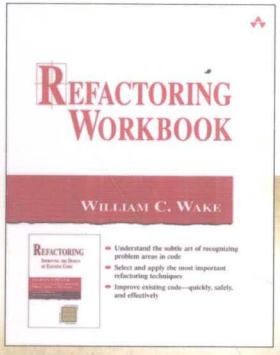
中国科学出版集团
新世纪书局
PEARSON

信息科学
经典著作大系

REFACTORING WORKBOOK

重构手册

《重构》之姊妹篇



（美）William C. Wake / 编著
黄湘情 张 波 / 译

关键重构技术及应用示例，指导您创建高效精妙的代码

科学出版社

重构手册

Refactoring Workbook

[美] William C. Wake 编著
黄湘情 张 波 译

阜阳师范学院图书馆
藏书

科学出版社

图字：01-2010-7769 号

内 容 简 介

本书采用实例手册的方式组织全书内容，帮助读者了解最重要的重构技术并将其应用于代码之中。作者精心组织了一系列问题，通过解决这些问题，让读者不仅在深层次上了解重构，而且会获得自己的一些心得体会。即使你的工作并非重构，本书也有助于你思考如何创建优质的代码。

本书面向有 Java 开发经验的程序员，但 C#和 C++程序员如果对 Java 有一定了解，也可以从本书获得较多受益。

著作权声明

Authorized translation from the English language edition, entitled Refactoring Workbook, 1E, 0321109295 by William C. Wake, published by Pearson Education, Inc, publishing as Addison-Wesley Professional, Copyright © 2004 by 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 SCIENCE PRESS Ltd. Copyright © 2010.

本书中文简体字版由培生教育出版公司授权科学出版社有限责任公司合作出版，未经出版者书面许可，不得以任何形式复制或抄袭本书的任何部分。

本书封面贴有 Pearson Education (培生教育出版集团) 激光防伪标签。无标签者不得销售。

图书在版编目（CIP）数据

重构手册/（美）威克（Wake, W.C.）著；黄湘情，张波译. —北京：科学出版社，2011.4

ISBN 978-7-03-030499-5

Ⅰ. ①重… Ⅱ. ①威… ②黄… ③张… Ⅲ. ①代码－程序设计 Ⅳ. ①TP311.11

中国版本图书馆 CIP 数据核字（2011）第 039386 号

责任编辑：何 武 刘秀青 / 责任校对：刘雪连

责任印刷：新世纪书局 / 封面设计：彭琳君

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

中国科学出版集团新世纪书局策划

三河市少明装订厂印刷

中国科学出版集团新世纪书局发行 各地新华书店经销

*

2011 年 6 月 第 一 版 开本：16 开

2011 年 6 月第一次印刷 印张：13

印数：1—2 500 字数：240 000

定价：42.00 元

（如有印装质量问题，我社负责调换）

前 言 Preface

什么是重构

重构是改善现有代码设计的一门艺术。重构不仅提供了识别问题代码的方法，而且给出了改善此类代码的秘方。

本书目标

本书是一本实战手册，你可以利用本书：

- 尝试找出最重要的“异味（smell）”（即问题）。
- 应用最重要的重构技术。
- 充分考虑如何编写优质代码。
- 尽享重构带来的乐趣。

具体来讲，本书是一本参考指南，提供了以下内容：

- 便于使用的“异味查找器”（smell finder）。
- 用一种标准格式来描述异味。
- 通过附录列出了支持重构的Java工具。
- 通过附录介绍了关键重构技术。

本书面向的读者

重构是一种面向代码的技术，因而这本书是特别为致力于编写和维护代码的程序员所著。

学生们也能够从重构中获益，不过我认为，只有在参与了一个中等规模或更大规模的程序开发，或者作为开发小组中的成员经过一番实战之后，才能体会重构的价值（无论是低年级学生、高年级学生还是研究生，都是如此）。

所需背景知识

如果有Martin Fowler等人所著的《重构：改善既有代码的设计》一书（以下简称《重构》），或者能访问www.Kfactoring.com网站，则可以参考其重构目录，这将对你大有裨益（你可以同时阅读本书和《重构》一书）。在《重构》中，Martin和他的合作伙伴们对多种重构技术进行了循序渐进的讲解，而

前 言 Preface

在本书中，我并不打算做简单的重复。此外，Martin等人还提供了精心设计的实例，并附上了大量精辟的论述和背景材料。有人可能会决定先阅读本书，这也未尝不可，但我还是建议你先看看《重构》。

本书中的示例均使用Java编写，这并不是因为Java是最易于完成重构的语言，而是因为这种语言相当流行，而且好的Java开发环境可以提供自动化重构支持。如果你是对Java有一定了解的C#或C++程序员，那么本书中的大多数问题对你来说也不难理解。不过，在本书后面部分，需要读者动手修改、测试和运行更大规模的程序，这对于使用非Java语言进行开发的程序员来说可能会有一定的困难。

Gamma等人所著的《设计模式》一书将模式描述为“重构的目标”。如果你对此比较熟悉，将有助于你学习本书，因为我会反复提到《设计模式》中所谈到的多种模式。如果你不了解《设计模式》，那么我再向你推荐一本书，即Steve Metsker编写的*Design Patterns Java Workbook*。

如何使用本书

相对来说，实际解决问题的难度要比仅找出解决方案要更大。本书附录提供了部分问题的答案，但是如果你能在查阅答案之前，先尝试自己解决问题的话，将会有更大收获。如果你确实自行解决了这些问题，甚至对我给出的某些答案持有异议，这对你我来说都不失为一大乐事。只是查看答案并一味地接受，真是没有多大乐趣可言。

如果你能够与别人合作（或者参与一个小组）将会更有意思，不过我发现这一愿望往往并不能实现。

后面（更长）的示例需要在一台计算机上完成。在不同计算机环境下阅读程序时，查找问题并确定解决问题的方式可能与本书有所不同。

联系作者

我的邮箱是William.Wake@acm.org。

我还有一个网站：www.xp123.com，这个网站重点讨论极限编程（Extreme Programming，XP），而重构是其中不可或缺的重要组成部分。在这个网站上，重构（以及本书）有专门的介绍页面：www.xp123.com/rwb。

此为试读，需要完整PDF请访问：www.ertongbook.com

我很想知道你做这些练习时的体会，以及你对重构的一般认识，所以请尽可能对我畅所欲言，给我来信。

致谢

写书过程中面临的一个难题是，很难一一谢过这么多曾帮助过我的人。我可能无法记住每一个伸出过援助之手的人的名字，而且必须承认的是，如果我能够采纳所有人的建议，这本书可能会比现在更加出色。

对于本书中的各种设计方案和练习，许多人都给出了建议，有Philippe Antras，Ron Crocker，Sven Gorts，Harris Kirk，Tom Kubit，Paul Michali（他还提供了一个示例），Edmund Schweppe，Steve Wake，Robert Wenner，等等，恕不在此一一列出。特别是Sven Gorts和Tom Kubit为我提供了非常详尽的反馈意见和建议，Ann Anderson，Ken Auer和Don Wells审阅了本书书稿。

我在Gene Codes Corporation（效力于Howard Cash）公司的程序员朋友们总是力图给我提供示例素材（本书中并没有他们编写的代码，不过在我考虑要表述哪些内容时，他们都曾启发过我，而且每个人都至少对一些示例提出了建议）。在此要感谢Lucy Hadden，Jonathan Hoyle，Anna Khizhnyak，Tom Kubit，Greg Poth和Dave Relyea。

这本书显然还要归功于Martin Fowler和Kent Beck先前所做的工作。他们的鼓励对我来说同样很重要。这本书的编写方式受到了Steve Metsker所著的*Design Patterns Java Workbook*一书的启发，与Steve Metsker的交谈也使我获益匪浅。

对于Addison-Wesley/Prentice Hall公司，我要向Mike Hendrickson，Ross Venables，Anne Garcia和Michelle Vincenti致以感谢，特别要感谢我的责任编辑Paul Petrali，以及BooksCraft的Don MacLaren和Ruth Frick的文字润色工作。这本书能够从一份书稿变成一本真正的书倾注了太多人的心血，可能还有些人我并不知道名字，对于他们的辛勤工作，我在此表示最诚挚的谢意。

前 言 Preface

Pearson公司允许我采用了如下一些书中的信息：

- Beck，*EXTREME PROGRAMMING EXPLAINED:EMBRACING CHANGE*，第57页，©2000 Kent Beck版权所有。经Pearson Education 出版公司许可，由Pearson Addison Wesley出版。
- Fowler，*REFACTORING: IMPROVING THE DESIGN OF EXISTING CODE*，封二（重构列表）；封三（异味和常用重构手法），©1999 Addison Wesley Longman公司版权所有。经Pearson Education 出版公司许可，由Pearson Addison Wesley出版。
- Gamma/Helm/Johnson/Vlissides，*DESIGN PATTERNS: ELEMENTS OF REUSABLE OBJECT-ORIENTED SOFTWARE*，第viii和ix页（目录中的模式列表）。©1995 Addison Wesley出版公司版权所有。经Pearson Education出版公司许可，由Pearson Addison Wesley出版。
- Wake，*EXTREME PROGRAMMING EXPLORED*，第24～25页，©2002 Pearson Education公司版权所有。经Pearson Education出版公司许可，由Pearson Addison Wesley出版。

最后，我要感谢我至爱的家人，你们不仅容忍我埋头写作，还不断地给予我鼓励、支持和爱，这是我最大的精神动力。

目 录

目 录

第2部分 类之间的异味

第3部分 待重构的程序

目 录

第4部分 附录

此为试读，需要完整PDF请访问：www.ertongbook.com

第 1 章

本书导读

1.1 概述

本书分为 3 大部分。第 1 部分关注的是类中出现的异味（smell，即问题）。第 2 部分强调类之间的异味。第 3 部分则提供了一些大程序，用于在不同领域实践重构。这些部分之间不时会穿插一些简要说明，或称其为“补充点”，它们是对《重构：改善既有代码的设计》（Martin Fowler 等著，后面简称为 Fowler 的《重构》）一书中重构名录分析的简要介绍，或者是对《设计模式》（Erich Gamma 等所著。同样，后面简称为 Gamma 的《设计模式》）一书中模式的分析说明。

在前两部分中，相关章节主要由异味（暗示潜在问题的一些信号）和实战练习组成。对于异味的描述，我使用了一种标准格式，如下：

异味（Smell）——异味的名字。
症状（Symptoms）——有助于找出问题的线索。
原因（Causes）——说明问题会如何发生。
采取的措施（What to do）——可能的重构。
收益（Payoff）——代码将在哪些方面有所改善。
不适用的情况（Contraindications）——不适合进行重构的情况。

通过这种格式，能更好地将这些有关异味的知识加以应用。即使在完成实战练习后，你也能够参考这些内容，并从中获益。

这里的实战练习形式各异，有些要求你分析代码，有些要求对某种状况

进行评估，还有一些则需要你修改代码。对于基于代码的练习，其中的代码均可在 www.xpl23.com/rwb 网站上找到。

本书中的练习难度各异。对于难度大的练习，我在题目之后标有“有难度”字样，而且你会发现，这些练习的答案也很灵活，可能存在多种答案。

附录 A 给出了大部分练习的解决方案（或有助于找出解决方案的思路）。对于后面的（基于代码的）练习，我不打算提供答案，因为它们是让你修改程序。

1.2 第 1 部分：类中的异味

第 2 章将简要介绍重构周期（refactoring cycle）。第 3 章讨论可度量的异味，它们可以通过简单的长度来度量。在第 4 章中，我们将了解到名字在代码简单性和可理解性方面的意义。第 5 章将探讨代码多余问题。

第 6 章的主题是重复，从许多方面来说，这是需要特别关注的一种核心异味。其他一些异味均可看做是重复的特例。

第 1 部分在第 7 章结束。第 7 章将讨论条件逻辑，即如何使条件和循环语句使用的表达式更清晰明了。

1.3 第 2 部分：类之间的异味

类中的数据有时表示的是丢失的对象，第 8 章将会讨论这个问题。

第 9 章将讨论如何权衡超类和子类的责任。第 10 章将进一步讨论这个问题，且会研究其他类之间的责任平衡问题。当确定如何最佳地连接对象时，有时必须对第 10 章讨论的异味进行权衡。

有时在进行某些修改时，最能体现出重复性问题，我们将在第 11 章介绍这一点。第 12 章作为第 2 部分的结束，讨论了使用库类时存在的一些问题。

1.4 第 3 部分：待重构的程序

本书最后一部分给出了一些有待重构的程序。

- 第 13 章是一个简单的选课系统，它使用了一个数据库。将代码和数据库一起重构是一个正在兴起的研究领域。在该程序的代码中，你可以看到存在大量重复问题需要修正。
- 第 14 章将介绍一个简单的游戏程序。小程序中也不乏需要做决策的地方，这就为大量异味带来了可乘之机。本章还将涉及测试驱动开发。
- 第 15 章所考虑的是权衡责任时面对的一些挑战。我们将使用重构来研究三种访问在线名录的方法。即使只用 3—4 个类，也能得到多种彼此迥异的解决方案。
- 第 16 章涉及一个图形用户界面（Graphical User Interface，GUI）。这个例子展示了一个常见的问题：你希望完成单元测试，以便能安全地进行重构，但是你又必须先进行重构，以使代码可测试。

最后在第 17 章提供了日后你可以在自己的工作中应用的一些练习。另外，这一章还提供了一些参考资源。

1.5 关于练习

要完成这些练习，有一种方法很快捷，即阅读问题，再直接查看答案，因为答案看上去可行，所以你点头称是，如此而已。但这会使你被我的想法所左右。

相应地，还有一种稍微复杂的做法，但却是一种更好地完成练习的方式，即阅读问题，再解决问题，然后才看答案。而且这样不会让你的思维被别人左右。

需要特别指出的是，对于要求你修改的代码，只有亲自动手实践才能有更多的收获。重构是一种需要反复实践的技能。

祝你好运！

第 1 部分

类中的异味

第 2 章

重构周期

本章我们将从两个方面来介绍重构：

1. 识别与修正问题的周期。
2. 在各种重构中所采用的处理类型。

2.1 什么是重构

重构是安全地改善现有代码设计的一门艺术，包括如下含义：

- *重构并不包括对系统的所有修改*。如果修改是对设计的改善，或是增加新功能，则均不能被认为是重构。在创建新代码的过程中，尽管重构可以作为其中的一个环节，但它并不承担增加新特性的任务。例如，极限编程（Extreme Programming 或 XP）（详见 Kent Beck 所著的 *Extreme Programming Explained: Embrace Change*）使用了测试驱动开发，这包括先编写一个测试，然后编写新代码来引入新的特性，最后再通过重构来改善设计。
- *重构并非从头开始重新编写*。尽管有时重新开始更合适，但是重构可以改变平衡点，从而有可能改善代码，而无需承担重新编写的风险。Sven Gorts 与我私下交谈时曾指出：重构能够保护现有代码中所含的信息（知识）不“流失”。
- *重构并不仅仅是某种改善代码的结构更改*。区别重构（refactoring）和更改结构（restructuring）的关键在于，重构力图达到一种安全的（safe）转换。即使是需要修改大量代码的大型（big）重构，也可以将它分解

为较小规模的安全重构（在最佳情况下，如果重构定义得当，还可以做到自动化重构）。如果所做的修改使得代码在超出一个工作期时就无法正常工作或运行（即不能通过测试），那么就不能认为这种修改是重构。

- *重构会改变预先设计（up-front design）和紧急设计（emergent design）之间的平衡点*。预先设计是在实现之前完成的设计，紧急设计是在实现之中进行的设计。如何对预先设计和紧急设计加以权衡，关键在于对于代码中可能存在的问题，我们能够做出何等程度的预测（anticipate）或估计（assess）。另外，以下两种做法哪种更简单也对权衡有着影响：即是先进行设计，再将设计“翻译”为代码，还是先编写代码，再对代码加以改善。重构可以降低紧急设计方法的开销和风险。（改善的程度究竟如何，人们对此可能会有不同的看法，但是重构确实会对此有所改善，相信这一点所有人都认同。）
- *重构规模可大可小*。许多重构规模都很小。一般说来，小型重构应用于较“完美”的情况，此时很少需要进行大型重构。即使应用大型重构，对所采用的方法也存在要求，即不能在进行重构的很长时间内（如长达 6 个月）没有任何新特性的增加，而是应当在进行重构的同时保证系统在任何时候都能正常运行。

2.2 异味就是问题

异味（smell），特别是代码异味（code smell），是一些指出代码中潜在问题的警示信号。并非所有异味都表示确实存在的问题，但是对于大多数异味，都有必要进行查看并做出相应决策。

有些人不喜欢用“异味”一词，而更倾向于把它们称作潜在问题（potential problem）或缺陷（flaw），不过，我认为“异味”这种说法很贴切。想象一下，如果冰箱里面有些东西发霉变质，而你正打开这个冰箱，那将是一种什么情况。有些东西的味道可能已经很重了（说明这些东西彻底坏了），显然我们知道该如何处理它。还有一些味道可能并不大，尚不能确定问题是出在原来剩下的青豆上，还是烧鸡有问题。冰箱里可能还有一些东西也已经坏了，但是并没有发出任何臭味。代码异味与此可谓异曲同工：有些很明显，有些

此为试读，需要完整PDF请访问：www.ertongbook.com