

Inhalt

Vorwort 33

Vorwort Version 2.0 42

1	Schon wieder eine neue Sprache? 43
1.1	Der erste Kontakt 43
1.2	Historischer Hintergrund 43
1.3	Eigenschaften von Java 44
1.3.1	Die virtuelle Maschine 45
1.3.2	Kein Präprozessor 46
1.3.3	Überladene Operatoren 47
1.3.4	Zeiger und Referenzen 47
1.3.5	Garbage-Collector 48
1.3.6	Ausnahmenbehandlung 49
1.3.7	Objektorientierung in Java 49
1.3.8	Java-Security-Model 49
1.4	Java im Vergleich zu anderen Sprachen 50
1.4.1	Java und JavaScript 51
1.4.2	Normierungsversuche 51
1.5	Die Rolle von Java im Web 51
1.6	Aufkommen von Stand-alone-Applikationen 52
1.7	Entwicklungs- und Laufzeitumgebungen 52
1.7.1	Aller Anfang mit dem Java SDK 52
1.7.2	Die Entwicklungsumgebung von Sun: Sun ONE Studie (früher Forte) und NetBeans 53
1.7.3	Umgebungen von IBM 53
1.7.4	TogetherJ 54
1.7.5	JBuilder von Borland 54
1.7.6	Die virtuelle Maschine Kaffe von Transvirtual Technologies 55
1.7.7	Die Entwicklungsumgebung CodeGuide 55
1.7.8	Ein Wort zu Microsoft, Java und zu J++ 56
1.7.9	Direkt ausführbare Programme für Windows compilieren 57
1.8	Installationsanleitung für das Java 2 SDK unter Microsoft Windows 57
1.8.1	Das Java 2 SDK beziehen 58
1.8.2	Java SDK installieren 58
1.8.3	Compiler und Interpreter nutzen 58
1.9	Das erste Programm compilieren und testen 59
1.9.1	Häufige Compiler- und Interpreterprobleme 61

2	Sprachbeschreibung 63
2.1	Anweisungen und Programme 63
2.2	Programme 65
2.2.1	Kommentare 67
2.2.2	Funktionsaufrufe als Anweisungen 68
2.2.3	Die leere Anweisung 69
2.2.4	Der Block 70
2.3	Elemente einer Programmiersprache 70
2.3.1	Textkodierung durch Unicode-Zeichen 70
2.3.2	Unicode-Tabellen unter Windows 72
2.3.3	Bezeichner 72
2.3.4	Reservierte Schlüsselwörter 73
2.3.5	Token 74
2.3.6	Semantik 75
2.4	Datentypen 75
2.4.1	Primitive Datentypen 76
2.4.2	Wahrheitswerte 76
2.4.3	Variablen Deklarationen 77
2.4.4	Ganzzahlige Datentypen 78
2.4.5	Die Fließkommazahlen 79
2.4.6	Zeichen 80
2.4.7	Die Typanpassung (das Casting) 81
2.4.8	Lokale Variablen, Blöcke und Sichtbarkeit 84
2.4.9	Initialisierung von lokalen Variablen 85
2.5	Ausdrücke 86
2.5.1	Zuweisungsoperator und Verbundoperator 87
2.5.2	Präfix- oder Postfix-Inkrement und -Dekrement 88
2.5.3	Unäres Minus und Plus 89
2.5.4	Arithmetische Operatoren 90
2.5.5	Die relationalen Operatoren 93
2.5.6	Logische Operatoren 93
2.5.7	Reihenfolge und Rang der Operatoren in der Auswertungsreihenfolge 94
2.5.8	Was C(++)-Programmierer vermissen könnten 96
2.6	Bedingte Anweisungen oder Fallunterscheidungen 96
2.6.1	Die if-Anweisung 96
2.6.2	Die Alternative wählen mit einer if/else-Anweisung 98
2.6.3	Die switch-Anweisung bietet die Alternative 101
2.7	Schleifen 103
2.7.1	Die while-Schleife 103
2.7.2	Schleifenbedingungen und Vergleiche mit == 104
2.7.3	Die do/while-Schleife 106
2.7.4	Die for-Schleife 107
2.7.5	Ausbruch planen mit break und Wiedereinstieg mit continue 110
2.7.6	break und continue mit Sprungmarken 111

2.8	Methoden einer Klasse	112
2.8.1	Bestandteil einer Funktion	112
2.8.2	Aufruf	114
2.8.3	Methoden ohne Parameter	115
2.8.4	Statische Methoden (Klassenmethoden)	115
2.8.5	Parameter und Wertübergabe	116
2.8.6	Methoden vorzeitig mit return beenden	117
2.8.7	Nicht erreichbarer Quellcode bei Funktionen	118
2.8.8	Rückgabewerte	118
2.8.9	Methoden überladen	122
2.8.10	Vorinitialisierte Parameter bei Funktionen	124
2.8.11	Finale lokale Variablen	124
2.8.12	Finale Referenzen in Objekten und das fehlende const	125
2.8.13	Rekursive Funktionen	126
2.8.14	Die Ackermann-Funktion	128
2.8.15	Die Türme von Hanoi	130
2.9	Weitere Operatoren	132
2.9.1	Bitoperationen	132
2.9.2	Vorzeichenlose Bytes in ein Integer und Char konvertieren	133
2.9.3	Variablen mit Xor vertauschen	134
2.9.4	Die Verschiebeoperatoren	135
2.9.5	Setzen, Löschen, Umdrehen und Testen von Bits	137
2.9.6	Der Bedingungsoperator	138
2.9.7	Überladenes Plus für Strings	140
2.10	Einfache Benutzereingaben	141

3 Klassen und Objekte 143

3.1	Objektorientierte Programmierung	143
3.1.1	Warum überhaupt OOP?	143
3.1.2	Modularität und Wiederverwertbarkeit	144
3.2	Klassen benutzen	144
3.2.1	Die Klasse Point	145
3.2.2	Etwas über die UML	146
3.2.3	Anlegen eines Exemplars einer Klasse	147
3.2.4	Zugriff auf Variablen und Methoden mit dem Punkt	149
3.2.5	Konstruktoren	150
3.2.6	Die null-Referenz	151
3.3	Mit Referenzen arbeiten	152
3.3.1	Zuweisungen bei Referenzen	152
3.3.2	Funktionen mit nichtprimitiven Parametern	153
3.3.3	Gleichheit von Objekten und die Methode equals()	154
3.4	Arrays	156
3.4.1	Deklaration von Arrays	156
3.4.2	Arrays mit Inhalt	157
3.4.3	Die Länge eines Arrays mit length	158

3.4.4	Zugriff auf die Elemente	159
3.4.5	Array-Objekte erzeugen	160
3.4.6	Fehler bei Arrays	161
3.4.7	Arrays mit nichtprimitiven Elementen	162
3.4.8	Arrays und Objekte	162
3.4.9	Initialisierte Array-Objekte	163
3.4.10	Mehrdimensionale Arrays	164
3.4.11	Die Wahrheit über die Array-Initialisierung	166
3.4.12	Arrays kopieren und füllen	167
3.4.13	Mehrere Rückgabeparameter	168
3.4.14	Parameter per Referenz übergeben	169
3.4.15	Der Einstiegspunkt für das Laufzeitsystem	169
3.4.16	Der Rückgabewert von main()	170
3.4.17	Die Klasse Arrays	170

