

Inhalt

Vorwort	31
1 Java ist auch eine Sprache	49
1.1 Historischer Hintergrund	49
1.2 Warum Java gut ist – die zentralen Eigenschaften	51
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 *	67
1.2.17 Java und das Web, Applets und JavaFX	69
1.2.18 Features, Enhancements (Erweiterungen) und ein JSR	71
1.2.19 Die Entwicklung von Java und seine Zukunftsaussichten	72
1.3 Java-Plattformen: Java SE, Java EE, Java ME und Java Card	73
1.3.1 Die Java SE-Plattform	73
1.3.2 Java für die Kleinen	75
1.3.3 Java für die ganz, ganz Kleinen	76
1.3.4 Java für die Großen	76
1.3.5 Echtzeit-Java (Real-time Java)	76
1.4 Die Installation der Java Platform Standard Edition (Java SE)	77
1.4.1 Die Java SE von Oracle	77
1.4.2 Download des JDK	78
1.4.3 Java SE unter Windows installieren	80
1.4.4 JDK/JRE deinstallieren	84

1.4.5	JDK unter Linux installieren	85
1.4.6	JDK unter Max OS X installieren	85
1.5	Das erste Programm compilieren und testen	86
1.5.1	Compilertest	87
1.5.2	Ein Quadratzahlen-Programm	88
1.5.3	Der Compilerlauf	89
1.5.4	Die Laufzeitumgebung	90
1.5.5	Häufige Compiler- und Interpreter-Probleme	90
1.6	Entwicklungsumgebungen im Allgemeinen	91
1.6.1	Die Entwicklungsumgebung Eclipse	91
1.6.2	NetBeans von Oracle	92
1.6.3	IntelliJ IDEA	93
1.6.4	Ein Wort zu Microsoft, Java und zu J++, J#	93
1.7	Eclipse im Speziellen	94
1.7.1	Eclipse entpacken und starten	96
1.7.2	Das erste Projekt anlegen	97
1.7.3	Verzeichnisstruktur für Java-Projekte *	97
1.7.4	Eine Klasse hinzufügen	99
1.7.5	Übersetzen und ausführen	99
1.7.6	Projekt einfügen, Workspace für die Programme wechseln	100
1.7.7	Plugins für Eclipse	100
1.8	NetBeans im Speziellen	101
1.8.1	NetBeans-Bundles	101
1.8.2	NetBeans installieren	102
1.8.3	NetBeans starten	103
1.8.4	Ein neues NetBeans-Projekt anlegen	103
1.8.5	Ein Java-Programm starten	104
1.9	Zum Weiterlesen	104
2	Imperative Sprachkonzepte	107
<hr/>		
2.1	Elemente der Programmiersprache Java	107
2.1.1	Token	107
2.1.2	Textkodierung durch Unicode-Zeichen	108
2.1.3	Bezeichner	108
2.1.4	Literale	111
2.1.5	Reservierte Schlüsselwörter	111

2.1.6	Zusammenfassung der lexikalischen Analyse	112
2.1.7	Kommentare	113
2.2	Von der Klasse zur Anweisung	115
2.2.1	Was sind Anweisungen?	115
2.2.2	Klassendeklaration	115
2.2.3	Die Reise beginnt am main(String[])	116
2.2.4	Der erste Methodenaufruf: println(...)	117
2.2.5	Atomare Anweisungen und Anweisungssequenzen	118
2.2.6	Mehr zu print(...), println(...) und printf(...) für Bildschirmausgaben	118
2.2.7	Die API-Dokumentation	120
2.2.8	Ausdrücke	122
2.2.9	Ausdrucksanweisung	123
2.2.10	Erste Idee der Objektorientierung	123
2.2.11	Modifizierer	125
2.2.12	Gruppieren von Anweisungen mit Blöcken	125
2.3	Datentypen, Typisierung, Variablen und Zuweisungen	126
2.3.1	Primitive Datentypen im Überblick	128
2.3.2	Variablendeklarationen	131
2.3.3	Konsoleneingaben	134
2.3.4	Fließkommazahlen mit den Datentypen float und double	136
2.3.5	Ganzzahlige Datentypen	137
2.3.6	Wahrheitswerte	139
2.3.7	Unterstriche in Zahlen *	139
2.3.8	Alphanumerische Zeichen	140
2.3.9	Gute Namen, schlechte Namen	141
2.3.10	Initialisierung von lokalen Variablen	142
2.4	Ausdrücke, Operanden und Operatoren	142
2.4.1	Zuweisungsoperator	143
2.4.2	Arithmetische Operatoren	145
2.4.3	Unäres Minus und Plus	148
2.4.4	Zuweisung mit Operation	149
2.4.5	Präfix- oder Postfix-Inkrement und -Dekrement	150
2.4.6	Die relationalen Operatoren und die Gleichheitsoperatoren	152
2.4.7	Logische Operatoren: Nicht, Und, Oder, XOR	154
2.4.8	Kurzschluss-Operatoren	155
2.4.9	Der Rang der Operatoren in der Auswertungsreihenfolge	156
2.4.10	Die Typanpassung (das Casting)	159
2.4.11	Überladenes Plus für Strings	164
2.4.12	Operator vermisst *	166

2.5	Bedingte Anweisungen oder Fallunterscheidungen	166
2.5.1	Verzweigung mit der if-Anweisung	166
2.5.2	Die Alternative mit einer if-else-Anweisung wählen	168
2.5.3	Der Bedingungsoperator	172
2.5.4	Die switch-Anweisung bietet die Alternative	174
2.6	Immer das Gleiche mit den Schleifen	180
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	Ausbruch planen mit break und Wiedereinstieg mit continue	190
2.6.6	break und continue mit Marken *	193
2.7	Methoden einer Klasse	197
2.7.1	Bestandteil einer Methode	197
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	206
2.7.10	Methoden überladen	211
2.7.11	Sichtbarkeit und Gültigkeitsbereich	213
2.7.12	Vorgegebener Wert für nicht aufgeführte Argumente *	215
2.7.13	Finale lokale Variablen	215
2.7.14	Rekursive Methoden *	216
2.7.15	Die Türme von Hanoi *	221
2.8	Zum Weiterlesen	223

3 Klassen und Objekte 225

3.1	Objektorientierte Programmierung (OOP)	225
3.1.1	Warum überhaupt OOP?	225
3.1.2	Denk ich an Java, denk ich an Wiederverwendbarkeit	226
3.2	Eigenschaften einer Klasse	227
3.2.1	Klassenarbeit mit Point	228
3.3	Natürlich modellieren mit der UML (Unified Modeling Language) *	228
3.3.1	Hintergrund und Geschichte der UML	229

