

# Inhalt

Vorwort .....	31
<b>1 Java ist auch eine Sprache</b> .....	<b>49</b>
<b>1.1 Historischer Hintergrund</b> .....	<b>49</b>
<b>1.2 Warum Java gut ist – die zentralen Eigenschaften</b> .....	<b>51</b>
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 .....	54
1.2.5 Objektorientierung in Java .....	55
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! .....	60
1.2.11 Ausnahmebehandlung .....	60
1.2.12 Angebot an Bibliotheken und Werkzeugen .....	61
1.2.13 Einfache Syntax der Programmiersprache Java .....	61
1.2.14 Java ist Open Source .....	64
1.2.15 Wofür sich Java weniger eignet .....	65
1.2.16 Java im Vergleich zu anderen Sprachen * .....	66
1.2.17 Java und das Web, Applets und JavaFX .....	68
1.2.18 Features, Enhancements (Erweiterungen) und ein JSR .....	70
1.2.19 Die Entwicklung von Java und seine Zukunftsaussichten .....	71
<b>1.3 Java-Plattformen: Java SE, Java EE, Java ME und Java Card</b> .....	<b>72</b>
1.3.1 Die Java SE-Plattform .....	72
1.3.2 Java für die Kleinen .....	74
1.3.3 Java für die ganz, ganz Kleinen .....	74
1.3.4 Java für die Großen .....	75
1.3.5 Echtzeit-Java (Real-time Java) .....	75
<b>1.4 Die Installation der Java Platform, Standard Edition (Java SE)</b> .....	<b>76</b>
1.4.1 Die Java SE von Oracle .....	76
1.4.2 Download des JDK .....	77
1.4.3 Java SE unter Windows installieren .....	79
1.4.4 JDK/JRE deinstallieren .....	83

1.4.5	JDK unter Linux installieren .....	83
1.4.6	JDK unter Max OS X installieren .....	84
<b>1.5</b>	<b>Das erste Programm compilieren und testen .....</b>	<b>85</b>
1.5.1	Compilertest .....	86
1.5.2	Ein Quadratzahlen-Programm .....	87
1.5.3	Der Compilerlauf .....	88
1.5.4	Die Laufzeitumgebung .....	89
1.5.5	Häufige Compiler- und Interpreter-Probleme .....	89
<b>1.6</b>	<b>Entwicklungsumgebungen im Allgemeinen .....</b>	<b>90</b>
1.6.1	Die Entwicklungsumgebung Eclipse .....	90
1.6.2	NetBeans von Oracle .....	91
1.6.3	IntelliJ IDEA .....	92
1.6.4	Ein Wort zu Microsoft, Java und zu J++, J# .....	92
<b>1.7</b>	<b>Eclipse im Speziellen .....</b>	<b>93</b>
1.7.1	Eclipse entpacken und starten .....	94
1.7.2	Das erste Projekt anlegen .....	96
1.7.3	Verzeichnisstruktur für Java-Projekte * .....	96
1.7.4	Eine Klasse hinzufügen .....	97
1.7.5	Übersetzen und ausführen .....	98
1.7.6	Projekt einfügen, Workspace für die Programme wechseln .....	99
1.7.7	Plugins für Eclipse .....	99
<b>1.8</b>	<b>NetBeans im Speziellen .....</b>	<b>100</b>
1.8.1	NetBeans-Bundles .....	100
1.8.2	NetBeans installieren .....	101
1.8.3	NetBeans starten .....	101
1.8.4	Ein neues NetBeans-Projekt anlegen .....	101
1.8.5	Ein Java-Programm starten .....	102
<b>1.9</b>	<b>Zum Weiterlesen .....</b>	<b>103</b>
<b>2</b>	<b>Imperative Sprachkonzepte .....</b>	<b>105</b>
<hr/>		
<b>2.1</b>	<b>Elemente der Programmiersprache Java .....</b>	<b>105</b>
2.1.1	Token .....	105
2.1.2	Textkodierung durch Unicode-Zeichen .....	106
2.1.3	Bezeichner .....	106
2.1.4	Literale .....	108
2.1.5	Reservierte Schlüsselwörter .....	109

2.1.6	Zusammenfassung der lexikalischen Analyse .....	110
2.1.7	Kommentare .....	111
<b>2.2</b>	<b>Von der Klasse zur Anweisung .....</b>	<b>112</b>
2.2.1	Was sind Anweisungen? .....	113
2.2.2	Klassendeklaration .....	113
2.2.3	Die Reise beginnt am main(String[]) .....	114
2.2.4	Der erste Methodenaufruf: println(...) .....	115
2.2.5	Atomare Anweisungen und Anweisungssequenzen .....	116
2.2.6	Mehr zu print(...), println(...) und printf(...) für Bildschirmausgaben .....	116
2.2.7	Die API-Dokumentation .....	118
2.2.8	Ausdrücke .....	120
2.2.9	Ausdrucksanweisung .....	120
2.2.10	Erste Idee der Objektorientierung .....	121
2.2.11	Modifizierer .....	122
2.2.12	Gruppieren von Anweisungen mit Blöcken .....	122
<b>2.3</b>	<b>Datentypen, Typisierung, Variablen und Zuweisungen .....</b>	<b>124</b>
2.3.1	Primitive Datentypen im Überblick .....	126
2.3.2	Variablendeklarationen .....	129
2.3.3	Konsoleneingaben .....	131
2.3.4	Fließkommazahlen mit den Datentypen float und double .....	133
2.3.5	Ganzzahlige Datentypen .....	135
2.3.6	Wahrheitswerte .....	137
2.3.7	Unterstriche in Zahlen * .....	137
2.3.8	Alphanumerische Zeichen .....	138
2.3.9	Gute Namen, schlechte Namen .....	139
2.3.10	Initialisierung von lokalen Variablen .....	139
<b>2.4</b>	<b>Ausdrücke, Operanden und Operatoren .....</b>	<b>140</b>
2.4.1	Zuweisungsoperator .....	141
2.4.2	Arithmetische Operatoren .....	142
2.4.3	Unäres Minus und Plus .....	146
2.4.4	Zuweisung mit Operation .....	146
2.4.5	Präfix- oder Postfix-Inkrement und -Dekrement .....	147
2.4.6	Die relationalen Operatoren und die Gleichheitsoperatoren .....	150
2.4.7	Logische Operatoren: Nicht, Und, Oder, XOR .....	152
2.4.8	Kurzschluss-Operatoren .....	153
2.4.9	Der Rang der Operatoren in der Auswertungsreihenfolge .....	154
2.4.10	Die Typumwandlung (das Casting) .....	157
2.4.11	Überladenes Plus für Strings .....	162
2.4.12	Operator vermisst * .....	163

<b>2.5</b>	<b>Bedingte Anweisungen oder Fallunterscheidungen</b> .....	164
2.5.1	Verzweigung mit der if-Anweisung .....	164
2.5.2	Die Alternative mit einer if-else-Anweisung wählen .....	166
2.5.3	Der Bedingungsoperator .....	171
2.5.4	Die switch-Anweisung bietet die Alternative .....	173
<b>2.6</b>	<b>Immer das Gleiche mit den Schleifen</b> .....	178
2.6.1	Die while-Schleife .....	179
2.6.2	Die do-while-Schleife .....	181
2.6.3	Die for-Schleife .....	183
2.6.4	Schleifenbedingungen und Vergleiche mit == .....	186
2.6.5	Ausbruch planen mit break und Wiedereinstieg mit continue .....	189
2.6.6	break und continue mit Marken * .....	192
<b>2.7</b>	<b>Methoden einer Klasse</b> .....	196
2.7.1	Bestandteil einer Methode .....	196
2.7.2	Signatur-Beschreibung in der Java-API .....	198
2.7.3	Aufruf einer Methode .....	199
2.7.4	Methoden ohne Parameter deklarieren .....	200
2.7.5	Statische Methoden (Klassenmethoden) .....	201
2.7.6	Parameter, Argument und Wertübergabe .....	202
2.7.7	Methoden vorzeitig mit return beenden .....	204
2.7.8	Nicht erreichbarer Quellcode bei Methoden * .....	204
2.7.9	Methoden mit Rückgaben .....	205
2.7.10	Methoden überladen .....	210
2.7.11	Sichtbarkeit und Gültigkeitsbereich .....	212
2.7.12	Vorgegebener Wert für nicht aufgeführte Argumente * .....	214
2.7.13	Finale lokale Variablen .....	214
2.7.14	Rekursive Methoden * .....	215
2.7.15	Die Türme von Hanoi * .....	220
<b>2.8</b>	<b>Zum Weiterlesen</b> .....	222
<b>3</b>	<b>Klassen und Objekte</b> .....	223
<hr/>		
<b>3.1</b>	<b>Objektorientierte Programmierung (OOP)</b> .....	223
3.1.1	Warum überhaupt OOP? .....	223
3.1.2	Denk ich an Java, denk ich an Wiederverwendbarkeit .....	224
<b>3.2</b>	<b>Eigenschaften einer Klasse</b> .....	225
3.2.1	Klassenarbeit mit Point .....	226

