

Robert C. Martin Series

Clean Code

A Handbook of Agile Software Craftsmanship



Foreword by James O. Coplien

Robert C. Martin



Clean Code

Robert C. Martin Series

The mission of this series is to improve the state of the art of software craftsmanship. The books in this series are technical, pragmatic, and substantial. The authors are highly experienced craftsmen and professionals dedicated to writing about what actually works in practice, as opposed to what might work in theory. You will read about what the author has done, not what he thinks you should do. If the book is about programming, there will be lots of code. If the book is about managing, there will be lots of case studies from real projects.

These are the books that all serious practitioners will have on their bookshelves. These are the books that will be remembered for making a difference and for guiding professionals to become true craftsman.

Managing Agile Projects

Sanjiv Augustine

Agile Estimating and Planning

Mike Cohn

Working Effectively with Legacy Code

Michael C. Feathers

Agile Java™: Crafting Code with Test-Driven Development

Jeff Langr

Agile Principles, Patterns, and Practices in C#

Robert C. Martin and Micah Martin

Agile Software Development: Principles, Patterns, and Practices

Robert C. Martin

Clean Code: A Handbook of Agile Software Craftsmanship

Robert C. Martin

UML For Java™ Programmers

Robert C. Martin

Fit for Developing Software: Framework for Integrated Tests

Rick Mugridge and Ward Cunningham

Agile Software Development with SCRUM

Ken Schwaber and Mike Beedle

Extreme Software Engineering: A Hands on Approach

Daniel H. Steinberg and Daniel W. Palmer

For more information, visit informit.com/martinseries



Clean Code

A Handbook of Agile Software Craftsmanship

The Object Mentors:

Robert C. Martin

Michael C. Feathers Timothy R. Ottinger
Jeffrey J. Langr Brett L. Schuchert
James W. Grenning Kevin Dean Wampler
Object Mentor Inc.

*Writing clean code is what you must do in order to call yourself a professional.
There is no reasonable excuse for doing anything less than your best.*



Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City



Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales
international@pearsoned.com

Includes bibliographical references and index.

ISBN 0-13-235088-2 (pbk. : alk. paper)

1. Agile software development. 2. Computer software—Reliability. I. Title.

QA76.76.D47M3652 2008

005.1—dc22

2008024750

Copyright © 2009 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax: (617) 671-3447

ISBN-13: 978-0-13-235088-4

ISBN-10: 0-13-235088-2

Text printed in the United States on recycled paper at Courier in Stoughton, Massachusetts.
First printing July, 2008



For Ann Marie: The ever enduring love of my life.



This page intentionally left blank



Contents

| | |
|--|-----------|
| Foreword | xix |
| Introduction | xxv |
| On the Cover..... | xxix |
| Chapter 1: Clean Code..... | 1 |
| There Will Be Code | 2 |
| Bad Code | 3 |
| The Total Cost of Owning a Mess | 4 |
| The Grand Redesign in the Sky..... | 5 |
| Attitude | 5 |
| The Primal Conundrum..... | 6 |
| The Art of Clean Code?..... | 6 |
| What Is Clean Code? | 7 |
| Schools of Thought | 12 |
| We Are Authors | 13 |
| The Boy Scout Rule | 14 |
| Prequel and Principles | 15 |
| Conclusion..... | 15 |
| Bibliography..... | 15 |
| Chapter 2: Meaningful Names | 17 |
| Introduction | 17 |
| Use Intention-Revealing Names | 18 |
| Avoid Disinformation | 19 |
| Make Meaningful Distinctions | 20 |
| Use Pronounceable Names..... | 21 |
| Use Searchable Names | 22 |



| | |
|---|----|
| Avoid Encodings | 23 |
| Hungarian Notation | 23 |
| Member Prefixes..... | 24 |
| Interfaces and Implementations | 24 |
| Avoid Mental Mapping | 25 |
| Class Names | 25 |
| Method Names..... | 25 |
| Don't Be Cute | 26 |
| Pick One Word per Concept..... | 26 |
| Don't Pun | 26 |
| Use Solution Domain Names | 27 |
| Use Problem Domain Names | 27 |
| Add Meaningful Context | 27 |
| Don't Add Gratuitous Context..... | 29 |
| Final Words | 30 |
| Chapter 3: Functions | 31 |
| Small!..... | 34 |
| Blocks and Indenting..... | 35 |
| Do One Thing..... | 35 |
| Sections within Functions | 36 |
| One Level of Abstraction per Function | 36 |
| Reading Code from Top to Bottom: <i>The Stepdown Rule</i> | 37 |
| Switch Statements | 37 |
| Use Descriptive Names..... | 39 |
| Function Arguments..... | 40 |
| Common Monadic Forms..... | 41 |
| Flag Arguments | 41 |
| Dyadic Functions..... | 42 |
| Triads..... | 42 |
| Argument Objects..... | 43 |
| Argument Lists | 43 |
| Verbs and Keywords | 43 |
| Have No Side Effects | 44 |
| Output Arguments | 45 |
| Command Query Separation | 45 |

