Inhalt

Vorwort 31

1 Java ist auch eine Sprache 47

1.1 Historischer Hintergrund 47

1.2 Warum Java gut ist – die zentralen Eigenschaften 49

1.2.1 Bytecode 50
1.2.2 Ausführung des Bytecodes durch eine virtuelle Maschine 50
1.2.3 Plattformunabhängigkeit 51
1.2.4 Java als Sprache, Laufzeitumgebung und Standardbibliothek 51
1.2.5 Objektorientierung in Java 52
1.2.6 Java ist verbreitet und bekannt 53
1.2.7 Java ist schnell – Optimierung und Just-in-time-Compilation 53
1.2.8 Das Java-Security-Modell 55
1.2.9 Zeiger und Referenzen 56
1.2.10 Bring den Müll raus, Garbage-Collector! 57
1.2.11 Ausnahmebehandlung 58
1.2.12 Angebot an Bibliotheken und Werkzeugen 58
1.2.13 Einfache Syntax der Programmiersprache Java 59
1.2.14 Java ist Open Source 61
1.2.15 Wofür sich Java weniger eignet 62
1.2.16 Java im Vergleich zu anderen Sprachen * 63
1.2.17 Java und das Web, Applets und JavaFX 64
1.2.18 Features, Enhancements (Erweiterungen) und ein JSR 67
1.2.19 Die Entwicklung von Java und seine Zukunftsaussichten 67

1.3 Java-Plattformen: Java SE, Java EE, Java ME, Java Card 69

1.3.1 Die Java SE-Plattform 69
1.3.2 Java ME: Java für die Kleinen 71
1.3.3 Java für die ganz, ganz Kleinen 71
1.3.4 Java für die Großen 72
1.3.5 Echtzeit-Java (Real-time Java) 72

1.4 Die Installation der Java Platform, Standard Edition (Java SE) 73

1.4.1 Die Java SE-Plattform 73
1.4.2 Download des JDK 74
1.4.3 Java SE unter Windows installieren 76

1.4.4 JDK unter Linux installieren ... 82
1.4.5 JDK unter macOS installieren ... 82

1.5 Das erste Programm compilieren und testen ... 84
1.5.1 Compilertest ... 84
1.5.2 Ein Quadratzahlen-Programm ... 85
1.5.3 Der Compilerlauf ... 86
1.5.4 Die Laufzeitumgebung ... 87
1.5.5 Häufige Compiler- und Interpreter-Probleme ... 87

1.6 Entwicklungsumgebungen im Allgemeinen ... 88
1.6.1 Die Entwicklungsumgebung Eclipse ... 88
1.6.2 IntelliJ IDEA ... 89
1.6.3 NetBeans ... 90
1.6.4 Ein Wort zu Microsoft, Java und zu J++, J# ... 90

1.7 Eclipse im Speziellen ... 91
1.7.1 Eclipse entpacken und starten ... 93
1.7.2 Das erste Projekt anlegen ... 94
1.7.3 Verzeichnisstruktur für Java-Projekte * ... 94
1.7.4 Eine Klasse hinzufügen ... 95
1.7.5 Übersetzen und ausführen ... 96
1.7.6 Projekt einfügen, Workspace für die Programme wechseln ... 97
1.7.7 Plugins für Eclipse ... 97

1.8 Zum Weiterlesen ... 98

2 Imperative Sprachkonzepte 99

2.1 Elemente der Programmiersprache Java ... 99
2.1.1 Token ... 99
2.1.2 Textkodierung durch Unicode-Zeichen ... 100
2.1.3 Bezeichner ... 100
2.1.4 Literale ... 103
2.1.5 Reservierte Schlüsselwörter ... 103
2.1.6 Zusammenfassung der lexikalischen Analyse ... 104
2.1.7 Kommentare ... 105

2.2 Von der Klasse zur Anweisung ... 107
2.2.1 Was sind Anweisungen? ... 107
2.2.2 Klassendeklaration ... 107
2.2.3 Die Reise beginnt am main(String[]) ... 108
2.2.4 Der erste Methodenaufruf: println(...) ... 109

2.2.5 Atomare Anweisungen und Anweisungssequenzen ... 110
2.2.6 Mehr zu print(...), println(...) und printf(...) für Bildschirmausgaben ... 111
2.2.7 Die API-Dokumentation ... 113
2.2.8 Ausdrücke ... 114
2.2.9 Ausdrucksanweisung ... 115
2.2.10 Erste Idee der Objektorientierung ... 115
2.2.11 Modifizierer ... 116
2.2.12 Gruppieren von Anweisungen mit Blöcken ... 117

2.3 Datentypen, Typisierung, Variablen und Zuweisungen ... 118
2.3.1 Primitive Datentypen im Überblick ... 120
2.3.2 Variablendeklarationen ... 123
2.3.3 Konsoleneingaben ... 126
2.3.4 Fließkommazahlen mit den Datentypen float und double ... 128
2.3.5 Ganzzahlige Datentypen ... 130
2.3.6 Wahrheitswerte ... 131
2.3.7 Unterstriche in Zahlen * ... 132
2.3.8 Alphanumerische Zeichen ... 133
2.3.9 Gute Namen, schlechte Namen ... 133
2.3.10 Initialisierung von lokalen Variablen ... 134

2.4 Ausdrücke, Operanden und Operatoren ... 135
2.4.1 Zuweisungsoperator ... 136
2.4.2 Arithmetische Operatoren ... 137
2.4.3 Unäres Minus und Plus ... 141
2.4.4 Zuweisung mit Operation ... 141
2.4.5 Präfix- oder Postfix-Inkrement und -Dekrement ... 142
2.4.6 Die relationalen Operatoren und die Gleichheitsoperatoren ... 145
2.4.7 Logische Operatoren: Nicht, Und, Oder, XOR ... 147
2.4.8 Kurzschluss-Operatoren ... 148
2.4.9 Der Rang der Operatoren in der Auswertungsreihenfolge ... 149
2.4.10 Die Typumwandlung (das Casting) ... 152
2.4.11 Überladenes Plus für Strings ... 157
2.4.12 Operator vermisst * ... 158

2.5 Bedingte Anweisungen oder Fallunterscheidungen ... 159
2.5.1 Verzweigung mit der if-Anweisung ... 159
2.5.2 Die Alternative mit einer if-else-Anweisung wählen ... 162
2.5.3 Der Bedingungsoperator ... 166
2.5.4 Die switch-Anweisung bietet die Alternative ... 168

2.6 Immer das Gleiche mit den Schleifen ... 174
2.6.1 Die while-Schleife ... 175
2.6.2 Die do-while-Schleife ... 176

2.6.3 Die for-Schleife ... 178
2.6.4 Schleifenbedingungen und Vergleiche mit == * ... 182
2.6.5 Schleifenabbruch mit break und zurück zum Test mit continue ... 184
2.6.6 break und continue mit Marken * ... 188

2.7 Methoden einer Klasse ... 191
2.7.1 Bestandteil einer Methode ... 192
2.7.2 Signatur-Beschreibung in der Java-API ... 193
2.7.3 Aufruf einer Methode ... 195
2.7.4 Methoden ohne Parameter deklarieren ... 196
2.7.5 Statische Methoden (Klassenmethoden) ... 197
2.7.6 Parameter, Argument und Wertübergabe ... 197
2.7.7 Methoden vorzeitig mit return beenden ... 199
2.7.8 Nicht erreichbarer Quellcode bei Methoden * ... 200
2.7.9 Methoden mit Rückgaben ... 201
2.7.10 Methoden überladen ... 206
2.7.11 Sichtbarkeit und Gültigkeitsbereich ... 208
2.7.12 Vorgegebener Wert für nicht aufgeführte Argumente * ... 209
2.7.13 Finale lokale Variablen ... 210
2.7.14 Rekursive Methoden * ... 211
2.7.15 Die Türme von Hanoi * ... 216

