Inhalt

Vorwort 31

1 Schon wieder eine neue Sprache? 41

1.1 Der erste Kontakt 41

1.2 Historischer Hintergrund 41

1.3 Eigenschaften von Java 42
1.3.1 Die virtuelle Maschine 43
1.3.2 Kein Präprozessor 44
1.3.3 Überladene Operatoren 45
1.3.4 Zeiger und Referenzen 45
1.3.5 Garbage-Collector 46
1.3.6 Ausnahmenbehandlung 47
1.3.7 Objektorientierung in Java 47
1.3.8 Java-Security-Model 47

1.4 Java im Vergleich zu anderen Sprachen 48
1.4.1 Java und JavaScript 49
1.4.2 Normierungsversuche 49

1.5 Die Rolle von Java im Web 49

1.6 Aufkommen von Stand-Alone-Applikationen 50

1.7 Entwicklungs- und Laufzeitumgebungen 50
1.7.1 Aller Anfang mit dem Java SDK 50
1.7.2 Kaffe von Transvirtual Technologies 51
1.7.3 JBuilder 51
1.7.4 Die Entwicklungsumgebung von Sun: Forté 53
1.7.5 Umgebungen von IBM 53
1.7.6 Visual Café von WebGain (ehemals Symantec) 54
1.7.7 Ein Wort zu Microsoft, Java und zu J++ 54
1.7.8 Kawa 55
1.7.9 Die Entwicklungsumgebung CodeGuide 55

1.8 Installationsanleitung für das Java 2 SDK unter Microsoft Windows 56
1.8.1 Das Java 2 SDK beziehen 57
1.8.2 Java SDK installieren 57
1.8.3 Compiler und Interpreter nutzen 58

1.9 Erstes Programm compilieren und testen 59
1.9.1 Häufige Compiler- und Interpreterprobleme 60

2 Sprachbeschreibung 63

2.1 Anweisungen und Programme 63

2.2 Programme 65

Bibliografische Informationen
http://d-nb.info/963256165

digitalisiert durch DEUTSCHE NATIONAL BIBLIOTHEK

2.2.1 Kommentare **67**
2.2.2 Funktionsaufrufe als Anweisungen **67**
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 Bezeichner **71**
2.3.3 Reservierte Schlüsselwörter **72**
2.3.4 Token **73**
2.3.5 Semantik **74**

2.4 Datentypen 74
2.4.1 Primitive Datentypen **75**
2.4.2 Wahrheitswerte **75**
2.4.3 Variablendeklarationen **76**
2.4.4 Ganzzahlige Datentypen **77**
2.4.5 Die Fließkommazahlen **78**
2.4.6 Zeichen **79**
2.4.7 Die Typanpassung (Das Casting) **80**
2.4.8 Lokale Variablen, Blöcke und Sichtbarkeit **83**
2.4.9 Initialisierung von lokalen Variablen **84**

2.5 Ausdrücke 85
2.5.1 Zuweisungsoperator und Verbundoperator **86**
2.5.2 Präfix- oder Postfix-Inkrement und -Dekrement **87**
2.5.3 Unäres Minus und Plus **88**
2.5.4 Arithmetische Operatoren **89**
2.5.5 Die relationalen Operatoren **92**
2.5.6 Logische Operatoren **93**
2.5.7 Reihenfolge und Rang der Operatoren in der Auswertungsreihenfolge **93**
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 **100**

2.7 Schleifen 102
2.7.1 Die while-Schleife **103**
2.7.2 Schleifenbedingungen und Vergleiche mit == **103**
2.7.3 Die do/while-Schleife **105**
2.7.4 Die for-Schleife **106**
2.7.5 Ausbruch planen mit break und Wiedereinstieg mit continue **109**
2.7.6 Break und Continue mit Sprungmarken **110**

2.8 Methoden einer Klasse 111
2.8.1 Bestandteil einer Funktion **111**
2.8.2 Aufruf **113**
2.8.3 Methoden ohne Parameter **113**

2.8.4 Parameter und Wertübergabe **114**
2.8.5 Methoden vorzeitig mit return beenden **116**
2.8.6 Nicht erreichbarer Quellcode bei Funktionen **116**
2.8.7 Rückgabewerte **117**
2.8.8 Methoden überladen **120**
2.8.9 Vorinitialisierte Parameter bei Funktionen **122**
2.8.10 Finale lokale Variablen **122**
2.8.11 Finale Referenzen in Objekten und das fehlende const **123**
2.8.12 Rekursive Funktionen **124**
2.8.13 Die Ackermann-Funktion **127**
2.8.14 Die Türme von Hanoi **128**

2.9 Noch mehr Operatoren 130
2.9.1 Bit-Operationen **130**
2.9.2 Vorzeichenlose Bytes in ein Integer und Char konvertieren **132**
2.9.3 Variablen mit Xor vertauschen **133**
2.9.4 Die Verschiebe-Operatoren **133**
2.9.5 Setzen, Löschen, Umdrehen, Testen von Bits **136**
2.9.6 Der Bedingungsoperator **136**
2.9.7 Überladenes Plus für Strings **138**

3 Klassen und Objekte 141

3.1 Objektorientierte Programmierung 141
3.1.1 Warum überhaupt OOP? **141**
3.1.2 Modularität und Wiederverwertbarkeit **142**

3.2 Klassen benutzen 142
3.2.1 Die Klasse Point **143**
3.2.2 Etwas über die UML **144**
3.2.3 Anlegen eines Exemplars einer Klasse **146**
3.2.4 Zugriff auf Variablen und Methoden mit dem Punkt **147**
3.2.5 Konstruktoren **148**
3.2.6 Die null-Referenz **148**

3.3 Mit Referenzen arbeiten 150
3.3.1 Zuweisungen bei Referenzen **150**
3.3.2 Funktionen mit nicht-primitiven Parametern **151**
3.3.3 Gleichheit von Objekten und die Methode equals() **152**

3.4 Arrays 154
3.4.1 Deklaration von Arrays **154**
3.4.2 Arrays mit Inhalt **155**
3.4.3 Die Länge eines Arrays mit length **156**
3.4.4 Zugriff auf die Elemente **156**
3.4.5 Array-Objekte erzeugen **158**
3.4.6 Fehler bei Arrays **159**
3.4.7 Arrays mit nicht-primitiven Elementen **160**
3.4.8 Arrays und Objekte **160**
3.4.9 Initialisierte Array-Objekte **161**

3.4.10 Mehrdimensionale Arrays **162**
3.4.11 Die Wahrheit über die Array-Initialisierung **164**
3.4.12 Arrays kopieren und füllen **165**
3.4.13 Mehrere Rückgabeparameter **166**
3.4.14 Parameter per Referenz übergeben **166**
3.4.15 Der Einstiegspunkt für das Laufzeitsystem **167**
3.4.16 Der Rückgabewert von main() **168**

4 Der Umgang mit Zeichenketten 169

4.1 Strings und deren Anwendung 169
4.1.1 String-Objekte für konstante Zeichenketten **169**
4.1.2 String-Objekte verraten viel **171**
4.1.3 Gut, dass wir verglichen haben **171**
4.1.4 Stringteile extrahieren **175**
4.1.5 Veränderte Strings liefern **177**
4.1.6 Typen in Zeichenketten konvertieren **179**

