Inhalt

ÜBERSICHT **XI**

Fahrplan und Rapid-Coding **xi**

Das erste Anliegen - Verständliche Texte **xiv**

Konventionen und Verabredungen **xvi**

DAS SOFTWARE-UNIVERSUM **1**

1. Das Buch der unbrauchbaren Softwareteile **2**

2. Ein Anfang – aber nicht mehr **5**

EINE KLEINE JAVA-RUNDREISE – TEIL 1 **7**

1. Intro **9**

2. 90 Prozent - Ein Plan und ein Stück Code - Was ein Entwickler wissen muss **9**

3. Methoden **13**
3.1. Die Signatur von Methoden 13
3.2. Werden Argumente und Returnwerte kopiert? 15
3.3. Returntyp und return-Statement 17
3.4. Aufteilung von Code in Methoden 18
3.5. Variabel lange Argumentlisten 21
3.6. Argumentprüfung 22

4. Lokale Variable, Blöcke, Scope und final **23**

5. Die Grundregel für zusammengesetzte Ausdrücke **27**

6. Weitere elementare Regeln und Begriffe **29**

7. Zuweisung **33**

8. Kombinierte Zuweisungen **34**

9. Inkrement und Dekrement **35**

10. Stringverknüpfung (+) **36**

11. Bedingung (if/else) **37**

12. Verzweigung (switch) **39**

13. Der new-Operator **42**

14. Gleichheits- und Identitäts-Operator 43

15. Logische Operatoren, Vergleiche, Verknüpfung von Bedingungen 44

16. Der Conditional-Operator 46

17. Zusammenfassung der Operatoren 47
17.1. Auswertungsreihenfolge 48
17.2. Auflistung nach Priorität 49
17.3. Bit-Operatoren, Bit-Komplement-Operator und Bit-Shift-Operatoren 51

18. Iterationen 52
18.1. Übersicht 52
18.2. Bedingungen und Zähler 53
18.3. Ablaufschema 55
18.4. break und continue 56
18.5. Iteration und robuste Programmierung 57
18.6. for/in 58

19. Elementare Programmiertechnik anhand eines Beispiels 60
19.1. Die Aufgabe 61
19.2. Start ohne Skrupel 62
19.3. Übersichtlicher Code 64
19.4. Zerlegung in Methoden 66
19.5. Welche Anforderungen bestehen und wer überprüft sie? 68
19.6. Aufsammeln der Zahlen 69
19.7. Abspaltung der Benutzereingabe 69

20. Objekte * 71
20.1. Bausteine 71
20.2. Methodenaufrufe 74
20.3. Anatomie von Klassen und Objekten 75
20.4. Statische Elemente 78
20.5. Referenzen, Nirvana und this 79

21. Typen und Interfaces * 82
21.1. Ein leichtgewichtiges Tool 82
21.2. Ableitung und Vererbung 84
21.3. Das Verbergen und das Abgreifen von Information 86
21.4. Upcast und Downcast 87
21.5. Typen 89
21.6. Interfaces erweitern Interfaces 90

22. Allgemeine Vererbung * 91

23. Packages, Import und private Typen 94
23.1. Packages 94
23.2. Nicht-public Typen 95
23.3. Package-Sichtbarkeit 95
23.4. Import-Statements 96
23.5. Static Imports 97

24. Module * 98

25. Exceptions * 99

26. Elementare Typen und ihre Verpackung in Objekten – Wrappers 101

27. Zahlen und Zufallszahlen 103
27.1. Literale und Arithmetik 103
27.2. Große Zahlen 108
27.3. Der Zahlengenerator Random 111

28. Konstanten, Konstanz und Unveränderbarkeit 113

29. Enums * 115

30. Threads * 117

31. Datum und Uhrzeit 120
31.1. Zeitstempel und Zeitdauer 121
31.2. Zusammensetzen, Beschränken und Kopieren 123
31.3. Tageszeit - LocalTime 124
31.4. Datum – LocalDate 126
31.5. Zeitzonen – ZonedDateTime 128
31.6. Alt - Date und Calendar 129
31.7. Formatieren von Datums- und Zeitangaben 131
31.8. Parsen 133

