

# Auf einen Blick

1	Java ist auch eine Sprache .....	49
2	Imperative Sprachkonzepte .....	97
3	Klassen und Objekte .....	225
4	Arrays und ihre Anwendungen .....	267
5	Der Umgang mit Zeichen und Zeichenketten .....	315
6	Eigene Klassen schreiben .....	399
7	Objektorientierte Beziehungsfragen .....	461
8	Schnittstellen, Aufzählungen, versiegelte Klassen, Records .....	519
9	Ausnahmen müssen sein .....	577
10	Geschachtelte Typen .....	643
11	Besondere Typen der Java SE .....	663
12	Generics<T> .....	737
13	Lambda-Ausdrücke und funktionale Programmierung .....	793
14	Architektur, Design und angewandte Objektorientierung .....	865
15	Java Platform Module System .....	879
16	Die Klassenbibliothek .....	903
17	Einführung in die nebenläufige Programmierung .....	949
18	Einführung in Datenstrukturen und Algorithmen .....	991
19	Einführung in grafische Oberflächen .....	1051
20	Einführung in Dateien und Datenströme .....	1077
21	Einführung ins Datenbankmanagement mit JDBC .....	1113
22	Bits und Bytes, Mathematisches und Geld .....	1119
23	Testen mit JUnit .....	1177
24	Die Werkzeuge des JDK .....	1201

# Inhalt

Materialien zum Buch .....	30
Vorwort .....	31

## **1 Java ist auch eine Sprache** 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 .....	53
1.2.3 Plattformunabhängigkeit .....	53
1.2.4 Java als Sprache, Laufzeitumgebung und Standardbibliothek .....	54
1.2.5 Objektorientierung in Java .....	54
1.2.6 Java ist verbreitet und bekannt .....	55
1.2.7 Java ist schnell – Optimierung und Just-in-time-Compilation .....	56
1.2.8 Das Java-Security-Modell .....	57
1.2.9 Zeiger und Referenzen .....	58
1.2.10 Bring den Müll raus, Garbage-Collector! .....	59
1.2.11 Ausnahmebehandlung .....	60
1.2.12 Das Angebot an Bibliotheken und Werkzeugen .....	61
1.2.13 Vergleichbar einfache Syntax .....	62
1.2.14 Java ist Open Source .....	63
1.2.15 Wofür sich Java weniger eignet .....	64
<b>1.3 Java im Vergleich zu anderen Sprachen *</b> .....	65
1.3.1 Java und C(++) .....	66
1.3.2 Java und JavaScript .....	66
1.3.3 Ein Wort zu Microsoft, Java und zu J++ .....	66
1.3.4 Java und C#/.NET .....	67
<b>1.4 Weiterentwicklung und Verluste</b> .....	68
1.4.1 Die Entwicklung von Java und seine Zukunftsaussichten .....	68
1.4.2 Features, Enhancements (Erweiterungen) und ein JSR .....	70
1.4.3 Applets .....	70
1.4.4 JavaFX .....	71
<b>1.5 Java-Plattformen: Java SE, Jakarta EE, Java ME, Java Card</b> .....	72
1.5.1 Die Java SE-Plattform .....	72
1.5.2 Java ME: Java für die Kleinen .....	75

1.5.3	Java für die ganz, ganz Kleinen .....	75
1.5.4	Java für die Großen: Jakarta EE (ehemals Java EE) .....	76
1.5.5	Echtzeit-Java (Real-time Java) .....	77
<b>1.6</b>	<b>Java SE-Implementierungen</b> .....	<b>77</b>
1.6.1	OpenJDK .....	78
1.6.2	Oracle JDK .....	79
<b>1.7</b>	<b>JDK installieren</b> .....	<b>80</b>
1.7.1	AdoptOpenJDK unter Windows installieren .....	80
<b>1.8</b>	<b>Das erste Programm compilieren und testen</b> .....	<b>82</b>
1.8.1	Ein Quadratzahlen-Programm .....	83
1.8.2	Der Compilerlauf .....	84
1.8.3	Die Laufzeitumgebung .....	85
1.8.4	Häufige Compiler- und Interpreter-Probleme .....	85
<b>1.9</b>	<b>Entwicklungsumgebungen</b> .....	<b>86</b>
1.9.1	Intellij IDEA .....	87
1.9.2	Eclipse IDE .....	88
1.9.3	NetBeans .....	95
<b>1.10</b>	<b>Zum Weiterlesen</b> .....	<b>96</b>

## **2 Imperative Sprachkonzepte** 97

---

<b>2.1</b>	<b>Elemente der Programmiersprache Java</b> .....	<b>97</b>
2.1.1	Token .....	98
2.1.2	Textkodierung durch Unicode-Zeichen .....	99
2.1.3	Bezeichner .....	99
2.1.4	Literale .....	101
2.1.5	(Reservierte) Schlüsselwörter .....	101
2.1.6	Zusammenfassung der lexikalischen Analyse .....	103
2.1.7	Kommentare .....	103
<b>2.2</b>	<b>Von der Klasse zur Anweisung</b> .....	<b>105</b>
2.2.1	Was sind Anweisungen? .....	106
2.2.2	Klassendeklaration .....	106
2.2.3	Die Reise beginnt am main(String[]) .....	107
2.2.4	Der erste Methodenaufruf: println(...) .....	108
2.2.5	Atomare Anweisungen und Anweisungssequenzen .....	109
2.2.6	Mehr zu print(...), println(...) und printf(...) für Bildschirmausgaben .....	110
2.2.7	Die API-Dokumentation .....	111
2.2.8	Ausdrücke .....	112

---

2.2.9	Ausdrucksanweisung .....	113
2.2.10	Erster Einblick in die Objektorientierung .....	114
2.2.11	Modifizierer .....	115
2.2.12	Gruppieren von Anweisungen mit Blöcken .....	115
<b>2.3</b>	<b>Datentypen, Typisierung, Variablen und Zuweisungen .....</b>	<b>117</b>
2.3.1	Primitive Datentypen im Überblick .....	119
2.3.2	Variablendeklarationen .....	122
2.3.3	Automatisches Feststellen der Typen mit var .....	124
2.3.4	Finale Variablen und der Modifizierer final .....	125
2.3.5	Konsoleneingaben .....	126
2.3.6	Fließkommazahlen mit den Datentypen float und double .....	128
2.3.7	Ganzzahlige Datentypen .....	130
2.3.8	Wahrheitswerte .....	132
2.3.9	Unterstriche in Zahlen .....	132
2.3.10	Alphanumerische Zeichen .....	133
2.3.11	Gute Namen, schlechte Namen .....	134
2.3.12	Keine automatische Initialisierung von lokalen Variablen .....	135
<b>2.4</b>	<b>Ausdrücke, Operanden und Operatoren .....</b>	<b>136</b>
2.4.1	Zuweisungsoperator .....	136
2.4.2	Arithmetische Operatoren .....	138
2.4.3	Unäres Minus und Plus .....	141
2.4.4	Präfix- oder Postfix-Inkrement und -Dekrement .....	142
2.4.5	Zuweisung mit Operation (Verbundoperator) .....	144
2.4.6	Die relationalen Operatoren und die Gleichheitsoperatoren .....	145
2.4.7	Logische Operatoren: Nicht, Und, Oder, XOR .....	147
2.4.8	Kurzschluss-Operatoren .....	149
2.4.9	Der Rang der Operatoren in der Auswertungsreihenfolge .....	150
2.4.10	Die Typumwandlung (das Casting) .....	153
2.4.11	Überladenes Plus für Strings .....	158
2.4.12	Operator vermisst * .....	160
<b>2.5</b>	<b>Bedingte Anweisungen oder Fallunterscheidungen .....</b>	<b>160</b>
2.5.1	Verzweigung mit der if-Anweisung .....	160
2.5.2	Die Alternative mit einer if-else-Anweisung wählen .....	163
2.5.3	Der Bedingungsoperator .....	167
2.5.4	Die switch-Anweisung bietet die Alternative .....	170
2.5.5	Switch-Ausdrücke .....	176
<b>2.6</b>	<b>Immer das Gleiche mit den Schleifen .....</b>	<b>179</b>
2.6.1	Die while-Schleife .....	180
2.6.2	Die do-while-Schleife .....	182
2.6.3	Die for-Schleife .....	184

