

Auf einen Blick

1	Neues in Java 9	43
2	Die Klassenbibliothek	67
3	Fortgeschrittene String-Verarbeitung	101
4	Datenstrukturen und Algorithmen	161
5	Threads und nebenläufige Programmierung	339
6	Raum und Zeit	459
7	Dateien, Verzeichnisse und Dateizugriffe	531
8	Datenströme	597
9	Dateiformate	689
10	Die eXtensible Markup Language (XML) und JSON	717
11	Netzwerkprogrammierung	805
12	RESTful und SOAP-Web-Services	865
13	Verteilte Programmierung mit RMI	875
14	Typen, Reflection und Annotationen	895
15	Logging und Monitoring	963
16	Datenbankmanagement mit JDBC	987
17	Grafische Oberflächen mit Swing	1033
18	Grafikprogrammierung	1223
19	JavaFX	1281
20	Sicherheitskonzepte	1327
21	Dynamische Übersetzung, Skriptsprachen, JShell	1351
22	Java Native Interface (JNI)	1383
23	Dienstprogramme für die Java-Umgebung	1401

Inhalt

Vorwort	35
1 Neues in Java 9	43
1.1 Klassenlader (Class Loader) und Modul-/Klassenpfad	43
1.1.1 Klassenladen auf Abruf	43
1.1.2 Klassenlader bei der Arbeit zusehen	44
1.1.3 JMOD-Dateien und JAR-Dateien	45
1.1.4 Woher die kleinen Klassen kommen: die Suchorte und spezielle Klassenlader	46
1.1.5 Setzen des Modulpfades	47
1.2 Module entwickeln und einbinden	49
1.2.1 Wer sieht wen	49
1.2.2 Plattform-Module und JMOD-Beispiel	50
1.2.3 Verbotene Plattformeigenschaften nutzen, --add-exports	51
1.2.4 Plattformmodule einbinden, --add-modules und --add-opens	53
1.2.5 Projektabhängigkeiten in Eclipse	55
1.2.6 Benannte Module und module-info.java	57
1.2.7 Automatische Module	61
1.2.8 Unbenanntes Modul	62
1.2.9 Lesbarkeit und Zugreifbarkeit	62
1.2.10 Modul-Migration	63
1.3 Sprachänderungen in Java 9	64
1.4 Bibliotheksänderungen in Java 9	65
1.5 Änderungen in den Werkzeugen von Java 9	65
1.6 JDK 9-HotSpot-JVM-Änderungen	66
1.7 Zum Weiterlesen	66
2 Die Klassenbibliothek	67
2.1 Die Java-Klassenphilosophie	67
2.1.1 Modul, Paket, Typ	67
2.1.2 Übersicht über die Pakete der Standardbibliothek	70

2.2	Die Utility-Klassen System und Properties	74
2.2.1	Systemeigenschaften der Java-Umgebung	76
2.2.2	Zeilenumbruchzeichen, line.separator	77
2.2.3	Eigene Properties von der Konsole aus setzen *	78
2.2.4	Umgebungsvariablen des Betriebssystems *	80
2.2.5	Einfache Zeitmessung und Profiling *	81
2.3	Versionskennungen auslesen, aufbauen, parsen und vergleichen	84
2.4	Einfache Benutzereingaben	86
2.4.1	Grafischer Eingabedialog über JOptionPane	86
2.4.2	Geschützte Passwort-Eingaben mit der Klasse Console *	87
2.5	Benutzereinstellungen *	88
2.5.1	Benutzereinstellungen mit der Preferences-API	89
2.5.2	Einträge einfügen, auslesen und löschen	90
2.5.3	Auslesen der Daten und Schreiben in einem anderen Format	92
2.5.4	Auf Ereignisse horchen	92
2.5.5	Zugriff auf die gesamte Windows-Registry	94
2.6	Maven: Build-Management und Abhängigkeiten auflösen	95
2.6.1	Beispielprojekt in Eclipse mit Maven	95
2.6.2	Properties hinzunehmen	96
2.6.3	Dependency hinzunehmen	96
2.6.4	Lokales- und Remote-Repository	98
2.6.5	Lebenszyklus, Phasen und Maven-Plugins	98
2.6.6	Archetypes	98
2.7	Zum Weiterlesen	99
3	Fortgeschrittene String-Verarbeitung	101
3.1	Erweiterte Zeicheneigenschaften	101
3.1.1	isXXX(...) -Methoden	101
3.1.2	Unicode-Blöcke	102
3.1.3	Unicode-Skripte	103
3.2	Reguläre Ausdrücke	104
3.2.1	Pattern.matches(...) bzw. String#matches(...)	104
3.2.2	Die Klassen Pattern und Matcher	108
3.2.3	Finden und nicht matchen	113
3.2.4	Gruppen	114
3.2.5	Gierige und nicht gierige Operatoren *	115

3.2.6	Mit MatchResult alle Ergebnisse einsammeln *	116
3.2.7	Suchen und Ersetzen mit Mustern	118
3.2.8	Hangman Version 2	121
3.3	Zerlegen von Zeichenketten	122
3.3.1	Zerlegen von Zeichensequenzen über String oder Pattern	122
3.3.2	Mehr vom Scanner	123
3.3.3	Die Klasse StringTokenizer *	129
3.3.4	BreakIterator als Zeichen-, Wort-, Zeilen- und Satztrenner *	130
3.3.5	StreamTokenizer *	133
3.4	Zeichenkodierungen, XML/HTML-Entities, Base64 *	137
3.4.1	Unicode und 8-Bit-Abbildungen/Unicode Encoding	137
3.4.2	Kodierungen über die Klasse String vornehmen	137
3.4.3	Das Paket java.nio.charset und der Typ Charset	138
3.4.4	Konvertieren mit OutputStreamWriter/InputStreamReader-Klassen	139
3.4.5	XML/HTML-Entities ausmaskieren	140
3.4.6	Base64-Kodierung	141
3.5	Ausgaben formatieren	142
3.5.1	Die Formatter-Klasse *	142
3.5.2	Formatieren mit Masken *	145
3.5.3	Format-Klassen	146
3.5.4	Zahlen, Prozente und Währungen mit NumberFormat und DecimalFormat formatieren *	149
3.5.5	MessageFormat und Pluralbildung mit ChoiceFormat	152
3.6	Sprachabhängiges Vergleichen und Normalisierung *	154
3.6.1	Die Klasse Collator	154
3.6.2	Effiziente interne Speicherung für die Sortierung	157
3.6.3	Normalisierung	158
3.7	Phonetische Vergleiche *	159
3.8	Zum Weiterlesen	160
4	Datenstrukturen und Algorithmen	161
4.1	Datenstrukturen und die Collection-API	161
4.1.1	Designprinzip mit Schnittstellen, abstrakten und konkreten Klassen	162
4.1.2	Die Basisschnittstellen Collection und Map	162
4.1.3	Die Utility-Klassen Collections und Arrays	163
4.1.4	Das erste Programm mit Container-Klassen	163

