



SUN核心技术丛书

Software Architecture Design Patterns in Java

Java

软件体系结构设计模式 标准指南

[美] Partha Kuchana 著

王卫军 楚宁志 等译

吕 军 审校



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

Software Architecture Design Patterns in Java

Java 软件体系结构 设计模式标准指南

[美] Partha Kuchana 著

王卫军 楚宁志 等译

吕 军 审校



電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书全面介绍了常用的 42 个设计模式,其中包括 23 个经典的 GoF 模式。新增并常用的 19 个新模式给读者带来更多的参考价值。本书的程序实例均采用 Java 语言,并且在 Internet 上可以获得源代码。本书通篇采用了 UML 标准图表作为描述工具,使得程序代码、设计模式以及文字说明能很好地融为一体。本书的最后一章为“案例研究”,作者把多个模式放在一个实际的应用场合里一起配合工作,这样的安排使读者能更好地对各种模式综合运用。本书内容深入浅出、清楚易懂,大量的 Java 程序实例和 UML 图表使内容更加清晰且更具可操作性。

本书适合软件开发和设计人员使用。



All Rights Reserved.

Authorized translation from English language edition published by Auerbach, part of Taylor & Francis Group LLC.

本书英文版由 Auerbach 出版,该公司已将中文版独家版权授予电子工业出版社及北京美迪亚电子信息有限公司。未经许可,不得以任何形式复制或抄袭本书内容。

版权贸易合同登记号 图字:01-2005-4178

图书在版编目 (CIP) 数据

Java 软件体系结构设计模式标准指南/ (美) 库察那 (Kuchana, P.) 著; 王卫军等译. —北京: 电子工业出版社, 2006.2

(SUN 核心技术丛书)

书名原文: Software Architecture Design Patterns in Java

ISBN 7-121-01873-X

I. J… II. ①库… ②王… III. Java 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2006) 第 006565 号

责任编辑: 朱巍

印 刷: 北京天竺颖华印刷厂

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

北京市海淀区翠微东里甲 2 号 邮编: 100036

经 销: 各地新华书店

开 本: 787×1092 1/16 印张: 23.25 字数: 580 千字

印 次: 2006 年 2 月第 1 次印刷

定 价: 37.00 元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。联系电话: (010) 68279077。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。

前 言

Partha Kuchana 是一位富有经验的企业级系统体系结构设计师。他体会到模式不仅仅是好的想法,更是对来自实践的知识的俘获。这个来之不易的认识正是作者想和本书的读者一起分享的。下面我想谈谈我对他将要在本书中介绍给读者的内容的喜欢之处。

本书给出了包括 23 个 GoF 模式在内的 42 个设计模式。这些模式分类如下:

- 7 个基本模式;
- 5 个创建模式;
- 4 个集合模式;
- 11 个结构模式;
- 11 个行为模式;
- 4 个并发模式。

针对每个模式的讨论都包括一个用 Java 语言实现的示例程序,而且这些示例程序的源代码在下面这个网址处可以找到: http://www.crcpress.com/e_products/downloads/download.asp。这样的安排还是很不错的。

Partha 选取通常难以理解的素材,却清楚地解释了其中的思想使之易于理解。这样的考虑很重要,它兼顾了初次接触这些材料的新手和那些只想从中迅速获取“马上就能拿来用的”要点的有经验开发人员。针对每个模式的讨论还包括习题部分。这部分主要是供个人提高之用,也适用于那些用本书做教材而时间又比较紧的教员。

Partha 在比较模式之间的异同方面着实花了些时间。例如,在讨论调停者模式时,他给出了一个表格以显示这个模式和外观模式的相似之处和不同之处。读者会发现这种分析比孤立地讲解每个模式更容易理解。文中还有介绍不同模式间关系的内容。

例如,在讨论调停者模式时,就有一段与前面章节中介绍过的命令模式有关的设计用例。

最后,在本书的结尾处,读者会欣喜地发现一个实例研究部分。这部分把一些模式放在一起以说明如何解决一个复杂的问题,以及如何把不同的模式放在一起工作。如果学习过 Christopher Alexander 的成果,读者就会认识到模式并非孤立地被使用,而是在一个特定的问题域中相互协同来解决各种问题的。

