

Auf einen Blick

1	Neues in Java 8 und Java 7	43
2	Fortgeschrittene String-Verarbeitung	121
3	Threads und nebenläufige Programmierung	177
4	Datenstrukturen und Algorithmen	275
5	Raum und Zeit	443
6	Dateien, Verzeichnisse und Dateizugriffe	509
7	Datenströme	573
8	Die eXtensible Markup Language (XML)	665
9	Dateiformate	749
10	Grafische Oberflächen mit Swing	777
11	Grafikprogrammierung	975
12	JavaFX	1031
13	Netzwerkprogrammierung	1077
14	Verteilte Programmierung mit RMI	1145
15	RESTful und SOAP-Web-Services	1165
16	Technologien für die Infrastruktur	1185
17	Typen, Reflection und Annotationen	1203
18	Dynamische Übersetzung und Skriptsprachen	1269
19	Logging und Monitoring	1293
20	Sicherheitskonzepte	1315
21	Datenbankmanagement mit JDBC	1337
22	Java Native Interface (JNI)	1383
23	Dienstprogramme für die Java-Umgebung	1401

Inhalt

Vorwort	35
1 Neues in Java 8 und Java 7	43
1.1 Sprachänderungen in Java 8	43
1.1.1 Statische ausprogrammierte Methoden in Schnittstellen	43
1.1.2 Default-Methoden	44
1.1.3 Erweiterte Schnittstellen deklarieren und nutzen	46
1.1.4 Erweiterte Schnittstellen, Mehrfachvererbung und Mehrdeutigkeiten *	49
1.1.5 Bausteine bilden mit Default-Methoden *	53
1.2 Lambda-Ausdrücke und funktionale Programmierung	59
1.2.1 Code = Daten	59
1.2.2 Funktionale Schnittstellen und Lambda-Ausdrücke im Detail	62
1.2.3 Methoden-Referenz	80
1.2.4 Konstruktor-Referenz	83
1.2.5 Implementierung von Lambda-Ausdrücken	86
1.2.6 Funktionale Programmierung mit Java	87
1.2.7 Funktionale Schnittstelle aus dem java.util.function-Paket	91
1.2.8 Optional ist keine Nullnummer	104
1.2.9 Was ist jetzt so funktional?	114
1.2.10 Zum Weiterlesen	116
1.3 Bibliotheksänderungen in Java 8	117
1.4 JDK 8-HotSpot-JVM-Änderungen	117
1.5 Auf Java 7/8-Syntax mit NetBeans und Eclipse migrieren	117
1.5.1 Java 8-Syntax-Migration mit NetBeans	118
1.5.2 Java 8-Syntax-Migration mit Eclipse	119
1.5.3 File-Klassen auf NIO.2 umstellen	119
1.6 Zum Weiterlesen	120
2 Fortgeschrittene String-Verarbeitung	121
2.1 Erweitere Zeicheneigenschaften	121
2.1.1 isXXX(...)-Methoden	121
2.1.2 Unicode-Blöcke	122
2.1.3 Unicode-Skripte	122

2.2	Reguläre Ausdrücke	123
2.2.1	Pattern.matches(...) bzw. String#matches(...)	124
2.2.2	Die Klassen Pattern und Matcher	127
2.2.3	Finden und nicht matchen	132
2.2.4	Gruppen	134
2.2.5	Gierige und nicht gierige Operatoren *	134
2.2.6	Mit MatchResult alle Ergebnisse einsammeln *	135
2.2.7	Suchen und Ersetzen mit Mustern	137
2.2.8	Hangman Version 2	139
2.3	Zerlegen von Zeichenketten	140
2.3.1	Zerlegen von Zeichensequenzen über String oder Pattern	140
2.3.2	Mehr vom Scanner	141
2.3.3	Die Klasse StringTokenizer *	146
2.3.4	BreakIterator als Zeichen-, Wort-, Zeilen- und Satztrenner *	148
2.3.5	StreamTokenizer *	151
2.4	Zeichenkodierungen, XML/HTML-Entities, Base64 *	154
2.4.1	Unicode und 8-Bit-Abbildungen	154
2.4.2	Kodierungen über die Klasse String vornehmen	154
2.4.3	Das Paket java.nio.charset und der Typ Charset	155
2.4.4	Konvertieren mit OutputStreamWriter-/InputStreamReader-Klassen	156
2.4.5	XML/HTML-Entities ausmaskieren	156
2.4.6	Base64-Kodierung	158
2.5	Ausgaben formatieren	159
2.5.1	Die Formatter-Klasse *	159
2.5.2	Formatieren mit Masken *	162
2.5.3	Format-Klassen	163
2.5.4	Zahlen, Prozente und Währungen mit NumberFormat und DecimalFormat formatieren *	166
2.5.5	MessageFormat und Pluralbildung mit ChoiceFormat	169
2.6	Sprachabhängiges Vergleichen und Normalisierung *	170
2.6.1	Die Klasse Collator	171
2.6.2	Effiziente interne Speicherung für die Sortierung	173
2.6.3	Normalisierung	174
2.7	Phonetische Vergleiche *	175
2.8	Zum Weiterlesen	176

3.1	Threads erzeugen	177
3.1.1	Threads über die Schnittstelle Runnable implementieren	177
3.1.2	Thread mit Runnable starten	178
3.1.3	Die Klasse Thread erweitern	180
3.2	Thread-Eigenschaften und -Zustände	182
3.2.1	Der Name eines Threads	182
3.2.2	Wer bin ich?	183
3.2.3	Die Zustände eines Threads *	183
3.2.4	Schläfer gesucht	184
3.2.5	Mit yield() auf Rechenzeit verzichten	186
3.2.6	Der Thread als Dämon	187
3.2.7	Freiheit für den Thread – das Ende	189
3.2.8	Einen Thread höflich mit Interrupt beenden	189
3.2.9	UncaughtExceptionHandler für unbehandelte Ausnahmen	191
3.2.10	Der stop() von außen und die Rettung mit ThreadDeath *	192
3.2.11	Ein Rendezvous mit join(...) *	194
3.2.12	Arbeit niederlegen und wieder aufnehmen *	196
3.2.13	Priorität *	196
3.3	Der Ausführer (Executor) kommt	197
3.3.1	Die Schnittstelle Executor	198
3.3.2	Glücklich in der Gruppe – die Thread-Pools	199
3.3.3	Threads mit Rückgabe über Callable	201
3.3.4	Mehrere Callable abarbeiten	204
3.3.5	ScheduledExecutorService für wiederholende Ausgaben und Zeitsteuerungen nutzen	205
3.4	Synchronisation über kritische Abschnitte	206
3.4.1	Gemeinsam genutzte Daten	206
3.4.2	Probleme beim gemeinsamen Zugriff und kritische Abschnitte	206
3.4.3	Punkte nebenläufig initialisieren	208
3.4.4	i++ sieht atomar aus, ist es aber nicht *	210
3.4.5	Kritische Abschnitte schützen	211
3.4.6	Kritische Abschnitte mit ReentrantLock schützen	212
3.4.7	Synchronisieren mit synchronized	219
3.4.8	Synchronized-Methoden der Klasse StringBuffer *	220
3.4.9	Mit synchronized synchronisierte Blöcke	221
3.4.10	Dann machen wir doch gleich alles synchronisiert!	222
3.4.11	Lock-Freigabe im Fall von Exceptions	223

