

# Auf einen Blick

1	Java ist auch eine Sprache .....	49
2	Imperative Sprachkonzepte .....	91
3	Klassen und Objekte .....	213
4	Arrays und ihre Anwendungen .....	255
5	Der Umgang mit Zeichen und Zeichenketten .....	299
6	Eigene Klassen schreiben .....	383
7	Objektorientierte Beziehungsfragen .....	445
8	Records, Schnittstellen, Aufzählungen, versiegelte Klassen .....	507
9	Ausnahmen müssen sein .....	569
10	Geschachtelte Typen .....	631
11	Besondere Typen der Java SE .....	649
12	Generics<T> .....	721
13	Lambda-Ausdrücke und funktionale Programmierung .....	777
14	Architektur, Design und angewandte Objektorientierung .....	847
15	Java Platform Module System .....	861
16	Die Klassenbibliothek .....	883
17	Einführung in die nebenläufige Programmierung .....	921
18	Einführung in Datenstrukturen und Algorithmen .....	959
19	Einführung in grafische Oberflächen .....	1021
20	Einführung in Dateien und Datenströme .....	1047
21	Einführung ins Datenbankmanagement mit JDBC .....	1083
22	Bits und Bytes, Mathematisches und Geld .....	1089
23	Testen mit JUnit .....	1147
24	Die Werkzeuge des JDK .....	1171

# Inhalt

Materialien zum Buch .....	31
Vorwort .....	33

<b>1 Java ist auch eine Sprache</b>	49
<b>1.1 Historischer Hintergrund</b>	50
<b>1.2 Warum Java populär ist – die zentralen Eigenschaften</b>	52
1.2.1 Bytecode .....	52
1.2.2 Ausführung des Bytecodes durch eine virtuelle Maschine .....	52
1.2.3 Plattformunabhängigkeit .....	53
1.2.4 Java als Sprache, Laufzeitumgebung und Standardbibliothek .....	53
1.2.5 Objektorientierung in Java .....	54
1.2.6 Java ist verbreitet und bekannt .....	54
1.2.7 Java ist schnell – Optimierung und Just-in-time-Compilation .....	55
1.2.8 Zeiger und Referenzen .....	57
1.2.9 Bring den Müll raus, Garbage-Collector! .....	58
1.2.10 Ausnahmebehandlung .....	58
1.2.11 Das Angebot an Bibliotheken und Werkzeugen .....	59
1.2.12 Vergleichbar einfache Syntax .....	60
1.2.13 Verzicht auf umstrittene Konzepte .....	61
1.2.14 Java ist Open Source .....	62
1.2.15 Wofür sich Java weniger eignet .....	62
<b>1.3 Java im Vergleich zu anderen Sprachen *</b>	63
1.3.1 Java und C(++) .....	64
1.3.2 Java und JavaScript .....	64
1.3.3 Ein Wort zu Microsoft, Java und zu J++ .....	65
1.3.4 Java und C#/.NET .....	66
<b>1.4 Weiterentwicklung und Verluste</b>	66
1.4.1 Die Entwicklung von Java und seine Zukunftsaussichten .....	67
1.4.2 Features, Enhancements (Erweiterungen) und ein JSR .....	68
1.4.3 Applets .....	68
1.4.4 JavaFX .....	69
<b>1.5 Java-Plattformen: Java SE, Jakarta EE, Java ME, Java Card</b>	70
1.5.1 Die Java SE-Plattform .....	70
1.5.2 Java ME: Java für die Kleinen .....	72

1.5.3	Java für die ganz, ganz Kleinen .....	73
1.5.4	Java für die Großen: Jakarta EE (ehemals Java EE) .....	73
<b>1.6</b>	<b>Java SE-Implementierungen</b> .....	<b>74</b>
1.6.1	OpenJDK .....	74
1.6.2	Oracle JDK .....	75
<b>1.7</b>	<b>JDK installieren</b> .....	<b>77</b>
1.7.1	Oracle JDK unter Windows installieren .....	77
<b>1.8</b>	<b>Das erste Programm compilieren und testen</b> .....	<b>79</b>
1.8.1	99 Flaschen Bier an der Wand: Der Countdown geht weiter! .....	79
1.8.2	Der Compilerlauf .....	80
1.8.3	Die Laufzeitumgebung .....	81
1.8.4	Häufige Compiler- und Interpreter-Probleme .....	82
<b>1.9</b>	<b>Entwicklungsumgebungen</b> .....	<b>83</b>
1.9.1	IntelliJ IDEA .....	83
1.9.2	Eclipse IDE .....	87
1.9.3	NetBeans .....	89
<b>1.10</b>	<b>Zum Weiterlesen</b> .....	<b>89</b>

## **2 Imperative Sprachkonzepte**

---

<b>2.1</b>	<b>Elemente der Programmiersprache Java</b> .....	<b>91</b>
2.1.1	Token .....	92
2.1.2	Textkodierung durch Unicode-Zeichen .....	93
2.1.3	Bezeichner .....	93
2.1.4	Literale .....	95
2.1.5	(Reservierte) Schlüsselwörter .....	95
2.1.6	Zusammenfassung der lexikalischen Analyse .....	97
2.1.7	Kommentare .....	98
<b>2.2</b>	<b>Von der Klasse zur Anweisung</b> .....	<b>99</b>
2.2.1	Was sind Anweisungen? .....	100
2.2.2	Klassendeklaration .....	100
2.2.3	Die Reise beginnt am main(String[]) .....	101
2.2.4	Der erste Methodenaufruf: println(..) .....	102
2.2.5	Atomare Anweisungen und Anweisungssequenzen .....	103
2.2.6	Mehr zu print(..), println(..) und printf(..) für Bildschirmausgaben .....	104
2.2.7	Die API-Dokumentation .....	106
2.2.8	Ausdrücke .....	107
2.2.9	Ausdrucksanweisung .....	107

2.2.10	Erster Einblick in die Objektorientierung .....	108
2.2.11	Modifizierer .....	109
2.2.12	Gruppieren von Anweisungen mit Blöcken .....	109
<b>2.3</b>	<b>Datentypen, Typisierung, Variablen und Zuweisungen</b> .....	<b>111</b>
2.3.1	Primitive Datentypen im Überblick .....	112
2.3.2	Variablen Deklarationen .....	115
2.3.3	Automatisches Feststellen der Typen mit var .....	117
2.3.4	Finale Variablen und der Modifizierer final .....	118
2.3.5	Konsoleneingaben .....	119
2.3.6	Wahrheitswerte .....	121
2.3.7	Ganzzahlige Datentypen .....	121
2.3.8	Unterstriche in Zahlen .....	123
2.3.9	Alphanumerische Zeichen .....	124
2.3.10	Fließkommazahlen mit den Datentypen float und double .....	124
2.3.11	Gute Namen, schlechte Namen .....	126
2.3.12	Keine automatische Initialisierung von lokalen Variablen .....	127
<b>2.4</b>	<b>Ausdrücke, Operanden und Operatoren</b> .....	<b>127</b>
2.4.1	Zuweisungsoperator .....	128
2.4.2	Arithmetische Operatoren .....	130
2.4.3	Unäres Minus und Plus .....	133
2.4.4	Präfix- oder Postfix-Inkrement und -Dekrement .....	133
2.4.5	Zuweisung mit Operation (Verbundoperator) .....	135
2.4.6	Die relationalen Operatoren und die Gleichheitsoperatoren .....	137
2.4.7	Logische Operatoren: Nicht, Und, Oder, XOR .....	139
2.4.8	Kurzschluss-Operatoren .....	140
2.4.9	Der Rang der Operatoren in der Auswertungsreihenfolge .....	141
2.4.10	Die Typumwandlung (das Casting) .....	144
2.4.11	Überladenes Plus für Strings .....	150
2.4.12	Operator vermisst * .....	151
<b>2.5</b>	<b>Bedingte Anweisungen oder Fallunterscheidungen</b> .....	<b>151</b>
2.5.1	Verzweigung mit der if-Anweisung .....	152
2.5.2	Die Alternative mit einer if-else-Anweisung wählen .....	154
2.5.3	Der Bedingungsoperator .....	158
2.5.4	Die switch-Anweisung bietet die Alternative .....	161
2.5.5	Switch-Ausdrücke .....	168
<b>2.6</b>	<b>Immer das Gleiche mit den Schleifen</b> .....	<b>170</b>
2.6.1	Die while-Schleife .....	171
2.6.2	Die do-while-Schleife .....	173
2.6.3	Die for-Schleife .....	175
2.6.4	Schleifenbedingungen und Vergleiche mit == * .....	179

2.6.5	Schleifenabbruch mit break und zurück zum Test mit continue .....	181
2.6.6	break und continue mit Marken *	184
<b>2.7</b>	<b>Methoden einer Klasse</b> .....	188
2.7.1	Bestandteile einer Methode .....	188
2.7.2	Signatur-Beschreibung in der Java-API .....	190
2.7.3	Aufruf einer Methode .....	191
2.7.4	Methoden ohne Parameter deklarieren .....	192
2.7.5	Statische Methoden (Klassenmethoden) .....	193
2.7.6	Parameter, Argument und Wertübergabe .....	194
2.7.7	Methoden vorzeitig mit return beenden .....	196
2.7.8	Methoden mit Rückgaben .....	197
2.7.9	Methoden überladen .....	202
2.7.10	Gültigkeitsbereich .....	204
2.7.11	Vorgegebener Wert für nicht aufgeführte Argumente *	206
2.7.12	Rekursive Methoden *	206
2.7.13	Die Türme von Hanoi *	210
<b>2.8</b>	<b>Zum Weiterlesen</b> .....	212

## **3 Klassen und Objekte**

---

<b>3.1</b>	<b>Objektorientierte Programmierung (OOP)</b> .....	213
3.1.1	Warum überhaupt OOP? .....	214
3.1.2	Denk ich an Java, denk ich an Wiederverwendbarkeit .....	214
<b>3.2</b>	<b>Eigenschaften einer Klasse</b> .....	215
3.2.1	Klassenarbeit mit Point .....	216
<b>3.3</b>	<b>Natürlich modellieren mit der UML (Unified Modeling Language) *</b> .....	217
3.3.1	Wichtige Diagrammtypen der UML *	217
<b>3.4</b>	<b>Neue Objekte erzeugen</b> .....	219
3.4.1	Ein Exemplar einer Klasse mit dem Schlüsselwort new anlegen .....	219
3.4.2	Deklarieren von Referenzvariablen .....	219
3.4.3	Jetzt mach mal 'nen Punkt: Zugriff auf Objektvariablen und -methoden .....	220
3.4.4	Der Zusammenhang von new, Heap und Garbage-Collector .....	225
3.4.5	Überblick über Point-Methoden .....	226
3.4.6	Konstruktoren nutzen .....	230
<b>3.5</b>	<b>ZZZZZnake</b> .....	231
<b>3.6</b>	<b>Pakete schnüren, Importe und Compilationseinheiten</b> .....	233
3.6.1	Java-Pakete .....	234

3.6.2	Pakete der Standardbibliothek .....	234
3.6.3	Volle Qualifizierung und import-Deklaration .....	234
3.6.4	Mit import p1.p2.* alle Typen eines Pakets erreichen .....	236
3.6.5	Hierarchische Strukturen über Pakete und die Spiegelung im Dateisystem ..	237
3.6.6	Die package-Deklaration .....	237
3.6.7	Unbenanntes Paket (default package) .....	239
3.6.8	Compilationseinheit (Compilation Unit) .....	239
3.6.9	Statischer Import * .....	240
<b>3.7</b>	<b>Mit Referenzen arbeiten, Vielfalt, Identität, Gleichwertigkeit .....</b>	<b>241</b>
3.7.1	null-Referenz und die Frage der Philosophie .....	241
3.7.2	Alles auf null? Referenzen testen .....	243
3.7.3	Zuweisungen bei Referenzen .....	244
3.7.4	Methoden mit Referenztypen als Parameter .....	246
3.7.5	Identität von Objekten .....	250
3.7.6	Gleichwertigkeit und die Methode equals(...) .....	251
<b>3.8</b>	<b>Zum Weiterlesen .....</b>	<b>253</b>

## **4 Arrays und ihre Anwendungen**

---

<b>4.1</b>	<b>Einfache Feldarbeit .....</b>	<b>255</b>
4.1.1	Grundbestandteile .....	256
4.1.2	Deklaration von Array-Variablen .....	257
4.1.3	Array-Objekte mit new erzeugen .....	258
4.1.4	Arrays mit { Inhalt } .....	259
4.1.5	Die Länge eines Arrays über die Objektvariable length auslesen .....	260
4.1.6	Zugriff auf die Elemente über den Index .....	260
4.1.7	Typische Array-Fehler .....	262
4.1.8	Arrays an Methoden übergeben .....	264
4.1.9	Mehrere Rückgabewerte * .....	264
4.1.10	Vorinitialisierte Arrays .....	265
<b>4.2</b>	<b>Die erweiterte for-Schleife .....</b>	<b>267</b>
<b>4.3</b>	<b>Methode mit variabler Argumentanzahl (Varargs) .....</b>	<b>271</b>
4.3.1	System.out.printf(...) nimmt eine beliebige Anzahl von Argumenten an .....	271
4.3.2	Durchschnitt finden von variablen Argumenten .....	272
4.3.3	Varargs-Designtipps * .....	273
<b>4.4</b>	<b>Mehrdimensionale Arrays * .....</b>	<b>273</b>
4.4.1	Nichtrechteckige Arrays * .....	276

<b>4.5 Bibliotheksunterstützung von Arrays .....</b>	279
4.5.1 Klonen kann sich lohnen – Arrays vermehren .....	279
4.5.2 Warum »können« Arrays so wenig? .....	280
4.5.3 Array-Inhalte kopieren .....	280
<b>4.6 Die Klasse Arrays zum Vergleichen, Füllen, Suchen und Sortieren nutzen .....</b>	282
4.6.1 Eine lange Schlange .....	292
<b>4.7 Der Einstiegspunkt für das Laufzeitsystem: main(..) .....</b>	295
4.7.1 Korrekte Deklaration der Startmethode .....	295
4.7.2 Kommandozeilenargumente verarbeiten .....	296
4.7.3 Der Rückgabetyp von main(..) und System.exit(int) * .....	297
<b>4.8 Zum Weiterlesen .....</b>	298

## **5 Der Umgang mit Zeichen und Zeichenketten** 299

---

<b>5.1 Von ASCII über ISO-8859-1 zu Unicode .....</b>	299
5.1.1 ASCII .....	299
5.1.2 ISO/IEC 8859-1 .....	300
5.1.3 Unicode .....	301
5.1.4 Unicode-Zeichenkodierung .....	303
5.1.5 Escape-Sequenzen/Fluchtsymbole .....	304
5.1.6 Schreibweise für Unicode-Zeichen und Unicode-Escapes .....	305
5.1.7 Java-Versionen gehen mit dem Unicode-Standard Hand in Hand * .....	306
<b>5.2 Datentypen für Zeichen und Zeichenfolgen .....</b>	307
<b>5.3 Die Character-Klasse .....</b>	308
5.3.1 Ist das so? .....	308
5.3.2 Zeichen in Großbuchstaben/Kleinbuchstaben konvertieren .....	311
5.3.3 Vom Zeichen zum String .....	312
5.3.4 Von char in int: vom Zeichen zur Zahl * .....	312
<b>5.4 Zeichenfolgen .....</b>	313
<b>5.5 Die Klasse String und ihre Methoden .....</b>	315
5.5.1 String-Literale als String-Objekte für konstante Zeichenketten .....	316
5.5.2 Konkatenation mit + .....	316
5.5.3 Mehrzeilige Textblöcke mit """ .....	317
5.5.4 String-Länge und Test auf Leer-String .....	322
5.5.5 Zugriff auf ein bestimmtes Zeichen mit charAt(int) .....	323
5.5.6 Nach enthaltenen Zeichen und Zeichenfolgen suchen .....	324
5.5.7 Das Hangman-Spiel .....	327

5.5.8	Gut, dass wir verglichen haben .....	329
5.5.9	String-Teile extrahieren .....	334
5.5.10	Strings anhängen, zusammenfügen, Groß-/Kleinschreibung und Weißraum .....	338
5.5.11	Gesucht, gefunden, ersetzt .....	342
5.5.12	String-Objekte mit Konstruktoren und aus Wiederholungen erzeugen * .....	344
<b>5.6</b>	<b>Veränderbare Zeichenketten mit StringBuilder und StringBuffer</b> .....	348
5.6.1	Anlegen von StringBuilder-Objekten .....	348
5.6.2	StringBuilder in andere Zeichenkettenformate konvertieren .....	349
5.6.3	Zeichen(folgen) erfragen .....	349
5.6.4	Daten anhängen .....	350
5.6.5	Zeichen(folgen) setzen, löschen und umdrehen .....	351
5.6.6	Länge und Kapazität eines StringBuilder-Objekts * .....	353
5.6.7	Vergleich von StringBuilder-Exemplaren und Strings mit StringBuilder .....	355
5.6.8	hashCode() bei StringBuilder * .....	356
<b>5.7</b>	<b>CharSequence als Basistyp</b> .....	357
<b>5.8</b>	<b>Konvertieren zwischen Primitiven und Strings</b> .....	360
5.8.1	Unterschiedliche Typen in String-Präsentationen konvertieren .....	360
5.8.2	String-Inhalt in einen primitiven Wert konvertieren .....	362
5.8.3	String-Präsentation in den Formaten Binär, Hex und Oktal * .....	364
<b>5.9</b>	<b>Strings zusammenhängen (konkatenieren)</b> .....	368
5.9.1	Strings mit StringJoiner zusammenhängen .....	368
<b>5.10</b>	<b>Zerlegen von Zeichenketten</b> .....	370
5.10.1	Splitten von Zeichenketten mit split(..) .....	371
5.10.2	Yes we can, yes we scan – die Klasse Scanner .....	372
<b>5.11</b>	<b>Ausgaben formatieren</b> .....	375
5.11.1	Formatieren und Ausgeben mit format() .....	375
<b>5.12</b>	<b>Zum Weiterlesen</b> .....	381

---

<b>6</b>	<b>Eigene Klassen schreiben</b>	383
<b>6.1</b>	<b>Eigene Klassen mit Eigenschaften deklarieren</b> .....	383
6.1.1	Objektvariablen deklarieren .....	384
6.1.2	Methoden deklarieren .....	387
6.1.3	Verdeckte (shadowed) Variablen .....	389
6.1.4	Die this-Referenz .....	391

<b>6.2</b>	<b>Privatsphäre und Sichtbarkeit</b>	394
6.2.1	Für die Öffentlichkeit: public	395
6.2.2	Kein Public Viewing – Passwörter sind privat	395
6.2.3	Wieso nicht freie Methoden und Variablen für alle?	397
6.2.4	Privat ist nicht ganz privat: Es kommt darauf an, wer's sieht *	397
6.2.5	Zugriffsmethoden für Objektvariablen deklarieren	398
6.2.6	Setter und Getter nach der JavaBeans-Spezifikation	398
6.2.7	Paketsichtbar	401
6.2.8	Zusammenfassung zur Sichtbarkeit	402
<b>6.3</b>	<b>Eine für alle – statische Methoden und Klassenvariablen</b>	405
6.3.1	Warum statische Eigenschaften sinnvoll sind	406
6.3.2	Statische Eigenschaften mit static	406
6.3.3	Statische Eigenschaften über Referenzen nutzen? *	408
6.3.4	Warum die Groß- und Kleinschreibung wichtig ist *	409
6.3.5	Statische Variablen zum Datenaustausch *	409
6.3.6	Statische Eigenschaften und Objekteigenschaften *	411
<b>6.4</b>	<b>Konstanten und Aufzählungen</b>	412
6.4.1	Konstanten über statische finale Variablen	412
6.4.2	Typsichere Aufzählungen	413
6.4.3	Aufzählungstypen: typsichere Aufzählungen mit enum	415
<b>6.5</b>	<b>Objekte anlegen und zerstören</b>	420
6.5.1	Konstruktoren schreiben	420
6.5.2	Verwandtschaft von Methode und Konstruktor	422
6.5.3	Der Standard-Konstruktor (default constructor)	423
6.5.4	Parametrisierte und überladene Konstruktoren	424
6.5.5	Copy-Konstruktor	426
6.5.6	Einen anderen Konstruktor der gleichen Klasse mit this(...) aufrufen	427
6.5.7	Immutable-Objekte und Wither-Methoden	430
6.5.8	Ihr fehlt uns nicht – der Garbage-Collector	432
<b>6.6</b>	<b>Klassen- und Objektinitialisierung *</b>	434
6.6.1	Initialisierung von Objektvariablen	434
6.6.2	Statische Blöcke als Klasseninitialisierer	436
6.6.3	Initialisierung von Klassenvariablen	437
6.6.4	Eincompilierte Belegungen der Klassenvariablen	437
6.6.5	Exemplarinitialisierer (Instanzinitialisierer)	438
6.6.6	Finale Werte im Konstruktor und in statischen Blöcken setzen	441
<b>6.7</b>	<b>Zum Weiterlesen</b>	443

<b>7.1</b>	<b>Assoziationen zwischen Objekten</b>	445
7.1.1	Unidirektionale 1:1-Beziehung	446
7.1.2	Zwei Freunde müsst ihr werden – bidirektionale 1:1-Beziehungen	448
7.1.3	Unidirektionale 1:n-Beziehung	449
<b>7.2</b>	<b>Vererbung</b>	455
7.2.1	Vererbung in Java	456
7.2.2	Ereignisse modellieren	456
7.2.3	Die implizite Basisklasse <code>java.lang.Object</code>	458
7.2.4	Einfach- und Mehrfachvererbung *	459
7.2.5	Sehen Kinder alles? Die Sichtbarkeit <code>protected</code>	459
7.2.6	Konstruktoren in der Vererbung und <code>super(...)</code>	460
<b>7.3</b>	<b>Typen in Hierarchien</b>	465
7.3.1	Automatische und explizite Typumwandlung	466
7.3.2	Das Substitutionsprinzip	469
7.3.3	Typen mit dem <code>instanceof</code> -Operator testen	471
7.3.4	Pattern-Matching bei <code>instanceof</code>	474
7.3.5	Pattern-Matching bei <code>switch</code>	476
<b>7.4</b>	<b>Methoden überschreiben</b>	479
7.4.1	Methoden in Unterklassen mit neuem Verhalten ausstatten	479
7.4.2	Mit <code>super</code> an die Eltern	483
<b>7.5</b>	<b>Drum prüfe, wer sich dynamisch bindet</b>	485
7.5.1	Gebunden an <code>toString()</code>	486
7.5.2	Implementierung von <code>System.out.println(Object)</code>	487
<b>7.6</b>	<b>Finale Klassen und finale Methoden</b>	488
7.6.1	Finale Klassen	488
7.6.2	Nicht überschreibbare (finale) Methoden	489
<b>7.7</b>	<b>Abstrakte Klassen und abstrakte Methoden</b>	490
7.7.1	Abstrakte Klassen	491
7.7.2	Abstrakte Methoden	492
<b>7.8</b>	<b>Weiteres zum Überschreiben und dynamischen Binden</b>	499
7.8.1	Nicht dynamisch gebunden bei privaten, statischen und finalen Methoden	499
7.8.2	Kovariante Rückgabetypen	500
7.8.3	Array-Typen und Kovarianz *	501
7.8.4	Dynamisch gebunden auch bei Konstruktoraufrufen *	502
7.8.5	Keine dynamische Bindung bei überdeckten Objektvariablen *	504
<b>7.9</b>	<b>Zum Weiterlesen und Programmieraufgabe</b>	505

## 8 Records, Schnittstellen, Aufzählungen, versiegelte Klassen

507

<b>8.1</b>	<b>Records</b>	507
8.1.1	Einfache Records	507
8.1.2	Records mit Methoden	509
8.1.3	Konstruktoren von Records anpassen	511
8.1.4	Konstruktoren ergänzen	513
8.1.5	Record-Patterns zur Destrukturierung	514
8.1.6	Records: Zusammenfassung	518
<b>8.2</b>	<b>Schnittstellen</b>	518
8.2.1	Schnittstellen sind neue Typen	518
8.2.2	Schnittstellen deklarieren	519
8.2.3	Abstrakte Methoden in Schnittstellen	519
8.2.4	Implementieren von Schnittstellen	520
8.2.5	Ein Polymorphie-Beispiel mit Schnittstellen	523
8.2.6	Records implementieren Schnittstellen	525
8.2.7	Die Mehrfachvererbung bei Schnittstellen	526
8.2.8	Keine Kollisionsgefahr bei Mehrfachvererbung *	529
8.2.9	Erweitern von Interfaces – Subinterfaces	530
8.2.10	Konstantendeklarationen bei Schnittstellen	531
8.2.11	Nachträgliches Implementieren von Schnittstellen *	531
8.2.12	Statische ausprogrammierte Methoden in Schnittstellen	532
8.2.13	Erweitern und Ändern von Schnittstellen	534
8.2.14	Default-Methoden	536
8.2.15	Erweiterte Schnittstellen deklarieren und nutzen	537
8.2.16	Öffentliche und private Schnittstellenmethoden	541
8.2.17	Erweiterte Schnittstellen, Mehrfachvererbung und Mehrdeutigkeiten *	541
8.2.18	Bausteine bilden mit Default-Methoden *	545
8.2.19	Markierungsschnittstellen *	548
8.2.20	(Abstrakte) Klassen und Schnittstellen im Vergleich	549
<b>8.3</b>	<b>Aufzählungstypen</b>	550
8.3.1	Methoden auf Enum-Objekten	550
8.3.2	Aufzählungen mit eigenen Methoden *	554
8.3.3	enum mit eigenen Konstruktoren und Initialisierern	556
8.3.4	enum kann Schnittstellen implementieren	559
<b>8.4</b>	<b>Versiegelte Klassen und Schnittstellen</b>	560
8.4.1	Versiegelte Klassen und Schnittstellen (sealed classes/interfaces)	562
8.4.2	Unterklassen sind final, sealed, non-sealed	564
8.4.3	Vollständige Abdeckung von Aufzählungen	564

8.4.4	Abkürzende Schreibweisen .....	565
8.4.5	Versiegelte Schnittstellen und Records .....	566
<b>8.5</b>	<b>Zum Weiterlesen .....</b>	<b>568</b>

## **9 Ausnahmen müssen sein**

---

<b>9.1</b>	<b>Problembereiche einzäunen .....</b>	<b>570</b>
9.1.1	Exceptions in Java mit try und catch .....	570
9.1.2	Geprüfte und ungeprüfte Ausnahmen .....	571
9.1.3	Eine NumberFormatException fliegt (ungeprüfte Ausnahme) .....	571
9.1.4	Datum-Zeitstempel in Textdatei anhängen (geprüfte Ausnahme) .....	573
9.1.5	Wiederholung abgebrochener Bereiche * .....	576
9.1.6	Bitte nicht schlucken – leere catch-Blöcke .....	576
9.1.7	Mehrere Ausnahmen auffangen .....	577
9.1.8	Zusammenfassen gleicher catch-Blöcke mit dem multi-catch .....	578
<b>9.2</b>	<b>Ausnahmen mit throws nach oben reichen .....</b>	<b>579</b>
9.2.1	throws bei Konstruktoren und Methoden .....	579
<b>9.3</b>	<b>Die Klassenhierarchie der Ausnahmen .....</b>	<b>580</b>
9.3.1	Eigenschaften des Exception-Objekts .....	580
9.3.2	Basistyp Throwable .....	581
9.3.3	Die Exception-Hierarchie .....	582
9.3.4	Oberausnahmen auffangen .....	583
9.3.5	Schon gefangen? .....	585
9.3.6	Ablauf einer Ausnahmesituation .....	586
9.3.7	Nicht zu allgemein fangen! .....	586
9.3.8	Bekannte RuntimeException-Klassen .....	588
9.3.9	Kann man abfangen, muss man aber nicht .....	589
<b>9.4</b>	<b>Abschlussbehandlung mit finally .....</b>	<b>590</b>
<b>9.5</b>	<b>Auslösen eigener Exceptions .....</b>	<b>595</b>
9.5.1	Mit throw Ausnahmen auslösen .....	595
9.5.2	Vorhandene Runtime-Ausnahmetypen kennen und nutzen .....	597
9.5.3	Parameter testen und gute Fehlermeldungen .....	600
9.5.4	Neue Exception-Klassen deklarieren .....	601
9.5.5	Eigene Ausnahmen als Unterklassen von Exception oder RuntimeException? .....	603
9.5.6	Ausnahmen abfangen und weiterleiten * .....	606
9.5.7	Aufruf-Stack von Ausnahmen verändern * .....	607
9.5.8	Geschachtelte Ausnahmen * .....	608

<b>9.6</b>	<b>try mit Ressourcen (automatisches Ressourcen-Management)</b> .....	611
9.6.1	try mit Ressourcen .....	612
9.6.2	Die Schnittstelle AutoCloseable .....	613
9.6.3	Ausnahmen vom close() .....	614
9.6.4	Typen, die AutoCloseable und Closeable sind .....	615
9.6.5	Mehrere Ressourcen nutzen .....	616
9.6.6	try mit Ressourcen auf null-Ressourcen .....	618
9.6.7	Unterdrückte Ausnahmen * .....	618
<b>9.7</b>	<b>Besonderheiten bei der Ausnahmebehandlung *</b> .....	621
9.7.1	Rückgabewerte bei ausgelösten Ausnahmen .....	622
9.7.2	Ausnahmen und Rückgaben verschwinden – das Duo return und finally .....	622
9.7.3	throws bei überschriebenen Methoden .....	623
9.7.4	Nicht erreichbare catch-Klauseln .....	625
<b>9.8</b>	<b>Harte Fehler – Error *</b> .....	627
<b>9.9</b>	<b>Assertions *</b> .....	627
9.9.1	Assertions in eigenen Programmen nutzen .....	628
9.9.2	Assertions aktivieren und Laufzeit-Errors .....	628
9.9.3	Assertions feiner aktivieren oder deaktivieren .....	630
<b>9.10</b>	<b>Zum Weiterlesen</b> .....	630

## **10 Geschachtelte Typen** 631

---

<b>10.1</b>	<b>Geschachtelte Klassen, Schnittstellen und Aufzählungen</b> .....	631
<b>10.2</b>	<b>Statische geschachtelte Typen</b> .....	633
<b>10.3</b>	<b>Nichtstatische geschachtelte Typen</b> .....	635
10.3.1	Exemplare innerer Klassen erzeugen .....	635
10.3.2	Die this-Referenz .....	636
10.3.3	Vom Compiler generierte Klassendateien * .....	637
<b>10.4</b>	<b>Lokale Klassen</b> .....	638
10.4.1	Beispiel mit eigener Klassendeklaration .....	638
10.4.2	Lokale Klasse für einen Timer nutzen .....	639
<b>10.5</b>	<b>Anonyme innere Klassen</b> .....	639
10.5.1	Nutzung einer anonymen inneren Klasse für den Timer .....	640
10.5.2	Umsetzung innerer anonymer Klassen * .....	641
10.5.3	Konstruktoren innerer anonymer Klassen .....	642

<b>10.6</b>	<b>Vermischtes</b>	644
10.6.1	Zugriff auf lokale Variablen aus lokalen und anonymen Klassen *	644
10.6.2	this in Unterklassen *	644
10.6.3	Geschachtelte Klassen greifen auf private Eigenschaften zu	645
10.6.4	Nester	647
<b>10.7</b>	<b>Zum Weiterlesen</b>	648

## 11 Besondere Typen der Java SE

---

<b>11.1</b>	<b>Object ist die Mutter aller Klassen</b>	650
11.1.1	Klassenobjekte	650
11.1.2	Objektidentifikation mit <code>toString()</code>	651
11.1.3	Objektgleichwertigkeit mit <code>equals(...)</code> und Identität	653
11.1.4	Klonen eines Objekts mit <code>clone()</code> *	659
11.1.5	Hashwerte über <code>hashCode()</code> liefern *	664
11.1.6	<code>System.identityHashCode(...)</code> und das Problem der nicht eindeutigen Objektverweise *	671
11.1.7	Synchronisation *	672
<b>11.2</b>	<b>Schwache Referenzen und Cleaner</b>	672
<b>11.3</b>	<b>Die Utility-Klasse <code>java.util.Objects</code></b>	673
11.3.1	Eingebaute null-Tests für <code>equals(...)/hashCode()</code>	674
11.3.2	<code>Objects.toString(...)</code>	675
11.3.3	null-Prüfungen mit eingebauter Ausnahmebehandlung	675
11.3.4	Tests auf null	676
11.3.5	Indexbezogene Programmargumente auf Korrektheit prüfen	677
<b>11.4</b>	<b>Vergleichen von Objekten und Ordnung herstellen</b>	677
11.4.1	Natürlich geordnet oder nicht?	678
11.4.2	<code>compare*()</code> -Methode der Schnittstellen Comparable und Comparator	679
11.4.3	Rückgabewerte kodieren die Ordnung	680
11.4.4	Mit einem Beispiel-Comparator Süßigkeiten nach Kalorien sortieren	680
11.4.5	Tipps für Comparator- und Comparable-Implementierungen	682
11.4.6	Statische und Default-Methoden in Comparator	683
<b>11.5</b>	<b>Wrapper-Klassen und Autoboxing</b>	686
11.5.1	Wrapper-Objekte erzeugen	688
11.5.2	Konvertierungen in eine String-Präsentation	689
11.5.3	Von einer String-Präsentation parsen	691

11.5.4	Die Basisklasse Number für numerische Wrapper-Objekte .....	691
11.5.5	Vergleiche durchführen mit compare*(...), compareTo(...), equals(...) und Hashwerten .....	693
11.5.6	Statische Reduzierungsmethoden in Wrapper-Klassen .....	696
11.5.7	Konstanten für die Größe eines primitiven Typs .....	697
11.5.8	Behandeln von vorzeichenlosen Zahlen * .....	697
11.5.9	Die Klassen Integer und Long .....	699
11.5.10	Die Klassen Double und Float für Fließkommazahlen .....	700
11.5.11	Die Boolean-Klasse .....	700
11.5.12	Autoboxing: Boxing und Unboxing .....	701
<b>11.6</b>	<b>Iterator, Iterable *</b> .....	706
11.6.1	Die Schnittstelle Iterator .....	706
11.6.2	Wer den Iterator liefert .....	710
11.6.3	Die Schnittstelle Iterable .....	710
11.6.4	Erweitertes for und Iterable .....	711
11.6.5	Interne Iteration .....	711
11.6.6	Ein eigenes Iterable implementieren * .....	712
<b>11.7</b>	<b>Annotationen in der Java SE</b> .....	714
11.7.1	Orte für Annotationen .....	714
11.7.2	Annotationstypen aus java.lang .....	715
11.7.3	@Deprecated .....	715
11.7.4	Annotationen mit zusätzlichen Informationen .....	716
11.7.5	@SuppressWarnings .....	716
<b>11.8</b>	<b>Zum Weiterlesen</b> .....	719

---

<b>12</b>	<b>Generics&lt;T&gt;</b>	721
<b>12.1</b>	<b>Einführung in Java Generics</b> .....	721
12.1.1	Mensch versus Maschine – Typprüfung des Compilers und der Laufzeitumgebung .....	721
12.1.2	Raketen .....	722
12.1.3	Generische Typen deklarieren .....	724
12.1.4	Generics nutzen .....	726
12.1.5	Diamonds are forever .....	729
12.1.6	Generische Schnittstellen .....	732
12.1.7	Generische Methoden/Konstruktoren und Typ-Inferenz .....	733
<b>12.2</b>	<b>Umsetzen der Generics, Typlösung und Raw-Types</b> .....	737
12.2.1	Realisierungsmöglichkeiten .....	737
12.2.2	Typlösung (Type Erasure) .....	738

12.2.3	Probleme der Typlösung .....	739
12.2.4	Raw-Type .....	743
<b>12.3</b>	<b>Die Typen über Bounds einschränken .....</b>	<b>745</b>
12.3.1	Einfache Einschränkungen mit extends .....	745
12.3.2	Weitere Obertypen mit & .....	748
<b>12.4</b>	<b>Typparameter in der throws-Klausel *</b> .....	<b>748</b>
12.4.1	Deklaration einer Klasse mit Typvariable <E extends Exception> .....	748
12.4.2	Parametrisierter Typ bei Typvariable <E extends Exception> .....	749
<b>12.5</b>	<b>Generics und Vererbung, Invarianz .....</b>	<b>752</b>
12.5.1	Arrays sind kovariant .....	752
12.5.2	Generics sind nicht kovariant, sondern invariant .....	752
12.5.3	Wildcards mit ? .....	753
12.5.4	Bounded Wildcards .....	756
12.5.5	Bounded-Wildcard-Typen und Bounded-Typvariablen .....	759
12.5.6	Das LESS-Prinzip .....	761
12.5.7	Enum<E extends Enum<E>> * .....	764
<b>12.6</b>	<b>Konsequenzen der Typlösung: Typ-Token, Arrays und Brücken *</b> .....	<b>766</b>
12.6.1	Typ-Token .....	766
12.6.2	Super-Type-Token .....	768
12.6.3	Generics und Arrays .....	769
12.6.4	Brückenmethoden * .....	770
<b>12.7</b>	<b>Zum Weiterlesen .....</b>	<b>775</b>

---

<b>13</b>	<b>Lambda-Ausdrücke und funktionale Programmierung</b>	<b>777</b>
<b>13.1</b>	<b>Funktionale Schnittstellen und Lambda-Ausdrücke .....</b>	<b>777</b>
13.1.1	Klassen implementieren Schnittstellen .....	777
13.1.2	Lambda-Ausdrücke implementieren Schnittstellen .....	779
13.1.3	Funktionale Schnittstellen .....	780
13.1.4	Der Typ eines Lambda-Ausdrucks ergibt sich durch den Zieltyp .....	781
13.1.5	Annotation @FunctionalInterface .....	786
13.1.6	Syntax für Lambda-Ausdrücke .....	787
13.1.7	Die Umgebung der Lambda-Ausdrücke und Variablenzugriffe .....	792
13.1.8	Ausnahmen in Lambda-Ausdrücken .....	797
13.1.9	Klassen mit einer abstrakten Methode als funktionale Schnittstelle? * .....	801
<b>13.2</b>	<b>Methodenreferenz .....</b>	<b>801</b>
13.2.1	Motivation .....	801
13.2.2	Methodenreferenzen mit :: .....	802

13.2.3	Varianten von Methodenreferenzen .....	803
<b>13.3</b>	<b>Konstruktorreferenz .....</b>	<b>805</b>
13.3.1	Konstruktorreferenzen schreiben .....	806
13.3.2	Parameterlose und parametrisierte Konstruktoren .....	807
13.3.3	Nützliche vordefinierte Schnittstellen für Konstruktorreferenzen .....	808
<b>13.4</b>	<b>Funktionale Programmierung .....</b>	<b>809</b>
13.4.1	Code = Daten .....	809
13.4.2	Programmierparadigmen: imperativ oder deklarativ .....	810
13.4.3	Das Wesen der funktionalen Programmierung .....	811
13.4.4	Funktionale Programmierung und funktionale Programmiersprachen .....	814
13.4.5	Funktionen höherer Ordnung am Beispiel von Comparator .....	816
13.4.6	Lambda-Ausdrücke als Abbildungen bzw. Funktionen betrachten .....	816
<b>13.5</b>	<b>Funktionale Schnittstellen aus dem <code>java.util.function</code>-Paket .....</b>	<b>818</b>
13.5.1	Blöcke mit Code und die funktionale Schnittstelle Consumer .....	818
13.5.2	Supplier .....	820
13.5.3	Prädikate und <code>java.util.function.Predicate</code> .....	821
13.5.4	Funktionen über die funktionale Schnittstelle <code>java.util.function.Function</code> .....	823
13.5.5	Ein bisschen Bi ... .....	827
13.5.6	Funktionale Schnittstellen mit Primitiven .....	830
<b>13.6</b>	<b>Optional ist keine Nullnummer .....</b>	<b>832</b>
13.6.1	Einsatz von null .....	833
13.6.2	Der Optional-Typ .....	835
13.6.3	Erst mal funktional mit Optional .....	838
13.6.4	Primitiv-Optionales mit speziellen <code>Optional*</code> -Klassen .....	841
<b>13.7</b>	<b>Was ist jetzt so funktional? .....</b>	<b>843</b>
<b>13.8</b>	<b>Zum Weiterlesen .....</b>	<b>845</b>

## **14 Architektur, Design und angewandte Objektorientierung** 847

---

<b>14.1</b>	<b>SOLIDe Modellierung .....</b>	<b>847</b>
14.1.1	DRY, KISS und YAGNI .....	848
14.1.2	SOLID .....	848
14.1.3	Sei nicht STUPID .....	850
<b>14.2</b>	<b>Architektur, Design und Implementierung .....</b>	<b>851</b>
<b>14.3</b>	<b>Design-Patterns (Entwurfsmuster) .....</b>	<b>852</b>
14.3.1	Motivation für Design-Patterns .....	852

14.3.2	Singleton .....	853
14.3.3	Fabrikmethoden .....	854
14.3.4	Das Beobachter-Pattern mit Listener realisieren .....	855
<b>14.4</b>	<b>Zum Weiterlesen .....</b>	<b>859</b>

## **15 Java Platform Module System**

---

<b>15.1</b>	<b>Klassenlader (Class Loader) und Modul-/Klassenpfad .....</b>	<b>861</b>
15.1.1	Klassenladen auf Abruf .....	861
15.1.2	Klassenlader bei der Arbeit zusehen .....	862
15.1.3	JMOD-Dateien und JAR-Dateien .....	863
15.1.4	Woher die Klassen kommen: Suchorte und spezielle Klassenlader .....	864
15.1.5	Setzen des Suchpfades .....	865
<b>15.2</b>	<b>Module im JPMS einbinden .....</b>	<b>867</b>
15.2.1	Wer sieht wen? .....	868
15.2.2	Plattform-Module und ein JMOD-Beispiel .....	869
15.2.3	Interne Plattformegenschaften nutzen, --add-exports .....	870
15.2.4	Neue Module einbinden .....	872
<b>15.3</b>	<b>Eigene Module entwickeln .....</b>	<b>873</b>
15.3.1	Modul com.tutego.candytester .....	873
15.3.2	Moduldeklaration mit module-info.java und exports .....	874
15.3.3	Modul com.tutego.main .....	875
15.3.4	Modulinfodatei mit requires .....	875
15.3.5	Moduleinsetzer schreiben, JVM-Schalter -p und -m .....	876
15.3.6	Modulinfodatei-Experimente .....	877
15.3.7	Automatische Module .....	878
15.3.8	Unbenanntes Modul .....	879
15.3.9	Lesbarkeit und Zugreifbarkeit .....	879
15.3.10	Modul-Migration .....	880
<b>15.4</b>	<b>Zum Weiterlesen .....</b>	<b>881</b>

## **16 Die Klassenbibliothek**

---

<b>16.1</b>	<b>Die Java-Klassenphilosophie .....</b>	<b>883</b>
16.1.1	Modul, Paket, Typ .....	883
16.1.2	Übersicht über die Pakete der Standardbibliothek .....	886

<b>16.2 Einfache Zeitmessung und Profiling *</b> .....	890
<b>16.3 Die Klasse Class</b> .....	891
16.3.1 An ein Class-Objekt kommen .....	892
16.3.2 Eine Class ist ein Type .....	894
<b>16.4 Die Utility-Klassen System und Properties</b> .....	894
16.4.1 Speicher der JVM .....	895
16.4.2 Anzahl der CPUs bzw. Kerne .....	896
16.4.3 Systemeigenschaften der Java-Umgebung .....	897
16.4.4 Eigene Properties von der Konsole aus setzen *	898
16.4.5 Zeilenumbruchzeichen, line.separator .....	900
16.4.6 Umgebungsvariablen des Betriebssystems .....	900
<b>16.5 Sprachen der Länder</b> .....	902
16.5.1 Sprachen in Regionen über Locale-Objekte .....	902
<b>16.6 Wichtige Datum-Klassen im Überblick</b> .....	906
16.6.1 Der 1.1.1970 .....	907
16.6.2 System.currentTimeMillis() .....	907
16.6.3 Einfache Zeitumrechnungen durch TimeUnit .....	908
<b>16.7 Date-Time-API</b> .....	909
16.7.1 Menschenzeit und Maschinenzeit .....	910
16.7.2 Die Datumsklasse LocalDate .....	913
<b>16.8 Logging mit Java</b> .....	914
16.8.1 Logging-APIs .....	914
16.8.2 Logging mit java.util.logging .....	914
<b>16.9 Maven: Build-Management und Abhängigkeiten auflösen</b> .....	917
16.9.1 Dependency hinzunehmen .....	917
16.9.2 Lokales und das Remote Repository .....	918
16.9.3 Lebenszyklus, Phasen und Maven-Plugins .....	918
<b>16.10 Zum Weiterlesen</b> .....	919

---

<b>17 Einführung in die nebenläufige Programmierung</b>	921
<b>17.1 Nebenläufigkeit und Parallelität</b> .....	921
17.1.1 Multitasking, Prozesse und Threads .....	922
17.1.2 Wie nebenläufige Programme die Geschwindigkeit steigern können .....	923
17.1.3 Java-Threads .....	924
17.1.4 Was Java für Nebenläufigkeit alles bietet .....	925

<b>17.2 Existierende Threads und neue Threads erzeugen</b> .....	925
17.2.1 Main-Thread .....	925
17.2.2 Wer bin ich? .....	926
17.2.3 Die Schnittstelle Runnable implementieren .....	926
17.2.4 Thread aufbauen .....	927
17.2.5 Thread mit Runnable starten .....	928
17.2.6 Runnable parametrisieren .....	930
17.2.7 Die Klasse Thread erweitern * .....	931
<b>17.3 Thread-Eigenschaften und Zustände</b> .....	931
17.3.1 Der Name eines Threads .....	931
17.3.2 Die Zustände eines Threads * .....	932
17.3.3 Schläfer gesucht .....	932
17.3.4 Wann Threads fertig sind .....	934
17.3.5 Einen Thread höflich mit Interrupt beenden .....	934
17.3.6 Unbehandelte Ausnahmen, Thread-Ende und UncaughtExceptionHandler .....	937
17.3.7 Der stop() von außen * .....	939
17.3.8 Ein Rendezvous mit join(..) * .....	939
17.3.9 Arbeit niederlegen und wieder aufnehmen * .....	941
17.3.10 Priorität * .....	941
<b>17.4 Der Ausführer (Executor) kommt</b> .....	943
17.4.1 Die Schnittstelle Executor .....	943
17.4.2 Glücklich in der Gruppe – die Thread-Pools .....	945
17.4.3 Threads mit Rückgabe über Callable .....	947
17.4.4 Erinnerungen an die Zukunft – die Future-Rückgabe .....	949
17.4.5 Mehrere Callable-Objekte abarbeiten .....	952
17.4.6 CompletionService und ExecutorCompletionService .....	953
17.4.7 ScheduledExecutorService: wiederholende Aufgaben und Zeitsteuerungen .....	955
17.4.8 Asynchrones Programmieren mit CompletableFuture (CompletionStage) ...	955
<b>17.5 Zum Weiterlesen</b> .....	958

---

<b>18 Einführung in Datenstrukturen und Algorithmen</b>	959
<b>18.1 Listen</b> .....	959
18.1.1 Erstes Listen-Beispiel .....	960
18.1.2 Auswahlkriterium ArrayList oder LinkedList .....	961
18.1.3 Die Schnittstelle List .....	961
18.1.4 ArrayList .....	968

18.1.5	LinkedList .....	969
18.1.6	Der Array-Adapter Arrays.asList(..) .....	971
18.1.7	toArray(..) von Collection verstehen – die Gefahr einer Falle erkennen .....	973
18.1.8	Primitive Elemente in Datenstrukturen verwalten .....	976
<b>18.2</b>	<b>Mengen (Sets) .....</b>	<b>977</b>
18.2.1	Ein erstes Mengen-Beispiel .....	977
18.2.2	Methoden der Schnittstelle Set .....	979
18.2.3	HashSet .....	981
18.2.4	TreeSet – die sortierte Menge .....	982
18.2.5	Die Schnittstellen NavigableSet und SortedSet .....	984
18.2.6	LinkedHashSet .....	986
<b>18.3</b>	<b>Assoziative Speicher .....</b>	<b>987</b>
18.3.1	Die Klassen HashMap und TreeMap, statische Map-Methoden .....	988
18.3.2	Einfügen und Abfragen des Assoziativspeichers .....	991
<b>18.4</b>	<b>Ausgewählte Basistypen .....</b>	<b>994</b>
<b>18.5</b>	<b>Java-Stream-API .....</b>	<b>994</b>
18.5.1	Deklaratives Programmieren .....	994
18.5.2	Interne versus externe Iteration .....	995
18.5.3	Was ist ein Stream? .....	996
<b>18.6</b>	<b>Einen Stream erzeugen .....</b>	<b>997</b>
18.6.1	Parallele oder sequenzielle Streams .....	1000
<b>18.7</b>	<b>Terminale Operationen .....</b>	<b>1001</b>
18.7.1	Die Anzahl der Elemente .....	1001
18.7.2	Und jetzt alle – forEach*(..) .....	1002
18.7.3	Einzelne Elemente aus dem Strom holen .....	1002
18.7.4	Existenztests mit Prädikaten .....	1003
18.7.5	Einen Strom auf sein kleinstes bzw. größtes Element reduzieren .....	1004
18.7.6	Einen Strom mit eigenen Funktionen reduzieren .....	1005
18.7.7	Ergebnisse in einen Container schreiben, Teil 1: collect(..) .....	1006
18.7.8	Ergebnisse in einen Container schreiben, Teil 2: Collector und Collectors .....	1007
18.7.9	Ergebnisse in einen Container schreiben, Teil 3: Gruppierungen .....	1009
18.7.10	Stream-Elemente in ein Array oder einen Iterator übertragen .....	1011
<b>18.8</b>	<b>Intermediäre Operationen .....</b>	<b>1012</b>
18.8.1	Element-Vorschau .....	1013
18.8.2	Filtern von Elementen .....	1013
18.8.3	Statusbehaftete intermediäre Operationen .....	1014
18.8.4	Präfix-Operation .....	1015
18.8.5	Abbildungen .....	1016
<b>18.9</b>	<b>Zum Weiterlesen .....</b>	<b>1019</b>

<b>19.1 GUI-Frameworks</b> .....	1021
19.1.1 Kommandozeile .....	1021
19.1.2 Grafische Benutzeroberfläche .....	1021
19.1.3 Abstract Window Toolkit (AWT) .....	1022
19.1.4 Java Foundation Classes und Swing .....	1022
19.1.5 JavaFX .....	1022
19.1.6 SWT (Standard Widget Toolkit) * .....	1024
<b>19.2 Deklarative und programmierte Oberflächen</b> .....	1025
19.2.1 GUI-Beschreibungen in JavaFX .....	1025
19.2.2 Deklarative GUI-Beschreibungen für Swing? .....	1026
<b>19.3 GUI-Builder</b> .....	1026
19.3.1 GUI-Builder für JavaFX .....	1027
19.3.2 GUI-Builder für Swing .....	1027
<b>19.4 Mit dem Eclipse WindowBuilder zur ersten Swing-Oberfläche</b> .....	1027
19.4.1 WindowBuilder installieren .....	1028
19.4.2 Mit WindowBuilder eine GUI-Klasse hinzufügen .....	1029
19.4.3 Das Layoutprogramm starten .....	1031
19.4.4 Grafische Oberfläche aufbauen .....	1032
19.4.5 Swing-Komponenten-Klassen .....	1035
19.4.6 Funktionalität geben .....	1036
<b>19.5 Grundlegendes zum Zeichnen</b> .....	1039
19.5.1 Die paint(Graphics)-Methode für den AWT-Frame .....	1040
19.5.2 Die ereignisorientierte Programmierung ändert Fensterinhalte .....	1041
19.5.3 Zeichnen von Inhalten auf einen JFrame .....	1043
19.5.4 Auffordern zum Neuzeichnen mit repaint(...) .....	1044
19.5.5 Java 2D-API .....	1045
<b>19.6 Zum Weiterlesen</b> .....	1045

<b>20.1 Alte und neue Welt in java.io und java.nio</b> .....	1047
20.1.1 java.io-Paket mit File-Klasse .....	1047
20.1.2 NIO.2 und das java.nio-Paket .....	1048
20.1.3 java.io.File oder java.nio.*-Typen? .....	1048

<b>20.2 Dateisysteme und Pfade</b> .....	1048
20.2.1 FileSystem und Path .....	1049
20.2.2 Die Utility-Klasse Files .....	1056
<b>20.3 Dateien mit wahlfreiem Zugriff</b> .....	1058
20.3.1 Ein RandomAccessFile zum Lesen und Schreiben öffnen .....	1059
20.3.2 Aus dem RandomAccessFile lesen .....	1059
20.3.3 Schreiben mit RandomAccessFile .....	1062
20.3.4 Die Länge des RandomAccessFile .....	1062
20.3.5 Hin und her in der Datei .....	1063
<b>20.4 Basisklassen für die Ein-/Ausgabe</b> .....	1064
20.4.1 Die vier abstrakten Basisklassen .....	1064
20.4.2 Die abstrakte Basisklasse OutputStream .....	1065
20.4.3 Die abstrakte Basisklasse InputStream .....	1067
20.4.4 Die abstrakte Basisklasse Writer .....	1069
20.4.5 Die Schnittstelle Appendable *	1070
20.4.6 Die abstrakte Basisklasse Reader .....	1071
20.4.7 Die Schnittstellen Closeable, AutoCloseable und Flushable .....	1074
<b>20.5 Lesen aus Dateien und Schreiben in Dateien</b> .....	1076
20.5.1 Byteorientierte Datenströme über Files beziehen .....	1076
20.5.2 Zeichenorientierte Datenströme über Files beziehen .....	1077
20.5.3 Die Funktion von OpenOption bei den Files.new*(..)-Methoden .....	1079
20.5.4 Ressourcen aus dem Modulpfad und aus JAR-Dateien laden .....	1081
<b>20.6 Zum Weiterlesen</b> .....	1082

---

<b>21 Einführung ins Datenbankmanagement mit JDBC</b>	1083
<b>21.1 Relationale Datenbanken und Java-Zugriffe</b> .....	1083
21.1.1 Das relationale Modell .....	1083
21.1.2 Java-APIs zum Zugriff auf relationale Datenbanken .....	1084
21.1.3 Die JDBC-API und Implementierungen: JDBC-Treiber .....	1085
21.1.4 H2 ist das Werkzeug auf der Insel .....	1085
<b>21.2 Eine Beispielabfrage</b> .....	1086
21.2.1 Schritte zur Datenbankabfrage .....	1086
21.2.2 Mit Java auf die relationale Datenbank zugreifen .....	1086
<b>21.3 Zum Weiterlesen</b> .....	1088

<b>22.1 Bits und Bytes</b> .....	1089
22.1.1 Die Bit-Operatoren Komplement, Und, Oder und XOR .....	1090
22.1.2 Repräsentation ganzer Zahlen in Java – das Zweierkomplement .....	1091
22.1.3 Das binäre (Basis 2), oktale (Basis 8), hexadezimale (Basis 16) Stellenwertsystem .....	1093
22.1.4 Auswirkung der Typumwandlung auf die Bit-Muster .....	1094
22.1.5 Vorzeichenlos arbeiten .....	1097
22.1.6 Die Verschiebeoperatoren .....	1099
22.1.7 Ein Bit setzen, löschen, umdrehen und testen .....	1102
22.1.8 Bit-Methoden der Integer- und Long-Klasse .....	1102
<b>22.2 Fließkomma-Arithmetik in Java</b> .....	1104
22.2.1 Spezialwerte für Unendlich, Null, NaN .....	1105
22.2.2 Standardnotation und wissenschaftliche Notation bei Fließkommazahlen * .....	1108
22.2.3 Mantisse und Exponent * .....	1108
<b>22.3 Die Eigenschaften der Klasse Math</b> .....	1110
22.3.1 Objektvariablen der Klasse Math .....	1110
22.3.2 Absolutwerte und Vorzeichen .....	1111
22.3.3 Maximum/Minimum .....	1111
22.3.4 Runden von Werten .....	1112
22.3.5 Rest der ganzzahligen Division * .....	1115
22.3.6 Division mit Rundung in Richtung negativ unendlich, alternativer Restwert * .....	1116
22.3.7 Multiply-Accumulate .....	1117
22.3.8 Wurzel- und Exponentialmethoden .....	1118
22.3.9 Der Logarithmus * .....	1119
22.3.10 Winkelmethoden * .....	1120
22.3.11 Zufallszahlen .....	1121
<b>22.4 Genauigkeit, Wertebereich eines Typs und Überlaufkontrolle *</b> .....	1121
22.4.1 Der größte und der kleinste Wert .....	1122
22.4.2 Überlauf und alles ganz exakt .....	1122
22.4.3 Was bitte macht eine ulp? .....	1125
<b>22.5 Zufallszahlen: Random, ThreadLocalRandom und SecureRandom</b> .....	1127
22.5.1 Die Klasse Random .....	1127
22.5.2 Die Klasse ThreadLocalRandom .....	1130
22.5.3 Die Klasse SecureRandom * .....	1131

<b>22.6</b>	<b>Große Zahlen *</b>	1131
22.6.1	Die Klasse BigInteger	1131
22.6.2	Beispiel: ganz lange Fakultäten mit BigInteger	1138
22.6.3	Große Fließkommazahlen mit BigDecimal	1139
22.6.4	Mit MathContext komfortabel die Rechengenauigkeit setzen	1142
22.6.5	Noch schneller rechnen durch mutable Implementierungen	1144
<b>22.7</b>	<b>Geld und Währung</b>	1144
22.7.1	Geldbeträge repräsentieren	1144
22.7.2	ISO 4217	1145
22.7.3	Währungen in Java repräsentieren	1145
<b>22.8</b>	<b>Zum Weiterlesen</b>	1146

## 23 Testen mit JUnit

1147

---

<b>23.1</b>	<b>Softwaretests</b>	1147
23.1.1	Vorgehen beim Schreiben von Testfällen	1148
<b>23.2</b>	<b>Das Test-Framework JUnit</b>	1148
23.2.1	Test-Driven Development und Test-First	1149
23.2.2	Testen, implementieren, testen, implementieren, testen, freuen	1150
23.2.3	JUnit-Tests ausführen	1152
23.2.4	assert*(...)-Methoden der Klasse Assertions	1153
23.2.5	Exceptions testen	1156
23.2.6	Grenzen für Ausführungszeiten festlegen	1157
23.2.7	Beschriftungen mit @DisplayName	1158
23.2.8	Verschachtelte Tests	1158
23.2.9	Tests ignorieren	1159
23.2.10	Mit Methoden der Assumptions-Klasse Tests abbrechen	1159
23.2.11	Parametrisierte Tests	1159
<b>23.3</b>	<b>Java-Assertions-Bibliotheken und AssertJ</b>	1161
23.3.1	AssertJ	1161
<b>23.4</b>	<b>Aufbau größerer Testfälle</b>	1163
23.4.1	Fixtures	1163
23.4.2	Sammlungen von Testklassen und Klassenorganisation	1165
<b>23.5</b>	<b>Wie gutes Design das Testen ermöglicht</b>	1165
<b>23.6</b>	<b>Dummy, Fake, Stub und Mock</b>	1167
<b>23.7</b>	<b>JUnit-Erweiterungen, Testzusätze</b>	1169
<b>23.8</b>	<b>Zum Weiterlesen</b>	1170

## 24 Die Werkzeuge des JDK

1171

---

<b>24.1</b>	<b>Übersicht</b>	1171
24.1.1	Aufbau und gemeinsame Schalter	1172
<b>24.2</b>	<b>Java-Quellen übersetzen</b>	1172
24.2.1	Der Java-Compiler des JDK	1172
24.2.2	Native Compiler	1173
<b>24.3</b>	<b>Die Java-Laufzeitumgebung</b>	1173
24.3.1	Schalter der JVM	1173
24.3.2	Der Unterschied zwischen java.exe und javaw.exe	1176
<b>24.4</b>	<b>Dokumentationskommentare mit Javadoc</b>	1176
24.4.1	Einen Dokumentationskommentar setzen	1176
24.4.2	Mit dem Werkzeug javadoc eine Dokumentation erstellen	1179
24.4.3	HTML-Tags in Dokumentationskommentaren *	1179
24.4.4	Generierte Dateien	1179
24.4.5	Dokumentationskommentare im Überblick *	1180
24.4.6	Javadoc und Doclets *	1182
24.4.7	Veraltete (deprecated) Typen und Eigenschaften	1182
24.4.8	Javadoc-Überprüfung mit DocLint	1185
<b>24.5</b>	<b>Das Archivformat JAR</b>	1186
24.5.1	Das Dienstprogramm jar benutzen	1186
24.5.2	Das Manifest	1186
24.5.3	Applikationen in JAR-Archiven starten; ausführbare JAR-Dateien	1187
<b>24.6</b>	<b>Zum Weiterlesen</b>	1188

## Anhang

1189

---

<b>A</b>	<b>Java SE-Module und Paketübersicht</b>	1189
Index	Index	1207