Growing Ob j ect-Oriented Software, Guided by Tests

Steve Freeman and Nat Pryce

AAddison-Wesley

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

Contents

Foreword xv

Preface xvii

Acknowledgments xxi

About the Authors xxiii

Part I: Introduction 1

Chapter 1: What Is the Point of Test-Driven Development? 3

Software Development as a Learning Process 3
Feedback Is the Fundamental Tool 4
Practices That Support Change 5
Test-Driven Development in a Nutshell 6
The Bigger Picture 7
Testing End-to-End 8
Levels of Testing 9
External and Internal Quality 10

Chapter 2: Test-Driven Development with Objects 13

A Web of Objects 13
Values and Objects 13
Follow the Messages 14
Tell, Don't Ask 17
But Sometimes Ask 17
Unit-Testing the Collaborating Objects 18
Support for TDD with Mock Objects 19

Chapter 3: An Introduction to the Tools **21**
Stop Me If You've Heard This One Before 21
A Minimal Introduction to JUnit 4 21
Hamcrest Matchers and assertThat() 24
jMock2: Mock Objects 25

Part II: The Process of Test-Driven Development **29**

Chapter 4: Kick-Starting the Test-Driven Cycle **31**
Introduction 31
First, Test a Walking Skeleton 32
Deciding the Shape of the Walking Skeleton 33
Build Sources of Feedback 35
Expose Uncertainty Early 36

Chapter 5: Maintaining the Test-Driven Cycle **39**
Introduction 39
Start Each Feature with an Acceptance Test 39
Separate Tests That Measure Progress from Those That Catch Regressions 40
Start Testing with the Simplest Success Case 41
Write the Test That You'd Want to Read 42
Watch the Test Fail 42
Develop from the Inputs to the Outputs 43
Unit-Test Behavior, Not Methods 43
Listen to the Tests 44
Tuning the Cycle **45**

Chapter 6: Object-Oriented Style **47**
Introduction 47
Designing for Maintainability 47
Internals vs. Peers 50
No And's, Or's, or But's 51
Object Peer Stereotypes 52
Composite Simpler Than the Sum of Its Parts 53
Context Independence 54
Hiding the Right Information 55
An Opinionated View 56

Chapter 7: Achieving Object-Oriented Design 57
How Writing a Test First Helps the Design 57
Communication over Classification 58

Value Types *59*
Where Do Objects Come From? 60
Identify Relationships with Interfaces *63*
Refactor Interfaces Too 63
Compose Objects to Describe System Behavior 64
Building Up to Higher-Level Programming *65*
And What about Classes? 67

Chapter 8: Building on Third-Party Code 69
Introduction *69*
Only Mock Types That You Own 69
Mock Application Objects in Integration Tests 71

Part III: A Worked Example 73

Chapter 9: Commissioning an Auction Sniper 75
To Begin at the Beginning 75
Communicating with an Auction 78
Getting There Safely *79*
This Isn't Real 81

Chapter 10: The Walking Skeleton 83
Get the Skeleton out of the Closet 83
Our Very First Test 84
Some Initial Choices 86

Chapter 11: Passing the First Test 89
Building the Test Rig 89
Failing and Passing the Test 95
The Necessary Minimum 102

Chapter 12: Getting Ready to Bid 105
An Introduction to the Market 105
A Test for Bidding 106
The AuctionMessageTranslator 112
Unpacking a Price Message 118
Finish the Job 121

Chapter 13: The Sniper Makes a Bid 123
Introducing AuctionSniper 123
Sending a Bid 126
Tidying Up the Implementation 131

Defer Decisions 136
Emergent Design 137

Chapter 14: The Sniper Wins the Auction 139
First, a Failing Test 139
Who Knows about Bidders? 140
The Sniper Has More to Say 143
The Sniper Acquires Some State 144
The Sniper Wins 146
Making Steady Progress 148

Chapter 15: Towards a Real User Interface 149
A More Realistic Implementation 149
Displaying Price Details 152
Simplifying Sniper Events 159
Follow Through 164
Final Polish 168
Observations 171

Chapter 16: Sniping for Multiple Items 175
Testing for Multiple Items 175
Adding Items through the User Interface 183
Observations 189

Chapter 17: Teasing Apart Main 191
Finding a Role 191
Extracting the Chat 192
Extracting the Connection 195
Extracting the SnipersTableModel 197
Observations 201

Chapter 18: Filling In the Details 205
A More Useful Application 205
Stop When We've Had Enough 205
Observations 212

Chapter 19: Handling Failure 215
What If It Doesn't Work? 215
Detecting the Failure 217
Displaying the Failure 218
Disconnecting the Sniper 219
Recording the Failure 221
Observations 225

Part FV: Sustainable Test-Driven Development **227**

Chapter 20: Listening to the Tests **229**
Introduction 229
I Need to Mock an Object I Can't Replace (without Magic) 230
Logging Is a Feature 233
Mocking Concrete Classes 235
Don't Mock Values 237
Bloated Constructor 238
Confused Object 240
Too Many Dependencies 241
Too Many Expectations 242
What the Tests Will Tell Us (If We're Listening) 244

Chapter 21: Test Readability **247**
Introduction 247
Test Names Describe Features 248
Canonical Test Structure 251
Streamline the Test Code 252
Assertions and Expectations 254
Literals and Variables 255

Chapter 22: Constructing Complex Test **Data** **257**
Introduction 257
Test Data Builders 258
Creating Similar Objects 259
Combining Builders 261
Emphasizing the Domain Model with Factory Methods 261
Removing Duplication at the Point of Use 262
Communication First 264

Chapter 23: Test Diagnostics **267**
Design to Fail 267
Small, Focused, Well-Named Tests 268
Explanatory Assertion Messages 268
Highlight Detail with Matchers 268
Self-Describing Value 269
Obviously Canned Value 270
Tracer Object 270
Explicitly Assert That Expectations Were Satisfied 271
Diagnostics Are a First-Class Feature 271

Chapter 24: Test Flexibility **273**
Introduction 273
Test for Information, Not Representation 274
Precise Assertions 275
Precise Expectations 277
"Guinea Pig" Objects 284

Part V: Advanced Topics **287**

Chapter 25: Testing Persistence **289**
Introduction 289
Isolate Tests That Affect Persistent State 290
Make Tests Transaction Boundaries Explicit 292
Testing an Object That Performs Persistence Operations 294
Testing That Objects Can Be Persisted 297
But Database Tests Are S-l-o-w! 300

Chapter 26: Unit Testing and Threads **301**
Introduction 301
Separating Functionality and Concurrency Policy 302
Unit-Testing Synchronization 306
Stress-Testing Passive Objects 311
Synchronizing the Test Thread with Background Threads 312
The Limitations of Unit Stress Tests 313

Chapter 27: Testing Asynchronous Code **315**
Introduction 315
Sampling or Listening 316
Two Implementations 318
Runaway Tests 322
Lost Updates 323
Testing That an Action Has No Effect 325
Distinguish Synchronizations and Assertions 326
Externalize Event Sources 326

Afterword: A Brief History of Mock Objects **329**

Appendix A: jMock2 Cheat Sheet **335**

Appendix B: Writing a Hamcrest Matcher **343**

Bibliography **347**

Index **349**