3.4.12	Deadlocks	224
3.4.13	Mit synchronized nachträglich synchronisieren *	226
3.4.14	Monitore sind reentrant – gut für die Geschwindigkeit *	227
3.4.15	Synchronisierte Methodenaufrufe zusammenfassen *	228
3.5	Synchronisation über Warten und Benachrichtigen	229
3.5.1	Die Schnittstelle Condition	230
3.5.2	It's Disco-Time *	233
3.5.3	Warten mit wait(...) und Aufwecken mit notify()/notifyAll() *	237
3.5.4	Falls der Lock fehlt – IllegalMonitorStateException *	239
3.6	Datensynchronisation durch besondere Concurrency-Klassen *	240
3.6.1	Semaphor	241
3.6.2	Barrier und Austausch	244
3.6.3	Stop and go mit Exchanger	246
3.7	Atomare Operationen und frische Werte mit volatile *	246
3.7.1	Der Modifizierer volatile bei Objekt-/Klassenvariablen	247
3.7.2	Das Paket java.util.concurrent.atomic	248
3.8	Teile und herrsche mit Fork und Join *	250
3.8.1	Algorithmendesign per »teile und herrsche«	250
3.8.2	Nebenläufiges Lösen von D&C-Algorithmen	252
3.8.3	Fork und Join	253
3.9	CompletionStage und CompletableFuture *	256
3.10	Mit dem Thread verbundene Variablen *	257
3.10.1	ThreadLocal	257
3.10.2	InheritableThreadLocal	259
3.10.3	ThreadLocalRandom als schneller nebenläufiger Zufallszahlengenerator ...	260
3.10.4	ThreadLocal bei der Performance-Optimierung	262
3.11	Threads in einer Thread-Gruppe *	263
3.11.1	Aktive Threads in der Umgebung	263
3.11.2	Etwas über die aktuelle Thread-Gruppe herausfinden	264
3.11.3	Threads in einer Thread-Gruppe anlegen	266
3.11.4	Methoden von Thread und ThreadGroup im Vergleich	269
3.12	Zeitgesteuerte Abläufe	270
3.12.1	Die Typen Timer und TimerTask	271
3.12.2	Job-Scheduler Quartz	272
3.13	Einen Abbruch der virtuellen Maschine erkennen	273
3.13.1	Shutdown-Hook	273
3.13.2	Signale	274
3.14	Zum Weiterlesen	274

4.1	Datenstrukturen und die Collection-API	275
4.1.1	Designprinzip mit Schnittstellen, abstrakten und konkreten Klassen	276
4.1.2	Die Basisschnittstellen Collection und Map	276
4.1.3	Die Utility-Klassen Collections und Arrays	277
4.1.4	Das erste Programm mit Container-Klassen	277
4.1.5	Die Schnittstelle Collection und Kernkonzepte	278
4.1.6	Schnittstellen, die Collection erweitern; und Map	282
4.1.7	Konkrete Container-Klassen	284
4.1.8	Generische Datentypen in der Collection-API	285
4.1.9	Die Schnittstelle Iterable und das erweiterte for	286
4.2	Listen	287
4.2.1	Erstes Listen-Beispiel	287
4.2.2	Auswahlkriterium ArrayList oder LinkedList	288
4.2.3	Die Schnittstelle List	289
4.2.4	ArrayList	295
4.2.5	LinkedList	296
4.2.6	Der Feld-Adapter Arrays.asList(...)	298
4.2.7	ListIterator *	300
4.2.8	toArray(...) von Collection verstehen – die Gefahr einer Falle erkennen	301
4.2.9	Primitive Elemente in Datenstrukturen verwalten	304
4.3	Mengen (Sets)	305
4.3.1	Ein erstes Mengen-Beispiel	305
4.3.2	Methoden der Schnittstelle Set	307
4.3.3	HashSet	309
4.3.4	TreeSet – die sortierte Menge	309
4.3.5	Die Schnittstellen NavigableSet und SortedSet	311
4.3.6	LinkedHashSet	313
4.4	Queues (Schlangen) und Deques	314
4.4.1	Queue-Klassen	315
4.4.2	Deque-Klassen	316
4.4.3	Blockierende Queues und Prioritätswarteschlangen	317
4.4.4	PriorityQueue	318
4.5	Stack (Kellerspeicher, Stapel)	323
4.5.1	Die Methoden von java.util.Stack	324
4.6	Assoziative Speicher	325
4.6.1	Die Klassen HashMap und TreeMap	325
4.6.2	Einfügen und Abfragen des Assoziativspeichers	327

4.6.3	Über die Bedeutung von equals(...) und hashCode() bei Elementen	334
4.6.4	Eigene Objekte hashen	335
4.6.5	LinkedHashMap und LRU-Implementierungen	336
4.6.6	IdentityHashMap	337
4.6.7	Das Problem veränderter Elemente	338
4.6.8	Aufzählungen und Ansichten des Assoziativspeichers	338
4.6.9	Die Arbeitsweise einer Hash-Tabelle *	342
4.6.10	Die Properties-Klasse	344
4.7	Mit einem Iterator durch die Daten wandern	348
4.8	Iterator-Schnittstelle	348
4.8.1	Der Iterator kann (eventuell auch) löschen	349
4.8.2	Operationen auf allen Elementen durchführen	349
4.8.3	Einen Zufallszahlen-Iterator schreiben	350
4.8.4	Iteratoren von Sammlungen, das erweiterte for und Iterable	351
4.8.5	Fail-Fast-Iterator und die ConcurrentModificationException	355
4.8.6	Die Schnittstelle Enumerator *	356
4.9	Algorithmen in Collections	358
4.9.1	Die Bedeutung von Ordnung mit Comparator und Comparable	359
4.9.2	Sortieren	360
4.9.3	Den größten und kleinsten Wert einer Collection finden	363
4.9.4	Nichtänderbare Datenstrukturen, immutable oder nur lesen?	365
4.9.5	Null Object Pattern und leere Sammlungen/Iteratoren zurückgeben	369
4.9.6	Echte typsichere Container	372
4.9.7	Mit der Halbierungssuche nach Elementen fahnden	373
4.9.8	Ersetzen, Kopieren, Füllen, Umdrehen, Rotieren *	375
4.9.9	Listen durchwürfeln *	376
4.9.10	Häufigkeit eines Elements *	377
4.9.11	Singletons *	378
4.9.12	nCopies(...) *	378
4.10	Datenstrukturen mit Änderungsmeldungen	379
4.10.1	Das Paket javafx.collections	379
4.10.2	Fabrikmethoden in FXCollections	380
4.10.3	Änderungen melden über InvalidationListener	382
4.10.4	Änderungen melden über XXXChangeListener	382
4.10.5	Change-Klassen	383
4.10.6	Weitere Hilfsmethoden einer ObservableList	385
4.10.7	Melden von Änderungen an Arrays	386
4.10.8	Transformierte FXCollections	387
4.10.9	Weitere statische Methoden in FXCollections	388

4.11	Stream-API	389
4.11.1	Stream erzeugen	391
4.11.2	Terminale Operationen	393
4.11.3	Intermediäre Operationen	404
4.11.4	Streams mit primitiven Werten	408
4.11.5	Stream-Beziehungen, AutoCloseable	419
4.11.6	Stream-Builder	421
4.11.7	Spliterator	422
4.11.8	Klasse StreamSupport	422
4.12	Spezielle threadsichere Datenstrukturen	423
4.12.1	Zu Beginn nur synchronisierte Datenstrukturen in Java 1.0	423
4.12.2	Nicht synchronisierte Datenstrukturen in der Standard-Collection-API	423
4.12.3	Nebenläufiger Assoziativspeicher und die Schnittstelle ConcurrentMap	424
4.12.4	ConcurrentLinkedQueue	424
4.12.5	CopyOnWriteArrayList und CopyOnWriteArraySet	424
4.12.6	Wrapper zur Synchronisation	425
4.12.7	Blockierende Warteschlangen	426
4.12.8	ArrayBlockingQueue und LinkedBlockingQueue	427
4.12.9	PriorityBlockingQueue	428
4.12.10	Transfer-Warteschlangen – TransferQueue und LinkedTransferQueue	432
4.13	Google Guava (Google Collections Library)	432
4.13.1	Beispiel Multi-Set und Multi-Map	433
4.13.2	Datenstrukturen aus Guava	433
4.13.3	Utility-Klassen von Guava	436
4.13.4	Prädikate	436
4.13.5	Transformationen	437
4.14	Die Klasse BitSet für Bitmengen *	437
4.14.1	Ein BitSet anlegen	438
4.14.2	BitSet füllen und Zustände erfragen	439
4.14.3	Mengenorientierte Operationen	440
4.14.4	Weitere Methoden von BitSet	441
4.14.5	Primzahlen in einem BitSet verwalten	442
4.15	Zum Weiterlesen	442
5	Raum und Zeit	443
5.1	Weltzeit *	443
5.2	Wichtige Datum-Klassen im Überblick	444
5.2.1	Der 1.1.1970	445

