Inhaltsverzeichnis

1 Einleitung 1
1.1 Über dieses Buch 1
1.1.1 Motivation 1
1.1.2 Was leistet dieses Buch und was nicht? 2
1.1.3 Wie und was soll mithilfe des Buchs gelernt werden? 2
1.1.4 Wer sollte dieses Buch lesen? 4
1.2 Aufbau des Buchs 4
1.3 Konventionen und ausführbare Programme 6

I Java-Grundlagen, Analyse und Design 11

2 Professionelle Arbeitsumgebung 13
2.1 Vorteile von IDEs am Beispiel von Eclipse 14
2.2 Projektorganisation 16
2.2.1 Projektstruktur in Eclipse 16
2.2.2 Projektstruktur für Maven und Gradle 18
2.3 Einsatz von Versionsverwaltungen 20
2.3.1 Arbeiten mit zentralen Versionsverwaltungen 23
2.3.2 Dezentrale Versionsverwaltungen 28
2.3.3 VCS und DVCS im Vergleich 34
2.4 Einsatz eines Unit-Test-Frameworks 37
2.4.1 Das JUnit-Framework 37
2.4.2 Vorteile von Unit Tests 42
2.5 Debugging 43
2.5.1 Fehlersuche mit einem Debugger 45
2.5.2 Remote Debugging 47
2.6 Deployment von Java-Applikationen 52
2.6.1 Das JAR-Tool im Kurzüberblick 54
2.6.2 JAR inspizieren und ändern, Inhalt extrahieren 55
2.6.3 Metainformationen und das Manifest 56
2.6.4 Inspizieren einer JAR-Datei 59

Michael Inden, Der Weg zum Java-Profi, dpunkt.verlag, ISBN 978-3-86490-483-7

2.7 Einsatz eines IDE-unabhängigen Build-Prozesses 61
2.7.1 Maven im Überblick 64
2.7.2 Builds mit Gradle 69
2.7.3 Vorteile von Maven und Gradle 82
2.8 Weiterführende Literatur 83

3 Objektorientiertes Design 85
3.1 OO-Grundlagen 86
3.1.1 Grundbegriffe 86
3.1.2 Beispielentwurf: Ein Zähler 98
3.1.3 Vom imperativen zum objektorientierten Entwurf 106
3.1.4 Diskussion der OO-Grundgedanken 111
3.1.5 Wissenswertes zum Objektzustand 114
3.2 Grundlegende OO-Techniken 123
3.2.1 Schnittstellen (Interfaces) 123
3.2.2 Basisklassen und abstrakte Basisklassen 128
3.2.3 Interfaces und abstrakte Basisklassen 130
3.3 Wissenswertes zu Vererbung 132
3.3.1 Probleme durch Vererbung 132
3.3.2 Delegation statt Vererbung 138
3.4 Fortgeschrittenere OO-Techniken 142
3.4.1 Read-only-Interface 142
3.4.2 Immutable-Klasse 148
3.4.3 Marker-Interface 153
3.4.4 Konstantensammlungen und Aufzählungen 154
3.4.5 Value Object (Data Transfer Object) 160
3.5 Prinzipien guten OO-Designs 162
3.5.1 Geheimnisprinzip nach Parnas 163
3.5.2 Law of Demeter 163
3.5.3 SOLID-Prinzipien 166
3.6 Formen der Varianz 179
3.6.1 Grundlagen der Varianz 179
3.6.2 Kovariante Rückgabewerte 183
3.7 Generische Typen (Generics) 185
3.7.1 Einführung 185
3.7.2 Generics und Auswirkungen der Type Erasure 190
3.8 Weiterführende Literatur 198

4 Java-Grundlagen 199
4.1 Die Klasse `Object` 199
4.1.1 Die Methode `toString()` 201
4.1.2 Die Methode `equals()` 205
4.2 Primitive Typen und Wrapper-Klassen 217

