原 版 风 暴 系 列

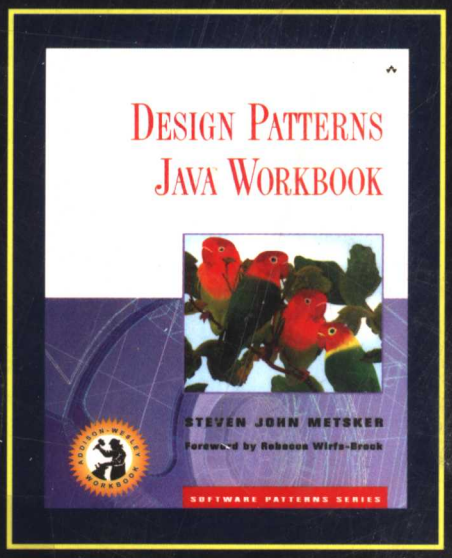
PEARSON
Addison Wesley

# Design Patterns Java Workbook

# 设计模式Java手册

（影印版）

[美] Steven John Metsker 著
Rebecca Wirfs-Brock 序




- 全面阐释《Design Patterns》中23种设计模式
- 透彻理解设计模式在Java应用开发中的作用
- 寓教于实践，充分增强应用模式的能力

中国电力出版社
www.infopower.com.cn

原版风暴系列

# Design Patterns Java Workbook

# 设计模式Java手册

（影印版）

[美] Steven John Metsker 著
Rebecca Wirfs-Brock 序



中国电力出版社

Design Patterns Java Workbook (ISBN 0-201-74397-3)
Steven John Metsker


图书在版编目（CIP）数据
设计模式 Java 手册 /（美）米斯科著. —影印本. —北京：中国电力出版社，2003
（原版风暴系列）
ISBN 7-5083-1409-3
Ⅰ.设... Ⅱ.米... Ⅲ.JAVA 语言－程序设计－英文 Ⅳ.TP312
中国版本图书馆 CIP 数据核字（2003）第 102752 号

丛 书 名：原版风暴系列
书　　名：设计模式Java手册（影印版）
编　　著：（美）Steven John Metsker
责任编辑：姚贵胜
出版发行：中国电力出版社
　　地址：北京市三里河路6号　　邮政编码：100044
　　电话：（010）88515918　　传真：（010）88518169
印　　刷：北京丰源印刷厂
开　　本：787×1092 1/16　　印　　张：30
书　　号：ISBN 7-5083-1409-3
版　　次：2004年1月北京第1版　2004年1月第1次印刷
定　　价：49.80 元

## Praise for *Design Patterns Java™ Workbook*

"An excellent book...I'm incredibly impressed with how readable it is. I understood every single chapter, and I think any reader with any Java familiarity would. This book is going to be required reading in a lot of places, including my office."

—Joshua Engel

"Provides a new, more Java-literate way to understand the 23 GoF patterns."

—Bob Hanmer

"This book translates *Design Patterns* into what Java programmers need to know. It is full of short, engaging programming and design problems with solutions—making it easy for programmers to work through solutions and really make patterns 'stick.'"

—Rebecca Wirfs-Brock

"This is one exciting book. It's approachable, readable, interesting, instructive, and just plain valuable. It'll eclipse all other books purporting to teach people the GoF patterns in Java—and perhaps any other language."

—John Vlissides

To Alison

Who fills our house with glimmering light

With her loving, cozy fire

And Emma-Kate and Sarah-Jane

Our precious elves, beloved sprites

Who hop as light as bird from brier.