4.1.5	Die Schnittstelle Collection und Kernkonzepte	164
4.1.6	Schnittstellen, die Collection erweitern, und Map	168
4.1.7	Konkrete Container-Klassen	170
4.1.8	Generische Datentypen in der Collection-API	172
4.1.9	Die Schnittstelle Iterable und das erweiterte for	173
4.2	Listen	173
4.2.1	Erstes Listen-Beispiel	174
4.2.2	Auswahlkriterium ArrayList oder LinkedList	175
4.2.3	Die Schnittstelle List	175
4.2.4	ArrayList	181
4.2.5	LinkedList	183
4.2.6	Der Array-Adapter Arrays.asList(...)	184
4.2.7	ListIterator *	186
4.2.8	toArray(...) von Collection verstehen – die Gefahr einer Falle erkennen	187
4.2.9	Primitive Elemente in Datenstrukturen verwalten	190
4.3	Mengen (Sets)	191
4.3.1	Ein erstes Mengen-Beispiel	191
4.3.2	Methoden der Schnittstelle Set	193
4.3.3	HashSet	195
4.3.4	TreeSet – die sortierte Menge	195
4.3.5	Die Schnittstellen NavigableSet und SortedSet	197
4.3.6	LinkedHashSet	200
4.4	Queues (Schlangen) und Deques	201
4.4.1	Queue-Klassen	201
4.4.2	Deque-Klassen	203
4.4.3	Blockierende Queues und Prioritätswarteschlangen	204
4.4.4	PriorityQueue	204
4.5	Stack (Kellerspeicher, Stapel)	209
4.5.1	Die Methoden von java.util.Stack	210
4.6	Assoziative Speicher	211
4.6.1	Die Klassen HashMap und TreeMap	212
4.6.2	Einfügen und Abfragen des Assoziativspeichers	214
4.6.3	Über die Bedeutung von equals(...) und hashCode() bei Elementen	222
4.6.4	Eigene Objekte hashen	222
4.6.5	LinkedHashMap und LRU-Implementierungen	224
4.6.6	IdentityHashMap	225
4.6.7	Das Problem veränderter Elemente	225
4.6.8	Aufzählungen und Ansichten des Assoziativspeichers	226
4.6.9	Die Arbeitsweise einer Hash-Tabelle *	229
4.6.10	Die Properties-Klasse	232

4.7	Immutable Datenstrukturen	235
4.7.1	Nichtänderbare Datenstrukturen, immutable oder nur Lesen?	236
4.7.2	Null Object Pattern und leere Sammlungen/Iteratoren zurückgeben	237
4.7.3	Immutable Datenstrukturen mit einem Element: Singletons	239
4.7.4	Collections.unmodifiableXXX(...)	240
4.7.5	Statische ofXXX(...)-Methoden zum Aufbau unveränderbarer Set-, List-, Map-Datenstrukturen	243
4.8	Mit einem Iterator durch die Daten wandern	245
4.8.1	Iterator-Schnittstelle	245
4.8.2	Der Iterator kann (eventuell auch) löschen	246
4.8.3	Operationen auf allen Elementen durchführen	247
4.8.4	Einen Zufallszahlen-Iterator schreiben	247
4.8.5	Iteratoren von Sammlungen, das erweiterte for und Iterable	248
4.8.6	Fail-Fast-Iterator und die ConcurrentModificationException	252
4.8.7	Die Schnittstelle Enumeration *	253
4.9	Algorithmen in Collections	255
4.9.1	Die Bedeutung von Ordnung mit Comparator und Comparable	255
4.9.2	Sortieren	256
4.9.3	Den größten und kleinsten Wert einer Collection finden	259
4.9.4	Echte typsichere Container	262
4.9.5	Mit der Halbierungssuche nach Elementen fahnden	263
4.9.6	Ersetzen, Kopieren, Füllen, Umdrehen, Rotieren *	264
4.9.7	Listen durchwürfeln *	266
4.9.8	Häufigkeit eines Elements *	267
4.9.9	nCopies(...) *	267
4.10	Datenstrukturen mit Änderungsmeldungen	268
4.10.1	Das Paket javafx.collections	268
4.10.2	Fabrikmethoden in FXCollections	269
4.10.3	Änderungen melden über InvalidationListener	271
4.10.4	Änderungen melden über XXXChangeListener	271
4.10.5	Change-Klassen	272
4.10.6	Weitere Hilfsmethoden einer ObservableList	275
4.10.7	Melden von Änderungen an Arrays	276
4.10.8	Transformierte FXCollections	276
4.10.9	Weitere statische Methoden in FXCollections	277
4.11	Stream-API	278
4.11.1	Stream erzeugen	281
4.11.2	Terminale Operationen	284
4.11.3	Intermediäre Operationen	295
4.11.4	Streams mit primitiven Werten	301

4.11.5	Stream-Beziehungen, AutoCloseable	312
4.11.6	Stream-Builder	314
4.11.7	Spliterator	315
4.11.8	Klasse StreamSupport	316
4.12	Spezielle threadsichere Datenstrukturen	317
4.12.1	Zu Beginn nur synchronisierte Datenstrukturen in Java 1.0	317
4.12.2	Nichtsynchronisierte Datenstrukturen in der Standard-Collection-API	317
4.12.3	Nebenläufiger Assoziativspeicher und die Schnittstelle ConcurrentMap	318
4.12.4	ConcurrentLinkedQueue	318
4.12.5	CopyOnWriteArrayList und CopyOnWriteArraySet	319
4.12.6	Wrapper zur Synchronisation	319
4.12.7	Blockierende Warteschlangen	320
4.12.8	ArrayBlockingQueue und LinkedBlockingQueue	321
4.12.9	PriorityBlockingQueue	323
4.12.10	Transfer-Warteschlangen – TransferQueue und LinkedTransferQueue	326
4.13	Google Guava (Google Collections Library)	327
4.13.1	Beispiel Multi-Set und Multi-Map	327
4.13.2	Datenstrukturen aus Guava	328
4.13.3	Utility-Klassen von Guava	330
4.13.4	Prädikate	331
4.13.5	Transformationen	332
4.14	Die Klasse BitSet für Bitmengen *	332
4.14.1	Ein BitSet anlegen	333
4.14.2	BitSet füllen und Zustände erfragen	334
4.14.3	Mengenorientierte Operationen	335
4.14.4	Weitere Methoden von BitSet	336
4.14.5	Primzahlen in einem BitSet verwalten	337
4.15	Zum Weiterlesen	338
5	Threads und nebenläufige Programmierung	339
5.1	Threads erzeugen	339
5.1.1	Threads über die Schnittstelle Runnable implementieren	339
5.1.2	Thread mit Runnable starten	340
5.1.3	Die Klasse Thread erweitern	342
5.2	Thread-Eigenschaften und Zustände	344
5.2.1	Der Name eines Threads	344
5.2.2	Wer bin ich?	345

5.2.3	Die Zustände eines Threads *	346
5.2.4	Schläfer gesucht	347
5.2.5	Mit yield() und onSpinWait() auf Rechenzeit verzichten	348
5.2.6	Der Thread als Dämon	349
5.2.7	Freiheit für den Thread – das Ende	351
5.2.8	Einen Thread höflich mit Interrupt beenden	352
5.2.9	UncaughtExceptionHandler für unbehandelte Ausnahmen	354
5.2.10	Der stop() von außen und die Rettung mit ThreadDeath *	355
5.2.11	Ein Rendezvous mit join(...) *	356
5.2.12	Arbeit niederlegen und wieder aufnehmen *	358
5.2.13	Priorität *	359
5.3	Der Ausführer (Executor) kommt	360
5.3.1	Die Schnittstelle Executor	361
5.3.2	Glücklich in der Gruppe – die Thread-Pools	362
5.3.3	Threads mit Rückgabe über Callable	364
5.3.4	Mehrere Callable-Objekte abarbeiten	368
5.3.5	ScheduledExecutorService für wiederholende Ausgaben und Zeitsteuerungen nutzen	369
5.3.6	Asynchrones Programmieren mit CompletableFuture (CompletionStage)	369
5.4	Synchronisation über kritische Abschnitte	372
5.4.1	Gemeinsam genutzte Daten	372
5.4.2	Probleme beim gemeinsamen Zugriff und kritische Abschnitte	372
5.4.3	Punkte nebenläufig initialisieren	374
5.4.4	i++ sieht atomar aus, ist es aber nicht *	376
5.4.5	Kritische Abschnitte schützen	377
5.4.6	Kritische Abschnitte mit ReentrantLock schützen	379
5.4.7	Synchronisieren mit synchronized	386
5.4.8	Synchronized-Methoden der Klasse StringBuffer *	387
5.4.9	Mit synchronized synchronisierte Blöcke	388
5.4.10	Dann machen wir doch gleich alles synchronisiert!	389
5.4.11	Lock-Freigabe im Fall von Exceptions	390
5.4.12	Deadlocks	391
5.4.13	Mit synchronized nachträglich synchronisieren *	393
5.4.14	Monitore sind reentrant – gut für die Geschwindigkeit *	394
5.4.15	Synchronisierte Methodenaufrufe zusammenfassen *	395
5.5	Synchronisation über Warten und Benachrichtigen	396
5.5.1	Die Schnittstelle Condition	397
5.5.2	It's Disco-Time *	400
5.5.3	Warten mit wait(...) und Aufwecken mit notify()/notifyAll() *	405
5.5.4	Falls der Lock fehlt – IllegalMonitorStateException *	406