4.2.1 Grundlagen ... 218
4.2.2 Konvertierung von Werten ... 225
4.2.3 Wissenswertes zu Auto-Boxing und Auto-Unboxing ... 228
4.2.4 Ausgabe und Verarbeitung von Zahlen ... 231
4.3 Stringverarbeitung ... 234
4.3.1 Die Klasse `String` ... 235
4.3.2 Die Klassen `StringBuffer` und `StringBuilder` ... 239
4.3.3 Ausgaben mit `format()` und `printf()` ... 243
4.3.4 Die Methode `split()` und reguläre Ausdrücke ... 244
4.4 Datumsverarbeitung ... 249
4.4.1 Fallstricke der Datums-APIs ... 249
4.4.2 Das `Date`-API ... 251
4.4.3 Das `Calendar`-API ... 253
4.5 Varianten innerer Klassen ... 255
4.6 Ein- und Ausgabe (I/O) ... 259
4.6.1 Dateibehandlung und die Klasse `File` ... 259
4.6.2 Ein- und Ausgabestreams im Überblick ... 264
4.6.3 Zeichencodierungen bei der Ein- und Ausgabe ... 267
4.6.4 Speichern und Laden von Daten und Objekten ... 269
4.6.5 Dateiverarbeitung in JDK 7 ... 277
4.6.6 Erweiterungen im NIO und der Klasse `Files` in JDK 8 ... 279
4.7 Fehlerbehandlung ... 281
4.7.1 Einstieg in die Fehlerbehandlung ... 281
4.7.2 Checked Exceptions und Unchecked Exceptions ... 287
4.7.3 Exception Handling und Ressourcenfreigabe ... 288
4.7.4 Besonderheiten beim Exception Handling mit JDK 7 ... 294
4.7.5 Assertions ... 298
4.8 Weiterführende Literatur ... 301

5 Lambdas, Methodenreferenzen und Defaultmethoden ... 303
5.1 Einstieg in Lambdas ... 303
5.1.1 Syntax von Lambdas ... 303
5.1.2 Functional Interfaces und SAM-Typen ... 304
5.1.3 Exceptions in Lambdas ... 308
5.2 Syntaxerweiterungen in Interfaces ... 312
5.2.1 Defaultmethoden ... 313
5.2.2 Statische Methoden in Interfaces ... 315
5.3 Methodenreferenzen ... 317
5.4 Externe vs. interne Iteration ... 318
5.5 Wichtige Functional Interfaces für Collections ... 319
5.5.1 Das Interface `Predicate<T>` ... 319
5.5.2 Das Interface `UnaryOperator<T>` ... 321
5.5.3 Fazit ... 322

Michael Inden, Der Weg zum Java-Profi, dpunkt.verlag, ISBN 978-3-86490-483-7

II Bausteine stabiler Java-Applikationen 323

6 Das Collections-Framework ... 325
6.1 Datenstrukturen und Containerklasse ... 325
6.1.1 Wahl einer geeigneten Datenstruktur ... 326
6.1.2 Arrays ... 328
6.1.3 Das Interface `Collection` ... 331
6.1.4 Das Interface `Iterator` ... 333
6.1.5 Listen und das Interface `List` ... 336
6.1.6 Mengen und das Interface `Set` ... 344
6.1.7 Grundlagen von hashbasierten Containern ... 349
6.1.8 Grundlagen automatisch sortierender Container ... 358
6.1.9 Die Methoden `equals()`, `hashCode()` und `compareTo()` im Zusammenspiel ... 366
6.1.10 Schlüssel-Wert-Abbildungen und das Interface `Map` ... 368
6.1.11 Erweiterungen am Beispiel der Klasse `HashMap` ... 376
6.1.12 Erweiterungen im Interface `Map` mit JDK 8 ... 380
6.1.13 Entscheidungshilfe zur Wahl von Datenstrukturen ... 384
6.2 Suchen und Sortieren ... 385
6.2.1 Suchen ... 385
6.2.2 Sortieren von Arrays und Listen ... 389
6.2.3 Sortieren mit Komparatoren ... 391
6.2.4 Erweiterungen im Interface `Comparator` mit JDK 8 ... 393
6.3 Utility-Klassen und Hilfsmethoden ... 398
6.3.1 Nützliche Hilfsmethoden ... 398
6.3.2 Dekorierer `synchronized`, `unmodifiable` und `checked` 401
6.3.3 Vordefinierte Algorithmen in der Klasse `Collections` .. 406
6.4 Containerklassen: Generics und Varianz ... 408
6.5 Die Klasse `Optional` ... 422
6.5.1 Grundlagen zur Klasse `Optional` ... 422
6.5.2 Weiterführendes Beispiel und Diskussion ... 425
6.5.3 Verkettete Methodenaufrufe ... 428
6.6 Fallstricke im Collections-Framework ... 429
6.6.1 Wissenswertes zu Arrays ... 429
6.6.2 Wissenswertes zu `Stack`, `Queue` und `Deque` ... 433
6.7 Weiterführende Literatur ... 436

