

# Auf einen Blick

Vorwort .....	11
1 Grundlagen .....	19
2 Entwicklungsumgebungen .....	73
3 Grafische Benutzungsoberflächen (GUI) .....	117
4 Ausführbare Programme .....	173
5 Portable Programmierung .....	285
6 Mac OS X-spezifische Programmierung .....	307
7 Grafik und Multimedia .....	351
8 Werkzeuge .....	383
9 Datenbanken und JDBC .....	421
10 Servlets und JavaServer Pages (JSP) .....	443
11 J2EE und Enterprise JavaBeans (EJB) .....	461
12 J2ME und MIDP .....	481
A Kurzeinführung in die Programmiersprache Java .....	509
B Java auf Mac OS 8/9/Classic .....	531
C Java 1.5 »Tiger« .....	541
D System-Properties .....	547
E VM-Optionen .....	565
F Xcode- und Project Builder-Einstellungen .....	577
G Mac OS X- und Java-Versionen .....	585
H Glossar .....	589
I Die Buch-CD .....	595
Index .....	597

# Inhalt

<b>Vorwort</b>	<b>11</b>
<b>1 Grundlagen</b>	<b>19</b>
1.1 Historisches .....	22
1.2 Aktuelles .....	24
1.3 Den Mac richtig bedienen .....	25
1.4 Eingeben, übersetzen, ausführen .....	35
1.5 Entwicklerwerkzeuge und -dokumentation .....	42
1.6 Up to date bleiben .....	44
1.7 Mac OS X erkennen .....	47
1.8 Konfigurationsdateien speichern .....	52
1.9 Aufbau des Apple-Java-Systems .....	55
1.9.1 Java-Struktur anderer Systeme .....	57
1.9.2 Java-Struktur von Mac OS X .....	60
1.10 Kleine Fallen .....	69
1.11 Literatur & Links .....	70
<b>2 Entwicklungsumgebungen</b>	<b>73</b>
2.1 Project Builder und Xcode .....	76
2.1.1 Projekttypen .....	76
2.1.2 Projektstruktur .....	79
2.1.3 Eingeben und übersetzen .....	82
2.1.4 Ausführen und debuggen .....	87
2.1.5 Build-System .....	90
2.1.6 Versionsverwaltung .....	94
2.2 Eclipse .....	98
2.3 NetBeans und Sun Java Studio Creator .....	103
2.4 IntelliJ IDEA .....	105
2.5 OmniCore CodeGuide .....	106
2.6 Borland JBuilder .....	107
2.7 Borland Together Control Center und Together Solo .....	108
2.8 Oracle JDeveloper .....	108
2.9 Metrowerks CodeWarrior .....	110

2.10	jEdit .....	111
2.11	Jext .....	112
2.12	JJEdit .....	113
2.13	BlueJ .....	113
2.14	DrJava .....	115
2.15	Literatur & Links .....	116

### **3 Grafische Benutzungsoberflächen (GUI) 117**

3.1	Das AWT und die Java 1.3-Apple-Erweiterungen .....	121
3.1.1	Eine einfache AWT-Anwendung .....	122
3.1.2	Fensterhintergrund .....	134
3.1.3	Kontext-Popup-Menüs .....	135
3.1.4	Dateiauswahl-Dialoge .....	136
3.1.5	Portables Einbinden der Java 1.3-Apple-Erweiterungen .....	139
3.2	Swing und die Java 1.4-Apple-Erweiterungen .....	142
3.2.1	Konfiguration des Aussehens (Look & Feel) .....	144
3.2.2	Ein Swing-Beispiel mit mehreren Dokument-Fenstern .....	146
3.2.3	Portables Einbinden der Java 1.4-Apple-Erweiterungen .....	163
3.2.4	Hinweise zu diversen Swing-Komponenten .....	165
3.3	Fenster mit »Brushed Metal«-Aussehen .....	167
3.4	Drag & Drop .....	169
3.5	»Headless«-Applikationen ohne Benutzungsoberfläche .....	170
3.6	Literatur & Links .....	171

### **4 Ausführbare Programme 173**

4.1	Shell-Skripte im Terminal .....	176
4.2	Doppelklickbare JAR-Archive .....	182
4.3	Mac OS X-Applikationen .....	187
4.3.1	Programmsymbole .....	188
4.3.2	Programmpakete .....	190
4.3.3	Jar Bundler .....	193
4.3.4	Project Builder, Xcode und Eclipse .....	203
4.3.5	Programmpakete von Hand erzeugen .....	206
4.4	Installationswerkzeuge .....	219
4.4.1	Mac OS X-Hausmittel .....	220
4.4.2	Kommerzielle Installer .....	227
4.5	Java Web Start .....	234
4.6	Applets .....	250
4.6.1	Applets programmieren und einsetzen .....	251
4.6.2	Web-Browser .....	265
4.6.3	Kommunikation mit dem Browser .....	271
4.7	Literatur & Links .....	282

## **5 Portable Programmierung 285**

5.1	Strategien zur portablen Programmierung .....	288
5.2	Häufige Problembereiche .....	292
5.2.1	Datei- und Pfadtrennzeichen .....	292
5.2.2	Zeilenenden .....	294
5.2.3	Zeichenkodierung .....	295
5.2.4	Runtime.exec() .....	297
5.3	Oft benötigte Lösungen .....	298
5.3.1	PDF-Dokumente .....	298
5.3.2	Web-Browser und HTML-Dokumente .....	299
5.3.3	Zugriff auf serielle Schnittstellen .....	301
5.3.4	Hochauflösende Zeitmessung .....	301
5.3.5	Rendezvous .....	302
5.4	Literatur & Links .....	305

## **6 Mac OS X-spezifische Programmierung 307**

6.1	JNI .....	309
6.1.1	JNI-Bibliotheken mit Xcode erzeugen .....	310
6.1.2	Installationsverzeichnisse für JNI-Bibliotheken .....	317
6.1.3	JNI-Bibliotheken mit gcc erzeugen .....	318
6.1.4	Abhängigkeiten von anderen Bibliotheken .....	319
6.1.5	Laufzeitumgebungen (JVMs) erzeugen .....	321
6.2	JDirect und JManager .....	324
6.2.1	JManager .....	325
6.2.2	JDirect .....	325
6.3	Zugriff auf Datei-Metadaten .....	327
6.4	Cocoa Java .....	330
6.5	AppleScript .....	340
6.5.1	AppleScript mit Cocoa Java .....	340
6.5.2	AppleScript mit Shell-Kommandos .....	342
6.6	Speech & Spelling Frameworks .....	343
6.6.1	Speech Framework .....	343
6.6.2	Spelling Framework .....	345
6.7	Weitere systemabhängige Bibliotheken .....	347
6.7.1	Posix-Aufrufe mit dem Suite/P Toolkit .....	347
6.7.2	Authorisation Toolkit .....	348
6.8	Literatur & Links .....	349

## **7 Grafik und Multimedia 351**

7.1	Java Advanced Imaging (JAI) und Java Image I/O (JIO) .....	353
-----	--	-----

7.2	Java 3D .....	356
7.3	OpenGL/JOGL .....	358
7.4	Java Media Framework (JMF) .....	362
7.5	QuickTime for Java (QTJava) .....	365
	7.5.1 QTJava 6.0 (QuickTime bis Version 6.3) .....	367
	7.5.2 QTJava 6.1 (QuickTime ab Version 6.4) .....	371
7.6	Sound/Musik .....	374
7.7	Drucken .....	375
7.8	Literatur & Links .....	380

## **8 Werkzeuge 383**

8.1	Ant und Maven .....	385
	8.1.1 Ant in Xcode .....	388
	8.1.2 Ant in Xcode 1.5 .....	393
	8.1.3 Ant in Eclipse .....	394
	8.1.4 Programmpakete mit Ant erzeugen .....	394
	8.1.5 Maven .....	396
8.2	JUnit .....	398
	8.2.1 JUnit in Xcode .....	401
	8.2.2 JUnit in Eclipse .....	402
8.3	Decompiler .....	403
8.4	Obfuscators .....	404
8.5	Bytecode Viewer (Disassembler) .....	406
8.6	Profiler .....	408
8.7	JavaBrowser .....	410
8.8	Jikes .....	412
8.9	Groovy .....	413
8.10	UML-Modellierung .....	416
8.11	<oxygen/> XML-Editor .....	417
8.12	Literatur & Links .....	418

## **9 Datenbanken und JDBC 421**

9.1	SQL .....	424
9.2	JDBC .....	426
	9.2.1 Datensätze lesen .....	427
	9.2.2 Datensätze schreiben .....	430
9.3	Datenbanken .....	431
	9.3.1 MySQL .....	432
	9.3.2 PostgreSQL .....	438
	9.3.3 Firebird .....	439

9.3.4	HSQLDB .....	440
9.3.5	Oracle .....	440
9.3.6	Sybase Adaptive Server Enterprise .....	441
9.3.7	Berkeley DB Java Edition .....	441
9.4	ODBC .....	441
9.5	Literatur & Links .....	442

## **10 Servlets und JavaServer Pages (JSP) 443**

10.1	Servlets .....	446
10.2	Tomcat .....	448
10.3	JavaServer Pages (JSP) .....	453
10.4	Webapplikationen .....	455
10.5	Literatur & Links .....	459

## **11 J2EE und Enterprise JavaBeans (EJB) 461**

11.1	JBoss .....	464
11.2	Enterprise JavaBeans (EJB) .....	467
11.2.1	Eine einfache Enterprise-Applikation .....	468
11.2.2	XDoclet .....	477
11.3	Literatur & Links .....	478

## **12 J2ME und MIDP 481**

12.1	MIDlets entwickeln und testen .....	485
12.2	Einsatz im mobilen Endgerät .....	496
12.2.1	Lokale Installation .....	497
12.2.2	Installation vom Web-Server .....	502
12.3	Benutzungsoberflächen und Grafik .....	503
12.4	Literatur & Links .....	508

## **A Kurzeinführung in die Programmiersprache Java 509**

A.1	Programme, Klassen, Pakete .....	509
A.2	Methoden und Anweisungen .....	513
A.3	Variablen und Datentypen .....	515
A.4	Fallunterscheidungen und Schleifen .....	516
A.5	Klassen und Objekte .....	518