5.6	Datensynchronisation durch besondere Concurrency-Klassen *	408
5.6.1	Semaphor	408
5.6.2	Barrier und Austausch	412
5.6.3	Stop-and-go mit Exchanger	413
5.7	Atomare Operationen und frische Werte mit volatile *	414
5.7.1	Der Modifizierer volatile bei Objekt-/Klassenvariablen	414
5.7.2	Das Paket java.util.concurrent.atomic	416
5.8	Teile und herrsche mit Fork und Join *	418
5.8.1	Algorithmendesign per »teile und herrsche«	418
5.8.2	Nebenläufiges Lösen von D&C-Algorithmen	420
5.8.3	Fork und Join	421
5.9	Mit dem Thread verbundene Variablen *	424
5.9.1	ThreadLocal	425
5.9.2	InheritableThreadLocal	427
5.9.3	ThreadLocalRandom als schneller nebenläufiger Zufallszahlengenerator	428
5.9.4	ThreadLocal bei der Performance-Optimierung	430
5.10	Threads in einer Thread-Gruppe *	430
5.10.1	Aktive Threads in der Umgebung	431
5.10.2	Etwas über die aktuelle Thread-Gruppe herausfinden	431
5.10.3	Threads in einer Thread-Gruppe anlegen	434
5.10.4	Methoden von Thread und ThreadGroup im Vergleich	437
5.11	Reaktive Programmierung und die Flow-API *	438
5.12	Zeitgesteuerte Abläufe	440
5.12.1	Die Typen Timer und TimerTask	440
5.13	Ausführen externer Programme, Starten von Prozessen *	442
5.13.1	ProcessBuilder und Prozesskontrolle mit Process	442
5.13.2	ProcessHandle und Prozess-IDs	447
5.13.3	Einen Browser, E-Mail-Client oder Editor aufrufen	449
5.14	Den Stack-Trace erfragen *	450
5.14.1	StackTraceElement	451
5.14.2	printStackTrace(...)	452
5.14.3	StackTraceElement vom Thread erfragen	452
5.14.4	StackWalker und Stack-Walking API	453
5.15	Einen Abbruch der virtuellen Maschine erkennen	455
5.15.1	Shutdown-Hook	455
5.15.2	Signale	456
5.16	Zum Weiterlesen	457

6	Raum und Zeit	459
6.1	Sprachen der Länder	459
6.1.1	Sprachen in Regionen über Locale-Objekte	459
6.2	Internationalisierung und Lokalisierung	463
6.2.1	ResourceBundle-Objekte und Ressource-Dateien	463
6.2.2	Ressource-Dateien zur Lokalisierung	464
6.2.3	Die Klasse ResourceBundle	465
6.2.4	Ladestrategie für ResourceBundle-Objekte	465
6.2.5	Ladeprozess und Format anpassen *	467
6.3	Weltzeit *	468
6.4	Wichtige Datum-Klassen im Überblick	469
6.4.1	Der 1.1.1970	470
6.4.2	System.currentTimeMillis()	470
6.4.3	Einfache Zeitumrechnungen durch TimeUnit	471
6.5	Die Klasse Date	472
6.5.1	Objekte erzeugen und Methoden nutzen	472
6.5.2	Date-Objekte sind nicht immutable	474
6.6	Date-Time-API	474
6.6.1	Menschenzeit und Maschinenzeit	476
6.6.2	Datumsklasse LocalDate	479
6.6.3	Ostertage *	480
6.6.4	Die Klasse YearMonth	481
6.6.5	Die Klasse MonthDay	482
6.6.6	Aufzählung DayOfWeek und Month	482
6.6.7	Klasse LocalTime	483
6.6.8	Klasse LocalDateTime	483
6.6.9	Klasse Year	484
6.6.10	Zeitzone-Klassen ZoneId und ZoneOffset	485
6.6.11	Temporale Klassen mit Zeitzoneinformationen	486
6.6.12	Klassen Period und Duration	489
6.6.13	Klasse Instant	492
6.6.14	Parse und Formatieren von temporalen Typen	493
6.6.15	Das Paket java.time.temporal *	496
6.6.16	Rock Around the Clock	502
6.6.17	Konvertierungen zwischen der klassischen API und der Date-Time-API	503
6.7	Calendar und GregorianCalendar	504
6.7.1	Die abstrakte Klasse Calendar	505
6.7.2	Calendar nach Date und Millisekunden fragen	506

6.7.3	Abfragen und Setzen von Datumselementen über Feldbezeichner	506
6.7.4	Kalender-Exemplare bauen über den Calendar.Builder	510
6.7.5	Wie viele Tage hat der Monat, oder wie viele Monate hat ein Jahr? *	511
6.7.6	Wann beginnt die Woche und wann die erste Woche im Jahr? *	512
6.7.7	Der gregorianische Kalender	513
6.8	Zeitzone in Java *	517
6.8.1	Zeitzone durch die Klasse TimeZone repräsentieren	517
6.8.2	SimpleTimeZone	518
6.8.3	Methoden von TimeZone	519
6.9	Formatieren und Parsen von Date-Objekten	520
6.9.1	Mit DateFormat und SimpleDateFormat formatieren	521
6.9.2	Parsen von Datumswerten	526
6.10	Die Default-Falle	528
6.11	Zum Weiterlesen	529
7	Dateien, Verzeichnisse und Dateizugriffe	531
7.1	Alte und neue Welt in java.io und java.nio	531
7.1.1	java.io-Paket mit File-Klasse	531
7.1.2	NIO.2 und java.nio-Paket	532
7.1.3	java.io.File oder java.nio.*?	532
7.2	Dateisysteme und Pfade	533
7.2.1	FileSystem und Path	533
7.2.2	Die Utility-Klasse Files	539
7.2.3	Dateien kopieren und verschieben	541
7.2.4	Dateiattribute *	543
7.2.5	Neue Dateien, Verzeichnisse, symbolische Verknüpfungen anlegen und löschen	552
7.2.6	MIME-Typen herausfinden *	554
7.2.7	Verzeichnislistings (DirectoryStream/Stream) und Filter *	556
7.2.8	Rekursives Ablaufen des Verzeichnisbaums *	558
7.2.9	Rekursiv nach Dateien/Ordern suchen mit Files.find(...) *	562
7.2.10	FileSystem: Abstraktion eines Dateisystems *	563
7.2.11	Verzeichnisse im Dateisystem überwachen *	566
7.3	Datei- und Verzeichnisoperationen mit der Klasse File	569
7.3.1	Dateien und Verzeichnisse mit der Klasse File	569
7.3.2	Existiert das Verzeichnis oder die Datei, und ist es ...?	573

7.3.3	Verzeichnis- und Dateieigenschaften/-attribute	573
7.3.4	Umbenennen und Verzeichnisse anlegen	576
7.3.5	Verzeichnisse auflisten und Dateien filtern	576
7.3.6	Dateien berühren, neue Dateien anlegen, temporäre Dateien	579
7.3.7	Dateien und Verzeichnisse löschen	580
7.3.8	Wurzelverzeichnis, Laufwerksnamen, Plattenspeicher *	581
7.3.9	URL-, URI- und Path-Objekte aus einem File-Objekt ableiten *	583
7.3.10	Mit Locking Dateien sperren *	584
7.3.11	Sicherheitsprüfung *	585
7.4	Dateien mit wahlfreiem Zugriff	585
7.4.1	Ein RandomAccessFile zum Lesen und Schreiben öffnen	586
7.4.2	Aus dem RandomAccessFile lesen	586
7.4.3	Schreiben mit RandomAccessFile	589
7.4.4	Die Länge des RandomAccessFile	590
7.4.5	Hin und her in der Datei	590
7.5	Wahlfreier Zugriff mit SeekableByteChannel und ByteBuffer *	591
7.5.1	SeekableByteChannel	591
7.5.2	ByteBuffer	592
7.5.3	Beispiel mit Path + SeekableByteChannel + ByteBuffer	592
7.5.4	FileChannel	593
7.6	Zum Weiterlesen	595
8	Datenströme	597
8.1	Basisklassen für die Ein-/Ausgabe	597
8.1.1	Die vier abstrakten Basisklassen	598
8.1.2	Die abstrakte Basisklasse OutputStream	598
8.1.3	Ein Datenschlucker *	601
8.1.4	Die abstrakte Basisklasse InputStream	601
8.1.5	Die abstrakte Basisklasse Writer	603
8.1.6	Die Schnittstelle Appendable *	605
8.1.7	Die abstrakte Basisklasse Reader	605
8.1.8	Die Schnittstellen Closeable, AutoCloseable und Flushable	608
8.2	Lesen aus Dateien und Schreiben in Dateien	610
8.2.1	Byteorientierte Datenströme über Files beziehen	610
8.2.2	Zeichenorientierte Datenströme über Files beziehen	611
8.2.3	Funktion von OpenOption bei den Files.newXXX(...) -Methoden	613
8.2.4	Ressourcen aus dem Modulpfad und aus JAR-Dateien laden	614