*Through the house give glimmering light*
*By the dead and drowsy fire;*
*Every elf and fairy sprite*
*Hop as light as bird from brier;*

—William Shakespeare
*A Midsummer-Night's Dream*

# FOREWORD

*Tell me and I forget. Teach me and I remember. Involve me and I learn.*
—Benjamin Franklin

WITH *Design Patterns Java™ Workbook*, Steve Metsker has done something truly amazing: He's packed a book with extensive coding examples and dozens of exercises that challenge you to truly grok design patterns. It uses software for a fictional company that manufactures and sells fireworks and puts on firework displays as an example. Not only are the coding examples more entertaining than the tired old ATM machine examples, but you'll find yourself learning obscure firework facts as you learn design patterns. The book is fun as well as inviting! And because it describes how each design pattern fits in with and extends Java language constructs, you may find yourself learning more about Java, too!

A pattern is a way of doing something, a way of pursuing an intent. A design pattern is a way of pursuing an intent using object technology: classes and their methods, inheritance, and interfaces. Each pattern has a name. If you and your teammates know about design patterns, you can work more effectively—because you share a common vocabulary, it's like speaking in shorthand! You can discuss your intentions without groping for the right words. And developers who routinely apply design patterns to their code end up with code that is more flexible and easier to read and modify.

Design patterns were originally described in the book *Design Patterns*, written by Erich Gamma and his colleagues (Addison-Wesley, 1995). That book presents a catalog of 23 proven design patterns for structuring, creating, and manipulating objects. In *Design Patterns Java™ Workbook*, Steve clearly explains each original design pattern from a Java programmer's perspective.

If you take up the challenges in this book, you'll have plenty of opportunity to learn patterns by writing and extending existing code, answering questions that force you to think carefully, and solving some interesting design problems. No matter how much you read about something, the best way to really learn is to put it to practice.

Rebecca Wirfs-Brock
Sherwood, Oregon
January 2002

# PREFACE

At OOPSLA[1] 2000 in Minneapolis, Minnesota, I asked Mike Hendrickson of Addison-Wesley what types of books he thought readers wanted. I was interested to hear that he felt that there is still a market for books to help readers understand design patterns. I suggested the idea of a Java workbook that would give readers a chance to expand and to exercise their understanding of patterns. This sounded good to Mike, and he introduced me to Paul Becker, who supports Addison-Wesley's Software Patterns Series. Paul's immediate response was that such a book "should have been written five years ago." I would like to thank Mike and Paul for their initial encouragement, which inspired me to take on this task.

Since that initial meeting, Paul has supported me throughout the entire development process, guiding this book toward publication. Early on, Paul asked John Vlissides, the Software Patterns Series editor, for his views on the project. John's reply was that Paul should support the project "in all wise," inspirational words that have stayed with me.

John Vlissides is also, of course, one of the four authors of *Design Patterns*. John and his coauthors—Erich Gamma, Ralph Johnson, and Richard Helm—produced the work that is in every way the foundation of this book. I referred to *Design Patterns* nearly every day that I worked on this book and can hardly overstate my reliance on it.

I also relied on many other existing books, which are listed in the bibliography. In particular, I depended on *The Unified Modeling Language User*

1. OOPSLA is a conference on object-oriented programming, systems, and applications, sponsored by the Association for Computing Machinery.

*Guide* (Booch, Rumbaugh, and Jacobson 1999) for its clear explanations of UML. For accuracy in Java-related topics I consulted *Java™ in a Nutshell* (Flanagan 1999b) almost daily. I also repeatedly drew on the insights in *Patterns in Java™* (Grand 1998) and *Java™ Design Patterns* (Cooper 2000).

During the months I was working on this book, I also worked at a financial services institution that has facilities in many locations. As the book emerged, I developed an instructor's course to go with it. I taught the course in Richmond, Virginia, and my associates Tim Snyder and Bill Trudell taught the course concurrently at other locations. I would like to thank these instructors and the students from all three courses for their inspiration and insights. In particular, I would like to thank Srinivasarao Katepalli, Brad Hughes, Thiaga Manian, Randy Fields, Macon Pegram, Joe Paulchell, Ron DiFrango, Ritch Linklater, Patti Richards, and Ben Lewis for their help and suggestions. I would also like to thank my friends Bill Wake and Gagan Kanjlia for their reviews of this book in its early stages and Kiran Raghunathan for his help in the later stages. I am grateful to the sharp-eyed and conscientious readers who pointed out errors in the first printing, especially Simon Bennett, Thierry Matusiak, Shun Nin Lau, Alec Noronha, Wagner Truppel, and Roy Wagner. Finally, I'd like to thank my friend Jeff Damukaitis for his suggestions, particularly his insistence that I make the book's code available to readers. (It is, at `oozinoz.com`).

