

Inhaltsverzeichnis

1	Einführung	1
1.1	Automatisierte Tests	2
1.2	Der grüne Balken	3
1.3	Funktionale Tests	4
1.4	Nichtfunktionale Tests	9
2	JUnit 3	11
2.1	Testklassen	12
2.2	Testmethoden	14
2.3	Assertion-Methoden	14
2.4	Testfixtures	21
2.5	Testsuites	28
2.6	Zusammenfassung	34
3	JUnit 4	35
3.1	Testklassen und -methoden	36
3.2	Die @Test-Annotation	38
3.3	Assertion-Methoden	42
3.4	Testfixtures mit @Before- und @After-Methoden auf- und abbauen	47
3.5	@Rule und eigene Testaspekte	51
3.6	@RunWith, Parameterized und eigene Runner	60
3.7	Testsuites	64
3.8	Testtheorien	70
3.9	Testgruppen/Testkategorien	74
3.10	Tests überspringen/ignorieren	80
3.11	Zusammenfassung	87

4 Testgetriebene Entwicklung	91
4.1 Einmal rundherum	91
4.2 Einen roten Test schreiben	92
4.3 Den roten Test grün machen	96
4.4 Codereview und Refactoring	106
4.5 ATDD – der Kontext für TDD	111
4.6 Zusammenfassung	114
5 Assertion-Bibliotheken	117
5.1 Hamcrest einbinden	117
5.2 Ein Blick unter die Motorhaube von Hamcrest	120
5.3 Eigene Hamcrest-Matcher schreiben	124
5.4 FEST Fluent Assertions	129
5.5 Zusammenfassung	135
6 Unit-Tests mit Mock-Objekten	137
6.1 Terminologie	137
6.1.1 Dummy-Objekt	138
6.1.2 Pseudo-Objekt	139
6.1.3 Fake-Objekt	141
6.1.4 Stub-Objekt	142
6.1.5 Mock-Objekt	144
6.1.6 Spy-Objekt	145
6.2 Mock-Objekte selbst schreiben	147
6.3 jMock	151
6.4 EasyMock	160
6.5 Mockito	169
6.6 Umgang mit unerwarteten Methodenaufrufen	181
6.7 Mock-Objekte injizieren	187
6.8 Mocken statischer Methoden	191
6.9 PowerMock	194
6.10 Zusammenfassung	199

7 Programmieren gut verständlicher Tests	201
7.1 Organisation und Benennung von Testklassen	201
7.2 Benennung von Testmethoden	207
7.3 Setup-Methoden	210
7.4 Das Test Data Builder Pattern	216
7.5 Der AAA-Stil	219
7.6 Das Page Object Pattern	221
7.7 Assertion-Messages	225
7.8 Zusammenfassung	227
8 Programmieren schneller Tests	229
8.1 Tests schneller machen	230
8.2 Testfixtures schneller machen	239
8.3 Tests zusammenfassen	242
8.4 Das Shared Testfixture Pattern	244
8.5 Tests parallel ausführen	251
8.6 Schnelles Feedback durch optimierte Testreihenfolge	257
8.7 Zusammenfassung	259
9 Tests abseits vom Happy Path	261
9.1 Exceptions im Test auslösen	261
9.2 Testen von Logmeldungen	263
9.3 Testen von Ausgaben auf System.out bzw. System.err	266
9.4 Testen von System.exit	267
9.5 Testen von Exceptions	269
9.6 Zusammenfassung	273
10 Nichtfunktionale Tests	275
10.1 Performance-Tests	275
10.2 Stresstests	283
10.3 Randomized Testing	288
10.4 Architekturtests	290
10.5 Zusammenfassung	296

11 JUnit und Eclipse	297
11.1 Wizards zum Erstellen von Testklassen	297
11.2 JUnit-Tests mit Eclipse ausführen	301
11.3 Erweiterte JUnit-Unterstützung durch das MoreUnit-Plug-in	305
11.4 Testabdeckung visualisieren mit EclEmma	306
12 JUnit und IntelliJ IDEA	309
12.1 Erstellen von Testklassen	310
12.2 JUnit-Tests mit IntelliJ IDEA ausführen	313
12.3 Testabdeckung visualisieren mit IntelliJ IDEA Ultimate Edition	319
13 JUnit und Ant	323
13.1 Die Ant-Tasks <code>junit</code> und <code>junitreport</code>	323
13.2 Testabdeckung messen mit JaCoCo	326
14 JUnit und Maven	331
14.1 JUnit-Tests mit dem Surefire-Plug-in ausführen	331
14.2 Tests parallel ausführen	333
14.3 Die Reihenfolge steuern, in der Tests ausgeführt werden	334
14.4 Nur bestimmte Tests ausführen bzw. bestimmte Tests ausschließen	335
14.5 Ausführen und Debuggen einzelner Tests	338
14.6 Ausführen von Integrationstests mit dem Failsafe-Plug-in	339
14.7 Testabdeckung mit Cobertura oder JaCoCo messen	342
15 Schlusswort	349
Literaturverzeichnis	351
Index	353