A.5.1	Vererbung .....	520
A.5.2	Assoziationen .....	522
A.5.3	Polymorphismus .....	523
A.5.4	Abstrakte Klassen und Interfaces .....	524
A.5.5	Innere und anonyme Klassen .....	526
A.6	Fehlerbehandlung mit Exceptions .....	528
A.7	Literatur & Links .....	530

## **B Java auf Mac OS 8/9/Classic 531**

## **C Java 1.5 »Tiger« 541**

## **D System-Properties 547**

D.1	Vordefinierte System-Properties .....	547
D.2	Konfigurierbare System-Properties .....	549

## **E VM-Optionen 565**

## **F Xcode- und Project Builder-Einstellungen 577**

F.1	Projekteinstellungen .....	578
F.2	Installationseinstellungen .....	578
F.3	Target-Einstellungen .....	579
F.4	Java-Einstellungen .....	580
F.5	Programmpaket-Einstellungen .....	582
F.6	Standardpfade .....	583

## **G Mac OS X- und Java-Versionen 585**

## **H Glossar 589**

## **I Die Buch-CD 595**

## **Index 597**

## Vorwort

*»Für die meiste Entwicklungsarbeit verwende ich mein PowerBook. Ich halte es für beträchtlich effizienter als ein Desktop-System, denn zum einen hat es alle Möglichkeiten eines vollständigen UNIX-Desktops, zum anderen kann ich es mitnehmen, weil es alle Laptop-Eigenschaften besitzt. Ich kann im Flugzeug arbeiten, zu Hause oder in irgendeiner Ecke, während ich in einer langweiligen Besprechung sitze. Und man kann damit nicht nur e-mailen und surfen, es besitzt voll funktionale, professionelle Entwicklungswerkzeuge.«  
(James Gosling, Erfinder der Java-Technologie)<sup>1</sup>*

### Liebe Leserin, lieber Leser!

Wenn der Erfinder der Java-Technologie so lobende Worte über ein Apple-System verliert, wird man hellhörig – was macht die Faszination an Apples Betriebssystem Mac OS X aus, dass in letzter Zeit immer mehr Entwickler auf diese Plattform umsteigen oder sie zumindest als zusätzliche Plattform nutzen? Und das bei einem System, dem vor ein paar Jahren kaum ein Außenstehender mehr zugetraut hat, als ein paar nette Grafiken zu bearbeiten – und dessen Maus sowieso (mindestens) eine Taste fehlt?

Zum einen bestechen Apple-Rechner seit Jahren durch gutes Design, das viele kleine Raffinessen bietet. Dadurch, dass Apple sowohl die Computer als auch das Betriebssystem und wichtige Anwendungen entwickelt, sind Hardware und Software perfekt aufeinander abgestimmt. Die einfache Bedienbarkeit des klassischen Mac OS ist legendär, und aktuelle Mac OS X-Versionen reichen an diese Benutzerfreundlichkeit wieder heran. Durch den UNIX-Kern von Mac OS X erreicht das System nicht nur eine hervorragende Stabilität, es ist auch in der Grundeinstellung bereits gut gegen Angriffe aus dem Internet geschützt – Sicherheitslücken sind selten und werden schnell behoben, Trojaner und Würmer können dementsprechend wenig anrichten, Viren sind nahezu unbekannt.

Der UNIX-Kern ist für Entwickler besonders interessant, schließlich können dadurch viele bekannte Open-Source-Werkzeuge unter Mac OS X genutzt werden – entweder werden diese bereits standardmäßig mitgeliefert, oder Sie können sie sich selbst kompilieren. Und falls Sie die grafische Oberfläche bevorzugen und Programme wie Adobe Acrobat oder Microsoft Office verwenden wollen, ist dies auch kein Problem.

---

<sup>1</sup> <http://www.apple.com/pro/science/gosling/>



Der wichtigste Punkt aber, warum Sie überhaupt dieses Buch aufgeschlagen haben: Java ist fest ins System integriert – Apple betrachtet Java als eine der Standard-Programmierungsumgebungen für Mac OS X!

### **Für wen ist dieses Buch geschrieben?**

Dieses Buch spricht sehr unterschiedliche Zielgruppen an: vom langjährigen Mac-Anwender, der nun Java programmieren möchte, bis hin zum langjährigen Java-Entwickler, der auf Mac OS X umsteigen oder zumindest seine Anwendungen daran anpassen will.

Damit trotz der verschiedenen Voraussetzungen alle zumindest das Zielsystem verstehen und mit den gängigen Begriffen umgehen können, bietet das erste Kapitel die Grundlagen sowohl für die Mac-Bedienung als auch für die Java-Entwicklung mit den Standard-Kommandozeilenwerkzeugen. Danach wird dann angenommen, dass Sie entweder Java schon beherrschen, gerne Beispiele ausprobieren (die lauffähig auf der Buch-CD beiliegen) oder nebenbei die Sprache mit einem weiteren Buch lernen (falls Ihnen das Anhang-Kapitel zu den Java-Grundlagen nicht ausreicht).

Dennoch finden sich in den Kapiteln immer wieder Hinweise, die für Profis vermutlich nichts Neues sind, die aber Einsteigern über viele kleine Fallen hinweghelfen. Die ausführliche Beschreibung der Beispiele richtet sich gerade in den ersten Kapiteln an die Einsteiger – Profis werden sich hier wohl eher an den abgedruckten Listings orientieren.

Was dieses Buch nicht ist: eine allgemeine Einführung in die Programmierung oder eine komplette Java-Referenz. Sie sollten bereits ein wenig Programmiererfahrung in irgendeiner Sprache besitzen, damit Sie etwas mit Begriffen wie »Variable« oder »Schleife« anfangen können. Und den kompletten Umfang von Java zu berücksichtigen, kann nicht Ziel dieses Buches sein – es konzentriert sich darauf, die Software-Entwicklung für die verschiedenen Java-Bereiche (J2SE, J2EE, J2ME) auf Mac OS X grundlegend vorzustellen und dabei vor allem die Mac-Besonderheiten ausführlich zu beschreiben.

### **Entscheidungshilfe für potentielle »Switcher«?**

Wenn Sie schon lange einen Mac besitzen oder auch kürzlich erst umgestiegen sind, stellt sich für Sie die Frage nach der Entwicklungsplattform nicht mehr – Sie möchten Java auf Mac OS X programmieren und portabel auf den diversen Java-Systemen einsetzen. Lesen Sie einfach beim nächsten Abschnitt weiter!

Wenn Sie hauptsächlich für andere Systeme entwickeln und Ihre Java-Anwendungen einfach nur möglichst gut an Mac OS X anpassen möchten: Seien Sie

beruhigt – Sie benötigen nicht zwingend einen Apple-Rechner, um die Informationen dieses Buches verwerten zu können. Die meisten Anpassungen für MacOS X lassen sich auch mit einer Entwicklungsumgebung unter Windows oder Linux realisieren. Allerdings sollten Sie bedenken, dass ein vernünftiger Softwaretest immer nur auf dem System stattfinden kann, wo die Software später auch eingesetzt wird. Daher sollten Sie sich zumindest das Betriebssystem selbst zulegen und dann mit einer geeigneten Emulationssoftware (beispielsweise »PearPC« von <http://pearpc.sourceforge.net/> bzw. <http://www.pearpc.net/>) auf einem schnellen PC einsetzen.

Sollten Sie schließlich zu der immer größer werdenden Gruppe der Java-Entwickler gehören, die sich gerade einen Umstieg auf MacOS X überlegen, kann Ihnen dieses Buch eine gute fachliche Entscheidungshilfe sein. Suchen Sie dazu alle Java-Themen zusammen, die Sie derzeit für Ihre Projekte benötigen, und klären Sie anhand der jeweiligen Kapitel, ob und wie diese auf MacOS X genutzt werden können – in der Regel wird es dabei kaum oder keine Probleme geben. Eine Frage, die in diesem Zusammenhang immer wieder gestellt wird: Ist die Java-Entwicklung mit den Mac-Entwicklungsumgebungen wirklich so viel langsamer als unter Windows? Dies kann ich Ihnen leider nicht pauschal beantworten. Wenn man beide Welten kennt, merkt man bei der Benutzungsoberfläche definitiv einen Unterschied. Aber was für die einen unerträglich und ein K.o.-Kriterium ist, stört die anderen wenig oder gar nicht. Ganz am Anfang haben Sie bereits James Gosling kennen gelernt, den Erfinder der Java-Technologie, der nach eigenem Bekunden für fast alle Entwicklungsaufgaben seinen portablen Apple-Rechner verwendet. Wenn Sie also die in diesem Buch gezeigten Java-Möglichkeiten auf MacOS X interessieren, gehen Sie doch einfach mal zu einem Mac-Händler in Ihrer Nähe und probieren Sie dort Eclipse oder NetBeans aus.

### **Wie sollten Sie dieses Buch lesen?**

Als **Einsteiger** sollten Sie zunächst die Java-Kurzeinführung im Anhang lesen, falls Sie Java nur wenig kennen oder Ihr Grundlagenwissen auffrischen möchten. Dann können Sie mit Kapitel 1 loslegen. Wenn Sie den Umgang mit MacOS X noch nicht lange gewohnt sind, erfahren Sie am Anfang des Kapitels, wie Sie den Mac richtig bedienen. Daran schließen sich die absoluten Grundlagen von Java auf MacOS X an. Das Ende des Kapitels über den Aufbau der Apple-Java-Implementierung ist zunächst vermutlich nicht so interessant für Sie. Lesen Sie dann auf jeden Fall noch Kapitel 2 über die diversen Entwicklungsumgebungen und Kapitel 4 über die unterschiedlichen Möglichkeiten, wie Sie ein Java-Programm als ausführbares Programm verteilen können. Die restlichen Kapitel lesen Sie dann bei entsprechendem Interesse.

Als **Profi** wird Sie zunächst der letzte Teil von Kapitel 1 über die Struktur des Apple-Java-Systems interessieren. Gerade wenn das Apple-Betriebssystem neu für Sie ist, kann es aber auch nicht schaden, wenn Sie sich mit den grundlegenden Bedienweisen des Macs am Anfang des Kapitels vertraut machen. Kapitel 2 über die Entwicklungsumgebungen können Sie kurz überfliegen, ebenso Kapitel 8 über die diversen Werkzeuge, die Ihnen zum Teil vielleicht schon von anderen Systemen her bekannt sind. Kapitel 4 über ausführbare Programme und die Formen der Programmdistribution sowie das kurze Kapitel 5 über portable Programmierung sollten Sie auf jeden Fall lesen; wenn Sie Anwendungen mit grafischen Benutzungsoberflächen entwickeln, gilt dies auch für Kapitel 3. Zur Feinoptimierung Ihrer Applikationen finden Sie im Anhang Listen der System-Properties und der VM-Optionen, ansonsten lesen Sie die restlichen Kapitel bei Bedarf.