<b>3.3</b>	<b>Natürlich modellieren mit der UML (Unified Modeling Language) *</b>	226
3.3.1	Hintergrund und Geschichte der UML	227
3.3.2	Wichtige Diagrammtypen der UML	228
3.3.3	UML-Werkzeuge	229
<b>3.4</b>	<b>Neue Objekte erzeugen</b>	230
3.4.1	Ein Exemplar einer Klasse mit dem Schlüsselwort new anlegen	231
3.4.2	Der Zusammenhang von new, Heap und Garbage-Collector	231
3.4.3	Deklarieren von Referenzvariablen	232
3.4.4	Jetzt mach mal 'nen Punkt: Zugriff auf Objektattribute und -methoden	234
3.4.5	Überblick über Point-Methoden	238
3.4.6	Konstruktoren nutzen	241
<b>3.5</b>	<b>ZZZZZnake</b>	242
<b>3.6</b>	<b>Pakete schnüren, Imports und Kompilationseinheiten</b>	245
3.6.1	Java-Pakete	245
3.6.2	Pakete der Standardbibliothek	245
3.6.3	Volle Qualifizierung und import-Deklaration	246
3.6.4	Mit import p1.p2.* alle Typen eines Pakets erreichen	247
3.6.5	Hierarchische Strukturen über Pakete	248
3.6.6	Die package-Deklaration	248
3.6.7	Unbenanntes Paket (default package)	249
3.6.8	Klassen mit gleichen Namen in unterschiedlichen Paketen *	250
3.6.9	Kompilationseinheit (Compilation Unit)	251
3.6.10	Statischer Import *	251
<b>3.7</b>	<b>Mit Referenzen arbeiten, Identität und Gleichheit (Gleichwertigkeit)</b>	252
3.7.1	null-Referenz und die Frage der Philosophie	252
3.7.2	Alles auf null? Referenzen testen	254
3.7.3	Zuweisungen bei Referenzen	256
3.7.4	Methoden mit Referenztypen als Parametern	257
3.7.5	Identität von Objekten	261
3.7.6	Gleichheit (Gleichwertigkeit) und die Methode equals(...)	262
<b>3.8</b>	<b>Arrays</b>	263
3.8.1	Grundbestandteile	264
3.8.2	Deklaration von Arrays	264
3.8.3	Arrays mit Inhalt	265
3.8.4	Die Länge eines Arrays über das Attribut length auslesen	266
3.8.5	Zugriff auf die Elemente über den Index	266
3.8.6	Array-Objekte mit new erzeugen	268
3.8.7	Typische Array-Fehler	269
3.8.8	Arrays als Methodenparameter	270
3.8.9	Vorinitialisierte Arrays	271

3.8.10	Die erweiterte for-Schleife .....	272
3.8.11	Arrays mit nichtprimitiven Elementen .....	274
3.8.12	Methode mit variabler Argumentanzahl (Vararg) .....	277
3.8.13	Mehrdimensionale Arrays * .....	280
3.8.14	Nichtrechteckige Arrays * .....	283
3.8.15	Die Wahrheit über die Array-Initialisierung * .....	286
3.8.16	Mehrere Rückgabewerte * .....	286
3.8.17	Klonen kann sich lohnen – Arrays vermehren * .....	287
3.8.18	Array-Inhalte kopieren * .....	288
3.8.19	Die Klasse Arrays zum Vergleichen, Füllen, Suchen, Sortieren nutzen .....	289
3.8.20	Eine lange Schlange .....	300
<b>3.9</b>	<b>Der Einstiegspunkt für das Laufzeitsystem: main(...)</b> .....	<b>303</b>
3.9.1	Korrekte Deklaration der Startmethode .....	303
3.9.2	Kommandozeilenargumente verarbeiten .....	304
3.9.3	Der Rückgabotyp von main(...) und System.exit(int) * .....	305
<b>3.10</b>	<b>Grundlagen von Annotationen und Generics</b> .....	<b>307</b>
3.10.1	Generics .....	307
3.10.2	Annotationen .....	308
3.10.3	Annotationstypen aus java.lang .....	310
3.10.4	@Deprecated .....	310
3.10.5	@SuppressWarnings .....	311
<b>3.11</b>	<b>Zum Weiterlesen</b> .....	<b>315</b>
<b>4</b>	<b>Der Umgang mit Zeichenketten</b> .....	<b>317</b>
<hr/>		
<b>4.1</b>	<b>Von ASCII über ISO-8859-1 zu Unicode</b> .....	<b>317</b>
4.1.1	ASCII .....	317
4.1.2	ISO/IEC 8859-1 .....	318
4.1.3	Unicode .....	319
4.1.4	Unicode-Zeichenkodierung .....	321
4.1.5	Escape-Sequenzen/Fluchtsymbole .....	321
4.1.6	Schreibweise für Unicode-Zeichen und Unicode-Escapes .....	322
4.1.7	Java-Versionen gehen mit Unicode-Standard Hand in Hand * .....	324
<b>4.2</b>	<b>Die Character-Klasse</b> .....	<b>326</b>
4.2.1	Ist das so? .....	326
4.2.2	Zeichen in Großbuchstaben/Kleinbuchstaben konvertieren .....	328
4.2.3	Ziffern einer Basis * .....	329

<b>4.3</b>	<b>Zeichenfolgen</b> .....	330
<b>4.4</b>	<b>Die Klasse String und ihre Methoden</b> .....	332
4.4.1	String-Literale als String-Objekte für konstante Zeichenketten .....	333
4.4.2	Konkatenation mit + .....	333
4.4.3	String-Länge und Test auf Leer-String .....	333
4.4.4	Zugriff auf ein bestimmtes Zeichen mit charAt(int) .....	335
4.4.5	Nach enthaltenen Zeichen und Zeichenfolgen suchen .....	335
4.4.6	Das Hangman-Spiel .....	338
4.4.7	Gut, dass wir verglichen haben .....	340
4.4.8	String-Teile extrahieren .....	344
4.4.9	Strings anhängen, zusammenfügen, Groß-/Kleinschreibung und Leerraum .....	348
4.4.10	Gesucht, gefunden, ersetzt .....	351
4.4.11	String-Objekte mit Konstruktoren erzeugen * .....	354
<b>4.5</b>	<b>Veränderbare Zeichenketten mit StringBuilder und StringBuffer</b> .....	357
4.5.1	Anlegen von StringBuilder-/StringBuffer-Objekten .....	359
4.5.2	StringBuilder/StringBuffer in andere Zeichenkettenformate konvertieren ....	360
4.5.3	Zeichen(folgen) erfragen .....	360
4.5.4	Daten anhängen .....	360
4.5.5	Zeichen(folgen) setzen, löschen und umdrehen .....	362
4.5.6	Länge und Kapazität eines StringBuilder-/StringBuffer-Objekts * .....	364
4.5.7	Vergleichen von String mit StringBuilder und StringBuffer .....	366
4.5.8	hashCode() bei StringBuilder/StringBuffer * .....	367
<b>4.6</b>	<b>CharSequence als Basistyp</b> .....	367
<b>4.7</b>	<b>Konvertieren zwischen Primitiven und Strings</b> .....	370
4.7.1	Unterschiedliche Typen in String-Repräsentationen konvertieren .....	370
4.7.2	String-Inhalt in einen primitiven Wert konvertieren .....	372
4.7.3	String-Repräsentation im Format Binär, Hex, Oktal * .....	374
4.7.4	parseXXX(...)- und printXXX()-Methoden in DatatypeConverter * .....	378
<b>4.8</b>	<b>Strings zusammenhängen (konkatenieren)</b> .....	379
4.8.1	Strings mit StringJoiner .....	380
<b>4.9</b>	<b>Zerlegen von Zeichenketten</b> .....	381
4.9.1	Splitten von Zeichenketten mit split(...) .....	382
4.9.2	Yes we can, yes we scan – die Klasse Scanner .....	383
<b>4.10</b>	<b>Ausgaben formatieren</b> .....	386
4.10.1	Formatieren und Ausgeben mit format() .....	387
<b>4.11</b>	<b>Zum Weiterlesen</b> .....	392