7 Bulk Operations on Collections ... 439
7.1 Streams ... 439
7.1.1 Streams erzeugen – Create Operations ... 440
7.1.2 Intermediate und Terminal Operations im Überblick ... 442
7.1.3 Zustandslose Intermediate Operations ... 444
7.1.4 Zustandsbehaftete Intermediate Operations ... 451

7.1.5 Terminal Operations 452
7.1.6 Wissenswertes zur Parallelverarbeitung 460
7.2 Filter-Map-Reduce 466
7.2.1 Herkömmliche Realisierung 466
7.2.2 Filter-Map-Reduce mit JDK 8 468
7.3 Praxisbeispiele 471
7.3.1 Aufbereiten von Gruppierungen und Histogrammen 471
7.3.2 Maps nach Wert sortieren 472
7.3.3 Verarbeitung von ZIP-Dateien 476

8 Applikationsbausteine 479
8.1 Einsatz von Bibliotheken 480
8.2 Google Guava im Kurzüberblick 483
8.2.1 String-Aktionen 485
8.2.2 Stringkonkatenation und -extraktion 486
8.2.3 Erweiterungen für Collections 490
8.2.4 Weitere Utility-Funktionalitäten 494
8.3 Wertebereichs- und Parameterprüfungen 500
8.3.1 Prüfung einfacher Wertebereiche und Wertemengen 501
8.3.2 Prüfung komplexerer Wertebereiche 503
8.4 Logging-Frameworks 507
8.4.1 Apache log4j2 507
8.4.2 Tipps und Tricks zum Einsatz von Logging mit log4j2 512
8.5 Konfigurationsparameter und -dateien 516
8.5.1 Einlesen von Kommandozeilenparametern 516
8.5.2 Verarbeitung von Properties 524
8.5.3 Die Klasse `Preferences` 531
8.5.4 Weitere Möglichkeiten zur Konfigurationsverwaltung 533

9 Multithreading 539
9.1 Threads und Runnables 541
9.1.1 Definition der auszuführenden Aufgabe 541
9.1.2 Start, Ausführung und Ende von Threads 542
9.1.3 Lebenszyklus von Threads und Thread-Zustände 546
9.1.4 Unterbrechungswünsche durch Aufruf von `interrupt()` 549
9.2 Zusammenarbeit von Threads 552
9.2.1 Konkurrierende Datenzugriffe 552
9.2.2 Locks, Monitore und kritische Bereiche 553
9.2.3 Deadlocks und Starvation 560
9.2.4 Kritische Bereiche und das Interface `Lock` 562

Michael Inden, Der Weg zum Java-Profi, dpunkt.verlag, ISBN 978-3-86490-483-7

9.3 Kommunikation von Threads 564
9.3.1 Kommunikation mit Synchronisation 564
9.3.2 Kommunikation über die Methoden `wait()`, `notify()` und `notifyAll()` 567
9.3.3 Abstimmung von Threads 574
9.3.4 Unerwartete `IllegalMonitorStateException`s 577
9.4 Das Java-Memory-Modell 578
9.4.1 Sichtbarkeit 579
9.4.2 Atomarität 579
9.4.3 Reorderings 581
9.5 Besonderheiten bei Threads 585
9.5.1 Verschiedene Arten von Threads 585
9.5.2 Exceptions in Threads 586
9.5.3 Sicheres Beenden von Threads 587
9.5.4 Zeitgesteuerte Ausführung 591
9.6 Die Concurrency Utilities 595
9.6.1 Concurrent Collections 596
9.6.2 Das Executor-Framework 602
9.6.3 Das Fork-Join-Framework 613
9.6.4 Erweiterungen im Bereich Concurrency mit JDK 8 615
9.7 Weiterführende Literatur 619