3.3.2	Wichtige Diagrammtypen der UML	230
3.3.3	UML-Werkzeuge	231
3.4	Neue Objekte erzeugen	232
3.4.1	Ein Exemplar einer Klasse mit dem new-Operator anlegen	232
3.4.2	Der Zusammenhang von new, Heap und Garbage-Collector	233
3.4.3	Deklarieren von Referenzvariablen	234
3.4.4	Zugriff auf Objektattribute und -methoden mit dem ».«	235
3.4.5	Überblick über Point-Methoden	239
3.4.6	Konstruktoren nutzen	242
3.5	ZZZZZnake	243
3.6	Pakete schnüren, Imports und Kompilationseinheiten	246
3.6.1	Java-Pakete	246
3.6.2	Pakete der Standardbibliothek	246
3.6.3	Volle Qualifizierung und import-Deklaration	247
3.6.4	Mit import p1.p2.* alle Typen eines Pakets erreichen	248
3.6.5	Hierarchische Strukturen über Pakete	249
3.6.6	Die package-Deklaration	249
3.6.7	Unbenanntes Paket (default package)	250
3.6.8	Klassen mit gleichen Namen in unterschiedlichen Paketen *	251
3.6.9	Kompilationseinheit (Compilation Unit)	252
3.6.10	Statischer Import *	252
3.7	Mit Referenzen arbeiten, Identität und Gleichheit	253
3.7.1	null-Referenz und die Frage der Philosophie	253
3.7.2	Alles auf null? Referenzen testen	255
3.7.3	Zuweisungen bei Referenzen	256
3.7.4	Methoden mit Referenztypen als Parametern	258
3.7.5	Identität von Objekten	262
3.7.6	Gleichheit und die Methode equals(...)	263
3.8	Arrays	265
3.8.1	Grundbestandteile	265
3.8.2	Deklaration von Arrays	266
3.8.3	Arrays mit Inhalt	266
3.8.4	Die Länge eines Arrays über das Attribut length auslesen	267
3.8.5	Zugriff auf die Elemente über den Index	267
3.8.6	Array-Objekte mit new erzeugen	269
3.8.7	Typische Feldfehler	270
3.8.8	Feld-Objekte als Parametertyp	272
3.8.9	Vorinitialisierte Arrays	272
3.8.10	Die erweiterte for-Schleife	273

3.8.11	Arrays mit nichtprimitiven Elementen	275
3.8.12	Methode mit variabler Argumentanzahl (Vararg)	278
3.8.13	Mehrdimensionale Arrays *	281
3.8.14	Nichtrechteckige Arrays *	284
3.8.15	Die Wahrheit über die Array-Initialisierung *	286
3.8.16	Mehrere Rückgabewerte *	287
3.8.17	Klonen kann sich lohnen – Arrays vermehren *	288
3.8.18	Feldinhalte kopieren *	289
3.8.19	Die Klasse Arrays zum Vergleichen, Füllen, Suchen, Sortieren nutzen	290
3.8.20	Eine lange Schlange	302
3.9	Der Einstiegspunkt für das Laufzeitsystem: main(...)	304
3.9.1	Korrekte Deklaration der Startmethode	304
3.9.2	Kommandozeilenargumente verarbeiten	305
3.9.3	Der Rückgabotyp von main(...) und System.exit(int) *	306
3.10	Grundlagen von Annotationen und Generics	308
3.10.1	Generics	308
3.10.2	Annotationen	309
3.10.3	Annotationstypen aus java.lang	311
3.10.4	@Deprecated	311
3.10.5	@SuppressWarnings	312
3.11	Zum Weiterlesen	316

4 Der Umgang mit Zeichenketten 317

4.1	Von ASCII über ISO-8859-1 zu Unicode	317
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	322
4.1.6	Schreibweise für Unicode-Zeichen und Unicode-Escapes	322
4.1.7	Java-Versionen gehen mit Unicode-Standard Hand in Hand *	325
4.2	Die Character-Klasse	326
4.2.1	Ist das so?	326
4.2.2	Zeichen in Großbuchstaben/Kleinbuchstaben konvertieren	328
4.2.3	Ziffern einer Basis *	330
4.3	Zeichenfolgen	330

4.4	Die Klasse String und ihre Methoden	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()	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	352
4.4.11	String-Objekte mit Konstruktoren neu anlegen *	354
4.5	Veränderbare Zeichenketten mit StringBuilder und StringBuffer	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	365
4.5.8	hashCode() bei StringBuilder/StringBuffer *	366
4.6	CharSequence als Basistyp	367
4.7	Strings mit StringJoiner zusammenhängen	369
4.8	Konvertieren zwischen Primitiven und Strings	371
4.8.1	Unterschiedliche Typen in String-Repräsentationen konvertieren	371
4.8.2	String-Inhalt in einen primitiven Wert konvertieren	373
4.8.3	String-Repräsentation im Format Binär, Hex, Oktal *	375
4.8.4	parseXXX(...)- und printXXX()-Methoden in DatatypeConverter *	378
4.9	Zerlegen von Zeichenketten	379
4.9.1	Splitten von Zeichenketten mit split(...)	380
4.9.2	Yes we can, yes we scan – die Klasse Scanner	381
4.10	Ausgaben formatieren	384
4.10.1	Formatieren und Ausgeben mit format()	385
4.11	Zum Weiterlesen	391