<b>5</b>	<b>Eigene Klassen schreiben</b>	395
<b>5.1</b>	<b>Eigene Klassen mit Eigenschaften deklarieren</b>	395
5.1.1	Attribute deklarieren	396
5.1.2	Methoden deklarieren	398
5.1.3	Die this-Referenz	402
<b>5.2</b>	<b>Privatsphäre und Sichtbarkeit</b>	406
5.2.1	Für die Öffentlichkeit: public	406
5.2.2	Kein Public Viewing – Passwörter sind privat	407
5.2.3	Wieso nicht freie Methoden und Variablen für alle?	408
5.2.4	Privat ist nicht ganz privat: Es kommt darauf an, wer's sieht *	409
5.2.5	Zugriffsmethoden für Attribute deklarieren	409
5.2.6	Setter und Getter nach der JavaBeans-Spezifikation	410
5.2.7	Paketsichtbar	413
5.2.8	Zusammenfassung zur Sichtbarkeit	414
<b>5.3</b>	<b>Eine für alle – statische Methode und statische Attribute</b>	416
5.3.1	Warum statische Eigenschaften sinnvoll sind	417
5.3.2	Statische Eigenschaften mit static	418
5.3.3	Statische Eigenschaften über Referenzen nutzen? *	419
5.3.4	Warum die Groß- und Kleinschreibung wichtig ist *	420
5.3.5	Statische Variablen zum Datenaustausch *	420
5.3.6	Statische Eigenschaften und Objekteigenschaften *	422
<b>5.4</b>	<b>Konstanten und Aufzählungen</b>	423
5.4.1	Konstanten über statische finale Variablen	423
5.4.2	Typunsichere Aufzählungen	424
5.4.3	Aufzählungstypen: Typsichere Aufzählungen mit enum	426
<b>5.5</b>	<b>Objekte anlegen und zerstören</b>	431
5.5.1	Konstruktoren schreiben	431
5.5.2	Verwandtschaft von Methode und Konstruktor	432
5.5.3	Der vorgegebene Konstruktor (default constructor)	433
5.5.4	Parametrisierte und überladene Konstruktoren	435
5.5.5	Copy-Konstruktor	437
5.5.6	Einen anderen Konstruktor der gleichen Klasse mit this(...) aufrufen	438
5.5.7	Ihr fehlt uns nicht – der Garbage-Collector	442
<b>5.6</b>	<b>Klassen- und Objektinitialisierung *</b>	444
5.6.1	Initialisierung von Objektvariablen	444
5.6.2	Statische Blöcke als Klasseninitialisierer	445
5.6.3	Initialisierung von Klassenvariablen	447
5.6.4	Eincompilierte Belegungen der Klassenvariablen	447



5.6.5	Exemplarinitialisierer (Instanzinitialisierer) .....	448
5.6.6	Finale Werte im Konstruktor und in statischen Blöcken setzen .....	451
<b>5.7</b>	<b>Zum Weiterlesen</b> .....	453
<b>6</b>	<b>Objektorientierte Beziehungsfragen</b> .....	455
<b>6.1</b>	<b>Assoziationen zwischen Objekten</b> .....	455
6.1.1	Unidirektionale 1:1-Beziehung .....	456
6.1.2	Zwei Freunde müsst ihr werden – bidirektionale 1:1-Beziehungen .....	457
6.1.3	Unidirektionale 1:n-Beziehung .....	458
<b>6.2</b>	<b>Vererbung</b> .....	461
6.2.1	Vererbung in Java .....	461
6.2.2	Spielobjekte modellieren .....	462
6.2.3	Die implizite Basisklasse java.lang.Object .....	464
6.2.4	Einfach- und Mehrfachvererbung * .....	464
6.2.5	Die Sichtbarkeit protected .....	465
6.2.6	Konstruktoren in der Vererbung und super(...) .....	465
<b>6.3</b>	<b>Typen in Hierarchien</b> .....	470
6.3.1	Automatische und explizite Typumwandlung .....	471
6.3.2	Das Substitutionsprinzip .....	473
6.3.3	Typen mit dem instanceof-Operator testen .....	475
<b>6.4</b>	<b>Methoden überschreiben</b> .....	477
6.4.1	Methoden in Unterklassen mit neuem Verhalten ausstatten .....	477
6.4.2	Mit super an die Eltern .....	481
6.4.3	Finale Klassen und finale Methoden .....	483
6.4.4	Kovariante Rückgabetypen .....	485
6.4.5	Array-Typen und Kovarianz * .....	486
<b>6.5</b>	<b>Drum prüfe, wer sich ewig dynamisch bindet</b> .....	487
6.5.1	Gebunden an toString() .....	487
6.5.2	Implementierung von System.out.println(Object) .....	490
6.5.3	Nicht dynamisch gebunden bei privaten, statischen und finalen Methoden .....	490
6.5.4	Dynamisch gebunden auch bei Konstruktoraufrufen * .....	492
6.5.5	Eine letzte Spielerei mit Javas dynamischer Bindung und überdeckten Attributen * .....	494
<b>6.6</b>	<b>Abstrakte Klassen und abstrakte Methoden</b> .....	495
6.6.1	Abstrakte Klassen .....	495
6.6.2	Abstrakte Methoden .....	497

<b>6.7</b>	<b>Schnittstellen</b> .....	501
6.7.1	Schnittstellen deklarieren .....	502
6.7.2	Implementieren von Schnittstellen .....	503
6.7.3	Ein Polymorphie-Beispiel mit Schnittstellen .....	505
6.7.4	Die Mehrfachvererbung bei Schnittstellen .....	506
6.7.5	Keine Kollisionsgefahr bei Mehrfachvererbung * .....	511
6.7.6	Erweitern von Interfaces – Subinterfaces .....	512
6.7.7	Konstantendeklarationen bei Schnittstellen .....	513
6.7.8	Statische ausprogrammierte Methoden in Schnittstellen .....	515
6.7.9	Erweitern von Schnittstellen .....	517
6.7.10	Default-Methoden .....	519
6.7.11	Erweiterte Schnittstellen deklarieren und nutzen .....	520
6.7.12	Erweiterte Schnittstellen, Mehrfachvererbung und Mehrdeutigkeiten * .....	523
6.7.13	Bausteine bilden mit Default-Methoden * .....	527
6.7.14	Initialisierung von Schnittstellenkonstanten * .....	533
6.7.15	Markierungsschnittstellen * .....	537
6.7.16	(Abstrakte) Klassen und Schnittstellen im Vergleich .....	537
<b>6.8</b>	<b>Zum Weiterlesen</b> .....	538