2.6.4	Schleifenbedingungen und Vergleiche mit == *	188
2.6.5	Schleifenabbruch mit break und zurück zum Test mit continue	190
2.6.6	break und continue mit Marken *	193
<b>2.7</b>	<b>Methoden einer Klasse</b>	<b>197</b>
2.7.1	Bestandteile einer Methode	198
2.7.2	Signatur-Beschreibung in der Java-API	199
2.7.3	Aufruf einer Methode	200
2.7.4	Methoden ohne Parameter deklarieren	201
2.7.5	Statische Methoden (Klassenmethoden)	202
2.7.6	Parameter, Argument und Wertübergabe	203
2.7.7	Methoden vorzeitig mit return beenden	205
2.7.8	Nicht erreichbarer Quellcode bei Methoden *	206
2.7.9	Methoden mit Rückgaben	207
2.7.10	Methoden überladen	212
2.7.11	Gültigkeitsbereich	214
2.7.12	Vorgegebener Wert für nicht aufgeführte Argumente *	215
2.7.13	Rekursive Methoden *	216
2.7.14	Die Türme von Hanoi *	221
<b>2.8</b>	<b>Zum Weiterlesen</b>	<b>223</b>

## **3 Klassen und Objekte** 225

---

<b>3.1</b>	<b>Objektorientierte Programmierung (OOP)</b>	<b>225</b>
3.1.1	Warum überhaupt OOP?	225
3.1.2	Denk ich an Java, denk ich an Wiederverwendbarkeit	226
<b>3.2</b>	<b>Eigenschaften einer Klasse</b>	<b>227</b>
3.2.1	Klassenarbeit mit Point	228
<b>3.3</b>	<b>Natürlich modellieren mit der UML (Unified Modeling Language) *</b>	<b>229</b>
3.3.1	Wichtige Diagrammtypen der UML *	229
<b>3.4</b>	<b>Neue Objekte erzeugen</b>	<b>231</b>
3.4.1	Ein Exemplar einer Klasse mit dem Schlüsselwort new anlegen	231
3.4.2	Deklarieren von Referenzvariablen	231
3.4.3	Jetzt mach mal 'nen Punkt: Zugriff auf Objektvariablen und -methoden	233
3.4.4	Der Zusammenhang von new, Heap und Garbage-Collector	237
3.4.5	Überblick über Point-Methoden	238
3.4.6	Konstruktoren nutzen	242
<b>3.5</b>	<b>ZZZZZnake</b>	<b>243</b>

<b>3.6</b>	<b>Pakete schnüren, Importe und Compilationseinheiten</b>	245
3.6.1	Java-Pakete	246
3.6.2	Pakete der Standardbibliothek	246
3.6.3	Volle Qualifizierung und import-Deklaration	246
3.6.4	Mit import p1.p2.* alle Typen eines Pakets erreichen	248
3.6.5	Hierarchische Strukturen über Pakete und die Spiegelung im Dateisystem	249
3.6.6	Die package-Deklaration	249
3.6.7	Unbenanntes Paket (default package)	251
3.6.8	Compilationseinheit (Compilation Unit)	252
3.6.9	Statischer Import *	252
<b>3.7</b>	<b>Mit Referenzen arbeiten, Vielfalt, Identität, Gleichwertigkeit</b>	254
3.7.1	null-Referenz und die Frage der Philosophie	254
3.7.2	Alles auf null? Referenzen testen	256
3.7.3	Zuweisungen bei Referenzen	257
3.7.4	Methoden mit Referenztypen als Parameter	259
3.7.5	Identität von Objekten	263
3.7.6	Gleichwertigkeit und die Methode equals(...)	263
<b>3.8</b>	<b>Zum Weiterlesen</b>	265

## **4 Arrays und ihre Anwendungen** 267

<b>4.1</b>	<b>Einfache Feldarbeit</b>	267
4.1.1	Grundbestandteile	268
4.1.2	Deklaration von Array-Variablen	268
4.1.3	Array-Objekte mit new erzeugen	270
4.1.4	Arrays mit { Inhalt }	270
4.1.5	Die Länge eines Arrays über die Objektvariable length auslesen	271
4.1.6	Zugriff auf die Elemente über den Index	272
4.1.7	Typische Array-Fehler	274
4.1.8	Arrays an Methoden übergeben	275
4.1.9	Mehrere Rückgabewerte *	276
4.1.10	Vorinitialisierte Arrays	277
<b>4.2</b>	<b>Die erweiterte for-Schleife</b>	278
<b>4.3</b>	<b>Methode mit variabler Argumentanzahl (Varargs)</b>	283
4.3.1	System.out.printf(...) nimmt eine beliebige Anzahl von Argumenten an	283
4.3.2	Durchschnitt finden von variablen Argumenten	283
4.3.3	Varargs-Designtipps *	285

<b>4.4</b>	<b>Mehrdimensionale Arrays *</b> .....	285
4.4.1	Nichtrechteckige Arrays *	288
<b>4.5</b>	<b>Bibliotheksunterstützung von Arrays</b> .....	291
4.5.1	Klonen kann sich lohnen – Arrays vermehren .....	291
4.5.2	Warum »können« Arrays so wenig? .....	292
4.5.3	Array-Inhalte kopieren .....	292
<b>4.6</b>	<b>Die Klasse Arrays zum Vergleichen, Füllen, Suchen und Sortieren nutzen</b> .....	293
4.6.1	Eine lange Schlange .....	307
<b>4.7</b>	<b>Der Einstiegspunkt für das Laufzeitsystem: main(...)</b> .....	310
4.7.1	Korrekte Deklaration der Startmethode .....	310
4.7.2	Kommandozeilenargumente verarbeiten .....	311
4.7.3	Der Rückgabetyt von main(...) und System.exit(int) * .....	311
<b>4.8</b>	<b>Zum Weiterlesen</b> .....	313
 <b>5 Der Umgang mit Zeichen und Zeichenketten</b> .....		315
<b>5.1</b>	<b>Von ASCII über ISO-8859-1 zu Unicode</b> .....	315
5.1.1	ASCII .....	315
5.1.2	ISO/IEC 8859-1 .....	316
5.1.3	Unicode .....	317
5.1.4	Unicode-Zeichenkodierung .....	319
5.1.5	Escape-Sequenzen/Fluchtsymbole .....	319
5.1.6	Schreibweise für Unicode-Zeichen und Unicode-Escapes .....	320
5.1.7	Java-Versionen gehen mit dem Unicode-Standard Hand in Hand * .....	322
<b>5.2</b>	<b>Datentypen für Zeichen und Zeichenfolgen</b> .....	324
<b>5.3</b>	<b>Die Character-Klasse</b> .....	324
5.3.1	Ist das so? .....	325
5.3.2	Zeichen in Großbuchstaben/Kleinbuchstaben konvertieren .....	327
5.3.3	Vom Zeichen zum String .....	328
5.3.4	Von char in int: vom Zeichen zur Zahl * .....	328
<b>5.4</b>	<b>Zeichenfolgen</b> .....	330
<b>5.5</b>	<b>Die Klasse String und ihre Methoden</b> .....	332
5.5.1	String-Literale als String-Objekte für konstante Zeichenketten .....	332
5.5.2	Konkatenation mit + .....	333
5.5.3	Mehrzeilige Textblöcke mit "" .....	333
5.5.4	String-Länge und Test auf Leer-String .....	338
5.5.5	Zugriff auf ein bestimmtes Zeichen mit charAt(int) .....	339