4 Der Umgang mit Zeichenketten 173

4.1	Strings und deren Anwendung	173
4.1.1	String-Objekte für konstante Zeichenketten	173
4.1.2	String-Länge	175
4.1.3	Gut, dass wir verglichen haben	175
4.1.4	Stringteile extrahieren	179
4.1.5	Veränderte Strings liefern	181
4.1.6	Typen in Zeichenketten konvertieren	183
4.2	Veränderbare Zeichenketten mit der Klasse StringBuffer	184
4.2.1	Anlegen von StringBuffer-Objekten	185
4.2.2	Die Länge eines StringBuffer-Objekts lesen und setzen	186
4.2.3	Daten anhängen	186
4.2.4	Zeichen(folgen) setzen, erfragen, löschen und umdrehen	186
4.3	Vergleiche von Zeichenketten als String und StringBuffer	187
4.3.1	Sollte es ein equals() und hash() bei StringBuffer geben?	188
4.4	Ein paar kleine Helfer	189
4.4.1	Strings einer gegebenen Länge erzeugen und rechtsbündig ausgeben	189
4.4.2	Teile im String ersetzen	190
4.5	Zeichenkodierungen umwandeln	191
4.6	Sprachabhängiges Vergleichen mit der Collator-Klasse	191
4.6.1	Effiziente interne Speicherung für die Sortierung	194
4.7	Die Klasse StringTokenizer	195
4.8	StreamTokenizer	197
4.9	Formatieren mit Format-Objekten	200
4.9.1	Prozente, Zahlen und Währungen ausgeben	202
4.9.2	Ausgaben formatieren	202
4.9.3	Dezimalzahlformatierung	205

4.10	Reguläre Ausdrücke	206
4.10.1	Splitten von Zeichenketten	207
4.10.2	split() in String	209
4.10.3	Das Paket gnu-regexp	209
4.11	Überprüfung der E-Mail-Adressen und Kreditkarteninformationen	209
4.11.1	Gültige E-Mail-Adressen	209
4.11.2	Kreditkartennummern testen	211

5 **Mathematisches** 215

5.1	Arithmetik in Java	215
5.1.1	Soll eine Division durch Null zur Übersetzungszeit erkannt werden?	216
5.2	Die Funktionen der Math-Klasse	217
5.2.1	Attribute	217
5.2.2	Winkelfunktionen (trigonometrische Funktionen und Arcus-Funktionen)	218
5.2.3	Runden von Werten	218
5.2.4	Exponentialfunktionen	220
5.2.5	Division	220
5.2.6	Absolutwerte und Maximum, Minimum	221
5.2.7	Zufallszahlen	221
5.3	Mathe bitte strikt	222
5.3.1	Strikt Fließkomma mit strictfp	222
5.3.2	Die Klassen Math und StrictMath	222
5.4	Die Random-Klasse	223
5.5	Große Zahlen	224
5.5.1	Die Klasse BigInteger	224
5.5.2	Ganz lange Fakultäten	227
5.6	Probleme mit Java und der Mathematik	228
5.7	Das Java-Matrix-Paket Jama	228

6 **Eigene Klassen schreiben** 231

6.1	Eigene Klassen definieren	231
6.1.1	Methodenaufrufe und Nebeneffekte	232
6.1.2	Argumentübergabe mit Referenzen	233
6.1.3	Die this-Referenz	234
6.1.4	Überdeckte Objektvariablen nutzen	235
6.2	Assoziationen zwischen Objekten	236
6.3	Privatsphäre und Sichtbarkeit	237
6.3.1	Wieso nicht freie Methoden und Variablen für alle?	238
6.3.2	Privat ist nicht ganz privat. Es kommt darauf an, wer's sieht	239
6.3.3	Zugriffsmethoden für Attribute definieren	241
6.3.4	Zusammenfassung zur Sichtbarkeit	242

6.4	Statische Methoden und Variablen	243
6.4.1	Warum statische Eigenschaften sinnvoll sind	243
6.4.2	Statische Eigenschaften mit static	243
6.4.3	Statische Eigenschaften als Objekteigenschaften nutzen	244
6.4.4	Statische Eigenschaften und Objekteigenschaften	245
6.4.5	Statische Variablen zum Datenaustausch	245
6.4.6	Warum die Groß- und Kleinschreibung wichtig ist	246
6.4.7	Konstanten mit dem Schlüsselwort final bei Variablen	247
6.4.8	Typsicherere Konstanten	248
6.4.9	Statische Blöcke	249
6.5	Objekte anlegen und zerstören	250
6.5.1	Konstruktoren schreiben	250
6.5.2	Einen anderen Konstruktor der gleichen Klasse aufrufen	253
6.5.3	Initialisierung der Objekt- und Klassenvariablen	255
6.5.4	Finale Werte im Konstruktor setzen	257
6.5.5	Exemplarinitialisierer (Instanzinitialisierer)	257
6.5.6	Zerstörung eines Objekts durch den Müllauflämmmer	258
6.5.7	Implizit erzeugte Stringobjekte	259
6.5.8	Zusammenfassung: Konstruktoren und Methoden	260
6.6	Veraltete (deprecated) Methoden/Konstruktoren	261
6.7	Vererbung	262
6.7.1	Vererbung in Java	262
6.7.2	Einfach- und Mehrfachvererbung	263
6.7.3	Kleidungsstücke modelliert	263
6.7.4	Sichtbarkeit	265
6.7.5	Das Substitutionsprinzip	265
6.7.6	Automatische und Explizite Typanpassung	266
6.7.7	Finale Klassen	267
6.7.8	Unterklassen prüfen mit dem Operator instanceof	267
6.8	Methoden überschreiben	268
6.8.1	super: Aufrufen einer Methode aus der Oberklasse	269
6.8.2	Nicht überschreibbare Funktionen	271
6.8.3	Fehlende kovariante Rückgabewerte	272
6.9	Die oberste aller Klassen: Object	272
6.9.1	Klassenobjekte	273
6.9.2	Hashcodes	273
6.9.3	Objektidentifikation mit <code>toString()</code>	273
6.9.4	Objektgleichheit mit <code>equals()</code> und Identität	274
6.9.5	Klonen eines Objekts mit <code>clone()</code>	276
6.9.6	Aufräumen mit <code>finalize()</code>	277
6.9.7	Synchronisation	277
6.10	Die Oberklasse gibt Funktionalität vor	278
6.10.1	Dynamisches Binden als Beispiel für Polymorphie	279
6.10.2	Keine Polymorphie bei privaten, statischen und finalen Methoden	281
6.10.3	Konstruktoren in der Vererbung	282

6.11	Abstrakte Klassen	286
6.11.1	Abstrakte Klassen	286
6.11.2	Abstrakte Methoden	287
6.11.3	Über abstract final	290
6.12	Schnittstellen	290
6.12.1	Die Mehrfachvererbung bei Schnittstellen	294
6.12.2	Erweitern von Interfaces – Subinterfaces	295
6.12.3	Vererbte Konstanten bei Schnittstellen	295
6.12.4	Vordefinierte Methoden einer Schnittstelle	297
6.12.5	CharSequence als Beispiel einer Schnittstelle	298
6.13	Innere Klassen	300
6.13.1	Geschachtelte Top-Level-Klassen und Schnittstellen	301
6.13.2	Mitglieds- oder Elementklassen	301
6.13.3	Lokale Klassen	304
6.13.4	Anonyme innere Klassen	305
6.13.5	Eine Sich-Selbst-Implementierung	308
6.13.6	this und Vererbung	309
6.13.7	Implementierung einer verketteten Liste	310
6.13.8	Funktionszeiger	312
6.14	Gegenseitige Abhängigkeiten von Klassen	314
6.15	Pakete	314