## **7 Ausnahmen müssen sein** 539

---

<b>7.1</b>	<b>Problembereiche einzäunen</b> .....	539
7.1.1	Exceptions in Java mit try und catch .....	540
7.1.2	Eine NumberFormatException auffangen .....	540
7.1.3	Eigenschaften vom Exception-Objekt .....	543
7.1.4	Wiederholung abgebrochener Bereiche * .....	545
7.1.5	Mehrere Ausnahmen auffangen .....	546
7.1.6	Ablauf einer Ausnahmesituation .....	548
7.1.7	throws im Methodenkopf angeben .....	549
7.1.8	Abschlussbehandlung mit finally .....	550
<b>7.2</b>	<b>RuntimeException muss nicht aufgefangen werden</b> .....	555
7.2.1	Beispiele für RuntimeException-Klassen .....	556
7.2.2	Kann man abfangen, muss man aber nicht .....	556
<b>7.3</b>	<b>Die Klassenhierarchie der Fehler</b> .....	557
7.3.1	Die Exception-Hierarchie .....	558
7.3.2	Oberausnahmen auffangen .....	558
7.3.3	Schon gefangen? .....	560
7.3.4	Alles geht als Exception durch .....	560
7.3.5	Zusammenfassen gleicher catch-Blöcke mit dem multi-catch .....	562

<b>7.4</b>	<b>Harte Fehler – Error *</b> .....	565
<b>7.5</b>	<b>Auslösen eigener Exceptions</b> .....	566
7.5.1	Mit throw Ausnahmen auslösen .....	566
7.5.2	Vorhandene Runtime-Fehlertypen kennen und nutzen .....	568
7.5.3	Parameter testen und gute Fehlermeldungen .....	571
7.5.4	Neue Exception-Klassen deklarieren .....	572
7.5.5	Eigene Ausnahmen als Unterklassen von Exception oder RuntimeException? .....	574
7.5.6	Ausnahmen abfangen und weiterleiten * .....	577
7.5.7	Aufruf-Stack von Ausnahmen verändern * .....	578
7.5.8	Präzises rethrow * .....	579
7.5.9	Geschachtelte Ausnahmen * .....	582
<b>7.6</b>	<b>Automatisches Ressourcen-Management (try mit Ressourcen)</b> .....	585
7.6.1	try mit Ressourcen .....	586
7.6.2	Die Schnittstelle AutoCloseable .....	588
7.6.3	Mehrere Ressourcen nutzen .....	590
7.6.4	try mit Ressourcen auf null-Ressourcen .....	591
7.6.5	Unterdrückte Ausnahmen * .....	591
<b>7.7</b>	<b>Besonderheiten bei der Ausnahmebehandlung *</b> .....	595
7.7.1	Rückgabewerte bei ausgelösten Ausnahmen .....	595
7.7.2	Ausnahmen und Rückgaben verschwinden – das Duo return und finally .....	596
7.7.3	throws bei überschriebenen Methoden .....	597
7.7.4	Nicht erreichbare catch-Klauseln .....	599
<b>7.8</b>	<b>Den Stack-Trace erfragen *</b> .....	600
7.8.1	StackTraceElement .....	601
7.8.2	printStackTrace(...) .....	601
7.8.3	StackTraceElement vom Thread erfragen .....	602
<b>7.9</b>	<b>Assertions *</b> .....	603
7.9.1	Assertions in eigenen Programmen nutzen .....	604
7.9.2	Assertions aktivieren .....	605
<b>7.10</b>	<b>Zum Weiterlesen</b> .....	607
<b>8</b>	<b>Äußere.innere Klassen</b> .....	609
<b>8.1</b>	<b>Geschachtelte (innere) Klassen, Schnittstellen, Aufzählungen</b> .....	609
<b>8.2</b>	<b>Statische innere Klassen und Schnittstellen</b> .....	610

<b>8.3</b>	<b>Mitglieds- oder Elementklassen</b> .....	612
8.3.1	Exemplare innerer Klassen erzeugen .....	612
8.3.2	Die this-Referenz .....	613
8.3.3	Vom Compiler generierte Klassendateien * .....	614
8.3.4	Erlaubte Modifizierer bei äußeren und inneren Klassen .....	615
8.3.5	Innere Klassen greifen auf private Eigenschaften zu .....	615
<b>8.4</b>	<b>Lokale Klassen</b> .....	617
8.4.1	Beispiel mit eigener Klassendeklaration .....	617
8.4.2	Lokale Klasse für einen Timer nutzen .....	618
<b>8.5</b>	<b>Anonyme innere Klassen</b> .....	619
8.5.1	Nutzung einer anonymen inneren Klasse für den Timer .....	619
8.5.2	Umsetzung innerer anonymer Klassen * .....	621
8.5.3	Konstruktoren innerer anonymer Klassen .....	621
<b>8.6</b>	<b>Zugriff auf lokale Variablen aus lokalen inneren und anonymen Klassen *</b> .....	623
<b>8.7</b>	<b>this in Unterklassen *</b> .....	624
<b>8.8</b>	<b>Zum Weiterlesen</b> .....	626

## **9 Besondere Typen der Java SE** 627

---

<b>9.1</b>	<b>Object ist die Mutter aller Klassen</b> .....	628
9.1.1	Klassenobjekte .....	628
9.1.2	Objektidentifikation mit toString() .....	629
9.1.3	Objektgleichheit mit equals(...) und Identität .....	631
9.1.4	Klonen eines Objekts mit clone() * .....	636
9.1.5	Hashwerte über hashCode() liefern * .....	641
9.1.6	System.identityHashCode(...) und das Problem der nicht eindeutigen Objektverweise * .....	648
9.1.7	Aufräumen mit finalize() * .....	649
9.1.8	Synchronisation * .....	651
<b>9.2</b>	<b>Die Utility-Klasse java.util.Objects</b> .....	651
<b>9.3</b>	<b>Vergleichen von Objekten</b> .....	654
9.3.1	Natürlich geordnet oder nicht? .....	655
9.3.2	Die Schnittstelle Comparable .....	655
9.3.3	Die Schnittstelle Comparator .....	656
9.3.4	Rückgabewerte kodieren die Ordnung .....	657
9.3.5	Statische und Default-Methoden in Comparator .....	659

<b>9.4</b>	<b>Wrapper-Klassen und Autoboxing</b> .....	663
9.4.1	Wrapper-Objekte erzeugen .....	664
9.4.2	Konvertierungen in eine String-Repräsentation .....	666
9.4.3	Von einer String-Repräsentation parsen .....	667
9.4.4	Die Basisklasse Number für numerische Wrapper-Objekte .....	667
9.4.5	Vergleiche durchführen mit compare(...), compareTo(...), equals(...) und Hashwerten .....	669
9.4.6	Statische Reduzierungsmethoden in Wrapper-Klassen .....	672
9.4.7	Die Größe eines primitiven Typs in den Wrapper-Konstanten SIZE und BYTES .....	673
9.4.8	Behandeln von vorzeichenlosen Zahlen * .....	673
9.4.9	Die Klasse Integer .....	675
9.4.10	Die Klassen Double und Float für Fließkommazahlen .....	676
9.4.11	Die Long-Klasse .....	676
9.4.12	Die Boolean-Klasse .....	676
9.4.13	Autoboxing: Boxing und Unboxing .....	677
<b>9.5</b>	<b>Iterator, Iterable *</b> .....	681
9.5.1	Die Schnittstelle Iterator .....	682
9.5.2	Wer den Iterator liefert .....	684
9.5.3	Die Schnittstelle Iterable .....	685
9.5.4	Erweitertes for und Iterable .....	686
9.5.5	Interne Iteration .....	686
9.5.6	Einen eigenen Iterable implementieren * .....	687
<b>9.6</b>	<b>Die Spezial-Oberklasse Enum</b> .....	688
9.6.1	Methoden auf Enum-Objekten .....	689
9.6.2	Aufzählungen mit eigenen Methoden und Initialisierern * .....	692
9.6.3	enum mit eigenen Konstruktoren * .....	695
<b>9.7</b>	<b>Zum Weiterlesen</b> .....	698
<b>10</b>	<b>Generics&lt;T&gt;</b> .....	699
<hr/>		
<b>10.1</b>	<b>Einführung in Java Generics</b> .....	699
10.1.1	Mensch versus Maschine – Typprüfung des Compilers und der Laufzeitumgebung .....	699
10.1.2	Taschen .....	700
10.1.3	Generische Typen deklarieren .....	702
10.1.4	Generics nutzen .....	703
10.1.5	Diamonds are forever .....	706