5.2.2	System.currentTimeMillis()	445
5.2.3	Einfache Zeitumrechnungen durch TimeUnit	445
5.3	Sprachen der Länder	446
5.3.1	Sprachen und Regionen über Locale-Objekte	447
5.4	Internationalisierung und Lokalisierung	449
5.4.1	ResourceBundle-Objekte und Ressource-Dateien	450
5.4.2	Ressource-Dateien zur Lokalisierung	450
5.4.3	Die Klasse ResourceBundle	452
5.4.4	Ladestrategie für ResourceBundle-Objekte	452
5.4.5	Ladeprozess und Format anpassen *	453
5.5	Die Klasse Date	455
5.5.1	Objekte erzeugen und Methoden nutzen	455
5.5.2	Date-Objekte sind nicht immutable	457
5.6	Calendar und GregorianCalendar	457
5.6.1	Die abstrakte Klasse Calendar	458
5.6.2	Calendar nach Date und Millisekunden fragen	459
5.6.3	Abfragen und Setzen von Datumselementen über Feldbezeichner	459
5.6.4	Kalender-Typen *	463
5.6.5	Kalender-Exemplare bauen über den Calendar.Builder	463
5.6.6	Wie viele Tage hat der Monat, oder wie viele Monate hat ein Jahr? *	464
5.6.7	Wann beginnt die Woche und wann die erste Woche im Jahr? *	465
5.6.8	Der gregorianische Kalender	467
5.7	Zeitzone in Java *	470
5.7.1	Zeitzone durch die Klasse TimeZone repräsentieren	470
5.7.2	SimpleTimeZone	471
5.7.3	Methoden von TimeZone	473
5.8	Formatieren und Parsen von Datumsangaben	474
5.8.1	Ausgaben mit printf(...)	474
5.8.2	Ausgaben mit Calendar-Methoden getDisplayName(...) *	475
5.8.3	Mit DateFormat und SimpleDateFormat formatieren	475
5.8.4	Parsen von Datumswerten	481
5.9	Date-Time-API in Java 8	483
5.9.1	Datumsklasse LocalDate	487
5.9.2	Ostertage *	488
5.9.3	Die Klasse YearMonth	489
5.9.4	Die Klasse MonthDay	490
5.9.5	Aufzählung DayOfWeek und Month	490
5.9.6	Klasse LocalTime	491
5.9.7	Klasse LocalDateTime	491

5.9.8	Klasse Year	492
5.9.9	Zeitzonen-Klassen ZoneId und ZoneOffset	492
5.9.10	Temporale Klassen mit Zeitzoneninformationen	493
5.9.11	Klassen Period und Duration	497
5.9.12	Klasse Instant	499
5.9.13	Parsen und Formatieren von Datumszeitwerten	500
5.9.14	Das Paket java.time.temporal *	500
5.9.15	Konvertierungen zwischen der klassischen API und Date-Time-API	505
5.10	Die Default-Falle	506
5.11	Zum Weiterlesen	507

6 Dateien, Verzeichnisse und Dateizugriffe 509

6.1	Alte und neue Welt in java.io und java.nio	509
6.1.1	java.io-Paket mit File-Klasse	509
6.1.2	NIO.2 und java.nio-Paket	510
6.2	Dateisysteme und Pfade	510
6.2.1	FileSystem und Path	510
6.2.2	Die Utility-Klasse Files	516
6.2.3	Dateien kopieren und verschieben	518
6.2.4	Dateiattribute *	520
6.2.5	Neue Dateien, Verzeichnisse, symbolische Verknüpfungen anlegen und löschen	529
6.2.6	MIME-Typen herausfinden *	531
6.2.7	Verzeichnislistings (DirectoryStream/Stream) und Filter *	532
6.2.8	Rekursives Ablaufen des Verzeichnisbaums *	535
6.2.9	Rekursiv nach Dateien/Ordern suchen mit Files.find(...) *	538
6.2.10	Dateisysteme und Dateisystemattribute *	539
6.2.11	Verzeichnisse im Dateisystem überwachen *	543
6.3	Datei- und Verzeichnis-Operationen mit der Klasse File	545
6.3.1	Dateien und Verzeichnisse mit der Klasse File	545
6.3.2	Verzeichnis oder Datei? Existiert es?	549
6.3.3	Verzeichnis- und Dateieigenschaften/-attribute	549
6.3.4	Umbenennen und Verzeichnisse anlegen	552
6.3.5	Verzeichnisse auflisten und Dateien filtern	552
6.3.6	Dateien berühren, neue Dateien anlegen, temporäre Dateien	556
6.3.7	Dateien und Verzeichnisse löschen	557
6.3.8	Wurzelverzeichnis, Laufwerksnamen, Plattenspeicher *	558
6.3.9	URL-, URI- und Path-Objekte aus einem File-Objekt ableiten *	561

6.3.10	Mit Locking Dateien sperren *	561
6.3.11	Sicherheitsprüfung *	563
6.3.12	Zugriff auf SMB-Server mit jCIFS *	563
6.4	Dateien mit wahlfreiem Zugriff	563
6.4.1	Ein RandomAccessFile zum Lesen und Schreiben öffnen	564
6.4.2	Aus dem RandomAccessFile lesen	565
6.4.3	Schreiben mit RandomAccessFile	567
6.4.4	Die Länge des RandomAccessFile	567
6.4.5	Hin und her in der Datei	568
6.5	Wahlfreier Zugriff mit SeekableByteChannel und ByteBuffer *	569
6.5.1	SeekableByteChannel	569
6.5.2	ByteBuffer	570
6.5.3	Beispiel mit Path + SeekableByteChannel + ByteBuffer	570
6.5.4	FileChannel	571
6.6	Zum Weiterlesen	572
7	Datenströme	573
7.1	Stream-Klassen für Bytes und Zeichen	573
7.1.1	Lesen aus Dateien und Schreiben in Dateien	574
7.1.2	Byteorientierte Datenströme über Files beziehen	574
7.1.3	Zeichenorientierte Datenströme über Files beziehen	575
7.1.4	Funktion von OpenOption bei den Files.newXXX(...)-Methoden	576
7.1.5	Ressourcen aus dem Klassenpfad und aus JAR-Archiven laden	578
7.1.6	Die Schnittstellen Closeable, AutoCloseable und Flushable	579
7.2	Basisklassen für die Ein-/Ausgabe	581
7.2.1	Die abstrakten Basisklassen	581
7.2.2	Übersicht über Ein-/Ausgabeklassen	581
7.2.3	Die abstrakte Basisklasse OutputStream	584
7.2.4	Ein Datenschlucker *	585
7.2.5	Die abstrakte Basisklasse InputStream	586
7.2.6	Ströme mit SequencelInputStream zusammensetzen *	587
7.2.7	Die abstrakte Basisklasse Writer	589
7.2.8	Die Schnittstelle Appendable *	591
7.2.9	Die abstrakte Basisklasse Reader	591
7.3	Formatierte Textausgaben	594
7.3.1	Die Klassen PrintWriter und PrintStream	594
7.3.2	System.out, System.err und System.in	600