32. Performancemessung 133

33. Strings * 136

34. Arrays 137
34.1. Basics 138
34.2. Initialisierung, elementare Elementtypen und Referenzelemente 140
34.3. Elementtyp, Vergleichbarkeit und das Tool Arrays 143
34.4. Mehrdimensionale Arrays 144

35. Collections * 146

36. Generische Klassen * 148

37. Generische Methoden * 152

38. Maps * 154

39. Essentials 157

40. Guidelines 174

RAPID-CODING EINSCHUB 1 177

1. Was ist Rapid-Coding? 178

2. Der Zeitfaktor 178

3. Die Aufgabe 178

4. Zerlegung und Verteilung 184

5. Alt-Bekanntes 186
5.1. Iterationen 187
5.2. Argumentprüfungen 188
5.3. Exceptions werfen 188
5.4. Konstanten vereinbaren 188
5.5. Es gibt sehr viel mehr 189

6. Ein kleiner Schritt 189

7. Ein großer Schritt 191

OBJEKTE 195

1. Einige kurze Fakten zu Objekten, die das Verständnis erleichtern 196
1.1. Man kann Java auch ohne Objekte nutzen 196
1.2. Man kann Java mit Objekten und ohne Objektorientierung betreiben 196
1.3. Instanzen und Objekte sind dasselbe 196
1.4. Wir konstruieren keine Objekte, sondern Klassen 197
1.5. Auch mit Objekten kann man Mist bauen 197
1.6. Klassen sind Typen 197
1.7. ‚Abstrakt' bedeutet ‚nicht implementiert' 198

2. Fange einfach irgendwo an 198
2.1. Straight-forward 198
2.2. Einsatz der üblichen Tools 201
2.3. Behandlung der Eingabefehler 202
2.4. Kommentare 204
2.5. Der gesamte Code 205
2.6. Das prozedurale Denken 208
2.7. Zäher Code 209

3. Die Klarheit der Objekte 209
3.1. Spielerischer Einstieg 209
3.2. Details der Komponenten 218
3.3. Resümee 220
3.4. Die Lottostatistik 221

4. Die Basics 222
4.1. Der Blick auf das Ganze 222
4.2. Instanzdaten und Kapselung 226
4.3. Wege, um die Kapselung zu brechen 229
4.4. Klassen-Elemente 232
4.5. Interface-Elemente 236
4.6. Konstruktoren 241
4.7. Object 251
4.8. Probleme mit equals() und hashCode() 256

5. Implementierung der Lotto-Applikation 262
5.1. LottoTicket 262

5.2. Win 270
5.3. LottoDrawing 271
5.4. RealLottery 273
5.5. Zustandsprogrammierung bei RealLottery 280
5.6. Objekte als robuste Maschinen 282

6. Statische Elemente 285
6.1. Syntax und Übersicht 285
6.2. Baustein ohne Instanzen 287
6.3. Singleton 289
6.4. Verwaltung von Instanzen 292

7. Vererbung 296
7.1. Cube 296
7.2. Terminologie 299
7.3. Konstruktoren 301
7.4. Schwierigkeiten, die genaues Hinsehen erfordern 303
7.5. Eine Art von Superclass-Chaining 307
7.6. Den Typ einer Instanz kann man nicht verbergen 309
7.7. Overriding 310
7.8. Der Downcast 313
7.9. Konsequenzen von später Bindung 314
7.10. Aufbau der Instanzen 317
7.11. NationalLottery 317
7.12. Vererbung von Implementierung bricht abgeleitete Klassen 322
7.13. Ist ein Math ein Cube? 323

8. Erzwingen und Verhindern von Overriding 326
8.1. Abstrakte Methoden und abstrakte Klassen 326
8.2. final Methoden und Klassen 328

9. Cloneable 328

10. Comparable 331
10.1. Vergleichbarkeit und natürliche Ordnung 331
10.3. Comparatoren sortieren Elemente mit und ohne natürliche Ordnung 332
10.4. Comparatoren in der Lotto-Applikation 333
10.5. Nur einfache Anforderungen an Listener 335