5.5.6	Nach enthaltenen Zeichen und Zeichenfolgen suchen .....	340
5.5.7	Das Hangman-Spiel .....	343
5.5.8	Gut, dass wir verglichen haben .....	345
5.5.9	String-Teile extrahieren .....	349
5.5.10	Strings anhängen, zusammenfügen, Groß-/Kleinschreibung und Weißraum .....	354
5.5.11	Gesucht, gefunden, ersetzt .....	357
5.5.12	String-Objekte mit Konstruktoren und aus Wiederholungen erzeugen * .....	359
<b>5.6</b>	<b>Veränderbare Zeichenketten mit StringBuilder und StringBuffer</b> .....	<b>363</b>
5.6.1	Anlegen von StringBuilder-Objekten .....	364
5.6.2	StringBuilder in andere Zeichenkettenformate konvertieren .....	365
5.6.3	Zeichen(folgen) erfragen .....	365
5.6.4	Daten anhängen .....	365
5.6.5	Zeichen(folgen) setzen, löschen und umdrehen .....	367
5.6.6	Länge und Kapazität eines StringBuilder-Objekts * .....	370
5.6.7	Vergleich von StringBuilder-Exemplaren und Strings mit StringBuilder .....	371
5.6.8	hashCode() bei StringBuilder * .....	373
<b>5.7</b>	<b>CharSequence als Basistyp</b> .....	<b>373</b>
<b>5.8</b>	<b>Konvertieren zwischen Primitiven und Strings</b> .....	<b>376</b>
5.8.1	Unterschiedliche Typen in String-Repräsentationen konvertieren .....	376
5.8.2	String-Inhalt in einen primitiven Wert konvertieren .....	378
5.8.3	String-Repräsentation im Format Binär, Hex und Oktal * .....	380
5.8.4	parse*(...)- und print*()-Methoden in DatatypeConverter * .....	384
<b>5.9</b>	<b>Strings zusammenhängen (konkateneren)</b> .....	<b>384</b>
5.9.1	Strings mit StringJoiner zusammenhängen .....	385
<b>5.10</b>	<b>Zerlegen von Zeichenketten</b> .....	<b>387</b>
5.10.1	Splitten von Zeichenketten mit split(...) .....	387
5.10.2	Yes we can, yes we scan – die Klasse Scanner .....	388
<b>5.11</b>	<b>Ausgaben formatieren</b> .....	<b>392</b>
5.11.1	Formatieren und Ausgeben mit format() .....	392
<b>5.12</b>	<b>Zum Weiterlesen</b> .....	<b>398</b>
<b>6</b>	<b>Eigene Klassen schreiben</b> .....	<b>399</b>
<b>6.1</b>	<b>Eigene Klassen mit Eigenschaften deklarieren</b> .....	<b>399</b>
6.1.1	Objektvariablen deklarieren .....	400
6.1.2	Methoden deklarieren .....	403



6.1.3	Verdeckte (shadowed) Variablen .....	406
6.1.4	Die this-Referenz .....	407
<b>6.2</b>	<b>Privatsphäre und Sichtbarkeit .....</b>	<b>411</b>
6.2.1	Für die Öffentlichkeit: public .....	411
6.2.2	Kein Public Viewing – Passwörter sind privat .....	411
6.2.3	Wieso nicht freie Methoden und Variablen für alle? .....	413
6.2.4	Privat ist nicht ganz privat: Es kommt darauf an, wer's sieht * .....	413
6.2.5	Zugriffsmethoden für Objektvariablen deklarieren .....	414
6.2.6	Setter und Getter nach der JavaBeans-Spezifikation .....	415
6.2.7	Paketsichtbar .....	417
6.2.8	Zusammenfassung zur Sichtbarkeit .....	419
<b>6.3</b>	<b>Eine für alle – statische Methoden und Klassenvariablen .....</b>	<b>421</b>
6.3.1	Warum statische Eigenschaften sinnvoll sind .....	422
6.3.2	Statische Eigenschaften mit static .....	423
6.3.3	Statische Eigenschaften über Referenzen nutzen? * .....	424
6.3.4	Warum die Groß- und Kleinschreibung wichtig ist * .....	425
6.3.5	Statische Variablen zum Datenaustausch * .....	426
6.3.6	Statische Eigenschaften und Objekteigenschaften * .....	427
<b>6.4</b>	<b>Konstanten und Aufzählungen .....</b>	<b>428</b>
6.4.1	Konstanten über statische finale Variablen .....	428
6.4.2	Typunsichere Aufzählungen .....	429
6.4.3	Aufzählungstypen: typsichere Aufzählungen mit enum .....	431
<b>6.5</b>	<b>Objekte anlegen und zerstören .....</b>	<b>436</b>
6.5.1	Konstruktoren schreiben .....	436
6.5.2	Verwandschaft von Methode und Konstruktor .....	438
6.5.3	Der Standard-Konstruktor (default constructor) .....	439
6.5.4	Parametrisierte und überladene Konstruktoren .....	440
6.5.5	Copy-Konstruktor .....	443
6.5.6	Einen anderen Konstruktor der gleichen Klasse mit this(...) aufrufen .....	444
6.5.7	Immutable-Objekte und Wither-Methoden .....	447
6.5.8	Ihr fehlt uns nicht – der Garbage-Collector .....	449
<b>6.6</b>	<b>Klassen- und Objektinitialisierung * .....</b>	<b>451</b>
6.6.1	Initialisierung von Objektvariablen .....	451
6.6.2	Statische Blöcke als Klasseninitialisierer .....	453
6.6.3	Initialisierung von Klassenvariablen .....	453
6.6.4	Eincompilierte Belegungen der Klassenvariablen .....	454
6.6.5	Exemplarinitialisierer (Instanzinitialisierer) .....	455
6.6.6	Finale Werte im Konstruktor und in statischen Blöcken setzen .....	458
<b>6.7</b>	<b>Zum Weiterlesen .....</b>	<b>460</b>

<b>7</b>	<b>Objektorientierte Beziehungsfragen</b>	461
<b>7.1</b>	<b>Assoziationen zwischen Objekten</b>	461
7.1.1	Unidirektionale 1:1-Beziehung	462
7.1.2	Zwei Freunde müsst ihr werden – bidirektionale 1:1-Beziehungen	463
7.1.3	Unidirektionale 1:n-Beziehung	465
<b>7.2</b>	<b>Vererbung</b>	471
7.2.1	Vererbung in Java	472
7.2.2	Ereignisse modellieren	472
7.2.3	Die implizite Basisklasse java.lang.Object	474
7.2.4	Einfach- und Mehrfachvererbung *	475
7.2.5	Sehen Kinder alles? Die Sichtbarkeit protected	475
7.2.6	Konstruktoren in der Vererbung und super(...)	476
<b>7.3</b>	<b>Typen in Hierarchien</b>	481
7.3.1	Automatische und explizite Typumwandlung	481
7.3.2	Das Substitutionsprinzip	485
7.3.3	Typen mit dem instanceof-Operator testen	487
7.3.4	Pattern-Matching bei instanceof	489
<b>7.4</b>	<b>Methoden überschreiben</b>	491
7.4.1	Methoden in Unterklassen mit neuem Verhalten ausstatten	492
7.4.2	Mit super an die Eltern	496
<b>7.5</b>	<b>Drum prüfe, wer sich dynamisch bindet</b>	498
7.5.1	Gebunden an toString()	498
7.5.2	Implementierung von System.out.println(Object)	500
<b>7.6</b>	<b>Finale Klassen und finale Methoden</b>	501
7.6.1	Finale Klassen	501
7.6.2	Nicht überschreibbare (finale) Methoden	501
<b>7.7</b>	<b>Abstrakte Klassen und abstrakte Methoden</b>	503
7.7.1	Abstrakte Klassen	503
7.7.2	Abstrakte Methoden	505
<b>7.8</b>	<b>Weiteres zum Überschreiben und dynamischen Binden</b>	511
7.8.1	Nicht dynamisch gebunden bei privaten, statischen und finalen Methoden	511
7.8.2	Kovariante Rückgabetypen	512
7.8.3	Array-Typen und Kovarianz *	513
7.8.4	Dynamisch gebunden auch bei Konstruktoraufrufen *	514
7.8.5	Keine dynamische Bindung bei überdeckten Objektvariablen *	516
<b>7.9</b>	<b>Zum Weiterlesen und Programmieraufgabe</b>	517