7.4	Die FileXXX-Stromklassen	602
7.4.1	Kopieren mit FileOutputStream und FileInputStream	603
7.4.2	Das FileDescriptor-Objekt *	606
7.4.3	Mit dem FileWriter Texte in Dateien schreiben	607
7.4.4	Zeichen mit der Klasse FileReader lesen	608
7.5	Schreiben und Lesen aus Strings und Byte-Feldern	609
7.5.1	Mit dem StringWriter ein String-Objekt füllen	609
7.5.2	CharArrayWriter	610
7.5.3	StringReader und CharArrayReader	611
7.5.4	Mit ByteArrayOutputStream in ein Byte-Feld schreiben	612
7.5.5	Mit ByteArrayInputStream aus einem Byte-Feld lesen	613
7.6	Datenströme filtern und verketten	614
7.6.1	Streams als Filter verketten (verschachteln)	615
7.6.2	Gepufferte Ausgaben mit BufferedWriter und BufferedOutputStream	616
7.6.3	Gepufferte Eingaben mit BufferedReader/BufferedInputStream	618
7.6.4	LineNumberReader zählt automatisch Zeilen mit *	619
7.6.5	Daten mit der Klasse PushbackReader zurücklegen *	620
7.6.6	DataOutputStream/DataInputStream *	623
7.6.7	Basisklassen für Filter *	623
7.6.8	Die Basisklasse FilterWriter *	624
7.6.9	Ein LowerCaseWriter *	625
7.6.10	Eingaben mit der Klasse FilterReader filtern *	626
7.6.11	Anwendungen für FilterReader und FilterWriter *	627
7.7	Vermittler zwischen Byte-Streams und Unicode-Strömen	633
7.7.1	Datenkonvertierung durch den OutputStreamWriter	633
7.7.2	Automatische Konvertierungen mit dem InputStreamReader	634
7.8	Kommunikation zwischen Threads mit Pipes *	636
7.8.1	PipedOutputStream und PipedInputStream	636
7.8.2	PipedWriter und PipedReader	638
7.9	Prüfsummen	640
7.9.1	Die Schnittstelle Checksum	640
7.9.2	Die Klasse CRC32	641
7.9.3	Die Adler32-Klasse	643
7.10	Persistente Objekte und Serialisierung	643
7.10.1	Objekte mit der Standardserialisierung speichern und lesen	644
7.10.2	Zwei einfache Anwendungen der Serialisierung *	647
7.10.3	Die Schnittstelle Serializable	648
7.10.4	Nicht serialisierbare Attribute aussparen	650
7.10.5	Das Abspeichern selbst in die Hand nehmen	652

7.10.6	Tiefe Objektkopien *	655
7.10.7	Versionenverwaltung und die SUID	657
7.10.8	Wie die ArrayList serialisiert *	659
7.10.9	Probleme mit der Serialisierung	660
7.11	Alternative Datenaustauschformate	661
7.11.1	Serialisieren in XML-Dateien	661
7.11.2	XML-Serialisierung von JavaBeans mit JavaBeans Persistence *	661
7.11.3	Die Open-Source-Bibliothek XStream *	663
7.11.4	Binäre Serialisierung mit Google Protocol Buffers *	664
7.12	Zum Weiterlesen	664

8 Die eXtensible Markup Language (XML) 665

8.1	Auszeichnungssprachen	665
8.1.1	Die Standard Generalized Markup Language (SGML)	665
8.1.2	Extensible Markup Language (XML)	666
8.2	Eigenschaften von XML-Dokumenten	666
8.2.1	Elemente und Attribute	666
8.2.2	Beschreibungssprache für den Aufbau von XML-Dokumenten	669
8.2.3	Schema – die moderne Alternative zu DTD	673
8.2.4	Namensraum (Namespace)	675
8.2.5	XML-Applikationen *	676
8.3	Die Java-APIs für XML	677
8.3.1	Das Document Object Model (DOM)	678
8.3.2	Simple API for XML Parsing (SAX)	678
8.3.3	Pull-API StAX	678
8.3.4	Java Document Object Model (JDOM)	678
8.3.5	JAXP als Java-Schnittstelle zu XML	679
8.3.6	DOM-Bäume einlesen mit JAXP *	680
8.4	Java Architecture for XML Binding (JAXB)	680
8.4.1	Bean für JAXB aufbauen	680
8.4.2	Utility-Klasse JAXB	681
8.4.3	Ganze Objektgraphen schreiben und lesen	682
8.4.4	JAXBContext und Marshaller/Unmarshaller nutzen	684
8.4.5	Validierung	686
8.4.6	Weitere JAXB-Annotationen *	690
8.4.7	Beans aus XML-Schema-Datei generieren	697

8.5	Serielle Verarbeitung mit StAX	704
8.5.1	Unterschiede der Verarbeitungsmodelle	704
8.5.2	XML-Dateien mit dem Cursor-Verfahren lesen	706
8.5.3	XML-Dateien mit dem Iterator-Verfahren verarbeiten *	709
8.5.4	Mit Filtern arbeiten *	712
8.5.5	XML-Dokumente schreiben	713
8.6	Serielle Verarbeitung von XML mit SAX *	716
8.6.1	Schnittstellen von SAX	716
8.6.2	SAX-Parser erzeugen	717
8.6.3	Operationen der Schnittstelle ContentHandler	718
8.6.4	ErrorHandler und EntityResolver	720
8.7	XML-Dateien mit JDOM verarbeiten	721
8.7.1	JDOM beziehen	721
8.7.2	Paketübersicht *	722
8.7.3	Die Document-Klasse	723
8.7.4	Eingaben aus der Datei lesen	724
8.7.5	Das Dokument im XML-Format ausgeben	725
8.7.6	Der Dokumenttyp *	726
8.7.7	Elemente	727
8.7.8	Zugriff auf Elementinhalte	730
8.7.9	Liste mit Unterelementen erzeugen *	732
8.7.10	Neue Elemente einfügen und ändern	732
8.7.11	Attributinhalt lesen und ändern	735
8.7.12	XPath	738
8.8	Transformationen mit XSLT *	742
8.8.1	Templates und XPath als Kernelemente von XSLT	742
8.8.2	Umwandlung von XML-Dateien mit JDOM und JAXP	744
8.9	XML-Schema-Validierung *	745
8.9.1	SchemaFactory und Schema	745
8.9.2	Validator	745
8.9.3	Validierung unterschiedlicher Datenquellen durchführen	746
8.10	Zum Weiterlesen	747
9	Dateiformate	749

9.1	Einfache Dateiformate für strukturierte Daten	750
9.1.1	Property-Dateien mit java.util.Properties lesen und schreiben	750

9.1.2	CSV-Dateien	752
9.1.3	JSON-Serialisierung mit Jackson	754
9.2	Dokumentenformate	755
9.2.1	(X)HTML	756
9.2.2	PDF-Dokumente	757
9.2.3	Microsoft Office-Dokumente	757
9.2.4	OASIS Open Document Format	759
9.3	Datenkompression *	759
9.3.1	Java-Unterstützung beim Komprimieren	760
9.3.2	Daten packen und entpacken	761
9.3.3	Datenströme komprimieren	763
9.3.4	ZIP-Archive	766
9.3.5	JAR-Archive	772
9.4	Bildformate	772
9.5	Audiodateien	773
9.5.1	Die Arbeit mit Applets AudioClip	773
9.5.2	AudioClip von JavaFX	773
9.5.3	MIDI-Dateien abspielen	774
9.5.4	ID-Tags aus mp3-Dateien	776
10	Grafische Oberflächen mit Swing	777
10.1	AWT, JavaFoundation Classes und Swing	777
10.1.1	Das Abstract Window Toolkit (AWT)	777
10.1.2	Java Foundation Classes (JFC)	778
10.1.3	Was Swing von AWT-Komponenten unterscheidet	781
10.2	Mit NetBeans zur ersten Swing-Oberfläche	782
10.2.1	Projekt anlegen	782
10.2.2	Eine GUI-Klasse hinzufügen	784
10.2.3	Programm starten	786
10.2.4	Grafische Oberfläche aufbauen	786
10.2.5	Swing-Komponenten-Klassen	788
10.2.6	Funktionalität geben	790
10.3	Aller Swing-Anfang – Fenster zur Welt	793
10.3.1	Eine Uhr, bei der die Zeit nie vergeht	793
10.3.2	Swing-Fenster mit javax.swing.JFrame darstellen	794
10.3.3	Mit add(...) auf den Container	795
10.3.4	Fenster schließbar machen – setDefaultCloseOperation(int)	795