5.1	Eigene Klassen mit Eigenschaften deklarieren	393
5.1.1	Attribute deklarieren	394
5.1.2	Methoden deklarieren	396
5.1.3	Die this-Referenz	400
5.2	Privatsphäre und Sichtbarkeit	403
5.2.1	Für die Öffentlichkeit: public	404
5.2.2	Kein Public Viewing – Passwörter sind privat	404
5.2.3	Wieso nicht freie Methoden und Variablen für alle?	405
5.2.4	Privat ist nicht ganz privat: Es kommt darauf an, wer's sieht *	406
5.2.5	Zugriffsmethoden für Attribute deklarieren	406
5.2.6	Setter und Getter nach der JavaBeans-Spezifikation	407
5.2.7	Paketsichtbar	410
5.2.8	Zusammenfassung zur Sichtbarkeit	411
5.3	Eine für alle – statische Methode und statische Attribute	414
5.3.1	Warum statische Eigenschaften sinnvoll sind	414
5.3.2	Statische Eigenschaften mit static	415
5.3.3	Statische Eigenschaften über Referenzen nutzen? *	416
5.3.4	Warum die Groß- und Kleinschreibung wichtig ist *	417
5.3.5	Statische Variablen zum Datenaustausch *	417
5.3.6	Statische Eigenschaften und Objekteigenschaften *	419
5.4	Konstanten und Aufzählungen	420
5.4.1	Konstanten über statische finale Variablen	420
5.4.2	Typunsichere Aufzählungen	421
5.4.3	Aufzählungstypen: Typsichere Aufzählungen mit enum	423
5.5	Objekte anlegen und zerstören	428
5.5.1	Konstruktoren schreiben	428
5.5.2	Der vorgegebene Konstruktor (default constructor)	429
5.5.3	Parametrisierte und überladene Konstruktoren	432
5.5.4	Copy-Konstruktor	434
5.5.5	Einen anderen Konstruktor der gleichen Klasse mit this(...) aufrufen	435
5.5.6	Ihr fehlt uns nicht – der Garbage-Collector	438
5.6	Klassen- und Objektinitialisierung *	440
5.6.1	Initialisierung von Objektvariablen	440
5.6.2	Statische Blöcke als Klasseninitialisierer	442
5.6.3	Initialisierung von Klassenvariablen	443
5.6.4	Eincompilierte Belegungen der Klassenvariablen	444

5.6.5	Exemplarinitialisierer (Instanzinitialisierer)	444
5.6.6	Finale Werte im Konstruktor und in statischen Blöcken setzen	448
5.7	Zum Weiterlesen	450
6	Objektorientierte Beziehungsfragen	451
6.1	Assoziationen zwischen Objekten	451
6.1.1	Unidirektionale 1:1-Beziehung	452
6.1.2	Zwei Freunde müsst ihr werden – bidirektionale 1:1-Beziehungen	453
6.1.3	Unidirektionale 1:n-Beziehung	454
6.2	Vererbung	456
6.2.1	Vererbung in Java	457
6.2.2	Spielobjekte modellieren	458
6.2.3	Die implizite Basisklasse java.lang.Object	459
6.2.4	Einfach- und Mehrfachvererbung *	460
6.2.5	Die Sichtbarkeit protected	460
6.2.6	Konstruktoren in der Vererbung und super(..)	461
6.3	Typen in Hierarchien	466
6.3.1	Automatische und explizite Typanpassung	466
6.3.2	Das Substitutionsprinzip	469
6.3.3	Typen mit dem instanceof-Operator testen	470
6.4	Methoden überschreiben	472
6.4.1	Methoden in Unterklassen mit neuem Verhalten ausstatten	473
6.4.2	Mit super an die Eltern	476
6.4.3	Finale Klassen und finale Methoden	478
6.4.4	Kovariante Rückgabetypen	480
6.4.5	Array-Typen und Kovarianz *	481
6.5	Drum prüfe, wer sich ewig dynamisch bindet	482
6.5.1	Gebunden an toString()	482
6.5.2	Implementierung von System.out.println(Object)	485
6.5.3	Nicht dynamisch gebunden bei privaten, statischen und finalen Methoden	485
6.5.4	Dynamisch gebunden auch bei Konstruktor-Aufrufen *	487
6.5.5	Eine letzte Spielerei mit Javas dynamischer Bindung und überschatteten Attributen *	489
6.6	Abstrakte Klassen und abstrakte Methoden	490
6.6.1	Abstrakte Klassen	490
6.6.2	Abstrakte Methoden	492

6.7	Schnittstellen	495
6.7.1	Schnittstellen deklarieren	496
6.7.2	Implementieren von Schnittstellen	497
6.7.3	Ein Polymorphie-Beispiel mit Schnittstellen	498
6.7.4	Die Mehrfachvererbung bei Schnittstellen	500
6.7.5	Keine Kollisionsgefahr bei Mehrfachvererbung *	504
6.7.6	Erweitern von Interfaces – Subinterfaces	505
6.7.7	Konstantendeklarationen bei Schnittstellen	506
6.7.8	Statische ausprogrammierte Methoden in Schnittstellen	508
6.7.9	Erweitern von Schnittstellen	510
6.7.10	Default-Methoden	511
6.7.11	Erweiterte Schnittstellen deklarieren und nutzen	513
6.7.12	Erweiterte Schnittstellen, Mehrfachvererbung und Mehrdeutigkeiten *	516
6.7.13	Bausteine bilden mit Default-Methoden *	520
6.7.14	Initialisierung von Schnittstellenkonstanten *	526
6.7.15	Markierungsschnittstellen *	530
6.7.16	Abstrakte Klassen und Schnittstellen im Vergleich	530
6.8	Zum Weiterlesen	531

7 Ausnahmen müssen sein 533

7.1	Problembereiche einzäunen	533
7.1.1	Exceptions in Java mit try und catch	534
7.1.2	Eine NumberFormatException auffangen	534
7.1.3	Eigenschaften vom Exception-Objekt	537
7.1.4	Wiederholung abgebrochener Bereiche *	539
7.1.5	Mehrere Ausnahmen auffangen	540
7.1.6	Ablauf einer Ausnahmesituation	542
7.1.7	throws im Methodenkopf angeben	543
7.1.8	Abschlussbehandlung mit finally	544
7.2	RuntimeException muss nicht aufgefangen werden	549
7.2.1	Beispiele für RuntimeException-Klassen	550
7.2.2	Kann man abfangen, muss man aber nicht	550
7.3	Die Klassenhierarchie der Fehler	551
7.3.1	Die Exception-Hierarchie	552
7.3.2	Oberausnahmen auffangen	552
7.3.3	Schon gefangen?	554
7.3.4	Alles geht als Exception durch	554
7.3.5	Zusammenfassen gleicher catch-Blöcke mit dem multi-catch	556