自 GoF 的书出版至今已有 10 年之久了。这个期间有很多的模式被确认和研究,也出现了很多这方面的书。本书与 GoF 的书类似,也像是一本目录手册,读者不大可能一次就从头读到尾,但它将会是读者需要经常查阅的一本书。当读者有了很多像“利用 Java 语言这件事该如何做”这样的问题时,一定希望身边有一个专家来回答问题吧。本书将会起到和专家同样的作用。

Linda Rising
Phoenix, AZ

献 辞

谨以此书献给我的家人

作者简介

Partha Kuchana 是一位富有经验的企业级系统体系结构设计师。他有 11 年的从业经验,涉及项目交付管理(本土或离岸方式)和企业级系统体系结构搭建、系统设计、开发、顾问指导以及培训的各个方面。他还是一位 Sun 认证的企业级体系结构设计师。

在过去的几年里,他在英国和美国的许多客户那里从事过大量客户-服务器、电子商务、Web 门户以及企业应用集成(EAI)方面的项目。这些项目涉及了迭代设计方法学,比如 Rational 统一过程(RUP)和极限编程。

在将设计模式应用于应用程序的体系结构搭建和设计方面他有着广泛的经验。他利用来自不同厂商的 Web 服务、中间件和消息通信产品成功地实现了 B2B 系统和复杂异构系统集成的体系结构搭建和系统的设计。另外,他还出版了几本软件方面的书。

主页: <http://members.ITJobsList.com/partha>

电子邮件: ParthaKuchana@ITJobsList.com

致 谢

首先感谢我的妻子,感谢她的耐心与支持;感谢她在 UML 和 Java 编程方面承担了我的一部分工作;感谢她在我沮丧时给予的鼓舞。还要感谢我的父母、姐姐、兄弟以及我亲爱的朋友们,你们在我生活中始终给予的支持与鼓励让我掌握了获得成功所需要的各种技能。

感谢 Venu Kuchana 和 D. R. Sudhakar,因为他们贡献了不同的 Java 程序。还要感谢 BalaLingam Kuchana,他在 UML 方面给了我帮助,而且负责书稿的排版工作。

感谢在 Auerbach 出版社的所有同仁。他们对这个项目做出了贡献,并且使之成为了一段不同寻常的经历。特别地,我对参考资料编辑 John Wyzalek 怀有深深的感激,因为他不仅和我分享了忘我的工作,而且提供了很好的建议与帮助。还要感谢主编 Claire Miller 女士给予的宝贵建议,她的安排组织使得本书有着不错的篇章结构。

真诚地感谢 Linda Rising 为本书写了前言。

真诚地感激和感谢下面这些书评撰写人,他们花费宝贵的时间阅读了书稿并提供了反馈信息。

- Pradyumn Sharma, CEO, Pragati Software Pvt. Ltd.
- Carsten Kuckuk, 项目带头人, Design Patterns Study Group Stuttgart, RIB Software AG
- Tim Kemper, Boulder Design Patterns Group
- Geoffrey Sparks, CEO, Sparx Systems P/L
- Edward L. Howe, 软件体系结构设计师, Employease, Inc.
- Christopher R. Gardner, 软件开发工程师, McKesson Information Solutions
- David Deriso, 资深软件工程师, Employease, Inc.
- Mike Heinrich, 软件工程师, Canada
- Rodney Waldoff, 系统体系结构主管, Encyclopedia Britannica Inc.
- Thomas SMETS, 软件工程师, Belgium
- Linda Rising 博士, 独立软件顾问, Arizona State University
- Ray Tayek, 项目协调人, LAJUG/OCJUG

特别感谢 Pradyumn Sharma、Carsten Kuckuk 和 Tim Kemper,感谢他们提供的很有深度的建议、他们的细致入微以及他们的宝贵建议,其中包括对设计模式有强烈好奇心的读者可能会提的那些疑难问题。

另外还要感谢 Mark Grand 在有关模式书籍书写的很多方面给予的鼓励和建议。

相信在这里我还是漏掉了很重要的一些人,如果是那样,请见谅为盼!