2.8 Zum Weiterlesen ... 218

3 Klassen und Objekte 219

3.1 Objektorientierte Programmierung (OOP) ... 219
3.1.1 Warum überhaupt OOP? ... 219
3.1.2 Denk ich an Java, denk ich an Wiederverwendbarkeit ... 220

3.2 Eigenschaften einer Klasse ... 221
3.2.1 Klassenarbeit mit Point ... 222

3.3 Natürlich modellieren mit der UML (Unified Modeling Language) * ... 222
3.3.1 Hintergrund und Geschichte der UML * ... 223
3.3.2 Wichtige Diagrammtypen der UML * ... 223
3.3.3 UML-Werkzeuge * ... 225

3.4 Neue Objekte erzeugen ... 226
3.4.1 Ein Exemplar einer Klasse mit dem Schlüsselwort new anlegen ... 226
3.4.2 Der Zusammenhang von new, Heap und Garbage-Collector ... 227
3.4.3 Deklarieren von Referenzvariablen ... 228
3.4.4 Jetzt mach mal 'nen Punkt: Zugriff auf Objektattribute und -methoden ... 229

3.4.5 Überblick über Point-Methoden ... 234
3.4.6 Konstruktoren nutzen ... 237

3.5 ZZZZZnake ... 238

3.6 Pakete schnüren, Imports und Kompilationseinheiten ... 241
3.6.1 Java-Pakete ... 241
3.6.2 Pakete der Standardbibliothek ... 241
3.6.3 Volle Qualifizierung und import-Deklaration ... 241
3.6.4 Mit import p1.p2.* alle Typen eines Pakets erreichen ... 243
3.6.5 Hierarchische Strukturen über Pakete ... 243
3.6.6 Die package-Deklaration ... 244
3.6.7 Unbenanntes Paket (default package) ... 245
3.6.8 Klassen mit gleichen Namen in unterschiedlichen Paketen * ... 246
3.6.9 Kompilationseinheit (Compilation Unit) ... 246
3.6.10 Statischer Import * ... 247

3.7 Mit Referenzen arbeiten, Identität und Gleichheit (Gleichwertigkeit) ... 248
3.7.1 null-Referenz und die Frage der Philosophie ... 248
3.7.2 Alles auf null? Referenzen testen ... 251
3.7.3 Zuweisungen bei Referenzen ... 252
3.7.4 Methoden mit Referenztypen als Parametern ... 253
3.7.5 Identität von Objekten ... 257
3.7.6 Gleichheit (Gleichwertigkeit) und die Methode equals(...) ... 258

3.8 Arrays ... 260
3.8.1 Grundbestandteile ... 261
3.8.2 Deklaration von Arrays ... 261
3.8.3 Arrays mit Inhalt ... 262
3.8.4 Die Länge eines Arrays über das Attribut length auslesen ... 262
3.8.5 Zugriff auf die Elemente über den Index ... 263
3.8.6 Array-Objekte mit new erzeugen ... 265
3.8.7 Typische Array-Fehler ... 266
3.8.8 Arrays als Methodenparameter ... 267
3.8.9 Vorinitialisierte Arrays ... 268
3.8.10 Die erweiterte for-Schleife ... 269
3.8.11 Arrays mit nichtprimitiven Elementen ... 271
3.8.12 Methode mit variabler Argumentanzahl (Vararg) ... 274
3.8.13 Mehrdimensionale Arrays * ... 277
3.8.14 Nichtrechteckige Arrays * ... 280
3.8.15 Die Wahrheit über die Array-Initialisierung * ... 282
3.8.16 Mehrere Rückgabewerte * ... 283
3.8.17 Klonen kann sich lohnen – Arrays vermehren * ... 284
3.8.18 Array-Inhalte kopieren * ... 285

3.8.19 Die Klasse Arrays zum Vergleichen, Füllen, Suchen, Sortieren nutzen ... 286
3.8.20 Eine lange Schlange ... 299

3.9 Der Einstiegspunkt für das Laufzeitsystem: main(...) ... 302
3.9.1 Korrekte Deklaration der Startmethode ... 302
3.9.2 Kommandozeilenargumente verarbeiten ... 303
3.9.3 Der Rückgabetyp von main(...) und System.exit(int) * ... 304

3.10 Zum Weiterlesen ... 306

4 Der Umgang mit Zeichenketten 307

4.1 Von ASCII über ISO-8859-1 zu Unicode ... 307
4.1.1 ASCII ... 307
4.1.2 ISO/IEC 8859-1 ... 308
4.1.3 Unicode ... 309
4.1.4 Unicode-Zeichenkodierung ... 310
4.1.5 Escape-Sequenzen/Fluchtsymbole ... 311
4.1.6 Schreibweise für Unicode-Zeichen und Unicode-Escapes ... 312
4.1.7 Java-Versionen gehen mit Unicode-Standard Hand in Hand * ... 314

4.2 Die Character-Klasse ... 316
4.2.1 Ist das so? ... 316
4.2.2 Zeichen in Großbuchstaben/Kleinbuchstaben konvertieren ... 318
4.2.3 Von char in int: vom Zeichen zur Zahl * ... 319

4.3 Zeichenfolgen ... 321

4.4 Die Klasse String und ihre Methoden ... 323
4.4.1 String-Literale als String-Objekte für konstante Zeichenketten ... 323
4.4.2 Konkatenation mit + ... 324
4.4.3 String-Länge und Test auf Leer-String ... 324
4.4.4 Zugriff auf ein bestimmtes Zeichen mit charAt(int) ... 325
4.4.5 Nach enthaltenen Zeichen und Zeichenfolgen suchen ... 326
4.4.6 Das Hangman-Spiel ... 329
4.4.7 Gut, dass wir verglichen haben ... 331
4.4.8 String-Teile extrahieren ... 335
4.4.9 Strings anhängen, zusammenfügen, Groß-/Kleinschreibung und Leerraum 339
4.4.10 Gesucht, gefunden, ersetzt ... 342
4.4.11 String-Objekte mit Konstruktoren erzeugen * ... 345

4.5 Veränderbare Zeichenketten mit StringBuilder und StringBuffer ... 348
4.5.1 Anlegen von StringBuilder-/StringBuffer-Objekten ... 350
4.5.2 StringBuilder/StringBuffer in andere Zeichenkettenformate konvertieren ... 351

4.5.3 Zeichen(folgen) erfragen ... 351
4.5.4 Daten anhängen ... 351
4.5.5 Zeichen(folgen) setzen, löschen und umdrehen ... 353
4.5.6 Länge und Kapazität eines StringBuilder-/StringBuffer-Objekts * ... 356
4.5.7 Vergleichen von Strings mit StringBuilder und StringBuffer ... 357
4.5.8 hashCode() bei StringBuilder/StringBuffer * ... 358