7.4	Harte Fehler – Error *	559
7.5	Auslösen eigener Exceptions	560
7.5.1	Mit throw Ausnahmen auslösen	560
7.5.2	Vorhandene Runtime-Fehlertypen kennen und nutzen	562
7.5.3	Parameter testen und gute Fehlermeldungen	565
7.5.4	Neue Exception-Klassen deklarieren	566
7.5.5	Eigene Ausnahmen als Unterklassen von Exception oder RuntimeException?	568
7.5.6	Ausnahmen abfangen und weiterleiten *	571
7.5.7	Aufruf-Stack von Ausnahmen verändern *	572
7.5.8	Präzises rethrow *	573
7.5.9	Geschachtelte Ausnahmen *	576
7.6	Automatisches Ressourcen-Management (try mit Ressourcen)	579
7.6.1	try mit Ressourcen	580
7.6.2	Die Schnittstelle AutoCloseable	582
7.6.3	Mehrere Ressourcen nutzen	584
7.6.4	try mit Ressourcen auf null-Ressourcen	585
7.6.5	Unterdrückte Ausnahmen *	585
7.7	Besonderheiten bei der Ausnahmebehandlung *	589
7.7.1	Rückgabewerte bei ausgelösten Ausnahmen	589
7.7.2	Ausnahmen und Rückgaben verschwinden – das Duo return und finally	589
7.7.3	throws bei überschriebenen Methoden	590
7.7.4	Nicht erreichbare catch-Klauseln	593
7.8	Den Stack-Trace erfragen *	594
7.8.1	StackTraceElement	594
7.8.2	printStackTrace(...)	595
7.8.3	StackTraceElement vom Thread erfragen	596
7.9	Assertions *	597
7.9.1	Assertions in eigenen Programmen nutzen	597
7.9.2	Assertions aktivieren	599
7.10	Zum Weiterlesen	601
8	Äußere.innere Klassen	603
8.1	Geschachtelte (innere) Klassen, Schnittstellen, Aufzählungen	603
8.2	Statische innere Klassen und Schnittstellen	604
8.3	Mitglieds- oder Elementklassen	606
8.3.1	Exemplare innerer Klassen erzeugen	606

8.3.2	Die this-Referenz	607
8.3.3	Vom Compiler generierte Klassendateien *	608
8.3.4	Erlaubte Modifizierer bei äußeren und inneren Klassen	609
8.3.5	Innere Klassen greifen auf private Eigenschaften zu	609
8.4	Lokale Klassen	611
8.4.1	Beispiel mit eigener Klassendeklaration	611
8.4.2	Lokale Klasse für einen Timer nutzen	612
8.5	Anonyme innere Klassen	613
8.5.1	Nutzung einer anonymen inneren Klasse für den Timer	613
8.5.2	Umsetzung innerer anonymer Klassen *	614
8.5.3	Konstruktoren innerer anonymer Klassen	615
8.6	Zugriff auf lokale Variablen aus lokalen inneren und anonymen Klassen *	616
8.7	this in Unterklassen *	617
8.8	Zum Weiterlesen	618
9	Besondere Typen der Java SE	619
<hr/>		
9.1	Vergleichen von Objekten	620
9.1.1	Natürlich geordnet oder nicht?	620
9.1.2	Die Schnittstelle Comparable	621
9.1.3	Die Schnittstelle Comparator	621
9.1.4	Rückgabewerte kodieren die Ordnung	622
9.1.5	Statische und Default-Methoden in Comparator	624
9.2	Wrapper-Klassen und Autoboxing	627
9.2.1	Wrapper-Objekte erzeugen	629
9.2.2	Konvertierungen in eine String-Repräsentation	630
9.2.3	Von einer String-Repräsentation parsen	631
9.2.4	Die Basisklasse Number für numerische Wrapper-Objekte	632
9.2.5	Vergleiche durchführen mit compare(...), compareTo(...), equals(...) und Hashwerten	633
9.2.6	Statische Reduzierungsmethoden in Wrapper-Klassen	636
9.2.7	Die Größe eines primitiven Typs in den Wrapper-Konstanten SIZE und BYTES	637
9.2.8	Behandeln von vorzeichenlosen Zahlen *	638
9.2.9	Die Klasse Integer	639
9.2.10	Die Klassen Double und Float für Fließkommazahlen	640
9.2.11	Die Long-Klasse	641

9.2.12	Die Boolean-Klasse	641
9.2.13	Autoboxing: Boxing und Unboxing	642
9.3	Object ist die Mutter aller Klassen	646
9.3.1	Klassenobjekte	646
9.3.2	Objektidentifikation mit toString()	647
9.3.3	Objektgleichheit mit equals(...) und Identität	649
9.3.4	Klonen eines Objekts mit clone() *	653
9.3.5	Hashwerte über hashCode() liefern *	658
9.3.6	System.identityHashCode(...) und das Problem der nicht eindeutigen Objektverweise *	665
9.3.7	Aufräumen mit finalize() *	666
9.3.8	Synchronisation *	668
9.4	Die Utility-Klasse java.util.Objects	669
9.5	Iterator, Iterable *	671
9.5.1	Die Schnittstelle Iterator	672
9.5.2	Wer den Iterator liefert	674
9.5.3	Die Schnittstelle Iterable	675
9.5.4	Erweitertes for und Iterable	676
9.5.5	Interne Iteration (seit Java 8)	676
9.5.6	Einen eigenen Iterable implementieren *	677
9.6	Die Spezial-Oberklasse Enum	678
9.6.1	Methoden auf Enum-Objekten	679
9.6.2	Aufzählungen mit eigenen Methoden *	682
9.6.3	enum mit eigenen Konstruktoren *	684
9.7	Zum Weiterlesen	687
10	Generics<T>	689
10.1	Einführung in Java Generics	689
10.1.1	Mensch versus Maschine – Typprüfung des Compilers und der Laufzeitumgebung	689
10.1.2	Taschen	690
10.1.3	Generische Typen deklarieren	692
10.1.4	Generics nutzen	694
10.1.5	Diamonds are forever	696
10.1.6	Generische Schnittstellen	699
10.1.7	Generische Methoden/Konstruktoren und Typ-Inferenz	701

10.2	Umsetzen der Generics, Typlöschung und Raw-Types	705
10.2.1	Realisierungsmöglichkeiten	705
10.2.2	Typlöschung (Type Erasure)	705
10.2.3	Probleme der Typlöschung	707
10.2.4	Raw-Type	711
10.3	Einschränken der Typen über Bounds	714
10.3.1	Einfache Einschränkungen mit extends	714
10.3.2	Weitere Obertypen mit &	716
10.4	Typparameter in der throws-Klausel *	717
10.4.1	Deklaration einer Klasse mit Typvariable <E extends Exception>	717
10.4.2	Parametrisierter Typ bei Typvariable <E extends Exception>	718
10.5	Generics und Vererbung, Invarianz	720
10.5.1	Arrays sind invariant	721
10.5.2	Generics sind kovariant	721
10.5.3	Wildcards mit ?	722
10.5.4	Bounded Wildcards	724
10.5.5	Bounded-Wildcard-Typen und Bounded-Typvariablen	728
10.5.6	Das LESS-Prinzip	730
10.5.7	Enum<E extends Enum<E>> *	732
10.6	Konsequenzen der Typlöschung: Typ-Token, Arrays und Brücken *	734
10.6.1	Typ-Token	734
10.6.2	Super-Type-Token	736
10.6.3	Generics und Arrays	737
10.6.4	Brückenmethoden	738
10.6.5	Zum Weiterlesen	744