---

<b>8</b>	<b>Schnittstellen, Aufzählungen, versiegelte Klassen, Records</b>	519
<b>8.1</b>	<b>Schnittstellen</b>	519
8.1.1	Schnittstellen sind neue Typen	519
8.1.2	Schnittstellen deklarieren	520
8.1.3	Abstrakte Methoden in Schnittstellen	520
8.1.4	Implementieren von Schnittstellen	521
8.1.5	Ein Polymorphie-Beispiel mit Schnittstellen	523
8.1.6	Die Mehrfachvererbung bei Schnittstellen	525
8.1.7	Keine Kollisionsgefahr bei Mehrfachvererbung *	528
8.1.8	Erweitern von Interfaces – Subinterfaces	529
8.1.9	Konstantendeklarationen bei Schnittstellen	530
8.1.10	Nachträgliches Implementieren von Schnittstellen *	531
8.1.11	Statische ausprogrammierte Methoden in Schnittstellen	531
8.1.12	Erweitern und Ändern von Schnittstellen	533
8.1.13	Default-Methoden	535
8.1.14	Erweiterte Schnittstellen deklarieren und nutzen	536
8.1.15	Öffentliche und private Schnittstellenmethoden	540
8.1.16	Erweiterte Schnittstellen, Mehrfachvererbung und Mehrdeutigkeiten *	540
8.1.17	Bausteine bilden mit Default-Methoden *	545
8.1.18	Markierungsschnittstellen *	550
8.1.19	(Abstrakte) Klassen und Schnittstellen im Vergleich	551
<b>8.2</b>	<b>Aufzählungstypen</b>	552
8.2.1	Methoden auf Enum-Objekten	552
8.2.2	Aufzählungen mit eigenen Methoden und Initialisierern *	556
8.2.3	enum mit eigenen Konstruktoren *	558
<b>8.3</b>	<b>Versiegelte Klassen und Schnittstellen</b>	562
8.3.1	Versiegelte Klassen und Schnittstellen (sealed classes/interfaces)	564
8.3.2	Unterklassen sind final, sealed, non-sealed	566
8.3.3	Abkürzende Schreibweisen	566
<b>8.4</b>	<b>Records</b>	567
8.4.1	Einfache Records	568
8.4.2	Records mit Methoden	569
8.4.3	Konstruktoren von Records anpassen	571
8.4.4	Konstruktoren ergänzen	573
8.4.5	Versiegelte Schnittstellen und Records	574
8.4.6	Zusammenfassung	575
<b>8.5</b>	<b>Zum Weiterlesen</b>	576

<b>9</b>	<b>Ausnahmen müssen sein</b>	<b>577</b>
<b>9.1</b>	<b>Problembereiche einzäunen</b>	<b>578</b>
9.1.1	Exceptions in Java mit try und catch	578
9.1.2	Geprüfte und ungeprüfte Ausnahmen	579
9.1.3	Eine NumberFormatException fliegt (ungeprüfte Ausnahme)	579
9.1.4	UUID in Textdatei anhängen (geprüfte Ausnahme)	581
9.1.5	Wiederholung abgebrochener Bereiche *	584
9.1.6	Bitte nicht schlucken – leere catch-Blöcke	584
9.1.7	Mehrere Ausnahmen auffangen	585
9.1.8	Zusammenfassen gleicher catch-Blöcke mit dem multi-catch	586
<b>9.2</b>	<b>Ausnahmen weiterleiten, throws am Kopf von Methoden/Konstruktoren</b>	<b>587</b>
9.2.1	throws bei Konstruktoren und Methoden	587
<b>9.3</b>	<b>Die Klassenhierarchie der Ausnahmen</b>	<b>588</b>
9.3.1	Eigenschaften des Exception-Objekts	588
9.3.2	Basistyp Throwable	589
9.3.3	Die Exception-Hierarchie	590
9.3.4	Oberausnahmen auffangen	591
9.3.5	Schon gefangen?	593
9.3.6	Ablauf einer Ausnahmesituation	594
9.3.7	Nicht zu allgemein fangen!	594
9.3.8	Bekannte RuntimeException-Klassen	596
9.3.9	Kann man abfangen, muss man aber nicht	597
<b>9.4</b>	<b>Abschlussbehandlung mit finally</b>	<b>597</b>
<b>9.5</b>	<b>Auslösen eigener Exceptions</b>	<b>603</b>
9.5.1	Mit throw Ausnahmen auslösen	603
9.5.2	Vorhandene Runtime-Ausnahmetypen kennen und nutzen	605
9.5.3	Parameter testen und gute Fehlermeldungen	608
9.5.4	Neue Exception-Klassen deklarieren	609
9.5.5	Eigene Ausnahmen als Unterklassen von Exception oder RuntimeException?	611
9.5.6	Ausnahmen abfangen und weiterleiten *	614
9.5.7	Aufruf-Stack von Ausnahmen verändern *	615
9.5.8	Präzises rethrow *	616
9.5.9	Geschachtelte Ausnahmen *	619
<b>9.6</b>	<b>try mit Ressourcen (automatisches Ressourcen-Management)</b>	<b>623</b>
9.6.1	try mit Ressourcen	623
9.6.2	Die Schnittstelle AutoCloseable	624
9.6.3	Ausnahmen vom close()	625

9.6.4	Typen, die AutoCloseable und Closeable sind .....	626
9.6.5	Mehrere Ressourcen nutzen .....	628
9.6.6	try mit Ressourcen auf null-Ressourcen .....	629
9.6.7	Unterdrückte Ausnahmen * .....	629
<b>9.7</b>	<b>Besonderheiten bei der Ausnahmebehandlung *</b> .....	<b>633</b>
9.7.1	Rückgabewerte bei ausgelösten Ausnahmen .....	633
9.7.2	Ausnahmen und Rückgaben verschwinden – das Duo return und finally .....	633
9.7.3	throws bei überschriebenen Methoden .....	634
9.7.4	Nicht erreichbare catch-Klauseln .....	637
<b>9.8</b>	<b>Harte Fehler – Error *</b> .....	<b>638</b>
<b>9.9</b>	<b>Assertions *</b> .....	<b>639</b>
9.9.1	Assertions in eigenen Programmen nutzen .....	639
9.9.2	Assertions aktivieren und Laufzeit-Errors .....	639
9.9.3	Assertions feiner aktivieren oder deaktivieren .....	641
<b>9.10</b>	<b>Zum Weiterlesen</b> .....	<b>642</b>
<b>10</b>	<b>Geschachtelte Typen</b> .....	<b>643</b>
<b>10.1</b>	<b>Geschachtelte Klassen, Schnittstellen und Aufzählungen</b> .....	<b>643</b>
<b>10.2</b>	<b>Statische geschachtelte Typen</b> .....	<b>645</b>
<b>10.3</b>	<b>Nichtstatische geschachtelte Typen</b> .....	<b>647</b>
10.3.1	Exemplare innerer Klassen erzeugen .....	647
10.3.2	Die this-Referenz .....	648
10.3.3	Vom Compiler generierte Klassendateien * .....	649
10.3.4	Erlaubte Modifizierer bei äußeren und inneren Klassen .....	650
<b>10.4</b>	<b>Lokale Klassen</b> .....	<b>650</b>
10.4.1	Beispiel mit eigener Klassendeklaration .....	650
10.4.2	Lokale Klasse für einen Timer nutzen .....	651
<b>10.5</b>	<b>Anonyme innere Klassen</b> .....	<b>652</b>
10.5.1	Nutzung einer anonymen inneren Klasse für den Timer .....	653
10.5.2	Umsetzung innerer anonymer Klassen * .....	654
10.5.3	Konstruktoren innerer anonymer Klassen .....	654
<b>10.6</b>	<b>Zugriff auf lokale Variablen aus lokalen und anonymen Klassen *</b> .....	<b>657</b>
<b>10.7</b>	<b>this in Unterklassen *</b> .....	<b>658</b>
10.7.1	Geschachtelte Klassen greifen auf private Eigenschaften zu .....	659