7 Exceptions 317

7.1	Problembereiche einzäunen	317
7.1.1	Exceptions in Java mit try und catch	317
7.1.2	Ablauf einer Ausnahmesituation	319
7.1.3	Wiederholung kritischer Bereiche	319
7.1.4	throws im Methodenkopf angeben	320
7.1.5	Abschließende Arbeiten mit finally	321
7.1.6	Nicht erreichbare catch-Klauseln	322
7.2	Die Klassenhierarchie der Fehler	323
7.2.1	Die Exception-Hierarchie	324
7.2.2	Ober-Ausnahmen fangen	324
7.2.3	Alles geht als Exception durch	325
7.2.4	Ausnahmen, die nicht gefangen werden müssen: RuntimeException	326
7.3	Werfen eigener Exceptions	327
7.3.1	Vorgefertigte Ausnahme-Objekte wiederverwenden	328
7.3.2	Typecast auf ein null-Objekt für eine NullPointerException	329
7.3.3	Neue Exception-Klassen definieren	329
7.4	Rückgabewerte bei ausgelösten Ausnahmen	331
7.5	Stack-Aufruf analysieren	332
7.6	Assertions	333
7.7	Sicherheitsfragen mit dem SecurityManager klären	335
7.7.1	Programm beenden	335

8	Die Funktionsbibliothek 337
8.1	Die Java-Klassenphilosophie 337
8.1.1	Übersicht über die Pakete der Standardbibliothek 337
8.2	Wrapper-Klassen 342
8.2.1	Die Character-Klasse 343
8.2.2	Die Boolean-Klasse 345
8.2.3	Die Number-Klasse 347
8.2.4	Die Klasse Integer 348
8.2.5	Behandlung von Überlauf 350
8.2.6	Unterschiedliche Ausgabeformate 351
8.2.7	Boxing und Unboxing 352
8.3	Ausführung von externen Programmen 353
8.3.1	DOS-Programme aufrufen 354
8.3.2	Die Windows Registry verwenden 356
8.4	Kompilieren von Klassen 357
8.4.1	Der Sun-Compiler 357
 9	 Threads und nebenläufige Programmierung 359
9.1	Prozesse und Threads 359
9.1.1	Wie parallele Programme die Geschwindigkeit heben können 360
9.2	Threads erzeugen 362
9.2.1	Threads über die Schnittstelle Runnable implementieren 362
9.2.2	Threads über Runnable starten 363
9.2.3	Die Klasse Thread erweitern 364
9.2.4	Erweitern von Thread oder implementieren von Runnable? 367
9.3	Threads schlafen 367
9.3.1	Eine Zeituhr 368
9.4	Die Klassen Timer und TimerTask 370
9.5	Die Zustände eines Threads 371
9.5.1	Das Ende eines Threads 371
9.5.2	Einen Thread höflich mit Interrupt beenden 371
9.5.3	Der stop() von außen 373
9.5.4	Das ThreadDeath-Objekt 374
9.5.5	Auf das Ende warten mit join() 375
9.6	Arbeit niederlegen und wieder aufnehmen 377
9.7	Priorität 377
9.7.1	Threads hoher Priorität und das AWT 378
9.7.2	Granularität und Vorrang 378
9.8	Dämonen (engl. Daemon) 379
9.9	Kooperative und nicht kooperative Threads 380
9.10	Synchronisation über kritische Abschnitte 381
9.10.1	Gemeinsam genutzte Daten 382
9.10.2	Probleme beim gemeinsamen Zugriff und kritische Abschnitte 382

9.10.3	Punkte parallel initialisieren	383
9.10.4	i++ sieht atomar aus, ist es aber nicht	383
9.10.5	Abschnitte mit synchronized schützen	384
9.10.6	Monitore	385
9.10.7	Synchronized-Methode am Beispiel der Klasse StringBuffer	386
9.10.8	Synchronisierte Blöcke	386
9.10.9	Vor- und Nachteile von synchronisierten Blöcken und Methoden	387
9.10.10	Nachträglich synchronisieren	388
9.10.11	Monitore sind reentrant, gut für die Geschwindigkeit	388
9.10.12	Deadlocks	389
9.11	Variablen mit volatile kennzeichnen	391
9.12	Synchronisation über Warten und Benachrichtigen	393
9.12.1	Warten mit wait() und Aufwecken mit notify()	394
9.12.2	Mehrere Wartende und notifyAll()	396
9.12.3	wait() mit einer Zeitspanne	396
9.12.4	Beispiel Erzeuger-Verbraucher-Programm	396
9.12.5	Semaphoren	399
9.13	Grenzen von Threads	401
9.14	Aktive Threads in der Umgebung	401
9.15	Gruppen von Threads in einer Thread-Gruppe	402
9.15.1	Etwas über die aktuelle Thread-Gruppe herausfinden	403
9.15.2	Threads in einer Thread-Gruppe anlegen	405
9.15.3	Methoden von Thread und ThreadGroup im Vergleich	407
9.16	Einen Abbruch der virtuellen Maschine erkennen	408

10	Raum und Zeit	411
10.1	Greenwich Mean Time (GMT)	411
10.2	Wichtige Datum-Klassen im Überblick	413
10.3	Zeitzonen und Sprachen der Länder	413
10.3.1	Zeitzonen durch die Klasse TimeZone repräsentieren	413
10.4	Sprachen der Länder	415
10.4.1	Sprachen in Java über Locale-Objekte	415
10.5	Einfache Übersetzung durch ResourceBundle-Objekte	418
10.6	Die Klasse Date	420
10.6.1	Objekte erzeugen und Methoden nutzen	420
10.6.2	Zeitmessung und Profiling	422
10.7	Calendar und GregorianCalendar	423
10.7.1	Die abstrakte Klasse Calendar	423
10.7.2	Der gregorianische Kalender	425
10.8	Formatieren der Datumsangaben	431
10.8.1	Mit DateFormat und SimpleDateFormat formatieren	431
10.8.2	Parsen von Datumswerten	438
10.8.3	Parsen und Formatieren ab bestimmten Positionen	440

11	Datenstrukturen und Algorithmen 441
11.1	Mit einem Iterator durch die Daten wandern 441
11.1.1	Bauernregeln aufzählen 442
11.2	Dynamische Datenstrukturen 444
11.3	Die Klasse Vector 444
11.3.1	Vektoren erzeugen 444
11.3.2	Funktionen 445
11.3.3	Arbeitsweise des internen Arrays 448
11.3.4	Die Größe eines Felds 449
11.3.5	Eine Aufzählung und gleichzeitiges Verändern 449
11.4	Stack, der Stapel 451
11.4.1	Die Methoden von Stack 451
11.4.2	Ein Stack ist ein Vektor – aha! 452
11.5	Die Klasse Hashtable und assoziative Speicher 452
11.5.1	Ein Objekt der Klasse Hashtable erzeugen 452
11.5.2	Einfügen und Abfragen der Datenstruktur 453
11.5.3	Die Arbeitsweise einer Hashtabelle 455
11.5.4	Aufzählen der Elemente 457
11.5.5	Ausgabe der Hashtabelle und Gleichheitstest 457
11.5.6	Klonen 458
11.6	Die abstrakte Klasse Dictionary 458
11.6.1	Zugriff und Abfrage 458
11.6.2	Metainformationen 459
11.6.3	Iterationen über die Elemente 459
11.7	Die Properties-Klasse 460
11.7.1	Über die Klasse Properties 460
11.7.2	put(), get() und getProperties() 462
11.7.3	Eigenschaften ausgeben 462
11.7.4	Systemeigenschaften der Java-Umgebung 463
11.7.5	Browser-Version abfragen 464
11.7.6	Properties von der Konsole aus setzen 464
11.8	Windows-typische INI-Dateien 466
11.9	Queue, die Schlange 466
11.10	Die Collection-API 467
11.10.1	Die Schnittstelle Collection 468
11.10.2	Schnittstellen, die Collection erweitern, und Map 469
11.10.3	Abstrakte Basisklassen für Container 471
11.10.4	Konkrete Container-Klassen 472
11.10.5	Unterschiede zu den älteren Datenstrukturen und die Synchronisation 472
11.10.6	Das erste Programm mit Container-Klassen 473
11.10.7	Iteratoren 474
11.10.8	Der Comparator 476
11.10.9	toArray() von Collection verstehen – Chance für eine Falle erkennen 478