Und wenn Sie sich **irgendwo zwischen Einsteiger und Profi** einschätzen, arbeiten Sie das Buch ganz einfach von vorne nach hinten durch!

Da dieses Buch aber nicht zwingend von vorne nach hinten durchgelesen werden muss, sondern Sie sich einzelne Kapitel nach Interesse herausuchen können, finden Sie die jeweils relevanten Literaturangaben und Internet-Links am Ende jedes Kapitels. Zusammen mit den Links aus dem Fließtext finden Sie die Webadressen auch auf meiner Webseite zum Buch (s. u.), wo sie regelmäßig aktualisiert werden. Die Links sind trotzdem im Text enthalten, damit Sie einen Anhaltspunkt haben, wonach Sie bei Google suchen könnten. Und damit Sie beim Lesen einzelner Kapitel nicht ständig blättern müssen, werden einige grundlegende Informationen vereinzelt mehrfach aufgeführt.

### **Welche Versionen werden behandelt?**

Der Fokus des Buchs liegt auf dem derzeit aktuellen Mac OS X 10.3 (»Panther«) mit Java 1.3 und Java 1.4. Bei allen Themen sind wichtige Unterschiede zu Mac OS X 10.2 (»Jaguar«) dokumentiert, damit Sie ältere Programme verstehen und gegebenenfalls auf den neuesten Stand bringen können. Hinweise für noch ältere Systeme tauchen nur vereinzelt auf, da die Versionen bis einschließlich Mac OS X 10.1 im praktischen Einsatz kaum noch von Bedeutung sind.

So weit wie derzeit möglich ist bei den Angaben sichergestellt, dass die Informationen auch für das kommende Mac OS X 10.4 (»Tiger«) gültig sind, das Java 1.5 (bzw. J2SE 5.0, wie die Version neuerdings auch heißt) enthalten wird. Im Anhang finden Sie eine kurze Einführung zu Java 1.5.

## Beispielprogramme

Die Quelltexte sind nicht immer vollständig abgedruckt, beispielsweise wenn es sich um Standardcode handelt, der nichts Besonderes zum jeweiligen Beispiel beiträgt. Die entsprechenden Stellen sind mit `// . . .` gekennzeichnet, aber auf der beiliegenden CD sind die Quelltexte natürlich komplett enthalten. Ebenso wurden im Buch die Quelltext-Kommentare entfernt, da die Beispiele im Fließtext beschrieben werden.

Im Buch wird bewusst kein durchgehendes Beispielprogramm verwendet, und aufbauende Beispiele werden nur dort eingesetzt, wo es einen wirklichen Mehrwert bringt. Am häufigsten finden Sie in den Abschnitten also unabhängige Beispiele, wodurch aber auch der Struktur des Buches Rechnung getragen wird und Sie die Kapitel weitestgehend unabhängig voneinander lesen können.

Die Beispiele sind bewusst einfach gehalten. Als Profi kennen Sie sowieso größere Projekte und möchten nur den Mac-relevanten Code kennen lernen, und als Einsteiger würden Sie bei größeren Beispielen zu leicht den Überblick verlieren. Daher verwenden die meisten Programme z.B. auch keine aufwändigen Oberflächen (abgesehen natürlich von denen im Kapitel über Benutzungsoberflächen), damit Sie schnell das Wesentliche erkennen und einfach aus dem Quelltext heraus kopieren können.

Wo Sie die Beispiele auf der beiliegenden Buch-CD finden und was darauf sonst noch enthalten ist, wird im Anhang kurz beschrieben. Alternativ lassen Sie sich einfach die Datei `index.html` aus dem CD-Wurzelverzeichnis in einem Web-Browser anzeigen.

## Schriftdarstellung

Dieses Buch verwendet folgende Konventionen für die Schriftdarstellung der unterschiedlichen Textbestandteile:

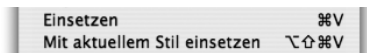
Textbestandteil	Darstellung
Programmquelltext (Listings)	<code>/* Schrift mit fester Zeichenbreite */</code>
Java-Bezeichner (Variablen, Methoden, Klassen)	Schrift mit fester Zeichenbreite: <code>meinObjekt</code> <code>indexOf()</code> <code>String</code>
Dateinamen und Pfadangaben	Schrift mit fester Zeichenbreite: <code>/Programme/TextEdit</code>

Verwendete Schriftkonventionen

Textbestandteil	Darstellung
Befehle und Programm- ausgaben	Schrift mit fester Zeichenbreite: java -version
Programmbeispiele auf der CD, hier »HalloWelt« in Kapitel 1	Schrift mit fester Zeichenbreite (es handelt sich um Pfadangaben auf der CD): //CD/examples/ch01/hallo/HalloWelt.java
Web-Adressen (URLs)	<i>Kursivschrift:</i> <i>http://www.muchsoft.com/java/</i>
Menüs und Menüeinträge, Texte auf dem Bildschirm	<b>Menütitel • Menüeintrag • Untermenüeintrag</b> usw.: <b>Bearbeiten • Kopieren</b> <b>Ablage • Sichern unter...</b> Einzelne Menüeinträge und Texte auf dem Bildschirm werden auch in Anführungszeichen gesetzt.
Tastaturkürzel (Shortcuts)	Die Umschalttasten werden ausgeschrieben. Einzelne Buch- staben werden großgeschrieben, $\square$ ist dabei aber nur zu drücken, wenn dies auch in der Tastenkombination steht. Sonstige Aktionen werden normal dargestellt: $\square + \square$ $\square + \square$ So angegebene Tastaturkürzel sind zusammen auszuführen (bzw. erst die Umschalttasten, dann der Rest).

Verwendete Schriftkonventionen (Forts.)

Der Mac besitzt diverse Umschalttasten mit verschiedenen Bezeichnungen: Shift, Ctrl (Steuerung), Alt (Wahl bzw. Option), Apfel (Command / Befehl bzw. Propeller). In Abbildung 0.1 sehen Sie beispielsweise den Menüeintrag »Einsetzen« mit dem Tastaturkürzel  $\square + \square$ , darunter den Eintrag »Mit aktuellem Stil einsetzen« mit dem Kürzel  $\square + \square + \square + \square$ . Die Umschalttaste  $\square$  wird beim Mac eher selten bei Menüeinträgen benutzt, dort würde dann das Dach (^) auftauchen.



Symbole der Umschalttasten

## Danksagung

Ein herzliches Dankeschön gilt dem gesamten Team von Galileo – ohne sie würden Sie dieses Buch schließlich nicht in den Händen halten! Ganz besonderem Dank bin ich dabei meinem Lektor Stephan Mattescheck verpflichtet – für die gute Betreuung und die unendliche Geduld, die ein Lektor wohl immer bei solchen Projekten aufbringen muss. Bedanken möchte ich mich auch beim Mac-Stammtisch Karlsruhe (<http://www.mac-ka.de/>), bei den Schwarzwäldern

und bei vielen Verwandten und Bekannten für ihr beständiges Interesse am Fortschritt (»Na, was macht das Buch? Immer noch nicht fertig?« ;-)) – durch ihre Fragen wurde mein Augenmerk immer wieder auf kleine, wichtige Details gelenkt, die es zu berücksichtigen galt. Und nicht zuletzt ein ganz liebes Dankeschön an Gesine, die das Buch von Anfang an begleitet und mich mit Kreativität und Motivation unterstützt hat!



### **Kontakt**

Wenn Sie Fragen, Anregungen oder Verbesserungsvorschläge haben, schreiben Sie mir an [macjava@muchsoft.com](mailto:macjava@muchsoft.com) oder schicken Sie Ihre Post an den Verlag Galileo Press. Schauen Sie aber am besten vorher auf meiner privaten Webseite zum Buch (<http://www.muchsoft.com/java/>) vorbei, ob Ihre Fragen und Wünsche dort vielleicht schon beantwortet sind. Und natürlich können Sie auch die Verlagsseite <http://www.galileocomputing.de/> besuchen, wo gegebenenfalls Online-Aktualisierungen für das Buch zur Verfügung stehen.

Zum Schluss bleibt mir nur noch, Ihnen viel Spaß beim Lesen und Ausprobieren zu wünschen – viel Erfolg mit der Java-Programmierung auf und für Mac OS X!

**Thomas Much**

Karlsruhe, im Oktober 2004

# 1 Grundlagen

1.1	Historisches .....	22
1.2	Aktuelles .....	24
1.3	Den Mac richtig bedienen .....	25
1.4	Eingeben, übersetzen, ausführen .....	35
1.5	Entwicklerwerkzeuge und -dokumentation .....	42
1.6	Up to date bleiben .....	44
1.7	Mac OS X erkennen .....	47
1.8	Konfigurationsdateien speichern .....	52
1.9	Aufbau des Apple-Java-Systems .....	55
1.10	Kleine Fallen .....	69
1.11	Literatur & Links .....	70

# **1 Grundlagen**

## **2 Entwicklungsumgebungen**

## **3 Grafische Benutzungsoberflächen (GUI)**

## **4 Ausführbare Programme**

## **5 Portable Programmierung**

## **6 Mac OS X-spezifische Programmierung**

## **7 Grafik und Multimedia**

## **8 Werkzeuge**

## **9 Datenbanken und JDBC**

## **10 Servlets und JavaServer Pages (JSP)**

## **11 J2EE und Enterprise JavaBeans (EJB)**

## **12 J2ME und MIDP**

## **A Kurzeinführung in die Programmiersprache Java**

## **B Java auf Mac OS 8/9/Classic**

## **C Java 1.5 »Tiger«**

## **D System-Properties**

## **E VM-Optionen**

## **F Xcode- und Project Builder-Einstellungen**

## **G Mac OS X- und Java-Versionen**

## **H Glossar**

## **I Die Buch-CD**



# 1 Grundlagen

*»Beginne am Anfang«, sagte der König ernst, »und fahre fort, bis du ans Ende kommst: Dann höre auf.«  
(Lewis Carroll)*

In diesem Buch treffen zwei Welten aufeinander, die zwar perfekt zueinander passen, die aber bisher oft noch nichts voneinander mitbekommen haben – die Welt der Apple-Macintosh-Programmierer, die bisher wenig mit Java zu tun hatten, und die Welt der Java-Programmierer, die bisher meistens unter Windows oder Linux entwickelt und Apple-Rechner und deren Betriebssystem MacOS X als »schönes Spielzeug« abgetan haben. Egal, welcher der beiden Welten Sie angehören – in diesem Grundlagen-Kapitel bekommen Sie eine schnelle und einfache Einführung sowohl in Mac OS X als auch in Java.