11 Lambda-Ausdrücke und funktionale Programmierung 745

11.1	Code = Daten	745
11.2	Funktionale Schnittstellen und Lambda-Ausdrücke im Detail	748
11.2.1	Funktionale Schnittstellen	749
11.2.2	Typ eines Lambda-Ausdrucks ergibt sich durch Zieltyp	750
11.2.3	Annotation @FunctionalInterface	754
11.2.4	Syntax für Lambda-Ausdrücke	754
11.2.5	Die Umgebung der Lambda-Ausdrücke und Variablenzugriffe	759
11.2.6	Ausnahmen in Lambda-Ausdrücken	762
11.2.7	Klassen mit einer abstrakten Methode als funktionale Schnittstelle? *	766

11.3 Methoden-Referenz	767
11.3.1 Varianten von Methoden-Referenzen	769
11.4 Konstruktor-Referenz	770
11.4.1 Standard- und parametrisierte Konstruktoren	772
11.4.2 Nützliche vordefinierte Schnittstellen für Konstruktor-Referenzen	773
11.5 Implementierung von Lambda-Ausdrücken *	774
11.6 Funktionale Programmierung mit Java	774
11.6.1 Programmierparadigmen: imperativ oder deklarativ	774
11.6.2 Funktionale Programmierung und funktionale Programmiersprachen	775
11.6.3 Funktionale Programmierung in Java am Beispiel vom Comparator	777
11.6.4 Lambda-Ausdrücke als Funktionen sehen	778
11.7 Funktionale Schnittstelle aus dem java.util.function-Paket	779
11.7.1 Blöcke mit Code und die funktionale Schnittstelle java.util.function.Consumer	780
11.7.2 Supplier	781
11.7.3 Prädikate und java.util.function.Predicate	782
11.7.4 Funktionen und die allgemeine funktionale Schnittstelle java.util.function.Function	784
11.7.5 Ein bisschen Bi	788
11.7.6 Funktionale Schnittstellen mit Primitiven	791
11.8 Optional ist keine Nullnummer	793
11.8.1 Optional-Typ	796
11.8.2 Primitive optionale Typen	798
11.8.3 Erstmals funktional mit Optional	799
11.9 Was ist jetzt so funktional?	803
11.10 Zum Weiterlesen	806

12 Architektur, Design und angewandte Objektorientierung 807

12.1 Architektur, Design und Implementierung	807
12.2 Design-Pattern (Entwurfsmuster)	808
12.2.1 Motivation für Design-Pattern	808
12.2.2 Singleton	809
12.2.3 Fabrikmethoden	812
12.2.4 Das Beobachter-Pattern (Observer/Observable)	813
12.2.5 Ereignisse über Listener	818

12.3	JavaBeans	822
12.3.1	Properties (Eigenschaften)	823
12.3.2	Einfache Eigenschaften	823
12.3.3	Indizierte Eigenschaften	824
12.3.4	Gebundene Eigenschaften und PropertyChangeListener	824
12.3.5	Veto-Eigenschaften – dagegen!	828
12.4	JavaFX Properties	831
12.4.1	javafx.beans-Paket mit XXXProperty-Klassen	832
12.4.2	Property-Veränderungen registrieren	833
12.4.3	Beans-Binding	834
12.4.4	Property-Schnittstelle und bindXXX()-Methoden	835
12.4.5	XXXProperty-Beziehungen (für Typ-Fetischisten) *	842
12.4.6	Ausblick	844
12.5	Zum Weiterlesen	844
13	Die Klassenbibliothek	845
<hr/>		
13.1	Die Java-Klassenphilosophie	845
13.1.1	Übersicht über die Pakete der Standardbibliothek	845
13.1.2	Compact-Profile	848
13.2	Sprachen der Länder	850
13.2.1	Sprachen und Regionen über Locale-Objekte	850
13.3	Die Klasse Date	853
13.3.1	Objekte erzeugen und Methoden nutzen	853
13.3.2	Date-Objekte sind nicht immutable	855
13.4	Calendar und GregorianCalendar	856
13.4.1	Die abstrakte Klasse Calendar	856
13.4.2	Calendar nach Date und Millisekunden fragen	857
13.4.3	Abfragen und Setzen von Datumselementen über Feldbezeichner	858
13.4.4	Kalender-Exemplare bauen über den Calendar.Builder	862
13.4.5	Der gregorianische Kalender	862
13.4.6	Date-Time-API in Java 8	864
13.5	Klassenlader (Class Loader) und Klassenpfad	869
13.5.1	Woher die kleinen Klassen kommen	869
13.5.2	Setzen des Klassenpfades	871
13.5.3	Die wichtigsten drei Typen von Klassenladern	871
13.6	Die Utility-Klasse System und Properties	872
13.6.1	Systemeigenschaften der Java-Umgebung	873

13.6.2	line.separator	875
13.6.3	Eigene Properties von der Konsole aus setzen *	875
13.6.4	Umgebungsvariablen des Betriebssystems *	878
13.6.5	Einfache Zeitmessung und Profiling *	879
13.7	Einfache Benutzereingaben	882
13.7.1	Grafischer Eingabedialog über JOptionPane	882
13.7.2	Geschützte Passwort-Eingaben mit der Klasse Console *	883
13.8	Ausführen externer Programme *	884
13.8.1	ProcessBuilder und Prozesskontrolle mit Process	885
13.8.2	Einen Browser, E-Mail-Client oder Editor aufrufen	891
13.9	Benutzereinstellungen *	892
13.9.1	Benutzereinstellungen mit der Preferences-API	892
13.9.2	Einträge einfügen, auslesen und löschen	894
13.9.3	Auslesen der Daten und Schreiben in einem anderen Format	896
13.9.4	Auf Ereignisse horchen	896
13.9.5	Zugriff auf die gesamte Windows-Registry	897
13.10	Zum Weiterlesen	898