10 Fortgeschrittene Java-Themen 621
10.1 Crashkurs Reflection 621
10.1.1 Grundlagen 623
10.1.2 Zugriff auf Methoden und Attribute 626
10.1.3 Spezialfälle 631
10.1.4 Type Erasure und Typinformationen bei Generics 634
10.2 Annotations 636
10.2.1 Einführung in Annotations 637
10.2.2 Standard-Annotations des JDKs 638
10.2.3 Definition eigener Annotations 640
10.2.4 Annotation zur Laufzeit auslesen 643
10.3 Serialisierung 644
10.3.1 Grundlagen der Serialisierung 645
10.3.2 Die Serialisierung anpassen 650
10.3.3 Versionsverwaltung der Serialisierung 653
10.3.4 Optimierung der Serialisierung 657
10.4 Garbage Collection 662
10.4.1 Grundlagen zur Garbage Collection 662
10.4.2 Herkömmliche Algorithmen zur Garbage Collection 666
10.4.3 Einflussfaktoren auf die Garbage Collection 668
10.4.4 Der Garbage Collector »G1« 670

10.4.5 Memory Leaks: Gibt es die auch in Java?! ... 670
10.5 Dynamic Proxies ... 673
10.5.1 Statischer Proxy ... 675
10.5.2 Dynamischer Proxy ... 677
10.6 »Nashorn« – die JavaScript-Engine ... 682
10.7 Weiterführende Literatur ... 685

11 Datumsverarbeitung seit JDK 8 ... 687
11.1 Überblick über die neu eingeführten Typen ... 687
11.1.1 Neue Aufzählungen, Klassen und Interfaces ... 688
11.1.2 Die Aufzählungen `DayOfWeek` und `Month` ... 690
11.1.3 Die Klassen `MonthDay`, `YearMonth` und `Year` ... 690
11.1.4 Die Klasse `Instant` ... 691
11.1.5 Die Klasse `Duration` ... 692
11.1.6 Die Aufzählung `ChronoUnit` ... 695
11.1.7 Die Klassen `LocalDate`, `LocalTime` und `LocalDateTime` ... 696
11.1.8 Die Klasse `Period` ... 698
11.1.9 Die Klasse `ZonedDateTime` ... 699
11.1.10 Zeitzonen und die Klassen `ZoneId` und `ZoneOffset` ... 700
11.1.11 Die Klasse `Clock` ... 702
11.1.12 Formatierung und Parsing ... 704
11.2 Datumsarithmetik ... 705
11.3 Interoperabilität mit Legacy-Code ... 709

12 GUIs mit JavaFX ... 711
12.1 Einführung – JavaFX im Überblick ... 711
12.1.1 Grundsätzliche Konzepte ... 711
12.1.2 Layoutmanagement ... 715
12.2 Deklarativer Aufbau des GUIs ... 725
12.2.1 Deklarative Beschreibung von GUIs ... 725
12.2.2 Hello-World-Beispiel mit FXML ... 725
12.2.3 Diskussion: Design und Funktionalität strikt trennen ... 728
12.3 Rich-Client Experience ... 730
12.3.1 Gestaltung mit CSS ... 730
12.3.2 Effekte ... 736
12.3.3 Animationen ... 738
12.3.4 Zeichnen in JavaFX-Komponenten ... 740
12.4 Properties, Bindings und Observable Collections ... 743
12.4.1 Properties ... 743
12.4.2 Bindings ... 744
12.4.3 Observable Collections ... 748
12.4.4 Dynamisches Filtern von `ObservableList` ... 751