10.3.5	Sichtbarkeit des Fensters	796
10.3.6	Größe und Position des Fensters verändern	796
10.3.7	Fenster- und Dialogdekoration, Transparenz *	797
10.3.8	Die Klasse Toolkit *	798
10.3.9	Zum Vergleich: AWT-Fenster darstellen *	799
10.4	Beschriftungen (JLabel)	800
10.4.1	Mehrzeiliger Text, HTML in der Darstellung	803
10.5	Icon und ImagemIcon für Bilder auf Swing-Komponenten	804
10.5.1	Die Klasse ImagemIcon	804
10.6	Es tut sich was – Ereignisse beim AWT	806
10.6.1	Die Ereignisquellen und Horcher (Listener) von Swing	806
10.6.2	Listener implementieren	807
10.6.3	Listener bei dem Ereignisauslöser anmelden/abmelden	810
10.6.4	Adapterklassen nutzen	812
10.6.5	Innere Mitgliedsklassen und innere anonyme Klassen	814
10.6.6	Aufrufen der Listener im AWT-Event-Thread	816
10.6.7	Ereignisse, etwas genauer betrachtet *	816
10.7	Schaltflächen	819
10.7.1	Normale Schaltflächen (JButton)	819
10.7.2	Der aufmerksame ActionListener	822
10.7.3	Schaltflächen-Ereignisse vom Typ(ActionEvent)	823
10.7.4	Basisklasse AbstractButton	823
10.7.5	Wechselknopf (JToggleButton)	825
10.8	Textkomponenten	825
10.8.1	Text in einer Eingabezeile	826
10.8.2	Die Oberklasse der Textkomponenten (JTextComponent)	827
10.8.3	Geschützte Eingaben (JPasswordField)	829
10.8.4	Validierende Eingabefelder (JFormattedTextField)	829
10.8.5	Einfache mehrzeilige Textfelder (JTextArea)	831
10.8.6	Editor-Klasse (JEditorPane) *	833
10.9	Swing Action *	836
10.10	JComponent und Component als Basis aller Komponenten	838
10.10.1	Hinzufügen von Komponenten	838
10.10.2	Tooltips (Kurzhinweise)	839
10.10.3	Rahmen (Border) *	840
10.10.4	Fokus und Navigation *	842
10.10.5	Ereignisse jeder Komponente *	844
10.10.6	Die Größe und Position einer Komponente *	847
10.10.7	Komponenten-Ereignisse *	848

10.10.8	UI-Delegate – der wahre Zeichner *	848
10.10.9	Undurchsichtige (opake) Komponente *	851
10.10.10	Properties und Listener für Änderungen *	852
10.11	Container	852
10.11.1	Standardcontainer (JPanel)	852
10.11.2	Bereich mit automatischen Rollbalken (JScrollPane)	853
10.11.3	Reiter (JTabbedPane)	853
10.11.4	Teilungskomponente (JSplitPane)	855
10.12	Alles Auslegungssache – die Layoutmanager	855
10.12.1	Übersicht über Layoutmanager	856
10.12.2	Zuweisen eines Layoutmanagers	856
10.12.3	Im Fluss mit FlowLayout	857
10.12.4	BoxLayout	859
10.12.5	Mit BorderLayout in alle Himmelsrichtungen	860
10.12.6	Rasteranordnung mit GridLayout	862
10.12.7	Der GridBagLayoutmanager *	864
10.12.8	Null-Layout *	869
10.12.9	Weitere Layoutmanager	869
10.13	Rollbalken und Schieberegler	870
10.13.1	Schieberegler (JSlider)	870
10.13.2	Rollbalken (JScrollBar) *	871
10.14	Kontrollfelder, Optionsfelder, Kontrollfeldgruppen	875
10.14.1	Kontrollfelder (JCheckBox)	876
10.14.2	ItemSelectable, ItemListener und das ItemEvent	878
10.14.3	Sich gegenseitig ausschließende Optionen (JRadioButton)	879
10.15	Fortschritte bei Operationen überwachen *	881
10.15.1	Fortschrittsbalken (JProgressBar)	881
10.15.2	Dialog mit Fortschrittsanzeige (ProgressMonitor)	883
10.16	Menüs und Symbolleisten	883
10.16.1	Die Menüleisten und die Einträge	884
10.16.2	Menüeinträge definieren	885
10.16.3	Einträge durch Action-Objekte beschreiben	887
10.16.4	Mit der Tastatur – Mnemonics und Shortcut	888
10.16.5	Der Tastatur-Shortcut (Accelerator)	888
10.16.6	Tastenkürzel (Mnemonics)	890
10.16.7	Symbolleisten alias Toolbars	891
10.16.8	Popup-Menüs	894
10.16.9	System-Tray nutzen *	898

10.17 Das Model-View-Controller-Konzept	899
10.18 Auswahlmenüs, Listen und Spinner	901
10.18.1 Listen (JList)	901
10.18.2 Auswahlmenü (JComboBox)	908
10.18.3 Drehfeld (JSpinner) *	913
10.18.4 Datumsauswahl	914
10.19 Tabellen (JTable)	915
10.19.1 Ein eigenes Tabellen-Modell	916
10.19.2 Basisklasse für eigene Modelle (AbstractTableModel)	917
10.19.3 Ein vorgefertigtes Standardmodell (DefaultTableModel)	920
10.19.4 Ein eigener Renderer für Tabellen	922
10.19.5 Zell-Editoren	925
10.19.6 Automatisches Sortieren und Filtern mit RowSorter *	926
10.20 Bäume (JTree)	929
10.20.1 JTree und sein TreeModel und TreeNode	929
10.20.2 Selektionen bemerken	931
10.20.3 Das TreeModel von JTree *	931
10.21 JRootPane und JDesktopPane *	934
10.21.1 Wurzelkomponente der Top-Level-Komponenten (JRootPane)	934
10.21.2 JDesktopPane und die Kinder von JInternalFrame	935
10.21.3 JLayeredPane	937
10.22 Dialoge und Window-Objekte	937
10.22.1 JWindow und JDialog	938
10.22.2 Modal oder nichtmodal?	938
10.22.3 Standarddialoge mit JOptionPane	939
10.22.4 Der Dateiauswahldialog	942
10.22.5 Der Farbauswahldialog JColorChooser *	945
10.23 Flexibles Java-Look-and-Feel	947
10.23.1 Look-and-Feel global setzen	948
10.23.2 UIManager	948
10.23.3 Die Windows-Optik mit JGoodies Looks verbessern *	950
10.24 Swing-Komponenten neu erstellen oder verändern *	951
10.24.1 Überlagerungen mit dem Swing-Komponenten-Dekorator JLayer	952
10.25 Die Zwischenablage (Clipboard)	953
10.25.1 Clipboard-Objekte	953
10.25.2 Mit Transferable auf den Inhalt zugreifen	954
10.25.3 DataFlavor ist das Format der Daten in der Zwischenablage	955

10.25.4 Einfügungen in der Zwischenablage erkennen	957
10.25.5 Drag & Drop	957
10.26 Undo durchführen *	958
10.27 AWT, Swing und die Threads	960
10.27.1 Ereignisschlange (EventQueue) und AWT-Event-Thread	960
10.27.2 Swing ist nicht threadsicher	961
10.27.3 invokeLater(...) und invokeAndWait(...)	963
10.27.4 SwingWorker	965
10.27.5 Eigene Ereignisse in die Queue setzen *	967
10.27.6 Auf alle Ereignisse hören *	968
10.28 Barrierefreiheit mit der Java Accessibility API	968
10.29 Zeitliches Ausführen mit dem javax.swing.Timer	969
10.30 Die Zusatzkomponentenbibliothek SwingX	970
10.30.1 Im Angebot: Erweiterte und neue Swing-Komponenten	970
10.30.2 Überblick über erweiterte Standard-Swing-Klassen	971
10.30.3 Neue Swing-Klassen	972
10.30.4 Weitere SwingX-Klassen	973
10.30.5 SwingX-Installation	973
10.31 Zum Weiterlesen	973