4.6 CharSequence als Basistyp ... 359

4.7 Konvertieren zwischen Primitiven und Strings ... 361
4.7.1 Unterschiedliche Typen in String-Repräsentationen konvertieren ... 362
4.7.2 String-Inhalt in einen primitiven Wert konvertieren ... 363
4.7.3 String-Repräsentation im Format Binär, Hex, Oktal * ... 365
4.7.4 parseXXX(...)- und printXXX()-Methoden in DatatypeConverter * ... 369

4.8 Strings zusammenhängen (konkatenieren) ... 370
4.8.1 Strings mit StringJoiner zusammenhängen ... 371

4.9 Zerlegen von Zeichenketten ... 373
4.9.1 Splitten von Zeichenketten mit split(...) ... 373
4.9.2 Yes we can, yes we scan – die Klasse Scanner ... 374

4.10 Ausgaben formatieren ... 378
4.10.1 Formatieren und Ausgeben mit format() ... 378

4.11 Zum Weiterlesen ... 384

5 Eigene Klassen schreiben 385

5.1 Eigene Klassen mit Eigenschaften deklarieren ... 385
5.1.1 Attribute deklarieren ... 386
5.1.2 Methoden deklarieren ... 388
5.1.3 Die this-Referenz ... 392

5.2 Privatsphäre und Sichtbarkeit ... 396
5.2.1 Für die Öffentlichkeit: public ... 396
5.2.2 Kein Public Viewing – Passwörter sind privat ... 397
5.2.3 Wieso nicht freie Methoden und Variablen für alle? ... 398
5.2.4 Privat ist nicht ganz privat: Es kommt darauf an, wer's sieht * ... 399
5.2.5 Zugriffsmethoden für Attribute deklarieren ... 399
5.2.6 Setter und Getter nach der JavaBeans-Spezifikation ... 400
5.2.7 Paketsichtbar ... 402
5.2.8 Zusammenfassung zur Sichtbarkeit ... 404

5.3 Eine für alle – statische Methode und statische Attribute ... 406
5.3.1 Warum statische Eigenschaften sinnvoll sind ... 407
5.3.2 Statische Eigenschaften mit static ... 407
5.3.3 Statische Eigenschaften über Referenzen nutzen? * ... 409
5.3.4 Warum die Groß- und Kleinschreibung wichtig ist * ... 409
5.3.5 Statische Variablen zum Datenaustausch * ... 410
5.3.6 Statische Eigenschaften und Objekteigenschaften * ... 412

5.4 Konstanten und Aufzählungen ... 413
5.4.1 Konstanten über statische finale Variablen ... 413
5.4.2 Typunsichere Aufzählungen ... 414
5.4.3 Aufzählungstypen: Typsichere Aufzählungen mit enum ... 416

5.5 Objekte anlegen und zerstören ... 421
5.5.1 Konstruktoren schreiben ... 421
5.5.2 Verwandtschaft von Methode und Konstruktor ... 422
5.5.3 Der Standard-Konstruktor (default constructor) ... 423
5.5.4 Parametrisierte und überladene Konstruktoren ... 424
5.5.5 Copy-Konstruktor ... 427
5.5.6 Einen anderen Konstruktor der gleichen Klasse mit this(...) aufrufen ... 428
5.5.7 Ihr fehlt uns nicht – der Garbage-Collector ... 431

5.6 Klassen- und Objektinitialisierung * ... 433
5.6.1 Initialisierung von Objektvariablen ... 433
5.6.2 Statische Blöcke als Klasseninitialisierer ... 435
5.6.3 Initialisierung von Klassenvariablen ... 436
5.6.4 Eincompilierte Belegungen der Klassenvariablen ... 437
5.6.5 Exemplarinitialisierer (Instanzinitialisierer) ... 438
5.6.6 Finale Werte im Konstruktor und in statischen Blöcken setzen ... 441

5.7 Zum Weiterlesen ... 443

6 Objektorientierte Beziehungsfragen ... 445

6.1 Assoziationen zwischen Objekten ... 445
6.1.1 Unidirektionale 1:1-Beziehung ... 446
6.1.2 Zwei Freunde müsst ihr werden – bidirektionale 1:1-Beziehungen ... 447
6.1.3 Unidirektionale 1:n-Beziehung ... 448

6.2 Vererbung ... 451
6.2.1 Vererbung in Java ... 452
6.2.2 Spielobjekte modellieren ... 453
6.2.3 Die implizite Basisklasse java.lang.Object ... 455

6.2.4 Einfach- und Mehrfachvererbung * ... 455
6.2.5 Die Sichtbarkeit protected ... 456
6.2.6 Konstruktoren in der Vererbung und super(...) ... 456

6.3 Typen in Hierarchien ... 462
6.3.1 Automatische und explizite Typumwandlung ... 462
6.3.2 Das Substitutionsprinzip ... 464
6.3.3 Typen mit dem instanceof-Operator testen ... 466

6.4 Methoden überschreiben ... 468
6.4.1 Methoden in Unterklassen mit neuem Verhalten ausstatten ... 468
6.4.2 Mit super an die Eltern ... 473
6.4.3 Finale Klassen und finale Methoden ... 475
6.4.4 Kovariante Rückgabetypen ... 477
6.4.5 Array-Typen und Kovarianz * ... 478

6.5 Drum prüfe, wer sich dynamisch bindet ... 479
6.5.1 Gebunden an toString() ... 480
6.5.2 Implementierung von System.out.println(Object) ... 482
6.5.3 Nicht dynamisch gebunden bei privaten, statischen und finalen Methoden 483
6.5.4 Dynamisch gebunden auch bei Konstruktoraufrufen * ... 484
6.5.5 Eine letzte Spielerei mit Javas dynamischer Bindung und überdeckten Attributen * ... 486

6.6 Abstrakte Klassen und abstrakte Methoden ... 487
6.6.1 Abstrakte Klassen ... 488
6.6.2 Abstrakte Methoden ... 490

6.7 Schnittstellen ... 495
6.7.1 Schnittstellen sind neue Typen ... 495
6.7.2 Schnittstellen deklarieren ... 496
6.7.3 Abstakte Methoden in Schnittstellen ... 496
6.7.4 Implementieren von Schnittstellen ... 497
6.7.5 Ein Polymorphie-Beispiel mit Schnittstellen ... 499
6.7.6 Die Mehrfachvererbung bei Schnittstellen ... 501
6.7.7 Keine Kollisionsgefahr bei Mehrfachvererbung * ... 505
6.7.8 Erweitern von Interfaces – Subinterfaces ... 506
6.7.9 Konstantendeklarationen bei Schnittstellen ... 507
6.7.10 Nachträgliches Implementieren von Schnittstellen * ... 510
6.7.11 Statische ausprogrammierte Methoden in Schnittstellen ... 510
6.7.12 Erweitern und Ändern von Schnittstellen ... 512
6.7.13 Default-Methoden ... 514
6.7.14 Erweiterte Schnittstellen deklarieren und nutzen ... 515
6.7.15 Öffentliche und private Schnittstellenmethoden ... 518
6.7.16 Erweiterte Schnittstellen, Mehrfachvererbung und Mehrdeutigkeiten * ... 519

6.7.17 Bausteine bilden mit Default-Methoden * ... 523
6.7.18 Initialisierung von Schnittstellenkonstanten * ... 529
6.7.19 Markierungsschnittstellen * ... 532
6.7.20 (Abstrakte) Klassen und Schnittstellen im Vergleich ... 533