Michael Inden, Der Weg zum Java-Profi, dpunkt.verlag, ISBN 978-3-86490-483-7

12.5 Wichtige Bedienelemente ... 754
12.5.1 Dialoge ... 754
12.5.2 Formatierte Eingabe in TextFields ... 756
12.5.3 Die Bedienelemente ComboBox und ListView ... 758
12.5.4 Tabellen und das Bedienelement TableView ... 762
12.5.5 Das Bedienelement TreeTableView ... 764
12.5.6 Menüs ... 769
12.6 Multithreading in JavaFX ... 770
12.6.1 Das Interface Worker ... 770
12.6.2 Die Klasse Task<V> ... 771
12.7 Von Swing zu JavaFX ... 774
12.7.1 JavaFX in Swing einbinden ... 774
12.7.2 Swing in JavaFX einbinden ... 776
12.8 Weiterführende Literatur ... 778

13 Basiswissen Internationalisierung ... 779
13.1 Internationalisierung im Überblick ... 779
13.1.1 Grundlagen und Normen ... 780
13.1.2 Die Klasse Locale ... 781
13.1.3 Die Klasse PropertyResourceBundle ... 785
13.1.4 Formatierte Ein- und Ausgabe ... 788
13.1.5 Zahlen und die Klasse NumberFormat ... 789
13.1.6 Datumswerte und die Klasse DateFormat ... 792
13.1.7 Textmeldungen und die Klasse MessageFormat ... 797
13.1.8 Stringvergleiche mit der Klasse Collator ... 799
13.2 Programmbausteine zur Internationalisierung ... 804
13.2.1 Unterstützung mehrerer Datumsformate ... 805
13.2.2 Nutzung mehrerer Sprachdateien ... 810

III Neuerungen in Java 9 821

14 Ergänzungen in Java 9 ... 823
14.1 Syntaxerweiterungen ... 823
14.1.1 Anonyme innere Klassen und der Diamond Operator ... 823
14.1.2 Erweiterung der @Deprecated-Annotation ... 823
14.1.3 Private Methoden in Interfaces ... 824
14.2 Neues und Änderungen im JDK ... 826
14.2.1 Das neue Process-API ... 826
14.2.2 Neuerungen im Stream-API ... 830
14.2.3 Erweiterungen rund um die Klasse Optional ... 833
14.2.4 Erweiterungen in der Klasse InputStream ... 839
14.2.5 Erweiterungen in der Klasse Objects ... 840

14.2.6 Erweiterungen in der Klasse `CompletableFuture` 841
14.2.7 Collection-Factory-Methoden 843
14.3 Änderungen in der JVM 845
14.3.1 Garbage Collection 845
14.3.2 Browser-Plugin ist deprecated 847
14.3.3 Änderung des Versionsschemas 847
14.3.4 HTML5 Javadoc 848
14.3.5 Java + REPL => `jshell` 849
14.4 Fazit 851

15 Modularisierung mit Project Jigsaw 853
15.1 Grundlagen 854
15.1.1 Begrifflichkeiten 854
15.1.2 Ziele von Project Jigsaw 855
15.2 Modularisierung im Überblick 856
15.2.1 Grundlagen zu Project Jigsaw 856
15.2.2 Beispiel mit zwei Modulen 863
15.2.3 Packaging 872
15.2.4 Linking 874
15.2.5 Abhängigkeiten und Modulgraphen 877
15.2.6 Module des JDKs einbinden 879
15.2.7 Arten von Modulen 886
15.3 Sichtbarkeiten und Zugriffsschutz 887
15.3.1 Sichtbarkeiten 887
15.3.2 Zugriffsschutz und Reflection 889
15.4 Kompatibilität und Migration 891
15.4.1 Kompatibilitätsmodus 892
15.4.2 Migrationsszenarien 894
15.4.3 Fallstrick bei der Bottom-up-Migration 898
15.4.4 Beispiel: Migration mit Automatic Modules 900
15.4.5 Beispiel: Automatic und Unnamed Module 902
15.4.6 Mögliche Schwierigkeiten bei Migrationen 904
15.4.7 Fazit 905
15.5 Zusammenfassung 905

