

PEARSON

Broadview®
www.broadview.com.cn

Design
Patterns
In Java™ (2nd Edition)

Java 设计模式

(第2版)



John Metsker 著
William C. Wake
张逸 史磊 译



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Java 设计模式

(第2版)

Design Patterns In JavaTM
(2nd Edition)

Steven John Metsker 著
William C. Wake
张逸 史磊 译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书通过一个完整的 Java 项目对经典著作 *Design Patterns* 一书介绍的 23 种设计模式进行了深入分析与讲解, 实践性强, 却又不失对模式本质的探讨。本书创造性地将这些模式分为 5 大类别, 以充分展现各个模式的重要特征, 并结合 UML 类图与对应的 Java 程序, 便于读者更好地理解。全书给出了大量的练习, 作为对读者的挑战, 以启发思考, 督促读者通过实践练习的方式来掌握设计模式。同时, 作者又给出了这些练习的参考答案, 使读者可以印证比较, 找出自己的不足, 提高设计技能。

本书适合各个层次的 Java 开发人员与设计人员阅读, 也可以作为学习 Java 与设计模式的参考读物或教材。

Authorized translation from the English language edition, entitled DESIGN PATTERNS IN JAVA, 2E, 9780321333025 by METSKER, STEVEN JOHN; WAKE, WILLIAM C., published by Pearson Education, Inc, publishing as Addison-Wesley Professional, Copyright© 2006 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and PUBLISHING HOUSE OF ELECTRONICS INDUSTRY Copyright©2012

本书简体中文版专有版权由 Pearson Education 培生教育出版亚洲有限公司授予电子工业出版社。未经出版者预先书面许可, 不得以任何方式复制或抄袭本书的任何部分。

本书简体中文版贴有 Pearson Education 培生教育出版集团激光防伪标签, 无标签者不得销售。

版权贸易合同登记号 图字: 01-2011-4224

图书在版编目 (CIP) 数据

Java 设计模式: 第 2 版 / (美) 梅特斯克 (Metsker, S.J.), (美) 维克 (Wake, W.J.) 著; 张逸, 史磊译.

北京: 电子工业出版社, 2012.9

书名原文: Design Patterns in Java: 2nd Edition

ISBN 978-7-121-17826-9

I. ①J… II. ①梅… ②维… ③张… ④史… III. ①JAVA 语言—程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字 (2012) 第 179321 号

策划编辑: 张春雨 符隆美

责任编辑: 刘 舫

印 刷: 北京丰源印刷厂

装 订: 三河市皇庄路通装订厂

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787×980 1/16

印张: 24.5 字数: 549 千字

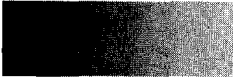
印 次: 2012 年 9 月第 1 次印刷

定 价: 75.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。



译者序

如今，介绍和讲解设计模式的书籍可谓汗牛充栋。无论是定义、解读、延伸还是扩展，都是基于面向对象的设计原则，用了放大镜对着 GoF (*Design Patterns* 一书) 提出的 23 种设计模式，如科学解剖一般，剖析每一道脉络，观察每一片纹理，细微至纤毫毕现，真可以说是道尽个中妙处；许多精妙阐述，又如黄钟大吕，发聩振聋，醍醐灌顶。

是否设计模式的精妙之处，业已为这些著作所穷尽？然，又未必尽然！以模式而论，若只局限在这 23 种模式的范围内，几乎每种模式的变化，都可以被悉心推演出来；每种模式的结构，也已被阐述得淋漓尽致。然而，若论及设计，则如大道苍穹，实则是不可穷尽的。基本上，设计的复杂程度已不亚于一个纷繁的世界，而软件，就是我们要构造的这个世界。

因此，再出现一本讲解设计模式的书，就不足为怪了。那么，它值得你去阅读吗？

讨论一本书是否值得阅读，应基于书本身的价值去判断，判断的标准则依据读者的目标而定。从读者而非译者的角度看待本书，个人认为，它确乎是有价值的。这些价值主要体现在三个方面。

GoF 对于 23 种设计模式的分类已经深入人心，即众所周知的创建型模式、结构型模式与行为型模式。这一分类浅显易懂，明白无误地表达了模式的意图与适用场景。但是，这一分类仍有不足之处。例如创建型模式除了关注对象的创建之外，还需处理好对象之间的结构；又如桥接模式对于抽象与实现的解耦，在一定程度上又体现了对行为的抽象；再比如行为型模式中的迭代器模式，其实还涵盖了创建迭代器的职责。本书对于设计模式的分类不落窠臼，根据作者