11.11	Listen 480
11.11.1	AbstractList 481
11.11.2	Optionale Methoden 482
11.11.3	ArrayList 485
11.11.4	LinkedList 486
11.12	Algorithmen 486
11.12.1	Datenmanipulation 488 ✓
11.12.2	Größten und kleinsten Wert einer Collection finden 489
11.12.3	Sortieren 490
11.12.4	Elemente in der Collection suchen 493
11.13	Typsichere Datenstrukturen 494
11.14	Die Klasse BitSet für Bitmengen 495
11.14.1	Ein BitSet anlegen und füllen 495
11.14.2	Mengenorientierte Operationen 497
11.14.3	Funktionsübersicht 497
11.14.4	Primzahlen in einem BitSet verwalten 498
11.15	Ein Design-Pattern durch Beobachten von Änderungen 499
11.15.1	Design-Pattern 499
11.15.2	Das Beobachter-Pattern (Observer/Observable) 500

12	Datenströme und Dateien 505
12.1	Dateien und Verzeichnisse 505
12.1.1	Dateien und Verzeichnisse mit der Klasse File 506
12.1.2	Dateieigenschaften und -attribute 506
12.1.3	Umbenennen, Verzeichnisse anlegen und Datei löschen 509
12.1.4	Die Wurzel aller Verzeichnisse 509
12.1.5	Verzeichnisse listen und Dateien filtern 511
12.1.6	Implementierungsmöglichkeiten für die Klasse File 515
12.1.7	Verzeichnisse nach Dateien rekursiv durchsuchen 516
12.2	Dateien mit wahlfreiem Zugriff 518
12.2.1	Ein RandomAccessFile öffnen 518
12.2.2	Aus dem RandomAccessFile lesen 519
12.2.3	Hin und her in der Datei 520
12.2.4	Die Länge des RandomAccessFile 521
12.3	Übersicht über wichtige Stream- und WriterReader 522
12.3.1	Die abstrakten Basisklassen 524
12.4	Eingabe- und Ausgabe-Klassen: InputStream und OutputStream 524
12.4.1	Die Klasse OutputStream 524
12.4.2	Ein Datenschlucker 526
12.4.3	Die Eingabeklasse InputStream 526
12.4.4	Anwenden der Klasse FileInputStream 527
12.4.5	Anwendung der Klasse FileOutputStream 529
12.4.6	Kopieren von Dateien 530
12.4.7	Daten filtern durch FilterInputStream und FilterOutputStream 531
12.4.8	Der besondere Filter PrintStream 532

12.4.9	System.in und System.out	534
12.4.10	Bytes in den Strom mit ByteArrayOutputStream	538
12.4.11	Ströme zusammensetzen mit SequenceInputStream	539
12.5	Ressourcen wie Grafiken aus dem Klassenpfad und aus Jar-Archiven laden	542
12.6	Die Unterklassen von Writer	543
12.6.1	Die abstrakte Klasse Writer	543
12.6.2	Datenkonvertierung durch den OutputStreamWriter	545
12.6.3	In Dateien schreiben mit der Klasse FileWriter	546
12.6.4	StringWriter und CharArrayWriter	547
12.6.5	Writer als Filter verketten	549
12.6.6	Gepufferte Ausgabe durch BufferedWriter	550
12.6.7	Ausgabemöglichkeiten durch PrintWriter erweitern	553
12.6.8	Daten mit FilterWriter filtern	554
12.7	Die Klassen um Reader	560
12.7.1	Die abstrakte Basisklasse Reader	560
12.7.2	Automatische Konvertierungen mit dem InputStreamReader	562
12.7.3	Dateien lesen mit der Klasse FileReader	563
12.7.4	StringReader und CharArrayReader	564
12.8	Schachteln von Eingabe-Streams	566
12.8.1	Gepufferte Eingaben mit der Klasse BufferedReader	566
12.8.2	LineNumberReader zählt automatisch Zeilen mit	567
12.8.3	Eingaben filtern mit der Klasse FilterReader	569
12.8.4	Daten zurücklegen mit der Klasse PushbackReader	572
12.9	Kommunikation zwischen Threads mit Pipes	575
12.9.1	PipedOutputStream und PipedInputStream	575
12.9.2	PipedWriter und PipedReader	577
12.10	Datenkompression	578
12.10.1	Datenströme komprimieren	579
12.10.2	Zip-Archive	582
12.11	Prüfsummen	590
12.11.1	Die Schnittstelle Checksum	591
12.11.2	Die Klasse CRC32	591
12.11.3	Die Adler32-Klasse	594
12.12	Persistente Objekte und Serialisierung	595
12.12.1	Objekte speichern	596
12.12.2	Objekte lesen	598
12.12.3	Die Schnittstelle Serializable	600
12.12.4	Tiefe Objektkopien	601
12.12.5	Felder sind implizit Serializable	603
12.12.6	Versionenverwaltung und die SUID	603
12.12.7	Beispiele aus den Standard-Klassen	606
12.12.8	Serialisieren in XML-Dateien	607
12.12.9	JSX (Java Serialization to XML)	608
12.12.10	XML-API von Sun	611
12.13	Die Logging-API	612

13	Die eXtensible Markup Language (XML) 617
13.1	Auszeichnungssprachen 617
13.1.1	Die Standard Generalized Markup Language (SGML) 617
13.1.2	Extensible Markup Language (XML) 618
13.2	Eigenschaften von XML-Dokumenten 618
13.2.1	Elemente und Attribute 618
13.2.2	Beschreibungssprache für den Aufbau von XML-Dokumenten 620
13.2.3	Schema – eine Alternative zu DTD 623
13.2.4	Namensraum (Namespace) 626
13.2.5	XML-Applikationen 627
13.3	Die Java APIs für XML 627
13.3.1	Das Document Object Model (DOM) 628
13.3.2	Simple API for XML Parsing (SAX) 628
13.3.3	Java Document Object Model (JDOM) 628
13.4	XML-Dateien mit JDOM verarbeiten 628
13.4.1	JDOM beziehen 629
13.4.2	Paketübersicht 629
13.4.3	Die Document-Klasse 631
13.4.4	Eingaben aus der Datei lesen 631
13.4.5	Das Dokument als XML-Datei ausgeben 632
13.4.6	Der Dokumenttyp 632
13.4.7	Elemente 633
13.4.8	Zugriff auf Elementinhalte 635
13.4.9	Liste mit Unterelementen erzeugen 637
13.4.10	Neue Elemente einfügen und ändern 638
13.4.11	Attributinhalte lesen und ändern 640
13.5	JAXP als Java-Schnittstelle zu XML 643
13.5.1	Einführung in XSLT 644
13.5.2	Umwandlung von XML-Dateien mit JDOM und JAXP 647
13.6	Serielle Verarbeitung von XML mit SAX 647
13.6.1	Ausgabe der Datei party.xml mit SAX 648
14	Grafikprogrammierung mit dem AWT 651
14.1	Das Abstract-Window-Toolkit 651
14.1.1	Java Foundation Classes 651
14.2	Fenster unter grafischen Oberflächen 652
14.2.1	Fenster öffnen 652
14.2.2	Größe und Position des Fensters verändern 655
14.2.3	Fenster und Dialog-Dekoration 656
14.3	Das Toolkit 656
14.3.1	Einen Hinweis beepen 657
14.4	Grundlegendes zum Zeichnen 657
14.4.1	Die paint()-Methode 657
14.4.2	Auffordern zum Neuzeichnen mit repaint() 659

