

# Contents

Foreword . . . . .	ix
Preface . . . . .	xi

## Part I — Unit-Testing Foundations

<b>1. Building Your First JUnit Test . . . . .</b>	<b>3</b>
Reasons to Write a Unit Test	3
Learning JUnit Basics: Our First Passing Test	4
Arrange, Act, and Assert Your Way to a Test	10
Is the Test Really Testing Anything?	12
After	12
<b>2. Getting Real with JUnit . . . . .</b>	<b>13</b>
Understanding What We're Testing: The Profile Class	13
Determining What Tests We Can Write	15
Covering One Path	17
Tackling a Second Test	19
Initializing Tests with @Before Methods	19
How Ya Feelin' Now?	22
After	23
<b>3. Digging Deeper into JUnit Assertions . . . . .</b>	<b>25</b>
Assertions in JUnit	25
Three Schools for Expecting Exceptions	31
After	34
<b>4. Organizing Your Tests . . . . .</b>	<b>35</b>
Keeping Tests Consistent with AAA	35
Testing Behavior Versus Testing Methods	36
Relationship Between Test and Production Code	37

The Value of Focused, Single-Purpose Tests	40
Tests as Documentation	41
More on @Before and @After (Common Initialization and Cleanup)	43
Green Is Good: Keeping Our Tests Relevant	45
After	47

## Part II — Mastering Manic Mnemonics!

<b>5. FIRST Properties of Good Tests</b>	<b>51</b>
FIRST It Helps to Remember That Good Tests Are FIRST	51
[F]IRST: [F]ast!	52
F[I]RST: [I]solate Your Tests	56
FI[R]ST: Good Tests Should Be [R]epeatable	57
FIR[S]T: [S]elf-Validating	59
FIRS[T]: [T]imely	61
After	62
<b>6. What to Test: The Right-BICEP</b>	<b>63</b>
[Right]-BICEP: Are the Results Right?	63
Right-[B]ICEP: Boundary Conditions	65
Remembering Boundary Conditions with CORRECT	67
Right-B[I]CEP: Checking Inverse Relationships	68
Right-BI[C]EP: Cross-Checking Using Other Means	70
Right-BIC[E]P: Forcing Error Conditions	71
Right-BICE[P]: Performance Characteristics	71
After	73
<b>7. Boundary Conditions: The CORRECT Way</b>	<b>75</b>
[C]ORRECT: [C]onformance	76
C[O]RRECT: [O]rdering	77
CO[R]RECT: [R]ange	78
COR[R]ECT: [R]eference	85
CORR[E]CT: [E]xistence	86
CORRE[C]T: [C]ardinality	87
CORREC[T]: [T]ime	89
After	91

## Part III — The Bigger Design Picture

<b>8.</b>	<b>Refactoring to Cleaner Code</b> . . . . .	<b>95</b>
	A Little Bit o' Refactor	95
	Finding Better Homes for Our Methods	98
	Automated and Manual Refactorings	100
	Taking Refactoring Too Far?	102
	After	105
<b>9.</b>	<b>Bigger Design Issues</b> . . . . .	<b>107</b>
	The Profile Class and the SRP	107
	Extracting a New Class	109
	Command-Query Separation	114
	The Cost of Maintaining Unit Tests	115
	Other Design Thoughts	118
	After	121
<b>10.</b>	<b>Using Mock Objects</b> . . . . .	<b>123</b>
	A Testing Challenge	123
	Replacing Troublesome Behavior with Stubs	125
	Changing Our Design to Support Testing	128
	Adding Smarts to Our Stub: Verifying Parameters	128
	Simplifying Testing Using a Mock Tool	130
	One Last Simplification: Introducing an Injection Tool	131
	What's Important to Get Right When Using Mocks	133
	After	134
<b>11.</b>	<b>Refactoring Tests</b> . . . . .	<b>135</b>
	Searching for an Understanding	135
	Test Smell: Unnecessary Test Code	137
	Test Smell: Missing Abstractions	138
	Test Smell: Irrelevant Information	140
	Test Smell: Bloated Construction	142
	Test Smell: Multiple Assertions	143
	Test Smell: Irrelevant Details in Test	144
	Test Smell: Misleading Organization	146
	Test Smell: Implicit Meaning	147
	Adding a New Test	148
	After	149

## Part IV — The Bigger Unit-Testing Picture

<b>12. Test-Driven Development</b>	<b>153</b>
The Primary Benefit of TDD	153
Starting Simple	154
Adding Another Increment	157
Cleaning Up Our Tests	158
Another Small Increment	161
Supporting Multiple Answers: A Small Design Detour	162
Expanding the Interface	164
Last Tests	166
Tests As Documentation	167
The Rhythm of TDD	169
After	169
<b>13. Testing Some Tough Stuff</b>	<b>171</b>
Testing Multithreaded Code	171
Testing Databases	180
After	186
<b>14. Testing on a Project</b>	<b>187</b>
Coming up to Speed	187
Getting on the Same Page with Your Team	188
Convergence with Continuous Integration	190
Code Coverage	192
After	196
<b>A1. Setting Up JUnit in IntelliJ IDEA and NetBeans</b>	<b>197</b>
IntelliJ IDEA	198
NetBeans	202
<b>Index</b>	<b>207</b>