对于设计模式的思考，别出心裁地给出了自己的一种分类，即分为接口型模式、职责型模式、构造型模式、操作型模式与扩展型模式。如果仔细阅读和思考这些模式，你会发现这 5 个分类很好地抓住了相关模式的设计本质。譬如，扩展型模式关注的是代码功能的扩展，因而很自然地就可以把装饰器模式与访问者模式归入这一类。

彰显本书价值的第二个方面在于贯穿本书始终的习题练习，作者将其称之为“挑战”。确实如此，这些挑战仿佛是作者故意给读者设定的“陷阱”、“障碍”，是登堂入室所必须跨过的门槛。最关键的一点是，通过这些“挑战”，就从单方面的灌输知识，变成了一定程度的双向互动。作者就像课堂上的老师，提出问题引人思考；读者就是学生，面对老师“咄咄逼人”的提问，必须打起十二分的精神，分析问题，寻找问题的答案。最后，循循善诱的老师给出了自己的解决方案。学生可以与之对比，以便发现自己在设计上还存在的问题。因此，本书不适合那些惫懒的读者，不适合那些喜欢被动接收知识输入，不善于思考，不善于总结的程序员。

真正让本书获得赞誉的还是本书给出的案例，不过，也很有可能因此收获负面的批评。本书的案例是一个虚拟的真实项目，Oozinoz 公司纯属子虚乌有，完全是由作者杜撰出来的一家虚拟公司。但这个案例又如此真实，既牵涉复杂的领域逻辑，又面对客户提出的种种需求变化，与我们工作中需要开发的项目何其相似！可能面临的批评是，为了学习设计模式，可能读者还需要成为一名焰火专家。然而，我谨以最谦卑的态度恳求诸位，在满怀怨气、恶毒诅咒作者（也可能包括躺着中枪的译者）之前，先想想我们平时开发的软件，是否存在相似复杂度的领域需求呢？让我们再仔细想想，倘若作者给出一个纯粹编造出来的玩具项目，贴近生活，浅显易懂，学习起来势如破竹，一路通关，是否真的意味着你已经明白如何在真实项目中运用设计模式？窃以为，学习尤其是技术学习，并不都是舒舒服服寓教于乐，躺着、玩着以及笑着也能学好设计模式。你以为的懂，以为的悟，其实还是一种虚妄。你抓住的是水中央的月影，一旦遇到真实案例，就好似石头打破水面的宁静，一切都会破碎。

本书的原版事实上获得了业界的广泛赞誉，同时也是 John Vlissides 主编的“软件模式”丛书之一。John Vlissides 就是著名的 GoF 其中的一位，可惜他已在多年前离开人世。本书作者是 John Vlissides 的生前好友，本书内容曾经得到过他的建议。从本书的内容来看，部分 Java 案例显得有些过时，不过，就设计而言，拥有悠久的历史，有时候意味着它可能成为经典。不错，与经典的 GoF 相比，本书无疑要失色许多。GoF 的光芒在于它的开创性。只要是讲解设计模式，没有哪一本书的光芒可以盖过 GoF 的著作。它就像是一颗恒星，其他有关设计模式的书籍，是围绕着它公转的一颗颗行星，都是借着恒星的光芒反射出属于自己的光亮。然而，从光芒的热

度与亮度来讲，也许行星才是当前的你最适合的。

阅读本书的读者，除了需要具备一些面向对象与设计模式的基础知识外，还需要有足够的耐心，并保存一份渴望与热情。耐心可以帮助你坚持细读与精读，持之以恒地深入理解本书的案例分析，努力面对作者给出的挑战。而这种耐心则需要提高技术能力的渴望，探求技术奥秘的热情来时刻保鲜。

本书的翻译由我的同事史磊与我共同完成，并最后由我完成审校工作。在翻译本书时，我还参考了由龚波、赵彩琳、陈蓓翻译的前一个版本，在此向他（她）们表示衷心的感谢。因为工作繁忙的缘故，本书的翻译工作一直断断续续持续了近一年的时间，如今交稿，既有卸下重任的轻松畅快，却又因为自己的怠慢使得翻译工作进展缓慢而深感愧疚。这里需要感谢本书编辑符隆美女士给予我的耐心与支持。

在写作这篇译者序时，同事史磊已经远赴 ThoughtWorks 南非工作，而我则从北京回到了 ThoughtWorks 成都。非常怀念我们在北京一起工作的日子。我们曾经在同一个项目结对编程，本书的翻译也可以说是结对完成，算是一次愉快的翻译体验。鉴于本人能力水平有限，翻译或有疏漏或错误，还请读者不吝赐教，并通过我的博客 www.agiledon.com 与我联系。