14.5	Punkte, Linien und Rechtecke aller Art	660
14.5.1	Linien	660
14.5.2	Rechtecke	661
14.6	Alles was rund ist	662
14.7	Polygone und Polylines	663
14.7.1	Die Polygon-Klasse	664
14.7.2	N-Ecke zeichnen	665
14.7.3	Vollschanke Linien zeichnen	667
14.8	Zeichenketten schreiben	668
14.8.1	Einen neuen Zeichensatz bestimmen	669
14.8.2	Zeichensätze des Systems ermitteln	671
14.8.3	Die Klasse FontMetrics	673
14.8.4	True Type Fonts	675
14.9	Clipping-Operationen	677
14.10	Farben	681
14.10.1	Zufällige Farbblöcke zeichnen	682
14.10.2	Farbanteile zurückgeben	683
14.10.3	Vordefinierte Farben	683
14.10.4	Farben aus Hexadezimalzahlen erzeugen	684
14.10.5	Einen helleren oder dunkleren Farbton wählen	685
14.10.6	Farbmodelle HSB und RGB	687
14.10.7	Die Farben des Systems	687
14.11	Bilder anzeigen und Grafiken verwalten	692
14.11.1	Eine Grafik zeichnen	694
14.11.2	Grafiken zentrieren	696
14.11.3	Laden von Bildern mit dem MediaTracker beobachten	697
14.11.4	Kein Flackern durch Double-Buffering	701
14.11.5	Bilder skalieren	703
14.12	Programm-Icon setzen	705
14.12.1	VolatileImage	706
14.13	Grafiken speichern	707
14.13.1	Bilder im GIF-Format speichern	707
14.13.2	Gif speichern mit dem ACME-Paket	709
14.13.3	JPEG-Dateien mit dem Sun-Paket schreiben	709
14.13.4	Java Image Management Interface (JIMI)	712
14.14	Von Produzenten, Konsumenten und Beobachtern	714
14.14.1	Producer und Consumer für Bilder	714
14.14.2	Beispiel für die Übermittlung von Daten	715
14.14.3	Bilder selbst erstellen	718
14.14.4	Die Bildinformationen wieder auslesen	721
14.15	Filter	724
14.15.1	Grundlegende Eigenschaft von Filtern	724
14.15.2	Konkrete Filterklassen	725
14.15.3	Mit CropImageFilter Teile ausschneiden	726
14.15.4	Transparenz	727

14.16	Alles wird bunt mit Farbmodellen	727
14.16.1	Die abstrakte Klasse ColorModel	728
14.16.2	Farbwerte im Pixel mit der Klasse DirectColorModel	730
14.16.3	Die Klasse IndexColorModel	731
14.17	Drucken	735
14.17.1	Drucken mit dem einfachen Ansatz	735
14.17.2	Ein PrintJob	736
14.17.3	Drucken der Inhalte	738
14.17.4	Komponenten drucken	738
14.17.5	Den Drucker am Parallelport ansprechen	739
14.18	Java 2D-API	739
14.18.1	Grafische Objekte zeichnen	740
14.18.2	Geometrische Objekte durch Shape gekennzeichnet	741
14.18.3	Eigenschaften geometrischer Objekte	743
14.18.4	Transformationen mit einem AffineTransform-Objekt	751
14.19	Graphic Layers Framework	752
14.20	Grafikverarbeitung ohne grafische Oberfläche	753
14.20.1	Xvfb-Server	753
14.20.2	Pure Java AWT Toolkit (PJA)	753

15 Komponenten, Container und Ereignisse 755

15.1	Es tut sich was – Ereignisse beim AWT	755
15.1.1	Was ist ein Ereignis?	755
15.1.2	Die Klasse AWTEvent	755
15.1.3	Events auf verschiedenen Ebenen	755
15.1.4	Ereignisquellen, -senken und Horcher (Listener)	758
15.1.5	Listener implementieren	758
15.1.6	Listener bei Ereignisauslöser anmelden	759
15.2	Varianten, das Fenster zu schließen	760
15.2.1	Eine Klasse implementiert die Schnittstelle WindowListener	760
15.2.2	Adapterklassen nutzen	763
15.2.3	Innere Mitgliedsklassen und innere anonyme Klassen	764
15.2.4	Generic Listener	765
15.3	Komponenten im AWT und in Swing	765
15.3.1	Peer-Klassen und Lightweight-Komponenten	766
15.3.2	Die Basis aller Komponenten: Component und JComponent	767
15.3.3	Proportionales Vergrößern eines Fensters	768
15.3.4	Dynamisches Layout während einer Größenänderung	770
15.3.5	Hinzufügen von Komponenten	770
15.4	Das Swing-Fenster JFrame	772
15.4.1	Kinder auf einem Swing-Fenster	772
15.4.2	Schließen eines Swing-Fensters	772
15.5	Ein Informationstext über die Klasse JLabel	773
15.5.1	Mehrzeiliger Text	776

15.6	Die Klasse ImageIcon 776
15.6.1	Die Schnittstelle Icon 778
15.6.2	Was Icon und Image verbindet 780
15.7	Eine Schaltfläche (JButton) 780
15.7.1	Der aufmerksame ActionListener 782
15.7.2	Generic Listener für Schaltflächen-Ereignisse verwenden 784
15.7.3	AbstractButton 787
15.7.4	JToggleButton 789
15.8	Tooltips 789
15.9	Horizontale und vertikale Schieberegler 790
15.9.1	Der AdjustmentListener, der auf Änderungen hört 793
15.10	JSlider 795
15.11	Ein Auswahlmenü – Choice, JComboBox 796
15.11.1	ItemListener 799
15.12	Eines aus vielen – Kontrollfelder (JCheckBox) 800
15.12.1	Ereignisse über ItemListener 801
15.13	Kontrollfeldgruppen, Optionsfelder und JRadioButton 802
15.14	Der Fortschrittsbalken JProgressBar 804
15.15	Rahmen (Borders) 805
15.16	Symbolleisten alias Toolbars 808
15.17	Menüs 810
15.17.1	Die Menüleisten und die Einträge 810
15.17.2	Menüeinträge definieren 811
15.17.3	Mnemonics und Short-Cuts (Accelerator) 812
15.17.4	Beispiel für ein Programm mit Menüleisten 814
15.18	Popup-Menüs 819
15.19	Alles Auslegungssache: Die Layoutmanager 822
15.19.1	Null-Layout 822
15.19.2	FlowLayout 823
15.19.3	BorderLayout 825
15.19.4	GridLayout 827
15.19.5	Der GridBagLayout-Manager 829
15.19.6	Weitere Layoutmanager 834
15.20	Der Inhalt einer Zeichenfläche: JPanel 834
15.21	Das Konzept des Model-View-Controllers 835
15.22	List-Boxen 837
15.23	JSpinner 839
15.24	Texteingabefelder 841
15.24.1	Text in einer Eingabezeile 841
15.24.2	Die Oberklasse der JText-Komponenten: JTextField 842
15.24.3	JPasswordField 843