6.8 Zum Weiterlesen ... 534

7 Ausnahmen müssen sein 535

7.1 Problembereiche einzäunen ... 535
7.1.1 Exceptions in Java mit try und catch ... 536
7.1.2 Eine NumberFormatException auffangen ... 536
7.1.3 Bitte nicht schlucken – leere catch-Blöcke ... 539
7.1.4 Wiederholung abgebrochener Bereiche * ... 540
7.1.5 Mehrere Ausnahmen auffangen ... 541
7.1.6 Ablauf einer Ausnahmesituation ... 543
7.1.7 throws im Methodenkopf angeben ... 544
7.1.8 Abschlussbehandlung mit finally ... 545

7.2 Die Klassenhierarchie der Fehler ... 550
7.2.1 Eigenschaften vom Exception-Objekt ... 550
7.2.2 Basistyp Throwable ... 551
7.2.3 Die Exception-Hierarchie ... 552
7.2.4 Oberausnahmen auffangen ... 552
7.2.5 Schon gefangen? ... 554
7.2.6 Alles geht als Exception durch ... 555
7.2.7 Zusammenfassen gleicher catch-Blöcke mit dem multi-catch ... 556

7.3 RuntimeException muss nicht aufgefangen werden ... 560
7.3.1 Beispiele für RuntimeException-Klassen ... 560
7.3.2 Kann man abfangen, muss man aber nicht ... 561

7.4 Harte Fehler – Error * ... 561

7.5 Auslösen eigener Exceptions ... 562
7.5.1 Mit throw Ausnahmen auslösen ... 563
7.5.2 Vorhandene Runtime-Fehlertypen kennen und nutzen ... 565
7.5.3 Parameter testen und gute Fehlermeldungen ... 567
7.5.4 Neue Exception-Klassen deklarieren ... 569
7.5.5 Eigene Ausnahmen als Unterklassen von Exception oder RuntimeException? ... 570
7.5.6 Ausnahmen abfangen und weiterleiten * ... 573
7.5.7 Aufruf-Stack von Ausnahmen verändern * ... 575

7.5.8 Präzises rethrow * ... 576
7.5.9 Geschachtelte Ausnahmen * ... 579

7.6 Automatisches Ressourcen-Management (try mit Ressourcen) ... 582
7.6.1 try mit Ressourcen ... 583
7.6.2 Die Schnittstelle AutoCloseable ... 585
7.6.3 Mehrere Ressourcen nutzen ... 587
7.6.4 try mit Ressourcen auf null-Ressourcen ... 588
7.6.5 Unterdrückte Ausnahmen * ... 588

7.7 Besonderheiten bei der Ausnahmebehandlung * ... 592
7.7.1 Rückgabewerte bei ausgelösten Ausnahmen ... 592
7.7.2 Ausnahmen und Rückgaben verschwinden – das Duo return und finally ... 592
7.7.3 throws bei überschriebenen Methoden ... 594
7.7.4 Nicht erreichbare catch-Klauseln ... 596

7.8 Assertions * ... 597
7.8.1 Assertions in eigenen Programmen nutzen ... 597
7.8.2 Assertions aktivieren ... 599

7.9 Zum Weiterlesen ... 601

8 Äußere.innere Typen 603

8.1 Geschachtelte (innere) Klassen, Schnittstellen, Aufzählungen ... 603

8.2 Statische innere Klassen und Schnittstellen ... 604

8.3 Mitglieds- oder Elementklassen ... 606
8.3.1 Exemplare innerer Klassen erzeugen ... 606
8.3.2 Die this-Referenz ... 607
8.3.3 Vom Compiler generierte Klassendateien * ... 608
8.3.4 Erlaubte Modifizierer bei äußeren und inneren Klassen ... 609
8.3.5 Innere Klassen greifen auf private Eigenschaften zu ... 609

8.4 Lokale Klassen ... 611
8.4.1 Beispiel mit eigener Klassendeklaration ... 611
8.4.2 Lokale Klasse für einen Timer nutzen ... 612

8.5 Anonyme innere Klassen ... 613
8.5.1 Nutzung einer anonymen inneren Klasse für den Timer ... 614
8.5.2 Umsetzung innerer anonymer Klassen * ... 615
8.5.3 Konstruktoren innerer anonymer Klassen ... 615

8.6 Zugriff auf lokale Variablen aus lokalen inneren und anonymen Klassen * ... 617

8.7 this in Unterklassen * ... 618

8.8 Zum Weiterlesen ... 620

9 Besondere Typen der Java SE 621

9.1 Object ist die Mutter aller Klassen ... 622
9.1.1 Klassenobjekte ... 622
9.1.2 Objektidentifikation mit toString() ... 623
9.1.3 Objektgleichheit mit equals(...) und Identität ... 625
9.1.4 Klonen eines Objekts mit clone() * ... 631
9.1.5 Hashwerte über hashCode() liefern * ... 636
9.1.6 System.identityHashCode(...) und das Problem der nicht eindeutigen Objektverweise * ... 642
9.1.7 Aufräumen mit finalize() * ... 644
9.1.8 Synchronisation * ... 646

9.2 Schwache Referenzen und Cleaner ... 647

9.3 Die Utility-Klasse java.util.Objects ... 648
9.3.1 Eingebaute null-Tests für equals(...)/hashCode() ... 648
9.3.2 Objects.toString(...) ... 649
9.3.3 null-Prüfungen mit eingebauter Ausnahmebehandlung ... 649
9.3.4 Tests auf null ... 651
9.3.5 Indexbezogene Programmargumente auf Korrektheit prüfen ... 651

9.4 Vergleichen von Objekten und Ordnung herstellen ... 652
9.4.1 Natürlich geordnet oder nicht? ... 652
9.4.2 Die Schnittstelle Comparable ... 653
9.4.3 Die Schnittstelle Comparator ... 654
9.4.4 Rückgabewerte kodieren die Ordnung ... 654
9.4.5 Statische und Default-Methoden in Comparator ... 657

9.5 Wrapper-Klassen und Autoboxing ... 660
9.5.1 Wrapper-Objekte erzeugen ... 662
9.5.2 Konvertierungen in eine String-Repräsentation ... 663
9.5.3 Von einer String-Repräsentation parsen ... 664
9.5.4 Die Basisklasse Number für numerische Wrapper-Objekte ... 665
9.5.5 Vergleiche durchführen mit compareXXX(...), compareTo(...), equals(...) und Hashwerten ... 666
9.5.6 Statische Reduzierungsmethoden in Wrapper-Klassen ... 670
9.5.7 Konstanten für die Größe eines primitiven Typs ... 671
9.5.8 Behandeln von vorzeichenlosen Zahlen * ... 671

9.5.9 Die Klasse Integer ... 673
9.5.10 Die Klassen Double und Float für Fließkommazahlen ... 674
9.5.11 Die Long-Klasse ... 674
9.5.12 Die Boolean-Klasse ... 674
9.5.13 Autoboxing: Boxing und Unboxing ... 676

9.6 Iterator, Iterable * ... 680
9.6.1 Die Schnittstelle Iterator ... 680
9.6.2 Wer den Iterator liefert ... 683
9.6.3 Die Schnittstelle Iterable ... 684
9.6.4 Erweitertes for und Iterable ... 684
9.6.5 Interne Iteration ... 685
9.6.6 Einen eigenen Iterable implementieren * ... 685