4.2 Veränderbare Zeichenketten mit der Klasse StringBuffer 180
4.2.1 Anlegen von StringBuffer-Objekten **181**
4.2.2 Die Länge eines StringBuffer-Objekts lesen und setzen **181**
4.2.3 Daten anhängen **182**
4.2.4 Zeichen(folgen) setzen, erfragen, löschen **182**

4.3 Vergleiche von Zeichenketten als String und StringBuffer 183
4.3.1 Sollte es ein equals() und hash() bei StringBuffer geben? **184**

4.4 Ein paar kleine Helfer 185
4.4.1 Strings einer gegebenen Länge erzeugen und rechtsbündig ausgeben **185**
4.4.2 Teile im String ersetzen **185**

4.5 Zeichenkodierungen umwandeln 186

4.6 Sprachabhängiges Vergleichen mit der Collator-Klasse 187
4.6.1 Effiziente interne Speicherung für Sortierung **189**

4.7 Die Klasse StringTokenizer 190

4.8 StreamTokenizer 193

4.9 Formatieren mit Format-Objekten 196
4.9.1 Ausgaben formatieren **198**
4.9.2 Dezimalzahlformatierung **200**

4.10 Reguläre Ausdrücke 202
4.10.1 Splitten von Zeichenketten **203**
4.10.2 split() in String **204**
4.10.3 Das Paket gnu.regexp **205**

4.11 Überprüfung der E-Mail-Adressen und Kreditkarteninformationen 205
4.11.1 Gültige E-Mail-Adressen **205**
4.11.2 Kreditkartennummern testen **206**

5 Mathematisches 211

5.1 Arithmetik in Java 211
5.1.1 Soll eine Division durch Null zur Übersetzungszeit erkannt werden? 212

5.2 Die Funktionen der Math-Klasse 213
5.2.1 Attribute 213
5.2.2 Winkelfunktionen (trigonometrische Funktionen und Arkusfunktionen) 213
5.2.3 Runden von Werten 214
5.2.4 Exponentialfunktionen 215
5.2.5 Division 216
5.2.6 Absolutwerte und Maximum, Minimum 216
5.2.7 Zufallszahlen 217

5.3 Mathe bitte strikt 217
5.3.1 Strikt Fließkomma mit strictfp 217
5.3.2 Die Klassen Math und StrictMath 218

5.4 Die Random-Klasse 218

5.5 Große Zahlen 220
5.5.1 Die Klasse BigInteger 220
5.5.2 Ganz lange Fakultäten 222

5.6 Probleme mit Java und der Mathematik 223

5.7 Das Java-Matrix-Paket Jama 223

6 Eigene Klassen schreiben 227

6.1 Eigene Klassen definieren 227
6.1.1 Methodenaufrufe und Nebeneffekte 229
6.1.2 Argumentübergabe mit Referenzen 229
6.1.3 Die this-Referenz 230

6.2 Assoziationen zwischen Objekten 232

6.3 Privatsphäre und Sichtbarkeit 233
6.3.1 Wieso nicht freie Methoden und Variablen für alle? 234
6.3.2 Privat ist nicht ganz privat. Es kommt darauf an wer's sieht 235
6.3.3 Zugriffsmethoden für Attribute definieren 237
6.3.4 Zusammenfassung zur Sichtbarkeit 239

6.4 Statische Methoden und Variablen 239
6.4.1 Warum statische Eigenschaften sinnvoll sind 239
6.4.2 Statische Eigenschaften mit static 240
6.4.3 Statische Eigenschaften als Objekteigenschaften nutzen 241
6.4.4 Statische Eigenschaften und Objekteigenschaften 241
6.4.5 Statische Variablen zum Datenaustausch 242
6.4.6 Warum die Groß- und Kleinschreibung wichtig ist 243
6.4.7 Konstanten mit dem Schlüsselwort final bei Variablen 243
6.4.8 Typsicherere Konstanten 244
6.4.9 Statische Blöcke 245

6.5 Objekte anlegen und zerstören 246
6.5.1 Konstruktoren schreiben **246**
6.5.2 Einen anderen Konstruktor der gleichen Klasse aufrufen **247**
6.5.3 Initialisierung der Objekt- und Klassenvariablen **249**
6.5.4 Finale Werte im Konstruktor setzen **251**
6.5.5 Exemplarinitialisierer (Instanzinitialisierer) **252**
6.5.6 Zerstörung eines Objekts durch den Müllaufsammler **253**
6.5.7 Implizit erzeugte Stringobjekte **254**
6.5.8 Zusammenfassung: Konstruktoren und Methoden **255**

6.6 Vererbung 255
6.6.1 Vererbung in Java **255**
6.6.2 Einfach- und Mehrfachvererbung **256**
6.6.3 Kleidungsstücke modelliert **256**
6.6.4 Sichtbarkeit **258**
6.6.5 Das Substitutionsprinzip **258**
6.6.6 Automatische und Explizite Typanpassung **259**
6.6.7 Finale Klassen **260**
6.6.8 Unterklassen prüfen mit dem Operator instanceof **260**

6.7 Methoden überschreiben 261
6.7.1 super: Aufrufen einer Methode aus der Oberklasse **262**
6.7.2 Nicht überschreibbare Funktionen **264**
6.7.3 Fehlende kovariante Rückgabewerte **265**

6.8 Die Oberste aller Klassen: Object 266
6.8.1 Klassenobjekte **266**
6.8.2 Hashcodes **267**
6.8.3 Objektidentifikation mit toString() **267**
6.8.4 Objektgleichheit mit equals() und Identität **267**
6.8.5 Klonen eines Objekts mit clone() **270**
6.8.6 Aufräumen mit finalize() **270**
6.8.7 Synchronisation **271**

6.9 Die Oberklasse gibt Funktionalität vor 271
6.9.1 Dynamisches Binden als Beispiel für Polymorphie **273**
6.9.2 Keine Polymorphie bei privaten, statischen und finalen Methoden **274**
6.9.3 Konstruktoren in der Vererbung **275**

6.10 Abstrakte Klassen 279
6.10.1 Abstrakte Klassen **279**
6.10.2 Abstrakte Methoden **280**
6.10.3 Über abstract final **283**

6.11 Schnittstellen 284
6.11.1 Die Mehrfachvererbung bei Schnittstellen **287**
6.11.2 Erweitern von Interfaces – Subinterfaces **288**
6.11.3 Vererbte Konstanten bei Schnittstellen **288**
6.11.4 Vordefinierte Methoden einer Schnittstelle **290**
6.11.5 CharSequence als Beispiel einer Schnittstelle **292**

6.12 Innere Klassen 294
6.12.1 Geschachtelte Top-Level Klassen und Schnittstellen **294**
6.12.2 Mitglieds- oder Elementklassen **295**
6.12.3 Lokale Klassen **298**
6.12.4 Anonyme innere Klassen **298**
6.12.5 Eine Sich-Selbst-Implementierung **301**
6.12.6 this und Vererbung **303**
6.12.7 Implementierung einer verketteten Liste **304**
6.12.8 Funktionszeiger **306**