10.1.6	Generische Schnittstellen .....	709
10.1.7	Generische Methoden/Konstruktoren und Typ-Inferenz .....	711
<b>10.2</b>	<b>Umsetzen der Generics, Typlöschung und Raw-Types</b> .....	<b>715</b>
10.2.1	Realisierungsmöglichkeiten .....	715
10.2.2	Typlöschung (Type Erasure) .....	715
10.2.3	Probleme der Typlöschung .....	717
10.2.4	Raw-Type .....	722
<b>10.3</b>	<b>Einschränken der Typen über Bounds</b> .....	<b>724</b>
10.3.1	Einfache Einschränkungen mit extends .....	725
10.3.2	Weitere Obertypen mit & .....	727
<b>10.4</b>	<b>Typparameter in der throws-Klausel *</b> .....	<b>728</b>
10.4.1	Deklaration einer Klasse mit Typvariable <E extends Exception> .....	728
10.4.2	Parametrisierter Typ bei Typvariable <E extends Exception> .....	728
<b>10.5</b>	<b>Generics und Vererbung, Invarianz</b> .....	<b>731</b>
10.5.1	Arrays sind kovariant .....	731
10.5.2	Generics sind nicht kovariant, sondern invariant .....	732
10.5.3	Wildcards mit ? .....	733
10.5.4	Bounded Wildcards .....	735
10.5.5	Bounded-Wildcard-Typen und Bounded-Typvariablen .....	739
10.5.6	Das LESS-Prinzip .....	741
10.5.7	Enum<E extends Enum<E>> * .....	743
<b>10.6</b>	<b>Konsequenzen der Typlöschung: Typ-Token, Arrays und Brücken *</b> .....	<b>745</b>
10.6.1	Typ-Token .....	745
10.6.2	Super-Type-Token .....	747
10.6.3	Generics und Arrays .....	748
10.6.4	Brückenmethoden .....	749
10.6.5	Zum Weiterlesen .....	754
<b>11</b>	<b>Lambda-Ausdrücke und funktionale Programmierung</b> .....	<b>757</b>
<b>11.1</b>	<b>Code = Daten</b> .....	<b>757</b>
<b>11.2</b>	<b>Funktionale Schnittstellen und Lambda-Ausdrücke im Detail</b> .....	<b>760</b>
11.2.1	Funktionale Schnittstellen .....	761
11.2.2	Typ eines Lambda-Ausdrucks ergibt sich durch Zieltyp .....	762
11.2.3	Annotation @FunctionalInterface .....	766
11.2.4	Syntax für Lambda-Ausdrücke .....	767
11.2.5	Die Umgebung der Lambda-Ausdrücke und Variablenzugriffe .....	771

11.2.6	Ausnahmen in Lambda-Ausdrücken .....	775
11.2.7	Klassen mit einer abstrakten Methode als funktionale Schnittstelle? * .....	779
<b>11.3</b>	<b>Methodenreferenz</b> .....	<b>780</b>
11.3.1	Varianten von Methodenreferenzen .....	781
<b>11.4</b>	<b>Konstruktorreferenz</b> .....	<b>783</b>
11.4.1	Standard- und parametrisierte Konstruktoren .....	785
11.4.2	Nützliche vordefinierte Schnittstellen für Konstruktorreferenzen .....	785
<b>11.5</b>	<b>Implementierung von Lambda-Ausdrücken *</b> .....	<b>787</b>
<b>11.6</b>	<b>Funktionale Programmierung mit Java</b> .....	<b>787</b>
11.6.1	Programmierparadigmen: imperativ oder deklarativ .....	787
11.6.2	Funktionale Programmierung und funktionale Programmiersprachen .....	788
11.6.3	Funktionale Programmierung in Java am Beispiel vom Comparator .....	790
11.6.4	Lambda-Ausdrücke als Funktionen sehen .....	790
<b>11.7</b>	<b>Funktionale Schnittstelle aus dem java.util.function-Paket</b> .....	<b>792</b>
11.7.1	Blöcke mit Code und die funktionale Schnittstelle java.util.function.Consumer .....	792
11.7.2	Supplier .....	794
11.7.3	Prädikate und java.util.function.Predicate .....	794
11.7.4	Funktionen und die allgemeine funktionale Schnittstelle java.util.function.Function .....	797
11.7.5	Ein bisschen Bi ... .....	800
11.7.6	Funktionale Schnittstellen mit Primitiven .....	803
<b>11.8</b>	<b>Optional ist keine Nullnummer</b> .....	<b>806</b>
11.8.1	Optional-Typ .....	808
11.8.2	Primitive optionale Typen .....	811
11.8.3	Erstmal funktional mit Optional .....	812
<b>11.9</b>	<b>Was ist jetzt so funktional?</b> .....	<b>816</b>
<b>11.10</b>	<b>Zum Weiterlesen</b> .....	<b>819</b>
<b>12</b>	<b>Architektur, Design und angewandte Objektorientierung</b> .....	<b>821</b>
<b>12.1</b>	<b>Architektur, Design und Implementierung</b> .....	<b>821</b>
<b>12.2</b>	<b>Design-Pattern (Entwurfsmuster)</b> .....	<b>822</b>
12.2.1	Motivation für Design-Pattern .....	822
12.2.2	Singleton .....	823

12.2.3	Fabrikmethoden .....	826
12.2.4	Das Beobachter-Pattern (Observer/Observable) .....	827
12.2.5	Ereignisse über Listener .....	832
<b>12.3</b>	<b>Zum Weiterlesen</b> .....	<b>836</b>

## **13 Komponenten, JavaBeans und Module** 837

---

<b>13.1</b>	<b>JavaBeans</b> .....	<b>837</b>
13.1.1	Properties (Eigenschaften) .....	838
13.1.2	Einfache Eigenschaften .....	839
13.1.3	Indizierte Eigenschaften .....	839
13.1.4	Gebundene Eigenschaften und PropertyChangeListener .....	839
13.1.5	Veto-Eigenschaften – dagegen! .....	843
<b>13.2</b>	<b>JavaFX Properties</b> .....	<b>846</b>
13.2.1	javafx.beans-Paket mit XXXProperty-Klassen .....	847
13.2.2	Property-Veränderungen registrieren .....	848
13.2.3	Beans-Binding .....	849
13.2.4	Property-Schnittstelle und bindXXX(...)-Methoden .....	850
13.2.5	XXXProperty-Beziehungen (für Typ-Fetischisten) * .....	857
13.2.6	Ausblick .....	859
<b>13.3</b>	<b>Klassenlader (Class Loader) und Klassenpfad</b> .....	<b>860</b>
13.3.1	Klassenladen auf Abruf .....	860
13.3.2	JAR-Dateien .....	861
13.3.3	Woher die kleinen Klassen kommen: die Suchorte und spezielle Klassenlader .....	861
13.3.4	Setzen des Klassenpfades .....	862
<b>13.4</b>	<b>Zum Weiterlesen</b> .....	<b>864</b>

## **14 Die Klassenbibliothek** 865

---

<b>14.1</b>	<b>Die Java-Klassenphilosophie</b> .....	<b>865</b>
14.1.1	Übersicht über die Pakete der Standardbibliothek .....	865
14.1.2	Compact-Profile .....	868
<b>14.2</b>	<b>Sprachen der Länder</b> .....	<b>869</b>
14.2.1	Sprachen und Regionen über Locale-Objekte .....	870