Als Umsteiger werden Sie sich sicherlich speziell für die Mac-Besonderheiten interessieren. Und da Sie für dieses für Sie neue System Software entwickeln wollen, müssen Sie diese Besonderheiten verstehen, damit Sie wie ein langjähriger Mac-Anwender denken lernen. Dazu gehört unter anderem die grundlegende Philosophie des Systems (Maus, Menü, Fenster), ebenso wie allgemein übliche Tastenkürzel und Bedienweisen. `java` und `javac` sind für Sie nichts Neues, aber wir fangen wie üblich trotzdem damit an, damit Sie einen schnellen Überblick über das Java-System von Mac OS X bekommen – danach werden Sie dann vermutlich sowieso ein bisschen selbst herumprobieren. Die bekannten sowie Mac-spezifische Entwicklungsumgebungen behandelt das nächste Kapitel.

Als Mac-Anwender kennen Sie die Bedienung des Systems bereits – aber wenn auch Sie ein »kleiner« Umsteiger von Mac OS 8 bzw. 9 oder Mac OS X 10.1 sind, werden Sie hier sicherlich noch einige nützliche Tipps entdecken. Im Wesentlichen möchten Sie aber Java kennen lernen. Deshalb eine kleine Warnung vorweg: Dies ist kein reines Java-Einsteigerbuch! Im Anhang gibt es zwar eine Kurzeinführung in die Sprache Java. Wenn Sie aber noch keine Programmiersprache beherrschen oder anhand vieler Beispiele lernen möchten, wie man in Java `for`-Schleifen programmiert, sollten Sie zusätzlich ein weiteres Buch zu Rate ziehen. Wenn Sie dagegen schon programmieren können, Java vielleicht vor einigen Jahren in der Schule oder an der Uni gesehen haben oder eine neue Sprache am besten mit kleinen Beispielprogrammen kennen lernen, werden Sie sich hier prima zurecht finden. Dementsprechend geht es in diesem Kapitel mit dem bekannten »Hallo Welt« los, mit dem Sie alle wichtigen Schritte bis zum fertigen Java-Programm einmal sehen werden.

Doch zunächst einmal starten wir mit einem kurzen Abriss der Entwicklungsgeschichte von Java im Allgemeinen und von Java auf dem Macintosh im Speziellen.

## 1.1 Historisches

Als **Sun** die Java-Technologie im Jahre 1995 ankündigte, war der Internet-Boom gerade dabei, so richtig loszulegen. Es gab mittlerweile grafische Browser für die wichtigsten Betriebssysteme, um Webseiten plattformübergreifend darzustellen, aber die Software musste immer noch für jedes System einzeln entwickelt werden. Man suchte also nach einer Lösung, um Programme für alle Systeme zusammen nur noch ein einziges Mal schreiben und übersetzen zu müssen, und Java kam dafür wie gerufen.

Anfang 1996 veröffentlichte Sun **Java 1.0**, und schon damals war alles Nötige für den »Write once, run anywhere«-Ansatz dabei: die Programmiersprache Java mit einer ziemlich exakten Spezifikation; ein passender Compiler, um aus Java-Quelltexten ausführbaren Java-Bytecode zu generieren; die Java Virtual Machine (auch JVM oder Java VM genannt) als »Interpreter« oder besser Laufzeitumgebung zur Ausführung des Java-Bytecodes; und schließlich die Klassenbibliotheken, die es überhaupt erst ermöglichen, plattformunabhängig Benutzungsoberflächen, Netzwerkzugriffe usw. zu realisieren. Mit dieser Version konnte man schon gut erahnen, welche Möglichkeiten Java bieten würde, aber wie bei jeder Version 1.0 gab es viele kleine Macken – und eine große: die recht gemächliche Ausführungsgeschwindigkeit.

Anfang 1997 war dann **Java 1.1** verfügbar. Es gab zur Geschwindigkeitsverbesserung jetzt einen Just-in-time-Compiler (JIT), der Java-Programme während der Ausführung nebenbei in Maschinencode übersetzen konnte. Außerdem wurde die Programmiersprache selbst ein bisschen aufgeräumt und erweitert.

Seit Ende 1998 gibt es **Java 1.2**, und da diese Version eine enorme Vergrößerung und Verbesserung der Klassenbibliotheken mitbrachte (z.B. Swing als umfangreiche Oberflächen-Bibliothek oder die Collections API für dynamische Datenstrukturen), spricht Sun seitdem von der »Java 2 Plattform«. Im Jahre 2000 kam dann **Java 1.3** heraus und brachte die »HotSpot« Virtual Machine mit, die deutlich besser just in time übersetzen konnte. Seitdem braucht sich Java geschwindigkeitsmäßig nicht mehr hinter C++ verstecken. **Java 1.4** wurde Ende 2002 fertig gestellt. Neben einer neuen, schnellen I/O-Bibliothek wurden zahlreiche bis dahin externe Pakete in die Standardbibliotheken integriert, darunter solche zur XML-Verarbeitung sowie reguläre Ausdrücke zur schnellen Textsuche.

Java bzw. das JDK (»Java Development Kit«) wurde von Sun zunächst nur für Solaris und Windows, seit 2001 dann auch für Linux zur Verfügung gestellt (ebenso das JRE, das »Java Runtime Environment«, das bei Endanwendern installiert werden kann und entsprechend die Entwicklungswerkzeuge nicht enthält). Als Java das Licht der Welt erblickte, verkaufte **Apple** gerade die letzten 68k-Computer (mit 68030- bzw. 68040-Prozessoren) und hatte den Umstieg auf die PowerPC-Architektur (PPC, die ersten Generationen der heutigen G4- und G5-Prozessoren) fast abgeschlossen. Aktuell war zu der Zeit Mac OS 7. Von Anfang an hat Apple die Entwicklung des Java-Systems für das Mac OS selbst in die Hand genommen (bzw. nehmen müssen) – trotz guter Kontakte zu Sun war der Apple-Marktanteil damals einfach zu klein, als dass Apple von Sun direkt unterstützt wurde.

Mit etwas über einjähriger Verzögerung erschien Anfang 1997 **MRJ 1.0** (»Macintosh Runtime for Java«), was Suns Java 1.0.2 entsprach. Es war Bestandteil von Mac OS 8 und war für 68k- und PPC-Rechner erhältlich. Bereits ein halbes Jahr später kam MRJ 1.5 auf den Markt, das einen JIT-Compiler und die Programmierschnittstelle MRJToolkit mitbrachte, aber Java-seitig blieb es noch beim alten JDK 1.0.2. Das änderte sich Ende 1997 (am 31.12., Apple hatte es noch für 1997 versprochen) mit **MRJ 2.0**, nun war endlich Java 1.1 (genauer das JDK 1.1.3) auf dem Mac verfügbar. Und dabei blieb es für Mac OS 8 und 9 (die so genannten »Classic«-Systeme) im Wesentlichen bis heute. Seit MRJ 2.2 ist das JDK auf dem Stand 1.1.8 (der letzten Sun-Version von Java 1.1), aber Java 1.2 oder neuer gibt es für das alte Mac OS nicht.

Dann kam Mac OS X (das X wird »Ten« gesprochen), Apples nächste Betriebssystemgeneration. Auf UNIX basierend, mit einer neuen, grafischen Benutzungsoberfläche, die trotzdem wieder die Schlichtheit und einfache Benutzbarkeit bieten sollte – Apple hatte sich viel vorgenommen. Im Herbst 2000 wurde eine »Public Beta« verteilt, im Frühjahr 2001 war dann **Mac OS X 10.0** (Code-name »Cheetah«) erhältlich – und Java 1.3 (das JDK bzw. JRE 1.3.0) wurde fertig installiert mitgeliefert! Intern wurde das komplett neu entwickelte Java-System MRJ 3.0 genannt, aber Apple verabschiedet sich nach und nach von dem Namen MRJ, um sich dem Sun-Versionsschema anzupassen. Leider merkte man diesem System seinen 1.0-Status an, entsprechend wird **Mac OS X 10.1** (»Puma« – Apple mag Raubkatzen), das im Herbst 2001 erschien, von vielen als erste benutzbare Mac OS X-Version angesehen. Enthalten war darin auch das JDK 1.3.1, das es zuvor schon als separates Update gegeben hatte.

Im Jahre 2002 wurde dann das nächste größere System-Update, **Mac OS X 10.2** (»Jaguar«) veröffentlicht. Ab Werk war immer noch das JDK 1.3.1 dabei, Java 1.4.1 konnte dann 2003 als Update installiert werden. Bei **Mac OS X 10.3**

(»**Panther**«), das seit Herbst 2003 verkauft wird, sind beide JDKs (1.3.1 und 1.4.1) bereits vorinstalliert.

Von Anfang an hat Apple auch immer eigene Java-Entwicklertools und -Dokumentationen kostenlos zur Verfügung gestellt. Früher war dies das MRJ SDK mit einer Sammlung kleiner Tools, bei Mac OS X ist dies eine komplette Entwicklungsumgebung.

MRJ 2.2.x kann heute immer noch auch unter Mac OS X verwendet werden, und zwar in der »Classic«-Umgebung, wo dann ein komplettes Mac OS 9 als separater Mac OS X-Prozess läuft. Da aber mittlerweile kaum noch jemand für Java 1.1 oder das alte MacOS entwickelt, wird dieses Thema nur kurz im Anhang behandelt.

## 1.2 Aktuelles

Für die Betriebssysteme Solaris, Linux und Windows steht auf <http://java.sun.com/> mittlerweile **Java 1.5** bzw. die **J2SE 5.0** zur Verfügung (»Java 2 Platform Standard Edition« Version 5.0 – Sun lässt bei den Versionsnummern nun die »1« weg). Die neue Version integriert unter anderem die »Generics« (grob vergleichbar mit C++-Templates) in die Sprache Java – im Anhang finden Sie einen kurzen Überblick über die neuen Möglichkeiten.