6.13 Gegenseitige Abhängigkeiten von Klassen 307

6.14 Pakete 308

7 Exceptions 311

7.1 Problembereiche einzäunen 311
7.1.1 Exceptions in Java mit try und catch **311**
7.1.2 Ablauf einer Ausnahmesituation **313**
7.1.3 Wiederholung kritischer Bereiche **313**
7.1.4 throws im Methodenkopf angeben **314**
7.1.5 Abschließende Arbeiten mit finally **315**
7.1.6 Nicht erreichbare catch-Klauseln **316**

7.2 Die Klassenhierarchie der Fehler 317
7.2.1 Die Exception-Hierarchie **318**
7.2.2 Ober-Ausnahmen fangen **318**
7.2.3 Alles geht als Exception durch **319**
7.2.4 Ausnahmen, die nicht gefangen werden müssen: RuntimeException **320**

7.3 Werfen eigener Exceptions 321
7.3.1 Vorgefertigte Ausnahme-Objekte wieder verwenden **322**
7.3.2 Typecast auf ein null-Objekt für eine NullPointerException **323**
7.3.3 Neue Exception-Klassen definieren **323**

7.4 Rückgabewerte bei ausgelösten Ausnahmen 325

7.5 Ein Assert in Java 326

7.6 Sicherheitsfragen mit dem SecurityManager klären 330
7.6.1 Programm beenden **330**

8 Die Funktionsbibliothek 333

8.1 Die Java-Klassenphilosophie 333
8.1.1 Paketübersicht **333**

8.2 Wrapper-Klassen 338
8.2.1 Die Character-Klasse **339**
8.2.2 Die Boolean-Klasse **341**
8.2.3 Die Number-Klasse **343**
8.2.4 Die Klasse Integer **344**

8.2.5 Unterschiedliche Ausgabeformate **346**
8.2.6 Behandlung von Überlauf **348**

8.3 Ausführung von externen Programmen 348
8.3.1 DOS-Programme aufrufen **350**
8.3.2 Die Windows Registry verwenden **351**

8.4 Compilieren von Klassen 352
8.4.1 Der Sun-Compiler **352**

9 Threads und nebenläufige Programmierung 355

9.1 Prozesse und Threads 355
9.1.1 Wie parallele Programme die Geschwindigkeit heben können **356**

9.2 Threads erzeugen 358
9.2.1 Threads über die Schnittestelle Runnable implementieren **358**
9.2.2 Threads über Runnable starten **359**
9.2.3 Die Klasse Thread erweitern **360**
9.2.4 Erweitern von Thread oder implementieren von Runnable? **363**

9.3 Threads schlafen 363
9.3.1 Eine Zeituhr **365**

9.4 Die Klassen Timer und TimerTask 366

9.5 Die Zustände eines Threads 367
9.5.1 Das Ende eines Threads **367**
9.5.2 Einen Thread höflich mit Interrupt beenden **367**
9.5.3 Der stop() von außen **370**
9.5.4 Das ThreadDeath-Objekt **370**
9.5.5 Auf das Ende warten mit join() **371**

9.6 Arbeit niederlegen und wieder aufnehmen 373

9.7 Priorität 374
9.7.1 Threads hoher Priorität und das AWT **374**
9.7.2 Granularität und Vorrang **375**

9.8 Dämonen (engl. Daemon) 375

9.9 Kooperative und nicht-kooperative Threads 377

9.10 Synchronisation über kritische Abschnitte 378
9.10.1 Gemeinsam genutzte Daten **378**
9.10.2 Probleme beim gemeinsamen Zugriff und kritische Abschnitte **378**
9.10.3 Punkte parallel initialisieren **379**
9.10.4 i++ sieht atomar aus, ist es aber nicht **379**
9.10.5 Abschnitte mit synchronized schützen **380**
9.10.6 Monitore **381**
9.10.7 Synchronized-Methode am Beispiel der Klasse StringBuffer **382**
9.10.8 Synchronisierte Blöcke **382**
9.10.9 Vor- und Nachteile von synchronisierten Blöcken und Methoden **383**
9.10.10 Nachträglich synchronisieren **384**

9.10.11 PMonitore sind reentrant, gut für die Geschwindigkeit 384
9.10.12 Deadlocks 385

9.11 Variablen mit volatile kennzeichnen 387

9.12 Synchronisation über Warten und Benachrichtigen 389
9.12.1 Warten mit wait() und Aufwecken mit notify() 390
9.12.2 Mehrere Wartende und notifyAll() 391
9.12.3 wait() mit einer Zeitspanne 392
9.12.4 Beispiel Erzeuger-Verbraucher-Programm 392
9.12.5 *Semaphoren* 395

9.13 Grenzen von Threads 397

9.14 Aktive Threads in der Umgebung 397

9.15 Gruppen von Threads in einer Thread-Gruppe 398
9.15.1 Etwas über die aktuelle Thread-Gruppe herausfinden 398
9.15.2 Threads in einer Thread-Gruppe anlegen 400
9.15.3 Methoden von Thread und ThreadGroup im Vergleich 403

9.16 Einen Abbruch der virtuellen Maschine erkennen 404

10 Raum und Zeit 407

10.1 Greenwich Mean Time (GMT) 407

10.2 Wichtige Datum-Klassen im Überblick 409
10.2.1 Zeitzonen durch die Klasse TimeZone repräsentiert 409

10.3 Sprachen der Länder 411
10.3.1 Sprachen in Java über Locale-Objekte 411

10.4 Einfache Übersetzung durch ResourceBundle-Objekte 414

10.5 Die Klasse Date 416
10.5.1 Die Date-Klasse 416
10.5.2 Zeitmessung und Profiling 418

10.6 Die abstrakte Klasse Calendar 419

10.7 Der gregorianische Kalender 421

10.8 Formatieren der Datumsangaben 426
10.8.1 Mit DateFormat und SimpleDateFormat formatieren 427
10.8.2 Parsen von Datumswerten 434
10.8.3 Parsen und Formatieren ab bestimmten Positionen 436

11 Datenstrukturen und Algorithmen 437

11.1 Mit einem Iterator durch die Daten wandern 437
11.1.1 Bauernregeln aufzählen 438

11.2 Dynamische Datenstrukturen 440

11.3 Die Klasse Vector 440
11.3.1 Vektoren erzeugen 440

11.3.2 Funktionen **441**
11.3.3 Arbeitsweise des internen Arrays **443**
11.3.4 Die Größe eines Felds **444**
11.3.5 Eine Aufzählung und gleichzeitiges Verändern **444**
11.3.6 Die Funktionalität eines Vektors erweitern **446**

11.4 Stack, der Stapel 446
11.4.1 Die Methoden vom Stack **446**
11.4.2 Das oberste Stack-Element duplizieren **447**
11.4.3 Ein Stack ist ein Vektor – Aha! **448**

11.5 Die Klasse BitSet für Bitmengen 449
11.5.1 BitSet anlegen und füllen **449**
11.5.2 Mengenorientierte Operationen **450**
11.5.3 Funktionsübersicht **451**
11.5.4 Primzahlen in einem BitSet verwalten **452**