11 Grafikprogrammierung 975

11.1 Grundlegendes zum Zeichnen	975
11.1.1 Die paint(Graphics)-Methode für das AWT-Frame	975
11.1.2 Die ereignisorientierte Programmierung ändert Fensterinhalte	977
11.1.3 Zeichnen von Inhalten auf ein JFrame	978
11.1.4 Auffordern zum Neuzeichnen mit repaint(...)	979
11.1.5 Java 2D-API	980
11.2 Einfache Zeichenmethoden	981
11.2.1 Linien	981
11.2.2 Rechtecke	981
11.2.3 Ovale und Kreisbögen	982
11.2.4 Polygone und Polylines	983
11.3 Zeichenketten schreiben und Fonts	985
11.3.1 Zeichenfolgen schreiben	985
11.3.2 Die Font-Klasse	986
11.3.3 Font-Metadaten durch FontMetrics *	987

11.4	Geometrische Objekte	990
11.4.1	Die Schnittstelle Shape	991
11.4.2	Pfade *	993
11.5	Das Innere und Äußere einer Form	993
11.5.1	Farben und die Paint-Schnittstelle	994
11.5.2	Farben mit der Klasse Color	994
11.5.3	Composite und XOR *	995
11.5.4	Dicke und Art der Linien von Formen bestimmen über Stroke *	995
11.6	Bilder	1000
11.6.1	Eine Übersicht über die Bilder-Bibliotheken	1000
11.6.2	Bilder mit ImageIO lesen	1000
11.6.3	Ein Bild zeichnen	1002
11.6.4	Splash-Screen *	1005
11.6.5	Bilder skalieren *	1005
11.6.6	Schreiben mit ImageIO	1008
11.6.7	Asynchrones Laden mit getImage(...) und dem MediaTracker *	1012
11.7	Weitere Eigenschaften von Graphics *	1013
11.7.1	Eine Kopie von Graphics erstellen	1013
11.7.2	Koordinatensystem verschieben	1014
11.7.3	Beschnitt (Clipping)	1015
11.7.4	Zeichenhinweise durch RenderingHints	1018
11.7.5	Transformationen mit einem AffineTransform-Objekt	1019
11.8	Drucken *	1021
11.8.1	Drucken der Inhalte	1021
11.8.2	Bekannte Drucker	1023
11.9	Benutzerinteraktionen automatisieren, Robot und Screenshots *	1024
11.9.1	Der Roboter	1024
11.9.2	Automatisch in die Tasten hauen	1025
11.9.3	Automatisierte Maus-Operationen	1025
11.9.4	Methoden zur Zeitsteuerung	1026
11.9.5	Bildschirmabzüge (Screenshots)	1026
11.9.6	Funktionsweise und Beschränkungen	1028
11.9.7	MouseInfo und PointerInfo	1028
11.10	Zum Weiterlesen	1030

12.1	Das erste Programm mit JavaFX	1031
12.2	Zentrale Typen in JavaFX	1034
12.2.1	Szenegraph-Knoten und Container-Typen	1034
12.2.2	Datenstrukturen	1035
12.3	JavaFX-Komponenten und Layout-Container-Klassen	1036
12.3.1	Überblick über die Komponenten	1036
12.3.2	Listener/Handler zur Ereignisbeobachtung	1037
12.3.3	Panels mit speziellen Layouts	1038
12.4	Webbrowser	1040
12.5	Geometrische Objekte	1041
12.5.1	Linien und Rechtecke	1042
12.5.2	Kreise, Ellipsen, Kreisförmiges	1044
12.5.3	Es werde kurvig – quadratische und kubische Splines	1045
12.5.4	Pfade *	1047
12.5.5	Polygone und Polylines	1050
12.5.6	Beschriftungen, Texte, Fonts	1050
12.5.7	Die Oberklasse Shape	1052
12.6	Füllart von Formen	1054
12.6.1	Farben mit der Klasse Color	1054
12.7	Grafiken	1057
12.7.1	Klasse Image	1058
12.7.2	ImageView	1058
12.7.3	Programm-Icon/Fenster-Icon setzen	1059
12.7.4	Zugriff auf die Pixel und neue Pixel-Bilder *	1060
12.8	Deklarative Oberflächen mit FXML	1062
12.9	Diagramme (Charts)	1065
12.9.1	Kuchendiagramm	1065
12.9.2	Balkendiagramm	1067
12.10	Animationen	1068
12.10.1	FadeTransition	1069
12.10.2	ScaleTransition	1070
12.10.3	Transitionen parallel oder sequenziell durchführen	1070
12.11	Medien abspielen	1071
12.12	Das Geometry-Paket *	1072

12.13	JavaFX-Szene in Swing-Applikationen einbetten	1073
12.14	Zum Weiterlesen	1075
13	Netzwerkprogrammierung	1077
13.1	Grundlegende Begriffe	1077
13.2	URI und URL	1078
13.2.1	Die Klasse URI	1079
13.2.2	Die Klasse URL	1079
13.2.3	Informationen über eine URL *	1081
13.2.4	Der Zugriff auf die Daten über die Klasse URL	1083
13.3	Die Klasse URLConnection *	1084
13.3.1	Methoden und Anwendung von URLConnection	1084
13.3.2	Protokoll- und Content-Handler	1087
13.3.3	Im Detail: Von der URL zur URLConnection	1088
13.3.4	Der Protokoll-Handler für JAR-Dateien	1089
13.3.5	Basic Authentication und Proxy-Authentifizierung	1090
13.4	Mit GET und POST Daten übergeben *	1092
13.4.1	Kodieren der Parameter für Serverprogramme	1092
13.4.2	In Wikipedia suchen und mit GET-Request absenden	1093
13.4.3	POST-Request absenden	1094
13.5	Host- und IP-Adressen	1095
13.5.1	Lebt der Rechner?	1097
13.5.2	IP-Adresse des lokalen Hosts	1098
13.5.3	Das Netz ist klasse *	1099
13.5.4	Networkinterface	1099
13.6	Mit dem Socket zum Server	1100
13.6.1	Das Netzwerk ist der Computer	1100
13.6.2	Sockets	1101
13.6.3	Eine Verbindung zum Server aufbauen	1101
13.6.4	Server unter Spannung – die Ströme	1103
13.6.5	Die Verbindung wieder abbauen	1103
13.6.6	Informationen über den Socket *	1104
13.6.7	Reine Verbindungsdaten über SocketAddress *	1106
13.7	Client-Server-Kommunikation	1107
13.7.1	Warten auf Verbindungen	1107
13.7.2	Ein Multiplikationsserver	1108
13.7.3	Blockierendes Lesen	1111

13.8	Apache HttpComponents und Commons Net *	1112
13.8.1	HttpComponents	1113
13.8.2	Apache Commons Net	1113
13.9	Arbeitsweise eines Webservers *	1114
13.9.1	Das Hypertext Transfer Protocol (HTTP)	1114
13.9.2	Anfragen an den Server	1115
13.9.3	Die Antworten vom Server	1116
13.9.4	Webserver mit com.sun.net.httpserver.HttpServer	1120
13.10	Verbindungen durch einen Proxy-Server *	1122
13.10.1	System-Properties	1122
13.10.2	Verbindungen durch die Proxy-API	1123
13.11	Datagram-Sockets *	1124
13.11.1	Die Klasse DatagramSocket	1126
13.11.2	Datagramme und die Klasse DatagramPacket	1126
13.11.3	Auf ein hereinkommendes Paket warten	1127
13.11.4	Ein Paket zum Senden vorbereiten	1128
13.11.5	Methoden der Klasse DatagramPacket	1128
13.11.6	Das Paket senden	1129
13.12	E-Mail *	1131
13.12.1	Wie eine Elektropost um die Welt geht	1131
13.12.2	Das Simple Mail Transfer Protocol und RFC 822	1131
13.12.3	POP (Post Office Protocol)	1132
13.12.4	Die JavaMail API	1132
13.12.5	E-Mails mittels POP3 abrufen	1134
13.12.6	Multipart-Nachrichten verarbeiten	1136
13.12.7	E-Mails versenden	1138
13.12.8	Ereignisse und Suchen	1141
13.13	Tiefer liegende Netzwerkeigenschaften *	1142
13.13.1	MAC-Adressen auslesen	1142
13.13.2	Internet Control Message Protocol (ICMP)	1143
13.14	Zum Weiterlesen	1143
14	Verteilte Programmierung mit RMI	1145
14.1	Entfernte Objekte und Methoden	1145
14.1.1	Stellvertreter helfen bei entfernten Methodenaufrufen	1145
14.1.2	Standards für entfernte Objekte	1147