14 Einführung in die nebenläufige Programmierung 901

14.1	Nebenläufigkeit und Parallelität	901
14.1.1	Multitasking, Prozesse, Threads	902
14.1.2	Threads und Prozesse	902
14.1.3	Wie nebenläufige Programme die Geschwindigkeit steigern können	903
14.1.4	Was Java für Nebenläufigkeit alles bietet	905
14.2	Threads erzeugen	905
14.2.1	Threads über die Schnittstelle Runnable implementieren	905
14.2.2	Thread mit Runnable starten	907
14.2.3	Die Klasse Thread erweitern	908
14.3	Thread-Eigenschaften und -Zustände	911
14.3.1	Der Name eines Threads	911
14.3.2	Wer bin ich?	911
14.3.3	Schläfer gesucht	912
14.3.4	Mit yield() auf Rechenzeit verzichten	914
14.3.5	Der Thread als Dämon	914
14.3.6	Freiheit für den Thread – das Ende	916
14.3.7	Einen Thread höflich mit Interrupt beenden	917
14.3.8	UncaughtExceptionHandler für unbehandelte Ausnahmen	919

14.4	Der Ausführer (Executor) kommt	920
14.4.1	Die Schnittstelle Executor	921
14.4.2	Glücklich in der Gruppe – die Thread-Pools	922
14.4.3	Threads mit Rückgabe über Callable	924
14.4.4	Mehrere Callable abarbeiten	927
14.4.5	ScheduledExecutorService für wiederholende Ausgaben und Zeitsteuerungen nutzen	928
14.5	Synchronisation über kritische Abschnitte	929
14.5.1	Gemeinsam genutzte Daten	929
14.5.2	Probleme beim gemeinsamen Zugriff und kritische Abschnitte	929
14.5.3	Punkte nebenläufig initialisieren	931
14.5.4	Kritische Abschnitte schützen	933
14.5.5	Kritische Abschnitte mit ReentrantLock schützen	935
14.5.6	Synchronisieren mit synchronized	939
14.5.7	Mit synchronized synchronisierte Blöcke	940
14.5.8	Dann machen wir doch gleich alles synchronisiert!	942
14.5.9	Lock-Freigabe im Fall von Exceptions	942
14.5.10	Deadlocks	943
14.6	Zum Weiterlesen	945

15 Einführung in Datenstrukturen und Algorithmen 947

15.1	Listen	947
15.1.1	Erstes Listen-Beispiel	947
15.1.2	Auswahlkriterium ArrayList oder LinkedList	948
15.1.3	Die Schnittstelle List	949
15.1.4	ArrayList	955
15.1.5	LinkedList	955
15.1.6	Der Feld-Adapter Arrays.asList(...)	957
15.1.7	toArray(...) von Collection verstehen – die Gefahr einer Falle erkennen	959
15.1.8	Primitive Elemente in Datenstrukturen verwalten	961
15.2	Mengen (Sets)	962
15.2.1	Ein erstes Mengen-Beispiel	962
15.2.2	Methoden der Schnittstelle Set	964
15.2.3	HashSet	966
15.2.4	TreeSet – die sortierte Menge	966
15.2.5	Die Schnittstellen NavigableSet und SortedSet	968
15.2.6	LinkedHashSet	971

15.3	Assoziative Speicher	972
15.3.1	Die Klassen HashMap und TreeMap	972
15.3.2	Einfügen und Abfragen des Assoziativspeichers	975
15.3.3	Über die Bedeutung von equals(...) und hashCode() bei Elementen	981
15.3.4	Eigene Objekte hashen	982
15.3.5	LinkedHashMap und LRU-Implementierungen	983
15.3.6	IdentityHashMap	984
15.3.7	Das Problem veränderter Elemente	984
15.3.8	Aufzählungen und Ansichten des Assoziativspeichers	985
15.3.9	Die Properties-Klasse	989
15.4	Mit einem Iterator durch die Daten wandern	991
15.5	Algorithmen in Collections	991
15.5.1	Die Bedeutung von Ordnung mit Comparator und Comparable	992
15.5.2	Sortieren	993
15.5.3	Den größten und kleinsten Wert einer Collection finden	995
15.5.4	Nichtänderbare Datenstrukturen, immutable oder nur lesen?	998
15.5.5	Null Object Pattern und leere Sammlungen/Iteratoren zurückgeben	1002
15.5.6	Echte typsichere Container	1005
15.5.7	Mit der Halbierungssuche nach Elementen fahnden	1006
15.6	Zum Weiterlesen	1008
 16 Einführung in grafische Oberflächen		1009
<hr/>		
16.1	GUI-Frameworks	1009
16.1.1	Kommandozeile	1009
16.1.2	Grafische Benutzeroberfläche	1009
16.1.3	Abstract Window Toolkit (AWT)	1010
16.1.4	Java Foundation Classes und Swing	1010
16.1.5	JavaFX	1010
16.1.6	SWT (Standard Widget Toolkit) *	1012
16.2	Deklarative und programmierte Oberflächen	1013
16.2.1	GUI-Beschreibungen in JavaFX	1014
16.2.2	Deklarative GUI-Beschreibungen für Swing?	1014
16.3	GUI-Builder	1015
16.3.1	GUI-Builder für JavaFX	1015
16.3.2	GUI-Builder für Swing	1016
16.4	Aller Swing-Anfang – Fenster zur Welt	1016
16.4.1	Eine Uhr, bei der die Zeit nie vergeht	1016

16.4.2	Swing-Fenster mit <code>javax.swing.JFrame</code> darstellen	1017
16.4.3	Mit <code>add(...)</code> auf den Container	1018
16.4.4	Fenster schließbar machen – <code>setDefaultCloseOperation(int)</code>	1018
16.4.5	Sichtbarkeit des Fensters	1019
16.4.6	Größe und Position des Fensters verändern	1019
16.5	Beschriftungen (JLabel)	1020
16.5.1	Mehrzeiliger Text, HTML in der Darstellung	1023
16.6	Es tut sich was – Ereignisse beim AWT	1023
16.6.1	Die Ereignisquellen und Horcher (Listener) von Swing	1024
16.6.2	Listener implementieren	1025
16.6.3	Listener bei dem Ereignisauslöser anmelden/abmelden	1027
16.6.4	Adapterklassen nutzen	1028
16.6.5	Innere Mitgliedsklassen und innere anonyme Klassen	1030
16.6.6	Aufrufen der Listener im AWT-Event-Thread	1032
16.7	Schaltflächen	1033
16.7.1	Normale Schaltflächen (JButton)	1033
16.7.2	Der aufmerksame ActionListener	1035
16.7.3	Schaltflächen-Ereignisse vom Typ <code>ActionEvent</code>	1036
16.7.4	Basisklasse <code>AbstractButton</code>	1036
16.7.5	Wechselknopf (<code>JToggleButton</code>)	1038
16.8	Alles Auslegungssache – die Layoutmanager	1038
16.8.1	Übersicht über Layoutmanager	1038
16.8.2	Zuweisen eines Layoutmanagers	1039
16.8.3	Im Fluss mit <code>FlowLayout</code>	1040
16.8.4	<code>BoxLayout</code>	1042
16.8.5	Mit <code>BorderLayout</code> in alle Himmelsrichtungen	1042
16.8.6	Rasteranordnung mit <code>GridLayout</code>	1045
16.8.7	Weitere Layoutmanager	1047
16.9	Textkomponenten	1047
16.9.1	Text in einer Eingabezeile	1048
16.9.2	Die Oberklasse der Textkomponenten (<code>JTextComponent</code>)	1049
16.9.3	Geschützte Eingaben (<code>JPasswordField</code>)	1050
16.9.4	Validierende Eingabefelder (<code>JFormattedTextField</code>)	1051
16.9.5	Einfache mehrzeilige Textfelder (<code>JTextArea</code>)	1052
16.10	Grundlegendes zum Zeichnen	1055
16.10.1	Die <code>paint(Graphics)</code> -Methode für das AWT-Frame	1055
16.10.2	Die ereignisorientierte Programmierung ändert Fensterinhalte	1057
16.10.3	Zeichnen von Inhalten auf ein <code>JFrame</code>	1058