15.24.4	Validierende Eingabefelder	844
15.24.5	Mehrzeilige Textfelder	845
15.24.6	Die Editor-Klasse JEditorPane	847
15.25	Bäume mit JTree-Objekten	850
15.25.1	Selektionen bemerken	851
15.26	Tabellen mit JTable	852
15.26.1	Ein eigenes Modell	853
15.26.2	AbstractTableModel	854
15.26.3	DefaultTableModel	856
15.26.4	Ein eigener Renderer für Tabellen	858
15.27	JRootPane und JLayeredPane	861
15.28	Dialoge	862
15.28.1	Der Farbauswahldialog JColorChooser	863
15.28.2	Der Dateiauswahldialog	865
15.29	Das Java Look&Feel	868
15.30	Die Zwischenablage (Clipboard)	869
15.31	Undo durchführen	872
15.32	Ereignisverarbeitung auf unterster Ebene	874
15.33	AWT, Swing und die Threads	875
15.33.1	Warum Swing nicht Thread-sicher ist	876
15.33.2	Swing-Elemente bedienen mit invokeLater() und invokeAndWait()	878
15.34	Selbst definierte Cursor	880
15.34.1	Flackern des Mauszeigers bei Animationen vermeiden	881
15.35	Mausrad-Unterstützung	882
15.36	Benutzerinteraktionen automatisieren	882
<hr/>		
16	Netzwerkprogrammierung	885
16.1	Grundlegende Begriffe	885
16.1.1	Internet-Standards und RFC	885
16.2	URL-Verbindungen und URL-Objekte	886
16.2.1	Die Klasse URL	887
16.2.2	Informationen über eine URL	889
16.2.3	Der Zugriff auf die Daten über die Klasse URL	892
16.3	Die Klasse URLConnection	894
16.3.1	Methoden und Anwendung von URLConnection	894
16.3.2	Protokoll- und Content-Handler	897
16.3.3	Im Detail: von URL zu URLConnection	898
16.3.4	Autorisierte URL-Verbindungen mit Basic Authentication	899
16.4	Das Common Gateway Interface	901
16.4.1	Parameter für ein CGI-Programm	901
16.4.2	Kodieren der Parameter für CGI-Programme	902
16.4.3	Eine Suchmaschine ansprechen	904

16.5	Host- und IP-Adressen	904
16.5.1	Klasse-K-Netz	906
16.5.2	IP-Adresse des lokalen Hosts	906
16.5.3	Die Methode getAllByName()	907
16.6	NetworkInterface	908
16.7	IPv6 für Java mit Jipsy	909
16.8	Socket-Programmierung	910
16.8.1	Das Netzwerk ist der Computer	910
16.8.2	Standarddienste unter Windows nachinstallieren	912
16.8.3	Stream-Sockets	912
16.8.4	Informationen über den Socket	914
16.8.5	Mit telnet an den Ports horchen	916
16.8.6	Ein kleines Echo – lebt der Rechner noch?	916
16.9	Client/Server-Kommunikation	917
16.9.1	Warten auf Verbindungen	918
16.9.2	Ein Multiplikations-Server	919
16.10	SLL-Verbindungen mit JSSE	920
16.11	Webprotokolle mit NetComponents nutzen	921
16.12	E-Mail	922
16.12.1	Wie eine E-Mail um die Welt geht	922
16.12.2	Übertragungsprotokolle	922
16.12.3	Das Simple Mail Transfer Protocol	925
16.12.4	E-Mails versenden mit Suns JavaMail-API	925
16.12.5	E-Mails mittels POP3 abrufen	926
16.13	Arbeitsweise eines Webservers	928
16.13.1	Das Hypertext Transfer Protocol (HTTP)	928
16.13.2	Anfragen an den Server	929
16.13.3	Die Antworten vom Server	932
16.14	Datagram-Sockets	935
16.14.1	Die Klasse DatagramSocket	937
16.14.2	Datagramme und die Klasse DatagramPacket	939
16.14.3	Auf ein hereinkommendes Paket warten	939
16.14.4	Ein Paket zum Senden vorbereiten	941
16.14.5	Methoden der Klasse DatagramPacket	941
16.14.6	Das Paket senden	942
16.14.7	Die Zeitdienste und ein eigener Server und Client	943
16.15	Internet Control Message Protocol (ICMP)	946
16.15.1	Ping	946
16.16	Multicast-Kommunikation	947

17	Servlets und Java Server Pages	949
17.1	Dynamische Webseiten und Servlets	949
17.1.1	Was sind Servlets?	949
17.1.2	Was sind Java Server Pages?	950
17.1.3	Vorteil von JSP/Servlets gegenüber CGI-Programmen	951
17.2	Vom Client zum Server und wieder zurück	952
17.2.1	Der bittende Client	952
17.2.2	Was erzeugt ein Webserver für eine Antwort?	954
17.2.3	Wer oder was ist MIME?	954
17.3	Servlets und Java Server Pages entwickeln und testen	955
17.3.1	Servlet-Container	956
17.3.2	Webserver mit Servlet-Funktionalität	956
17.3.3	Tomcat	957
17.4	Java Server Pages	958
17.4.1	JSP mit Tomcat nutzen	958
17.5	Skript-Elemente	959
17.5.1	Scriptlets	959
17.5.2	Ausdrücke	960
17.5.3	Deklarationen	960
17.5.4	Kommentare und Quoting	961
17.6	Webapplikationen	961
17.7	Implizite Objekte	962
17.8	Entsprechende XML-Tags	963
17.9	Was der Browser mit auf den Weg gibt – HttpServletRequest	964
17.9.1	Verarbeiten der Header	964
17.9.2	Hilfsfunktion im Umgang mit Headern	965
17.9.3	Übersicht der Browser-Header	965
17.10	Formulardaten	966
17.11	Das HttpServletResponse-Objekt	968
17.11.1	Automatisches Neuladen	968
17.11.2	Seiten umlenken	969
17.12	JSP-Direktiven	970
17.12.1	page-Direktiven im Überblick	970
17.12.2	include-Direktive	972
17.13	Aktionen	972
17.13.1	Aktion include	973
17.13.2	Aktion forward	973
17.13.3	Aktion plugin	973
17.14	Beans	974
17.14.1	Beans in JSP-Seiten anlegen, Attribute setzen und erfragen	975
17.14.2	Der schnelle Zugriff auf Parameter	976

17.15	Kleine Kekse: die Klasse Cookies	976
17.15.1	Cookies erzeugen und setzen	977
17.15.2	Cookies vom Servlet einlesen	977
17.15.3	Kleine Helfer für Cookies	979
17.15.4	Cookie-Status ändern	979
17.15.5	Langlebige Cookies	981
17.15.6	Ein Warenkorbsystem	981
17.16	Sitzungsverfolgung (Session Tracking)	982
17.16.1	Das mit einer Sitzung verbundene Objekt HttpSession	983
17.16.2	Werte mit einer Sitzung assoziieren und auslesen	983
17.16.3	URL-Rewriting	984
17.16.4	Zusätzliche Informationen	985
17.17	Tag-Libraries	987
17.17.1	Standard Tag Library (JSTL) der Apache-Gruppe	988
17.17.2	Beispiel mit einer Taglib-Direktive	989
17.18	Das erste Servlet kompilieren und ausführen	991
17.18.1	Servlets kompilieren	991
17.18.2	Wohin mit dem Servlet?	992
17.19	Der Lebenszyklus eines Servlets	992
17.19.1	Initialisierung in init()	993
17.19.2	Abfragen bei service()	995
17.19.3	Mehrere Anfragen beim Servlet und die Thread-Sicherheit	996
17.19.4	Das Ende eines Servlets	997
17.20	Das HttpServletResponse-Objekt	997
17.20.1	Wir generieren eine Webseite	997
17.20.2	Binärdaten senden	999
17.20.3	Komprimierte Daten mit Content-Encoding	1000
17.20.4	Noch mehr über Header, die der Server setzt	1001
17.21	Servlets und Sessions	1002
17.22	Weiterleiten und Einbinden von Servlet-Inhalten	1003
17.23	Inter-Servlet-Kommunikation	1004
17.23.1	Daten zwischen Servlets teilen	1004
17.24	Internationalisierung	1005
17.24.1	Die Länderkennung des Anfragers auslesen	1005
17.24.2	Länderkennung für die Ausgabe setzen	1005
17.24.3	Westeuropäische Texte senden	1006
17.25	Sonstiges zu den Servern	1007
17.25.1	Den internen Compiler bei Tomcat für JSP ändern	1007
17.26	Tomcat: Spezielles	1007
17.26.1	Tomcat als Service unter Windows NT ausführen	1007
17.26.2	MIME-Types mit Tomcat verbinden	1008
17.26.3	Servlets beim Start laden	1008

