

- ❑ 没有 Java 语言基础的读者，可以先了解一些基本语法，然后结合本形式书对 Java 语言进行深入学习。
- ❑ 对于每个设计模式的定义不要去死记硬背，而是在理解的基础上形成自己的想法。
- ❑ 每看完一个模式，可以回想一下之前自己做过或接触过的项目，可以怎样把模式应用到这些项目当中，并且思考一下应用模式后会给系统带来怎样的好处。
- ❑ 代码最好都能亲自动手试验，并考虑如何能进一步完善这些代码。

本书作者

本书主要由刘径舟和张玉华主笔编写。其他参与编写的人员有陈世琼、陈欣、陈智敏、董加强、范礼、郭秋滢、郝红英、蒋春蕾、黎华、刘建准、刘霄、刘亚军、刘仲义、柳刚、罗永峰。在此一并表示感谢。

阅读本书的过程中如有疑问，请和我们联系。E-mail: bookservice2008@163.com。

编者

目 录

第 1 篇 六大设计原则

第 1 章 设计模式的定义及设计原则	2
1.1 初谈设计模式	2
1.2 单一职责原则	3
1.2.1 “超人”也是人	3
1.2.2 简单也很难做到	5
1.2.3 帮“超人”减负	5
1.2.4 单一职责原则能否扩展	7
1.2.5 小结与思考	9
1.3 里氏代换原则	10
1.3.1 鲨鱼到底是不是鱼	10
1.3.2 子类与父类	11
1.3.3 里氏代换原则的定义	11
1.3.4 里氏代换原则的精髓	11
1.3.5 小结与思考	14
1.4 开放-封闭原则	14
1.4.1 添加而不是修改	14
1.4.2 应用广泛的开放-封闭原则	15
1.4.3 开放-封闭原则的意义所在	17
1.4.4 小结与思考	20
1.5 依赖倒转原则	20
1.5.1 再谈面向对象	20
1.5.2 依赖倒转原则的定义	20
1.5.3 依赖倒转原则的重要意义	21
1.5.4 小结与思考	23
1.6 合成/聚合复用原则	24
1.6.1 合成/聚合复用原则的定义	24
1.6.2 何时使用合成/聚合复用原则	24
1.6.3 合成/聚合复用原则的优缺点	28
1.6.4 小结与思考	28

1.7 迪米特法则	28
1.7.1 不要和陌生人说话	28
1.7.2 迪米特法则的定义	29
1.7.3 耦合性问题的讨论	29
1.7.4 小结与思考	33
第 2 篇 创建型模式	
第 2 章 工厂方法模式——面向对象的工业时代	36
2.1 代码工业时代初期	36
2.2 如何烤出各种各样的面包	39
2.3 代码工业时代的二次革命	41
2.4 小结与思考	43
第 3 章 抽象工厂模式——如果想开分厂	45
3.1 产品单一打不开市场	45
3.2 面包与披萨可以兼得	46
3.3 投资需谨慎	50
3.4 建厂方案大评比	51
3.5 小结与思考	52
第 4 章 单例模式——非你莫属	54
4.1 婚姻法出台	54
4.2 老婆只能娶一个	54
4.3 娶老婆的代码实现	57
4.4 婚礼现场直播怎么办	62
4.5 小结与思考	65
第 5 章 建造者模式——丰富的生产线	66
5.1 化学老师的作业	66
5.2 神奇的化学实验	66
5.3 老师来指挥	71
5.4 做实验不是烤面包	75
5.5 小结与思考	76
第 6 章 原型模式——打印报名信息	77
6.1 考证达人	77
6.2 考试太多报名信息别打错	77
6.3 线程也要安全生产	84
6.4 别被 Java 欺骗——深复制与浅复制	84
6.5 小结与思考	90