16.10.4	Auffordern zum Neuzeichnen mit repaint(...)	1060
16.10.5	Java 2D-API	1060
16.11	Zum Weiterlesen	1061

17 Einführung in Dateien und Datenströme 1063

17.1	Dateisysteme unter NIO.2	1063
17.1.1	java.io-Paket mit File-Klasse	1063
17.1.2	NIO.2 und java.nio-Paket	1064
17.2	Datei und Verzeichnis	1064
17.2.1	FileSystem und Path	1065
17.2.2	Die Utility-Klasse Files	1070
17.2.3	Dateien kopieren und verschieben	1072
17.2.4	Neue Dateien, Verzeichnisse, symbolische Verknüpfungen anlegen und löschen	1075
17.3	Dateien mit wahlfreiem Zugriff	1077
17.3.1	Ein RandomAccessFile zum Lesen und Schreiben öffnen	1077
17.3.2	Aus dem RandomAccessFile lesen	1078
17.3.3	Schreiben mit RandomAccessFile	1079
17.3.4	Die Länge des RandomAccessFile	1080
17.3.5	Hin und her in der Datei	1080
17.4	Stream-Klassen für Bytes und Zeichen	1081
17.4.1	Lesen aus Dateien und Schreiben in Dateien	1082
17.4.2	Byteorientierte Datenströme über Files beziehen	1082
17.4.3	Zeichenorientierte Datenströme über Files beziehen	1083
17.4.4	Funktion von OpenOption bei den Files.newXXX(...) -Methoden	1084
17.4.5	Ressourcen aus dem Klassenpfad und aus JAR-Archiven laden	1086
17.4.6	Die Schnittstellen Closeable, AutoCloseable und Flushable	1087
17.5	Basisklassen für die Ein-/Ausgabe	1089
17.5.1	Die abstrakten Basisklassen	1089
17.5.2	Übersicht über Ein-/Ausgabeklassen	1090
17.5.3	Die abstrakte Basisklasse OutputStream	1092
17.5.4	Die abstrakte Basisklasse InputStream	1093
17.5.5	Die abstrakte Basisklasse Writer	1094
17.5.6	Die abstrakte Basisklasse Reader	1095
17.6	Datenströme filtern und verkettten	1098
17.6.1	Streams als Filter verkettten (verschachteln)	1099

17.6.2	Gepufferte Ausgaben mit <code>BufferedWriter</code> und <code>BufferedOutputStream</code>	1101
17.6.3	Gepufferte Eingaben mit <code>BufferedReader/BufferedInputStream</code>	1103
17.7	Vermittler zwischen Byte-Streams und Unicode-Strömen	1104
17.7.1	Datenkonvertierung durch den <code>OutputStreamWriter</code>	1104
17.7.2	Automatische Konvertierungen mit dem <code>InputStreamReader</code>	1106
17.8	Zum Weiterlesen	1107
18	Einführung ins Datenbankmanagement mit JDBC	1109
<hr/>		
18.1	Relationale Datenbanken	1109
18.1.1	Das relationale Modell	1109
18.2	Datenbanken und Tools	1110
18.2.1	HSQldb	1110
18.2.2	Eclipse Data Tools Platform (DTP) zum Durchschauen von Datenbanken	1111
18.3	JDBC und Datenbanktreiber	1113
18.4	Eine Beispielabfrage	1114
18.4.1	Schritte zur Datenbankabfrage	1114
18.4.2	Ein Client für die HSQldb-Datenbank	1115
18.5	Zum Weiterlesen	1116
19	Einführung in <XML>	1117
<hr/>		
19.1	Auszeichnungssprachen	1117
19.1.1	Die Standard Generalized Markup Language (SGML)	1117
19.1.2	Extensible Markup Language (XML)	1118
19.2	Eigenschaften von XML-Dokumenten	1118
19.2.1	Elemente und Attribute	1118
19.2.2	Beschreibungssprache für den Aufbau von XML-Dokumenten	1121
19.2.3	Schema – die moderne Alternative zu DTD	1125
19.2.4	Namensraum (Namespace)	1128
19.2.5	XML-Applikationen *	1129
19.3	Die Java-APIs für XML	1129
19.3.1	Das Document Object Model (DOM)	1130
19.3.2	Simple API for XML Parsing (SAX)	1130
19.3.3	Pull-API StAX	1131
19.3.4	Java Document Object Model (JDOM)	1131