<b>10.8 Nester</b> .....	660
<b>10.9 Zum Weiterlesen</b> .....	662
<b>11 Besondere Typen der Java SE</b> .....	663
<b>11.1 Object ist die Mutter aller Klassen</b> .....	664
11.1.1 Klassenobjekte .....	664
11.1.2 Objektidentifikation mit toString() .....	665
11.1.3 Objektgleichwertigkeit mit equals(...) und Identität .....	667
11.1.4 Klonen eines Objekts mit clone() * .....	673
11.1.5 Hashwerte über hashCode() liefern * .....	678
11.1.6 System.identityHashCode(...) und das Problem der nicht eindeutigen Objektverweise * .....	685
11.1.7 Aufräumen mit finalize() * .....	686
11.1.8 Synchronisation * .....	688
<b>11.2 Schwache Referenzen und Cleaner</b> .....	688
<b>11.3 Die Utility-Klasse java.util.Objects</b> .....	690
11.3.1 Eingebaute null-Tests für equals(...)/hashCode() .....	690
11.3.2 Objects.toString(...) .....	691
11.3.3 null-Prüfungen mit eingebauter Ausnahmebehandlung .....	691
11.3.4 Tests auf null .....	692
11.3.5 Indexbezogene Programmargumente auf Korrektheit prüfen .....	693
<b>11.4 Vergleichen von Objekten und Ordnung herstellen</b> .....	694
11.4.1 Natürlich geordnet oder nicht? .....	694
11.4.2 compare*()-Methode der Schnittstellen Comparable und Comparator .....	695
11.4.3 Rückgabewerte kodieren die Ordnung .....	696
11.4.4 Beispiel-Comparator: den kleinsten Raum einer Sammlung finden .....	696
11.4.5 Tipps für Comparator- und Comparable-Implementierungen .....	698
11.4.6 Statische und Default-Methoden in Comparator .....	699
<b>11.5 Wrapper-Klassen und Autoboxing</b> .....	703
11.5.1 Wrapper-Objekte erzeugen .....	704
11.5.2 Konvertierungen in eine String-Repräsentation .....	706
11.5.3 Von einer String-Repräsentation parsen .....	707
11.5.4 Die Basisklasse Number für numerische Wrapper-Objekte .....	707
11.5.5 Vergleiche durchführen mit compare*(...), compareTo(...), equals(...) und Hashwerten .....	709
11.5.6 Statische Reduzierungsmethoden in Wrapper-Klassen .....	712
11.5.7 Konstanten für die Größe eines primitiven Typs .....	713

11.5.8	Behandeln von vorzeichenlosen Zahlen *	714
11.5.9	Die Klasse Integer	715
11.5.10	Die Klassen Double und Float für Fließkommazahlen	716
11.5.11	Die Long-Klasse	717
11.5.12	Die Boolean-Klasse	717
11.5.13	Autoboxing: Boxing und Unboxing	718
<b>11.6</b>	<b>Iterator, Iterable *</b>	<b>723</b>
11.6.1	Die Schnittstelle Iterator	723
11.6.2	Wer den Iterator liefert	726
11.6.3	Die Schnittstelle Iterable	727
11.6.4	Erweitertes for und Iterable	727
11.6.5	Interne Iteration	728
11.6.6	Ein eigenes Iterable implementieren *	729
<b>11.7</b>	<b>Annotations in der Java SE</b>	<b>730</b>
11.7.1	Orte für Annotationen	730
11.7.2	Annotationstypen aus java.lang	731
11.7.3	@Deprecated	732
11.7.4	Annotationen mit zusätzlichen Informationen	732
11.7.5	@SuppressWarnings	733
<b>11.8</b>	<b>Zum Weiterlesen</b>	<b>736</b>

## 12 Generics<T> 737

---

<b>12.1</b>	<b>Einführung in Java Generics</b>	<b>737</b>
12.1.1	Mensch versus Maschine – Typprüfung des Compilers und der Laufzeitumgebung	737
12.1.2	Raketen	738
12.1.3	Generische Typen deklarieren	740
12.1.4	Generics nutzen	742
12.1.5	Diamonds are forever	744
12.1.6	Generische Schnittstellen	747
12.1.7	Generische Methoden/Konstruktoren und Typ-Inferenz	749
<b>12.2</b>	<b>Umsetzen der Generics, Typlöschung und Raw-Types</b>	<b>753</b>
12.2.1	Realisierungsmöglichkeiten	753
12.2.2	Typlöschung (Type Erasure)	753
12.2.3	Probleme der Typlöschung	755
12.2.4	Raw-Type	760

<b>12.3 Die Typen über Bounds einschränken</b> .....	762
12.3.1 Einfache Einschränkungen mit extends .....	762
12.3.2 Weitere Obertypen mit & .....	765
<b>12.4 Typparameter in der throws-Klausel *</b> .....	765
12.4.1 Deklaration einer Klasse mit Typvariable <E extends Exception> .....	765
12.4.2 Parametrisierter Typ bei Typvariable <E extends Exception> .....	766
<b>12.5 Generics und Vererbung, Invarianz</b> .....	769
12.5.1 Arrays sind kovariant .....	769
12.5.2 Generics sind nicht kovariant, sondern invariant .....	769
12.5.3 Wildcards mit ? .....	770
12.5.4 Bounded Wildcards .....	773
12.5.5 Bounded-Wildcard-Typen und Bounded-Typvariablen .....	776
12.5.6 Das LESS-Prinzip .....	778
12.5.7 Enum<E extends Enum<E>> * .....	781
<b>12.6 Konsequenzen der Typlöschung: Typ-Token, Arrays und Brücken *</b> .....	783
12.6.1 Typ-Token .....	783
12.6.2 Super-Type-Token .....	784
12.6.3 Generics und Arrays .....	786
12.6.4 Brückenmethoden .....	787
<b>12.7 Zum Weiterlesen</b> .....	792

## 13 Lambda-Ausdrücke und funktionale Programmierung

793

<b>13.1 Funktionale Schnittstellen und Lambda-Ausdrücke</b> .....	793
13.1.1 Klassen implementieren Schnittstellen .....	793
13.1.2 Lambda-Ausdrücke implementieren Schnittstellen .....	795
13.1.3 Funktionale Schnittstellen .....	796
13.1.4 Der Typ eines Lambda-Ausdrucks ergibt sich durch den Zieltyp .....	797
13.1.5 Annotation @FunctionalInterface .....	802
13.1.6 Syntax für Lambda-Ausdrücke .....	803
13.1.7 Die Umgebung der Lambda-Ausdrücke und Variablenzugriffe .....	808
13.1.8 Ausnahmen in Lambda-Ausdrücken .....	814
13.1.9 Klassen mit einer abstrakten Methode als funktionale Schnittstelle? * .....	817
<b>13.2 Methodenreferenz</b> .....	819
13.2.1 Motivation .....	819
13.2.2 Methodenreferenzen mit :: .....	819
13.2.3 Varianten von Methodenreferenzen .....	820