9.7 Die Spezial-Oberklasse Enum ... 687
9.7.1 Methoden auf Enum-Objekten ... 688
9.7.2 Aufzählungen mit eigenen Methoden und Initialisierern * ... 691
9.7.3 enum mit eigenen Konstruktoren * ... 694

9.8 Annotationen in der Java SE ... 697
9.8.1 Orte für Annotationen ... 698
9.8.2 Annotationstypen aus java.lang ... 698
9.8.3 @Deprecated ... 699
9.8.4 Annotationen mit zusätzlichen Informationen ... 699
9.8.5 @SuppressWarnings ... 700

9.9 Zum Weiterlesen ... 703

10 Generics<T> 705

10.1 Einführung in Java Generics ... 705
10.1.1 Mensch versus Maschine – Typprüfung des Compilers und der Laufzeitumgebung ... 705
10.1.2 Taschen ... 706
10.1.3 Generische Typen deklarieren ... 708
10.1.4 Generics nutzen ... 709
10.1.5 Diamonds are forever ... 712
10.1.6 Generische Schnittstellen ... 715
10.1.7 Generische Methoden/Konstruktoren und Typ-Inferenz ... 717

10.2 Umsetzen der Generics, Typlöschung und Raw-Types ... 721
10.2.1 Realisierungsmöglichkeiten ... 721
10.2.2 Typlöschung (Type Erasure) ... 721

10.2.3 Probleme der Typlöschung ... 723
10.2.4 Raw-Type ... 728

10.3 Einschränken der Typen über Bounds ... 731
10.3.1 Einfache Einschränkungen mit extends ... 731
10.3.2 Weitere Obertypen mit & ... 733

10.4 Typparameter in der throws-Klausel * ... 734
10.4.1 Deklaration einer Klasse mit Typvariable <E extends Exception> ... 734
10.4.2 Parametrisierter Typ bei Typvariable <E extends Exception> ... 735

10.5 Generics und Vererbung, Invarianz ... 737
10.5.1 Arrays sind kovariant ... 738
10.5.2 Generics sind nicht kovariant, sondern invariant ... 738
10.5.3 Wildcards mit ? ... 739
10.5.4 Bounded Wildcards ... 741
10.5.5 Bounded-Wildcard-Typen und Bounded-Typvariablen ... 745
10.5.6 Das LESS-Prinzip ... 747
10.5.7 Enum<E extends Enum<E>> * ... 749

10.6 Konsequenzen der Typlöschung: Typ-Token, Arrays und Brücken * ... 751
10.6.1 Typ-Token ... 751
10.6.2 Super-Type-Token ... 753
10.6.3 Generics und Arrays ... 754
10.6.4 Brückenmethoden ... 755

10.7 Zum Weiterlesen ... 761

11 Lambda-Ausdrücke und funktionale Programmierung 763

11.1 Code = Daten ... 763

11.2 Funktionale Schnittstellen und Lambda-Ausdrücke im Detail ... 766
11.2.1 Funktionale Schnittstellen ... 767
11.2.2 Typ eines Lambda-Ausdrucks ergibt sich durch Zieltyp ... 768
11.2.3 Annotation @FunctionalInterface ... 772
11.2.4 Syntax für Lambda-Ausdrücke ... 773
11.2.5 Die Umgebung der Lambda-Ausdrücke und Variablenzugriffe ... 778
11.2.6 Ausnahmen in Lambda-Ausdrücken ... 781
11.2.7 Klassen mit einer abstrakten Methode als funktionale Schnittstelle? * ... 785

11.3 Methodenreferenz ... 786
11.3.1 Varianten von Methodenreferenzen ... 788

11.4 Konstruktorreferenz ... 790
11.4.1 Parameterlose und parametrisierte Konstruktoren ... 792
11.4.2 Nützliche vordefinierte Schnittstellen für Konstruktorreferenzen ... 792

11.5 Implementierung von Lambda-Ausdrücken * ... 793

11.6 Funktionale Programmierung mit Java ... 794
11.6.1 Programmierparadigmen: imperativ oder deklarativ ... 794
11.6.2 Funktionale Programmierung und funktionale Programmiersprachen ... 795
11.6.3 Funktionale Programmierung in Java am Beispiel vom Comparator ... 796
11.6.4 Lambda-Ausdrücke als Funktionen sehen ... 797

11.7 Funktionale Schnittstelle aus dem java.util.function-Paket ... 798
11.7.1 Blöcke mit Code und die funktionale Schnittstelle Consumer ... 799
11.7.2 Supplier ... 801
11.7.3 Prädikate und java.util.function.Predicate ... 801
11.7.4 Funktionen und die allgemeine funktionale Schnittstelle java.util.function.Function ... 804
11.7.5 Ein bisschen Bi 807
11.7.6 Funktionale Schnittstellen mit Primitiven ... 810

11.8 Optional ist keine Nullnummer ... 813
11.8.1 Optional-Typ ... 815
11.8.2 Primitive optionale Typen ... 818
11.8.3 Erstmal funktional mit Optional ... 819

11.9 Was ist jetzt so funktional? ... 824

11.10 Zum Weiterlesen ... 826

12 Architektur, Design und angewandte Objektorientierung 829

12.1 Architektur, Design und Implementierung ... 829

12.2 Design-Pattern (Entwurfsmuster) ... 830
12.2.1 Motivation für Design-Pattern ... 830
12.2.2 Singleton ... 831
12.2.3 Fabrikmethoden ... 834
12.2.4 Das Beobachter-Pattern mit Listener realisieren ... 835

12.3 Zum Weiterlesen ... 839

13 Komponenten, JavaBeans und Module 841

13.1 JavaBeans ... 841
13.1.1 Properties (Eigenschaften) ... 842
13.1.2 Einfache Eigenschaften ... 843
13.1.3 Indizierte Eigenschaften ... 843
13.1.4 Gebundene Eigenschaften und PropertyChangeListener ... 843
13.1.5 Veto-Eigenschaften – dagegen! ... 847

13.2 JavaFX Properties ... 850
13.2.1 javafx.beans-Paket mit XXXProperty-Klassen ... 851
13.2.2 Property-Veränderungen registrieren ... 852
13.2.3 Beans-Binding ... 853
13.2.4 Property-Schnittstelle und bindXXX(...)-Methoden ... 854
13.2.5 XXXProperty-Beziehungen (für Typ-Fetischisten) * ... 861
13.2.6 Ausblick ... 863

13.3 Klassenlader (Class Loader) und Modul-/Klassenpfad ... 864
13.3.1 Klassenladen auf Abruf ... 864
13.3.2 Klassenlader bei der Arbeit zusehen ... 865
13.3.3 JMOD-Dateien und JAR-Dateien ... 866
13.3.4 Woher die kleinen Klassen kommen: die Suchorte und spezielle Klassenlader ... 867
13.3.5 Setzen des Modulpfades ... 867

13.4 Module entwickeln und einbinden ... 869
13.4.1 Wer sieht wen ... 870
13.4.2 Plattform-Module und JMOD-Beispiel ... 871
13.4.3 Verbotene Plattformeigenschaften nutzen, --add-exports ... 872
13.4.4 Plattformmodule einbinden, --add-modules und --add-opens ... 873
13.4.5 Projektabhängigkeiten in Eclipse ... 875
13.4.6 Benannte Module und module-info.java ... 877
13.4.7 Automatische Module ... 881
13.4.8 Unbenanntes Modul ... 882
13.4.9 Lesbarkeit und Zugreifbarkeit ... 883
13.4.10 Modul-Migration ... 884