**Mac OS X 10.2** hat Updates bis auf Version 10.2.8 erfahren, bringt Java 1.3.1 mit und kann mit dem Java 1.4.1-Update erweitert werden (vermutlich wird dies auch die letzte Java-Version für diese Systemversion sein). Als Entwicklungsumgebung stellt Apple den »Project Builder« zur Verfügung.

**Mac OS X 10.3** bringt von Anfang an Java 1.3.1 und 1.4.1 mit. Apples neue Entwicklungsumgebung hierfür sind die »Xcode Tools«, eine Weiterentwicklung des Project Builders. Mit einigen Monaten Verspätung steht für dieses System nun auch das JDK 1.4.2 als Update zur Verfügung, ebenso wie die Erweiterungen »Java 3D« und »Java Advanced Imaging« (JAI).

Im ersten Halbjahr 2005 soll dann **Mac OS X 10.4 (»Tiger«)** erscheinen. Welche Java-Versionen mitgeliefert werden, ist derzeit noch nicht abzusehen. Es ist aber sehr wahrscheinlich, dass Apple das aktuelle Java 1.5 fest ins Betriebssystem integriert.

Apple liefert nicht nur das »normale« Mac OS X für Client-Systeme aus, sondern auch die Server-Variante **Mac OS X Server**. Da sich beide Varianten aus Java-Sicht nicht wesentlich unterscheiden, wird in diesem Buch nur bei Bedarf auf Mac OS X Server hingewiesen. Die wichtigsten Unterschiede betreffen die Einstellungen der Java Virtual Machine, die im Anhang-Kapitel über die VM-

Op-tionen besprochen werden. Weitere Informationen zu Mac OS X Server finden Sie auf der Seite <http://www.apple.com/server/macosx/>.

Was Sie am historischen Überblick und an den aktuellen Daten sehen können: Da das Java-System von Mac OS X eine Eigenentwicklung von Apple ist (auch wenn diese zunehmend mit den Sun-Source synchronisiert ist), gibt es immer gewisse Verzögerungen, bis eine neue Java-Version auch für den Mac angeboten wird. Der Zeitabstand zwischen dem öffentlichen Sun-Release und Apples Mac OS X-Anpassung ist meistens gar nicht so groß. Die Lücke wirkt allerdings deutlich länger, da Sun oft Monate vor dem öffentlichen Release Entwickler-Vorabversionen freigibt, und diese sind in der Regel nicht für den Mac erhältlich.

Wenn Sie damit leben können, nicht immer die allerneuesten Java-Vorabversionen zur Verfügung zu haben, besitzen Sie mit Mac OS X nicht nur ein hervorragendes System für den Einsatz Ihrer Java-Applikationen, sondern Sie können damit genauso gut Java-Software entwickeln, die dann natürlich auch wieder auf anderen Plattformen eingesetzt werden kann! Und bevor wir nun gleich mit der eigentlichen Java-Programmierung beginnen, lernen Sie erst noch den Mac und seine Oberfläche besser kennen.

### 1.3 Den Mac richtig bedienen

Wenn die grafische Benutzeroberfläche (GUI, »Graphic User Interface«) von Mac OS X neu für Sie ist, nehmen Sie sich bitte die Zeit, sich in die Mac-Philosophie ein bisschen einzudenken. Denn auch wenn Desktop-Rechner heutzutage meistens mit einer grafischen Oberfläche bedient werden, gibt es doch zum Teil erhebliche Unterschiede – nicht nur im Aussehen, sondern auch in der Funktionsweise. Apple hat eine über 20-jährige Erfahrung mit grafischen Benutzerschnittstellen (1983 kam der Lisa-Computer auf den Markt, 1984 der erste Macintosh), und in dieser Zeit wurden die Konzepte oft getestet und in vielen nützlichen Kleinigkeiten verbessert. Das Problem dabei ist: Wenn Sie bisher Oberflächen wie Windows oder Linux-KDE gewohnt sind, haben Sie sich wahrscheinlich schon an deren »Denkweise« gewöhnt und kommen manchmal nicht mehr darauf, wie man den Schreibtisch (Desktop) intuitiv bedienen könnte (das ist nicht negativ gemeint, das entspricht einfach nur regelmäßigen Erfahrungen von Umsteigern).

Da Sie Java-Anwendungen entwickeln wollen, die auch von langjährigen Mac-Benutzern als »normale« Mac-Applikation erkannt und bedient werden sollen, finden Sie in diesem Abschnitt die wichtigsten Ideen und Begriffe rund um die aktuelle Mac-Oberfläche, damit Sie diese auch in Java berücksichtigen können.

Vor Mac OS X sah die Mac-Oberfläche deutlich anders aus, daher ist es auch für Umsteiger von solchen Systemen ratsam, das Folgende wenigstens zu überfliegen. Und falls Sie sich bisher generell geweigert haben, die »Designer-Oberfläche« vom Mac anzufassen, hier noch eine kleine Beruhigung: Gerade in Mac OS X 10.3 hat Apple einige Funktionen zusätzlich eingebaut, die Ihnen als Anwender von z.B. Windows eine etwas gewohntere Umgebung präsentieren.

Zunächst einmal gibt es seit der Einführung von Mac OS X eine vernünftige Benutzerverwaltung. Wenn Sie Mac OS X selbst installiert haben, haben Sie sich ein Benutzerkonto mit Namen und Passwort eingerichtet, mit dem Sie sich nach dem Rechnerstart anmelden können. Es ist auch möglich, einen bestimmten Benutzer automatisch anmelden zu lassen, aber aus Sicherheitsgründen sollten Sie dies in den Systemeinstellungen deaktivieren. Die Systemeinstellungen finden Sie normalerweise unten am Bildschirmrand im so genannten »Dock«, in das Sie übrigens beliebige Anwendungen und (rechts vom Trennstrich) Dokumente ziehen können, die dort dann als Verknüpfung abgelegt werden. In den Systemeinstellungen können Sie dann im Bereich »Benutzer« unter »Anmelde-Optionen« festlegen, dass kein Benutzer automatisch angemeldet wird. In Zukunft werden Sie dies teilweise abgekürzt als **Systemeinstellungen · Benutzer · Anmelde-Optionen** lesen. Man kann das Dock unter **Systemeinstellungen · Dock** übrigens auch so konfigurieren, dass es beispielsweise rechts am Bildschirmrand erscheint.



Abbildung 11 Das Dock und die Systemeinstellungen

Nach der Anmeldung (dem Login) sehen Sie den »Finder« samt Desktop. Mit dem Finder verwalten Sie das Dateisystem und navigieren durch die Verzeichnishierarchie. Er ist grob mit dem Datei-Explorer anderer Systeme vergleichbar, wir sehen ihn uns gleich noch genauer an.

Oben am Bildschirmrand befindet sich die **Menüleiste**. Es gibt immer nur eine einzige sichtbare Menüleiste, und zwar die der gerade aktiven (obersten) Applikation. Wenn eine andere Anwendung aktiv wird, schaltet Mac OS X die Menüleiste entsprechend um. Innerhalb von Fenstern haben Menüleisten bei Mac-Applikationen nichts zu suchen, dort tauchen höchstens Toolbars auf.

Ganz wichtig, Sie sehen es bereits: Es kann eine Menüleiste geben, ohne dass die zugehörige Applikation ein Fenster geöffnet hat! Wenn Sie ein Fenster schließen, schließen Sie im Normalfall wirklich nur das Fenster und beenden nicht gleichzeitig die Anwendung. Dies ist erfahrungsgemäß fast die größte Hürde für Windows-Anwender. Denken Sie daran, Ihre Applikationen zu beenden! Apple geht allerdings nach und nach dazu über, Programme, die nur aus einem einzigen Dialogfenster bestehen (»grafische Tools«), beim Schließen dieses Dialogs automatisch zu beenden, beispielsweise die Systemeinstellungen.

Ganz links finden Sie in jeder Menüleiste das **Apfel-Menü**. Dies gehört nicht der Applikation, sondern dem System, dementsprechend müssen Sie sich darum bei der Programmierung auch nicht kümmern. Im Apfel-Menü haben Sie Zugang zu den Systemeinstellungen, können sich als Benutzer abmelden, den Rechner ausschalten usw.

Direkt daneben liegt immer das **Programm-Menü**, das den Namen der aktiven Applikation trägt (daran erkennen Sie am einfachsten, welche Anwendung gerade aktiv ist!). Alle Tastatureingaben – sofern sie nicht vom System abgefangen werden, z.B. spezielle Tastenkürzel – landen bei der aktiven Applikation. Im Programm-Menü können Sie das aktive Programm ausblenden (dann sind weder Fenster noch die Menüleiste sichtbar), beenden und den Einstellungen-Dialog aufrufen. Dieser Menüpunkt befindet sich immer hier und nicht in irgendeinem Extras-Menü! Gleiches gilt für die Programminformationen »Über *Programmname*«, die auch hier (und nicht im Hilfe-Menü) abzurufen sind.

Beim Finder sehen Sie im Programm-Menü, dass er eine besondere Stellung im System hat – er wird automatisch gestartet und kann nicht beendet werden. Dies macht Sinn, denn er kümmert sich ja um den Desktop, und dieser ist immer im Hintergrund zu sehen. Mit dem Desktop können Sie auch jederzeit zum Finder umschalten, indem Sie einfach auf den Desktop-Hintergrund klicken. Wenn die gerade aktive Applikation offene Fenster hat, werden diese inaktiviert, anschließend wird der Finder zur obersten Applikation.

Generell können Sie jede Applikation aktivieren, indem Sie auf deren Icon (Programmsymbol) im Dock klicken. Im Dock sieht man an einem kleinen Dreieck unter dem Programmsymbol, dass die Anwendung derzeit läuft und man durch Anklicken des Symbols sofort zum Programm umschaltet (nebenbei können Sie damit auch ausgeblendete Programme wieder einblenden). Fehlt das Dreieck, wird die Anwendung beim Anklicken erst noch gestartet. Wenn Sie lieber mit der Tastatur als mit der Maus arbeiten, können Sie auch mit  $\text{⌘} + \text{⌘}$  zwischen den laufenden Applikationen wechseln. Bei Mac OS X 10.3 wird der Wechsel groß auf dem Bildschirm angezeigt, bei 10.2 nur klein im Dock.