8.3 Übersicht über Ein-/Ausgabeklassen	616
8.3.1 Eingaben lesen	616
8.3.2 Ausgaben schreiben	617
8.3.3 Klassen gruppiert nach Ressourcen	617
8.3.4 Ströme mit <code>SequenceInputStream</code> zusammensetzen *	618
8.4 Formatierte Textausgaben	620
8.4.1 Die Klassen <code>PrintWriter</code> und <code>PrintStream</code>	620
8.4.2 <code>System.out</code> , <code>System.err</code> und <code>System.in</code>	625
8.5 Die FileXXX-Stromklassen	628
8.5.1 <code>FileOutputStream</code> und <code>FileInputStream</code>	629
8.5.2 Das <code>FileDescriptor</code> -Objekt *	632
8.5.3 Mit dem <code>FileWriter</code> Texte in Dateien schreiben	633
8.5.4 Zeichen mit der Klasse <code>FileReader</code> lesen	634
8.6 Schreiben und Lesen aus Strings und Byte-Feldern	635
8.6.1 Mit dem <code>StringWriter</code> ein String-Objekt füllen	636
8.6.2 <code>CharArrayWriter</code>	637
8.6.3 <code>StringReader</code> und <code>CharArrayReader</code>	637
8.6.4 Mit <code>ByteArrayOutputStream</code> in ein Byte-Feld schreiben	639
8.6.5 Mit <code>ByteArrayInputStream</code> aus einem Byte-Feld lesen	640
8.7 Datenströme filtern und verketteten	640
8.7.1 Streams als Filter verketteten (verschachteln)	641
8.7.2 Gepufferte Ausgaben mit <code>BufferedWriter</code> und <code>BufferedOutputStream</code>	642
8.7.3 Gepufferte Eingaben mit <code>BufferedReader/BufferedInputStream</code>	644
8.7.4 <code>LineNumberReader</code> zählt automatisch Zeilen mit *	645
8.7.5 Daten mit der Klasse <code>PushbackReader</code> zurücklegen *	646
8.7.6 <code>DataOutputStream/DataInputStream</code> *	649
8.7.7 Basisklassen für Filter *	650
8.7.8 Die Basisklasse <code>FilterWriter</code> *	651
8.7.9 Eingaben mit der Klasse <code>FilterReader</code> filtern *	652
8.8 Vermittler zwischen Byte-Streams und Unicode-Strömen	653
8.8.1 Datenkonvertierung durch den <code>OutputStreamWriter</code>	653
8.8.2 Automatische Konvertierungen mit dem <code>InputStreamReader</code>	655
8.9 Kommunikation zwischen Threads mit Pipes *	657
8.9.1 <code>PipedOutputStream</code> und <code>PipedInputStream</code>	657
8.9.2 <code>PipedWriter</code> und <code>PipedReader</code>	659
8.10 Prüfsummen	660
8.10.1 Die Schnittstelle <code>Checksum</code>	661
8.10.2 Die Klasse <code>CRC32</code>	662
8.10.3 Die <code>Adler32</code> -Klasse	664

8.11 Persistente Objekte und Serialisierung	664
8.11.1 Objekte mit der Standardserialisierung speichern und lesen	665
8.11.2 Zwei einfache Anwendungen der Serialisierung *	668
8.11.3 Die Schnittstelle <code>Serializable</code>	669
8.11.4 Nicht serialisierbare Attribute aussparen	671
8.11.5 Deserialisierung absichern mit einem <code>ObjectInputFilter</code> *	673
8.11.6 Das Abspeichern selbst in die Hand nehmen	676
8.11.7 Tiefe Objektkopien *	679
8.11.8 Versionenverwaltung und die SUID	681
8.11.9 Wie die <code>ArrayList</code> serialisiert *	683
8.11.10 Probleme mit der Serialisierung	684
8.12 Alternative Datenaustauschformate	685
8.12.1 Serialisieren in XML-Dateien	685
8.12.2 XML-Serialisierung von JavaBeans mit <code>JavaBeans Persistence</code> *	685
8.12.3 Die Open-Source-Bibliothek <code>XStream</code> *	687
8.12.4 Binäre Serialisierung mit <code>Google Protocol Buffers</code> *	688
8.13 Zum Weiterlesen	688
9 Dateiformate	689
9.1 Einfache Dateiformate für strukturierte Daten	690
9.1.1 Property-Dateien mit <code>java.util.Properties</code> lesen und schreiben	690
9.1.2 CSV-Dateien	692
9.2 Dokumentenformate	694
9.2.1 (X)HTML	694
9.2.2 PDF-Dokumente	695
9.2.3 Microsoft-Office-Dokumente	696
9.2.4 OASIS Open Document Format	697
9.3 Datenkompression *	697
9.3.1 Java-Unterstützung beim Komprimieren	698
9.3.2 Daten packen und entpacken	699
9.3.3 Datenströme komprimieren	701
9.3.4 ZIP-Archive	704
9.3.5 JAR-Archive	711
9.4 Bildformate	711
9.5 Audiodateien	711
9.5.1 Die Arbeit mit <code>AudioSystem</code> und <code>Clip</code>	711
9.5.2 <code>AudioClip</code> von <code>JavaFX</code>	712

9.5.3	MIDI-Dateien abspielen	712
9.5.4	ID-Tags aus MP3-Dateien	714
9.6	Zum Weiterlesen	715
10	Die eXtensible Markup Language (XML) und JSON	717
10.1	Auszeichnungssprachen	717
10.1.1	Die Standard Generalized Markup Language (SGML)	717
10.1.2	Extensible Markup Language (XML)	718
10.2	Eigenschaften von XML-Dokumenten	718
10.2.1	Elemente und Attribute	718
10.2.2	Beschreibungssprache für den Aufbau von XML-Dokumenten	721
10.2.3	Schema – die moderne Alternative zu DTD	725
10.2.4	Namensraum (Namespace)	727
10.2.5	XML-Applikationen *	729
10.3	Die Java-APIs für XML	729
10.3.1	Das Document Object Model (DOM)	730
10.3.2	Simple API for XML Parsing (SAX)	730
10.3.3	Pull-API StAX	730
10.3.4	Java Document Object Model (JDOM)	731
10.3.5	JAXP als Java-Schnittstelle zu XML	731
10.3.6	DOM-Bäume einlesen mit JAXP *	732
10.4	Java Architecture for XML Binding (JAXB)	732
10.4.1	Bean für JAXB aufbauen	733
10.4.2	Utility-Klasse JAXB	734
10.4.3	Ganze Objektgraphen schreiben und lesen	735
10.4.4	JAXBContext und Marshaller/Unmarshaller nutzen	736
10.4.5	Validierung	739
10.4.6	Weitere JAXB-Annotationen *	742
10.4.7	JAXB-Beans aus XML-Schema-Datei generieren	749
10.5	Serielle Verarbeitung mit StAX	755
10.5.1	Unterschiede der Verarbeitungsmodelle	756
10.5.2	XML-Dateien mit dem Cursor-Verfahren lesen	757
10.5.3	XML-Dateien mit dem Iterator-Verfahren verarbeiten *	761
10.5.4	Mit Filtern arbeiten *	763
10.5.5	XML-Dokumente schreiben	764
10.6	Serielle Verarbeitung von XML mit SAX *	768