第3篇 结构型模式

第7章 桥接模式——课程太多别看花眼	94
7.1 新学期选课	94
7.2 为什么不能选其他专业的课程	98
7.3 数学课还不一样?	99
7.4 最佳选课方式	99
7.5 小结与思考	103
第8章 适配器模式——走到哪里都不怕	105
8.1 出国前的准备	105
8.2 笔记本到哪里都能充电	106
8.3 对象的“电源适配器”	107
8.4 充电不是代替工作	112
8.5 小结与思考	112
第9章 装饰模式——七十二变	114
9.1 超级化妆师	114
9.2 古装现代都能演	119
9.3 还需要招新演员?	121
9.4 穿越也不怕	122
9.5 小结与思考	122
第10章 组合模式——好大一棵树	124
10.1 复习数据结构	124
10.2 大学是一棵树	127
10.3 帮校长管理学校	128
10.4 任务顺利完成	132
10.5 小结与思考	132
第11章 享元模式——分享是一种美德	134
11.1 “对象”也有胖瘦	134
11.2 开发连连看游戏	135
11.3 游戏占内存太大怎么办	137
11.4 有付出就有回报	140
11.5 小结与思考	142
第12章 外观模式——机构融合的艰辛	144
12.1 收购加工厂	144
12.2 管理新厂的麻烦	145
12.3 设置“办事处”	148
12.4 好方法能推广到别处吗	150

12.5 小结与思考	150
第 13 章 代理模式——类也需要红娘	151
13.1 “翻墙”时代	151
13.2 找个代理来上网	153
13.3 并非所有网站都要“翻墙”	155
13.4 J2SE 中动态代理简介	156
13.5 小结与思考	157

第 4 篇 行为模式

第 14 章 模板方法模式——有模板，能量产	160
14.1 做最好的手机	160
14.2 限量版是个好卖点	164
14.3 给手机装上不同的操作系统	169
14.4 小结与思考	170
第 15 章 策略模式——自有妙计	171
15.1 变幻莫测的市场	171
15.2 蛋糕生产策略的制定	174
15.3 策略模式的可扩展性	175
15.4 小结与思考	176
第 16 章 状态模式——火车跑得快，全凭车头带	178
16.1 时快时慢的火车	178
16.2 提速与让车	179
16.3 高铁时代	181
16.4 小结与思考	185
第 17 章 观察者模式——你做坏事我知道	186
17.1 知己知彼，百战不殆	186
17.2 又见 007	186
17.3 一举一动都知道	191
17.4 间谍也不是万能的	194
17.5 小结与思考	197
第 18 章 备忘录模式——数据也穿越	198
18.1 突如其来的停电	198
18.2 游戏进度不会丢	199
18.3 为了进度升级计算机	202
18.4 停电不是每天都有	209

18.5 小结与思考	210
第 19 章 中介者模式——“万能”的客服 MM	211
19.1 一次网购体验	211
19.2 客服 MM “无所不能”	212
19.3 网站扩大忙坏了客服 MM	218
19.4 合理对待投诉	219
19.5 小结与思考	219
第 20 章 命令模式——军令如山倒	221
20.1 赤壁之战	221
20.2 曹军的传令官	222
20.3 如果没有铁索连舟	226
20.4 “后悔药”的代码实现	227
20.5 小结与思考	229
第 21 章 访问者模式——检查报表更方便	230
21.1 领导来检查	230
21.2 乱七八糟的报表	233
21.3 访问者模式的尴尬境地	237
21.4 何时使用访问者模式	238
21.5 小结与思考	241
第 22 章 职责链模式——切莫先斩后奏	242
22.1 钦差大人查访	242
22.2 先斩后奏惹麻烦	242
22.3 事情还是让皇上裁断吧	245
22.4 灵活使用职责链	247
22.5 小结与思考	247
第 23 章 迭代器模式——重复也不会乏味	249
23.1 招聘会的收获	249
23.2 快速筛选简历	250
23.3 简历筛选迭代器	250
23.4 小结与思考	254
第 24 章 解释器模式——出国需要翻译	255
24.1 短信“轰炸”	255
24.2 骚扰短信挑选机	256
24.3 骗子远离我	258
24.4 小结与思考	261