Zur Verbesserung des  $\text{⌘} + \text{⌘}$ -Programmwechsels wird – gerade unter Mac OS X 10.2 – gerne die Software »LiteSwitch« verwendet: <http://www.proteron.com/liteswitchx/>.

Sie haben gerade ein Tastaturkürzel (Shortcut) gesehen – auch ein Mac kann mit der Tastatur bedient werden, selbst wenn die Oberfläche zunächst einmal sehr Maus-lastig ist. An folgende Namen von Umschalttasten sollten Sie sich gewöhnen:  $\text{⌘}$  (auch Propeller- oder Befehlstaste genannt),  $\text{⌥}$  (Option- oder Wahl-taste),  $\text{Ctrl}$  (Steuerung) und  $\text{⇧}$  (Umschalttaste). Mobile Rechner haben zusätzlich noch eine  $\text{Fn}$ -Taste, die aber für Programm-Shortcuts nicht relevant ist. Die wichtigsten Kürzel sind wohl  $\text{⌘} + \text{W}$  zum Schließen des obersten (aktiven) Fensters und  $\text{⌘} + \text{Q}$  zum Beenden der aktiven Applikation.

Vielleicht sind Sie es von Ihrem System gewohnt, dass Sie sich mit  $\text{⌘}$  durch alle Elemente eines Dialogs bewegen können. Beim Mac OS bewegt man sich damit eigentlich nur durch die Texteingabefelder. Sie können aber die Option »Tastatursteuerung einschalten« in **Systemeinstellungen · Tastatur&Maus · Tastatur-Kurzbefehle** aktivieren, dann sind alle Elemente eines Dialog mit  $\text{⌘}$  bzw. den Pfeiltasten erreichbar. Ausgeführt wird das markierte Element mit der Leertaste.

Zurück zur Menüleiste und den typischen Menüs. Neben dem Programm-Menü sollte sich immer das **Ablage-Menü** mit den Dateioperationen befinden, daneben das **Bearbeiten-Menü**. Hier tauchen die Menüeinträge »Ausschneiden«, »Kopieren« und »Einsetzen« mit den Kürzeln  $\text{⌘} + \text{X}/\text{C}/\text{V}$  auf. Beachten Sie, dass Windows mangels  $\text{⌘}$ -Taste stattdessen einfach  $\text{Strg}$  (Control) verwendet. Bei Java-Programmen können Sie glücklicherweise mit nur geringem Aufwand die Entscheidung »Apfel- oder Strg-Taste?« dem System überlassen. Andere Standardeinträge in diesem Menü sind »Widerrufen« (Undo) und »Alles auswählen«.



Am rechten Ende der Menüleiste gibt es üblicherweise ein **Fenster-Menü** mit allen offenen Fenstern der aktiven Applikation sowie das **Hilfe-Menü**.

Sehen wir uns den **Desktop** noch etwas genauer an. Am rechten oberen Rand liegen normalerweise alle angemeldeten Laufwerke (das können auch die einzelnen Partitionen – beim Mac »Volumes« genannt – einer Festplatte sein). Sollten diese dort nicht auftauchen, können Sie die Anzeige über den Menüpunkt **Finder • Einstellungen** einschalten (siehe Abbildung 1.2).

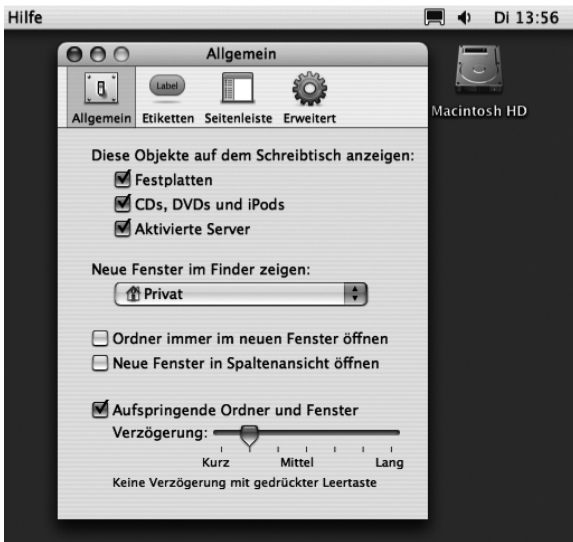


Abbildung 1.2 Finder-Einstellungen

Auf dem Desktop können Sie alles Mögliche ablegen (so wie Sie es von Ihrem »echten« Schreibtisch gewohnt sind ;-). Per Drag & Drop mit der Maus können Sie hierhin Dateien aus beliebigen Verzeichnissen ziehen, Text in einem Fenster selektieren und als Text-Clip auf den Desktop ziehen, URLs aus der Browser-Adressleiste als URL-Clip ablegen ... Und natürlich können Sie solche Clips dann auch wieder zurück in die Anwendung ziehen (oder einfach doppelt anklicken).

Um im Dateisystem zu arbeiten, öffnen Sie ein Finder-Fenster durch Doppelklick auf ein Laufwerkssymbol. Wenn der Finder die aktive Applikation ist, können Sie auch **⌘ + N** drücken, um ein neues Fenster zu öffnen. Es können beliebig viele Fenster gleichzeitig offen sein, zwischen denen Sie dann hin- und herkopieren können. Wichtig: Der Mac kennt keine MDI-(Multiple Document Interface)-Dokumente! Es gibt also keine Fenster, die die eigentlichen Dokumentfenster enthalten. Alle Dokumente liegen in eigenständigen, »normalen« Fenstern.

Wenn Sie bei so vielen offenen Fenstern den Überblick verlieren, hilft zum einen das Fenster-Menü in jeder Menüleiste. Zum anderen können Sie ab Mac OS X 10.3 einfach **F9** drücken, womit Sie die Exposé-Funktion aktivieren – alle Fenster werden verkleinert nebeneinander auf dem Bildschirm angezeigt, und Sie können das gewünschte Fenster aus dieser Übersicht auswählen.



Abbildung 1.3 Finder-Fenster

Einen Ordner öffnen Sie durch einen Doppelklick im selben Fenster. Wenn Sie zusätzlich zum Klick die **⌘**-Taste gedrückt halten, geht der Ordner in einem neuen Fenster auf. Ein Doppelklick auf ein Programm startet dieses. Alternativ können Sie das Programmsymbol auch nur selektieren (z.B. mit einem Einfachklick oder den Pfeiltasten) und dann mit **⌘**+**0** starten.

Für UNIX-Anwender: In Abbildung 1.3 sehen Sie das System-Wurzelverzeichnis aus Anwendersicht. Wenn Sie in der Shell `ls /` eingeben, tauchen auch die Ihnen geläufigen Verzeichnisse `/bin`, `/etc`, `/usr` usw. auf.

Apple liefert übrigens alle Mäuse immer noch mit nur einer Maustaste aus. Während dies Einsteigern entgegenkommt, wünschen Sie sich als Programmierer eventuell mehr Tasten und ein Scrollrad. Kein Problem: Schließen Sie einfach Ihre Lieblings-USB-Maus an, Mac OS X unterstützt die zusätzlichen Tasten, ohne dass Sie irgendwelche Treiber installieren müssen. Und bei einer Eintasten-Maus können Sie den Rechtsklick durch **Ctrl**+Klick emulieren.

Jetzt ist es leider an der Zeit, mit einem Mythos aufzuräumen ... Auch Mac-Programme können abstürzen! Und falls eine Applikation einmal »hängt«, können

Sie sie jederzeit im »Programme sofort beenden«-Dialog hart terminieren (»abschießen« oder »killen«), den Sie mit  $\boxed{\text{⌘}} + \boxed{\text{⌘}} + \boxed{\text{ESC}}$  erreichen. Seit Mac OS X 10.3 können Sie die oberste Applikation direkt ohne Nachfrage mit  $\boxed{\text{⌘}} + \boxed{\text{⌘}} + \boxed{\text{⌘}} + \boxed{\text{ESC}}$  aus dem Speicher entfernen. Aber Achtung: Bei beiden Methoden gehen geänderte Daten in den Anwendungen verloren!

Im Finder-Fenster »Macintosh HD« sehen Sie die Standardverzeichnisse des Systems, darunter `Programme` (hier befinden sich Applikationen für alle Anwender des Rechners) und `Benutzer` (hier hat jeder Anwender des Rechners sein eigenes, geschütztes Home-Verzeichnis). Falls Sie stattdessen die englischen Bezeichnungen `Applications` und `Users` sehen, sollten Sie in **Finder · Einstellungen · Erweitert** die Option »Alle Suffixe zeigen« ausschalten. Die Beispiele in diesem Buch verwenden die deutschen Namen, für das Funktionieren ist dies aber egal.

In den Verzeichnissen (und damit auch auf dem Desktop) können Sie auch Verknüpfungen (Aliase, Links) erzeugen. Ziehen Sie eine beliebige Datei einfach mit gedrückter  $\boxed{\text{⌘}}$ - und  $\boxed{\text{⌘}}$ -Taste in das Zielverzeichnis. Dort wird dann ein Symbol mit einem kleinen Pfeil an der unteren linken Ecke angezeigt. Achtung: Dies ist leider kein UNIX-Softlink! Softlinks erzeugen Sie nur mit `ln -s` in der Shell, die der Finder dann aber trotzdem als Alias verwenden kann.

Wenn Sie mehr über die Konzepte der Mac OS X-Benutzeroberfläche (auch »Aqua« genannt) wissen möchten, sehen Sie sich bitte die ausführlichen »Human Interface Guidelines« von Apple an, die Sie im Web auf <http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/> finden.

Ein paar Anmerkungen noch zu Festplatten am Mac. Eine Festplatte kann in mehrere Volumes (Partitionen) aufgeteilt sein. Auf jedem Volume können Sie bei Bedarf ein komplettes Betriebssystem installieren und so unterschiedliche Mac OS X-Versionen zum Testen auf einem Rechner verfügbar haben. Das Volume des aktiven Betriebssystems wird **Startvolume** genannt und kann unter **Systemsteuerung · Startvolume** für die künftigen Rechnerstarts festgelegt werden. Wenn Sie dagegen nur einmalig von einem anderen Volume booten wollen, drücken Sie beim Rechnerstart einfach die Taste  $\boxed{\text{C}}$ , um von einer eingelegten CD-ROM zu starten. Wenn Sie stattdessen  $\boxed{\text{⌘}}$  gedrückt halten, landen Sie im Boot-Manager, der Ihnen alle verfügbaren Systeme zur Auswahl anbietet.