14.2	Java Remote Method Invocation	1147
14.2.1	Zusammenspiel von Server, Registry und Client	1147
14.2.2	Wie die Stellvertreter die Daten übertragen	1148
14.2.3	Probleme mit entfernten Methoden	1148
14.2.4	Nutzen von RMI bei Middleware-Lösungen	1150
14.2.5	Zentrale Klassen und Schnittstellen	1150
14.2.6	Entfernte und lokale Objekte im Vergleich	1151
14.3	Auf der Serverseite	1151
14.3.1	Entfernte Schnittstelle deklarieren	1151
14.3.2	Remote-Objekt-Implementierung	1152
14.3.3	Stellvertreterobjekte	1153
14.3.4	Der Namensdienst (Registry)	1153
14.3.5	Remote-Objekt-Implementierung exportieren und beim Namensdienst anmelden	1155
14.3.6	Einfaches Logging	1157
14.3.7	Aufräumen mit dem DGC *	1158
14.4	Auf der Client-Seite	1159
14.5	Entfernte Objekte übergeben und laden	1160
14.5.1	Klassen vom RMI-Klassenspeicher nachladen	1160
14.6	Weitere Eigenschaften von RMI	1161
14.6.1	RMI und CORBA	1161
14.6.2	RMI über HTTP getunnelt	1161
14.6.3	Automatische Remote-Objekt-Aktivierung	1162
14.7	Java Message Service (JMS)	1163
14.8	Zum Weiterlesen	1163

15 RESTful und SOAP-Web-Services 1165

15.1	Web-Services	1165
15.2	RESTful Web-Services	1166
15.2.1	Aus Prinzip REST	1166
15.2.2	Jersey	1168
15.2.3	JAX-RS-Annotationen für den ersten REST-Service	1169
15.2.4	Test-Server starten	1169
15.2.5	REST-Services konsumieren	1170
15.2.6	Content-Handler, Marshaller und verschiedene MIME-Typen	1171
15.2.7	REST-Parameter	1174

15.2.8	REST-Services mit Parametern über die Jersey-Client-API aufrufen	1176
15.2.9	PUT-Anforderungen und das Senden von Daten	1177
15.2.10	PUT/POST/DELETE-Sendungen mit der Jersey-Client-API absetzen	1177
15.3	Daily Soap und das SOAP-Protokoll	1177
15.3.1	Die technische Realisierung	1178
15.3.2	Web-Service-APIs und Implementierungen	1179
15.3.3	@WebService	1179
15.3.4	Einen Web-Service definieren	1180
15.3.5	Web-Services veröffentlichen	1181
15.3.6	Einen JAX-WS-Client implementieren	1181
15.4	Zum Weiterlesen	1183
16	Technologien für die Infrastruktur	1185
<hr/>		
16.1	Property-Validierung durch Bean Validation	1185
16.1.1	Technische Abhängigkeiten und POJOs	1195
16.2	Wie eine Implementierung an die richtige Stelle kommt	1197
16.2.1	Arbeiten mit dem ServiceLoader	1198
16.2.2	Die Utility-Klasse Lookup als ServiceLoader-Fassade	1199
16.2.3	Contexts and Dependency Injection (CDI) aus dem JSR-299	1200
16.3	Zum Weiterlesen	1202
17	Typen, Reflection und Annotationen	1203
<hr/>		
17.1	Metadaten	1203
17.1.1	Metadaten durch Javadoc-Tags	1203
17.2	Metadaten der Typen mit dem Class-Objekt	1204
17.2.1	An ein Class-Objekt kommen	1204
17.2.2	Eine Class ist ein Type	1206
17.3	Klassenlader	1207
17.3.1	Das Verzeichnis jre/lib/endorsed *	1207
17.3.2	Die Klasse java.lang.ClassLoader	1208
17.3.3	Hot Deployment mit dem URL-Classloader *	1209
17.4	Metadaten der Typen mit dem Class-Objekt	1212
17.4.1	Der Name des Typs	1212
17.4.2	Was das Class-Objekt beschreibt *	1215

17.4.3	instanceof mit Class-Objekten *	1217
17.4.4	Oberklassen finden *	1218
17.4.5	Implementierte Interfaces einer Klasse oder eines Interfaces *	1219
17.4.6	Modifizierer und die Klasse Modifier *	1219
17.4.7	Die Arbeit auf dem Feld *	1221
17.5	Attribute, Methoden und Konstruktoren	1222
17.5.1	Reflections – Gespür für die Attribute einer Klasse	1224
17.5.2	Schnittstelle Member für Eigenschaften	1225
17.5.3	Field-Klasse	1226
17.5.4	Methoden einer Klasse erfragen	1227
17.5.5	Properties einer Bean erfragen	1230
17.5.6	Konstruktoren einer Klasse	1231
17.5.7	Annotationen	1232
17.6	Objekte erzeugen und manipulieren	1232
17.6.1	Objekte erzeugen	1232
17.6.2	Die Belegung der Variablen erfragen	1234
17.6.3	Eine generische eigene toString()-Methode *	1236
17.6.4	Variablen setzen	1238
17.6.5	Bean-Zustände kopieren *	1240
17.6.6	Private Attribute ändern	1240
17.6.7	Methoden aufrufen	1241
17.6.8	Statische Methoden aufrufen	1243
17.6.9	Dynamische Methodenaufrufe bei festen Methoden beschleunigen *	1243
17.6.10	java.lang.reflect.Parameter	1245
17.7	Eigene Annotationstypen *	1247
17.7.1	Annotationen zum Laden von Ressourcen	1247
17.7.2	Neue Annotationen deklarieren	1248
17.7.3	Annotationen mit genau einem Attribut	1248
17.7.4	Element-Wert-Paare (Attribute) hinzufügen	1249
17.7.5	Annotationsattribute vom Typ einer Aufzählung	1250
17.7.6	Felder von Annotationsattributen	1251
17.7.7	Vorbelegte Attribute	1252
17.7.8	Annotieren von Annotationstypen	1253
17.7.9	Deklarationen für unsere Ressourcen-Annotationen	1259
17.7.10	Annotierte Elemente auslesen	1260
17.7.11	Auf die Annotationsattribute zugreifen	1262
17.7.12	Komplettbeispiel zum Initialisieren von Ressourcen	1263
17.7.13	Mögliche Nachteile von Annotationen	1266
17.8	Zum Weiterlesen	1267

18 Dynamische Übersetzung und Skriptsprachen

1269

18.1	Codegenerierung	1270
18.1.1	Generierung von Java-Quellcode	1271
18.1.2	Codetransformationen	1273
18.1.3	Erstellen von Java-Bytecode	1273
18.2	Programme mit der Java Compiler API übersetzen	1274
18.2.1	Java Compiler API	1274
18.2.2	Fehlerdiagnose	1276
18.2.3	Eine im String angegebene Kompilationseinheit übersetzen	1279
18.2.4	Wenn Quelle und Ziel der Speicher sind	1280
18.3	Ausführen von Skripten	1283
18.3.1	Java-Programme mit JavaScript schreiben	1284
18.3.2	Kommandozeilenprogramme jrunscript und jjs	1284
18.3.3	javax.script-API	1285
18.3.4	JavaScript-Programme ausführen	1285
18.3.5	Alternative Sprachen für die JVM	1286
18.3.6	Von den Schwierigkeiten, dynamische Programmiersprachen auf die JVM zu bringen *	1288
18.4	Zum Weiterlesen	1292