17.27	Ein Servlet generiert WAP-Seiten für das Handy	1008
17.27.1	Ein WAP-Handy simulieren	1009
17.27.2	Übersicht der wichtigsten Tags	1010
17.27.3	Der Gateway	1011
17.27.4	WML-Seiten aufbauen	1013
17.27.5	Interessante Links zum Thema Servlets/JSP	1013
17.28	Text in HTML-konformen Text umwandeln	1014

18 Verteilte Programmierung mit RMI und SOAP 1017

18.1	Entfernte Methoden	1017
18.1.1	Wie entfernte Methoden arbeiten	1017
18.1.2	Stellvertreter (Proxy)	1017
18.1.3	Wie die Stellvertreter die Daten übertragen	1018
18.1.4	Probleme mit entfernten Methoden	1019
18.2	Nutzen von RMI bei Middleware-Lösungen	1020
18.3	Die Lösung für Java ist RMI	1021
18.3.1	Entfernte Objekte programmieren	1021
18.3.2	Entfernte und lokale Objekte im Vergleich	1021
18.3.3	RMI und CORBA	1022
18.4	Definition einer entfernten Schnittstelle	1022
18.5	Das entfernte Objekt	1023
18.5.1	Der Bauplan für entfernte Objekte	1024
18.5.2	Der Konstruktor	1024
18.5.3	Implementierung der entfernten Methoden	1026
18.5.4	UnicastRemoteObject, RemoteServer und RemoteObject	1026
18.6	Stellvertreterobjekte erzeugen	1028
18.6.1	Das Dienstprogramm rmic	1028
18.7	Der Namensdienst (Registry)	1029
18.7.1	Der Port	1030
18.8	Der Server: entfernte Objekte beim Namensdienst anmelden	1030
18.8.1	Automatisches Anmelden bei Bedarf	1031
18.9	Einen Client programmieren	1032
18.9.1	Einfaches Logging	1033
18.10	Aufräumen mit dem DGC	1033
18.11	Entfernte Objekte übergeben und laden	1033
18.11.1	Klassen vom RMI-Klassenlader nachladen	1034
18.11.2	Sicherheitsmanager	1034
18.12	Registry wird vom Server gestartet	1036
18.13	RMI über die Firewall	1037
18.13.1	RMI über HTTP getunnelt	1037
18.14	Daily Soap	1037
18.14.1	SOAP-Implementierung der Apache-Gruppe	1039

18.14.2	Einen Client mit der Apache-Bibliothek implementieren	1039
18.14.3	Der Seifen-Server	1041
18.15	Java-API für XML Messaging (JAXM)	1041
18.16	Java Message Service (JMS)	1042
18.16.1	OpenJMS	1043
18.16.2	Beispiel mit Konsument und Produzent im Publish-Subscribe-Modell	1043
<hr/>		
19	Applets, Midlets und Sound	1045
19.1	Applets und Applikationen – wer darf was?	1045
19.2	Das erste Hallo-Applet	1045
19.3	Die Zyklen eines Applets	1047
19.4	Parameter an das Applet übergeben	1047
19.4.1	Wie das Applet den Browserinhalt ändern kann	1047
19.4.2	Den Ursprung des Applets erfragen	1048
19.4.3	Was ein Applet alles darf	1050
19.5	Fehler in Applets finden	1050
19.6	Browserabhängiges Verhalten	1050
19.6.1	Java im Browser aktiviert?	1050
19.6.2	Läuft das Applet unter Netscape oder Microsoft Explorer?	1051
19.6.3	Datenaustausch zwischen Applets und Java-Skripten	1052
19.7	Datenaustausch zwischen Applets	1052
19.8	Musik in einem Applet und in Applikationen	1055
19.8.1	Fest verdrahtete Musikdatei in einem Applet	1056
19.8.2	Variable Musikdatei über einen Parameter	1056
19.8.3	WAV- und MIDI-Dateien abspielen	1057
19.9	Webstart	1058
19.10	Java 2 Micro Edition	1058
19.10.1	Konfigurationen	1058
19.10.2	Profile	1059
<hr/>		
20	Datenbankmanagement mit JDBC	1063
20.1	Das relationale Modell	1063
20.1.1	Relationale und objektorientierte Datenbanken	1064
20.2	JDBC: der Zugriff auf Datenbanken über Java	1064
20.3	Die Rolle von SQL	1065
20.3.1	Ein Rundgang durch SQL-Anfragen	1065
20.3.2	Datenabfrage mit der Data Query Language (DQL)	1066
20.4	Datenbanktreiber für den Zugriff	1068

20.5	Datenbanken und Ihre Treiber	1070
20.5.1	Datenbank Interbase und Firebird	1070
20.5.2	Interbase JDBC-Treiber	1071
20.5.3	Die freie Datenbank MySQL	1072
20.5.4	JDBC-Treiber für MySQL	1074
20.5.5	Die Datenbank Microsoft Access	1074
20.5.6	Ein Typ 4-Treiber für den Microsoft SQL Server 2000	1074
20.5.7	Die JDBC-ODBC Bridge	1074
20.5.8	ODBC einrichten und Access damit verwenden	1075
20.5.9	Oracle8i Enterprise Edition	1076
20.6	Eine Beispiel-Abfrage	1077
20.7	Mit Java an eine Datenbank andocken	1078
20.7.1	Der Treibermanager	1078
20.7.2	Eine Aufzählung aller Treiber	1078
20.7.3	Log-Informationen	1079
20.7.4	Den Treiber laden	1080
20.7.5	Wie Treiber programmiert sind	1081
20.7.6	Verbindung zur Datenbank	1084
20.8	Datenbankabfragen	1087
20.8.1	Abfragen über das Statement-Objekt	1087
20.8.2	Ergebnisse einer Abfrage in ResultSet	1088
20.8.3	wasNull() bei ResultSet	1089
20.9	Java und SQL-Datentypen	1089
20.9.1	Die getXXX()-Methoden	1090
20.10	Transaktionen	1092
20.11	Elemente einer Datenbank hinzufügen und aktualisieren	1092
20.11.1	Batch-Updates	1093
20.12	Vorbereitete Anweisungen (Prepared Statements)	1094
20.12.1	PreparedStatement-Objekte vorbereiten	1095
20.12.2	Werte für die Platzhalter eines PreparedStatement	1096
20.13	Metadaten	1096
20.13.1	Metadaten über die Tabelle	1096
20.13.2	Informationen über die Datenbank	1100
20.14	Die Ausnahmen bei JDBC	1101
20.15	Java Data Objects (JDO)	1102
21	Reflection	1105
21.1	Einfach mal reinschauen	1105
21.2	Mit dem Class-Objekt etwas über Klassen erfahren	1105
21.2.1	An ein Class-Objekt kommen	1105
21.2.2	Was das Class-Objekt beschreibt	1107
21.2.3	Der Name der Klasse	1109