Nach der Standardinstallation besitzt Ihr Rechner genau ein Volume mit dem Namen »Macintosh HD«. Obwohl Sie diesen Namen jederzeit ändern können, belassen wir es in diesem Buch dabei – mit »Macintosh HD« ist hier immer das Startvolume gemeint.

Wenn Sie Ihre Festplatte in mehr als ein Volume aufteilen wollen, geht das nur, wenn von dieser Platte gerade kein System gestartet wurde. Entweder partitionieren Sie also eine externe FireWire- oder USB-Festplatte, oder Sie teilen die Platte gleich bei der Installation von Mac OS X passend auf. Das System bringt dafür das Festplatten-Dienstprogramm mit, das Sie am Anfang der Mac OS X-Installation über das Installer-Menü starten können. Später können Sie das Festplatten-Dienstprogramm jederzeit im Dienstprogramme-Ordner innerhalb des Programme-Ordners aufrufen.

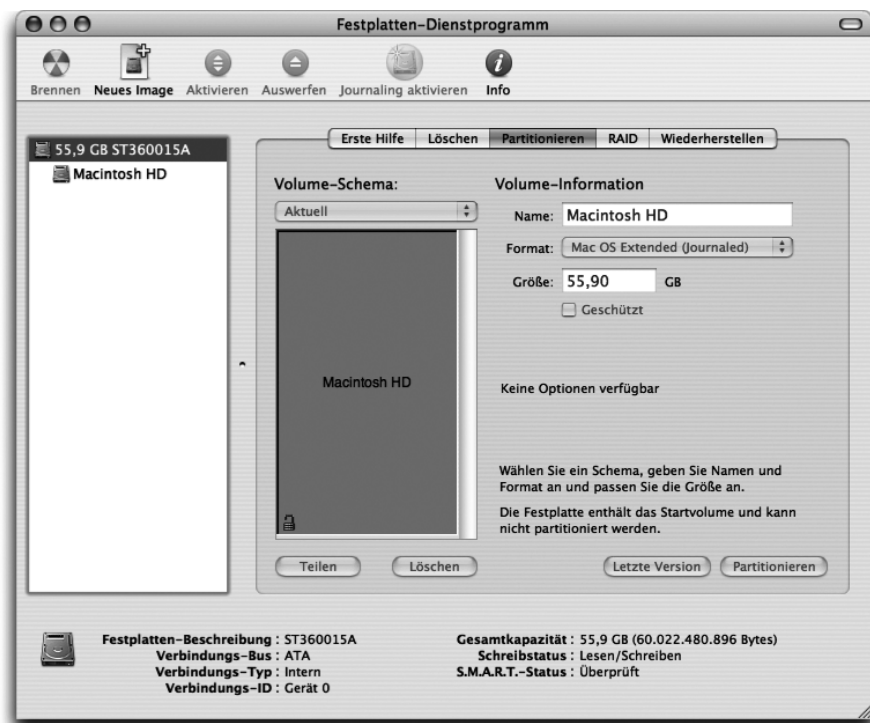


Abbildung 1.4 Festplatten-Dienstprogramm

Es ist übrigens eine gute Idee, vor jedem Wechsel zu einer neuen Mac OS X-Version die Datei-Zugriffsrechte des Volumes reparieren zu lassen. Einige Programme bringen diese leider ab und zu durcheinander, und dann geht ein Update manchmal schief ... Im Festplatten-Dienstprogramm wählen Sie dazu das Startvolume aus und klicken dann auf das Tab »Erste Hilfe«. Dort können Sie nun »Volume-Zugriffsrechte reparieren« starten.

Wenn Sie sich dafür interessieren, was »unter der Haube« passiert, bringt Mac OS X 10.3 auch dazu ein passendes Werkzeug mit. Mit dem Programm Aktivitäts-Anzeige aus dem Verzeichnis /Programme/Dienstprogramme

können Sie sich alle laufenden Prozesse übersichtlich anzeigen lassen, ebenso die Speicherauslastung, Festplattenaktivität usw. Alternativ können Sie sich diese Informationen im Terminal mit `top` und `ps -x` besorgen.

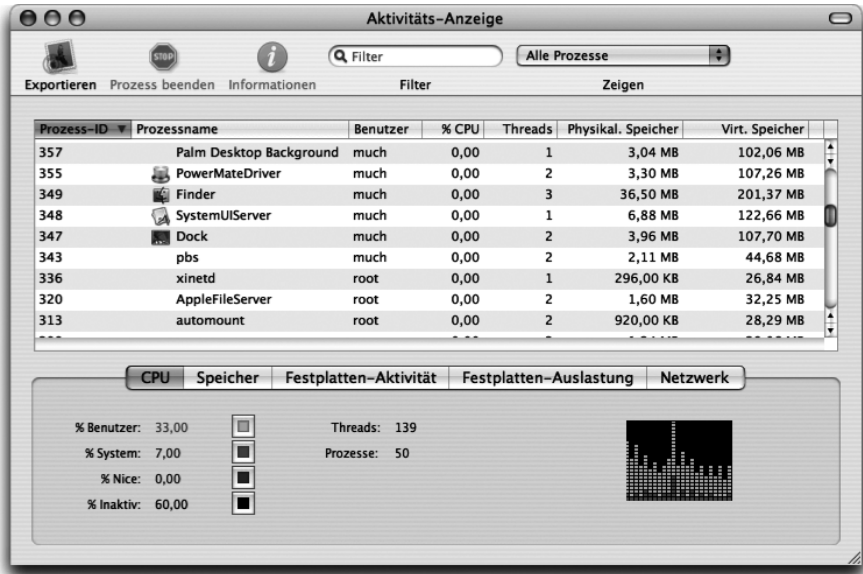


Abbildung 1.5 Aktivitäts-Anzeige

### Hilfe finden

Wenn Sie weitere Hilfe zu Mac OS X benötigen, können Sie über das Finder-Hilfe-Menü die »Mac Hilfe« aufrufen (alternativ drücken Sie einfach `⌘+?`). Während diese – gerade ab Mac OS X 10.3 – für die Anwendung des Systems oft ausreichende Informationen liefert, enthält sie kaum Informationen zur Software-Entwicklung. Wo Sie diese erhalten, erfahren Sie später in diesem Kapitel (Abschnitte 1.5 und 1.6).

Tastenkürzel	Auswirkung
<code>⌘+A</code>	Alles auswählen
<code>⌘+C</code>	Markiertes Element in die Zwischenablage kopieren
<code>⌘+F</code> / <code>⌘+G</code>	Suchen/Weitersuchen

Tabelle 1.1 Die wichtigsten Mac OS X-Tastenkürzel

1 Eine Übersicht über viele weitere Tastenkürzel finden Sie auf der Seite <http://www.xshorts.de/>.

Tastenkürzel	Auswirkung
⌘ + H	Anwendung ausblenden (»Hide«)
⌘ + I	Informationen zum selektierten Element anzeigen
⌘ + M	Fenster ins Dock minimieren
⌘ + N	Neues Dokument öffnen
⌘ + O	Vorhandenes Dokument öffnen (laden)
⌘ + P	Drucken (»Print«)
⌘ + Q	Programm beenden
⌘ + S	Dokument speichern
⇧ + ⌘ + S	Sichern unter einem neuen Namen
⌘ + V	Inhalt der Zwischenablage an der Cursorposition einfügen
⌘ + W	Fenster schließen
⌘ + X	Markiertes Element in die Zwischenablage kopieren und danach das markierte Element löschen
⌘ + Z	Letzte Aktion widerrufen (»Undo«)
⌘ + ⇧ + ⌘	Wechsel zwischen laufenden Programmen
⌘ + ?	Hilfe
⌘ + ,	Einstellungen
⌘ + . bzw. Esc	Laufende Aktion oder angezeigten Dialog abbrechen
⌘ + ⌘ + Esc	Dialog »Programme sofort beenden« öffnen (vergleichbar mit dem Task Manager unter Windows)
⇧ + ⌘ + ⌘ + Esc	Aktives Programm ohne Nachfrage beenden. Daten werden nicht gesichert! Erst ab Mac OS X 10.3
⌘ + ←	Zum Zeilenanfang/-ende springen
⌘ + →	
⌘ + ←	Zum vorhergehenden/nächsten Wort springen
⌘ + →	
F9	Alle offenen Fenster im Überblick darstellen. Erst ab Mac OS X 10.3
F10	Alle offenen Fenster der aktiven Applikation im Überblick darstellen. Erst ab Mac OS X 10.3
F11	Alle Fenster ausblenden, Finder-Desktop anzeigen. Erst ab Mac OS X 10.3

**Tabelle 1.1** Die wichtigsten Mac OS X-Tastenkürzel (Forts.)

Tastenkürzel	Auswirkung
<code>Ctrl</code> + <code>⏻</code>	Dialog »Computer jetzt ausschalten?« öffnen. Hier können Sie mit <code>⏻</code> den Rechner ausschalten, mit <code>R</code> (»Reboot«) neu starten und mit <code>S</code> (»Sleep«) in den Ruhezustand versetzen. <code>Esc</code> bricht den Dialog ab.
<code>Fn</code> + <code>⏻</code>	Entfernen (auf mobilen Rechnern)

Tabelle 1.1 Die wichtigsten Mac OS X-Tastenkürzel (Forts.)

## 1.4 Eingeben, übersetzen, ausführen

Nun ist es an der Zeit, Ihr erstes Java-Programm auf dem Mac zu starten! Dazu nehmen wir das allseits beliebte »Hallo Welt«-Programm, das eben diesen Text ausgibt. Während das Programm technisch nicht sonderlich interessant ist, können Sie mit den Schritten bis zur Ausführung das Java-System von Mac OS X schnell und einfach kennen lernen – nicht nur als Einsteiger, sondern auch als Java-Profi. Sollten Sie mit dem Quelltext selbst noch Probleme haben, lesen Sie sich bitte im Anhang die Kurzeinführung in die Sprache Java durch. Es handelt sich dort aber wirklich nur um eine extrem knappe Einführung, die Ihnen hilft, wenn Sie eine andere (vorzugsweise objektorientierte) Sprache beherrschen. Zum gründlichen Erlernen von Java sollten Sie zusätzlich ein anderes Buch lesen, beispielsweise den »Einstieg in Java« von B. Steppan.

