

Auf einen Blick

1	Java ist auch eine Sprache	51
2	Sprachbeschreibung	89
3	Klassen und Objekte	185
4	Der Umgang mit Zeichenketten	237
5	Eigene Klassen schreiben	327
6	Exceptions	443
7	Generics<T>	477
8	Äußere.innere Klassen	489
9	Besondere Klassen der Java SE	503
10	Architektur, Design und angewandte Objektorientierung	549
11	Die Klassenbibliothek	573
12	Bits und Bytes und Mathematisches	597
13	Datenstrukturen und Algorithmen	633
14	Threads und nebenläufige Programmierung	713
15	Raum und Zeit	781
16	Dateien, Verzeichnisse und Dateizugriffe	815
17	Datenströme	839
18	Die eXtensible Markup Language (XML)	933
19	Grafische Oberflächen mit Swing	1005
20	Grafikprogrammierung	1179
21	Netzwerkprogrammierung	1239
22	Verteilte Programmierung mit RMI	1277
23	JavaServer Pages und Servlets	1295
24	Datenbankmanagement mit JDBC	1335
25	Reflection und Annotationen	1385
26	Dienstprogramme für die Java-Umgebung	1435

Inhalt

Vorwort	37
---------------	----

1 Java ist auch eine Sprache	50
1.1 Der erste Kontakt	51
1.2 Historischer Hintergrund	51
1.3 Eigenschaften von Java	53
1.3.1 Bytecode und die virtuelle Maschine	53
1.3.2 Objektorientierung in Java	55
1.3.3 Das Java-Security-Modell	55
1.3.4 Zeiger und Referenzen	56
1.3.5 Bring den Müll raus, Garbage-Collector!	57
1.3.6 Ausnahmebehandlung	57
1.3.7 Kein Präprozessor für Textersetzungen	58
1.3.8 Keine benutzerdefinierten überladenen Operatoren	58
1.3.9 Java als Sprache, Laufzeitumgebung und Standardbibliothek	59
1.3.10 Java ist Open Source	60
1.3.11 Wofür sich Java weniger eignet	61
1.3.12 Java im Vergleich zu anderen Sprachen	62
1.3.13 Java und das Web, Applets statt Apples	63
1.3.14 Features, Enhancements (Erweiterungen) und ein JSR	65
1.3.15 Entwicklung von Java und Zukunftsaussichten	65
1.4 Java-Plattformen: Java SE, Java EE und Java ME	67
1.4.1 Die Java SE-Plattform	67
1.4.2 Java für die Kleinen	68
1.4.3 Java für die ganz ganz Kleinen	69
1.4.4 Java für die Großen	69
1.5 Die Installation der Java Platform Standard Edition (Java SE)	70
1.5.1 Die Java SE von Oracle	70
1.5.2 Download des JDK	70
1.5.3 Java SE unter Windows installieren	71
1.6 Das erste Programm compilieren und testen	73
1.6.1 Ein Quadratzahlen-Programm	73
1.6.2 Der Compilerlauf	74
1.6.3 Die Laufzeitumgebung	75
1.6.4 Häufige Compiler- und Interpreterprobleme	76
1.7 Entwicklungsumgebungen im Allgemeinen	76
1.7.1 Die Entwicklungsumgebung Eclipse	77
1.7.2 NetBeans von Oracle	77
1.7.3 IntelliJ IDEA	78

1.7.4	Ein Wort zu Microsoft, Java und zu J++	78
1.8	Eclipse im Speziellen	79
1.8.1	Eclipse starten	79
1.8.2	Das erste Projekt anlegen	80
1.8.3	Eine Klasse hinzufügen	82
1.8.4	Übersetzen und Ausführen	83
1.8.5	JDK statt JRE	84
1.8.6	Start eines Programms ohne Speicheraufforderung	84
1.8.7	Projekt einfügen, Workspace für die Programme wechseln	85
1.8.8	Plugins für Eclipse	86
1.9	Zum Weiterlesen	86

2 Sprachbeschreibung 89

2.1	Elemente der Programmiersprache Java	89
2.1.1	Token	89
2.1.2	Textkodierung durch Unicode-Zeichen	90
2.1.3	Literale	92
2.1.4	Bezeichner	92
2.1.5	Reservierte Schlüsselwörter	95
2.1.6	Zusammenfassung der lexikalischen Analyse	95
2.1.7	Kommentare	96
2.2	Anweisungen formen Programme	98
2.2.1	Was sind Anweisungen?	98
2.2.2	Klassendeklaration	98
2.2.3	Die Reise beginnt am main()	99
2.2.4	Der erste Methodenaufruf: println()	100
2.2.5	Atomare Anweisungen und Anweisungssequenzen	101
2.2.6	Mehr zu print(), println() und printf() für Bildschirmausgaben	101
2.2.7	Die API-Dokumentation	103
2.2.8	Ausdrucksanweisung	104
2.2.9	Erste Idee der Objektorientierung	105
2.2.10	Modifizierer	105
2.3	Datentypen, Typisierung, Variablen und Zuweisungen	106
2.3.1	Primitive Datentypen im Überblick	107
2.3.2	Variablendeklarationen	110
2.3.3	Variablendeklaration mit Wertinitialisierung	110
2.3.4	Zuweisungsoperator	111
2.3.5	Wahrheitswerte	113
2.3.6	Ganzzahlige Datentypen und Literale	113
2.3.7	Das binäre (Basis 2), oktale (Basis 8), hexadezimale (Basis 16) Stellenwertsystem *	115
2.3.8	Die Fließkommazahlen »float« und »double«	116

2.3.9	Alphanumerische Zeichen	117
2.3.10	Gute Namen, schlechte Namen	118
2.4	Blöcke, Initialisierung und Sichtbarkeit	119
2.4.1	Gruppieren von Anweisungen mit Blöcken	119
2.4.2	Initialisierung von lokalen Variablen	120
2.4.3	Sichtbarkeit und Gültigkeitsbereich	120
2.5	Ausdrücke, Operanden und Operatoren	122
2.5.1	Ausdrücke	122
2.5.2	Arithmetische Operatoren	123
2.5.3	Unäres Minus und Plus	125
2.5.4	Zuweisung mit Operation	126
2.5.5	Präfix- oder Postfix-Inkrement und -Dekrement	127
2.5.6	Die relationalen Operatoren und die Gleichheitsoperatoren	129
2.5.7	Logische Operatoren: Nicht, Und, Oder, Xor	130
2.5.8	Der Rang der Operatoren in der Auswertungsreihenfolge	132
2.5.9	Die Typanpassung (das Casting)	135
2.5.10	Überladenes Plus für Strings	138
2.5.11	Operator vermisst *	139
2.6	Bedingte Anweisungen oder Fallunterscheidungen	140
2.6.1	Die if-Anweisung	140
2.6.2	Die Alternative mit einer if-else-Anweisung wählen	142
2.6.3	Der Bedingungsoperator	144
2.6.4	Die switch-Anweisung bietet die Alternative	146
2.7	Schleifen	149
2.7.1	Die while-Schleife	150
2.7.2	Die do-while-Schleife	151
2.7.3	Die for-Schleife	152
2.7.4	Schleifenbedingungen und Vergleiche mit ==	155
2.7.5	Ausbruch planen mit break und Wiedereinstieg mit »continue«	157
2.7.6	»break« und »continue« mit Marken *	160
2.8	Methoden einer Klasse	163
2.8.1	Bestandteil einer Methode	163
2.8.2	Signatur-Beschreibung in der Java-API	165
2.8.3	Aufruf einer Methode	166
2.8.4	Methoden ohne Parameter deklarieren	166
2.8.5	Statische Methoden (Klassenmethoden)	167
2.8.6	Parameter, Argument und Wertübergabe	168
2.8.7	Methoden vorzeitig mit »return« beenden	170
2.8.8	Nicht erreichbarer Quellcode bei Methoden	170
2.8.9	Rückgabewerte	171
2.8.10	Methoden überladen	174

2.8.11	Vorgegebener Wert für nicht aufgeführte Argumente *	176
2.8.12	Finale lokale Variablen	176
2.8.13	Rekursive Methoden *	178
2.8.14	Die Türme von Hanoi *	181
2.9	Zum Weiterlesen	183