19.3.5	JAXP als Java-Schnittstelle zu XML	1131
19.3.6	DOM-Bäume einlesen mit JAXP *	1132
19.4	Java Architecture for XML Binding (JAXB)	1133
19.4.1	Bean für JAXB aufbauen	1133
19.4.2	Utility-Klasse JAXB	1134
19.4.3	Ganze Objektgraphen schreiben und lesen	1135
19.4.4	JAXBContext und Marshaller/Unmarshaller nutzen	1137
19.4.5	Validierung	1139
19.4.6	Weitere JAXB-Annotationen *	1143
19.4.7	Beans aus XML-Schema-Datei generieren	1150
19.5	XML-Dateien mit JDOM verarbeiten	1158
19.5.1	JDOM beziehen	1158
19.5.2	Paketübersicht *	1159
19.5.3	Die Document-Klasse	1160
19.5.4	Eingaben aus der Datei lesen	1161
19.5.5	Das Dokument im XML-Format ausgeben	1162
19.5.6	Der Dokumenttyp *	1163
19.5.7	Elemente	1164
19.5.8	Zugriff auf Elementinhalte	1167
19.5.9	Liste mit Unterelementen erzeugen *	1169
19.5.10	Neue Elemente einfügen und ändern	1170
19.5.11	Attributinhalt lesen und ändern	1173
19.5.12	XPath	1176
19.6	Zum Weiterlesen	1180
20	Testen mit JUnit	1181
<hr/>		
20.1	Softwaretests	1181
20.1.1	Vorgehen bei Schreiben von Testfällen	1182
20.2	Das Test-Framework JUnit	1182
20.2.1	Test-Driven-Development und Test-First	1182
20.2.2	Testen, implementieren, testen, implementieren, testen, freuen	1184
20.2.3	JUnit-Tests ausführen	1185
20.2.4	assertXXX(...)-Methoden der Klasse Assert	1186
20.2.5	Matcher-Objekte und Hamcrest	1187
20.2.6	Exceptions testen	1192
20.2.7	Tests ignorieren und Grenzen für Ausführungszeiten festlegen	1193
20.2.8	Mit Methoden der Assume-Klasse Tests abbrechen	1194

20.3	Wie gutes Design das Testen ermöglicht	1194
20.4	Aufbau größerer Testfälle	1197
20.4.1	Fixtures	1197
20.4.2	Sammlungen von Testklassen und Klassenorganisation	1199
20.5	Dummy, Fake, Stub und Mock	1200
20.6	JUnit-Erweiterungen, Testzusätze	1201
20.7	Zum Weiterlesen	1202

21 Bits und Bytes und Mathematisches 1203

21.1	Bits und Bytes *	1203
21.1.1	Die Bit-Operatoren Komplement, Und, Oder und XOR	1203
21.1.2	Repräsentation ganzer Zahlen in Java – das Zweierkomplement	1205
21.1.3	Das binäre (Basis 2), oktale (Basis 8), hexadezimale (Basis 16) Stellenwertsystem	1206
21.1.4	Auswirkung der Typanpassung auf die Bitmuster	1207
21.1.5	Vorzeichenlos arbeiten	1209
21.1.6	Die Verschiebeoperatoren	1212
21.1.7	Ein Bit setzen, löschen, umdrehen und testen	1214
21.1.8	Bit-Methoden der Integer- und Long-Klasse	1214
21.2	Fließkomma-Arithmetik in Java	1216
21.2.1	Spezialwerte für Unendlich, Null, NaN	1217
21.2.2	Standardnotation und wissenschaftliche Notation bei Fließkommazahlen *	1220
21.2.3	Mantisse und Exponent *	1220
21.3	Die Eigenschaften der Klasse Math	1222
21.3.1	Attribute	1222
21.3.2	Absolutwerte und Vorzeichen	1222
21.3.3	Maximum/Minimum	1223
21.3.4	Runden von Werten	1224
21.3.5	Rest der ganzzahligen Division *	1226
21.3.6	Division mit Rundung Richtung negativ unendlich, alternativer Restwert *	1227
21.3.7	Wurzel- und Exponentialmethoden	1229
21.3.8	Der Logarithmus *	1230
21.3.9	Winkelmethoden *	1231
21.3.10	Zufallszahlen	1232

21.4 Genauigkeit, Wertebereich eines Typs und Überlaufkontrolle *	1233
21.4.1 Der größte und der kleinste Wert	1233
21.4.2 Überlauf	1233
21.4.3 Was bitte macht ein <code>ulp</code> ?	1235
21.5 Zufallszahlen: <code>Random</code>, <code>SecureRandom</code>, <code>SplittableRandom</code>	1237
21.5.1 Die Klasse <code>Random</code>	1237
21.5.2 <code>Random</code> -Objekte mit dem Samen aufbauen	1237
21.5.3 Einzelne Zufallszahlen erzeugen	1238
21.5.4 Pseudo-Zufallszahlen in der Normalverteilung *	1239
21.5.5 Strom von Zufallszahlen generieren *	1239
21.5.6 Die Klasse <code>SecureRandom</code> *	1240
21.5.7 <code>SplittableRandom</code> *	1241
21.6 Große Zahlen *	1241
21.6.1 Die Klasse <code>BigInteger</code>	1242
21.6.2 Beispiel: Ganz lange Fakultäten mit <code>BigInteger</code>	1248
21.6.3 Große Fließkommazahlen mit <code>BigDecimal</code>	1250
21.6.4 Mit <code>MathContext</code> komfortabel die Rechengenauigkeit setzen	1252
21.7 Mathe bitte strikt *	1253
21.7.1 Strikte Fließkommaberechnungen mit <code>strictfp</code>	1254
21.7.2 Die Klassen <code>Math</code> und <code>StrictMath</code>	1254
21.8 Zum Weiterlesen	1255

22 Die Werkzeuge des JDK 1257

22.1 Java-Quellen übersetzen	1257
22.1.1 Java-Compiler vom JDK	1258
22.1.2 Alternative Compiler	1258
22.1.3 Native Compiler	1259
22.1.4 Java-Programme in ein natives ausführbares Programm einpacken	1259
22.2 Die Java-Laufzeitumgebung	1260
22.2.1 Schalter der JVM	1260
22.2.2 Der Unterschied zwischen <code>java.exe</code> und <code>javaw.exe</code>	1263
22.3 Mit RoboVM geht's für Java in das iOS-Land *	1263
22.4 Dokumentationskommentare mit Javadoc	1264
22.4.1 Einen Dokumentationskommentar setzen	1264
22.4.2 Mit dem Werkzeug <code>javadoc</code> eine Dokumentation erstellen	1266
22.4.3 HTML-Tags in Dokumentationskommentaren *	1267
22.4.4 Generierte Dateien	1267

22.4.5	Dokumentationskommentare im Überblick *	1268
22.4.6	Javadoc und Doclets *	1270
22.4.7	Veraltete (deprecated) Typen und Eigenschaften	1270
22.4.8	Javadoc-Überprüfung mit DocLint	1273
22.5	Das Archivformat JAR	1273
22.5.1	Das Dienstprogramm jar benutzen	1274
22.5.2	Das Manifest	1276
22.5.3	Applikationen in JAR-Archiven starten	1277
22.5.4	Pack200-Format *	1278
Anhang		
A	Java SE Paketübersicht	1281
Index		1293