译者序

本书全面介绍了软件开发和设计人员在工作和学习中常用的 42 个设计模式。

也许读者会有这样的疑问,近几年有关设计模式的书面世的已经很多了,为什么还要出版关于同样题材的书呢?译者认为本书有如下特点值得一提。

第一,本书包含了除 23 个经典的 GoF 模式之外的许多实用模式。新增的模式有:7 个基本模式、4 个集合模式、4 个结构模式和 4 个并发模式。这些新增和常用的模式无疑会给读者提供更多的参考价值。

第二,本书的示例程序均采用 Java 语言,并且可以在 Internet 上获得源代码。由于经典的 GoF 原著是在 Java 尚未出现和流行时出版的,因此原书中的程序代码采用的仅仅是 C++。本书弥补了这一缺憾。

第三,本书通篇采用 UML 标准图表作为描述工具,使得程序代码、设计模式以及文字说明能很好地融为一体;而且,本书还专门用一章的篇幅对 UML 统一建模语言进行介绍,方便了对 UML 并不十分了解的一部分初学者使用。

第四,作者在本书中花了很多精力对不同但相关或相似的模式进行比较或对照。这样的安排使得读者能更牢固地掌握每个设计模式的本质思想。

第五,本书的最后一章为“案例研究”,作者把多个模式放在一个实际的应用场合里一起配合工作。这样的安排使读者能更好地对各种模式综合运用。

第六,几乎在每一章中,作者都给出了习题,以加强读者对所学内容的理解和对模式的实际使用能力。因此,从这个角度上看,这本书更像是一本教科书。

参加本书翻译工作的有 Sun 中国工程研究院的王卫军、楚宁志、李志宏、曲靖、邹林志、孙阳和吕军,由吕军负责协调和统稿。翻译工作得到了王垦耀院长、交流对外合作部佟佳珏经理和项目经理荣文铮的大力支持和帮助,在此一并表示感谢。由于时间和水平有限,书中难免出现错误或笔误,恳请专家和读者不吝指正。

目 录

第一部分 设计模式简介

第1章 设计模式的起源与历史	2	关于本书	4
从建筑学模式到软件设计模式	2	源代码	4
何谓设计模式	2	源代码声明	4
关于设计模式的更多定义	2		

第二部分 UML

第2章 UML:简介	6	异常(Exception)	9
结构图	6	注释(Note)	9
行为图	6	通用化(Generalization)	9
模型管理图	6	接口(Interface)	9
类图	6	接口实现(Realization)	9
类(Class)	7	依赖(Dependency)	10
内部类(inner class)	7	类的联合关系(Class association)	11
访问关键字(Access specifier)	8	序列图	12
静态(Static)	8	对象(Object)	12
抽象类(Abstract Class)/		消息(Message)	12
方法(Method)	8	自我调用(Self Call)	13

第三部分 基本模式

第3章 接口	16	第6章 存取器方法	29
说明	16	说明	29
示例	17	存取器方法的命名	29
习题	21	示例	30
第4章 抽象父类	22	直接引用与存取器方法的比较	31
说明	22	习题	33
示例	23	第7章 常量数据管理器	34
抽象父类与接口	25	说明	34
习题	25	示例	34
第5章 私有方法	27	习题	37
说明	27	第8章 不变对象	38
示例	27	说明	38
习题	28	示例	42

习题	43	示例	44
第 9 章 管程	44	习题	45
说明	44		

第四部分 创建模式

第 10 章 工厂方法	48	说明	71
说明	48	浅复制与深复制	71
示例	50	浅复制示例	72
习题	53	深复制示例	74
第 11 章 单例	54	示例 I	76
说明	54	设计 HostingPlanKit 类的重要部分	78
职责	54	示例 II	79
示例	54	重新设计 UserAccount 类	81
使构造器成为专用构造器	55	创建原型工厂类	82
访问实例的静态公共接口	55	习题	83
习题	57	第 14 章 构造者	84
第 12 章 抽象工厂	58	说明	84
说明	58	示例 I	86
抽象工厂与抽象方法	58	边注	90
示例 I	59	返回到应用示例	90
示例 II	66	示例 II	93
运行应用时的逻辑流	69	示例 III	98
习题	70	习题	101
第 13 章 原型	71		

第五部分 集合模式

第 15 章 合成	104	客户/容器交互	114
说明	104	外部过滤迭代子示例	115
示例	104	习题	120
设计方法 1	104	第 17 章 享元	122
FileComponent	104	说明	122
DirComponent	105	如何使用 Java 设计享元	122
设计方法 2	106	设计重点	123
习题	109	示例	125
第 16 章 迭代子	111	设计方法 1	126
说明	111	设计方法 2	131
Java 中的迭代子	111	习题	134
过滤迭代子	112	第 18 章 访问者	136
内部迭代子和外部迭代子	112	说明	136
内部迭代子示例	112	设计思想 1	136