IV Fallstricke und Lösungen im Praxisalltag 907

16 Bad Smells 909
16.1 Programmdesign 911
16.1.1 Bad Smell: Verwenden von Magic Numbers 911
16.1.2 Bad Smell: Konstanten in Interfaces definieren 912

Michael Inden, Der Weg zum Java-Profi, dpunkt.verlag, ISBN 978-3-86490-483-7

16.1.3 Bad Smell: Zusammengehörende Konstanten nicht als Typ definiert 914
16.1.4 Bad Smell: Programmcode im Logging-Code 916
16.1.5 Bad Smell: Dominanter Logging-Code 917
16.1.6 Bad Smell: Unvollständige Betrachtung aller Alternativen . 919
16.1.7 Bad Smell: Unvollständige Änderungen nach Copy-Paste 920
16.1.8 Bad Smell: Casts auf unbekannte Subtypen 922
16.1.9 Bad Smell: Pre-/Post-Increment in komplexeren Statements 923
16.1.10 Bad Smell: Keine Klammern um Blöcke 925
16.1.11 Bad Smell: Mehrere aufeinanderfolgende Parameter gleichen Typs 927
16.1.12 Bad Smell: Grundloser Einsatz von Reflection 928
16.1.13 Bad Smell: `System.exit()` mitten im Programm 930
16.1.14 Bad Smell: Variablendeklaration nicht im kleinstmöglichen Sichtbarkeitsbereich 931
16.2 Klassendesign 933
16.2.1 Bad Smell: Unnötigerweise veränderliche Attribute 933
16.2.2 Bad Smell: Herausgabe von `this` im Konstruktor 935
16.2.3 Bad Smell: Aufruf abstrakter Methoden im Konstruktor . . . 937
16.2.4 Bad Smell: Referenzierung von Subklassen in Basisklassen 941
16.2.5 Bad Smell: Mix abstrakter und konkreter Basisklassen . . . 943
16.2.6 Bad Smell: Öffentlicher Defaultkonstruktor lediglich zum Zugriff auf Hilfsmethoden 945
16.3 Fehlerbehandlung und Exception Handling 947
16.3.1 Bad Smell: Unbehandelte Exception 947
16.3.2 Bad Smell: Unpassender Exception-Typ 948
16.3.3 Bad Smell: Exceptions zur Steuerung des Kontrollflusses 950
16.3.4 Bad Smell: Fangen der allgemeinsten Exception 951
16.3.5 Bad Smell: Rückgabe von `null` statt Exception im Fehlerfall 953
16.3.6 Bad Smell: Unbedachte Rückgabe von `null` 954
16.3.7 Bad Smell: Sonderbehandlung von Randfällen 957
16.3.8 Bad Smell: Keine Gültigkeitsprüfung von Eingabeparametern 958
16.3.9 Bad Smell: Fehlerhafte Fehlerbehandlung 960
16.3.10 Bad Smell: I/O ohne `finally` oder ARM 962
16.3.11 Bad Smell: Resource Leaks durch Exceptions im Konstruktor 964
16.4 Häufige Fallstricke 968
16.5 Weiterführende Literatur 983