<b>14.3 Die Klasse Date</b> .....	873
14.3.1 Objekte erzeugen und Methoden nutzen .....	873
14.3.2 Date-Objekte sind nicht immutable .....	875
<b>14.4 Calendar und GregorianCalendar</b> .....	875
14.4.1 Die abstrakte Klasse Calendar .....	876
14.4.2 Calendar nach Date und Millisekunden fragen .....	877
14.4.3 Abfragen und Setzen von Datumselementen über Feldbezeichner .....	878
14.4.4 Kalender-Exemplare bauen über den Calendar.Builder .....	882
14.4.5 Der gregorianische Kalender .....	882
14.4.6 Date-Time-API .....	884
14.4.7 Menschenzeit und Maschinenzeit .....	885
<b>14.5 Die Utility-Klasse System und Properties</b> .....	889
14.5.1 Systemeigenschaften der Java-Umgebung .....	890
14.5.2 Zeilenumbruchzeichen, line.separator .....	892
14.5.3 Eigene Properties von der Konsole aus setzen * .....	892
14.5.4 Umgebungsvariablen des Betriebssystems * .....	895
14.5.5 Einfache Zeitmessung und Profiling * .....	896
<b>14.6 Einfache Benutzereingaben</b> .....	899
14.6.1 Grafischer Eingabedialog über JOptionPane .....	899
14.6.2 Geschützte Passwort-Eingaben mit der Klasse Console * .....	900
<b>14.7 Benutzereinstellungen *</b> .....	901
14.7.1 Benutzereinstellungen mit der Preferences-API .....	902
14.7.2 Einträge einfügen, auslesen und löschen .....	903
14.7.3 Auslesen der Daten und Schreiben in einem anderen Format .....	905
14.7.4 Auf Ereignisse horchen .....	905
14.7.5 Zugriff auf die gesamte Windows-Registry .....	907
<b>14.8 Zum Weiterlesen</b> .....	908

## **15 Einführung in die nebenläufige Programmierung** 909

---

<b>15.1 Nebenläufigkeit und Parallelität</b> .....	909
15.1.1 Multitasking, Prozesse, Threads .....	910
15.1.2 Threads und Prozesse .....	910
15.1.3 Wie nebenläufige Programme die Geschwindigkeit steigern können .....	911
15.1.4 Was Java für Nebenläufigkeit alles bietet .....	913
<b>15.2 Threads erzeugen</b> .....	913
15.2.1 Threads über die Schnittstelle Runnable implementieren .....	913

15.2.2	Thread mit Runnable starten .....	915
15.2.3	Die Klasse Thread erweitern .....	916
<b>15.3</b>	<b>Thread-Eigenschaften und Zustände .....</b>	<b>919</b>
15.3.1	Der Name eines Threads .....	919
15.3.2	Wer bin ich? .....	919
15.3.3	Schläfer gesucht .....	920
15.3.4	Mit yield() auf Rechenzeit verzichten .....	922
15.3.5	Der Thread als Dämon .....	922
15.3.6	Freiheit für den Thread – das Ende .....	924
15.3.7	Einen Thread höflich mit Interrupt beenden .....	925
15.3.8	UncaughtExceptionHandler für unbehandelte Ausnahmen .....	927
<b>15.4</b>	<b>Der Ausführer (Executor) kommt .....</b>	<b>928</b>
15.4.1	Die Schnittstelle Executor .....	929
15.4.2	Glücklich in der Gruppe – die Thread-Pools .....	930
15.4.3	Threads mit Rückgabe über Callable .....	932
15.4.4	Mehrere Callables abarbeiten .....	935
15.4.5	ScheduledExecutorService für wiederholende Ausgaben und Zeitsteuerungen nutzen .....	936
<b>15.5</b>	<b>Synchronisation über kritische Abschnitte .....</b>	<b>937</b>
15.5.1	Gemeinsam genutzte Daten .....	937
15.5.2	Probleme beim gemeinsamen Zugriff und kritische Abschnitte .....	937
15.5.3	Punkte nebenläufig initialisieren .....	939
15.5.4	Kritische Abschnitte schützen .....	941
15.5.5	Kritische Abschnitte mit ReentrantLock schützen .....	943
15.5.6	Synchronisieren mit synchronized .....	947
15.5.7	Mit synchronized synchronisierte Blöcke .....	948
15.5.8	Dann machen wir doch gleich alles synchronisiert! .....	950
15.5.9	Lock-Freigabe im Fall von Exceptions .....	950
15.5.10	Deadlocks .....	951
<b>15.6</b>	<b>Zum Weiterlesen .....</b>	<b>953</b>
 <b>16 Einführung in Datenstrukturen und Algorithmen</b>		<b>955</b>
<hr/>		
<b>16.1</b>	<b>Listen .....</b>	<b>955</b>
16.1.1	Erstes Listenbeispiel .....	955
16.1.2	Auswahlkriterium ArrayList oder LinkedList .....	956
16.1.3	Die Schnittstelle List .....	957
16.1.4	ArrayList .....	963

16.1.5	LinkedList .....	963
16.1.6	Der Feld-Adapter Arrays.asList(...) .....	965
16.1.7	toArray(...) von Collection verstehen – die Gefahr einer Falle erkennen .....	967
16.1.8	Primitive Elemente in Datenstrukturen verwalten .....	969
<b>16.2</b>	<b>Mengen (Sets) .....</b>	<b>970</b>
16.2.1	Ein erstes Mengenbeispiel .....	970
16.2.2	Methoden der Schnittstelle Set .....	972
16.2.3	HashSet .....	974
16.2.4	TreeSet – die sortierte Menge .....	974
16.2.5	Die Schnittstellen NavigableSet und SortedSet .....	976
16.2.6	LinkedHashSet .....	978
<b>16.3</b>	<b>Assoziative Speicher .....</b>	<b>980</b>
16.3.1	Die Klassen HashMap und TreeMap .....	980
16.3.2	Einfügen und Abfragen des Assoziativspeichers .....	982
16.3.3	Über die Bedeutung von equals(...) und hashCode() bei Elementen .....	989
16.3.4	Eigene Objekte hashen .....	989
16.3.5	LinkedHashMap und LRU-Implementierungen .....	991
16.3.6	IdentityHashMap .....	992
16.3.7	Das Problem veränderter Elemente .....	992
16.3.8	Aufzählungen und Ansichten des Assoziativspeichers .....	993
16.3.9	Die Properties-Klasse .....	996
<b>16.4</b>	<b>Mit einem Iterator durch die Daten wandern .....</b>	<b>998</b>
<b>16.5</b>	<b>Algorithmen in Collections .....</b>	<b>999</b>
16.5.1	Die Bedeutung von Ordnung mit Comparator und Comparable .....	1000
16.5.2	Sortieren .....	1001
16.5.3	Den größten und kleinsten Wert einer Collection finden .....	1003
16.5.4	Nichtänderbare Datenstrukturen, immutable oder nur lesen? .....	1006
16.5.5	Null Object Pattern und leere Sammlungen/Iteratoren zurückgeben .....	1009
16.5.6	Echte typsichere Container .....	1013
16.5.7	Mit der Halbierungssuche nach Elementen fahnden .....	1014
<b>16.6</b>	<b>Zum Weiterlesen .....</b>	<b>1016</b>
<b>17</b>	<b>Einführung in grafische Oberflächen .....</b>	<b>1017</b>
<b>17.1</b>	<b>GUI-Frameworks .....</b>	<b>1017</b>
17.1.1	Kommandozeile .....	1017
17.1.2	Grafische Benutzeroberfläche .....	1017
17.1.3	Abstract Window Toolkit (AWT) .....	1018