| | |
|---|----|
| Prefer Exceptions to Returning Error Codes | 46 |
| Extract Try/Catch Blocks | 46 |
| Error Handling Is One Thing..... | 47 |
| The Error.java Dependency Magnet | 47 |
| Don't Repeat Yourself | 48 |
| Structured Programming | 48 |
| How Do You Write Functions Like This? | 49 |
| Conclusion..... | 49 |
| SetupTeardownIncluder | 50 |
| Bibliography..... | 52 |
| Chapter 4: Comments | 53 |
| Comments Do Not Make Up for Bad Code..... | 55 |
| Explain Yourself in Code | 55 |
| Good Comments | 55 |
| Legal Comments..... | 55 |
| Informative Comments..... | 56 |
| Explanation of Intent..... | 56 |
| Clarification..... | 57 |
| Warning of Consequences | 58 |
| TODO Comments..... | 58 |
| Amplification..... | 59 |
| Javadocs in Public APIs..... | 59 |
| Bad Comments | 59 |
| Mumbling | 59 |
| Redundant Comments | 60 |
| Misleading Comments..... | 63 |
| Mandated Comments..... | 63 |
| Journal Comments..... | 63 |
| Noise Comments | 64 |
| Scary Noise | 66 |
| Don't Use a Comment When You Can Use a Function or a Variable..... | 67 |
| Position Markers..... | 67 |
| Closing Brace Comments | 67 |
| Attributions and Bylines..... | 68 |



| | |
|---|------------|
| Commented-Out Code..... | 68 |
| HTML Comments | 69 |
| Nonlocal Information..... | 69 |
| Too Much Information | 70 |
| Inobvious Connection..... | 70 |
| Function Headers..... | 70 |
| Javadocs in Nonpublic Code | 71 |
| Example..... | 71 |
| Bibliography..... | 74 |
| | |
| Chapter 5: Formatting | 75 |
| The Purpose of Formatting | 76 |
| Vertical Formatting | 76 |
| The Newspaper Metaphor | 77 |
| Vertical Openness Between Concepts | 78 |
| Vertical Density | 79 |
| Vertical Distance | 80 |
| Vertical Ordering | 84 |
| Horizontal Formatting | 85 |
| Horizontal Openness and Density | 86 |
| Horizontal Alignment..... | 87 |
| Indentation..... | 88 |
| Dummy Scopes..... | 90 |
| Team Rules..... | 90 |
| Uncle Bob's Formatting Rules..... | 90 |
| | |
| Chapter 6: Objects and Data Structures | 93 |
| Data Abstraction..... | 93 |
| Data/Object Anti-Symmetry | 95 |
| The Law of Demeter..... | 97 |
| Train Wrecks | 98 |
| Hybrids | 99 |
| Hiding Structure | 99 |
| Data Transfer Objects..... | 100 |
| Active Record..... | 101 |
| Conclusion..... | 101 |
| Bibliography..... | 101 |



| | |
|---|-----|
| Chapter 7: Error Handling | 103 |
| Use Exceptions Rather Than Return Codes | 104 |
| Write Your Try-Catch-Finally Statement First | 105 |
| Use Unchecked Exceptions | 106 |
| Provide Context with Exceptions | 107 |
| Define Exception Classes in Terms of a Caller's Needs..... | 107 |
| Define the Normal Flow | 109 |
| Don't Return Null..... | 110 |
| Don't Pass Null | 111 |
| Conclusion..... | 112 |
| Bibliography..... | 112 |
| Chapter 8: Boundaries | 113 |
| Using Third-Party Code..... | 114 |
| Exploring and Learning Boundaries..... | 116 |
| Learning log4j | 116 |
| Learning Tests Are Better Than Free..... | 118 |
| Using Code That Does Not Yet Exist | 118 |
| Clean Boundaries | 120 |
| Bibliography..... | 120 |
| Chapter 9: Unit Tests | 121 |
| The Three Laws of TDD | 122 |
| Keeping Tests Clean | 123 |
| Tests Enable the -ilities..... | 124 |
| Clean Tests | 124 |
| Domain-Specific Testing Language..... | 127 |
| A Dual Standard | 127 |
| One Assert per Test | 130 |
| Single Concept per Test..... | 131 |
| F.I.R.S.T..... | 132 |
| Conclusion..... | 133 |
| Bibliography..... | 133 |
| Chapter 10: Classes | 135 |
| Class Organization | 136 |
| Encapsulation | 136 |