11.6 Die Klasse Hashtable und assoziative Speicher 452
11.6.1 Ein Objekt der Klasse Hashtable erzeugen **453**
11.6.2 Einfügen und Abfragen der Datenstruktur **453**
11.6.3 Die Arbeitsweise einer Hashtabelle **455**
11.6.4 Aufzählen der Elemente **457**
11.6.5 Ausgabe der Hashtabelle und Gleichheitstest **457**
11.6.6 Klonen **458**

11.7 Die abstrakte Klasse Dictionary 458
11.7.1 Zugriff und Abfrage **459**
11.7.2 Metainformationen **460**
11.7.3 Iterationen über die Elemente **460**

11.8 Die Properties-Klasse 460
11.8.1 Über die Klasse Properties **461**
11.8.2 put(), get() und getProperties() **462**
11.8.3 Eigenschaften ausgeben **463**
11.8.4 Systemeigenschaften der Java-Umgebung **463**
11.8.5 Browser-Version abfragen **464**
11.8.6 Properties von der Konsole setzen **465**

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 **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 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 Ein Design-Pattern durch Beobachten von Änderungen 495
11.14.1 Design-Pattern **496**
11.14.2 Das Beobachter-Pattern (Observer/Observable) **496**

12 Datenströme und Dateien 501

12.1 Dateien und Verzeichnisse 501
12.1.1 Dateien und Verzeichnisse mit der Klasse File **502**
12.1.2 Dateieigenschaften und -attribute **503**
12.1.3 Umbenennen, Verzeichnisse anlegen und Datei löschen **505**
12.1.4 Die Wurzel aller Verzeichnisse **506**
12.1.5 Verzeichnisse listen und Dateien filtern **507**
12.1.6 Implementierungsmöglichkeiten für die Klasse File **511**
12.1.7 Verzeichnisse nach Dateien rekursiv durchsuchen **512**

12.2 Dateien mit wahlfreiem Zugriff 514
12.2.1 Eine RandomAccessFile öffnen **514**
12.2.2 Aus dem RandomAccessFile lesen **515**
12.2.3 Hin und her in der Datei **517**
12.2.4 Die Länge des RandomAccessFile **517**

12.3 Übersicht über wichtige Stream- und Writer/Reader 518
12.3.1 Die abstrakten Basisklassen **520**

12.4 Eingabe- und Ausgabe-Klassen: InputStream und OutputStream 521
12.4.1 Die Klasse OutputStream **521**
12.4.2 Ein Datenschlucker **522**
12.4.3 Die Eingabeklasse InputStream **523**
12.4.4 Anwenden der Klasse FileInputStream **524**
12.4.5 Anwendung der Klasse FileOutputStream **526**
12.4.6 Kopieren von Dateien **526**
12.4.7 Daten filtern durch FilterInputStream und FilterOutputStream **528**
12.4.8 Der besondere Filter PrintStream **528**
12.4.9 System Standard-Ein- und Ausgabe und Input- bzw. PrintStreams **530**
12.4.10 Bytes in den Strom mit ByteArrayOutputStream **535**
12.4.11 Ströme zusammensetzen mit SequenceInputStream **535**

12.5 Die Unterklassen von Writer 538
12.5.1 Die abstrakte Klasse Writer **539**
12.5.2 Datenkonvertierung durch den OutputStreamWriter **540**
12.5.3 In Dateien schreiben mit der Klasse FileWriter **542**
12.5.4 StringWriter und CharArrayWriter **543**
12.5.5 Gepufferte Ausgabe durch BufferedWriter **546**
12.5.6 Ausgabemöglichkeiten durch PrintWriter erweitern **549**
12.5.7 Daten mit FilterWriter filtern **550**
12.5.8 Die abstrakte Basisklasse Reader **556**
12.5.9 Automatische Konvertierungen mit dem InputStreamReader **558**
12.5.10 Dateien lesen mit der Klasse FileReader **559**
12.5.11 StringReader und CharArrayReader **560**
12.5.12 Schachteln von Eingabe-Streams **562**
12.5.13 Gepufferte Eingaben mit der Klasse BufferedReader **562**
12.5.14 LineNumberReader zählt automatisch Zeilen mit **563**
12.5.15 Eingaben filtern mit der Klasse FilterReader **565**
12.5.16 Daten zurücklegen mit der Klasse PushbackReader **568**

12.6 Kommunikation zwischen Threads mit Pipes 570
12.6.1 PipedOutputStream und PipedInputStream **571**
12.6.2 PipedWriter und PipedReader **572**
12.6.3 Datenströme komprimieren **575**
12.6.4 Zip-Archive **578**

12.7 Prüfsummen 586
12.7.1 Die Schnittstelle Checksum **587**
12.7.2 Die Klasse CRC32 **587**
12.7.3 Die Adler32-Klasse **590**

12.8 Persistente Objekte und Serialisierung 591
12.8.1 Objekte speichern **592**
12.8.2 Objekte lesen **594**
12.8.3 Die Schnittstelle Serializable **596**
12.8.4 Ian Wilmut und tiefe Objektkopien **597**
12.8.5 Felder sind implizit Serializable **599**
12.8.6 Versionenverwaltung und die SUID **599**
12.8.7 Beispiele aus den Standard-Klassen **602**
12.8.8 Serialisieren in XML-Dateien **603**
12.8.9 JSX (Java Serialization to XML) **604**
12.8.10 XML-API von Sun **607**

12.9 Die Logging-API 609

13 Die eXtensible Markup Language (XML) 613

13.1 Auszeichnungssprachen 613
13.1.1 Die Standard Generalized Markup Language (SGML) **613**
13.1.2 Extensible Markup Language (XML) **614**

13.2 Eigenschaften von XML-Dokumenten 614
13.2.1 Elemente und Attribute **614**

13.2.2 Beschreibungssprache für den Aufbau von XML-Dokumenten **616**
13.2.3 Schema, eine Alternative zu DTD **619**
13.2.4 Namensraum (Namespace) **622**
13.2.5 XML-Applikationen **623**

13.3 Die Java APIs für XML 623
13.3.1 Das Document Object Model (DOM) **623**
13.3.2 Simple API for XML Parsing (SAX) **623**
13.3.3 Java Document Object Model (JDOM) **624**

13.4 *XML-Dateien mit JDOM verarbeiten* 624
13.4.1 JDOM beziehen **625**
13.4.2 Paketübersicht **625**
13.4.3 Die Document-Klasse **626**
13.4.4 Eingaben aus der Datei lesen **627**
13.4.5 Das Dokument als XML-Datei ausgeben **627**
13.4.6 Der Dokumententyp **628**
13.4.7 Elemente **628**
13.4.8 Zugriff auf Elementinhalte **631**
13.4.9 Liste mit Unterelementen erzeugen **633**
13.4.10 Neue Elemente einfügen und ändern **633**
13.4.11 Attributinhalte lesen und ändern **636**

13.5 JAXP als Java-Schnittstelle zu XML 639
13.5.1 Einführung in XSLT **639**
13.5.2 Umwandlung von XML-Dateien mit JDOM und JAXP **642**