10.6.1	Schnittstellen von SAX	768
10.6.2	SAX-Parser erzeugen	769
10.6.3	Operationen der Schnittstelle ContentHandler	769
10.6.4	ErrorHandler und EntityResolver	772
10.7	XML-Dateien mit JDOM verarbeiten	772
10.7.1	JDOM beziehen	773
10.7.2	Paketübersicht *	774
10.7.3	Die Document-Klasse	775
10.7.4	Eingaben aus der Datei lesen	776
10.7.5	Das Dokument im XML-Format ausgeben	777
10.7.6	Der Dokumenttyp *	778
10.7.7	Elemente	779
10.7.8	Zugriff auf Elementinhalte	781
10.7.9	Liste mit Unterelementen erfragen *	784
10.7.10	Neue Elemente einfügen und ändern	784
10.7.11	Attributinhalt lesen und ändern	787
10.7.12	XPath	790
10.8	Transformationen mit XSLT *	794
10.8.1	Templates und XPath als Kernelemente von XSLT	794
10.8.2	Umwandlung von XML-Dateien mit JDOM und JAXP	797
10.9	XML-Schema-Validierung *	797
10.9.1	SchemaFactory und Schema	798
10.9.2	Validator	798
10.9.3	Validierung unterschiedlicher Datenquellen durchführen	799
10.10	JSON-Serialisierung mit Jackson-Datenformat	799
10.10.1	JSON im Kontext von JavaScript	799
10.10.2	JSON mit JavaScript-Engine von Java	800
10.10.3	JSON-Verarbeitung mit der Java API for JSON Processing	801
10.11	Zum Weiterlesen	803
11	Netzwerkprogrammierung	805
11.1	Grundlegende Begriffe	805
11.2	URI und URL	807
11.2.1	Die Klasse URI	807
11.2.2	Die Klasse URL	807
11.2.3	Informationen über eine URL *	809
11.2.4	Der Zugriff auf die Daten über die Klasse URL	811

11.3	URL-Parameter kodieren	812
11.4	Die Klassen URLConnection und HttpURLConnection	813
11.4.1	Methoden und Anwendung von URLConnection	815
11.4.2	HttpURLConnection	816
11.4.3	Webseiten aufrufen, mit GET und POST Daten übergeben *	817
11.4.4	POST-Request absenden	819
11.4.5	Der Protokoll-Handler für JAR-Dateien	820
11.4.6	Basic Authentication und Proxy-Authentifizierung	821
11.5	Host- und IP-Adressen	822
11.5.1	Lebt der Rechner?	824
11.5.2	IP-Adresse des lokalen Hosts	825
11.5.3	Das Netz ist klasse *	826
11.5.4	NetworkInterface	826
11.6	Socket-Verbindungen	827
11.6.1	Das Netzwerk ist der Computer	827
11.6.2	Sockets	828
11.6.3	Eine Verbindung zum Server aufbauen	829
11.6.4	Sockets unter Spannung – die Ströme	830
11.6.5	Die Verbindung wieder abbauen	830
11.6.6	Zeitserver ansprechen	831
11.6.7	Informationen über den Socket *	832
11.6.8	Reine Verbindungsdaten über SocketAddress *	834
11.6.9	Die Serverseite mit ServerSocket vorbereiten	835
11.6.10	Ein Multiplikationsserver	836
11.6.11	Blockierendes Lesen	839
11.7	HTTP Client API in Java 9	841
11.8	Neue externe Netzwerkbibliotheken *	843
11.8.1	HttpComponents	843
11.8.2	Async Http Client	844
11.8.3	Apache Commons Net	844
11.9	Arbeitsweise eines Webservers *	844
11.9.1	Das Hypertext Transfer Protocol (HTTP)	845
11.9.2	Anfragen an den Server	845
11.9.3	Die Antworten vom Server	847
11.9.4	Webserver mit com.sun.net.httpserver.HttpServer	851
11.10	Verbindungen durch einen Proxy-Server *	853
11.10.1	System-Properties	853
11.10.2	Verbindungen durch die Proxy-API	854

11.11	Bidirektionale binäre Webkommunikation über WebSockets *	855
11.11.1	WebSocket-Standard	855
11.11.2	WebSocket-Implementierungen	856
11.12	Datagram-Sockets *	856
11.12.1	Die Klasse DatagramSocket	858
11.12.2	Datagramme und die Klasse DatagramPacket	859
11.12.3	Auf ein hereinkommendes Paket warten	859
11.12.4	Ein Paket zum Senden vorbereiten	860
11.12.5	Methoden der Klasse DatagramPacket	861
11.12.6	Das Paket senden	861
11.13	Tiefer liegende Netzwerkeigenschaften *	862
11.13.1	MAC-Adressen auslesen	863
11.13.2	Internet Control Message Protocol (ICMP)	863
11.14	Zum Weiterlesen	863
12	RESTful und SOAP-Web-Services	865
12.1	Web-Services	865
12.2	RESTful Web-Services	866
12.2.1	Aus Prinzip REST	867
12.2.2	Jersey	868
12.3	Daily Soap und das SOAP-Protokoll	868
12.3.1	Die technische Realisierung	869
12.3.2	Web-Service-APIs und Implementierungen	869
12.3.3	@WebService	870
12.3.4	Web-Service-Modul angeben	870
12.3.5	Einen Web-Service definieren	870
12.3.6	Web-Services veröffentlichen	872
12.3.7	Einen JAX-WS-Client implementieren	872
12.4	Zum Weiterlesen	874
13	Verteilte Programmierung mit RMI	875
13.1	Entfernte Objekte und Methoden	875
13.1.1	Stellvertreter helfen bei entfernten Methodenaufrufen	875
13.1.2	Standards für entfernte Objekte	877

13.2 Java Remote Method Invocation	877
13.2.1 Zusammenspiel von Server, Registry und Client	877
13.2.2 Wie die Stellvertreter die Daten übertragen	878
13.2.3 Probleme mit entfernten Methoden	878
13.2.4 Nutzen von RMI bei Middleware-Lösungen	880
13.2.5 Zentrale Klassen und Schnittstellen	880
13.2.6 Entfernte und lokale Objekte im Vergleich	881
13.3 Auf der Serverseite	881
13.3.1 Entfernte Schnittstelle deklarieren	882
13.3.2 Remote-Objekt-Implementierung	882
13.3.3 Stellvertreterobjekte	883
13.3.4 Der Namensdienst (Registry)	883
13.3.5 Remote-Objekt-Implementierung exportieren und beim Namensdienst anmelden	885
13.3.6 Einfaches Logging	888
13.3.7 Aufräumen mit dem DGC *	889
13.4 Auf der Client-Seite	889
13.5 Entfernte Objekte übergeben und laden	890
13.5.1 Klassen vom RMI-Klassenlader nachladen	891
13.6 Automatische Remote-Objekt-Aktivierung	891
13.7 Java Message Service (JMS)	892
13.8 Zum Weiterlesen	892
14 Typen, Reflection und Annotationen	895
14.1 Metadaten	895
14.1.1 Metadaten durch Javadoc-Tags	895
14.2 Die Klasse Class	896
14.2.1 An ein Class-Objekt kommen	896
14.2.2 Eine Class ist ein Type	899
14.3 Klassenlader	899
14.3.1 Die Klasse java.lang.ClassLoader	900
14.3.2 Hot Deployment mit dem URL-Classloader *	901
14.4 Metadaten der Typen mit dem Class-Objekt	905
14.4.1 Der Name des Typs	905
14.4.2 Was das Class-Objekt beschreibt *	908
14.4.3 instanceof mit Class-Objekten *	910