| | |
|--|-----|
| Classes Should Be Small! | 136 |
| The Single Responsibility Principle | 138 |
| Cohesion | 140 |
| Maintaining Cohesion Results in Many Small Classes | 141 |
| Organizing for Change | 147 |
| Isolating from Change | 149 |
| Bibliography | 151 |
| Chapter 11: Systems | 153 |
| How Would You Build a City? | 154 |
| Separate Constructing a System from Using It | 154 |
| Separation of Main | 155 |
| Factories | 155 |
| Dependency Injection | 157 |
| Scaling Up | 157 |
| Cross-Cutting Concerns | 160 |
| Java Proxies | 161 |
| Pure Java AOP Frameworks | 163 |
| AspectJ Aspects | 166 |
| Test Drive the System Architecture | 166 |
| Optimize Decision Making | 167 |
| Use Standards Wisely, When They Add <i>Demonstrable Value</i> | 168 |
| Systems Need Domain-Specific Languages | 168 |
| Conclusion | 169 |
| Bibliography | 169 |
| Chapter 12: Emergence | 171 |
| Getting Clean via Emergent Design | 171 |
| Simple Design Rule 1: Runs All the Tests | 172 |
| Simple Design Rules 2–4: Refactoring | 172 |
| No Duplication | 173 |
| Expressive | 175 |
| Minimal Classes and Methods | 176 |
| Conclusion | 176 |
| Bibliography | 176 |
| Chapter 13: Concurrency | 177 |
| Why Concurrency? | 178 |
| Myths and Misconceptions | 179 |

| | |
|---|-----|
| Challenges | 180 |
| Concurrency Defense Principles | 180 |
| Single Responsibility Principle | 181 |
| Corollary: Limit the Scope of Data | 181 |
| Corollary: Use Copies of Data | 181 |
| Corollary: Threads Should Be as Independent as Possible | 182 |
| Know Your Library | 182 |
| Thread-Safe Collections | 182 |
| Know Your Execution Models | 183 |
| Producer-Consumer..... | 184 |
| Readers-Writers..... | 184 |
| Dining Philosophers | 184 |
| Beware Dependencies Between Synchronized Methods | 185 |
| Keep Synchronized Sections Small | 185 |
| Writing Correct Shut-Down Code Is Hard..... | 186 |
| Testing Threaded Code | 186 |
| Treat Spurious Failures as Candidate Threading Issues | 187 |
| Get Your Nonthreaded Code Working First..... | 187 |
| Make Your Threaded Code Pluggable | 187 |
| Make Your Threaded Code Tunable..... | 187 |
| Run with More Threads Than Processors..... | 188 |
| Run on Different Platforms | 188 |
| Instrument Your Code to Try and Force Failures | 188 |
| Hand-Coded | 189 |
| Automated | 189 |
| Conclusion..... | 190 |
| Bibliography..... | 191 |
| Chapter 14: Successive Refinement..... | 193 |
| Args Implementation | 194 |
| How Did I Do This? | 200 |
| Args: The Rough Draft | 201 |
| So I Stopped | 212 |
| On Incrementalism | 212 |
| String Arguments | 214 |
| Conclusion..... | 250 |



| | |
|--|-----|
| Chapter 15: JUnit Internals | 251 |
| The JUnit Framework..... | 252 |
| Conclusion..... | 265 |
| | |
| Chapter 16: Refactoring SerialDate | 267 |
| First, Make It Work..... | 268 |
| Then Make It Right..... | 270 |
| Conclusion..... | 284 |
| Bibliography..... | 284 |
| | |
| Chapter 17: Smells and Heuristics | 285 |
| Comments | 286 |
| C1: <i>Inappropriate Information</i> | 286 |
| C2: <i>Obsolete Comment</i> | 286 |
| C3: <i>Redundant Comment</i> | 286 |
| C4: <i>Poorly Written Comment</i> | 287 |
| C5: <i>Commented-Out Code</i> | 287 |
| Environment | 287 |
| E1: <i>Build Requires More Than One Step</i> | 287 |
| E2: <i>Tests Require More Than One Step</i> | 287 |
| Functions..... | 288 |
| F1: <i>Too Many Arguments</i> | 288 |
| F2: <i>Output Arguments</i> | 288 |
| F3: <i>Flag Arguments</i> | 288 |
| F4: <i>Dead Function</i> | 288 |
| General | 288 |
| G1: <i>Multiple Languages in One Source File</i> | 288 |
| G2: <i>Obvious Behavior Is Unimplemented</i> | 288 |
| G3: <i>Incorrect Behavior at the Boundaries</i> | 289 |
| G4: <i>Overridden Safeties</i> | 289 |
| G5: <i>Duplication</i> | 289 |
| G6: <i>Code at Wrong Level of Abstraction</i> | 290 |
| G7: <i>Base Classes Depending on Their Derivatives</i> | 291 |
| G8: <i>Too Much Information</i> | 291 |
| G9: <i>Dead Code</i> | 292 |
| G10: <i>Vertical Separation</i> | 292 |
| G11: <i>Inconsistency</i> | 292 |
| G12: <i>Clutter</i> | 293 |