17.1.4	Java Foundation Classes und Swing .....	1018
17.1.5	JavaFX .....	1018
17.1.6	SWT (Standard Widget Toolkit) * .....	1020
<b>17.2</b>	<b>Deklarative und programmierte Oberflächen</b> .....	<b>1021</b>
17.2.1	GUI-Beschreibungen in JavaFX .....	1022
17.2.2	Deklarative GUI-Beschreibungen für Swing? .....	1022
<b>17.3</b>	<b>GUI-Builder</b> .....	<b>1023</b>
17.3.1	GUI-Builder für JavaFX .....	1023
17.3.2	GUI-Builder für Swing .....	1024
<b>17.4</b>	<b>Aller Swing-Anfang – Fenster zur Welt</b> .....	<b>1024</b>
17.4.1	Eine Uhr, bei der die Zeit nie vergeht .....	1024
17.4.2	Swing-Fenster mit javax.swing.JFrame darstellen .....	1025
17.4.3	Mit add(...) auf den Container .....	1026
17.4.4	Fenster schließbar machen – setDefaultCloseOperation(int) .....	1026
17.4.5	Sichtbarkeit des Fensters .....	1027
17.4.6	Größe und Position des Fensters verändern .....	1027
<b>17.5</b>	<b>Beschriftungen (JLabel)</b> .....	<b>1028</b>
17.5.1	Mehrzeiliger Text, HTML in der Darstellung .....	1031
<b>17.6</b>	<b>Es tut sich was – Ereignisse beim AWT</b> .....	<b>1031</b>
17.6.1	Die Ereignisquellen und Horcher (Listener) von Swing .....	1031
17.6.2	Listener implementieren .....	1033
17.6.3	Listener bei dem Ereignisauslöser anmelden/abmelden .....	1035
17.6.4	Adapterklassen nutzen .....	1036
17.6.5	Innere Mitgliedsklassen und innere anonyme Klassen .....	1038
17.6.6	Aufrufen der Listener im AWT-Event-Thread .....	1040
<b>17.7</b>	<b>Schaltflächen</b> .....	<b>1041</b>
17.7.1	Normale Schaltflächen (JButton) .....	1041
17.7.2	Der aufmerksame ActionListener .....	1043
17.7.3	Schaltflächen-Ereignisse vom Typ(ActionEvent) .....	1044
17.7.4	Basisklasse AbstractButton .....	1045
17.7.5	Wechselknopf (JToggleButton) .....	1046
<b>17.8</b>	<b>Alles Auslegungssache – die Layoutmanager</b> .....	<b>1046</b>
17.8.1	Übersicht über Layoutmanager .....	1046
17.8.2	Zuweisen eines Layoutmanagers .....	1047
17.8.3	Im Fluss mit FlowLayout .....	1048
17.8.4	BoxLayout .....	1050
17.8.5	Mit BorderLayout in alle Himmelsrichtungen .....	1050
17.8.6	Rasteranordnung mit GridLayout .....	1053
17.8.7	Weitere Layoutmanager .....	1055

<b>17.9</b>	<b>Textkomponenten</b> .....	1055
17.9.1	Text in einer Eingabezeile .....	1056
17.9.2	Die Oberklasse der Textkomponenten (JTextComponent) .....	1057
17.9.3	Geschützte Eingaben (JPasswordField) .....	1058
17.9.4	Validierende Eingabefelder (JFormattedTextField) .....	1058
17.9.5	Einfache mehrzeilige Textfelder (JTextArea) .....	1060
<b>17.10</b>	<b>Grundlegendes zum Zeichnen</b> .....	1063
17.10.1	Die paint(Graphics)-Methode für den AWT-Frame .....	1063
17.10.2	Die ereignisorientierte Programmierung ändert Fensterinhalte .....	1065
17.10.3	Zeichnen von Inhalten auf einen JFrame .....	1066
17.10.4	Auffordern zum Neuzeichnen mit repaint(...) .....	1068
17.10.5	Java 2D-API .....	1068
<b>17.11</b>	<b>Zum Weiterlesen</b> .....	1069
<b>18</b>	<b>Einführung in Dateien und Datenströme</b> .....	1071
<hr/>		
<b>18.1</b>	<b>API für Dateien, Verzeichnisse und Verzeichnissysteme</b> .....	1071
18.1.1	java.io-Paket mit File-Klasse .....	1071
18.1.2	NIO.2 und java.nio-Paket .....	1072
<b>18.2</b>	<b>Datei und Verzeichnis</b> .....	1072
18.2.1	FileSystem und Path .....	1073
18.2.2	Die Utility-Klasse Files .....	1078
18.2.3	Dateien kopieren und verschieben .....	1080
18.2.4	Neue Dateien, Verzeichnisse, symbolische Verknüpfungen anlegen und löschen .....	1083
<b>18.3</b>	<b>Dateien mit wahlfreiem Zugriff</b> .....	1084
18.3.1	Ein RandomAccessFile zum Lesen und Schreiben öffnen .....	1085
18.3.2	Aus dem RandomAccessFile lesen .....	1085
18.3.3	Schreiben mit RandomAccessFile .....	1087
18.3.4	Die Länge des RandomAccessFile .....	1087
18.3.5	Hin und her in der Datei .....	1088
<b>18.4</b>	<b>Stream-Klassen für Bytes und Zeichen</b> .....	1089
18.4.1	Lesen aus Dateien und Schreiben in Dateien .....	1089
18.4.2	Byteorientierte Datenströme über Files beziehen .....	1090
18.4.3	Zeichenorientierte Datenströme über Files beziehen .....	1090
18.4.4	Funktion von OpenOption bei den Files.newXXX(...)-Methoden .....	1092
18.4.5	Ressourcen aus dem Klassenpfad und aus JAR-Archiven laden .....	1094
18.4.6	Die Schnittstellen Closeable, AutoCloseable und Flushable .....	1095

<b>18.5</b>	<b>Basisklassen für die Ein-/Ausgabe</b> .....	1097
18.5.1	Die abstrakten Basisklassen .....	1097
18.5.2	Übersicht über Ein-/Ausgabeklassen .....	1097
18.5.3	Die abstrakte Basisklasse OutputStream .....	1100
18.5.4	Die abstrakte Basisklasse InputStream .....	1100
18.5.5	Die abstrakte Basisklasse Writer .....	1102
18.5.6	Die abstrakte Basisklasse Reader .....	1103
<b>18.6</b>	<b>Datenströme filtern und verketten</b> .....	1106
18.6.1	Streams als Filter verketten (verschachteln) .....	1106
18.6.2	Gepufferte Ausgaben mit BufferedWriter und BufferedOutputStream .....	1108
18.6.3	Gepufferte Eingaben mit BufferedReader/BufferedInputStream .....	1110
<b>18.7</b>	<b>Vermittler zwischen Byte-Streams und Unicode-Strömen</b> .....	1111
18.7.1	Datenkonvertierung durch den OutputStreamWriter .....	1112
18.7.2	Automatische Konvertierungen mit dem InputStreamReader .....	1113
<b>18.8</b>	<b>Zum Weiterlesen</b> .....	1114
<b>19</b>	<b>Einführung ins Datenbankmanagement mit JDBC</b> .....	1115
<hr/>		
<b>19.1</b>	<b>Relationale Datenbanken</b> .....	1115
19.1.1	Das relationale Modell .....	1115
<b>19.2</b>	<b>Datenbanken und Tools</b> .....	1116
19.2.1	HSQLDB .....	1116
19.2.2	Eclipse Data Tools Platform (DTP) zum Durchschauen von Datenbanken .....	1117
<b>19.3</b>	<b>JDBC und Datenbanktreiber</b> .....	1119
<b>19.4</b>	<b>Eine Beispielabfrage</b> .....	1120
19.4.1	Ein Client für die HSQLDB-Datenbank .....	1121
<b>19.5</b>	<b>Zum Weiterlesen</b> .....	1122
<b>20</b>	<b>Einführung in &lt;XML&gt;</b> .....	1123
<hr/>		
<b>20.1</b>	<b>Auszeichnungssprachen</b> .....	1123
20.1.1	Die Standard Generalized Markup Language (SGML) .....	1123
20.1.2	Extensible Markup Language (XML) .....	1124
<b>20.2</b>	<b>Eigenschaften von XML-Dokumenten</b> .....	1124
20.2.1	Elemente und Attribute .....	1124