Eine kleine Bemerkung noch vorneweg: Für alle Beispiele und Beschreibungen ist mindestens Mac OS X 10.2 Voraussetzung. Screenshots sind in der Regel mit Mac OS X 10.3 erstellt. Obwohl die meisten Programme – sofern sie mit Java 1.3 auskommen – auch unter Mac OS X 10.1 laufen dürften, ist dieses System zu alt und aus heutiger Sicht zu unausgereift, um hier speziell berücksichtigt zu werden.

Starten Sie den Standard-Texteditor von Mac OS X, TextEdit, der normalerweise das RTF-Format verwendet, aber ebenso Word-Dokumente laden und speichern und ganz normale Texte verarbeiten kann. Sie finden den Editor im Programme-Verzeichnis:

```
Macintosh HD
  Programme
    TextEdit
```

Wenn es der Übersichtlichkeit dient, werden Sie im Folgenden Pfadangaben auch in dieser Form finden, ansonsten wird die Kurzform `Macintosh HD/Programme/TextEdit` verwendet (oder nur `/Programme/TextEdit` – wenn ein UNIX-Pfad mit dem Schrägstrich beginnt, ist damit der Pfad ausgehend vom Startvolumen gemeint).

Geben Sie den Programmtext aus Abbildung 1.6 in das leere TextEdit-Fenster ein oder laden Sie den Quelltext `HalloWelt.java` mit dem Menüpunkt **Ablage · Öffnen...** (bzw. `⌘+O`) aus dem Verzeichnis `/examples/ch01/hallo/` von der Buch-CD. Wenn Sie die Datei laden, erkennt TextEdit automatisch, dass es sich um unformatierten Text handelt, ansonsten müssen Sie den Modus mit dem Menüpunkt **Format · In reinen Text umwandeln** umschalten.



Abbildung 1.6 »Hallo Welt« in TextEdit

Wenn Sie die roten Unterstreichungen stören, die in einem Programm Quelltext nichts zu suchen haben, deaktivieren Sie die Rechtschreibprüfung unter **Bearbeiten · Rechtschreibung · Während der Texteingabe prüfen**. Die Rechtschreibkontrolle ist ein Systemdienst, den alle Anwendungen – auch Java-Programme! – nutzen können.

TextEdit ist ein einfacher, aber brauchbarer Editor mit grafischer Benutzungsoberfläche. Kleinere Programme kann man sicherlich damit tippen, aber ein ausgewachsener Quelltext-Editor ist es definitiv nicht. Als universeller Editor wird auf dem Mac gerne BBEdit oder der preiswertere TextWrangler von Bare Bones Software eingesetzt. Kostenlos ist BBEdit Lite erhältlich, und obwohl diese Version seit längerem nicht mehr weiterentwickelt wird, lässt sie sich nach wie vor gut und problemlos verwenden. Sie können BBEdit Lite von

<http://www.barebones.com/products/bblite/index.shtml>

oder direkt von

[ftp://ftp.barebones.com/pub/freeware/BBEdit\\_Lite\\_612.smi.hqx](ftp://ftp.barebones.com/pub/freeware/BBEdit_Lite_612.smi.hqx)

herunterladen.<sup>2</sup> Im nächsten Kapitel sehen Sie dann weitere und vor allem spezialisiertere Editoren und Entwicklungsumgebungen.

2 smi- und hqx-Dateien sind ältere Mac OS-Formate. Klicken Sie nach dem Herunterladen einfach doppelt auf das Archiv, dann wird es vom Stuffit Expander, der Mac OS X beiliegt, ausgepackt.



Wenn Sie aus der UNIX-Welt kommen und Ihnen grafische Oberflächen nur bedingt zusagen, können Sie das Terminal – das Sie in Kürze sehen werden – öffnen und in der Shell `vi`, `emacs` oder `pico` aufrufen.

Während Sie tippen, ist Ihnen vielleicht aufgefallen, dass im roten Fenster-Schließ-Knopf oben links ein Punkt angezeigt wird. Dies signalisiert, dass das Dokument verändert wurde und dass beim Schließen des Fensters nun nachgefragt wird, ob man den Inhalt speichern oder verwerfen möchte.

Speichern Sie den Quelltext nun als `HalloWelt.java` in Ihrem Benutzer-Verzeichnis (Home) ab, am einfachsten mit **Ablage · Sichern** (oder **Ablage · Sichern unter...**, was bei neuen Dokumenten aber auf dasselbe hinaus läuft). Im erscheinenden Sichern-Dialog können Sie mehr Informationen einblenden, indem Sie auf den Knopf mit dem Dreieck (oben rechts) klicken. Das Home-Verzeichnis wird mit Ihrem Kurznamen bezeichnet, hier im Beispiel also »much«, das Sie oben im Popup oder links in der Favoritenleiste auswählen. In Abbildung 1.7 werden die Dateien und Ordner in der Listendarstellung angezeigt, die Sie vielleicht von anderen Betriebssystemen in ähnlicher Form kennen. Sie können aber mit den beiden Buttons oben links auch zur Spalten-(Browser-)Darstellung umschalten, die von erfahrenen Mac-Anwendern bevorzugt wird, da sie eine erheblich schnellere Navigation im Dateisystem ermöglicht. Unter Mac OS X 10.2 sieht der Sichern-Dialog ähnlich aus, bietet aber weniger Komfort.



Abbildung 1.7 »Sichern unter...«-Dialog

Beachten Sie, dass der Sichern-Dialog am jeweiligen Textfenster »klebt«. MacOS X nennt solche Fenster-modalen Dialoge »Sheets«, und diese können auch aus Java heraus angesprochen werden.

Das Übersetzen und Ausführen des Programms erledigen Sie nun in der Shell innerhalb der **Terminal**-Applikation (auch Kommandozeile oder Eingabeaufforderung genannt, je nachdem, mit welchem System Sie bisher gearbeitet haben). Starten Sie dazu die Anwendung `/Programme/Dienstprogramme/Terminal`. Es öffnet sich ein Terminal-Fenster mit einer Zeile




```
[straylight:~] much%
```

oder

```
Straylight:~ much$
```

je nachdem, welche Shell bei Ihnen als Standardshell konfiguriert ist (für das folgende Beispiel ist dies egal). Sie sehen den Rechnernamen<sup>3</sup>, die Tilde `~` sowie den Kurznamen des aktuell angemeldeten Benutzers. Die Tilde bezeichnet unter UNIX das Home-Verzeichnis und kann auch in Pfadangaben als Abkürzung verwendet werden.

Wenn Sie einmal Sonderzeichen eingeben möchten, können Sie sich in MacOS X 10.3 eine Tastaturübersicht einblenden lassen. Dazu müssen Sie in den Systemeinstellungen unter **Landeseinstellungen • Tastaturmenü** das Tastaturmenü in der Menüleiste aktivieren.

Dort können Sie dann auch Tastenkombinationen für bestimmte Sonderzeichen herausfinden. Die Tilde beispielsweise erhalten Sie, wenn Sie + und danach  drücken.

Geben Sie nun den Befehl `java -version` ein um zu testen, ob die JVM gestartet werden kann (und welche Version aktiv ist). Sie sollten dann ungefähr folgenden Text sehen (die Zahlen können variieren):

```
java version "1.4.2_05"  
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_  
05-141)  
Java HotSpot(TM) Client VM (build 1.4.2-38, mixed mode)
```

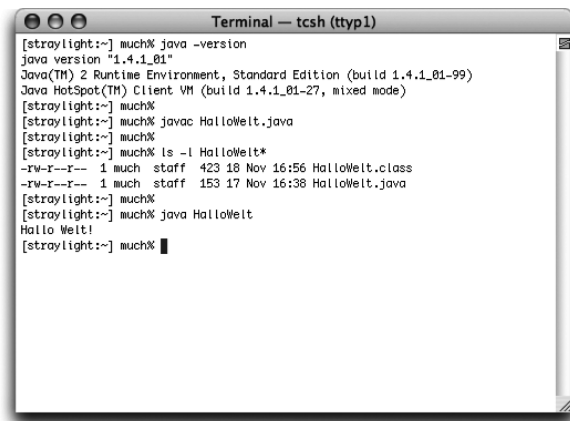
Mit `javac HalloWelt.java` starten Sie den Java-Compiler und übersetzen damit den Quelltext in so genannten Java Bytecode, der dann von der Virtual

---

<sup>3</sup> Der Rechnername lautet in diesem Fall »Straylight«. Sie können den Namen Ihres Rechners in den Systemeinstellungen unter **Sharing • Geräte** konfigurieren.

Machine ausgeführt werden kann. Das Ergebnis sehen Sie mit dem Kommando `ls -l` (etwa »list long«, zeige den Verzeichnisinhalt in ausführlicher Form), dort taucht die Klassendatei `HalloWelt.class` auf. Wenn Sie stattdessen von `javac` Fehlermeldungen bekommen, haben Sie vermutlich das Programm falsch abgetippt oder unter einem falschen Namen gespeichert (beachten Sie die Groß-/Kleinschreibung!).

Wenn das Übersetzen geklappt hat, können Sie die class-Datei durch Aufruf der JVM mit dem Befehl `java HalloWelt` ausführen lassen. Im Terminal sollte dann die Zeichenkette »Hallo Welt!« ausgegeben werden. Falls das nicht klappt, haben Sie vermutlich entweder die `main`-Methode falsch abgetippt oder beim Aufruf von `java` einen falschen Klassennamen angegeben.



```

Terminal — tcsh (tty1)
[straylight:~] much% java -version
java version "1.4.1_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_01-99)
Java HotSpot(TM) Client VM (build 1.4.1_01-27, mixed mode)
[straylight:~] much%
[straylight:~] much% javac HalloWelt.java
[straylight:~] much%
[straylight:~] much% ls -l HalloWelt*
-rw-r--r--  1 much  staff  423 18 Nov 16:56 HalloWelt.class
-rw-r--r--  1 much  staff  153 17 Nov 16:38 HalloWelt.java
[straylight:~] much%
[straylight:~] much% java HalloWelt
Hallo Welt!
[straylight:~] much% █
  
```

Abbildung 1.8 Die tcsh-Shell im Terminal

Wenn Sie die Shell beenden möchten, geben Sie einfach `exit` ein. Alternativ – und das ist meistens bequemer – schließen Sie einfach das Fenster oder beenden Sie die Terminal-Applikation. