软 件 工 程 系 列

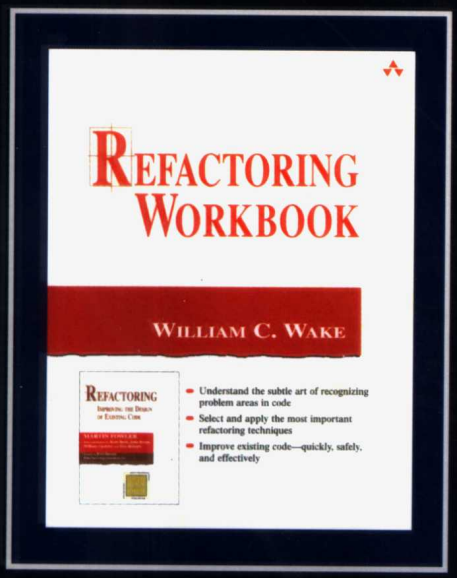
PEARSON
Addison Wesley

Refactoring Workbook

重构手册

和《重构》中文版配套使用效果最佳!

[美] William C. Wake 著
林琪 江健 译



- 体会将代码中有问题之处尽收眼底的精妙
- 如何选择和应用最重要的重构技术
- 大量实例助你快速安全高效地改善既有代码

Pearson Education

中国电力出版社
www.infopower.com.cn

软 件 工 程 系 列

Refactoring Workbook

重构手册

[美] William C. Wake 著
林琪 江健 译

中国电力出版社
www.infopower.com.cn

Refactoring Workbook (ISBN 0-321-10929-5)
William C. Wake
Copyright ©2004 Pearson Education, Inc.
Original English Language Edition Published by Pearson Education, Inc.
All rights reserved.
Translation edition published by PEARSON EDUCATION ASIA LTD and CHINA ELECTRIC POWER PRESS, Copyright © 2004.

本书翻译版由 Pearson Education 授权中国电力出版社在中国境内（香港、澳门特别行政区和台湾地区除外）独家出版、发行。
未经出版者书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 Pearson Education 防伪标签，无标签者不得销售。
北京市版权局著作权合同登记号 图字：01-2004-1299 号

图书在版编目（CIP）数据
重构手册 /（美）韦克著；林琪，江健译. —北京：中国电力出版社，2004
（软件工程系列）
ISBN 7-5083-2278-9
Ⅰ.重... Ⅱ.①韦...②林...③江... Ⅲ.代码－程序设计 Ⅳ.TP311.11
中国版本图书馆 CIP 数据核字（2004）第 050382 号

丛 书 名：软件工程系列
书　　名：重构手册
编　　著：（美）William C. Wake
翻　　译：林 琪　江 健
责任编辑：牛贵华
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：16　　字　　数：235 千字
书　　号：ISBN 7-5083-2278-9
版　　次：2004 年 6 月北京第 1 版　　2004 年 6 月第 1 次印刷
定　　价：29.80 元
版权所有　翻印必究

谨献给

Booker Tyler Wake, 1913—2002

我最为崇敬的叔父

前　言

什么是重构？

*重构是改善既有代码设计的一门艺术。*重构不仅提供了相应途径以找出存在问题的代码，而且还对症下药，针对如何改善此类代码开出了处方。

本书目标

本书是一本实例手册，专门用来帮助读者达到以下目的：

- 尝试找出最重要的“坏味道（*smell*）”（即，存在的问题）
- 应用最为重要的重构技术
- 充分考虑如何创建优质的代码
- 尽享重构带来的乐趣！

具体来讲，本书是一本参考书，提供了以下内容：

- 便于使用的坏味道嗅探工具（*smell finder*）
- 描述坏味道的一种标准格式
- 附录中给出了支持重构的 Java 工具列表
- 附录中还提供了对关键重构技术的介绍

本书面向的对象

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

此为试读，需要完整PDF请访问：www.ertongbook.com

学生们也能够从重构中大为获益，尽管我认为，只有当他们有机会参与开发一个中等规模或更大型的程序，或者作为成员之一在开发小组中经过一番历练之后，才会对重构的价值有所体会（无论是低年级学生、高年级学生还是研究生，都免不了有此磨练过程）。

所需背景知识

如果你手边有 Martin Fowler 等人所著的《重构：改善既有代码的设计》[①]一书（以下简称为《重构》），或者至少能够访问 *www.refactoring.com* 网站，则可以参考其重构名录，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 和他的合作伙伴们为多种重构技术给出了循序渐进的讲解，而在本书中，我并不打算做简单的重复，因为倘若如此不免沦为“鹦鹉学舌”。此外，Martin 等人还提供了一个精心设计的实际示例，并附有大量精辟的论述和背景材料。也许有人希望在不了解《重构》的情况下读完本书，这并非完全不可以，但我对此持保留态度，还是建议你能够阅读《重构》，这会使你获益匪浅。

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，而且最好的 Java 开发环境可以提供自动化重构支持。如果一个 C#或 C++程序员对 Java 有足够的了解，那么这里的大多数问题对他来说也不难理解。不过，在本书后面的部分中，你将着手修改、测试和运行更大规模的程序，这对于使用非 Java 的其他语言进行开发的程序员来说可能会产生一定的困难。

Gamma 等人所著的《设计模式》一书将模式描述为“重构的目标”。如果你对该书中的有关思想有所熟悉，则将有助于你学习本书，因为我将反复提到《设计模式》中所谈到的多种模式。如果你对《设计模式》尚不了解，那么我还将推荐另一本书，即 Steve Metsker 所著的《设计模式 Java 实例手册》[②]。

如何使用本书？

相对于找出一个解决方案，实际解决问题的难度更大。在本书的后面提供了对部分问题的解答，但是在“偷看”答案之前，如果能够先行尝试解决问题，你将得

① 该书中文版已由中国电力出版社出版。
② 该书中文版已由中国电力出版社出版。

到更大收获。倘若你确实自行解决了这些问题，甚至还有可能会对我的某些答案持有异议，这对你我来说都不失为一大乐事。毕竟如果你只是查看答案，并一味地惟命是从，那真是没有多大的乐趣可言。

我想如果能够与别人精诚合作（可以有一个合作伙伴，也可以参与一个小组），将会更有意思，不过我发现这一愿望往往并不能兑现。

后面（更长）的示例需要在一台计算机上完成。当在你的环境下阅读一个程序，寻找其中存在的问题并确定如何加以解决时，很可能与在此所述有所不同。

可以联系作者吗？

当然可以：*William.Wake@acm.org*。

我还有一个网站：*www.xp123.com*。这个网站所关注的是极限编程（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）享有其自己的一席之地：*www.xp123.com/rwb*。

我很想知道你对这些练习的体会，另外你对重构的一般认识也是我所感兴趣的，所以尽可以对我畅所欲言。

致谢

编写这样一本书，最大的难题可谓是很难对提供过帮助的人逐一谢过。要知道，我们无法记住每一个伸出过援手的人的名字，而且必须承认，如果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（或书中的练习），许多人都提出了建议，他们包括 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，还有其他一些人恕不在此一一列出。特别值得一提的是 Sven Gorts 和 Tom Kubit，他们为我提供了尤其详尽的反馈意见和建议。Ann Adnerson、Ken Auer 和 Don Wells 则审阅了本书书稿。

我在 Gene Codes Corporation（效力于 Howard Cash）的程序员朋友们总是担心他

们的某些做法会成为书中反例的素材（这里并没有他们的代码，不过在我考虑要表述哪些内容时，他们都曾对我有所启迪，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。这本书所采用的形式则是受了 Steve Metsker 所著《设计模式 Java 实例手册》一书的启发；与 Steve Metsker 的讨论也使我获益匪浅。

对于 Addison-Wesley/Prentice Hall 公司，我要向 Mike Hendrickson、Ross Venables、Anne Garcia、Michelle Vincenti 致以感谢，特别要感谢我的责任编辑 Paul Petralia。BooksCraft 的 Don MacLaren 和 Ruth Frick 对文字做了相当大的改进。这本书能够从一份书稿变成一本真正的书，其中一定倾注了许多人的心血；我并不知道你们的名字，但是对于你们的辛勤工作，在此表示最诚挚的谢意。

Pearson 公司允许我采用了其他一些书中的信息，如下所列：

- Beck，《EXTREME PROGRAMMING EXPLAINED: EMBRACING CHANGE》（《解析极限编程——拥抱变化》）[①]，第 57 页，©2000 Kent Beck 版权所有。经 Pearson Education 出版公司许可，由 Pearson Addison Wesley 重印。
- Fowler，《REFACTORING: IMPROVING THE DESIGN OF EXISTING CODE》（《重构：改善既有代码的设计》），封二（重构列表）；封三（坏味道和常用重构手法），©1999 Addison Wesley Longman 公司版权所有。经 Pearson Education 出版公司许可，由 Pearson Addison Wesley 重印。
- Gamma/Helm/Johnson/Vlissides，《DESIGN PATTERNS: ELEMENTS OF REUSABLE OBJECT-ORIENTED SOFTWARE》第 viii 和 ix 页（内容表中的模式列表）。©1995 Addison Wesley 出版公司版权所有。经 Pearson Education 出版公司许可，由 Pearson Addison Wesley 重印。
- 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 重印。

最后，绝对不能忘记我的家庭，如果没有向你们致以感谢，怎么能说得过去！你们不仅容忍了我的埋首写作，还不断地给我鼓励、支持和爱。有了这些，夫复何求？

① 该书影印版已由中国电力出版社出版。

目 录

第 2 部分　类之间的坏味道

第 3 部分 待重构的程序

此为试读，需要完整PDF请访问：www.ertongbook.com

第 4 部分 附录

第1章　路线图

1.1　概述

本书分为三大部分。第 1 部分所关注的是出现在类*之中*的坏味道（*smell*，即问题）。第 2 部分强调出现在类*之间*的坏味道。第 3 部分则提供了一些大规模的程序，可以用于实践不同领域的重构。贯穿于这些部分中，不时会出现一些简要说明，我称之为中场休息（interlude），这是对《重构：改善既有代码的设计》（Martin Fowler 等所著，以后将称之为 Fowler 的《重构》）中重构名录所做分析的简要介绍，或者是对《设计模式》（Erich Gamma 等所著。同样，后面将称之为 Gamma 的《设计模式》）中模式的分析说明。

在前两部分中，有关章节主要由坏味道（潜在问题所具有的一些警示信号）和实战（练习）组成。对于坏味道的描述，我采用了一种标准格式：

坏味道（Smell）——坏味道的名字
症状（Symptoms）——有助于找出问题的线索
原因（Causes）——对问题如何会发生的说明
采取的措施（What to do）——可能的重构
收益（Payoff）——代码在哪些方面有所改善
不适用的情况（Contraindications）——在何种情况下不适于进行重构

通过采用这种格式，可以更好地将这些有关坏味道的说明加以使用，即使你已经完成了实战练习，也能够从中有所参考。

这里的实战练习形式各异；有些要求你分析代码，另外一些要求对某种状况进行评估，还有一些则需要你对代码加以修改。对于基于代码的练习，其代码均可在 *www.xp123.com/rwb* 网站上在线找到。

并非所有练习都难度相当。对于难度大的练习，在题号和题目之间标有文字“有难度”；你会看到，这些练习的答案也有很大的调整余地，即可能存在多种答案。

附录A中为大多数练习提供了相应的解决方案（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，则并不打算为之提供答案，因为其目的正是要求你来修改程序。

1.2 第1部分：类之中的坏味道

在第2章中，我们将简要地介绍重构周期（refactoring cycle）。第3章所讨论的是可加以量度的坏味道，即能够由简单的长度（*length*）来度量。在第4章中，我们将了解到名字对于代码的简单性和理解性有着怎样的作用。第5章将考虑不必要代码的有关问题。

第6章的主题是重复（duplication），从许多方面来说，这也是需要特别关注的一种主要的坏味道，而且另外的一些坏味道均可以被看作是重复的特例。

第1部分以第7章作为结束，在此将讨论条件逻辑：即对于条件和循环语句所用的表达式，如何使之更为清晰明了。

1.3 第2部分：类之间的坏味道

类中的数据有时所表示的是“丢失”的对象，即相应对象已经无从引用；第8章将会考虑这个问题。

第9章将讨论如何对超类和子类的责任加以权衡，而第10章将进一步考虑这个问题，即研究如何让类和其他类分担责任以期平衡。当确定如何最佳地连接对象时，有时必须对第10章所述的一些坏味道进行折衷。

若试图做出某些修改，则此时最能够体现出重复的存在，我们将在第11章了解到这一点。第12章作为第2部分的结束章节，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

本书最后一部分提供了一些有待重构的程序。

第13章是一个简单的选课系统，其中使用了一个数据库。将代码和数据库一同重构是一个正在兴起的研究领域；在该程序的代码中，可以看到存在大量的重复问

题需要加以修正。

第 14 章将介绍一个简单的游戏程序。即使是一个小型的程序，其中也不乏需要做抉择的地方，这就为大量坏味道带来了可乘之机。在此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不同的方法，以建立一个在线编目。即使只用了 3～4 个类，仍可以得到多种彼此迥异的解决方案。

第 16 章涉及一个图形化用户界面（graphical user interface，GUI）。这个例子展示了一个常见的问题：你可能希望完成单元测试，从而能够安全地进行重构，但是又必须先行重构以使代码可供测试（译者注：正如“鸡生蛋，还是蛋生鸡”）。

最后，第 17 章提供了一些练习，可以将其应用于你自己的工作中，另外在此还对进一步阅读哪些资料有所建议。

1.5　关于练习

要完成这些练习，有一条捷径：即阅读问题，再直接查看答案，因为答案看上去可行，所以点头称是，如此而已。但这会使你被我的想法所左右。

相应地，还有一种稍显艰难的做法，却是一种更好的完成练习的方式：即阅读问题，再解决问题，在此之后才查看答案。这样才更有机会让你自由的思维不受别人的左右。

特别需要指出的是，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　类之中的坏味道

此为试读，需要完整PDF请访问：www.ertongbook.com