设计思想 2	136	设计方法 4(访问者模式)	139
在对象集上定义新的操作	137	应用流程	143
在对象集上添加新类型的		在订单对象集中定义一种新的	
对象	137	操作	143
示例	137	在对象集中加入一种新的订单	
设计方法 1	137	类型	144
设计方法 2	138	习题	146
设计方法 3(合成模式)	139		

第六部分 结构模式

第 19 章 装饰器	148	代理和其他模式的对比	186
说明	148	代理—装饰器	186
装饰器所具有的特性	148	代理—外观	187
示例	148	代理—责任链	187
具体的日志装饰器	151	RMI 简介	187
HTMLLogger	151	RMI 组件	187
EncryptLogger	152	RMI 通讯机制	188
添加新的消息日志记录器	153	RMI 和代理模式	188
添加新的装饰器	154	示例	189
习题	154	附注	196
第 20 章 适配器	155	编译和部署	196
说明	155	习题	198
类适配器与对象适配器的比较	155	第 24 章 桥接	199
类适配器	155	说明	199
对象适配器	155	示例	200
示例	156	抽象体实现设计	201
设计为对象适配器的地址适		抽象体接口设计	202
配器	161	抽象体接口类的设计要点	203
习题	164	桥接模式和适配器模式	205
第 21 章 责任链	166	习题	206
说明	166	第 25 章 虚代理	207
示例	166	说明	207
习题	173	优点	207
第 22 章 外观	174	缺点	207
说明	174	示例	207
示例	175	习题	210
重要提示	184	第 26 章 计数代理	211
习题	184	说明	211
第 23 章 代理	185	示例	211
说明	185	习题	213

第 27 章 聚合强制器	214	finally 语句	218
说明	214	示例	219
示例	214	理想的情形	219
设计方法 1(按需初始化)	214	异常情形 1	219
设计方法 2(早期初始化)	215	异常情形 2	220
设计方法 3(final 变量)	215	习题	222
习题	217	第 29 章 对象缓存器	223
第 28 章 显式对象释放	218	说明	223
说明	218	示例	223
finalize 方法	218	习题	226

第七部分 行为模式

第 30 章 命令	230	说明	267
说明	230	示例	267
示例 1	232	中缀表达式到后缀表达式转换	
示例 2	236	(见清单 34.8)	272
应用流程	238	构造树结构(见清单 34.9)	274
习题	241	后序遍历树	275
第 31 章 调停者	242	附加注释	275
说明	242	中缀到后缀转换	275
调停者模式与外观模式的对比	243	转换逻辑	275
示例 1	243	二进制树遍历技术	277
Mediator 的客户用法	247	习题	278
界面对象:调停者交互	248	第 35 章 状态	279
示例 2	248	说明	279
习题	250	有状态的对象:示例	279
第 32 章 备忘录	251	示例	280
说明	251	习题	290
示例	251	第 36 章 策略	291
DataConverter(原发者)	252	说明	291
DCClient(客户)	254	策略与其他方法的比较	291
MementoHandler	255	策略模式与状态模式的比较	292
习题	257	示例	292
第 33 章 观察者	258	SimpleEncryption	292
说明	258	CaesarCypher	293
增加新的观察者	258	SubstitutionCypher	293
示例	259	CodeBookCypher	293
目标-观察者联系	262	习题	299
习题	266	第 37 章 空对象	300
第 34 章 解释器	267	说明	300

示例	300	习题	312
习题	304	第 39 章 对象认证器	313
第 38 章 模版方法	305	说明	313
说明	305	示例	313
虚类	305	习题	317
具体类	305	第 40 章 通用属性注册表	318
示例	306	说明	318
附加注释	311	示例	321
关于 Mod 10 数字检查算法	311	习题	324

第八部分 并发模式

第 41 章 临界区	326	说明	334
说明	326	示例	335
示例	326	wait() 和 notify() 在 ParkingLot 类	
方法 1(临界区)	327	设计里的使用	336
方法 2(提前初始化)	327	习题	337
习题	328	第 44 章 读写锁	338
第 42 章 协同锁序	329	说明	338
说明	329	ReadWriteLock 类的设计要点	340
示例	330	示例	340
习题	332	习题	343
第 43 章 安全挂起	334		

第九部分 案例研究

第 45 章 案例研究:网络主机服务公司		任务级别	352
应用案例	346	错误处理	353
目标	346	企业服务设计	353
KPS 公司解决方案:简要介绍	346	地址验证	353
需求	346	信用卡服务	354
功能需求	346	搜索管理	355
技术需求	347	客户管理	355
业务目标以及它们之间的关系	347	结论	356
应用服务处理框架	348	附录 A 设计模式清单	357
企业服务层次	349	附录 B 参考书目	359