14 Grafikprogrammierung mit dem AWT 645

14.1 Das abstrakte Window-Toolkit 645
14.1.1 Java Foundation Classes **645**

14.2 Fenster unter grafischen Oberflächen 646
14.2.1 Fenster öffnen **646**
14.2.2 Größe und Position des Fensters verändern **649**
14.2.3 Fenster und Dialog-Dekoration **650**

14.3 Das Toolkit 650
14.3.1 Einen Hinweis beepen **650**

14.4 Grundlegendes zum Zeichnen 651
14.4.1 Die paint()-Methode **651**
14.4.2 Auffordern zum Neuzeichnen mit repaint() **653**

14.5 Punkte, Linien und Rechtecke aller Art 653
14.5.1 Linien **654**
14.5.2 Rechtecke **654**

14.6 Alles was rund ist 655

14.7 Polygone und Polylines 657
14.7.1 Die Polygon-Klasse **657**
14.7.2 N-Ecke zeichnen **659**

14.7.3 Vollschlanke Linien zeichnen **660**

14.8 Zeichenketten schreiben 661
14.8.1 Einen neuen Zeichensatz bestimmen **662**
14.8.2 Zeichensätze des Systems ermitteln **664**
14.8.3 Die Klasse FontMetrics **666**
14.8.4 True Type Fonts **668**

14.9 Clipping-Operationen 670

14.10 Farben 673
14.10.1 Zufällige Farbblöcke zeichnen **674**
14.10.2 Farbanteile zurückgeben **676**
14.10.3 Vordefinierte Farben **676**
14.10.4 Farben aus Hexadezimalzahlen erzeugen **677**
14.10.5 Einen helleren oder dunkleren Farbton wählen **678**
14.10.6 Farbmodelle HSB und RGB **680**
14.10.7 Die Farben des Systems **680**

14.11 Bilder anzeigen und Grafiken verwalten 685
14.11.1 Eine Grafik zeichnen **687**
14.11.2 Grafiken zentrieren **689**
14.11.3 Laden von Bildern mit dem MediaTracker beobachten **690**
14.11.4 Kein Flackern durch Double-Buffering **694**
14.11.5 Bilder skalieren **696**

14.12 Programmicon setzen 698
14.12.1 VolatileImage **699**

14.13 Grafiken speichern 700
14.13.1 Bilder im GIF-Format speichern **700**
14.13.2 Gif speichern mit dem ACME-Paket **702**
14.13.3 JPEG-Dateien mit dem Sun-Paket schreiben **702**
14.13.4 Java Image Management Interface (JIMI) **705**

14.14 Von Produzenten, Konsumenten und Beobachtern 707
14.14.1 Producer und Consumer für Bilder **707**
14.14.2 Beispiel für die Übermittlung von Daten **708**
14.14.3 Bilder selbst erstellen **711**
14.14.4 Die Bildinformationen wieder auslesen **716**

14.15 Filter 718
14.15.1 Grundlegende Eigenschaft von Filtern **718**
14.15.2 Konkrete Filterklassen **719**
14.15.3 Mit CropImageFilter Teile ausschneiden **720**
14.15.4 Transparenz **721**

14.16 Alles wird bunt mit Farbmodellen 722
14.16.1 Die abstrakte Klasse ColorModel **722**
14.16.2 Farbwerte im Pixel mit der Klasse DirectColorModel **724**
14.16.3 Die Klasse IndexColorModel **726**

14.17 Drucken 730
14.17.1 Drucken mit dem einfachen Ansatz **730**

14.17.2 Ein PrintJob **731**
14.17.3 Drucken der Inhalte **732**
14.17.4 Komponenten drucken **733**
14.17.5 Den Drucker am Parallelport ansprechen **733**

14.18 Java 2D API 734
14.18.1 Grafische Objekte zeichnen **734**
14.18.2 Geometrische Objekte durch Shape gekennzeichnet **736**
14.18.3 Eigenschaften geometrischer Objekte **738**
14.18.4 Transformationen mit einem AffineTransform-Objekt **745**

14.19 Graphic Layers Framework 747

14.20 Grafikverarbeitung ohne grafische Oberfläche 747
14.20.1 Xvfb-Server **747**
14.20.2 Pure Java AWT Toolkit (PJA) **748**

15 Komponenten, Ereignisse und Container 749

15.1 Peer-Klassen und Lightweight-Komponenten 749

15.2 Es tut sich was – Ereignisse beim AWT 750
15.2.1 Was ist ein Ereignis? **750**
15.2.2 *Die Klasse AWTEvent* **750**
15.2.3 Events auf verschiedenen Ebenen **751**
15.2.4 Ereignisquellen, -senken und Horcher (Listener) **753**
15.2.5 Listener implementieren **753**
15.2.6 Listener bei Ereignisauslöser anmelden **754**

15.3 Varianten, das Fenster zu schließen 754
15.3.1 Eine Klasse implementiert die Schnittstelle WindowListener **755**
15.3.2 Adapterklassen nutzen **757**
15.3.3 Innere Mitgliedsklassen und innere anonyme Klassen **758**
15.3.4 Generic Listener **759**

15.4 Komponenten 759
15.4.1 Die Basis aller Komponenten: Die Klasse Components **759**
15.4.2 Proportionales Vergrößern eines Fensters **760**
15.4.3 Hinzufügen von Komponenten **762**

15.5 Ein Informationstext über die Klasse Label 763
15.5.1 Mehrzeiliger Text **766**

15.6 Eine Schaltfläche (Button) 767
15.6.1 Der aufmerksame ActionListener **769**
15.6.2 Generic Listener für Schaltflächen-Ereignisse verwenden **771**

15.7 Horizontale und vertikale Schieberegler 774
15.7.1 Der AdjustmentListener, der auf Änderungen hört **777**

15.8 Ein Auswahlmenü – Das Choice-Menü 778
15.8.1 *ItemListener* **780**

15.9 Eines aus vielen – Kontrollfeld (Checkbox) 782
15.9.1 Ereignisse über ItemListener **784**

15.10 Optionsfelder 784

15.11 List-Boxen 785

15.12 Texteingabefelder 788
15.12.1 Text in einer Eingabezeile **789**
15.12.2 Passwort-Felder **789**
15.12.3 Mehrzeilige Textfelder **789**

15.13 Menüs 792
15.13.1 Die Menüleisten und die Einträge **792**
15.13.2 Menüeinträge definieren **793**
15.13.3 Shortcuts **795**
15.13.4 Beispiel für ein Programm mit Menüleisten **796**

15.14 Popup-Menüs 799

15.15 Selbstdefinierte Cursor 802

15.16 Alles Auslegungssache: Die Layout-Manager 803
15.16.1 FlowLayout **804**
15.16.2 BorderLayout **806**
15.16.3 GridLayout **807**
15.16.4 Der GridBagLayout-Manager **809**