As the book came along, Paul Becker arranged for many excellent reviewers to help guide its progress. I'd like to thank John Vlissides again for his reviews. In every review, John somehow convinced me that he liked the book while simultaneously pointing out scores of significant improvements. I'd like to thank Luke Hohmann, Bob Hanmer, Robert Martin, and Joshua Kerievsky for their help at various stages. Each of them made this book better. I'd like to thank Joshua Engel, who has an amazing ability to blend sharp insight with a gentle touch. I'd like to thank Rebecca Wirfs-Brock, who had many great suggestions, including completely reorganizing the book. I had initially not taken care to put important but understandable patterns up front. The book is much stronger now because of Rebecca's advice and the help of all the book's reviewers. Finally, I would like to thank Tyrrell Albaugh and the production staff at Addison-Wesley for transforming a collection of words into an attractive and usable book.

Steve Metsker (Steve.Metsker@acm.org)

# CONTENTS

# 1 INTRODUCTION TO PATTERNS

THIS BOOK is for developers who know Java and who have had some exposure to the book *Design Patterns* (Gamma et al. 1995). The premise of this book is that you want to

- Deepen your understanding of the patterns that *Design Patterns* describes
- Build confidence in your ability to recognize these patterns
- Strengthen your ability to apply these patterns in your own Java programs

## Why Patterns?

A **pattern** is a way of doing something, or a way of pursuing an intent. This idea applies to cooking, making fireworks, developing software, and to any other craft. In any craft that is mature or that is starting to mature, you can find common, effective methods for achieving aims and solving problems in various contexts. The community of people who practice a craft usually invent jargon that helps them talk about their craft. This jargon often refers to patterns, or standardized ways of achieving certain aims. Writers document these patterns, helping to standardize the jargon. Writers also ensure that the accumulated wisdom of a craft is available to future generations of practitioners.

Christopher Alexander was one of the first writers to encapsulate a craft's best practices by documenting its patterns. His work relates to architecture—of buildings, not software. *A Pattern Language: Towns, Buildings,*

*Construction* (Alexander, Ishikawa, and Silverstein 1977) provides patterns for architecting successful buildings and towns. Alexander's writing is powerful and has influenced the software community, partially because of the way he looks at intent.

You might state the intent of architectural patterns as "to design buildings." But Alexander makes it clear that the intent of architectural patterns is to serve and to inspire the people who will occupy buildings and towns. Alexander's work showed that patterns are an excellent way to capture and to convey the wisdom of a craft. He also established that properly perceiving and documenting the intent of a craft is a critical, philosophical, and elusive challenge.

The software community has resonated with Alexander's approach and has created many books that document patterns of software development. These books record best practices for software process, software analysis, and high-level and class-level design. Table 1.1 lists books that record best practices in various aspects of software development. This list of books is not comprehensive, and new books appear every year. If you are choosing a book about patterns to read you should spend some time reading reviews of available books and try to select the book that will help you the most.

## Why Design Patterns?

A **design pattern** is a pattern—a way to pursue an intent—that uses classes and their methods in an object-oriented language. Developers often start thinking about design after learning a programming language and writing code for a while. You might notice that someone else's code seems simpler and works better than yours does, and you might wonder how that person achieves this simplicity. Design patterns are a level up from code and typically show how to achieve a goal, using one to ten classes. Other people have figured out how to program effectively in object-oriented languages. If you want to become a powerful Java programmer, you should study design patterns, especially those in *Design Patterns.*