20.2.2	Beschreibungssprache für den Aufbau von XML-Dokumenten .....	1127
20.2.3	Schema – die moderne Alternative zu DTD .....	1131
20.2.4	Namensraum (Namespace) .....	1134
20.2.5	XML-Applikationen * .....	1135
<b>20.3</b>	<b>Die Java-APIs für XML</b> .....	<b>1135</b>
20.3.1	Das Document Object Model (DOM) .....	1136
20.3.2	Simple API for XML Parsing (SAX) .....	1136
20.3.3	Pull-API StAX .....	1137
20.3.4	Java Document Object Model (JDOM) .....	1137
20.3.5	JAXP als Java-Schnittstelle zu XML .....	1137
20.3.6	DOM-Bäume einlesen mit JAXP * .....	1138
<b>20.4</b>	<b>Java Architecture for XML Binding (JAXB)</b> .....	<b>1139</b>
20.4.1	Bean für JAXB aufbauen .....	1139
20.4.2	Utility-Klasse JAXB .....	1140
20.4.3	Ganze Objektgraphen schreiben und lesen .....	1141
20.4.4	JAXBContext und Marshaller/Unmarshaller nutzen .....	1143
20.4.5	Validierung .....	1145
20.4.6	Weitere JAXB-Annotationen * .....	1148
20.4.7	Beans aus XML-Schema-Datei generieren .....	1156
<b>20.5</b>	<b>XML-Dateien mit JDOM verarbeiten</b> .....	<b>1159</b>
20.5.1	JDOM beziehen .....	1160
20.5.2	Paketübersicht * .....	1160
20.5.3	Die Document-Klasse .....	1161
20.5.4	Eingaben aus der Datei lesen .....	1162
20.5.5	Das Dokument im XML-Format ausgeben .....	1163
20.5.6	Der Dokumenttyp * .....	1164
20.5.7	Elemente .....	1165
20.5.8	Zugriff auf Elementinhalte .....	1168
20.5.9	Liste mit Unterelementen erzeugen * .....	1170
20.5.10	Neue Elemente einfügen und ändern .....	1171
20.5.11	Attributinhalte lesen und ändern .....	1174
20.5.12	XPath .....	1177
<b>20.6</b>	<b>Zum Weiterlesen</b> .....	<b>1181</b>
<b>21</b>	<b>Testen mit JUnit</b> .....	<b>1183</b>
<b>21.1</b>	<b>Softwaretests</b> .....	<b>1183</b>
21.1.1	Vorgehen beim Schreiben von Testfällen .....	1184

<b>21.2</b>	<b>Das Test-Framework JUnit</b> .....	1184
21.2.1	Test-Driven Development und Test-First .....	1184
21.2.2	Testen, implementieren, testen, implementieren, testen, freuen .....	1186
21.2.3	JUnit-Tests ausführen .....	1187
21.2.4	assertXXX(...)-Methoden der Klasse Assert .....	1188
21.2.5	Matcher-Objekte und Hamcrest .....	1190
21.2.6	Exceptions testen .....	1194
21.2.7	Tests ignorieren und Grenzen für Ausführungszeiten festlegen .....	1195
21.2.8	Mit Methoden der Assume-Klasse Tests abbrechen .....	1196
<b>21.3</b>	<b>Wie gutes Design das Testen ermöglicht</b> .....	1196
<b>21.4</b>	<b>Aufbau größerer Testfälle</b> .....	1199
21.4.1	Fixtures .....	1199
21.4.2	Sammlungen von Testklassen und Klassenorganisation .....	1201
<b>21.5</b>	<b>Dummy, Fake, Stub und Mock</b> .....	1202
<b>21.6</b>	<b>JUnit-Erweiterungen, Testzusätze</b> .....	1203
<b>21.7</b>	<b>Zum Weiterlesen</b> .....	1204
<b>22</b>	<b>Bits und Bytes und Mathematisches</b> .....	1205
<hr/>		
<b>22.1</b>	<b>Bits und Bytes *</b> .....	1205
22.1.1	Die Bit-Operatoren Komplement, Und, Oder und XOR .....	1206
22.1.2	Repräsentation ganzer Zahlen in Java – das Zweierkomplement .....	1207
22.1.3	Das binäre (Basis 2), oktale (Basis 8), hexadezimale (Basis 16) Stellenwertsystem .....	1208
22.1.4	Auswirkung der Typumwandlung auf die Bit-Muster .....	1209
22.1.5	Vorzeichenlos arbeiten .....	1212
22.1.6	Die Verschiebeoperatoren .....	1215
22.1.7	Ein Bit setzen, löschen, umdrehen und testen .....	1217
22.1.8	Bit-Methoden der Integer- und Long-Klasse .....	1217
<b>22.2</b>	<b>Fließkomma-Arithmetik in Java</b> .....	1219
22.2.1	Spezialwerte für Unendlich, Null, NaN .....	1219
22.2.2	Standardnotation und wissenschaftliche Notation bei Fließkommazahlen *	1222
22.2.3	Mantisse und Exponent *	1223
<b>22.3</b>	<b>Die Eigenschaften der Klasse Math</b> .....	1225
22.3.1	Attribute .....	1225
22.3.2	Absolutwerte und Vorzeichen .....	1225



22.3.3	Maximum/Minimum .....	1226
22.3.4	Runden von Werten .....	1226
22.3.5	Rest der ganzzahligen Division * .....	1229
22.3.6	Division mit Rundung Richtung negativ unendlich, alternativer Restwert * ..	1230
22.3.7	Wurzel- und Exponentialmethoden .....	1232
22.3.8	Der Logarithmus * .....	1233
22.3.9	Winkelmethoden * .....	1234
22.3.10	Zufallszahlen .....	1235
<b>22.4</b>	<b>Genauigkeit, Wertebereich eines Typs und Überlaufkontrolle *</b> .....	1235
22.4.1	Der größte und der kleinste Wert .....	1235
22.4.2	Überlauf .....	1236
22.4.3	Was bitte macht eine ulp? .....	1238
<b>22.5</b>	<b>Zufallszahlen: Random, SecureRandom, SplittableRandom</b> .....	1240
22.5.1	Die Klasse Random .....	1240
22.5.2	Random-Objekte mit dem Samen aufbauen .....	1240
22.5.3	Einzelne Zufallszahlen erzeugen .....	1241
22.5.4	Pseudo-Zufallszahlen in der Normalverteilung * .....	1241
22.5.5	Strom von Zufallszahlen generieren * .....	1242
22.5.6	Die Klasse SecureRandom * .....	1243
22.5.7	SplittableRandom * .....	1244
<b>22.6</b>	<b>Große Zahlen *</b> .....	1244
22.6.1	Die Klasse BigInteger .....	1244
22.6.2	Beispiel: ganz lange Fakultäten mit BigInteger .....	1251
22.6.3	Große Fließkommazahlen mit BigDecimal .....	1252
22.6.4	Mit MathContext komfortabel die Rechengenauigkeit setzen .....	1255
<b>22.7</b>	<b>Mathe bitte strikt *</b> .....	1256
22.7.1	Strikte Fließkommaberechnungen mit strictfp .....	1256
22.7.2	Die Klassen Math und StrictMath .....	1257
<b>22.8</b>	<b>Zum Weiterlesen</b> .....	1257
<b>23</b>	<b>Die Werkzeuge des JDK</b> .....	1259
<hr/>		
<b>23.1</b>	<b>Java-Quellen übersetzen</b> .....	1259
23.1.1	Java-Compiler vom JDK .....	1260
23.1.2	Alternative Compiler .....	1260
23.1.3	Native Compiler .....	1261
23.1.4	Java-Programme in ein natives ausführbares Programm einpacken .....	1261

<b>23.2 Die Java-Laufzeitumgebung</b> .....	1262
23.2.1 Schalter der JVM .....	1262
23.2.2 Der Unterschied zwischen java.exe und javaw.exe .....	1265
<b>23.3 Mit RoboVM geht's für Java in das iOS-Land *</b> .....	1265
<b>23.4 Dokumentationskommentare mit Javadoc</b> .....	1266
23.4.1 Einen Dokumentationskommentar setzen .....	1266
23.4.2 Mit dem Werkzeug javadoc eine Dokumentation erstellen .....	1268
23.4.3 HTML-Tags in Dokumentationskommentaren * .....	1269
23.4.4 Generierte Dateien .....	1269
23.4.5 Dokumentationskommentare im Überblick * .....	1270
23.4.6 Javadoc und Doclets * .....	1271
23.4.7 Veraltete (deprecated) Typen und Eigenschaften .....	1272
23.4.8 Javadoc-Überprüfung mit DocLint .....	1275
<b>23.5 Das Archivformat JAR</b> .....	1275
23.5.1 Das Dienstprogramm jar benutzen .....	1276
23.5.2 Das Manifest .....	1278
23.5.3 Applikationen in JAR-Archiven starten .....	1279
23.5.4 Pack200-Format * .....	1280
<b>23.6 Zum Weiterlesen</b> .....	1281
Java SE-Paketübersicht .....	1283
Index .....	1295