张逸

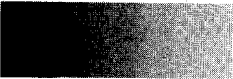
ThoughtWorks 高级咨询师

To Alison, Emma-Kate, and Sarah-Jane

—Steve

To May Lyn, Tyler, and Fiona

—Bill



序

设计模式是面向对象设计中常见问题的类级与方法级的解决方案。本书面向那些希望更上一层楼的初、中级 Java 程序员，以及那些不具备设计模式知识的高级 Java 程序员。

本书采用工具书的方式进行编写。每一章针对一个特定的模式。为了更好地阐释模式，每章还包含了大量的练习，要求读者回答问题，或者编写代码解决问题。

我们强烈地建议读者在阅读本书时，不要对自我挑战中的问题视而不见。通过完成这些挑战，你将获益匪浅，即使你一周只能阅读一至两章的内容。

修订版

本书是对 *Design Patterns Java WorkbookTM* 和 *Design Patterns in C#* 两本书的集成与修订。本书综合了前者面向 Java 语言的特点，又保留了后者独立陈述的写作方式。如果你业已阅读并理解这两本书，可以不必再阅读本书。

编码规范

本书代码可以在线获得。若欲了解获取代码的方法，请参阅本书附录 C。

本书采用了与 Sun 编码规范一致的通用风格。如有可能，会省略一些能够省略的括号，以保证本书排版上的一致性。为了适应版面，有的变量名比我们真正使用的要短。为了避免源代码管理的麻烦，我们直接在文件名后面添加数字后缀，以表示多个版本的文件（例如 `ShowBallistics2`）。

致谢

写书是一项挑战。在写书期间，许多审稿人为本书提出了非常有价值的建议：Daryl Richter、Adewale Oshineye、Steven M. Luplow、Tom Kubit、Rex Jaeschke、Jim Fox 和 David E. DeLano。他们每个人的建议都使得本书的质量得以改善。早期版本的读者以及审稿人也做出了同样的贡献。

还要感谢 Addison-Wesley 的编辑团队，尤其是 Chris Guzikowski、Jessica D'Amico 和 Tyrrell Albaugh。其他编辑包括 Mary O'Brien 和 John Wait 也为我们提供了诸多帮助。

感谢已经辞世的 John Vlissides 对本书以及其他书籍的鼓励与建议。John 是软件模式丛书的编辑，经典著作 *Design Patterns* 的合著者，我的朋友，灵感迭出的天才。

除了主要参考了 *Design Patterns* 一书外，我们还从其他诸多著作中获益，详见本书参考书目。其中，*Unified Modeling Language User Guide*（由 Booch、Rumbaugh 和 Jacobsen 在 1999 年编写）提供了对 UML 清晰的阐释，*JavaTM in a Nutshell*（由 Flanagan 在 2005 年编写）对 Java 语言简明扼要的介绍，让我受益颇丰。*The Chemistry of Fireworks*（由 Russell 在 2000 年编写）则是我获取焰火知识的主要来源。

最后，我们要感谢出版社所有员工的辛勤劳动与努力奉献，正是你们的工作使得本书得以付梓出版。

Steve Metsker（Steve.Metsker@acm.org）

Bill Wake（William.Wake@acm.org）



目 录

序	xv
第 1 章 绪论	1
为何需要模式	1
为何需要设计模式	2
为何选择 Java	3
UML	3
挑战	4
本书的组织	4
欢迎来到 Oozinoz 公司	6
小结	6
 第 1 部分 接口型模式	
第 2 章 接口型模式介绍	8
接口与抽象类	8
接口与职责	10
小结	11
超越普通接口	12

第3章 适配器（Adapter）模式	13
接口适配	13
类与对象适配器	17
JTable 对数据的适配	20
识别适配器	24
小结	25
第4章 外观（Facade）模式	27
外观类、工具类和示例类	27
重构到外观模式	29
小结	38
第5章 合成（Composite）模式	39
常规组合	39
合成模式中的递归行为	40
组合、树与环	42
含有环的合成模式	47
环的影响	50
小结	51
第6章 桥接（Bridge）模式	52
常规抽象：桥接模式的一种方法	52
从抽象到桥接模式	54
使用桥接模式的驱动器	57
数据库驱动	57
小结	59

第2部分 职责型模式

第7章 职责型模式介绍	62
常规的职责型模式	62
根据可见性控制职责	64

小结	65
超越普通职责	65
第 8 章 单例（Singleton）模式	67
单例模式机制	67
单例和线程	68
识别单例	70
小结	71
第 9 章 观察者（Observer）模式	72
经典范例：GUI 中的观察者模式	72
模型/视图/控制器	76
维护 Observable 对象	82
小结	84
第 10 章 调停者（Mediator）模式	85
经典范例：GUI 调停者（Mediator）	85
关系一致性中的调停者模式	89
小结	96
第 11 章 代理（Proxy）模式	97
经典范例：图像代理	97
重新思考图片代理	102
远程代理	104
动态代理	109
小结	114
第 12 章 职责链（Chain of Responsibility）模式	115
现实中的职责链模式	115
重构为职责链模式	117
固定职责链	119
没有组合结构的职责链模式	121
小结	121