19 Logging und Monitoring

1293

19.1	Logging mit Java	1293
19.1.1	Logging-APIs	1293
19.1.2	Logging mit java.util.logging	1294
19.1.3	Logging mit log4j *	1300
19.1.4	Die Simple Logging Facade	1303
19.1.5	Aktuelle Entwicklungen der Java-Logging-APIs	1303
19.2	Systemzustände überwachen	1303
19.3	MBean-Typen, MBean-Server und weitere Begriffe	1304
19.3.1	MXBeans des Systems	1305
19.4	Geschwätzige Programme und JConsole	1306
19.4.1	JConsole	1307
19.5	Der MBeanServer	1308

19.6	Eine eigene Standard-MBean	1309
19.6.1	Management-Schnittstelle	1309
19.6.2	Implementierung der Managed-Ressource	1310
19.6.3	Anmeldung beim Server	1311
19.6.4	Eine eigene Bean in JConsole einbringen	1311
19.7	Zum Weiterlesen	1313

20 Sicherheitskonzepte 1315

20.1	Zentrale Elemente der Java-Sicherheit	1315
20.1.1	Sichere Java Virtual Machine	1315
20.1.2	Der Sandkasten (Sandbox)	1315
20.1.3	Security-API der Java SE	1316
20.1.4	Cryptographic Service Providers	1317
20.2	Sicherheitsmanager (Security-Manager)	1318
20.2.1	Der Sicherheitsmanager bei Applets	1318
20.2.2	Sicherheitsmanager aktivieren	1319
20.2.3	Rechte durch Policy-Dateien vergeben	1321
20.2.4	Erstellen von Rechedateien mit dem grafischen Policy-Tool	1323
20.2.5	Kritik an den Policies	1324
20.3	Signierung	1326
20.3.1	Warum signieren?	1326
20.3.2	Digitale Ausweise und die Zertifizierungsstelle	1326
20.3.3	Mit keytool Schlüssel erzeugen	1327
20.3.4	Signieren mit jarsigner	1328
20.4	Kryptografische Hashfunktion	1328
20.4.1	Die MDx-Reihe	1329
20.4.2	Secure Hash Algorithm (SHA)	1329
20.4.3	Mit der Security-API einen Fingerabdruck berechnen	1330
20.4.4	Die Klasse MessageDigest	1330
20.5	Verschlüsseln von Daten(-strömen) *	1333
20.5.1	Den Schlüssel, bitte	1333
20.5.2	Verschlüsseln mit Cipher	1334
20.5.3	Verschlüsseln von Datenströmen	1335
20.6	Zum Weiterlesen	1336

21.1	Relationale Datenbanken	1337
21.1.1	Das relationale Modell	1337
21.2	Datenbanken und Tools	1338
21.2.1	HSQLDB	1338
21.2.2	Weitere Datenbanken *	1340
21.2.3	Eclipse Data Tools Platform (DTP) zum Durchschauen von Datenbanken	1341
21.3	JDBC und Datenbanktreiber	1343
21.3.1	Treibertypen *	1344
21.3.2	JDBC-Versionen *	1345
21.4	Eine Beispielabfrage	1346
21.4.1	Schritte zur Datenbankabfrage	1346
21.4.2	Ein Client für die HSQLDB-Datenbank	1346
21.5	Mit Java an eine Datenbank andocken	1348
21.5.1	Der Treiber-Manager *	1348
21.5.2	Den Treiber laden	1349
21.5.3	Eine Aufzählung aller Treiber *	1350
21.5.4	Log-Informationen *	1351
21.5.5	Verbindung zur Datenbank auf- und abbauen	1351
21.6	Datenbankabfragen	1354
21.6.1	Abfragen über das Statement-Objekt	1354
21.6.2	Ergebnisse einer Abfrage in ResultSet	1357
21.6.3	Java und SQL-Datentypen	1358
21.6.4	Date, Time und Timestamp	1361
21.6.5	Unicode in der Spalte korrekt auslesen	1363
21.6.6	Eine SQL-NULL und wasNull() bei ResultSet	1363
21.6.7	Wie viele Zeilen hat ein ResultSet? *	1364
21.7	Elemente einer Datenbank ändern	1364
21.7.1	Einzelne INSERT-, UPDATE- oder DELETE-Anweisungen senden	1364
21.7.2	Aktualisierbares ResultSet	1365
21.7.3	Batch-Updates	1366
21.8	Die Ausnahmen bei JDBC, SQLException und Unterklassen	1367
21.8.1	JDBC-Fehlerbasisklasse SQLException	1367
21.8.2	SQLWarning	1368
21.9	ResultSet und RowSet *	1370
21.9.1	Die Schnittstelle RowSet	1370
21.9.2	Implementierungen von RowSet	1370

21.9.3	Der Typ <code>CachedRowSet</code>	1371
21.9.4	Der Typ <code>WebRowSet</code>	1372
21.10	Vorbereitete Anweisungen (Prepared Statements)	1375
21.10.1	<code>PreparedStatement</code> -Objekte vorbereiten	1375
21.10.2	Werte für die Platzhalter eines <code>PreparedStatement</code>	1376
21.11	Transaktionen	1377
21.12	Vorbereitete Datenbankverbindungen	1378
21.12.1	<code>DataSource</code>	1378
21.12.2	Gepoolte Datenbankverbindungen	1381
21.13	Zum Weiterlesen	1382

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	1385
22.3.1	Die Header-Datei erzeugen	1386
22.3.2	Implementierung der Funktion in C	1387
22.3.3	Die C-Programme übersetzen und die dynamische Bibliothek erzeugen	1388
22.4	Nativ die String-Länge ermitteln	1390
22.5	Erweiterte JNI-Eigenschaften	1392
22.5.1	Klassendefinitionen	1392
22.5.2	Zugriff auf Attribute	1392
22.5.3	Methoden aufrufen	1395
22.5.4	Threads und Synchronisation	1396
22.5.5	@Native Markierungen *	1396
22.6	Einfache Anbindung von existierenden Bibliotheken	1397
22.6.1	JNA (Java Native Access)	1397
22.6.2	BridJ	1397
22.6.3	Generieren von JNI-Wrappern aus C++-Klassen und C-Headern	1398
22.6.4	COM-Schnittstellen anzapfen	1398
22.7	Invocation-API	1398
22.8	Zum Weiterlesen	1399

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	1404
23.2.6	VisualVM	1406
23.3	Programmieren mit der Tools-API	1411
23.3.1	Java-Tools in Java implementiert	1411
23.3.2	Tools aus eigenen Java-Programmen ansprechen	1411
23.3.3	API-Dokumentation der Tools	1412
23.3.4	Eigene Doclets	1412
23.3.5	Auf den Compiler-AST einer Klasse zugreifen	1414
23.4	Ant	1416
23.4.1	Bezug und Installation von Ant	1416
23.4.2	Das Build-Skript build.xml	1417
23.4.3	Build den Build	1418
23.4.4	Properties	1418
23.4.5	Externe und vordefinierte Properties	1419
23.4.6	Weitere Ant-Tasks	1420
23.5	Disassembler, Decompiler und Obfuscator	1421
23.5.1	Der Disassembler javap *	1422
23.5.2	Decompiler	1426
23.5.3	Obfuscatoren	1428
23.6	Weitere Dienstprogramme	1430
23.6.1	Sourcecode Beautifier	1430
23.6.2	Java-Programme als Systemdienst ausführen	1431
23.7	Zum Weiterlesen	1431
	Index	1433