第一部分 设计模式简介

第 1 章 设计模式的起源与历史

在 20 世纪 70 年代后期，一个叫 Christopher Alexander 的建筑师在建筑模式领域开展了现在所知最早的一项工作。有一次在试图发现和描绘一些高质量建筑设计的完整性和其中体现的艺术活力时，Alexander 和他的同事们研究了为解决同一个问题而设计出的不同建筑结构，他发现了那些高质量的设计中的相似性。于是，在下面这些书中他用“模式”这个术语来指代这种相似性。

- 一种模式语言：城市、建筑业与建筑物（牛津大学出版社，1977 年）

- 永久的建筑风格（牛津大学出版社，1979 年）

由 Alexander 发现并记录的这些模式仅仅是涉及诸如建筑物、花园和道路之类的建筑学方面的模式。

从建筑学模式到软件设计模式

1987 年，受 Alexander 作品的影响，Kent Beck 和 Ward Cunningham 把建筑学上的模式观点应用于软件的设计和开发。他们利用 Alexander 的一些观点开发了一系列模式，利用 Smalltalk 语言实现了雅致的用户界面。利用这个工作成果，他们在 1987 年召开的面向对象的编程系统、语言 and 应用程序（OOPSLA）研讨会上做了一个以《在面向对象编程中使用模式语言》为题的演讲。从那以后，面向对象领域中的许多名人发表了很多与模式相关的论文和演讲。

1994 年，由 Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides 合作的以《设计模式：可复用的面向对象软件的基本原理》为题目的书籍出版了。这本书解释了模式的用处，同时也使得设计模式得到广泛的普及。这四位作者被合称为“四人小组”（GoF）。在书中，作者们记录了在他们近四年半的工作中发现的 23 个模式。

从那儿以后，许多涉及设计模式和其他软件工程最佳实践方面的书陆续出版。

何谓设计模式

一个设计模式就是一个已被记录的最佳实践或一个解决方案，这个最佳实践或解决方案已被成功应用在许多环境中，它解决了在某种特定情境中重复发生的某个问题。

建筑师 Christopher Alexander 把一个建筑模式描绘成是“针对在某个特定背景和作用力系统中发生的通用问题的常见解决方案”。在这个定义中，术语“背景”指的是一个给定模式能够适用的条件/情境；而作用力系统指的是在这个特定背景下的约束条件集合。

关于设计模式的更多定义

- 设计模式是表达或传达从高质量的设计中所得到的学习成果的一种有效途径。这里

的学习成果是：

——一种共享语言，它用来传达在处理和对待那些常见问题及其解决方案方面获得的经验；

——一种为讨论问题解决方案时方便之用而在系统设计基本原理中采用的通用词汇；
一种复用已获得认识并基于这种认识构建系统的方式，这种方式能在软件可维护性和可复用性方面带来质量上的提高。

■ 一个设计模式并非一项新发明，而是一种文档化的表达方式，用来记录通过在对大量的软件系统的研究学习和构建过程中观察或发现得出的针对某个具体问题的最佳解决方法。

■ 对于设计模式的若干常见错误看法中，有一种看法认为设计模式仅适用于面向对象的环境。事实上，尽管关于设计模式的讨论在典型情况下指的是面向对象的软件开发，但在其他领域它们同样适用。如果对其做一些小的变化，一个对设计模式的具体描述可以指代某些通用的软件设计模式。从前面讲到的设计模式的起源与历史中可以看出，模式在建筑业的早期已经存在，而且远远早于面向对象的设计和面向对象的编程时代。

■ 设计模式并非理论构想。一个设计模式可以被看做一个对于一种可复用的解决方案的包装，而被包装的解决方案已被成功地用来解决一类常见的设计问题。

■ 尽管设计模式指的是已知的解决很多问题的最好方式，然而并非所有的这些最佳实践都被认为是模式。一个最佳实践必须满足所谓“三次规定”才能被认为是一个设计模式。“三次规定”指出：一个给定的解决方案必须被验证是一个重复出现的现象，最好至少已经在三个已知系统中存在。否则，这个解决方案不会被认为是一个模式。这么做的目的是保证社区中的软件专业人员确实将某个模式中描述的解决方案应用在了实际的软件设计问题中。满足“三次规定”意味着一个设计模式确实是针对现实世界中的问题而给出的实用解决方案。