第 13 章 享元 (Flyweight) 模式	122
不变性	122
抽取享元中不可变的部分	123
共享享元	125
小结	128

第 3 部分 构造型模式

第 14 章 构造型模式介绍	130
构造函数的挑战	130
小结	132
超出常规的构造函数	132
第 15 章 构建者 (Builder) 模式	134
常规的构建者	134
在约束条件下构建对象	137
可容错的构建者	139
小结	140
第 16 章 工厂方法 (Factory Method) 模式	141
经典范例: 迭代器	141
识别工厂方法	142
控制要实例化的类	143
并行层次结构中的工厂方法模式	145
小结	147
第 17 章 抽象工厂 (Abstract Factory) 模式	148
经典范例: 图形用户界面工具箱	148
抽象工厂和工厂方法	153
包和抽象工厂	157
小结	157

第 18 章	原型（Prototype）模式	158
	作为工厂的原型	158
	利用克隆进行原型化	159
	小结	162
第 19 章	备忘录（Memento）模式	163
	经典范例：使用备忘录模式执行撤销操作	163
	备忘录的持久性	170
	跨会话的持久性备忘录	170
	小结	174

第 4 部分 操作型模式

第 20 章	操作型模式介绍	176
	操作和方法	176
	签名	177
	异常	178
	算法和多态	179
	小结	180
	超越常规的操作	181
第 21 章	模板方法（Template Method）模式	182
	经典范例：排序	182
	完成一个算法	186
	模板方法钩子	188
	重构为模板方法模式	189
	小结	191
第 22 章	状态（State）模式	193
	对状态进行建模	193
	重构为状态模式	197

	使状态成为常量	201
	小结	203
第 23 章	策略（Strategy）模式	204
	策略建模	204
	重构到策略模式	207
	比较策略模式与状态模式	211
	比较策略模式和模板方法模式	211
	小结	212
第 24 章	命令（Command）模式	213
	经典范例：菜单命令	213
	使用命令模式来提供服务	216
	命令钩子	217
	命令模式与其他模式的关系	219
	小结	220
第 25 章	解释器（Interpreter）模式	221
	一个解释器示例	221
	解释器、语言和解析器	233
	小结	234

第 5 部分 扩展型模式

第 26 章	扩展型模式介绍	236
	面向对象设计的原则	236
	Liskov 替换原则	237
	迪米特法则	238
	消除代码的坏味道	239
	超越常规的扩展	240
	小结	241

第 27 章 装饰器 (Decorator) 模式	242
经典范例：流和输出器	242
函数包装器	250
装饰器模式和其他设计模式的关系	257
小结	258
第 28 章 迭代器 (Iterator) 模式	259
普通的迭代	259
线程安全的迭代	261
基于合成结构的迭代	267
小结	277
第 29 章 访问者 (Visitor) 模式	278
访问者模式机制	278
常规的访问者模式	280
Visitor 环	286
访问者模式的危机	290
小结	292
附录 A 指南	293
附录 B 答案	297
附录 C Oozinoz 源代码	366
附录 D UML 概览	369
参考文献	375

第 1 章

绪论

本书涵盖了 Erich Gamma、Richard Helm、Ralph Johnson 与 John Vlissides 在 1995 年编写的经典著作 *Design Patterns* 相同的技术，并提供了 Java 范例。书中还包括了许多“挑战”（练习）——这些练习能够帮助我们提高在软件开发中运用设计模式的能力。

本书面向那些了解 Java 语言的开发人员，以及希望提高设计能力的设计人员。

为何需要模式

模式是做事的方法，是实现目标，研磨技术的方法。这种对高效技术不懈追求的思想，广泛见于诸多领域，例如制作精美的佳肴、生产绚烂的烟花、开发出色的软件及其他技艺。对于任何一种迈向成熟的全新技术，身处这个行当的人们都需要寻找通用而有效的方法，以达到他们的目标，并解决不同场景的问题。实践技艺的团体通常会发明一些行话，用来讨论他们的技艺。一些被公认的行话反映了某种模式，或者建立了实现准确目标的技术。随着技艺的增长，以及这些行话的发展，一些行话的创作者开始扮演重要的角色。他们对技艺模式进行编档，为这些行话设定标准，然后发布这些有效的技艺。

Christopher Alexander 是最早对技艺的最佳实践进行模式编档的一位作者。他的著作与建筑有关，而非软件。在 *A Pattern Language: Towns, Buildings Construction*（由 Alexander、Ishikouwa