<b>13.3</b>	<b>Konstruktorreferenz</b>	823
13.3.1	Parameterlose und parametrisierte Konstruktoren	825
13.3.2	Nützliche vordefinierte Schnittstellen für Konstruktorreferenzen	825
<b>13.4</b>	<b>Funktionale Programmierung</b>	827
13.4.1	Code = Daten	827
13.4.2	Programmierparadigmen: imperativ oder deklarativ	828
13.4.3	Das Wesen der funktionalen Programmierung	829
13.4.4	Funktionale Programmierung und funktionale Programmiersprachen	831
13.4.5	Funktionen höherer Ordnung am Beispiel von Comparator	834
13.4.6	Lambda-Ausdrücke als Abbildungen bzw. Funktionen betrachten	834
<b>13.5</b>	<b>Funktionale Schnittstellen aus dem java.util.function-Paket</b>	835
13.5.1	Blöcke mit Code und die funktionale Schnittstelle Consumer	836
13.5.2	Supplier	838
13.5.3	Prädikate und java.util.function.Predicate	838
13.5.4	Funktionen über die funktionale Schnittstelle java.util.function.Function	840
13.5.5	Ein bisschen Bi ...	844
13.5.6	Funktionale Schnittstellen mit Primitiven	847
<b>13.6</b>	<b>Optional ist keine Nullnummer</b>	850
13.6.1	Einsatz von null	850
13.6.2	Der Optional-Typ	853
13.6.3	Erst mal funktional mit Optional	855
13.6.4	Primitiv-Optionales mit speziellen Optional*-Klassen	858
<b>13.7</b>	<b>Was ist jetzt so funktional?</b>	861
<b>13.8</b>	<b>Zum Weiterlesen</b>	863
<b>14</b>	<b>Architektur, Design und angewandte Objektorientierung</b>	865
<b>14.1</b>	<b>SOLIDe Modellierung</b>	865
14.1.1	DRY, KISS und YAGNI	866
14.1.2	SOLID	866
14.1.3	Sei nicht STUPID	868
<b>14.2</b>	<b>Architektur, Design und Implementierung</b>	869
<b>14.3</b>	<b>Design-Patterns (Entwurfsmuster)</b>	870
14.3.1	Motivation für Design-Patterns	870
14.3.2	Singleton	871

14.3.3	Fabrikmethoden .....	872
14.3.4	Das Beobachter-Pattern mit Listener realisieren .....	873
<b>14.4</b>	<b>Zum Weiterlesen .....</b>	<b>877</b>

---

## **15 Java Platform Module System** 879

---

<b>15.1</b>	<b>Klassenlader (Class Loader) und Modul-/Klassenpfad .....</b>	<b>879</b>
15.1.1	Klassenladen auf Abruf .....	879
15.1.2	Klassenlader bei der Arbeit zusehen .....	880
15.1.3	JMOD-Dateien und JAR-Dateien .....	881
15.1.4	Woher die Klassen kommen: Suchorte und spezielle Klassenlader .....	882
15.1.5	Setzen des Modulpfades .....	883
<b>15.2</b>	<b>Module entwickeln und einbinden .....</b>	<b>885</b>
15.2.1	Wer sieht wen? .....	885
15.2.2	Plattform-Module und ein JMOD-Beispiel .....	886
15.2.3	Interne Plattformeigenschaften nutzen, --add-exports .....	887
15.2.4	Neue Module einbinden, --add-modules und --add-opens .....	889
15.2.5	Projektabhängigkeiten in Eclipse .....	891
15.2.6	Benannte Module und module-info.java .....	893
15.2.7	Automatische Module .....	897
15.2.8	Unbenanntes Modul .....	898
15.2.9	Lesbarkeit und Zugreifbarkeit .....	898
15.2.10	Modul-Migration .....	899
<b>15.3</b>	<b>Zum Weiterlesen .....</b>	<b>901</b>

---

## **16 Die Klassenbibliothek** 903

---

<b>16.1</b>	<b>Die Java-Klassenphilosophie .....</b>	<b>903</b>
16.1.1	Modul, Paket, Typ .....	903
16.1.2	Übersicht über die Pakete der Standardbibliothek .....	906
<b>16.2</b>	<b>Einfache Zeitmessung und Profiling * .....</b>	<b>910</b>
<b>16.3</b>	<b>Die Klasse Class .....</b>	<b>913</b>
16.3.1	An ein Class-Objekt kommen .....	914
16.3.2	Eine Class ist ein Type .....	916
<b>16.4</b>	<b>Klassenlader .....</b>	<b>917</b>
16.4.1	Die Klasse java.lang.ClassLoader .....	918

<b>16.5 Die Utility-Klassen System und Properties</b> .....	918
16.5.1 Speicher der JVM .....	919
16.5.2 Anzahl der CPUs bzw. Kerne .....	920
16.5.3 Systemeigenschaften der Java-Umgebung .....	921
16.5.4 Eigene Properties von der Konsole aus setzen * .....	922
16.5.5 Zeilenumbruchzeichen, line.separator .....	925
16.5.6 Umgebungsvariablen des Betriebssystems .....	925
<b>16.6 Sprachen der Länder</b> .....	927
16.6.1 Sprachen in Regionen über Locale-Objekte .....	927
<b>16.7 Wichtige Datum-Klassen im Überblick</b> .....	931
16.7.1 Der 1.1.1970 .....	932
16.7.2 System.currentTimeMillis() .....	932
16.7.3 Einfache Zeitumrechnungen durch TimeUnit .....	933
<b>16.8 Date-Time-API</b> .....	934
16.8.1 Menschenzeit und Maschinenzeit .....	935
16.8.2 Die Datumsklasse LocalDate .....	938
<b>16.9 Logging mit Java</b> .....	939
16.9.1 Logging-APIs .....	939
16.9.2 Logging mit java.util.logging .....	940
<b>16.10 Maven: Build-Management und Abhängigkeiten auflösen</b> .....	942
16.10.1 Beispielprojekt in Eclipse mit Maven .....	943
16.10.2 Properties hinzunehmen .....	943
16.10.3 Dependency hinzunehmen .....	944
16.10.4 Lokales und das Remote-Repository .....	945
16.10.5 Lebenszyklus, Phasen und Maven-Plugins .....	946
16.10.6 Archetypes .....	946
<b>16.11 Zum Weiterlesen</b> .....	946
<b>17 Einführung in die nebenläufige Programmierung</b> .....	949
<hr/>	
<b>17.1 Nebenläufigkeit und Parallelität</b> .....	949
17.1.1 Multitasking, Prozesse und Threads .....	950
17.1.2 Threads und Prozesse .....	950
17.1.3 Wie nebenläufige Programme die Geschwindigkeit steigern können .....	952
17.1.4 Was Java für Nebenläufigkeit alles bietet .....	953
<b>17.2 Existierende Threads und neue Threads erzeugen</b> .....	954
17.2.1 Main-Thread .....	954

17.2.2	Wer bin ich? .....	954
17.2.3	Die Schnittstelle Runnable implementieren .....	955
17.2.4	Thread mit Runnable starten .....	956
17.2.5	Runnable parametrisieren .....	958
17.2.6	Die Klasse Thread erweitern .....	958
<b>17.3</b>	<b>Thread-Eigenschaften und Zustände .....</b>	<b>961</b>
17.3.1	Der Name eines Threads .....	961
17.3.2	Die Zustände eines Threads * .....	962
17.3.3	Schläfer gesucht .....	962
17.3.4	Wann Threads fertig sind .....	964
17.3.5	Einen Thread höflich mit Interrupt beenden .....	964
17.3.6	Unbehandelte Ausnahmen, Thread-Ende und UncaughtExceptionHandler .....	967
17.3.7	Der stop() von außen und die Rettung mit ThreadDeath * .....	968
17.3.8	Ein Rendezvous mit join(...) * .....	970
17.3.9	Arbeit niederlegen und wieder aufnehmen * .....	972
17.3.10	Priorität * .....	972
<b>17.4</b>	<b>Der Ausführer (Executor) kommt .....</b>	<b>974</b>
17.4.1	Die Schnittstelle Executor .....	974
17.4.2	Glücklich in der Gruppe – die Thread-Pools .....	976
17.4.3	Threads mit Rückgabe über Callable .....	978
17.4.4	Erinnerungen an die Zukunft – die Future-Rückgabe .....	980
17.4.5	Mehrere Callable-Objekte abarbeiten .....	983
17.4.6	CompletionService und ExecutorCompletionService .....	984
17.4.7	ScheduledExecutorService: wiederholende Aufgaben und Zeitsteuerungen .....	986
17.4.8	Asynchrones Programmieren mit CompletableFuture (CompletionStage) ....	986
<b>17.5</b>	<b>Zum Weiterlesen .....</b>	<b>989</b>
<b>18</b>	<b>Einführung in Datenstrukturen und Algorithmen .....</b>	<b>991</b>
<b>18.1</b>	<b>Listen .....</b>	<b>991</b>
18.1.1	Erstes Listen-Beispiel .....	992
18.1.2	Auswahlkriterium ArrayList oder LinkedList .....	993
18.1.3	Die Schnittstelle List .....	993
18.1.4	ArrayList .....	1000
18.1.5	LinkedList .....	1002
18.1.6	Der Array-Adapter Arrays.asList(...) .....	1003
18.1.7	ListIterator * .....	1005