第5篇 模式总结

第 25 章 模式总结——最后大阅兵	264
25.1 各种模式之间的关系详解	264
25.2 灵活运用各种模式	265
25.3 设计模式思考图	265
25.4 关于架构师	266
25.5 小结与思考	267
附录 A UML 基础	276
A.1 UML 的组成	276
A.2 UML 的类图	277
A.3 类之间的关系	277
A.4 小结	279
附录 B 面向对象基础	280
B.1 面向对象开发方法的优点	280
B.2 什么是面向对象程序设计	280
B.3 面向对象的重要概念	281
B.4 类与实例的关系	282
B.5 面向对象程序设计的软件生命周期	282

第 1 篇 六大设计原则

▶▶ 第 1 章 设计模式的定义及设计原则

第 1 章 设计模式的定义及设计原则

设计模式一直是程序员的热门话题，熟练掌握并且合理运用设计模式也是衡量程序员的技能标准之一。本章先介绍设计模式的历史和定义，然后讲解设计模式中必须掌握的六大设计原则：单一职责原则、里氏代换原则、开放-封闭原则、依赖倒转原则、合成/聚合复用原则和迪米特法则。

1.1 初谈设计模式

我们首先了解设计模式的起源，然后说明为什么要学习设计模式，以及如何学习设计模式。

设计模式（Design Pattern）是一套面向对象的代码设计经验总结，是在编程领域被反复使用、被多数人知晓、而且经过分类整理的代码设计方法。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、保证代码可靠性。很显然，合理地使用设计模式对于开发人员和需要开发的系统都是有利的，设计模式可以让代码编写实现工程化，可以说，设计模式等同于软件工程项目的基石，如同大厦的一块块砖石一样。

设计模式理论的奠基人 GoF（Gang of Four，又称“四人帮”，即 Erich Gamma、Richard Helm、Ralph Johnson、John Vlissides 4 人）编写的 *Design Patterns: Elements of Reusable Object-Oriented Software*（1995 年出版，出版社：Addison Wesley Longman, Inc），第一次将设计模式提升到理论高度，实现了设计模式的规范化。该书通过总结和归纳，提出了 23 种基本设计模式，也就是我们现在常用的设计模式的框架。时至今日，在可复用面向对象软件的发展过程中，新的设计模式仍然在涌现。

设计模式是一种思想，也是一种技术，其思想上的成分要占绝大多数。一种设计模式就是一种解决方案，通过使用这种解决方案，开发者可以快速地完成某项工作；一种设计模式也是一个指路者，指明了开发的方向，引导程序员到达正确的目标。

在工作中，为了完成一项工作任务，要掌握完成该任务需要的理论和实践能力。设计模式是一种理论与实践相融合的知识，好比修理电路的人需要掌握与电路有关的理论知识，还要会使用专业的电气工具。软件开发人员在早期的学习生涯中通常都会被教授一些开发语言的规则以及用法，掌握了以后通常可以写出一些能在计算机上运行的程序。但是并没有更多地了解设计模式的理论，在生产可复用代码方面也实践甚少。

现实中，真正能做到面向对象所提出的可复用要求的代码少之又少。在这种情况下，设计模式应运而生。通过 23 个经典的设计模式以及对应的实例，真正的可复用的面向对象的编程不再只是纸上谈兵。

说明：每一个设计模式虽然有既定的结构图和基本程序代码，但是希望读者不要照本宣科，而应自己尝试着进行变通以达到真正理解设计模式精髓的目的。设计模式有23种但不仅仅限于23种，只要认真研究其思想，每个人都可以创造出属于自己的设计模式。

1.2 单一职责原则

编程者进入面向对象的世界之后，接触最多的名词就是“类”。程序员们开始设计各式各样的类来突出自己的程序符合“面向对象设计”原则，是不是只要用到了类就是面向对象了，究竟应该怎样去设计一个类来满足功能要求呢？别着急，本章会给出部分答案。