11. Essentials 338

12. Guidelines 350

RAPID-CODING EINSCHUB 2 353

1. Das Beispiel 354

2. Das einfache Schema der Klassen 356

3. Konzept, Typ und Erfassbarkeit 356

4. Trennung von Typ und Implementierung 360

MODULE **363**

1. Arbeiten mit und ohne Module **364**

2. Gespräch über Module **364**

3. Benannte Module und Kapselung **366**

4. Freigaben **367**

5. Module und Objekte **370**

6. Im Kreis **372**

7. Essentials **374**

ANONYME KLASSEN UND LAMBDAS **385**

1. Innere Typen **386**
1.1. Intro 386
1.2. Wechselseitiger Zugriff 388
1.3. Noch eine neue Syntax: new 391
1.4. Wie der Compiler den freien wechselseitigen Zugriff sieht 392
1.5. Übersicht zu inneren Typen 392

2. Syntax und Einsatz von anonymen Klassen **404**
2.1. Der definierende Typ 404
2.2. Drei oder vier Schritte zu einer Formel 406
2.3. Für anonyme Klassen können keine Konstruktoren geschrieben werden 408
2.4. Code für eine spätere Ausführung 410
2.5. Klasse oder Methode? 412
2.6. Fallbeispiel mit einer Defaultmethode 414
2.7. Adapter als simple Hilfsmittel 416

3. Essentials zu inneren Typen **418**

4. Lambdas **422**
4.1. Intro 422
4.2. Von den anonymen Klassen zu den Lambdas 423
4.3. Die Syntax der Lambdas 424
4.4. Lambda-Interfaces 427
4.5. Methodenreferenzen 437
4.6. Der nächste Schritt: Kombination von Lambdas 444

5. Intro zu Streams * **458**
5.1. Was ist ein Stream? 458
5.2. Eine eigenartige Konzeption 460
5.3. Was enthält ein Stream? 461
5.4. Als Beispiel - Die Arbeits- und Funktionsweise von reduce() 462

6. Essentials zu Lambdas **465**

7. Guidelines zu Lambdas **474**

EXCEPTIONS **477**

1. Wirf eine Exception **479**

2. Eine einfache Entscheidung **481**

3. Stack-Traces, Methodenlabyrinthe und Ariadne-Fäden **482**

4. Die methodenübergreifende Struktur von try-catch **485**

5. Der passende Exception-Handler **490**

6. finally **492**

7. Checked- und Unchecked-Exceptions **494**

8. Mehrfach-Handler (Multi-Catch) **496**

9. Wiederauswurf – Rethrow **499**

10. Smarter Compiler **502**

11. Ressourcen-Behandlung **505**
11.2. Aufwendige Ressourcen-Freigabe mit finally 506
11.3. Die Automatismen von try-with-resources 509

12. Suppressed Exceptions **510**

13. Selbst verfasste Exceptiontypen **513**

14. Exceptions und Threads **514**

15. UncaughtExceptionHandler **515**

16. Exceptions tatsächlich behandeln **516**
16.1. Die Wiederholung 516
16.2. Unterscheide Produktivzeit und Entwicklungszeit 516
16.3. Exceptions zur Entwicklungszeit verlässlich anzeigen 517
16.4. Die Unterscheidung von Checked- und Unchecked-Exceptions ist unnötig 518
16.5. Etwas suchen 518
16.6. Auf Bauteile verzichten 519
16.7. Jedes Bauteil braucht ein Fehlerkonzept 519
16.8. Bei allen Methoden Exceptions deklarieren und spezifizieren 520

17. Essentials **521**

18. Guidelines **527**

TEIL 2 DER JAVA-RUNDREISE – TOOLS, OHNE DIE ES NICHT GEHT **529**

1. Class und Reflection **530**

2. System-Properties 533

3. Logging 534
3.1. Der Standard-Logging-Framework 534
3.2. Austauschbare Implementierungen und Tag-basiertes Logging 547
3.3. Guidelines zum Logging 552

4. Intro zu Threads 553
4.1. Ein Ariadne-Faden 553
4.2. Dem Faden folgen 553
4.3. Parallele Fäden 553
4.4. Threads sind leichtgewichtige Prozesse 554
4.5. Objektorientierung und parallele Abläufe sind zwei verschiedene Welten 554
4.6. Ausführung in nur einem Thread 555
4.7. Ausführung in zwei Threads 556
4.8. Threads erzeugen und starten 558
4.9. Auf einen Thread warten - join() 560
4.10. Thread-Zustände 561
4.11. Blockierte Threads 562
4.12. Eine kurze Exkursion zum Wait-Set 564
4.13. Einen Thread beenden 565
4.14. Dämonen 567
4.15. Zeitscheiben und kooperatives Verhalten 567
4.16. Priorität und Id 568
4.17. Auflistung der Threads 569
4.18. Timer 570
4.20. Essentials zu Threads 572

5. Streams 575
5.1. Intro 575
5.2. Streams verwenden 578
5.3. Einzelne Zeichen und Arrays 582
5.4. Streams können blockieren 584
5.5. Eingangs-Streams 588
5.6. Übersicht der Byte-Streams 590
5.7. Übersicht der Character-Streams 594
5.8. Ausgangs-Streams 596
5.9. Typische Stream-Verwendungen 597
5.10. Daten-Streams 600
5.11. Essentials zu Streams 603

6. Files 607
6.1. Pfade 607
6.2. Die Klasse File 609
6.3. Absolute und relative Pfade 611
6.4. Path 612
6.5. Files lesen 614
6.6. Files schreiben 616
6.7. Ressource-Pfade 618
6.8. Ein kleiner Test zu den Ressource-Pfaden 621
6.9. Standardcode: Files lesen 622
6.10. Standardcode: Files schreiben 623
6.11. Directory-Iterator und list() 625
6.12. Mehr über Files: copy(), move(), find() und walk() 626

6.13. Files und das Visitor-Pattern 629
6.14. Übersicht der Files-Methoden 635
6.15. Essentials zu Files 638

ANHANG 643

1. Die erste Klasse 644

2. Java und der JDK 647
2.1. Sourcecode, Bytecode und Virtuelle Maschine 647
2.2. Java-Versionen 647
2.3. Download des JDK 650
2.4. Installation des JDK 650
2.5. Die Path-Umgebungsvariable 651
2.6. Einige Tools 652
2.7. Entwicklung ohne und mit Entwicklungsumgebung 652

3. Modulepath und Classpath 653
3.1. Eine Liste von Orten, um nach referenzierten Typen zu suchen 654
3.2. Modulepath und Modulesourcepath 655
3.3. Disassembler 655
3.4. Vermeintliche und tatsächliche Classpath-Probleme 656

4. Compiler 657
4.1. Einfache Aufrufe 657
4.2. Module 659

5. Archive 660
5.1. Übersicht 660
5.2. Modulare Jar-Files 662

6. Interpreter 663

7. Eclipse-Intro 666

8. Das Richtige tun 670

INDEX 673

Abbildungen

Abbildung 1: Java-Arrays und Indizes *140*

Abbildung 2: Einige Oberflächenelemente der Lotto-Applikation *199*

Abbildung 3: Die Idee des Drawing-Bausteins *210*

Abbildung 4: Leistung des Drawing-Bausteins *211*

Abbildung 5: Bausteine und ihr Zusammenwirken *213*

Abbildung 6: Der Typ Gambler *215*

Abbildung 7: Der Typ Person *216*

Abbildung 8: Ein ClassRoom für BirthDayPersons *259*

Abbildung 9: Der Basistyp steht per Konvention oben, der abgeleitete Typ unten *299*

Abbildung 10: Drei Example-Instanzen mit der Instanz ihrer Klasse *530*

Abbildung 11: Die Standard-Log-Handler *541*

Abbildung 12: Logger und Handler sind die Hauptakteure beim Logging *542*

Abbildung 13: Zwei Welten: Character-Rohre und Byte-Rohre *575*

Abbildung 14: Vier unabhängige Grundtypen: InputStream, OutputStream, Reader und Writer *576*

Abbildung 15: Die vier Stream-Grundtypen und zugeordnete Interfaces *577*

Abbildung 16: Filter - Streams werden als Rohre ineinander gesteckt *582*

Abbildung 17: Reader und InputStream *588*

Abbildung 18: Die Byte-Klassen des Stream-Frameworks *590*

Abbildung 19: ByteArrayInputStreams werden zu einer Sequenz von Streams verknüpft *591*

Abbildung 20: Die Character-Klassen des Stream-Frameworks *594*

Abbildung 21: Writer und OutputStream *597*

Abbildung 22: DataInputStream und DataOutputStream *601*