18.1.8	toArray(...) von Collection verstehen – die Gefahr einer Falle erkennen .....	1006
18.1.9	Primitive Elemente in Datenstrukturen verwalten .....	1010
<b>18.2</b>	<b>Mengen (Sets)</b> .....	1010
18.2.1	Ein erstes Mengen-Beispiel .....	1011
18.2.2	Methoden der Schnittstelle Set .....	1013
18.2.3	HashSet .....	1015
18.2.4	TreeSet – die sortierte Menge .....	1015
18.2.5	Die Schnittstellen NavigableSet und SortedSet .....	1017
18.2.6	LinkedHashSet .....	1019
<b>18.3</b>	<b>Assoziative Speicher</b> .....	1021
18.3.1	Die Klassen HashMap und TreeMap .....	1021
18.3.2	Einfügen und Abfragen des Assoziativspeichers .....	1024
<b>18.4</b>	<b>Java-Stream-API</b> .....	1026
18.4.1	Deklaratives Programmieren .....	1026
18.4.2	Interne versus externe Iteration .....	1027
18.4.3	Was ist ein Stream? .....	1028
<b>18.5</b>	<b>Einen Stream erzeugen</b> .....	1029
18.5.1	Parallele oder sequenzielle Streams .....	1032
<b>18.6</b>	<b>Terminale Operationen</b> .....	1033
18.6.1	Die Anzahl der Elemente .....	1033
18.6.2	Und jetzt alle – forEach*(...) .....	1034
18.6.3	Einzelne Elemente aus dem Strom holen .....	1034
18.6.4	Existenztests mit Prädikaten .....	1035
18.6.5	Einen Strom auf sein kleinstes bzw. größtes Element reduzieren .....	1036
18.6.6	Einen Strom mit eigenen Funktionen reduzieren .....	1037
18.6.7	Ergebnisse in einen Container schreiben, Teil 1: collect(...) .....	1038
18.6.8	Ergebnisse in einen Container schreiben, Teil 2: Collector und Collectors .....	1039
18.6.9	Ergebnisse in einen Container schreiben, Teil 3: Gruppierungen .....	1041
18.6.10	Stream-Elemente in ein Array oder einen Iterator übertragen .....	1043
<b>18.7</b>	<b>Intermediäre Operationen</b> .....	1044
18.7.1	Element-Vorschau .....	1045
18.7.2	Filtern von Elementen .....	1045
18.7.3	Statusbehaftete intermediäre Operationen .....	1045
18.7.4	Präfix-Operation .....	1047
18.7.5	Abbildungen .....	1048
<b>18.8</b>	<b>Zum Weiterlesen</b> .....	1050

<b>19</b>	<b>Einführung in grafische Oberflächen</b>	<b>1051</b>
<b>19.1</b>	<b>GUI-Frameworks</b>	<b>1051</b>
19.1.1	Kommandozeile	1051
19.1.2	Grafische Benutzeroberfläche	1051
19.1.3	Abstract Window Toolkit (AWT)	1052
19.1.4	Java Foundation Classes und Swing	1052
19.1.5	JavaFX	1052
19.1.6	SWT (Standard Widget Toolkit) *	1054
<b>19.2</b>	<b>Deklarative und programmierte Oberflächen</b>	<b>1055</b>
19.2.1	GUI-Beschreibungen in JavaFX	1055
19.2.2	Deklarative GUI-Beschreibungen für Swing?	1056
<b>19.3</b>	<b>GUI-Builder</b>	<b>1056</b>
19.3.1	GUI-Builder für JavaFX	1057
19.3.2	GUI-Builder für Swing	1057
<b>19.4</b>	<b>Mit dem Eclipse WindowBuilder zur ersten Swing-Oberfläche</b>	<b>1057</b>
19.4.1	WindowBuilder installieren	1058
19.4.2	Mit WindowBuilder eine GUI-Klasse hinzufügen	1059
19.4.3	Das Layoutprogramm starten	1061
19.4.4	Grafische Oberfläche aufbauen	1062
19.4.5	Swing-Komponenten-Klassen	1065
19.4.6	Funktionalität geben	1066
<b>19.5</b>	<b>Grundlegendes zum Zeichnen</b>	<b>1069</b>
19.5.1	Die paint(Graphics)-Methode für den AWT-Frame	1069
19.5.2	Die ereignisorientierte Programmierung ändert Fensterinhalte	1071
19.5.3	Zeichnen von Inhalten auf einen JFrame	1072
19.5.4	Auffordern zum Neuzeichnen mit repaint(...)	1074
19.5.5	Java 2D-API	1074
<b>19.6</b>	<b>Zum Weiterlesen</b>	<b>1075</b>
<b>20</b>	<b>Einführung in Dateien und Datenströme</b>	<b>1077</b>
<b>20.1</b>	<b>Alte und neue Welt in java.io und java.nio</b>	<b>1077</b>
20.1.1	java.io-Paket mit File-Klasse	1077
20.1.2	NIO.2 und das java.nio-Paket	1078
20.1.3	java.io.File oder java.nio.*-Typen?	1078

<b>20.2</b>	<b>Dateisysteme und Pfade</b> .....	1079
20.2.1	FileSystem und Path .....	1079
20.2.2	Die Utility-Klasse Files .....	1085
<b>20.3</b>	<b>Dateien mit wahlfreiem Zugriff</b> .....	1088
20.3.1	Ein RandomAccessFile zum Lesen und Schreiben öffnen .....	1088
20.3.2	Aus dem RandomAccessFile lesen .....	1089
20.3.3	Schreiben mit RandomAccessFile .....	1092
20.3.4	Die Länge des RandomAccessFile .....	1092
20.3.5	Hin und her in der Datei .....	1093
<b>20.4</b>	<b>Basisklassen für die Ein-/Ausgabe</b> .....	1094
20.4.1	Die vier abstrakten Basisklassen .....	1094
20.4.2	Die abstrakte Basisklasse OutputStream .....	1095
20.4.3	Die abstrakte Basisklasse InputStream .....	1097
20.4.4	Die abstrakte Basisklasse Writer .....	1099
20.4.5	Die Schnittstelle Appendable * .....	1100
20.4.6	Die abstrakte Basisklasse Reader .....	1101
20.4.7	Die Schnittstellen Closeable, AutoCloseable und Flushable .....	1104
<b>20.5</b>	<b>Lesen aus Dateien und Schreiben in Dateien</b> .....	1106
20.5.1	Byteorientierte Datenströme über Files beziehen .....	1106
20.5.2	Zeichenorientierte Datenströme über Files beziehen .....	1107
20.5.3	Die Funktion von OpenOption bei den Files.new*(...)-Methoden .....	1109
20.5.4	Ressourcen aus dem Modulpfad und aus JAR-Dateien laden .....	1110
<b>20.6</b>	<b>Zum Weiterlesen</b> .....	1112
 <b>21 Einführung ins Datenbankmanagement mit JDBC</b>		<b>1113</b>
<hr/>		
<b>21.1</b>	<b>Relationale Datenbanken und Java-Zugriffe</b> .....	1113
21.1.1	Das relationale Modell .....	1113
21.1.2	Java-APIs zum Zugriff auf relationale Datenbanken .....	1114
21.1.3	Die JDBC-API und Implementierungen: JDBC-Treiber .....	1115
21.1.4	H2 ist das Werkzeug auf der Insel .....	1115
<b>21.2</b>	<b>Eine Beispielabfrage</b> .....	1116
21.2.1	Schritte zur Datenbankabfrage .....	1116
21.2.2	Mit Java auf die relationale Datenbank zugreifen .....	1116
<b>21.3</b>	<b>Zum Weiterlesen</b> .....	1118

<b>22</b>	<b>Bits und Bytes, Mathematisches und Geld</b>	1119
<b>22.1</b>	<b>Bits und Bytes</b>	1119
22.1.1	Die Bit-Operatoren Komplement, Und, Oder und XOR	1120
22.1.2	Repräsentation ganzer Zahlen in Java – das Zweierkomplement	1121
22.1.3	Das binäre (Basis 2), oktale (Basis 8), hexadezimale (Basis 16) Stellenwertsystem	1123
22.1.4	Auswirkung der Typumwandlung auf die Bit-Muster	1124
22.1.5	Vorzeichenlos arbeiten	1127
22.1.6	Die Verschiebeoperatoren	1129
22.1.7	Ein Bit setzen, löschen, umdrehen und testen	1132
22.1.8	Bit-Methoden der Integer- und Long-Klasse	1132
<b>22.2</b>	<b>Fließkomma-Arithmetik in Java</b>	1134
22.2.1	Spezialwerte für Unendlich, Null, NaN	1135
22.2.2	Standardnotation und wissenschaftliche Notation bei Fließkommazahlen *	1138
22.2.3	Mantisse und Exponent *	1138
<b>22.3</b>	<b>Die Eigenschaften der Klasse Math</b>	1140
22.3.1	Objektvariablen der Klasse Math	1140
22.3.2	Absolutwerte und Vorzeichen	1140
22.3.3	Maximum/Minimum	1141
22.3.4	Runden von Werten	1142
22.3.5	Rest der ganzzahligen Division *	1145
22.3.6	Division mit Rundung in Richtung negativ unendlich, alternativer Restwert *	1145
22.3.7	Multiply-Accumulate	1147
22.3.8	Wurzel- und Exponentialmethoden	1147
22.3.9	Der Logarithmus *	1149
22.3.10	Winkelmethoden *	1149
22.3.11	Zufallszahlen	1151
<b>22.4</b>	<b>Genauigkeit, Wertebereich eines Typs und Überlaufkontrolle *</b>	1151
22.4.1	Der größte und der kleinste Wert	1151
22.4.2	Überlauf und alles ganz exakt	1152
22.4.3	Was bitte macht eine ulp?	1154
<b>22.5</b>	<b>Zufallszahlen: Random, SecureRandom und SplittableRandom</b>	1156
22.5.1	Die Klasse Random	1156
22.5.2	Random-Objekte mit dem Samen aufbauen	1156
22.5.3	Einzelne Zufallszahlen erzeugen	1157
22.5.4	Pseudo-Zufallszahlen in der Normalverteilung *	1158
22.5.5	Strom von Zufallszahlen generieren *	1158



22.5.6	Die Klasse <code>SecureRandom</code> *	1160
22.5.7	<code>SplittableRandom</code> *	1160
<b>22.6</b>	<b>Große Zahlen *</b>	<b>1161</b>
22.6.1	Die Klasse <code>BigInteger</code>	1161
22.6.2	Beispiel: ganz lange Fakultäten mit <code>BigInteger</code>	1168
22.6.3	Große Fließkommazahlen mit <code>BigDecimal</code>	1169
22.6.4	Mit <code>MathContext</code> komfortabel die Rechengenauigkeit setzen	1172
22.6.5	Noch schneller rechnen durch mutable Implementierungen	1174
<b>22.7</b>	<b>Geld und Währung</b>	<b>1174</b>
22.7.1	Geldbeträge repräsentieren	1174
22.7.2	ISO 4217	1175
22.7.3	Währungen in Java repräsentieren	1175
<b>22.8</b>	<b>Zum Weiterlesen</b>	<b>1176</b>

## **23 Testen mit JUnit** 1177

---

<b>23.1</b>	<b>Softwaretests</b>	<b>1177</b>
23.1.1	Vorgehen beim Schreiben von Testfällen	1178
<b>23.2</b>	<b>Das Test-Framework JUnit</b>	<b>1178</b>
23.2.1	Test-Driven Development und Test-First	1179
23.2.2	Testen, implementieren, testen, implementieren, testen, freuen	1181
23.2.3	JUnit-Tests ausführen	1183
23.2.4	<code>assert*(...)</code> -Methoden der Klasse <code>Assertions</code>	1183
23.2.5	Exceptions testen	1186
23.2.6	Grenzen für Ausführungszeiten festlegen	1187
23.2.7	Beschriftungen mit <code>@DisplayName</code>	1188
23.2.8	Verschachtelte Tests	1188
23.2.9	Tests ignorieren	1189
23.2.10	Mit Methoden der <code>Assumptions</code> -Klasse Tests abbrechen	1189
23.2.11	Parametrisierte Tests	1189
<b>23.3</b>	<b>Java-Assertions-Bibliotheken und <code>AssertJ</code></b>	<b>1191</b>
23.3.1	<code>AssertJ</code>	1191
<b>23.4</b>	<b>Aufbau größerer Testfälle</b>	<b>1193</b>
23.4.1	Fixtures	1193
23.4.2	Sammlungen von Testklassen und Klassenorganisation	1195
<b>23.5</b>	<b>Wie gutes Design das Testen ermöglicht</b>	<b>1195</b>
<b>23.6</b>	<b>Dummy, Fake, Stub und Mock</b>	<b>1198</b>

<b>23.7</b>	<b>JUnit-Erweiterungen, Testzusätze</b> .....	1199
<b>23.8</b>	<b>Zum Weiterlesen</b> .....	1200
<b>24</b>	<b>Die Werkzeuge des JDK</b> .....	1201
<b>24.1</b>	<b>Übersicht</b> .....	1201
24.1.1	Aufbau und gemeinsame Schalter .....	1202
<b>24.2</b>	<b>Java-Quellen übersetzen</b> .....	1202
24.2.1	Der Java-Compiler des JDK .....	1202
24.2.2	Alternative Compiler .....	1203
24.2.3	Native Compiler .....	1204
<b>24.3</b>	<b>Die Java-Laufzeitumgebung</b> .....	1204
24.3.1	Schalter der JVM .....	1205
24.3.2	Der Unterschied zwischen java.exe und javaw.exe .....	1207
<b>24.4</b>	<b>Dokumentationskommentare mit Javadoc</b> .....	1207
24.4.1	Einen Dokumentationskommentar setzen .....	1208
24.4.2	Mit dem Werkzeug javadoc eine Dokumentation erstellen .....	1210
24.4.3	HTML-Tags in Dokumentationskommentaren * .....	1211
24.4.4	Generierte Dateien .....	1211
24.4.5	Dokumentationskommentare im Überblick * .....	1212
24.4.6	Javadoc und Doclets * .....	1213
24.4.7	Veraltete (deprecated) Typen und Eigenschaften .....	1214
24.4.8	Javadoc-Überprüfung mit DocLint .....	1217
<b>24.5</b>	<b>Das Archivformat JAR</b> .....	1217
24.5.1	Das Dienstprogramm jar benutzen .....	1218
24.5.2	Das Manifest .....	1219
24.5.3	Applikationen in JAR-Archiven starten .....	1219
<b>24.6</b>	<b>jlink: der Java Linker</b> .....	1220
<b>24.7</b>	<b>Zum Weiterlesen</b> .....	1221

## Anhang

<b>A</b>	<b>Java SE-Module und Paketübersicht</b> .....	1223
	Index .....	1241