15.17 Dynamisches Layout während einer Größenänderung 814

15.18 Dialoge 814
15.18.1 Der Dateiauswahl-Dialog **814**

15.19 Die Zwischenablage (Clipboard) 817

15.20 Ereignisverarbeitung auf unterster Ebene 820

15.21 Benutzerinteraktionen automatisieren 821

16 Let's Swing 825

16.1 Das Konzept vom Model-View-Controller 825

16.2 Der Unterschied zwischen AWT und Swing 826
16.2.1 Schließen eines Swing-Fensters **827**

16.3 JLabel 827

16.4 Die Klasse ImageIcon 828
16.4.1 Die Schnittstelle Icon **831**
16.4.2 Was Icon und Image verbindet **832**

16.5 Die Schaltflächen von Swing 833
16.5.1 JButton **833**
16.5.2 AbstractButton **834**
16.5.3 JToggleButton **836**
16.5.4 JCheckBox **836**
16.5.5 Radiogruppen **837**

16.6 **Tooltips** 837
16.7 **JScrollBar** 838
16.8 **JSlider** 838
16.9 **JList** 840
16.10 **JComboBox** 841
16.11 **Der Fortschrittsbalken JProgressBar** 842
16.12 **Symbolleisten alias Toolbars** 843
16.13 **Texteingaben** 844
16.13.1 JPasswordField **844**
16.13.2 Die Editor-Klasse JEditorPane **844**
16.14 **Rahmen (Borders)** 845
16.15 **Dialoge** 847
16.15.1 Der Farbauswahl-Dialog JColorChooser **848**
16.16 **Mausradunterstützung** 849
16.17 **Der Inhalt einer Zeichenfläche: JPanel** 851
16.18 **JRootPane und JLayeredPane** 852
16.19 **Tabellen mit JTable** 852
16.19.1 Ein eigenes Modell **853**
16.19.2 AbstractTableModel **854**
16.19.3 DefaultTableModel **857**
16.19.4 Einen eigenen Renderer für Tabellen **858**
16.20 **AWT, Swing und die Threads** 861
16.20.1 Warum Swing nicht threadsicher ist **861**
16.20.2 Swing-Elemente bedienen mit invokeLater(), invokeAndWait() **864**
16.21 **Das Java Look&Feel** 865

17 **Netzwerkprogrammierung** 867

17.1 **Grundlegende Begriffe** 867
17.1.1 Internet-Standards und RFC **867**
17.2 **URL-Verbindungen** 868
17.2.1 URL-Objekte erzeugen **869**
17.2.2 Informationen über eine URL **871**
17.2.3 Der Zugriff auf die Daten über die Klasse URL **874**
17.3 **Die Klasse URLConnection** 876
17.3.1 Methoden und Anwendung von URLConnection **876**
17.3.2 Protokoll- und Content-Handler **879**
17.3.3 Im Detail: Von URL zu URLConnection **880**
17.4 **Das Common Gateway Interface** 881
17.4.1 Parameter für ein CGI-Programm **882**
17.4.2 Codieren der Parameter für CGI-Programme **882**

17.4.3 Eine Suchmaschine ansprechen **884**

17.5 Host-Adresse und IP-Adressen 885
17.5.1 IPv6 für Java mit Jipsy **888**

17.6 Socket-Programmierung 888
17.6.1 Das Netzwerk ist der Computer **889**
17.6.2 Standarddienste unter Windows nachinstallieren **890**
17.6.3 Stream-Sockets **891**
17.6.4 Informationen über den Socket **894**
17.6.5 Mit telnet an den Ports horchen **894**
17.6.6 Ein kleines Ping – lebt der Rechner noch? **894**

17.7 Client/Server-Kommunikation 895
17.7.1 Ein Multiplikations-Server **897**

17.8 Webprotokolle mit NetComponents nutzen 899

17.9 E-Mail verschicken 899
17.9.1 Wie eine E-Mail um die Welt geht **899**
17.9.2 Übertragungsprotokolle **900**
17.9.3 Das Simple Mail Transfer Protocol **903**
17.9.4 Demoprogramm, welches eine E-Mail abschickt **906**

17.10 Arbeitsweise eines Web-Servers 907
17.10.1 Das Hypertext Transfer Protocol (HTTP) **907**
17.10.2 Anfragen an den Server **908**
17.10.3 Die Antworten vom Server **910**

17.11 Datagram-Sockets 913
17.11.1 Die Klasse DatagramSocket **915**
17.11.2 Datagramme und die Klasse DatagramPacket **917**
17.11.3 Auf ein hereinkommendes Paket warten **917**
17.11.4 Ein Paket zum Senden vorbereiten **919**
17.11.5 Methoden der Klasse DatagramPacket **919**
17.11.6 Das Paket senden **920**
17.11.7 Die Zeitdienste und ein eigener Server und Client **921**

17.12 Internet Control Message Protocol (ICMP) 924
17.12.1 Ping **924**

17.13 Multicast-Kommunikation 925

18 Servlets und Java Server Pages 927

18.1 Dynamische Web-Seiten und Servlets 927
18.1.1 Was Servlets sind **927**
18.1.2 Vorteil von Servlets gegenüber CGI-Programmen **928**
18.1.3 Das erste Servlet **929**

18.2 Vom Client zum Server und wieder zurück 929
18.2.1 Der bittende Client **929**
18.2.2 Was ein Webserver erzeugt **931**
18.2.3 Wer oder was ist MIME? **932**

18.3 Servlets entwickeln und testen 933
18.3.1 Servlet-Container **933**
18.3.2 Webserver mit Servlet-Funktionalität **933**
18.3.3 Tomcat **935**
18.3.4 Das erste Servlet compilieren und ausführen **936**

18.4 Der Lebenszyklus 938
18.4.1 Initialisierung in init() **939**
18.4.2 Abfragen bei service() **941**
18.4.3 Mehrere Anfragen beim Servlet und Threadsicherheit **942**
18.4.4 Das Ende eines Servlets **943**

18.5 Das HttpServletResponse-Objekt 943
18.5.1 Wir generieren eine Web-Seite **943**
18.5.2 Binärdaten senden **945**
18.5.3 Automatisches Neuladen **946**
18.5.4 Komprimierte Daten mit Content-Encoding **947**
18.5.5 Noch mehr über Header, die der Server setzt **948**
18.5.6 Seiten umlenken **949**

18.6 Was der Browser mit auf den Weg gibt – HttpServletRequest 950
18.6.1 Hilfsfunktion im Umgang mit Headern **952**
18.6.2 Übersicht der Browser-Header **952**
18.6.3 Formulardaten auslesen **954**

18.7 Kleine Kekse: die Klasse Cookies 956
18.7.1 Kekse erzeugen und setzen **957**
18.7.2 Cookies vom Servlet einlesen **957**
18.7.3 Kleine Helfer für Cookies **959**
18.7.4 Cookiestatus ändern **960**
18.7.5 Langlebige Cookies **961**
18.7.6 Ein Warenkorbsystem **961**

18.8 Sitzungsverfolgung (Session Tracking) 963
18.8.1 Das mit einer Sitzung verbundene Objekt HttpSession **964**
18.8.2 Werte mit einer Sitzung assoziieren und auslesen **965**
18.8.3 *Zusätzliche Informationen* **966**

18.9 URL-Rewriting 969

18.10 Weiterleitung und Einbinden von Servlet-Inhalten 970

18.11 Inter-Servlet-Kommunikation 971
18.11.1 Daten zwischen Servlets teilen **971**

18.12 Internationalisierung 972
18.12.1 Die Länderkennung des Anfragers auslesen **972**
18.12.2 Länderkennung für die Ausgabe setzen **972**
18.12.3 Westeuropäische Texte senden **972**