17 Refactorings ... 985
17.1 Refactorings am Beispiel ... 986
17.2 Das Standardvorgehen ... 994
17.3 Kombination von Basis-Refactorings ... 997
17.3.1 Refactoring-Beispiel: Ausgangslage und Ziel ... 997
17.3.2 Auflösen der Abhängigkeiten ... 999
17.3.3 Vereinfachungen ... 1006
17.3.4 Verlagern von Funktionalität ... 1010
17.4 Der Refactoring-Katalog ... 1011
17.4.1 Reduziere die Sichtbarkeit von Attributen ... 1011
17.4.2 Minimiere veränderliche Attribute ... 1014
17.4.3 Reduziere die Sichtbarkeit von Methoden ... 1018
17.4.4 Ersetze Mutator- durch Business-Methode ... 1020
17.4.5 Minimiere Zustandsänderungen ... 1021
17.4.6 Führe ein Interface ein ... 1021
17.4.7 Spalte ein Interface auf ... 1022
17.4.8 Führe ein Read-only-Interface ein ... 1023
17.4.9 Führe ein Read-Write-Interface ein ... 1023
17.4.10 Lagere Funktionalität in Hilfsmethoden aus ... 1024
17.4.11 Trenne Informationsbeschaffung und -verarbeitung ... 1026
17.4.12 Wandle Konstantensammlung in `enum` um ... 1033
17.4.13 Entferne Exceptions zur Steuerung des Kontrollflusses ... 1036
17.4.14 Wandle in Utility-Klasse mit statischen Hilfsmethoden um 1038
17.4.15 Löse `if-else` / `instanceof` durch Polymorphie auf ... 1042
17.5 Defensives Programmieren ... 1045
17.5.1 Führe eine Zustandsprüfung ein ... 1045
17.5.2 Überprüfe Eingabeparameter ... 1046
17.6 Falllstricke bei Refactorings ... 1051
17.7 Weiterführende Literatur ... 1053

18 Entwurfsmuster ... 1055
18.1 Erzeugungsmuster ... 1058
18.1.1 Erzeugungsmethode ... 1058
18.1.2 Fabrikmethode (Factory method) ... 1061
18.1.3 Erbauer (Builder) ... 1064
18.1.4 Singleton ... 1067
18.1.5 Prototyp (Prototype) ... 1072
18.2 Strukturmuster ... 1076
18.2.1 Fassade (Façade) ... 1076
18.2.2 Adapter ... 1078
18.2.3 Dekorierer (Decorator) ... 1080
18.2.4 Kompositum (Composite) ... 1084

Michael Inden, Der Weg zum Java-Profi, dpunkt.verlag, ISBN 978-3-86490-483-7

18.3 Verhaltensmuster ... 1088
18.3.1 Iterator ... 1088
18.3.2 Null-Objekt (Null Object) ... 1090
18.3.3 Schablonenmethode (Template method) ... 1093
18.3.4 Strategie (Strategy) ... 1097
18.3.5 Befehl (Command) ... 1109
18.3.6 Proxy ... 1116
18.3.7 Beobachter (Observer) ... 1118
18.3.8 MVC-Architektur ... 1127
18.4 Weiterführende Literatur ... 1129

V Qualitätssicherungsmaßnahmen 1131

19 Programmierstil und Coding Conventions ... 1133
19.1 Grundregeln eines guten Programmierstils ... 1133
19.1.1 Keep It Human-Readable ... 1134
19.1.2 Keep It Simple And Short (KISS) ... 1134
19.1.3 Keep It Natural ... 1134
19.1.4 Keep It Clean ... 1135
19.2 Die Psychologie beim Sourcecode-Layout ... 1135
19.2.1 Gesetz der Ähnlichkeit ... 1135
19.2.2 Gesetz der Nähe ... 1137
19.3 Coding Conventions ... 1138
19.3.1 Grundlegende Namens- und Formatierungsregeln ... 1139
19.3.2 Namensgebung ... 1142
19.3.3 Dokumentation ... 1145
19.3.4 Programmdesign ... 1147
19.3.5 Klassendesign ... 1152
19.3.6 Parameterlisten ... 1155
19.3.7 Logik und Kontrollfluss ... 1157
19.4 Sourcecode-Prüfung mit Tools ... 1159
19.4.1 Metriken ... 1161
19.4.2 Sourcecode-Prüfung im Build-Prozess ... 1165

20 Unit Tests ... 1175
20.1 Testen im Überblick ... 1175
20.1.1 Was versteht man unter Testen? ... 1175
20.1.2 Testarten im Überblick ... 1176
20.1.3 Zuständigkeiten beim Testen ... 1179
20.1.4 Testen und Qualität ... 1181