21.2.4	Oberklassen finden 1111
21.2.5	Implementierte Interfaces einer Klasse oder eines Interfaces 1112
21.2.6	Modifizierer und die Klasse Modifier 1112
21.2.7	Die Attribute einer Klasse 1114
21.2.8	Methoden einer Klasse erfragen 1117
21.2.9	Konstruktoren einer Klasse 1120
21.3	Objekte manipulieren 1122
21.3.1	Objekte erzeugen 1122
21.3.2	Die Belegung der Variablen erfragen 1124
21.3.3	Variablen setzen 1126
21.3.4	Private Attribute ändern 1127
21.4	Methoden aufrufen 1128
21.4.1	Statische Methoden aufrufen 1129
21.4.2	Dynamische Methodenaufrufe bei festen Methoden beschleunigen 1130
21.5	Informationen und Identifizierung von Paketen 1132
21.5.1	Geladene Pakete 1133

22 Komponenten durch Bohnen 1135

22.1	Grundlagen der Komponententechnik 1135
22.1.1	Brauchen wir überhaupt Komponenten? 1135
22.1.2	Visuelle und nicht visuelle Komponenten 1136
22.1.3	Andere Komponententechnologien oder: Was uns Microsoft brachte 1136
22.2	Das Java-Beans Development Kit (BDK) 1137
22.2.1	Eine Beispielsitzung im BDK 1138
22.2.2	Verknüpfungen zwischen Komponenten 1139
22.2.3	Beans speichern 1140
22.3	Die kleinste Bohne der Welt 1140
22.4	Jar-Archive für Komponenten 1141
22.5	Worauf JavaBeans basieren 1142
22.6	Eigenschaften 1143
22.6.1	Einfache Eigenschaften 1144
22.6.2	Boolesche Eigenschaften 1144
22.6.3	Indizierte Eigenschaften 1145
22.7	Ereignisse 1146
22.7.1	Multicast und Unicast 1146
22.7.2	Namenskonvention 1147
22.8	Weitere Eigenschaften 1149
22.8.1	Gebundene Eigenschaften 1149
22.8.2	Anwendung von PropertyChange bei AWT-Komponenten 1152
22.8.3	Veto-Eigenschaften. Dagegen! 1152

22.9	Bean-Eigenschaften anpassen	1153
22.9.1	Customizer	1154
22.10	Property-Editoren	1154
22.11	BeanInfo	1155
22.12	Beliebte Fehler	1155
23	Java Native Interface (JNI)	1157
23.1	Java Native Interface und Invocation-API	1157
23.2	Die Schritte zur Einbindung einer C-Funktion in ein Java-Programm	1158
23.2.1	Schreiben des Java-Codes	1158
23.2.2	Kompilieren des Java-Codes	1158
23.2.3	Erzeugen der Header-Datei	1159
23.2.4	Implementierung der Methode in C	1160
23.2.5	Übersetzen der C-Programme und Erzeugen der dynamischen Bibliothek	1160
23.2.6	Setzen der Umgebungsvariable	1161
23.3	Erweiterung unseres Programms	1161
23.4	Erweiterte JNI-Eigenschaften	1162
23.4.1	Klassendefinitionen	1162
23.4.2	Zugriff auf Attribute	1163
24	Sicherheitskonzepte	1167
24.1	Der Sandkasten (Sandbox)	1167
24.2	Sicherheitsmanager (Security Manager)	1167
24.2.1	Der Sicherheitsmanager bei Applets	1169
24.2.2	Sicherheitsmanager aktivieren	1171
24.2.3	Wie nutzen die Java-Bibliotheken den Sicherheitsmanager?	1172
24.2.4	Rechte vergeben durch Policy-Dateien	1172
24.2.5	Erstellen von Rechtedateien mit dem grafischen Policy-Tool	1174
24.3	Klassenlader (Class Loader)	1175
24.3.1	Wie heißt die Klasse mit der Methode main()?	1175
24.4	Digitale Unterschriften	1176
24.4.1	Die MDx-Reihe	1176
24.4.2	Secure Hash Algorithm (SHA)	1177
24.4.3	Mit der Security-API einen Fingerabdruck berechnen	1178
24.4.4	Die Klasse MessageDigest	1178
24.4.5	Unix-Crypt	1181
24.5	Verschlüsseln von Datenströmen	1181

25	Dienstprogramme für die Java-Umgebung	1183
25.1	Die Werkzeuge im Überblick	1183
25.2	Der Compiler javac	1183
25.2.1	Der Java-Interpreter java	1184
25.3	Das Archivformat Jar	1185
25.3.1	Das Dienstprogramm Jar benutzen	1186
25.3.2	Das Manifest	1188
25.3.3	Jar-Archive für Applets und Applikation	1189
25.4	Mit JavaDoc und Doclets dokumentieren	1190
25.4.1	Mit JavaDoc Dokumentationen erstellen	1190
25.4.2	Wie JavaDoc benutzt wird	1191
25.4.3	Eine Dokumentation erstellen	1193
25.4.4	JavaDoc und Doclets	1196
25.5	Dienstprogramme zur Signierung	1196
25.5.1	Mit keytool Schlüssel erzeugen	1196
25.5.2	Signieren mit jarsigner	1197
25.6	Konvertierung von Java Byte Code in ein Windows-Exe mit JET	1198
25.7	Manteln von Javaklassen in ein Windows-Exe mit JexePack	1198
25.8	Decompiler	1198
25.8.1	Jad, ein schneller Decompiler	1199
25.8.2	SourceAgain	1202
25.8.3	Dekompilieren erschweren	1202
25.9	Obfuscate Programm RetroGuard	1203
25.10	Source-Code Beautifier	1203
25.11	Ant	1204
25.11.1	Bezug und Installation von Ant	1205
25.11.2	Properties	1206
25.11.3	Externe und vordefinierte Properties	1207
25.11.4	Weitere Leistungen	1208
26	Style-Guide	1209
26.1	Programmierrichtlinien	1209
26.2	Allgemeine Richtlinien	1209
26.3	Quellcode kommentieren	1210
26.3.1	Bemerkungen über JavaDoc	1212
26.3.2	Gotcha-Schlüsselwörter	1213
26.4	Bezeichnernamen	1214
26.4.1	Ungarische Notation	1214
26.4.2	Vorschlag für die Namensgebung	1215
26.5	Formatierung	1216
26.5.1	Einrücken von Programmcode – die Vergangenheit	1216
26.5.2	Verbundene Ausdrücke	1217

- 26.5.3 Kontrollierter Datenfluss 1217 /
- 26.5.4 Funktionen 1218
- 26.6 Ausdrücke 1220**
- 26.7 Anweisungen 1221**
- 26.7.1 Schleifen 1221
- 26.7.2 Switch, case und Durchfallen 1223
- 26.8 Reihenfolge der Eigenschaften in Klassen 1224**
- 26.9 Zugriffsrechte und Zugriffsmethoden 1224 ~**
- 26.9.1 Accessors/Zugriffsmethoden 1225
- 26.10 Verweise 1225**

A Literatur 1227

B Spenden 1233

C Die Begleit-CD 1235

Index 1237