13.5 Zum Weiterlesen ... 885

14 Die Klassenbibliothek 887

14.1 Die Java-Klassenphilosophie ... 887
14.1.1 Modul, Paket, Typ ... 887
14.1.2 Übersicht über die Pakete der Standardbibliothek ... 891

14.2 Die Klasse Class ... 895
14.2.1 An ein Class-Objekt kommen ... 895
14.2.2 Eine Class ist ein Type ... 898

14.3 Metadaten der Typen mit dem Class-Objekt ... 898
14.3.1 Der Name des Typs ... 899
14.3.2 Was das Class-Objekt beschreibt * ... 902
14.3.3 instanceof mit Class-Objekten * ... 904
14.3.4 Oberklassen finden * ... 905
14.3.5 Implementierte Interfaces einer Klasse oder eines Interfaces * ... 906
14.3.6 Modifizierer und die Klasse Modifier * ... 906
14.3.7 Die Arbeit auf dem Feld * ... 908

14.4 Die Utility-Klassen System und Properties ... 909
14.4.1 Systemeigenschaften der Java-Umgebung ... 910
14.4.2 Zeilenumbruchzeichen, line.separator ... 912
14.4.3 Eigene Properties von der Konsole aus setzen * ... 912
14.4.4 Umgebungsvariablen des Betriebssystems * ... 914
14.4.5 Einfache Zeitmessung und Profiling * ... 916

14.5 Sprachen der Länder ... 919
14.5.1 Sprachen in Regionen über Locale-Objekte ... 919

14.6 Wichtige Datum-Klassen im Überblick ... 923
14.6.1 Der 1.1.1970 ... 924
14.6.2 System.currentTimeMillis() ... 924
14.6.3 Einfache Zeitumrechnungen durch TimeUnit ... 924

14.7 Date-Time-API ... 925
14.7.1 Menschenzeit und Maschinenzeit ... 927
14.7.2 Datumsklasse LocalDate ... 930
14.7.3 Die Klasse YearMonth ... 931
14.7.4 Die Klasse MonthDay ... 931
14.7.5 Aufzählung DayOfWeek und Month ... 932
14.7.6 Klasse LocalTime ... 933
14.7.7 Klasse LocalDateTime ... 933
14.7.8 Klasse Year ... 934

14.8 Logging mit java.util.logging ... 934

14.9 Maven: Build-Management und Abhängigkeiten auflösen ... 941

14.9.1 Beispielprojekt in Eclipse mit Maven ... 942
14.9.2 Properties hinzunehmen ... 942
14.9.3 Dependency hinzunehmen ... 942
14.9.4 Lokales- und Remote-Repository ... 944
14.9.5 Lebenszylus, Phasen und Maven-Plugins ... 944
14.9.6 Archetypes ... 944

14.10 Zum Weiterlesen ... 945

15 Einführung in die nebenläufige Programmierung 947

15.1 Nebenläufigkeit und Parallelität ... 947

15.1.1 Multitasking, Prozesse, Threads ... 948
15.1.2 Threads und Prozesse ... 948
15.1.3 Wie nebenläufige Programme die Geschwindigkeit steigern können ... 949
15.1.4 Was Java für Nebenläufigkeit alles bietet ... 951

15.2 Threads erzeugen ... 951

15.2.1 Threads über die Schnittstelle Runnable implementieren ... 951
15.2.2 Thread mit Runnable starten ... 953
15.2.3 Die Klasse Thread erweitern ... 955

15.3 Thread-Eigenschaften und Zustände ... 957

15.3.1 Der Name eines Threads ... 957
15.3.2 Wer bin ich? ... 958
15.3.3 Die Zustände eines Threads * ... 958
15.3.4 Schläfer gesucht ... 959
15.3.5 Mit yield() und onSpinWait() auf Rechenzeit verzichten ... 961
15.3.6 Der Thread als Dämon ... 962
15.3.7 Freiheit für den Thread – das Ende ... 964
15.3.8 Einen Thread höflich mit Interrupt beenden ... 964
15.3.9 Ein Rendezvous mit join(...) * ... 967
15.3.10 Arbeit niederlegen und wieder aufnehmen * ... 969

15.4 Der Ausführer (Executor) kommt ... 969

15.4.1 Die Schnittstelle Executor ... 970
15.4.2 Glücklich in der Gruppe – die Thread-Pools ... 972
15.4.3 Threads mit Rückgabe über Callable ... 973
15.4.4 Mehrere Callable-Objekte abarbeiten ... 977

15.4.5 ScheduledExecutorService für wiederholende Ausgaben und Zeitsteuerungen nutzen ... 978
15.4.6 Asynchrones Programmieren mit CompletableFuture (CompletionStage) ... 979

15.5 Zum Weiterlesen ... 981

16 Einführung in Datenstrukturen und Algorithmen 983

16.1 Listen ... 983
16.1.1 Erstes Listen-Beispiel ... 984
16.1.2 Auswahlkriterium ArrayList oder LinkedList ... 985
16.1.3 Die Schnittstelle List ... 985
16.1.4 ArrayList ... 991
16.1.5 LinkedList ... 993
16.1.6 Der Array-Adapter Arrays.asList(...) ... 994
16.1.7 toArray(...) von Collection verstehen – die Gefahr einer Falle erkennen ... 996
16.1.8 Primitive Elemente in Datenstrukturen verwalten ... 999

16.2 Mengen (Sets) ... 1000
16.2.1 Ein erstes Mengen-Beispiel ... 1000
16.2.2 Methoden der Schnittstelle Set ... 1002
16.2.3 HashSet ... 1003
16.2.4 TreeSet – die sortierte Menge ... 1004
16.2.5 Die Schnittstellen NavigableSet und SortedSet ... 1006
16.2.6 LinkedHashSet ... 1008

16.3 Assoziative Speicher ... 1009
16.3.1 Die Klassen HashMap und TreeMap ... 1009
16.3.2 Einfügen und Abfragen des Assoziativspeichers ... 1012
16.3.3 Über die Bedeutung von equals(...) und hashCode() bei Elementen ... 1020
16.3.4 Eigene Objekte hashen ... 1020
16.3.5 LinkedHashMap und LRU-Implementierungen ... 1022
16.3.6 IdentityHashMap ... 1023
16.3.7 Das Problem veränderter Elemente ... 1023
16.3.8 Aufzählungen und Ansichten des Assoziativspeichers ... 1024
16.3.9 Die Arbeitsweise einer Hash-Tabelle * ... 1027
16.3.10 Die Properties-Klasse ... 1030

16.4 Immutable Datenstrukturen ... 1033
16.4.1 Nichtänderbare Datenstrukturen, immutable oder nur Lesen? ... 1034
16.4.2 Null Object Pattern und leere Sammlungen/Iteratoren zurückgeben ... 1035
16.4.3 Immutable Datenstrukturen mit einem Element: Singletons ... 1037

16.4.4 Collections.unmodifiableXXX(...) ... 1038
16.4.5 Statische ofXXX(...)-Methoden zum Aufbau unveränderbarer Set-, List-, Map-Datenstrukturen ... 1041

16.5 Stream-API ... 1043
16.5.1 Stream erzeugen ... 1046
16.5.2 Terminale Operationen ... 1049
16.5.3 Intermediäre Operationen ... 1060