14.4.4 Oberklassen finden *	911
14.4.5 Implementierte Interfaces einer Klasse oder eines Interfaces *	912
14.4.6 Modifizierer und die Klasse Modifier *	912
14.4.7 Die Arbeit auf dem Feld *	914
14.5 Attribute, Methoden und Konstruktoren	915
14.5.1 Reflections – Gespür für die Attribute einer Klasse	917
14.5.2 Schnittstelle Member für Eigenschaften	918
14.5.3 Field-Klasse	919
14.5.4 Methoden einer Klasse erfragen	920
14.5.5 Properties einer Bean erfragen	923
14.5.6 Konstruktoren einer Klasse	924
14.5.7 Annotationen	925
14.5.8 Reflexionen über Module	925
14.6 Objekte erzeugen und manipulieren	927
14.6.1 Objekte erzeugen	927
14.6.2 Die Belegung der Variablen erfragen	928
14.6.3 Eine generische eigene toString()-Methode *	930
14.6.4 Variablen setzen	932
14.6.5 Bean-Zustände kopieren *	934
14.6.6 Private Attribute auslesen/ändern und der Typ AccessibleObject	934
14.6.7 Methoden aufrufen	936
14.6.8 Statische Methoden aufrufen	937
14.6.9 Dynamische Methodenaufrufe bei festen Methoden beschleunigen *	938
14.6.10 java.lang.reflect.Parameter	940
14.7 Schneller aufrufen mit MethodType und MethodHandle	941
14.8 Eigene Annotationstypen *	942
14.8.1 Annotationen zum Laden von Ressourcen	943
14.8.2 Neue Annotationen deklarieren	943
14.8.3 Annotationen mit genau einem Attribut	944
14.8.4 Element-Wert-Paare (Attribute) hinzufügen	945
14.8.5 Annotationsattribute vom Typ einer Aufzählung	946
14.8.6 Felder von Annotationsattributen	946
14.8.7 Vorbelegte Attribute	948
14.8.8 Annotieren von Annotationstypen	949
14.8.9 Deklarationen für unsere Ressourcen-Annotationen	954
14.8.10 Annotierte Elemente auslesen	956
14.8.11 Auf die Annotationsattribute zugreifen	958
14.8.12 Komplettbeispiel zum Initialisieren von Ressourcen	959
14.8.13 Mögliche Nachteile von Annotationen	961
14.9 Zum Weiterlesen	962

15	Logging und Monitoring	963
15.1	Logging mit Java	963
15.1.1	Logging-APIs	963
15.1.2	Logging mit java.util.logging	964
15.1.3	Die Simple Logging Facade (SLF4J)	970
15.1.4	Logging mit log4j 2 *	972
15.1.5	Aktuelle Entwicklungen der Java-Logging-APIs	975
15.1.6	System-Logging *	975
15.2	Systemzustände überwachen	976
15.3	MBean-Typen, MBean-Server und weitere Begriffe	976
15.3.1	MBeans des Systems	977
15.4	Geschwätzige Programme und JConsole	979
15.4.1	JConsole	979
15.5	Der MBeanServer	981
15.6	Eine eigene Standard-MBean	982
15.6.1	Management-Schnittstelle	982
15.6.2	Implementierung der Managed Ressource	982
15.6.3	Anmeldung beim Server	983
15.6.4	Eine eigene Bean in JConsole einbringen	983
15.7	Zum Weiterlesen	986
16	Datenbankmanagement mit JDBC	987
16.1	Relationale Datenbanken und Datenbankmanagementsysteme	987
16.1.1	Das relationale Modell	987
16.1.2	Datenbanken und Tools	988
16.1.3	HSQLDB	988
16.1.4	Weitere Datenbanken *	990
16.1.5	Eclipse Data Tools Platform (DTP) zum Durchschauen von Datenbanken	991
16.2	JDBC und Datenbanktreiber	993
16.2.1	JDBC-Versionen *	994
16.3	Eine Beispielabfrage	995
16.3.1	Schritte zur Datenbankabfrage	995
16.3.2	Ein Client für die HSQLDB-Datenbank	995

16.4	Mit Java an eine Datenbank andocken	996
16.4.1	Der Treiber-Manager *	997
16.4.2	Den Treiber laden	997
16.4.3	Eine Aufzählung aller Treiber *	999
16.4.4	Log-Informationen *	999
16.4.5	Verbindung zur Datenbank auf- und abbauen	1000
16.5	Datenbankabfragen	1003
16.5.1	Abfragen über das Statement-Objekt	1003
16.5.2	Ergebnisse einer Abfrage in ResultSet	1005
16.5.3	Java und SQL-Datentypen	1007
16.5.4	Date, Time und Timestamp	1009
16.5.5	Unicode in der Spalte korrekt auslesen	1011
16.5.6	Eine SQL-NULL und wasNull() bei ResultSet	1012
16.5.7	Wie viele Zeilen hat ein ResultSet? *	1012
16.6	Elemente einer Datenbank ändern	1013
16.6.1	Einzelne INSERT-, UPDATE- oder DELETE-Anweisungen senden	1013
16.6.2	Aktualisierbares ResultSet	1014
16.6.3	Batch-Updates	1015
16.7	Die Ausnahmen bei JDBC, SQLException und Unterklassen	1016
16.7.1	JDBC-Fehlerbasisklasse SQLException	1016
16.7.2	SQLWarning	1017
16.8	ResultSet und RowSet *	1019
16.8.1	Die Schnittstelle RowSet	1019
16.8.2	Implementierungen von RowSet	1019
16.8.3	Der Typ CachedRowSet	1020
16.8.4	Der Typ WebRowSet	1021
16.9	Vorbereitete Anweisungen (Prepared Statements)	1023
16.9.1	PreparedStatement-Objekte vorbereiten	1024
16.9.2	Werte für die PreparedStatement-Platzhalter	1025
16.10	Transaktionen	1026
16.11	Vorbereitete Datenbankverbindungen	1027
16.11.1	DataSource	1027
16.11.2	Gepoolte Datenbankverbindungen	1030
16.12	Zum Weiterlesen	1031

17 Grafische Oberflächen mit Swing	1033
17.1 AWT, JavaFoundation Classes und Swing	1033
17.1.1 Das Abstract Window Toolkit (AWT)	1033
17.1.2 Java Foundation Classes (JFC)	1034
17.1.3 Was Swing von AWT-Komponenten unterscheidet	1037
17.2 Mit NetBeans zur ersten Swing-Oberfläche	1038
17.2.1 Projekt anlegen	1038
17.2.2 Eine GUI-Klasse hinzufügen	1039
17.2.3 Programm starten	1041
17.2.4 Grafische Oberfläche aufbauen	1042
17.2.5 Swing-Komponenten-Klassen	1044
17.2.6 Funktionalität geben	1046
17.3 Aller Swing-Anfang – Fenster zur Welt	1049
17.3.1 Eine Uhr, bei der die Zeit nie vergeht	1050
17.3.2 Swing-Fenster mit javax.swing.JFrame darstellen	1051
17.3.3 Mit add(...) auf den Container	1051
17.3.4 Fenster schließbar machen – setDefaultCloseOperation(int)	1051
17.3.5 Sichtbarkeit des Fensters	1052
17.3.6 Größe und Position des Fensters verändern	1053
17.3.7 Fenster- und Dialogdekoration, Transparenz *	1054
17.3.8 Die Klasse Toolkit *	1055
17.3.9 Zum Vergleich: AWT-Fenster darstellen *	1055
17.4 Beschriftungen (JLabel)	1057
17.4.1 Mehrzeiliger Text, HTML in der Darstellung	1059
17.5 Icon und ImageIcon für Bilder auf Swing-Komponenten	1060
17.5.1 Die Klasse ImageIcon	1060
17.6 Es tut sich was – Ereignisse beim AWT	1062
17.6.1 Die Ereignisquellen und Horcher (Listener) von Swing	1063
17.6.2 Listener implementieren	1064
17.6.3 Listener bei dem Ereignisauslöser anmelden/abmelden	1066
17.6.4 Adapterklassen nutzen	1068
17.6.5 Listener-Code in inneren Klassen und Lambda-Ausdrücken	1070
17.6.6 Aufrufen der Listener im AWT-Event-Thread	1072
17.6.7 Ereignisse, etwas genauer betrachtet *	1072
17.7 Schaltflächen	1076
17.7.1 Normale Schaltflächen (JButton)	1076
17.7.2 Der aufmerksame ActionListener	1078

17.7.3 Schaltflächen-Ereignisse vom Typ ActionEvent	1079
17.7.4 Basisklasse AbstractButton	1079
17.7.5 Wechselknopf (JToggleButton)	1081
17.8 Textkomponenten	1081
17.8.1 Text in einer Eingabezeile	1082
17.8.2 Die Oberklasse der Textkomponenten (JTextComponent)	1083
17.8.3 Geschützte Eingaben (JPasswordField)	1085
17.8.4 Validierende Eingabefelder (JFormattedTextField)	1086
17.8.5 Einfache mehrzeilige Textfelder (JTextArea)	1087
17.8.6 Editor-Klasse (JEditorPane) *	1090
17.9 Swing Action *	1093
17.10 JComponent und Component als Basis aller Komponenten	1095
17.10.1 Hinzufügen von Komponenten	1095
17.10.2 Tooltips (Kurzhinweise)	1096
17.10.3 Rahmen (Border) *	1096
17.10.4 Fokus und Navigation *	1099
17.10.5 Ereignisse jeder Komponente *	1100
17.10.6 Die Größe und Position einer Komponente *	1103
17.10.7 Komponenten-Ereignisse *	1104
17.10.8 UI-Delegate – der wahre Zeichner *	1105
17.10.9 Undurchsichtige (opake) Komponente *	1108
17.10.10 Properties und Listener für Änderungen *	1108
17.11 Container	1108
17.11.1 Standard-Container (JPanel)	1109
17.11.2 Bereich mit automatischen Rollbalken (JScrollPane)	1109
17.11.3 Reiter (JTabbedPane)	1110
17.11.4 Teilungskomponente (JSplitPane)	1111
17.12 Alles Auslegungssache – die Layoutmanager	1112
17.12.1 Übersicht über Layoutmanager	1112
17.12.2 Zuweisen eines Layoutmanagers	1113
17.12.3 Im Fluss mit FlowLayout	1114
17.12.4 BorderLayout	1116
17.12.5 Mit BorderLayout in alle Himmelsrichtungen	1116
17.12.6 Rasteranordnung mit GridLayout	1119
17.12.7 Der GridBagLayoutmanager *	1121
17.12.8 Null-Layout *	1126
17.13 Rollbalken und Schieberegler	1126
17.13.1 Schieberegler (JSlider)	1126
17.13.2 Rollbalken (JScrollBar) *	1128

17.14 Kontrollfelder, Optionsfelder, Kontrollfeldgruppen	1131
17.14.1 Kontrollfelder (JCheckBox)	1132
17.14.2 ItemSelectable, ItemListener und das ItemEvent	1134
17.14.3 Sich gegenseitig ausschließende Optionen (JRadioButton)	1136
17.15 Fortschritte bei Operationen überwachen *	1137
17.15.1 Fortschrittsbalken (JProgressBar)	1137
17.15.2 Dialog mit Fortschrittsanzeige (ProgressMonitor)	1139
17.16 Menüs und Symbolleisten	1139
17.16.1 Die Menüleisten und die Einträge	1139
17.16.2 Menüeinträge definieren	1141
17.16.3 Einträge durch Action-Objekte beschreiben	1142
17.16.4 Mit der Tastatur – Mnemonics und Shortcut	1143
17.16.5 Der Tastatur-Shortcut (Accelerator)	1144
17.16.6 Tastenkürzel (Mnemonics)	1146
17.16.7 Symbolleisten alias Toolbars	1146
17.16.8 Popup-Menüs	1150
17.16.9 System-Tray nutzen *	1154
17.17 Das Model-View-Controller-Konzept	1155
17.18 Auswahlmenüs, Listen und Spinner	1157
17.18.1 Listen (JList)	1157
17.18.2 Auswahlmenü (JComboBox)	1164
17.18.3 Drehfeld (JSpinner) *	1170
17.18.4 Datumsauswahl	1171
17.19 Tabellen (JTable)	1172
17.19.1 Ein eigenes Tabellen-Model	1173
17.19.2 Basisklasse für eigene Modelle (AbstractTableModel)	1174
17.19.3 Ein vorgefertigtes Standardmodell (DefaultTableModel)	1178
17.19.4 Ein eigener Renderer für Tabellen	1179
17.19.5 Zell-Editoren	1182
17.19.6 Automatisches Sortieren und Filtern mit RowSorter *	1183
17.20 Bäume (JTree)	1187
17.20.1 JTree und sein TreeModel und TreeNode	1187
17.20.2 Selektionen bemerken	1189
17.20.3 Das TreeModel von JTree *	1189
17.21 Dialoge und Window-Objekte	1192
17.21.1 JWindow und JDialog	1192
17.21.2 Modal oder nichtmodal?	1193
17.21.3 Standarddialoge mit JOptionPane	1193
17.21.4 Der Dateiauswahldialog	1196

17.22 Flexibles Java-Look-and-Feel	1201
17.22.1 UIManager	1201
17.23 Swing-Komponenten neu erstellen oder verändern *	1203
17.23.1 Überlagerungen mit dem Swing-Komponenten-Dekorator JLayer	1204
17.24 Die Zwischenablage (Clipboard)	1206
17.24.1 Clipboard-Objekte	1206
17.24.2 Mit Transferable auf den Inhalt zugreifen	1206
17.24.3 DataFlavor ist das Format der Daten in der Zwischenablage	1207
17.24.4 Einfügungen in der Zwischenablage erkennen	1209
17.24.5 Drag & Drop	1210
17.25 AWT, Swing und die Threads	1211
17.25.1 Ereignisschlange (EventQueue) und AWT-Event-Thread	1211
17.25.2 Swing ist nicht threadsicher	1212
17.25.3 invokeLater(...) und invokeAndWait(...)	1214
17.25.4 SwingWorker	1216
17.25.5 Eigene Ereignisse in die Queue setzen *	1218
17.25.6 Auf alle Ereignisse hören *	1219
17.26 Barrierefreiheit mit der Java Accessibility API	1219
17.27 Zeitliches Ausführen mit dem javax.swing.Timer	1220
17.28 Zum Weiterlesen	1221
18 Grafikprogrammierung	1223
18.1 Grundlegendes zum Zeichnen	1223
18.1.1 Die paint(Graphics)-Methode für das AWT-Frame	1223
18.1.2 Die ereignisorientierte Programmierung ändert Fensterinhalte	1225
18.1.3 Zeichnen von Inhalten auf ein JFrame	1226
18.1.4 Auffordern zum Neuzeichnen mit repaint(...)	1228
18.1.5 Java 2D-API	1228
18.2 Einfache Zeichenmethoden	1229
18.2.1 Linien	1229
18.2.2 Rechtecke	1230
18.2.3 Ovale und Kreisbögen	1231
18.2.4 Polygone und Polylines	1232
18.3 Zeichenketten schreiben und Fonts	1233
18.3.1 Zeichenfolgen schreiben	1233

18.3.2	Die Font-Klasse	1234
18.3.3	Font-Metadaten durch FontMetrics *	1235
18.4	Geometrische Objekte	1239
18.4.1	Die Schnittstelle Shape	1240
18.4.2	Pfade *	1242
18.5	Das Innere und Äußere einer Form	1242
18.5.1	Farben und die Paint-Schnittstelle	1243
18.5.2	Farben mit der Klasse Color	1243
18.5.3	Composite und XOR *	1244
18.5.4	Dicke und Art der Linien von Formen bestimmen über Stroke *	1244
18.6	Bilder	1249
18.6.1	Eine Übersicht über die Bilder-Bibliotheken	1249
18.6.2	Bilder mit ImageIO lesen	1250
18.6.3	Ein Bild zeichnen	1251
18.6.4	Splash-Screen *	1254
18.6.5	Bilder skalieren *	1255
18.6.6	Schreiben mit ImageIO	1257
18.6.7	Asynchrones Laden mit getImage(...) und dem MediaTracker *	1261
18.7	Weitere Eigenschaften von Graphics *	1263
18.7.1	Eine Kopie von Graphics erstellen	1263
18.7.2	Koordinatensystem verschieben	1264
18.7.3	Beschnitt (Clipping)	1264
18.7.4	Zeichenhinweise durch RenderingHints	1268
18.7.5	Transformationen mit einem AffineTransform-Objekt	1269
18.8	Drucken *	1271
18.8.1	Drucken der Inhalte	1271
18.8.2	Bekannte Drucker	1273
18.9	Benutzerinteraktionen automatisieren, Robot und Screenshots *	1274
18.9.1	Der Roboter	1274
18.9.2	Automatisch in die Tasten hauen	1275
18.9.3	Automatisierte Mausoperationen	1275
18.9.4	Methoden zur Zeitsteuerung	1276
18.9.5	Bildschirmabzüge (Screenshots)	1276
18.9.6	Funktionsweise und Beschränkungen	1278
18.9.7	MouseInfo und PointerInfo	1278
18.10	Zum Weiterlesen	1280

19	JavaFX	1281
19.1	Das erste Programm mit JavaFX	1281
19.2	Zentrale Typen in JavaFX	1284
19.2.1	Szenegraph-Knoten und Container-Typen	1284
19.2.2	Datenstrukturen	1285
19.3	JavaFX-Komponenten und Layout-Container-Klassen	1286
19.3.1	Überblick über die Komponenten	1286
19.3.2	Listener/Handler zur Ereignisbeobachtung	1287
19.3.3	Panels mit speziellen Layouts	1288
19.4	Webbrowser	1290
19.5	Geometrische Objekte	1291
19.5.1	Linien und Rechtecke	1292
19.5.2	Kreise, Ellipsen, Kreisförmiges	1294
19.5.3	Es werde kurvig – quadratische und kubische Splines	1295
19.5.4	Pfade *	1297
19.5.5	Polygone und Polylines	1300
19.5.6	Beschriftungen, Texte, Fonts	1300
19.5.7	Die Oberklasse Shape	1302
19.6	Füllart von Formen	1304
19.6.1	Farben mit der Klasse Color	1304
19.7	Grafiken	1308
19.7.1	Klasse Image	1308
19.7.2	ImageView	1309
19.7.3	Programm-Icon/Fenster-Icon setzen	1310
19.7.4	Zugriff auf die Pixel und neue Pixel-Bilder *	1310
19.8	Deklarative Oberflächen mit FXML	1313
19.9	Diagramme (Charts)	1315
19.9.1	Kuchendiagramm	1315
19.9.2	Balkendiagramm	1317
19.10	Animationen	1319
19.10.1	FadeTransition	1320
19.10.2	ScaleTransition	1320
19.10.3	Transitionen parallel oder sequenziell durchführen	1321
19.11	Medien abspielen	1322
19.12	Java 3D	1323
19.13	Das Geometry-Paket *	1323

19.14	JavaFX-Scene in Swing-Applikationen einbetten	1324
19.15	Zum Weiterlesen	1326
20	Sicherheitskonzepte	1327
20.1	Zentrale Elemente der Java-Sicherheit	1327
20.1.1	Sichere Java Virtual Machine	1327
20.1.2	Der Sandkasten (Sandbox)	1327
20.1.3	Security-API der Java SE	1328
20.1.4	Cryptographic Service Providers	1329
20.2	Sicherheitsmanager (Security-Manager)	1330
20.2.1	Der Sicherheitsmanager bei Applets	1330
20.2.2	Sicherheitsmanager aktivieren	1331
20.2.3	Rechte durch Policy-Dateien vergeben	1334
20.2.4	Erstellen von Rechedateien mit dem grafischen Policy-Tool	1336
20.2.5	Kritik an den Policies	1337
20.3	Signierung	1338
20.3.1	Warum signieren?	1339
20.3.2	Digitale Ausweise und die Zertifizierungsstelle	1339
20.3.3	Mit keytool Schlüssel erzeugen	1339
20.3.4	Signieren mit jarsigner	1340
20.4	Kryptografische Hashfunktion	1341
20.4.1	Die MDx-Reihe	1342
20.4.2	Secure Hash Algorithm (SHA)	1342
20.4.3	Mit der Security-API einen Fingerabdruck berechnen	1343
20.4.4	Die Klasse MessageDigest	1343
20.5	Verschlüsseln von Daten(-strömen) *	1346
20.5.1	Den Schlüssel, bitte	1346
20.5.2	Verschlüsseln mit Cipher	1347
20.5.3	Verschlüsseln von Datenströmen	1348
20.6	Zum Weiterlesen	1349
21	Dynamische Übersetzung, Skriptsprachen, JShell	1351
21.1	Codegenerierung	1352
21.1.1	Generierung von Java-Quellcode	1353

21.1.2	Codetransformationen	1355
21.1.3	Erstellen von Java-Bytecode	1355
21.2	Programme mit der Java Compiler API übersetzen	1356
21.2.1	Java Compiler API	1356
21.2.2	Fehlerdiagnose	1359
21.2.3	Eine im String angegebene Kompilationseinheit übersetzen	1361
21.2.4	Wenn Quelle und Ziel der Speicher sind	1362
21.3	Ausführen von Skripten	1366
21.3.1	Java-Programme mit JavaScript schreiben	1366
21.3.2	Kommandozeilenprogramme jrunscript und jjs	1367
21.3.3	javax.script-API	1367
21.3.4	JavaScript-Programme ausführen	1368
21.3.5	Alternative Sprachen für die JVM	1369
21.3.6	Von den Schwierigkeiten, dynamische Programmiersprachen auf die JVM zu bringen *	1371
21.4	JShell, die interaktive REPL-Shell	1375
21.4.1	JShell API	1380
21.5	Zum Weiterlesen	1381
22	Java Native Interface (JNI)	1383
22.1	Java Native Interface und Invocation-API	1383
22.2	Eine C-Funktion in ein Java-Programm einbinden	1384
22.2.1	Den Java-Code schreiben	1384
22.3	Dynamische Bibliotheken erzeugen	1386
22.3.1	Die Header-Datei erzeugen	1386
22.3.2	Implementierung der Funktion in C	1388
22.3.3	Die C-Programme übersetzen und die dynamische Bibliothek erzeugen	1389
22.3.4	Abhängige Bibliotheken entdecken	1390
22.3.5	Nativ die String-Länge ermitteln	1390
22.4	Erweiterte JNI-Eigenschaften	1392
22.4.1	Klassendefinitionen	1392
22.4.2	Zugriff auf Attribute	1392
22.4.3	Methoden aufrufen	1395
22.4.4	Threads und Synchronisation	1396
22.4.5	@Native Markierungen *	1397

22.5 Einfache Anbindung von existierenden Bibliotheken	1397
22.5.1 JNA (Java Native Access)	1397
22.5.2 BridJ	1398
22.5.3 Generieren von JNI-Wrappern aus C++-Klassen und C-Headern	1399
22.5.4 COM-Schnittstellen anzapfen	1399
22.6 Invocation-API	1399
22.7 Zum Weiterlesen	1400
23 Dienstprogramme für die Java-Umgebung	1401
<hr/>	
23.1 Programme des JDK	1401
23.2 Monitoringprogramme vom JDK	1401
23.2.1 jps	1401
23.2.2 jstat	1402
23.2.3 jmap	1402
23.2.4 jstack	1403
23.2.5 jcmd	1405
23.2.6 jhsdb	1407
23.2.7 VisualVM	1407
23.3 Ant	1411
23.3.1 Bezug und Installation von Ant	1412
23.3.2 Das Build-Skript build.xml	1412
23.3.3 Build den Build	1413
23.3.4 Properties	1413
23.3.5 Externe und vordefinierte Properties	1415
23.3.6 Weitere Ant-Tasks	1416
23.4 Disassembler, Decompiler und Obfuscator	1417
23.4.1 Der Disassembler javap*	1417
23.4.2 Decompiler	1422
23.4.3 Obfuscatoren	1424
23.5 Weitere Dienstprogramme	1426
23.5.1 Java-Programme als Systemdienst ausführen	1426
23.6 Zum Weiterlesen	1426
Index	1427