3	Klassen und Objekte	185
3.1	Objektorientierte Programmierung (OOP)	185
3.1.1	Warum überhaupt OOP?	185
3.1.2	Denk ich an Java, denk ich an Wiederverwendbarkeit	186
3.2	Eigenschaften einer Klasse	187
3.2.1	Die Klasse »Point«	187
3.3	Die UML (Unified Modeling Language) *	188
3.3.1	Hintergrund und Geschichte zur UML	188
3.3.2	Wichtige Diagrammtypen der UML	189
3.3.3	UML-Werkzeuge	190
3.4	Neue Objekte erzeugen	191
3.4.1	Ein Exemplar einer Klasse mit dem new-Operator anlegen	191
3.4.2	Garbage-Collector (GC) – Es ist dann mal weg	193
3.4.3	Deklaren von Referenzvariablen	193
3.4.4	Zugriff auf Variablen und Methoden mit dem ».«	194
3.4.5	Konstruktoren nutzen	196
3.5	Mit Referenzen arbeiten, Identität und Gleichheit	197
3.5.1	Die null-Referenz	197
3.5.2	null-Referenzen testen	198
3.5.3	Zuweisungen bei Referenzen	199
3.5.4	Methoden mit nicht-primitiven Parametern	200
3.5.5	Identität von Objekten	202
3.5.6	Gleichheit und die Methode »equals()«	202
3.6	Kompilationseinheiten, Imports und Pakete schnüren	204
3.6.1	Volle Qualifizierung und import-Deklaration	205
3.6.2	Mit import p1.p2.* alle Typen eines Pakets erreichen	205
3.6.3	Hierarchische Strukturen über Pakete	206
3.6.4	Die package-Deklaration	207
3.6.5	Unbenanntes Paket (default package)	207
3.6.6	Klassen mit gleichen Namen in unterschiedlichen Paketen *	208
3.6.7	Compilationseinheit (Compilation Unit)	208
3.6.8	Statischer Import	209
3.6.9	Eine Verzeichnisstruktur für eigene Projekte *	210
3.7	Arrays	210
3.7.1	Deklaration von Arrays	211
3.7.2	Arrays mit Inhalt	211

3.7.3	Die Länge eines Arrays über das Attribut <code>length</code> auslesen	211
3.7.4	Zugriff auf die Elemente über den Index	212
3.7.5	Array-Objekte mit <code>new</code> erzeugen	213
3.7.6	Fehler bei Arrays	214
3.7.7	Die erweiterte <code>for</code> -Schleife	215
3.7.8	Arrays mit nicht-primitiven Elementen	216
3.7.9	Mehrdimensionale Arrays *	217
3.7.10	Vorinitialisierte Arrays *	220
3.7.11	Mehrere Rückgabewerte *	222
3.7.12	Methode mit variabler Argumentanzahl (<code>Vararg</code>)	222
3.7.13	Klonen kann sich lohnen – Arrays vermehren *	224
3.7.14	Feldinhalte kopieren *	224
3.7.15	Die Klasse »Arrays« zum Vergleichen, Füllen und Suchen nutzen	225
3.8	Der Einstiegspunkt für das Laufzeitsystem: » <code>main()</code> «	231
3.8.1	Kommandozeilenargumente verarbeiten	232
3.8.2	Der Rückgabewert von » <code>main()</code> « und » <code>System.exit()</code> «	232
3.9	Annotationen	233
3.9.1	Annotationstypen <code>@Override</code> , <code>@Deprecated</code> , <code>@SuppressWarnings</code>	234
3.10	Zum Weiterlesen	236

4 Der Umgang mit Zeichenketten 37

4.1	Einzelne Zeichen behandeln	237
4.1.1	Von ASCII über ISO-8859-1 zu Unicode	237
4.1.2	Die <code>Character</code> -Klasse	240
4.2	Strings und deren Anwendung	243
4.2.1	String-Literale als String-Objekte für konstante Zeichenketten ...	246
4.2.2	String-Länge und Test auf Leerstring	246
4.2.3	Nach enthaltenen Zeichen und Zeichenfolgen suchen	247
4.2.4	Gut, dass wir verglichen haben	249
4.2.5	Phonetische Vergleiche	252
4.2.6	String-Teile extrahieren	253
4.2.7	Strings anhängen, Groß-/Kleinschreibung und Leerraum	256
4.2.8	Suchen und ersetzen	259
4.2.9	String-Objekte mit Konstruktoren neu anlegen *	261
4.3	Konvertieren zwischen Primitiven und Strings	265
4.3.1	Unterschiedliche Typen in String-Repräsentationen konvertieren	265
4.3.2	Stringinhalt in primitiven Wert konvertieren	266
4.3.3	Unterschiedliche Ausgabeformate (Binär, Hex, Oktal) *	267
4.4	Veränderbare Zeichenketten mit <code>StringBuilder</code> und <code>StringBuffer</code>	270
4.4.1	Anlegen von <code>StringBuilder</code> / <code>StringBuffer</code> -Objekten	271

4.4.2	StringBuilder/StringBuffer in andere Zeichenkettenformate konvertieren	271
4.4.3	Daten anhängen	272
4.4.4	Zeichen(folgen) setzen, erfragen, löschen und umdrehen	273
4.4.5	Länge und Kapazität eines StringBuilder/ StringBuffer-Objekts *	274
4.4.6	Vergleichen von String mit StringBuilder und StringBuffer	275
4.4.7	»hashCode()« bei StringBuilder/StringBuffer *	276
4.5	CharSequence als Basistyp *	277
4.6	Sprachabhängiges Vergleichen und Normalisierung *	279
4.6.1	Die Klasse »Collator«	279
4.6.2	Effiziente interne Speicherung für die Sortierung	282
4.6.3	Normalisierung	283
4.7	Reguläre Ausdrücke	284
4.7.1	Arbeiten mit der Fassade: String#matches()	284
4.7.2	Die Klassen »Pattern« und »Matcher«	286
4.7.3	Finden und nicht matchen	291
4.7.4	Gierige und nicht gierige Operatoren *	292
4.7.5	Mit MatchResult alle Ergebnisse einsammeln *	293
4.7.6	Suchen und Ersetzen mit Mustern	294
4.8	Zerlegen von Zeichenketten	295
4.8.1	Splitten von Zeichenketten mit »split()«	296
4.8.2	Die Klasse »Scanner«	297
4.8.3	Die Klasse »StringTokenizer« *	302
4.8.4	BreakIterator als Zeichen-, Wort-, Zeilen- und Satztrenner *	304
4.9	Zeichenkodierungen, XML/HTML-Entitys, Base64 *	307
4.9.1	Unicode und 8-Bit-Abbildungen	307
4.9.2	Konvertieren mit »OutputStreamWriter«-Klassen	308
4.9.3	Das Paket »java.nio.charset«	309
4.9.4	XML/HTML-Entitys ausmaskieren	309
4.9.5	Base64-Kodierung	310
4.10	Ausgaben formatieren	311
4.10.1	Formatieren und Ausgeben mit »format()«	312
4.10.2	Die Formatter-Klasse *	317
4.10.3	Formatieren mit Masken *	318
4.10.4	Format-Klassen	319
4.10.5	Zahlen, Prozente und Währungen mit »NumberFormat« und »DecimalFormat« formatieren *	321
4.11	Zum Weiterlesen	325

5	Eigene Klassen schreiben	327
5.1	Eigene Klassen mit Eigenschaften deklarieren	327
5.1.1	Attribute deklarieren	327
5.1.2	Methoden deklarieren	329
5.1.3	Die this-Referenz	333
5.2	Privatsphäre und Sichtbarkeit	336
5.2.1	Für die Öffentlichkeit: public	336
5.2.2	Kein Public Viewing – Passwörter sind privat	337
5.2.3	Wieso nicht freie Methoden und Variablen für alle?	338
5.2.4	Privat ist nicht ganz privat: Es kommt darauf an, wer's sieht *	338
5.2.5	Zugriffsmethoden für Attribute deklarieren	339
5.2.6	Setter und Getter nach der JavaBeans-Spezifikation	340
5.2.7	Paketsichtbar	342
5.2.8	Zusammenfassung zur Sichtbarkeit	343
5.3	Statische Methoden und statische Attribute	345
5.3.1	Warum statische Eigenschaften sinnvoll sind	346
5.3.2	Statische Eigenschaften mit static	346
5.3.3	Statische Eigenschaften über Referenzen nutzen? *	348
5.3.4	Warum die Groß- und Kleinschreibung wichtig ist *	348
5.3.5	Statische Variablen zum Datenaustausch *	349
5.3.6	Statische Eigenschaften und Objekteigenschaften *	350
5.4	Konstanten und Aufzählungen	351
5.4.1	Konstanten über öffentliche statische finale Variablen	351
5.4.2	Typ(un)sichere Aufzählungen *	352
5.4.3	Aufzählungen mit »enum«	353
5.5	Objekte anlegen und zerstören	356
5.5.1	Konstruktoren schreiben	356
5.5.2	Der vorgegebene Konstruktor (engl. »default constructor«)	358
5.5.3	Parametrisierte und überladene Konstruktoren	360
5.5.4	Copy-Konstruktor	362
5.5.5	Einen anderen Konstruktor der gleichen Klasse mit »this()« aufrufen	363
5.5.6	Ihr fehlt uns nicht – der Garbage-Collector	366
5.5.7	Private Konstruktoren, Utility-Klassen, Singleton, Fabriken	367
5.6	Klassen- und Objektinitialisierung *	369
5.6.1	Initialisierung von Objektvariablen	370
5.6.2	Statische Blöcke als Klasseninitialisierer	371
5.6.3	Initialisierung von Klassenvariablen	372
5.6.4	Eincompilierte Belegungen der Klassenvariablen	373
5.6.5	Exemplarinitialisierer (Instanzinitialisierer)	374
5.6.6	Finale Werte im Konstruktor und in statischen Blöcken setzen ...	376

5.7	Assoziationen zwischen Objekten	378
5.7.1	Unidirektionale 1:1-Beziehung	379
5.7.2	Bidirektionale 1:1-Beziehungen	380
5.7.3	Unidirektionale 1:n-Beziehung	381
5.8	Vererbung	383
5.8.1	Vererbung in Java	384
5.8.2	Spielobjekte modellieren	384
5.8.3	Die implizite Basisklasse »java.lang.Object«	386
5.8.4	Einfach- und Mehrfachvererbung *	386
5.8.5	Die Sichtbarkeit »protected«	386
5.8.6	Konstruktoren in der Vererbung und »super()«	387
5.9	Typen in Hierarchien	392
5.9.1	Automatische und explizite Typanpassung	392
5.9.2	Das Substitutionsprinzip	394
5.9.3	Typen mit dem binären Operator »instanceof« testen	396
5.10	Methoden überschreiben	398
5.10.1	Methoden in Unterklassen mit neuem Verhalten ausstatten	398
5.10.2	Mit »super« an die Eltern	401
5.10.3	Finale Klassen und finale Methoden	403
5.10.4	Kovariante Rückgabetypen	404
5.10.5	Array-Typen und Kovarianz *	405
5.11	Dynamisches Binden	406
5.11.1	Gebunden an »toString()«	407
5.11.2	Implementierung von »System.out.println(Object)«	408
5.11.3	Nicht dynamisch gebunden bei privaten, statischen und finalen Methoden	409
5.11.4	Dynamisch gebunden auch bei Konstruktoraufrufen *	411
5.11.5	Eine letzte Spielerei mit Javas dynamischer Bindung und überschatteten Attributen *	413
5.12	Abstrakte Klassen und abstrakte Methoden	414
5.12.1	Abstrakte Klassen	414
5.12.2	Abstrakte Methoden	415
5.13	Schnittstellen	419
5.13.1	Schnittstellen deklarieren	419
5.13.2	Implementieren von Schnittstellen	420
5.13.3	Markierungsschnittstellen *	422
5.13.4	Ein Polymorphie-Beispiel mit Schnittstellen	422
5.13.5	Die Mehrfachvererbung bei Schnittstellen *	423
5.13.6	Keine Kollisionsgefahr bei Mehrfachvererbung *	427
5.13.7	Erweitern von Interfaces – Subinterfaces	428
5.13.8	Konstantendeklarationen bei Schnittstellen	428
5.13.9	Initialisierung von Schnittstellenkonstanten *	431
5.13.10	Abstrakte Klassen und Schnittstellen im Vergleich	434

5.14	Dokumentationskommentare mit Javadoc	435
5.14.1	Einen Dokumentationskommentar setzen	435
5.14.2	Mit dem Werkzeug javadoc eine Dokumentation erstellen	437
5.14.3	HTML-Tags in Dokumentationskommentaren *	437
5.14.4	Generierte Dateien	438
5.14.5	Dokumentationskommentare im Überblick *	438
5.14.6	Javadoc und Doclets *	440
5.14.7	Veraltete (deprecated) Typen und Eigenschaften	440

6 Exceptions

6.1	Problembereiche einzäunen	443
6.1.1	Exceptions in Java mit try und catch	443
6.1.2	Eine NumberFormatException auffangen	444
6.1.3	Ablauf einer Ausnahmesituation	446
6.1.4	Eigenschaften vom Exception-Objekt	446
6.1.5	Wiederholung abgebrochener Bereiche *	448
6.1.6	Mehrere Ausnahmen auffangen	448
6.1.7	throws im Methodenkopf angeben	451
6.1.8	Abschlussbehandlung mit »finally«	452
6.2	Die Klassenhierarchie der Fehler	456
6.2.1	Die Exception-Hierarchie	456
6.2.2	Oberausnahmen auffangen	457
6.2.3	Alles geht als Exception durch	458
6.2.4	Zusammenfassen gleicher catch-Blöcke	460
6.3	RuntimeException muss nicht aufgefangen werden	461
6.3.1	Beispiele für RuntimeException-Klassen	461
6.3.2	Kann man abfangen, muss man aber nicht	462
6.4	Harte Fehler: Error *	462
6.5	Auslösen eigener Exceptions	463
6.5.1	Mit throw Ausnahmen auslösen	463
6.5.2	Vorhandene Runtime-Fehlertypen kennen und nutzen	464
6.5.3	Parameter testen und gute Fehlermeldungen	466
6.5.4	Neue Exception-Klassen deklarieren	468
6.5.5	Eigene Ausnahmen als Unterklassen von Exception oder RuntimeException?	469
6.5.6	Abfangen und weiterleiten *	471
6.5.7	Geschachtelte Ausnahmen *	472
6.6	Assertions *	474
6.6.1	Assertions in eigenen Programmen nutzen	475
6.6.2	Assertions aktivieren	476

7	Generics	477
7.1	Einführung in Java Generics	477
7.1.1	Mensch versus Maschine: Typprüfung des Compilers und der Laufzeitumgebung	477
7.1.2	Taschen	478
7.1.3	Generische Typen deklarieren	480
7.1.4	Generics nutzen	481
7.1.5	Generische Schnittstellen	483
7.1.6	Generische Methoden/Konstruktoren und Typ-Inferenz	485
8	Außere innere Klassen	489
8.1	Geschachtelte (innere) Klassen, Schnittstellen, Aufzählungen	489
8.1.1	Statische innere Klassen und Schnittstellen	490
8.1.2	Mitglieds- oder Elementklassen	491
8.1.3	Lokale Klassen	495
8.1.4	Anonyme innere Klassen	496
8.1.5	Zugriff auf lokale Variablen aus lokalen inneren und anonymen Klassen *	499
8.1.6	»this« und Vererbung *	500
9	Besondere Klassen der Java SE	503
9.1	Vergleichen von Objekten	503
9.1.1	Natürlich geordnet oder nicht?	503
9.1.2	Die Schnittstelle Comparable	504
9.1.3	Die Schnittstelle Comparator	504
9.1.4	Rückgabewerte kodieren die Ordnung	505
9.2	Wrapper-Klassen und Autoboxing	507
9.2.1	Wrapper-Objekte erzeugen	508
9.2.2	Konvertierungen in eine String-Repräsentation	509
9.2.3	Die Basisklasse Number für numerische Wrapper-Objekte	510
9.2.4	Vergleiche durchführen mit »compare()«, »compareTo()« und »equals()«	512
9.2.5	Die Klasse »Integer«	514
9.2.6	Die Klassen »Double« und »Float« für Fließkommazahlen	515
9.2.7	Die Boolean-Klasse	516
9.2.8	Autoboxing: Boxing und Unboxing	518
9.3	Object ist die Mutter aller Klassen	521
9.3.1	Klassenobjekte	522
9.3.2	Objektidentifikation mit »toString()«	522
9.3.3	Objektgleichheit mit »equals()« und Identität	524
9.3.4	Klonen eines Objekts mit »clone()« *	528

9.3.5	Hashcodes über »hashCode()« liefern *	531
9.3.6	Aufräumen mit »finalize()« *	536
9.3.7	Synchronisation *	538
9.4	Die Spezial-Oberklasse »Enum«	538
9.4.1	Methoden auf Enum-Objekten	539
9.4.2	»Enum« mit eigenen Konstruktoren und Methoden *	542
9.5	Erweitertes »for« und »Iterable«	545
9.5.1	Die Schnittstelle »Iterable«	545
9.5.2	Einen eigenen Iterable implementieren *	545
9.6	Zum Weiterlesen	547

10 Architektur, Design und angewandte Objektorientierung

549

10.1	Architektur, Design und Implementierung	549
10.2	Design-Pattern (Entwurfsmuster)	550
10.2.1	Motivation für Design-Pattern	550
10.2.2	Das Beobachter-Pattern (Observer/Observable)	551
10.2.3	Ereignisse über Listener	556
10.3	JavaBean	560
10.3.1	Properties (Eigenschaften)	561
10.3.2	Einfache Eigenschaften	561
10.3.3	Indizierte Eigenschaften	562
10.3.4	Gebundene Eigenschaften und PropertyChangeListener	562
10.3.5	Veto-Eigenschaften – dagegen!	565
10.3.6	Ein POJO (Plain Old Java Object) ohne technische Abhängigkeiten	569
10.4	Zum Weiterlesen	571

11 Die Klassenbibliothek

573

11.1	Die Java-Klassenphilosophie	573
11.1.1	Übersicht über die Pakete der Standardbibliothek	573
11.2	Klassenlader (Class Loader)	575
11.2.1	Woher die kleinen Klassen kommen	575
11.2.2	Setzen des Klassenpfades	577
11.2.3	Die wichtigsten drei Typen von Klassenladern	578
11.2.4	Die Klasse »java.lang.ClassLoader« *	578
11.2.5	Hot Deployment mit dem URL-Classloader *	580
11.2.6	Das Verzeichnis jre/lib/endorsed *	582
11.3	Die Utility-Klasse System und Properties	583
11.3.1	Systemeigenschaften der Java-Umgebung	584
11.3.2	line.separator	585
11.3.3	Property von der Konsole aus setzen *	586
11.3.4	Umgebungsvariablen des Betriebssystems *	587

11.4	Einfache Benutzereingaben	588
11.4.1	Grafischer Eingabedialog über JOptionPane	589
11.4.2	Geschützte Passwort-Eingaben mit der Klasse »Console« *	590
11.5	Ausführen externer Programme *	591
11.5.1	»ProcessBuilder« und Prozesskontrolle mit Process	591
11.5.2	Einen Browser, E-Mail-Client oder Editor aufrufen	595
11.6	Zum Weiterlesen	596

12 Bits und Bytes und Mathematisches	597
---	------------

12.1	Bits und Bytes *	597
12.1.1	Die Bit-Operatoren Komplement, Und, Oder und Xor	597
12.1.2	Repräsentation ganzer Zahlen in Java – das Zweierkomplement	599
12.1.3	Auswirkung der Typanpassung auf die Bitmuster	600
12.1.4	»byte« als vorzeichenlosen Datentyp nutzen	602
12.1.5	Die Verschiebeoperatoren	603
12.1.6	Ein Bit setzen, löschen, umdrehen und testen	605
12.1.7	Bit-Methoden der Integer- und Long-Klasse	605
12.2	Fließkommaarithmetik in Java	606
12.2.1	Spezialwerte für Unendlich, Null, NaN	607
12.2.2	Standard-Notation und wissenschaftliche Notation bei Fließkommazahlen *	609
12.2.3	Mantisse und Exponent *	610
12.3	Die Eigenschaften der Klasse »Math«	611
12.3.1	Attribute	612
12.3.2	Absolutwerte und Vorzeichen	612
12.3.3	Maximum/Minimum	613
12.3.4	Runden von Werten	613
12.3.5	Wurzel und Exponentialmethoden	615
12.3.6	Der Logarithmus *	616
12.3.7	Rest der ganzzahligen Division *	617
12.3.8	Winkelmethode *	618
12.3.9	Zufallszahlen	619
12.4	Mathe bitte strikt *	619
12.4.1	Strikte Fließkommaberechnungen mit strictfp	619
12.4.2	Die Klassen »Math« und »StrictMath«	620
12.5	Die Random-Klasse	620
12.5.1	Objekte aufbauen mit dem Samen	621
12.5.2	Zufallszahlen erzeugen	621
12.5.3	Pseudo-Zufallszahlen in der Normalverteilung *	622
12.6	Große Zahlen *	622
12.6.1	Die Klasse »BigInteger«	622

12.6.2	Methoden von »BigInteger«	625
12.6.3	Ganz lange Fakultäten	627
12.6.4	Große Fließkommazahlen mit BigDecimal	628
12.6.5	Mit MathContext komfortabel die Rechengenauigkeit setzen	630
12.7	Zum Weiterlesen	631

13 Datenstrukturen und Algorithmen



13.1	Datenstrukturen und die Collection-API	633
13.1.1	Designprinzip mit Schnittstellen, abstrakten und konkreten Klassen	634
13.1.2	Die Basis-Schnittstellen Collection und Map	634
13.1.3	Das erste Programm mit Container-Klassen	634
13.1.4	Die Schnittstelle Collection und Kernkonzepte	636
13.1.5	Schnittstellen, die Collection erweitern, und Map	639
13.1.6	Konkrete Container-Klassen	641
13.1.7	Welche Container-Klasse nehmen?	642
13.1.8	Generische Datentypen in der Collection-API	642
13.1.9	Die Schnittstelle »Iterable« und das erweiterte »for«	644
13.2	Mit einem Iterator durch die Daten wandern	644
13.2.1	Die Schnittstellen Enumeration und Iterator	645
13.2.2	Iteratoren von Sammlungen und das erweiterte »for«	647
13.2.3	Fail-Fast-Iterator und die ConcurrentModificationException	649
13.3	Listen	650
13.3.1	Auswahlkriterium ArrayList oder LinkedList	651
13.3.2	Die Schnittstelle List	651
13.3.3	ListIterator *	656
13.3.4	ArrayList	657
13.3.5	LinkedList	660
13.3.6	Der Feld-Adapter »Arrays.asList()«	661
13.3.7	»toArray()« von Collection verstehen – die Gefahr einer Falle erkennen	662
13.3.8	Primitive Elemente in den Collection-Datenstrukturen	665
13.4	Datenstrukturen mit Ordnung	665
13.4.1	Algorithmen mit Such- und Sortiermöglichkeiten	665
13.4.2	Den größten und kleinsten Wert einer Collection finden	666
13.4.3	Sortieren	667
13.5	Mengen (Sets)	670
13.5.1	HashSet	672
13.5.2	TreeSet – die Menge durch Bäume	673
13.5.3	LinkedHashSet	676
13.6	Stack (Kellerspeicher, Stapel)	676
13.6.1	Die Methoden von »Stack«	677

13.6.2	Ein »Stack« ist ein »Vector« – aha!	678
13.7	Queues (Schlangen) und Deques	678
13.7.1	Die Schnittstelle »Queue«	678
13.7.2	Blockierende Queues und Prioritätswarteschlangen	680
13.7.3	»Deque«-Klassen	680
13.8	Assoziative Speicher	681
13.8.1	Die Klassen »HashMap« und »TreeMap«	681
13.8.2	Einfügen und Abfragen der Datenstruktur	683
13.8.3	Über die Bedeutung von »equals()«, »hashCode()«	685
13.8.4	IdentityHashMap	687
13.8.5	Das Problem von veränderten Elementen	687
13.8.6	Aufzählungen und Ansichten des Assoziativspeichers	688
13.8.7	Der Gleichheitstest, Hash-Wert und Klon einer Hash-Tabelle*	690
13.8.8	Die Arbeitsweise einer Hash-Tabelle *	691
13.9	Die Properties-Klasse	693
13.9.1	Properties setzen und lesen	694
13.9.2	Properties verketteten	694
13.9.3	Hierarchische Eigenschaften	695
13.9.4	Eigenschaften ausgeben *	695
13.9.5	Properties laden und speichern	696
13.10	Algorithmen in Collections	697
13.10.1	Nicht-änderbare Datenstrukturen	698
13.10.2	Null Object Pattern und leere Sammlungen zurückgeben	698
13.10.3	Mit der Halbierungssuche nach Elementen fahnden	701
13.10.4	Ersetzen, Kopieren, Füllen, Umdrehen, Rotieren, Durchmischen *	703
13.10.5	Häufigkeit eines Elements *	704
13.10.6	nCopies() *	704
13.10.7	Singletons *	705
13.11	Synchronisation der Datenstrukturen	706
13.11.1	Lock-free-Algorithmen aus java.util.concurrent	706
13.11.2	Wrapper zur Synchronisation	707
13.11.3	»CopyOnWriteArrayList« und »CopyOnWriteArraySet«	708
13.12	Die Klasse »BitSet« für Bitmengen *	708
13.12.1	Ein »BitSet« anlegen, füllen und erfragen	708
13.12.2	Mengenorientierte Operationen	709
13.12.3	Methodenübersicht	710
13.12.4	Primzahlen in einem »BitSet« verwalten	711
13.13	Zum Weiterlesen	712

14 Threads und nebenläufige Programmierung	713
14.1 Nebenläufigkeit	713
14.1.1 Threads und Prozesse	713
14.1.2 Wie parallele Programme die Geschwindigkeit steigern können	715
14.1.3 Was Java für Nebenläufigkeit alles bietet	716
14.2 Threads erzeugen	717
14.2.1 Threads über die Schnittstelle Runnable implementieren	717
14.2.2 Thread mit Runnable starten	718
14.2.3 Die Klasse »Thread« erweitern	719
14.3 Thread-Eigenschaften und -Zustände	722
14.3.1 Der Name eines Threads	722
14.3.2 Wer bin ich?	722
14.3.3 Die Zustände eines Threads *	723
14.3.4 Schläfer gesucht	723
14.3.5 Mit »yield()« auf Rechenzeit verzichten	725
14.3.6 Der Thread als Dämon	725
14.3.7 Das Ende eines Threads	727
14.3.8 Einen Thread höflich mit Interrupt beenden	728
14.3.9 »UncaughtExceptionHandler« für unbehandelte Ausnahmen	730
14.3.10 Der »stop()« von außen und die Rettung mit ThreadDeath *	731
14.3.11 Ein Rendezvous mit »join()« *	732
14.3.12 Arbeit niederlegen und wieder aufnehmen *	734
14.3.13 Priorität *	735
14.4 Der Ausführer (Executor) kommt	736
14.4.1 Die Schnittstelle »Executor«	736
14.4.2 Die Thread-Pools	738
14.4.3 Threads mit Rückgabe über Callable	739
14.4.4 Mehrere Callable abarbeiten	742
14.4.5 Mit ScheduledExecutorService wiederholende Ausgaben und Zeitsteuerungen	743
14.5 Synchronisation über kritische Abschnitte	743
14.5.1 Gemeinsam genutzte Daten	744
14.5.2 Probleme beim gemeinsamen Zugriff und kritische Abschnitte ...	744
14.5.3 Punkte parallel initialisieren	746
14.5.4 »i++« sieht atomar aus, ist es aber nicht *	747
14.5.5 Kritische Abschnitte schützen	748
14.5.6 Schützen mit ReentrantLock	750
14.5.7 Synchronisieren mit »synchronized«	756
14.5.8 Synchronized-Methoden der Klasse »StringBuffer« *	757
14.5.9 Mit synchronized synchronisierte Blöcke	758
14.5.10 Dann machen wir doch gleich alles synchronisiert!	759

14.5.11	Lock-Freigabe im Fall von Exceptions	760
14.5.12	Deadlocks	761
14.5.13	Mit »synchronized« nachträglich synchronisieren *	763
14.5.14	Monitore sind reentrant – gut für die Geschwindigkeit *	764
14.5.15	Synchronisierte Methodenaufrufe zusammenfassen *	765
14.6	Synchronisation über Warten und Benachrichtigen	765
14.6.1	Die Schnittstelle »Condition«	766
14.6.2	It's Disco-Time *	770
14.6.3	Warten mit »wait()« und Aufwecken mit »notify()« *	774
14.6.4	Falls der Lock fehlt: IllegalMonitorStateException *	775
14.7	Zeitgesteuerte Abläufe	777
14.7.1	Die Klassen »Timer« und »TimerTask«	777
14.7.2	Job-Scheduler Quartz	778
14.8	Einen Abbruch der virtuellen Maschine erkennen	779
14.9	Zum Weiterlesen	780

15	Raum und Zeit	781
15.1	Weltzeit *	781
15.2	Wichtige Datum-Klassen im Überblick	782
15.2.1	Der 1.1.1970	782
15.2.2	System.currentTimeMillis()	783
15.2.3	Einfach Zeitumrechnungen durch »TimeUnit«	783
15.3	Sprachen der Länder	784
15.3.1	Sprachen und Regionen über Locale-Objekte	784
15.4	Internationalisierung und Lokalisierung	787
15.4.1	ResourceBundle-Objekte und Ressource-Dateien	788
15.4.2	Ressource-Dateien zur Lokalisierung	788
15.4.3	Die Klasse »ResourceBundle«	789
15.4.4	Ladestrategie für ResourceBundle-Objekte	790
15.5	Die Klasse »Date«	791
15.5.1	Objekte erzeugen und Methoden nutzen	791
15.5.2	Date-Objekte nicht immutable	793
15.6	Calendar und GregorianCalendar	793
15.6.1	Die abstrakte Klasse »Calendar«	794
15.6.2	Der gregorianische Kalender	795
15.6.3	»Calendar« nach »Date« und Millisekunden fragen	797
15.6.4	Ostertage *	798
15.6.5	Abfragen und Setzen von Datumselementen über Feldbezeichner	799
15.7	Formatieren und Parsen von Datumsangaben	805
15.7.1	Ausgaben mit »printf()«	805

15.7.2	Mit »DateFormat« und »SimpleDateFormat« formatieren	806
15.7.3	Parsen von Datumswerten	811
15.8	Zum Weiterlesen	813

16 Dateien, Verzeichnisse und Dateisysteme 816

16.1	Datei und Verzeichnis	816
16.1.1	Dateien und Verzeichnisse mit der Klasse »File«	816
16.1.2	Verzeichnis oder Datei? Existiert es?	818
16.1.3	Verzeichnis- und Dateieigenschaften/-attribute	819
16.1.4	Umbenennen und Verzeichnisse anlegen	821
16.1.5	Verzeichnisse listen und Dateien filtern	822
16.1.6	Dateien berühren, neue Dateien anlegen, temporäre Dateien	825
16.1.7	Dateien und Verzeichnisse löschen	826
16.1.8	Verzeichnisse nach Dateien iterativ durchsuchen *	827
16.1.9	Wurzelverzeichnis, Laufwerksnamen, Plattenspeicher *	829
16.1.10	URL- und URI-Objekte aus einem File-Objekt ableiten *	831
16.1.11	Mit Locking Dateien sperren *	831
16.2	Dateien mit wahlfreiem Zugriff	832
16.2.1	Ein »RandomAccessFile« zum Lesen und Schreiben öffnen	832
16.2.2	Aus dem »RandomAccessFile« lesen	833
16.2.3	Schreiben mit »RandomAccessFile«	835
16.2.4	Die Länge des »RandomAccessFile«	836
16.2.5	Hin und her in der Datei	836
16.3	Zum Weiterlesen	837

17 Datenströme 839

17.1	Stream-Klassen und Reader/Writer am Beispiel von Dateien	839
17.1.1	Mit dem FileWriter Texte in Dateien schreiben	840
17.1.2	Zeichen mit der Klasse »FileReader« lesen	841
17.1.3	Kopieren mit »FileOutputStream« und »FileInputStream«	842
17.1.4	Das FileDescriptor-Objekt *	845
17.2	Basisklassen für die Ein-/Ausgabe	845
17.2.1	Die abstrakten Basisklassen	846
17.2.2	Übersicht über Ein-/Ausgabeklassen	846
17.2.3	Die abstrakte Basisklasse »OutputStream«	848
17.2.4	Die Schnittstellen »Closeable« und »Flushable«	849
17.2.5	Ein Datenschluckler *	850
17.2.6	Die abstrakte Basisklasse »InputStream«	851
17.2.7	Ressourcen aus dem Klassenpfad und aus Jar-Archiven laden	852
17.2.8	Ströme mit SequenceInputStream zusammensetzen *	852
17.2.9	Die abstrakte Basisklasse »Writer«	854

17.2.10	Die Schnittstelle »Appendable« *	856
17.2.11	Die abstrakte Basisklasse »Reader«	856
17.3	Formatierte Textausgaben	859
17.3.1	Die Klassen »PrintWriter« und »PrintStream«	859
17.3.2	»System.out«, »System.err« und »System.in«	864
17.4	Schreiben und Lesen aus Strings und Byte-Feldern	866
17.4.1	Mit dem »StringWriter« ein String-Objekt füllen	866
17.4.2	CharArrayWriter	867
17.4.3	»StringReader« und »CharArrayReader«	868
17.4.4	Mit »ByteArrayOutputStream« in ein Byte-Feld schreiben	869
17.4.5	Mit »ByteArrayInputStream« aus einem Byte-Feld lesen	870
17.5	Datenströme filtern und verketten	870
17.5.1	Streams als Filter verketten (verschalen)	871
17.5.2	Gepufferte Ausgaben mit »BufferedWriter«/ »BufferedOutputStream«	871
17.5.3	Gepufferte Eingaben mit »BufferedReader«/ »BufferedInputStream«	873
17.5.4	»LineNumberReader« zählt automatisch Zeilen mit *	875
17.5.5	Daten mit der Klasse »PushbackReader« zurücklegen *	876
17.5.6	DataOutputStream/DataInputStream *	879
17.5.7	Basisklassen für Filter *	879
17.5.8	Die Basisklasse »FilterWriter« *	880
17.5.9	Ein LowerCaseWriter *	881
17.5.10	Eingaben mit der Klasse »FilterReader« filtern *	882
17.5.11	Anwendungen für »FilterReader« und »FilterWriter« *	883
17.6	Vermittler zwischen Byte-Streams und Unicode-Strömen	889
17.6.1	Datenkonvertierung durch den »OutputStreamWriter«	889
17.6.2	Automatische Konvertierungen mit dem »InputStreamReader«	890
17.7	Kommunikation zwischen Threads mit Pipes *	891
17.7.1	»PipedOutputStream« und »PipedInputStream«	891
17.7.2	»PipedWriter« und »PipedReader«	893
17.8	Datenkompression *	895
17.8.1	Java-Unterstützung beim Komprimieren	896
17.8.2	Datenströme komprimieren	896
17.8.3	Zip-Archive	900
17.8.4	Jar-Archive	906
17.9	Prüfsummen	906
17.9.1	Die Schnittstelle Checksum	906
17.9.2	Die Klasse »CRC32«	907
17.9.3	Die Adler32-Klasse	909
17.10	Persistente Objekte und Serialisierung	909

17.10.1	Objekte mit der Standard-Serialisierung speichern und lesen	911
17.10.2	Zwei einfache Anwendungen der Serialisierung *	913
17.10.3	Die Schnittstelle »Serializable«	914
17.10.4	Nicht serialisierbare Attribute aussparen	916
17.10.5	Das Abspeichern selbst in die Hand nehmen	917
17.10.6	Tiefe Objektkopien *	921
17.10.7	Versionenverwaltung und die SUID	922
17.10.8	Wie die »ArrayList« serialisiert *	924
17.10.9	Probleme mit der Serialisierung	925
17.11	Alternative Datenaustauschformate	926
17.11.1	Serialisieren in XML-Dateien	926
17.11.2	XML-Serialisierung von JavaBeans mit JavaBeans Persistence *	926
17.11.3	Open-Source Bibliothek XStream *	929
17.12	Tokenizer *	929
17.12.1	StreamTokenizer	929
17.13	Zum Weiterlesen	932

18 Die extensible Markup Language (XML)	938
--	------------

18.1	Auszeichnungssprachen	933
18.1.1	Die Standard Generalized Markup Language (SGML)	933
18.1.2	Extensible Markup Language (XML)	934
18.2	Eigenschaften von XML-Dokumenten	934
18.2.1	Elemente und Attribute	934
18.2.2	Beschreibungssprache für den Aufbau von XML-Dokumenten ...	937
18.2.3	Schema – eine Alternative zu DTD	940
18.2.4	Namensraum (Namespace)	942
18.2.5	XML-Applikationen *	943
18.3	Die Java-APIs für XML	944
18.3.1	Das Document Object Model (DOM)	945
18.3.2	Simple API for XML Parsing (SAX)	945
18.3.3	Pull-API StAX	945
18.3.4	Java Document Object Model (JDOM)	945
18.3.5	JAXP als Java-Schnittstelle zu XML	946
18.3.6	DOM-Bäume einlesen mit JAXP *	946
18.4	Java Architecture for XML Binding (JAXB)	947
18.4.1	Bean für JAXB aufbauen	947
18.4.2	JAXBContext und die Marshaller	948
18.4.3	Ganze Objektgraphen schreiben und lesen	949
18.4.4	Validierung	952
18.4.5	Weitere JAXB-Annotationen *	955
18.4.6	Beans aus XML-Schema-Datei generieren	961

18.5	Serielle Verarbeitung mit StAX	965
18.5.1	Unterschiede der Verarbeitungsmodelle	965
18.5.2	XML-Dateien mit dem Cursor-Verfahren lesen	966
18.5.3	XML-Dateien mit dem Iterator-Verfahren verarbeiten *	969
18.5.4	Mit Filtern arbeiten *	971
18.5.5	XML-Dokumente schreiben	972
18.6	Serielle Verarbeitung von XML mit SAX *	975
18.6.1	Schnittstellen von SAX	975
18.6.2	SAX-Parser erzeugen	976
18.6.3	Operationen der Schnittstelle »ContentHandler«	977
18.6.4	ErrorHandler und EntityResolver	979
18.7	XML-Dateien mit JDOM verarbeiten	980
18.7.1	JDOM beziehen	980
18.7.2	Paketübersicht *	980
18.7.3	Die Document-Klasse	982
18.7.4	Eingaben aus der Datei lesen	983
18.7.5	Das Dokument im XML-Format ausgeben	984
18.7.6	Der Dokumenttyp *	984
18.7.7	Elemente	985
18.7.8	Zugriff auf Elementinhalte	987
18.7.9	Liste mit Unterelementen erzeugen *	989
18.7.10	Neue Elemente einfügen und ändern	990
18.7.11	Attributinhalt lesen und ändern	992
18.7.12	XPath	995
18.8	Transformationen mit XSLT *	998
18.8.1	Templates und XPath als Kernelemente von XSLT	998
18.8.2	Umwandlung von XML-Dateien mit JDOM und JAXP	1000
18.9	XML-Schema-Validierung *	1001
18.9.1	SchemaFactory und Schema	1002
18.9.2	Validator	1002
18.9.3	Validierung unterschiedlicher Datenquellen durchführen	1002
18.10	Zum Weiterlesen	1003

19 Grafische Oberflächen mit Swing	1005
------------------------------------	------

19.1	Das Abstract Window Toolkit und Swing	1005
19.1.1	SwingSet-Demos	1005
19.1.2	Abstract Window Toolkit (AWT)	1005
19.1.3	Java Foundation Classes	1007
19.1.4	Was Swing von AWT unterscheidet	1009
19.2	Mit NetBeans zur ersten Oberfläche	1010
19.2.1	Projekt anlegen	1011
19.2.2	Gui-Klasse hinzufügen	1012

19.2.3	Programm starten	1014
19.2.4	Grafische Oberfläche aufbauen	1014
19.2.5	Swing-Komponenten-Klassen	1017
19.2.6	Funktionalität geben	1018
19.3	Fenster unter grafischen Oberflächen	1021
19.3.1	Swing-Fenster mit javax.swing.JFrame darstellen	1021
19.3.2	Fenster schließbar machen – setDefaultCloseOperation()	1023
19.3.3	Sichtbarkeit des Fensters	1023
19.3.4	Größe und Position des Fensters verändern	1024
19.3.5	Fenster- und Dialog-Dekoration, Transparenz *	1025
19.3.6	Dynamisches Layout während einer Größenänderung *	1026
19.4	Beschriftungen (JLabel)	1026
19.4.1	Mehrzeiliger Text, HTML in der Darstellung	1029
19.5	Icon und ImageIcon für Bilder auf Swing-Komponenten	1030
19.5.1	Die Klasse »Icon«	1030
19.5.2	Die Schnittstelle Icon und eigene Icons *	1032
19.6	Es tut sich was – Ereignisse beim AWT	1034
19.6.1	Swings Ereignisquellen und Horcher (Listener)	1034
19.6.2	Listener implementieren	1035
19.6.3	Listener bei dem Ereignisauslöser anmelden/abmelden	1037
19.6.4	Aufrufen der Listener im AWT-Event-Thread	1038
19.6.5	Adapterklassen nutzen	1038
19.6.6	Innere Mitgliedsklassen und innere anonyme Klassen	1041
19.6.7	Ereignisse etwas genauer betrachtet *	1042
19.7	Schaltflächen	1044
19.7.1	Normale Schaltflächen (JButton)	1044
19.7.2	Der aufmerksame »ActionListener«	1046
19.7.3	Schaltflächen-Ereignisse vom Typ »ActionEvent«	1047
19.7.4	Basisklasse »AbstractButton«	1048
19.7.5	Wechselknopf (JToggleButton)	1050
19.8	Swing Action *	1050
19.9	JComponent und Component als Basis aller Komponenten	1052
19.9.1	Hinzufügen von Komponenten	1053
19.9.2	Tooltips (Kurzhinweise)	1053
19.9.3	Rahmen (Border) *	1054
19.9.4	Fokus und Navigation *	1056
19.9.5	Ereignisse jeder Komponente *	1057
19.9.6	Die Größe und Position einer Komponente *	1060
19.9.7	Komponenten-Ereignisse *	1061
19.9.8	Undurchsichtige (opaque) Komponente *	1061
19.9.9	Properties und Listener für Änderungen *	1062

19.10	Container	1062
19.10.1	Standardcontainer (JPanel)	1063
19.10.2	Bereich mit automatischen Rollbalken (JScrollPane)	1063
19.10.3	Reiter (JTabbedPane)	1064
19.10.4	Teilungs-Komponente (JSplitPane)	1065
19.11	Alles Auslegungssache: die Layoutmanager	1066
19.11.1	Übersicht über Layoutmanager	1066
19.11.2	Zuweisen eines Layoutmanagers	1067
19.11.3	Im Fluss mit FlowLayout	1068
19.11.4	BoxLayout	1069
19.11.5	Mit BorderLayout in alle Himmelsrichtungen	1070
19.11.6	Rasteranordnung mit GridLayout	1072
19.11.7	Der GridBagLayoutmanager *	1073
19.11.8	Null-Layout *	1078
19.11.9	Weitere Layoutmanager	1079
19.12	Rollbalken und Schieberegler	1079
19.12.1	Schieberegler (JSlider)	1080
19.12.2	Rollbalken (JScrollBar) *	1081
19.13	Kontrollfelder, Optionsfelder, Kontrollfeldgruppen	1085
19.13.1	Kontrollfelder (JCheckBox)	1086
19.13.2	ItemSelectable, ItemListener und das ItemEvent	1087
19.13.3	Sich gegenseitig ausschließende Optionen (JRadioButton)	1089
19.14	Fortschritte bei Operationen überwachen *	1090
19.14.1	Fortschrittsbalken (JProgressBar)	1090
19.14.2	Dialog mit Fortschrittsanzeige (ProgressMonitor)	1092
19.15	Menüs und Symbolleisten	1092
19.15.1	Die Menüleisten und die Einträge	1093
19.15.2	Menüeinträge definieren	1094
19.15.3	Einträge durch Action-Objekte beschreiben	1095
19.15.4	Mit der Tastatur: Mnemonics und Shortcut	1096
19.15.5	Der Tastatur-Shortcut (Accelerator)	1097
19.15.6	Tastenkürzel (Mnemonics)	1099
19.15.7	Symbolleisten alias Toolbars	1099
19.15.8	Popup-Menüs	1101
19.16	Das Model-View-Controller-Konzept	1105
19.17	Auswahlmenüs, Listen und Spinner	1107
19.17.1	Auswahlmenü (JComboBox)	1107
19.17.2	Zuordnung einer Taste mit einem Eintrag *	1110
19.17.3	Datumsauswahl	1111
19.17.4	Listen (JList)	1111
19.17.5	Drehfeld (JSpinner) *	1116

19.18	Textkomponenten	1118
19.18.1	Text in einer Eingabezeile	1118
19.18.2	Die Oberklasse der Text-Komponenten (JTextComponent)	1119
19.18.3	Geschützte Eingaben (JPasswordField)	1121
19.18.4	Validierende Eingabefelder (JFormattedTextField)	1121
19.18.5	Einfache mehrzeilige Textfelder (JTextArea)	1122
19.18.6	Editor-Klasse (JEditorPane) *	1125
19.19	Tabellen (JTable)	1127
19.19.1	Ein eigenes Tabellen-Model	1128
19.19.2	Basisklasse für eigene Modelle (AbstractTableModel)	1129
19.19.3	Vorgefertigtes Standard-Modell (DefaultTableModel)	1133
19.19.4	Ein eigener Renderer für Tabellen	1134
19.19.5	Zell-Editoren	1137
19.19.6	Größe und Umrandung der Zellen *	1138
19.19.7	Spalteninformationen*	1138
19.19.8	Tabellenkopf von Swing-Tabellen *	1139
19.19.9	Selektionen einer Tabelle *	1140
19.19.10	Automatisches Sortieren und Filtern mit RowSorter *	1141
19.20	Bäume (JTree)	1143
19.20.1	JTree und sein TreeModel und TreeNode	1143
19.20.2	Selektionen bemerken	1144
19.20.3	Das TreeModel von JTree *	1145
19.21	JRootPane und JDesktopPane *	1147
19.21.1	Wurzelkomponente der Top-Level-Komponenten (JRootPane)	1147
19.21.2	JDesktopPane und die Kinder JInternalFrame	1148
19.21.3	JLayeredPane	1149
19.22	Dialoge und Window-Objekte	1150
19.22.1	JWindow und JDialog	1150
19.22.2	Modal oder nicht-modal	1151
19.22.3	Standarddialoge mit JOptionPane	1151
19.22.4	Der Dateiauswahldialog	1154
19.22.5	Der Farbauswahldialog JColorChooser *	1158
19.23	Flexibles Java-Look-and-Feel	1160
19.23.1	Look and Feel global setzen	1160
19.23.2	UIManager	1161
19.23.3	Windowsoptik mit JGoodies Looks verbessern *	1162
19.24	Swing-Komponenten neu erstellen oder verändern *	1162
19.25	Die Zwischenablage (Clipboard)	1163
19.25.1	Clipboard-Objekte	1163
19.25.2	Auf den Inhalt zugreifen mit »Transferable«	1164
19.25.3	DataFlavor ist das Format der Daten in der Zwischenablage	1165

19.25.4	Einfügungen in der Zwischenablage erkennen	1167
19.25.5	Drag & Drop	1167
19.26	AWT, Swing und die Threads	1168
19.26.1	Ereignisschlange (EventQueue) und AWT-Event-Thread	1168
19.26.2	Swing ist nicht thread-sicher	1169
19.26.3	»invokeLater()« und »invokeAndWait()«	1171
19.26.4	SwingWorker	1173
19.26.5	Eigene Ereignisse in die Queue setzen *	1175
19.26.6	Auf alle Ereignisse hören *	1175
19.27	Barrierefreiheit mit der Java Accessibility API	1176
19.28	Zeitliches Ausführen mit dem javax.swing.Timer	1177
19.29	Zum Weiterlesen	1177

20	Grafikprogrammierung	1179
20.1	Grundlegendes zum Zeichnen	1179
20.1.1	Die paint()-Methode für das AWT-Frame	1179
20.1.2	Zeichnen von Inhalten auf ein JFrame	1181
20.1.3	Auffordern zum Neuzeichnen mit »repaint()«	1182
20.1.4	Java 2D-API	1182
20.2	Einfache Zeichenmethoden	1183
20.2.1	Linien	1183
20.2.2	Rechtecke	1184
20.2.3	Ovale und Kreisbögen	1185
20.2.4	Polygone und Polylines	1186
20.3	Zeichenketten schreiben und Fonts	1188
20.3.1	Zeichenfolgen schreiben	1189
20.3.2	Die Font-Klasse	1189
20.3.3	Einen neuen Font aus einem gegebenen Font ableiten	1191
20.3.4	Zeichensätze des Systems ermitteln *	1192
20.3.5	Neue TrueType-Fonts in Java nutzen	1193
20.3.6	Font-Metadaten durch FontMetrics *	1194
20.4	Geometrische Objekte	1197
20.4.1	Die Schnittstelle Shape	1198
20.4.2	Kreisförmiges	1200
20.4.3	Kurviges *	1200
20.4.4	Area und die konstruktive Flächengeometrie *	1200
20.4.5	Pfade *	1201
20.4.6	Punkt in einer Form, Schnitt von Linien, Abstand Punkt/Linie *	1204
20.5	Das Innere und Äußere einer Form	1205
20.5.1	Farben und die Paint-Schnittstelle	1206
20.5.2	Farben mit der Klasse »Color«	1206

20.5.3	Die Farben des Systems über SystemColor *	1211
20.5.4	Composite und Xor *	1214
20.5.5	Dicke und Art der Linien von Formen bestimmen über »Stroke« *	1215
20.6	Bilder	1219
20.6.1	Eine Übersicht über die Bilder-Bibliotheken	1220
20.6.2	Bilder mit »ImagelO« lesen	1221
20.6.3	Ein Bild zeichnen	1223
20.6.4	Programm-Icon/Fenster-Icon setzen	1226
20.6.5	Splash-Screen *	1227
20.6.6	Bilder im Speicher erzeugen *	1227
20.6.7	Pixel für Pixel auslesen und schreiben *	1229
20.6.8	Bilder skalieren *	1231
20.6.9	Schreiben mit »ImagelO«	1232
20.6.10	Asynchrones Laden mit getImage() und dem MediaTracker *	1236
20.7	Zum Weiterlesen	1237

21 Netzwerkprogrammierung 1239

21.1	Grundlegende Begriffe	1239
21.2	URI und URL	1241
21.2.1	Die Klasse »URI«	1241
21.2.2	Die Klasse »URL«	1242
21.2.3	Informationen über eine URL *	1244
21.2.4	Der Zugriff auf die Daten über die Klasse »URL«	1246
21.3	Die Klasse URLConnection *	1247
21.3.1	Methoden und Anwendung von URLConnection	1247
21.3.2	Protokoll- und Content-Handler	1249
21.3.3	Im Detail: vom URL zur URLConnection	1250
21.3.4	Der Protokoll-Handler für Jar-Dateien	1251
21.3.5	Basic Authentication/Proxy-Authentifizierung	1253
21.4	Mit GET und POST Daten übergeben *	1254
21.4.1	Kodieren der Parameter für Serverprogramme	1255
21.4.2	Eine Suchmaschine mit GET-Request ansprechen	1256
21.4.3	POST-Request absenden	1257
21.5	Host- und IP-Adressen	1258
21.5.1	Lebt der Rechner?	1260
21.5.2	IP-Adresse des lokalen Hosts	1260
21.5.3	Das Netz ist Klasse ...	1261
21.5.4	NetworkInterface	1261
21.6	Mit dem Socket zum Server	1262
21.6.1	Das Netzwerk ist der Computer	1263
21.6.2	Sockets	1263

21.6.3	Eine Verbindung zum Server aufbauen	1264
21.6.4	Server unter Spannung: die Ströme	1265
21.6.5	Die Verbindung wieder abbauen	1265
21.6.6	Informationen über den Socket *	1266
21.6.7	Reine Verbindungsdaten über SocketAddress *	1267
21.7	Client-Server-Kommunikation	1268
21.7.1	Warten auf Verbindungen	1269
21.7.2	Ein Multiplikationsserver	1270
21.7.3	Blockierendes Lesen	1273
21.7.4	Von außen erreichbar sein *	1274
21.8	Apache HttpComponents und Commons Net	1274
21.8.1	HttpComponents	1275
21.8.2	Jakarta Commons Net	1275
21.9	Zum Weiterlesen	1276

22 Verteilte Programmierung mit RMI 1277

22.1	Entfernte Objekte und Methoden	1277
22.1.1	Stellvertreter helfen bei entfernten Methodenaufrufen	1277
22.1.2	Standards für entfernte Objekte	1279
22.2	Java Remote Method Invocation	1279
22.2.1	Zusammenspiel von Server, Registry und Client	1279
22.2.2	Wie die Stellvertreter die Daten übertragen	1279
22.2.3	Probleme mit entfernten Methoden	1280
22.2.4	Nutzen von RMI bei Middleware-Lösungen	1282
22.2.5	Zentrale Klassen und Schnittstellen	1282
22.2.6	Entfernte und lokale Objekte im Vergleich	1283
22.3	Auf der Serverseite	1283
22.3.1	Entfernte Schnittstelle deklarieren	1283
22.3.2	Remote-Objekt-Implementierung	1284
22.3.3	Stellvertreterobjekte	1285
22.3.4	Der Namensdienst (Registry)	1285
22.3.5	Remote-Objekt-Implementierung exportieren und beim Namensdienst anmelden	1287
22.3.6	Einfaches Logging	1289
22.3.7	Aufräumen mit dem DGC	1290
22.4	Auf der Clientseite	1290
22.5	Entfernte Objekte übergeben und laden	1291
22.5.1	Klassen vom RMI-Klassenlader nachladen	1292
22.6	Zum Weiterlesen	1293

23	JavaServer Pages und Servlets	(295)
23.1	Dynamisch generierte Webseiten	1295
23.1.1	Was sind Servlets?	1296
23.1.2	Was sind JavaServer Pages?	1297
23.2	Servlets und JSPs mit Tomcat entwickeln	1298
23.2.1	Servlet-Container	1298
23.2.2	Entwicklung der Servlet-/JSP-Spezifikationen	1298
23.2.3	Webserver mit Servlet-Funktionalität	1299
23.2.4	Tomcat installieren	1299
23.2.5	Ablageort für eigene JSPs	1300
23.2.6	Webapplikationen	1301
23.2.7	Zuordnung von Webapplikationen zu physikalischen Verzeichnissen	1301
23.2.8	Web-Projekt mit Eclipse IDE for Java EE Developers	1302
23.3	Statisches und Dynamisches	1303
23.3.1	Statischer Template-Code	1303
23.3.2	Dynamische Inhalte	1303
23.3.3	Kommentare	1304
23.4	Die Expression Language (EL)	1304
23.4.1	Operatoren der EL	1305
23.4.2	Literale	1305
23.4.3	Implizite EL-Objekte	1306
23.5	Formulardaten	1306
23.5.1	Einen Parameter auslesen	1307
23.5.2	HTML-Formulare	1307
23.6	Auf Beans zurückgreifen	1308
23.6.1	Beans in JSPs anlegen	1309
23.6.2	Properties einer Bean im EL-Ausdruck erfragen	1309
23.6.3	Properties mit <jsp:setProperty> setzen	1309
23.6.4	Bean-Klasse zum Testen von E-Mail-Adressen	1310
23.6.5	Parameterwerte in Bean übertragen	1311
23.7	JSP-Tag-Libraries	1312
23.7.1	Standard Tag Library (JSTL)	1312
23.8	Einbinden und Weiterleiten	1316
23.8.1	Einbinden von Inhalten	1316
23.8.2	Forward und Redirect	1317
23.8.3	Applets einbinden	1318
23.9	Skripting-Elemente in JSPs	1318
23.9.1	Scriptlets	1318
23.9.2	JSP-Ausdrücke	1319
23.9.3	JSP-Deklarationen	1319
23.9.4	Quoting	1320

23.9.5	Entsprechende XML-Tags	1320
23.9.6	Implizite Objekte für Scriptlets und JSP-Ausdrücke	1320
23.10	JSP-Direktiven	1321
23.10.1	page-Direktiven im Überblick	1321
23.10.2	Mit JSPs Bilder generieren	1323
23.11	Sitzungsverfolgung (Session Tracking)	1324
23.11.1	Lösungen für Sitzungsverfolgung	1324
23.11.2	Sitzungen in JSPs	1325
23.11.3	Auf Session-Dateien zurückgreifen	1325
23.12	Servlets	1326
23.12.1	Servlets compilieren	1329
23.12.2	Servlet-Mapping	1330
23.12.3	Der Lebenszyklus eines Servlets	1331
23.12.4	Mehrere Anfragen beim Servlet und die Thread-Sicherheit	1331
23.12.5	Servlets und Sessions	1332
23.12.6	Weiterleiten und Einbinden von Servlet-Inhalten	1332
23.13	Zum Weiterlesen	1333

24 Datenbankmanagement mit JDBC 1335

24.1	Relationale Datenbanken	1335
24.1.1	Das relationale Modell	1335
24.2	Datenbanken und Tools	1336
24.2.1	HSQLDB	1336
24.2.2	Weitere Datenbanken *	1337
24.2.3	Eclipse-Plugins zum Durchschauen von Datenbanken	1340
24.3	JDBC und Datenbanktreiber	1342
24.3.1	Treibertypen *	1343
24.3.2	JDBC-Versionen *	1344
24.4	Eine Beispielabfrage	1345
24.4.1	Schritte zur Datenbankabfrage	1345
24.4.2	Client für HSQLDB-Datenbank	1346
24.4.3	Datenbankbrowser und eine Beispielabfrage unter NetBeans	1348
24.5	Mit Java an eine Datenbank andocken	1351
24.5.1	Der Treiber-Manager *	1351
24.5.2	Den Treiber laden	1352
24.5.3	Eine Aufzählung aller Treiber *	1353
24.5.4	Log-Informationen *	1353
24.5.5	Verbindung zur Datenbank auf- und abbauen	1354
24.6	Datenbankabfragen	1357
24.6.1	Abfragen über das Statement-Objekt	1357
24.6.2	Ergebnisse einer Abfrage in ResultSet	1359
24.6.3	Java und SQL-Datentypen	1360

24.6.4	Date, Time und Timestamp	1362
24.6.5	Unicode in der Spalte korrekt auslesen	1364
24.6.6	Eine SQL-NULL und »wasNull()« bei ResultSet	1364
24.6.7	Wie viele Zeilen hat ein ResultSet? *	1365
24.7	Elemente einer Datenbank hinzufügen und aktualisieren	1365
24.7.1	Batch-Updates	1366
24.7.2	Die Ausnahmen bei JDBC, SQLException und Unterklassen	1367
24.8	Vorbereitete Anweisungen (Prepared Statements)	1370
24.8.1	PreparedStatement-Objekte vorbereiten	1370
24.8.2	Werte für die Platzhalter eines PreparedStatement	1371
24.9	Transaktionen	1372
24.10	Metadaten *	1373
24.10.1	Metadaten über die Tabelle	1373
24.10.2	Informationen über die Datenbank	1376
24.11	Vorbereitete Datenbankverbindungen	1376
24.11.1	DataSource	1376
24.11.2	Gepoolte Verbindungen	1379
24.12	Einführung in SQL	1380
24.12.1	Ein Rundgang durch SQL-Abfragen	1381
24.12.2	Datenabfrage mit der Data Query Language (DQL)	1382
24.12.3	Tabellen mit der Data Definition Language (DDL) anlegen	1384
24.13	Zum Weiterlesen	1384

25 Reflection und Annotationen

1385

25.1	Metadaten	1385
25.1.1	Metadaten durch JavaDoc-Tags	1385
25.2	Metadaten der Klassen mit dem Class-Objekt	1386
25.2.1	An ein Class-Objekt kommen	1386
25.2.2	Was das Class-Objekt beschreibt *	1388
25.2.3	Der Name der Klasse	1390
25.2.4	»instanceof« mit Class-Objekten *	1392
25.2.5	Oberklassen finden *	1393
25.2.6	Implementierte Interfaces einer Klasse oder eines Interfaces *	1393
25.2.7	Modifizierer und die Klasse »Modifier« *	1394
25.2.8	Die Arbeit auf dem Feld *	1396
25.3	Attribute, Methoden und Konstruktoren	1396
25.3.1	Reflections – Gespür für Attribute einer Klasse	1398
25.3.2	Methoden einer Klasse erfragen	1401
25.3.3	Properties einer Bean erfragen	1404
25.3.4	Konstruktoren einer Klasse	1405
25.3.5	Annotationen	1407

25.4	Objekte erzeugen und manipulieren	1407
25.4.1	Objekte erzeugen	1407
25.4.2	Die Belegung der Variablen erfragen	1409
25.4.3	Eine generische eigene toString()-Methode *	1411
25.4.4	Variablen setzen	1412
25.4.5	Bean-Zustände kopieren *	1414
25.4.6	Private Attribute ändern	1415
25.5	Methoden aufrufen	1415
25.5.1	Statische Methoden aufrufen	1417
25.6	Eigene Annotationstypen *	1417
25.6.1	Annotationen zum Laden von Ressourcen	1417
25.6.2	Neue Annotationen deklarieren	1418
25.6.3	Annotationen mit genau einem Attribut	1419
25.6.4	Element-Werte-Paare (Attribute) hinzufügen	1420
25.6.5	Annotationsattribute vom Typ einer Aufzählung	1421
25.6.6	Felder von Annotationsattributen	1421
25.6.7	Vorbelegte Attribute	1423
25.6.8	Annotieren von Annotationstypen	1424
25.6.9	Deklarationen für unsere Ressourcen-Annotationen	1427
25.6.10	Annotierte Elemente auslesen	1428
25.6.11	Auf die Annotationsattribute zugreifen	1429
25.6.12	Komplettbeispiel zum Initialisieren von Ressourcen	1430
25.6.13	Mögliche Nachteile von Annotationen	1433
25.7	Zum Weiterlesen	1434

26 Dienstprogramme für die Java-Umgebung	1435
---	-------------

26.1	Die Werkzeuge vom JDK	1435
26.2	Java-Compiler und Java-Laufzeitumgebung	1436
26.2.1	Bytecode-Compiler javac	1436
26.2.2	Native Compiler	1436
26.2.3	Java-Programme in ein natives ausführbares Programm einpacken	1437
26.2.4	Der Java-Interpreter java	1437
26.3	Das Archivformat Jar	1439
26.3.1	Das Dienstprogramm jar benutzen	1440
26.3.2	Das Manifest	1442
26.3.3	Applikationen in Jar-Archiven starten	1442
26.3.4	Applets in Jar-Archiven	1443
26.4	Monitoringprogramme	1444
26.4.1	jps	1444
26.4.2	jstat	1444
26.4.3	jmap	1445

26.4.4	jstack	1445
26.4.5	VisualVM	1446
26.5	Ant	1450
26.5.1	Bezug und Installation von Ant	1451
26.5.2	Das Build-Skript build.xml	1451
26.5.3	Build den Build	1452
26.5.4	Properties	1452
26.5.5	Externe und vordefinierte Properties	1453
26.5.6	Weitere Ant-Tasks	1454
26.6	Weitere Dienstprogramme	1455
26.6.1	Sourcecode Beautifier	1455
26.6.2	Java-Programme als Systemdienst ausführen	1456
26.7	Zum Weiterlesen	1457
Anhang:		1459
A	Die Begleit-DVD	1459
Index		1461