16.6 Zum Weiterlesen ... 1067

17 Einführung in grafische Oberflächen 1069

17.1 GUI-Frameworks ... 1069
17.1.1 Kommandozeile ... 1069
17.1.2 Grafische Benutzeroberfläche ... 1069
17.1.3 Abstract Window Toolkit (AWT) ... 1070
17.1.4 Java Foundation Classes und Swing ... 1070
17.1.5 JavaFX ... 1070
17.1.6 SWT (Standard Widget Toolkit) * ... 1072

17.2 Deklarative und programmierte Oberflächen ... 1073
17.2.1 GUI-Beschreibungen in JavaFX ... 1074
17.2.2 Deklarative GUI-Beschreibungen für Swing? ... 1074

17.3 GUI-Builder ... 1075
17.3.1 GUI-Builder für JavaFX ... 1075
17.3.2 GUI-Builder für Swing ... 1076

17.4 Aller Swing-Anfang – Fenster zur Welt ... 1076
17.4.1 Eine Uhr, bei der die Zeit nie vergeht ... 1076
17.4.2 Swing-Fenster mit javax.swing.JFrame darstellen ... 1077
17.4.3 Mit add(...) auf den Container ... 1078
17.4.4 Fenster schließbar machen – setDefaultCloseOperation(int) ... 1078
17.4.5 Sichtbarkeit des Fensters ... 1079
17.4.6 Größe und Position des Fensters verändern ... 1079
17.4.7 Fenster- und Dialogdekoration, Transparenz * ... 1080
17.4.8 Die Klasse Toolkit * ... 1081
17.4.9 Zum Vergleich: AWT-Fenster darstellen * ... 1082

17.5 Es tut sich was – Ereignisse beim AWT ... 1084
17.5.1 Die Ereignisquellen und Horcher (Listener) von Swing ... 1084
17.5.2 Listener implementieren ... 1085
17.5.3 Listener bei dem Ereignisauslöser anmelden/abmelden ... 1088

17.5.4 Adapterklassen nutzen ... 1090
17.5.5 Listener-Code in inneren Klassen und Lambda-Ausdrücken ... 1092
17.5.6 Aufrufen der Listener im AWT-Event-Thread ... 1094
17.5.7 Ereignisse, etwas genauer betrachtet * ... 1094

17.6 Schaltflächen ... 1097
17.6.1 Normale Schaltflächen (JButton) ... 1097
17.6.2 Der aufmerksame ActionListener ... 1099
17.6.3 Schaltflächen-Ereignisse vom Typ ActionEvent ... 1101
17.6.4 Basisklasse AbstractButton ... 1101
17.6.5 Wechselknopf (JToggleButton) ... 1103

17.7 Textkomponenten ... 1103
17.7.1 Text in einer Eingabezeile ... 1104
17.7.2 Die Oberklasse der Textkomponenten (JTextComponent) ... 1106
17.7.3 Geschützte Eingaben (JPasswordField) ... 1107
17.7.4 Validierende Eingabefelder (JFormattedTextField) ... 1108
17.7.5 Einfache mehrzeilige Textfelder (JTextArea) ... 1109
17.7.6 Editor-Klasse (JEditorPane) * ... 1112

17.8 Grundlegendes zum Zeichnen ... 1114
17.8.1 Die paint(Graphics)-Methode für den AWT-Frame ... 1115
17.8.2 Die ereignisorientierte Programmierung ändert Fensterinhalte ... 1116
17.8.3 Zeichnen von Inhalten auf ein JFrame ... 1118
17.8.4 Auffordern zum Neuzeichnen mit repaint(...) ... 1119
17.8.5 Java 2D-API ... 1120

17.9 Zum Weiterlesen ... 1120

18 Einführung in Dateien und Datenströme 1121

18.1 Alte und neue Welt in java.io und java.nio ... 1121
18.1.1 java.io-Paket mit File-Klasse ... 1121
18.1.2 NIO.2 und java.nio-Paket ... 1122
18.1.3 java.io.File oder java.nio.*? ... 1122

18.2 Dateisysteme und Pfade ... 1123
18.2.1 FileSystem und Path ... 1123
18.2.2 Die Utility-Klasse Files ... 1129
18.2.3 Dateien kopieren und verschieben ... 1131

18.3 Dateien mit wahlfreiem Zugriff ... 1134
18.3.1 Ein RandomAccessFile zum Lesen und Schreiben öffnen ... 1134
18.3.2 Aus dem RandomAccessFile lesen ... 1135

18.3.3 Schreiben mit RandomAccessFile ... 1137
18.3.4 Die Länge des RandomAccessFile ... 1138
18.3.5 Hin und her in der Datei ... 1138

18.4 Basisklassen für die Ein-/Ausgabe ... 1139
18.4.1 Die vier abstrakten Basisklassen ... 1140
18.4.2 Die abstrakte Basisklasse OutputStream ... 1140
18.4.3 Ein Datenschlucker * ... 1143
18.4.4 Die abstrakte Basisklasse InputStream ... 1143
18.4.5 Die abstrakte Basisklasse Writer ... 1145
18.4.6 Die Schnittstelle Appendable * ... 1147
18.4.7 Die abstrakte Basisklasse Reader ... 1147
18.4.8 Die Schnittstellen Closeable, AutoCloseable und Flushable ... 1150

18.5 Lesen aus Dateien und Schreiben in Dateien ... 1152
18.5.1 Byteorientierte Datenströme über Files beziehen ... 1152
18.5.2 Zeichenorientierte Datenströme über Files beziehen ... 1153
18.5.3 Funktion von OpenOption bei den Files.newXXX(...)-Methoden ... 1155
18.5.4 Ressourcen aus dem Modulpfad und aus JAR-Dateien laden ... 1157

18.6 Vermittler zwischen Byte-Streams und Unicode-Strömen ... 1158
18.6.1 Datenkonvertierung durch den OutputStreamWriter ... 1158
18.6.2 Automatische Konvertierungen mit dem InputStreamReader ... 1160

18.7 Zum Weiterlesen ... 1161

19 Einführung ins Datenbankmanagement mit JDBC 1163

19.1 Relationale Datenbanken und Datenbankmanagementsysteme ... 1163
19.1.1 Das relationale Modell ... 1163
19.1.2 Datenbanken und Tools ... 1164
19.1.3 HSQLDB ... 1164
19.1.4 Weitere Datenbanken * ... 1166
19.1.5 Eclipse Data Tools Platform (DTP) zum Durchschauen von Datenbanken ... 1167

19.2 JDBC und Datenbanktreiber ... 1169
19.2.1 JDBC-Versionen * ... 1170

19.3 Eine Beispielabfrage ... 1171
19.3.1 Schritte zur Datenbankabfrage ... 1171
19.3.2 Ein Client für die HSQLDB-Datenbank ... 1172

19.4 Zum Weiterlesen ... 1173

20 Einführung in <XML> 1175

20.1 Auszeichnungssprachen 1175
20.1.1 Die Standard Generalized Markup Language (SGML) 1175
20.1.2 Extensible Markup Language (XML) 1176

20.2 Eigenschaften von XML-Dokumenten 1176
20.2.1 Elemente und Attribute 1176
20.2.2 Beschreibungssprache für den Aufbau von XML-Dokumenten 1179
20.2.3 Schema – die moderne Alternative zu DTD 1183
20.2.4 Namensraum (Namespace) 1186
20.2.5 XML-Applikationen * 1187