20.2 Wissenswertes zu Testfällen 1185
20.2.1 Testfälle mit JUnit 4 definieren 1185
20.2.2 Problem der Kombinatorik 1193
20.3 Motivation für Unit Tests aus der Praxis 1197
20.4 JUnit Rules und parametrierte Tests 1206
20.4.1 JUnit Rules im Überblick 1206
20.4.2 Parametrierte Tests 1210
20.5 Fortgeschrittene Unit-Test-Techniken 1215
20.5.1 Stellvertreterobjekte / Test-Doubles 1218
20.5.2 Vorarbeiten für das Testen mit Stubs und Mocks 1222
20.5.3 Die Technik EXTRACT AND OVERRIDE 1224
20.5.4 Einstieg in das Testen mit Mocks und Mockito 1231
20.5.5 Abhängigkeiten mit Mockito auflösen 1239
20.5.6 Unit Tests von privaten Methoden 1242
20.6 Unit Tests mit Threads und Timing 1244
20.6.1 Funktionale Erweiterung: Aggregation und Versand 1244
20.6.2 Test der Aggregation und des Versands 1246
20.6.3 Test des nebenläufigen Versands 1249
20.7 Test Smells 1251
20.8 Nützliche Tools für Unit Tests 1256
20.8.1 Hamcrest 1256
20.8.2 MoreUnit 1262
20.8.3 Infinitest 1262
20.8.4 JaCoCo 1263
20.8.5 EclEmma 1266
20.9 Ausblick auf JUnit 5 1268
20.9.1 Einführendes Beispiel 1268
20.9.2 Wichtige Neuerungen in JUnit 5 im Überblick 1269
20.10 Weiterführende Literatur 1273

21 Codereviews 1275
21.1 Definition 1275
21.2 Probleme und Tipps zur Durchführung 1277
21.3 Vorteile von Codereviews 1279
21.4 Codereview-Checkliste 1282

22 Optimierungen 1283
22.1 Grundlagen 1284
22.1.1 Optimierungsebenen und Einflussfaktoren 1285
22.1.2 Optimierungstechniken 1286
22.1.3 CPU-bound-Optimierungsebenen am Beispiel 1288
22.1.4 Messungen – Erkennen kritischer Bereiche 1292
22.1.5 Abschätzungen mit der O-Notation 1299

Michael Inden, Der Weg zum Java-Profi, dpunkt.verlag, ISBN 978-3-86490-483-7

22.2 Einsatz geeigneter Datenstrukturen ... 1302
22.2.1 Einfluss von Arrays und Listen ... 1303
22.2.2 Optimierungen für `Set` und `Map` ... 1307
22.2.3 Design eines Zugriffsinterface ... 1309
22.3 Lazy Initialization ... 1312
22.3.1 Lazy Initialization am Beispiel ... 1312
22.3.2 Konsequenzen des Einsatzes der Lazy Initialization ... 1315
22.3.3 Lazy Initialization mithilfe des PROXY-Musters ... 1317
22.4 Optimierungen am Beispiel ... 1320
22.5 I/O-bound-Optimierungen ... 1327
22.5.1 Technik – Wahl passender Strategien ... 1327
22.5.2 Technik – Caching und Pooling ... 1331
22.5.3 Technik – Vermeidung unnötiger Aktionen ... 1331
22.6 Memory-bound-Optimierungen ... 1334
22.6.1 Technik – Wahl passender Strategien ... 1334
22.6.2 Technik – Caching und Pooling ... 1337
22.6.3 Optimierungen der Stringverarbeitung ... 1343
22.6.4 Technik – Vermeidung unnötiger Aktionen ... 1345
22.7 CPU-bound-Optimierungen ... 1348
22.7.1 Technik – Wahl passender Strategien ... 1348
22.7.2 Technik – Caching und Pooling ... 1350
22.7.3 Technik – Vermeidung unnötiger Aktionen ... 1351
22.8 Weiterführende Literatur ... 1354

23 Schlussgedanken ... 1355

VI Anhang 1357

A Grundlagen zur Java Virtual Machine ... 1359
A.1 Wissenswertes rund um die Java Virtual Machine ... 1359
A.1.1 Ausführung eines Java-Programms ... 1359
A.1.2 Sicherheit und Speicherverwaltung ... 1360
A.1.3 Sicherheit und Classloading ... 1361

Literaturverzeichnis ... 1363

Index ... 1367