18.13 Java Server Pages (JSP) 974
18.13.1 Was Java Server Pages sind **974**

18.14 Skript-Elemente 975
18.14.1 Kommentare und Quoting **976**

18.14.2 Scriptlets **976**
18.14.3 Vordefinierte Variablen **977**
18.14.4 Ausdrücke **978**
18.14.5 Deklarationen **978**

18.15 Entsprechende XML-Tags 979

18.16 Direktiven 979
18.16.1 Wichtige page-Direktiven im Überblick **979**

18.17 Aktionen 981
18.17.1 Beans **981**

18.18 Sonstiges zu den Servern 982
18.18.1 Den internen Compiler bei Tomcat für JSP ändern **982**

18.19 Tomcat: Spezielles 982
18.19.1 Tomcat als Service unter Windows NT ausführen **982**
18.19.2 MIME-Types mit Tomcat verbinden **982**
18.19.3 Servlets beim Start laden **983**

18.20 Ein Servlet generiert WAP-Seiten für das Handy 983
18.20.1 Ein WAP-Handy simulieren **984**
18.20.2 Übersicht der wichtigsten Tags **985**
18.20.3 Der Gateway **986**
18.20.4 WML-Seiten aufbauen **987**
18.20.5 Interessante Links zum Thema Servlets/JSP **988**

18.21 Text in HTML-konformen Text umwandeln 989

19 RMI 991

19.1 Entfernte Methoden 991
19.1.1 Wie entfernte Methoden arbeiten **991**
19.1.2 Stellvertreter (Proxy) **991**
19.1.3 Wie die Stellvertreter die Daten übertragen **992**
19.1.4 Probleme mit entfernten Methoden **993**

19.2 Nutzen von RMI bei Middleware-Lösungen 994

19.3 Die Lösung für Java ist RMI 995
19.3.1 Entfernte Objekte programmieren **995**
19.3.2 Entfernte und lokale Objekte im Vergleich **995**
19.3.3 RMI und CORBA **996**

19.4 Definition einer entfernten Schnittstelle 996

19.5 Das entfernte Objekt 997
19.5.1 Der Bauplan für entfernte Objekte **997**
19.5.2 Der Konstruktor **998**
19.5.3 Implementierung der entfernten Methoden **1000**
19.5.4 UnicastRemoteObjekt, RemoteServer und RemoteObject **1000**

19.6 Stellvertreterobjekte erzeugen 1001
19.6.1 Das Dienstprogramm rmic **1002**

19.7 Der Namendienst (Registry) 1002
19.7.1 Der Port **1003**

19.8 Der Server: Entfernte Objekte beim Namensdienst anmelden 1004
19.8.1 Automatisches Anmelden bei Bedarf **1005**

19.9 Einen Client programmieren 1005
19.9.1 Einfaches Logging **1006**

19.10 Aufräumen mit dem DGC 1007

19.11 Entfernte Objekte übergeben und laden 1007
19.11.1 Klassen vom RMI-Klassenlader nachladen **1008**
19.11.2 Sicherheitsmanager **1008**

19.12 Registry wird vom Server gestartet 1010

19.13 RMI über die Firewall 1010
19.13.12RMI über HTTP getunnelt **1010**

19.14 Java API für XML Messaging (JAXM) 1011

19.15 Java Message Service (JMS) 1011

20 Applets 1013

20.1 Applets und Applikationen – wer darf was 1013

20.2 Das erste Hallo-Applet 1013

20.3 Die Zyklen eines Applets 1014

20.4 Parameter an das Applet übergeben 1014
20.4.1 Wie das Applet den Browserinhalt ändern kann **1015**
20.4.2 Den Ursprung des Applets erfragen **1016**
20.4.3 Was ein Applet alles darf **1017**

20.5 Musik in einem Applet 1017
20.5.1 Fest verdrahtete Musikdatei **1018**
20.5.2 Variable Musikdatei über einen Parameter **1018**
20.5.3 WAV- und MIDI-Dateien abspielen **1019**

20.6 Browserabhängiges Verhalten 1020
20.6.1 Java im Browser aktiviert? **1020**
20.6.2 Läuft das Applet unter Netscape oder Microsoft Explorer? **1020**
20.6.3 Datenaustausch zwischen Applets und Java Skripten **1021**

20.7 Applets und Applikationen kombinieren 1022

20.8 Datenaustausch zwischen Applets 1022

20.9 Webstart 1025

21 Datenbankmanagement mit JDBC 1027

21.1 Das relationale Modell 1027
21.1.1 Relationale und objektorientierte Datenbanken **1028**

21.2 JDBC: Der Zugriff auf Datenbanken über Java 1028

21.3 Die Rolle von SQL 1029
21.3.1 Ein Rundgang durch SQL-Anfragen **1029**
21.3.2 Datenabfrage mit der Data Query Language (DQL) **1030**

21.4 Datenbanktreiber für den Zugriff 1033

21.5 Datenbanken und ihre Treiber 1034
21.5.1 Datenbank Interbase **1034**
21.5.2 Interbase JDBC-Treiber **1035**
21.5.3 Die Datenbank Microsoft Access **1036**
21.5.4 Die JDBC-ODBC Bridge **1036**
21.5.5 ODBC einrichten und Access damit verwenden **1037**
21.5.6 Oracle8i Enterprise Edition **1038**

21.6 Eine Beispiel-Abfrage 1038

21.7 Mit Java an eine Datenbank andocken 1039
21.7.1 Der Treibermanager **1040**
21.7.2 Eine Aufzählung aller Treiber **1040**
21.7.3 Log-Informationen **1041**
21.7.4 Den Treiber laden **1042**
21.7.5 Wie Treiber programmiert sind **1042**
21.7.6 Verbindung zur Datenbank **1045**

21.8 Datenbankabfragen 1047
21.8.1 Abfragen über das Statement-Objekt **1047**
21.8.2 Ergebnisse einer Abfrage im ResultSet **1048**

21.9 Java und SQL-Datentypen 1049
21.9.1 Die getXXX()-Methoden **1050**

21.10 Transaktionen 1052

21.11 Elemente einer Datenbank hinzufügen und aktualisieren 1053
21.11.1 Batch-Updates **1053**

21.12 Vorbereitete Anweisungen (Prepared Statements) 1054
21.12.1 PreparedStatement-Objekte vorbeiten **1055**
21.12.2 Werte für die Platzhalter eines PreparedStatement **1055**

21.13 Metadaten 1057
21.13.1 Metadaten über die Tabelle **1057**
21.13.2 Informationen über die Datenbank **1060**

21.14 Die Ausnahmen bei JDBC 1062

21.15 Java Data Objects (JDO) 1062

22 Reflection 1065

22.1 Einfach mal reinschauen 1065

22.2 Mit dem Class-Objekt etwas über Klassen erfahren 1065
22.2.1 An ein Class-Objekt kommen **1066**

22.2.2 Was das Class-Objekt beschreibt **1069**
22.2.3 Der Name der Klasse **1070**
22.2.4 Oberklassen finden **1072**
22.2.5 Implementierte Interfaces einer Klasse oder eines Inferfaces **1073**
22.2.6 Modifizierer und die Klasse Modifier **1073**
22.2.7 Die Attribute einer Klasse **1076**
22.2.8 Methoden **1079**
22.2.9 Konstruktoren einer Klasse **1082**