20.3 Die Java-APIs für XML 1188
20.3.1 Das Document Object Model (DOM) 1188
20.3.2 Simple API for XML Parsing (SAX) 1189
20.3.3 Pull-API StAX 1189
20.3.4 Java Document Object Model (JDOM) 1189
20.3.5 JAXP als Java-Schnittstelle zu XML 1190
20.3.6 DOM-Bäume einlesen mit JAXP * 1190

20.4 Java Architecture for XML Binding (JAXB) 1191
20.4.1 Bean für JAXB aufbauen 1191
20.4.2 Utility-Klasse JAXB 1192
20.4.3 Ganze Objektgraphen schreiben und lesen 1193
20.4.4 JAXBContext und Marshaller/Unmarshaller nutzen 1195
20.4.5 Validierung 1197
20.4.6 Weitere JAXB-Annotationen * 1201
20.4.7 JAXB-Beans aus XML-Schema-Datei generieren 1208

20.5 Zum Weiterlesen 1214

21 Bits und Bytes, Mathematisches und Geld 1217

21.1 Bits und Bytes * 1217
21.1.1 Die Bit-Operatoren Komplement, Und, Oder und XOR 1218
21.1.2 Repräsentation ganzer Zahlen in Java – das Zweierkomplement 1219
21.1.3 Das binäre (Basis 2), oktale (Basis 8), hexadezimale (Basis 16) Stellenwertsystem 1220
21.1.4 Auswirkung der Typumwandlung auf die Bit-Muster 1221
21.1.5 Vorzeichenlos arbeiten 1224
21.1.6 Die Verschiebeoperatoren 1227

21.1.7 Ein Bit setzen, löschen, umdrehen und testen ... 1229
21.1.8 Bit-Methoden der Integer- und Long-Klasse ... 1229

21.2 Fließkomma-Arithmetik in Java ... 1231
21.2.1 Spezialwerte für Unendlich, Null, NaN ... 1231
21.2.2 Standardnotation und wissenschaftliche Notation bei Fließkommazahlen * 1234
21.2.3 Mantisse und Exponent * ... 1235

21.3 Die Eigenschaften der Klasse Math ... 1237
21.3.1 Attribute ... 1237
21.3.2 Absolutwerte und Vorzeichen ... 1237
21.3.3 Maximum/Minimum ... 1238
21.3.4 Runden von Werten ... 1238
21.3.5 Rest der ganzzahligen Division * ... 1241
21.3.6 Division mit Rundung Richtung negativ unendlich, alternativer Restwert * ... 1242
21.3.7 Multiply-Accumulate ... 1244
21.3.8 Wurzel- und Exponentialmethoden ... 1244
21.3.9 Der Logarithmus * ... 1245
21.3.10 Winkelmethoden * ... 1246
21.3.11 Zufallszahlen ... 1247

21.4 Genauigkeit, Wertebereich eines Typs und Überlaufkontrolle * ... 1248
21.4.1 Der größte und der kleinste Wert ... 1248
21.4.2 Überlauf und alles ganz exakt ... 1248
21.4.3 Was bitte macht eine ulp? ... 1251

21.5 Zufallszahlen: Random, SecureRandom, SplittableRandom ... 1253
21.5.1 Die Klasse Random ... 1253
21.5.2 Random-Objekte mit dem Samen aufbauen ... 1253
21.5.3 Einzelne Zufallszahlen erzeugen ... 1254
21.5.4 Pseudo-Zufallszahlen in der Normalverteilung * ... 1255
21.5.5 Strom von Zufallszahlen generieren * ... 1255
21.5.6 Die Klasse SecureRandom * ... 1257
21.5.7 SplittableRandom * ... 1257

21.6 Große Zahlen * ... 1257
21.6.1 Die Klasse BigInteger ... 1258
21.6.2 Beispiel: ganz lange Fakultäten mit BigInteger ... 1265
21.6.3 Große Fließkommazahlen mit BigDecimal ... 1266
21.6.4 Mit MathContext komfortabel die Rechengenauigkeit setzen ... 1269

21.7 Mathe bitte strikt * ... 1270
21.7.1 Strikte Fließkommaberechnungen mit strictfp ... 1271
21.7.2 Die Klassen Math und StrictMath ... 1271

21.8 Geld und Währung ... 1272
21.8.1 Geldbeträge repräsentieren ... 1272
21.8.2 ISO 4217 ... 1272
21.8.3 Währungen in Java repräsentieren ... 1273
21.9 Zum Weiterlesen ... 1274

22 Testen mit JUnit 1275

22.1 Softwaretests ... 1275
22.1.1 Vorgehen beim Schreiben von Testfällen ... 1276
22.2 Das Test-Framework JUnit ... 1276
22.2.1 Test-Driven Development und Test-First ... 1277
22.2.2 Testen, implementieren, testen, implementieren, testen, freuen ... 1278
22.2.3 JUnit-Tests ausführen ... 1279
22.2.4 assertXXX(...)-Methoden der Klasse Assert ... 1280
22.2.5 Matcher-Objekte und Hamcrest ... 1282
22.2.6 Exceptions testen ... 1286
22.2.7 Tests ignorieren und Grenzen für Ausführungszeiten festlegen ... 1287
22.2.8 Mit Methoden der Assume-Klasse Tests abbrechen ... 1288
22.3 Wie gutes Design das Testen ermöglicht ... 1288
22.4 Aufbau größerer Testfälle ... 1291
22.4.1 Fixtures ... 1291
22.4.2 Sammlungen von Testklassen und Klassenorganisation ... 1293
22.5 Dummy, Fake, Stub und Mock ... 1294
22.6 JUnit-Erweiterungen, Testzusätze ... 1295
22.7 Zum Weiterlesen ... 1296

23 Die Werkzeuge des JDK 1299

23.1 Übersicht ... 1299
23.1.1 Aufbau und gemeinsame Schalter ... 1300
23.2 Java-Quellen übersetzen ... 1300
23.2.1 Java-Compiler vom JDK ... 1301
23.2.2 Alternative Compiler ... 1301

23.2.3 Native Compiler ... 1302
23.2.4 Java-Programme in ein natives ausführbares Programm einpacken ... 1302

23.3 Die Java-Laufzeitumgebung ... 1303
23.3.1 Schalter der JVM ... 1303
23.3.2 Der Unterschied zwischen java.exe und javaw.exe ... 1306

23.4 jlink: der Java Linker ... 1306

23.5 Dokumentationskommentare mit Javadoc ... 1307
23.5.1 Einen Dokumentationskommentar setzen ... 1308
23.5.2 Mit dem Werkzeug javadoc eine Dokumentation erstellen ... 1310
23.5.3 HTML-Tags in Dokumentationskommentaren * ... 1311
23.5.4 Generierte Dateien ... 1311
23.5.5 Dokumentationskommentare im Überblick * ... 1311
23.5.6 Javadoc und Doclets * ... 1313
23.5.7 Veraltete (deprecated) Typen und Eigenschaften ... 1313
23.5.8 Javadoc-Überprüfung mit DocLint ... 1316

23.6 Das Archivformat JAR ... 1317
23.6.1 Das Dienstprogramm jar benutzen ... 1318
23.6.2 Das Manifest ... 1320
23.6.3 Applikationen in JAR-Archiven starten ... 1320
23.6.4 Pack200-Format * ... 1322

23.7 Zum Weiterlesen ... 1323

Java SE-Module und Paketübersicht ... 1325
Index ... 1345