1.2.1 “超人”也是人

假设世界就是一个程序，聪明的程序员们为了维护世界和平，设计出了一个“超人”类，来维护世界的和平。既然是“超人”，那就应该是非常强大的，于是程序员们给了超人各种各样的能力。十项全能那是浮云，这个超人简直就是N项全能。去看看“超人”每天的工作吧。

“超人，快来救我。”原来是一个小女孩找不到家了，听到召唤的超人立刻出现。动用自己的超能力以后，很快超人把小女孩送到了父母那里。

“超人，有人抢银行。”几分钟以后，监狱里多了几个被超人送进去的坏人。抹了一把头上的汗，超人觉得非常有成就感。咦，好像又有人求助，超人还没来得及喘口气，就马不停蹄地到达了求助者身边。

“超人，我年纪大了，洗完衣服拧不动，你帮我拧一下吧。”邻家大婶举着刚洗好还滴水的衣服对着满头黑线的超人说。既然是民众有需求，那就要满足，于是超人发挥超能力，迅速地把衣服甩干了。当然，也因此得到了邻家大婶的盛赞。告别了邻家大婶，超人又踏上了帮助弱小群众的旅程。

“超人，快来帮我除一下后院的草。”

“超人，帮我做家庭作业吧。”

“超人，帮我写代码吧。”

“超人，我饿了。”

.....

晚上，超人带着一身的疲惫回到基地。

又是一个晴朗的早晨，但是超人却怎么也高兴不起来。事情是这样的，超人在帮助上学的小朋友过马路时，被小朋友们缠住，结果没来得及抽身去挽救本可以阻止的一场交通事故。当然，由此引来了民众们对超人的指责。超人很是郁闷地去向程序员们求助。

程序员们为了帮助超人解决这个问题可谓煞费苦心，首先将超人设计成了可以分身（多线程）的超人二代，然后又为超人增加了很多奇妙的功能，比如写代码啦，找bug啦，装操作系统啊什么的。

```
//超人类
public class SuperMan
{
    public void 维护世界和平()
    {
        //超人在维护世界和平
        .....
    }

    public void 除草()
    {
        //超人在帮邻家大婶除草
        .....
    }

    public void 写代码()
    {
        //超人化身为程序员
        .....
    }

    public void 写作业()
    {
        //超人在帮小朋友写作业，这样做是不对的
        .....
    }

    public void .....
    .....
    .....
}
```

忙活完上面这些，程序员们自豪地看着眼前的超人二代，功能齐全，样样都能干——真是完美的作品啊。

超人似乎也很高兴自己的变化，他终于拥有了众多的能力，打开窗户嗖的一下就飞出去继续维护世界和平去了，如图 1.1 所示。



图 1.1 超人飞走执行任务

然而事情也许并没有超人和程序员们自己想得那么容易，没过多久新的问题又出现了。程序员们发现超人洗衣服的功能设计得不是很好，于是不得不将超人叫回来重新修改代码。但是修改代码是需要时间的，在超人被“召回”这段期间，很多罪恶的事情发生了但是超人却抽不开身去管。

程序员们不由得叹了一口气：“看来不应该把超人设计得这么复杂啊！”

1.2.2 简单也很难做到

程序员们开始重新考虑设计超人的问题，既然是超人，那么其肩负的任务就应该是保护世界，而不是连保姆的活都一起干了。也就是说超人担负了过多的职责。超人把精力都用在了杂七杂八的事情上，真正只有超人才能干的活——比如处理坏人、维护和平等，超人反而没有时间处理了，如图 1.2 所示，超人被设计了清洗马桶功能。

那么怎么才能让超人简单一点呢？



图 1.2 超人清洗马桶

是让超人只负责保护世界，还是有选择地给超人赋予少量的职责呢？这又难倒了程序员，一方面想让超人变得更加健壮，一方面又想让超人能做更多的事情（这样才配得上超人的名号）。

1.2.3 帮“超人”减负

还好有一个面向对象的原则可以帮到程序员们，那就是单一职责原则。

单一职责原则的定义比较简单——对于一个类，应该只有一个引起它变化的原因。