22.3 Objekte manipulieren 1083
22.3.1 Objekte erzeugen **1083**
22.3.2 Die Belegung der Variablen erfragen **1085**
22.3.3 Variablen setzen **1087**

22.4 Methoden aufrufen 1089
22.4.1 Dynamische Methodenaufrufe bei festen Methoden beschleunigen **1090**

23 Komponenten durch Bohnen 1093

23.1 Grundlagen der Komponententechnik 1093
23.1.1 Brauchen wir überhaupt Komponenten? **1093**
23.1.2 Visuelle und nicht-visuelle Komponenten **1094**
23.1.3 Andere Komponententechnologien – oder was uns Microsoft brachte **1094**

23.2 Das Java-Beans Development Kit (BDK) 1095
23.2.1 Eine Beispielsitzung im BDK **1096**
23.2.2 Verknüpfungen zwischen Komponenten **1097**
23.2.3 Beans speichern **1098**

23.3 Die kleinste Bohne der Welt 1098

23.4 Jar-Archive für Komponenten 1099

23.5 Worauf JavaBeans basieren 1100

23.6 Eigenschaften 1101
23.6.1 Einfache Eigenschaften **1101**
23.6.2 Boolesche Eigenschaften **1102**
23.6.3 Indizierte Eigenschaften **1103**

23.7 Ereignisse 1104
23.7.1 Multicast und Unicast **1104**
23.7.2 Namenskonvention **1104**

23.8 Noch mehr Eigenschaften 1107
23.8.1 Gebundene Eigenschaften **1107**
23.8.2 Anwendung von PropertyChange bei AWT-Komponenten **1109**
23.8.3 Veto-Eigenschaften. Dagegen! **1110**

23.9 Bean-Eingenschaften anpassen 1111

23.10 Property-Editoren 1112

23.11 **BeanInfo 1112**

23.12 **Beliebte Fehler 1113**

24 Java Native Interface (JNI) 1115

24.1 **Java Native Interface und Invocation-API 1115**

24.2 **Die Schritte zur Einbindung einer C-Funktion in ein Java-Programm 1116**

24.2.1 Schreiben des Java-Codes **1116**

24.2.2 Kompilieren des Java-Codes **1116**

24.2.3 Erzeugen der Header-Datei **1117**

24.2.4 Implementierung der Methode in C **1118**

24.2.5 Übersetzen der C-Programme und Erzeugen der dynamischen Bibliothek **1118**

24.2.6 Setzen der Umgebungsvariable **1119**

24.3 **Erweiterung unseres Programms 1119**

25 Sicherheitskonzepte 1121

25.1 **Der Sandkasten (Sandbox) 1121**

25.2 **Sicherheitsmanager (Security Manager) 1121**

25.2.1 Der Sicherheitsmanager bei Applets **1123**

25.2.2 Sicherheitsmanager aktivieren **1126**

25.2.3 Der Sicherheitsmanager in den Java-Bibliotheken **1126**

25.2.4 Ein eigener Sicherheitsberater **1127**

25.2.5 Übersicht über die Methoden **1131**

25.3 **Klassenlader (Class Loader) 1134**

25.3.1 Wie die Klasse mit der Methode main() heißt **1134**

25.4 **Digitale Unterschriften 1135**

25.4.1 Die MDx-Reihe **1136**

25.4.2 Secure Hash AlgorithmSecure Hash Algorithm (SHASHA) **1137**

25.4.3 Mit der Security API einen Fingerabdruck berechnen **1137**

25.4.4 Die Klasse MessageDigest **1138**

25.4.5 Unix-Crypt **1141**

26 Dienstprogramme für die Java-Umgebung 1143

26.1 **Die Werkzeuge im Überblick 1143**

26.2 **Der Compiler javac 1143**

26.2.1 Der Java-Interpreter java **1144**

26.3 **Das Archivformat Jar 1145**

26.3.1 Das Dienstprogramm Jar benutzen **1146**

26.3.2 Das Manifest **1148**

26.3.3 Jar-Archive für Applets und Applikation **1149**

26.4 Mit JavaDoc und Doclets dokumentieren 1150
26.4.1 Mit JavaDoc Dokumentationen erstellen **1150**
26.4.2 Wie JavaDoc benutzt wird **1151**
26.4.3 Dokumentation erstellen **1153**
26.4.4 JavaDoc und Doclets **1156**
26.4.5 Doclets programmieren **1156**
26.4.6 Das Standard-Doclet **1157**

26.5 Dienstprogramme zur Signierung von Applets 1160
26.5.1 keytool **1160**
26.5.2 jarsigner **1161**
26.5.3 policytool **1162**

26.6 Konverter von Java nach C 1163
26.6.1 Toba **1163**
26.6.2 Arbeitsweise von Toba **1164**
26.6.3 Abstriche des Konverters **1164**

26.7 Konverter von Java Byte Code in ein Windows-Exe mit JET 1165

26.8 Manteln von Javaklassen in ein Windows-Exe mit JToExe 1165

26.9 Decompiler 1166
26.9.1 Jad, ein schneller Decompiler **1167**
26.9.2 SourceAgain **1169**
26.9.3 Decompilieren erschweren **1170**

26.10 Obfuscate Programm RetroGuard 1171

26.11 Source-Code Beautifier 1171

27 Style-Guide 1173

27.1 Programmierrichtlinien 1173

27.2 Allgemeine Richtlinien 1173

27.3 Quellcode kommentieren 1174
27.3.1 Bemerkungen über JavaDoc **1176**
27.3.2 Gotcha-Schlüsselwörter **1177**

27.4 Bezeichnernamen 1178
27.4.1 Ungarische Notation **1178**
27.4.2 Vorschlag für die Namensgebung **1178**

27.5 Formatierung 1180
27.5.1 Einrücken von Programmcode – die Vergangenheit **1180**
27.5.2 Verbundene Ausdrücke **1181**
27.5.3 Kontrollierter Datenfluss **1181**
27.5.4 Funktionen **1182**

27.6 Ausdrücke 1184

27.7 Anweisungen 1185
27.7.1 Schleifen **1185**
27.7.2 Switch, Case und Durchfallen **1186**

27.8 Klassen 1187

27.9 Zugriffsrechte 1188

27.9.1 Accessors/Zugriffsmethoden 1188

27.10 Verweise 1189

A Die Java-Grammatik 1191

A.1 Die lexikalische Struktur 1191

A.2 Typen, Werte und Variablen 1191

A.3 Bezeichner 1192

A.4 Pakete 1192

A.5 Produktionen für die LALR(1) Grammatik 1193

A.6 Klassen 1193

A.7 Statische Initialisierungen 1195

A.8 Konstruktoren 1195

A.9 Schnittstellen 1196

A.10 Felder 1196

A.11 Blöcke und Anweisungen 1196

A.12 Ausdrücke 1200

B Quellenverzeichnis 1205

Index 1211