■ 设计模式并不能对现实世界中在软件设计和开发中发现的每个问题都给出解决方案，而是针对某种“特定上下文”中通常都会遇到的某种软件开发问题而给出的简练和可重用的一些解决方案。这就是说，某个设计模式在某种“特定上下文”中能给出最佳解决方案，然而在“不同的上下文”中可能并不能给出一种有效的解决方案。有些情况下，某个设计模式中提出的解决方案可能在一个不同的上下文中根本无法适用。

软件框架（Frameworks）和设计模式易被混淆。它们两者是紧密相关的。表 1.1 列出了两者的相似点和不同点。

表 1.1 设计模式与框架

设计模式	框架
设计模式是在某种特定上下文中，针对一个软件应用程序生命过程中出现的问题而给出的多次适用的解决方案	一个框架是一组软件组件，它们互相协作提供了针对某个给定的问题领域中的应用程序所用到的一种可复用的体系结构
根本目标是：	根本目标是：
• 用来帮助提高以下方面的软件质量：可复用性、可维护性、可扩展性等	• 用来帮助提高以下方面的软件质量：可复用性、可维护性、可扩展性等
• 节省开发时间	• 节省开发时间

(续表)

设计模式	框架
模式本质上是逻辑概念	框架本质上更为具体可见，它们作为软件的形式而存在
模式描述通常是独立于编程语言或具体实现的	由于是作为一种软件的形式而存在，框架是针对具体实现的
模式本质上更为一般笼统，可在几乎任何应用程序中应用	框架提供的是与特定领域相关的功能
一个模式本身并不是以软件组件的形式存在的，它需要在每次使用时被具体实现	框架本身并不是完整的应用程序，然而完整的应用程序可以通过直接继承这些软件组件来构建
模式提供了一种方式用来完成一个“良好的”设计，而且模式也可用来辅助设计框架	设计模式可在一个框架的设计和实现过程中运用，或者说框架是一些设计模式的典型体现

关于本书

本书的目的在于通过轻松易懂的方式结合简单实例来讨论设计模式。本书讨论了包括 GoF 提出的 23 个模式在内的 42 个设计模式。这些模式分为 6 类：

- 7 个基本模式——第三部分（第 3 章到第 9 章）；
- 5 个创建模式——第四部分（第 10 章到第 14 章）；
- 4 个集合模式——第五部分（第 15 章到第 18 章）；
- 11 个构造模式——第六部分（第 19 章到第 29 章）；
- 11 个行为模式——第七部分（第 30 章到第 40 章）；
- 4 个并发模式——第八部分（第 41 章到第 44 章）。

关于每个模式的讨论都以关于这个模式的一段说明开始，后面紧跟一个用 Java 编程语言实现的例子。一个给出的模式如何被应用在这个实例中是结合代码段和 UML 图表（类图、序列图）来具体讨论的。在针对每个模式的讨论结束时，书中给出了几个习题以便加强读者对这个模式的理解。只要有可能，书中对相似的模式进行了比较。

本书中的实例简单易懂，目的是通过实例来强化针对每个模式的解释以使读者更容易理解。

UML 章节部分给出了统一建模语言的概述，并讨论了类图和序列图中的各种元素。

书末的实例学习部分展示了在现实世界中的应用程序设计场景中不同设计模式的整体综合运用。这一部分讨论了如何使用模式来为一个假想的 Web hosting 公司设计一个可重用的应用程序框架。

源代码

本书中的所有实例程序源代码在以下网址可以得到：

http://www.crcpress.com/e_products/downloads/default.asp

源代码声明

作者和出版商未做关于软件适用性方面的任何明指或暗指的表示或保证，其中包括并且不限于暗指的销路方面的保证、针对某种特定目的的适用性或不侵权。作者和出版商均不对使用、修改或分发软件或其衍生软件所带来的破坏性后果负责。

Java 是 Sun Microsystems 公司的商标。Windows 是 Microsoft 公司的注册商标。

第二部分 UML

OMG 是一家为企业应用制定计算机工业标准和规范的非赢利组织。UML 是 OMG 的一个应用建模规范。UML 的主要目的就是简化复杂的软件工程流程。使用 UML，人们能够详细地描述、可视化和建立软件或非软件系统的模型。但是必须注意的是，UML 只是一种建模语言，就是说，它只定义了一些语言和语法，但是并没有去强制规定一些建模的流程。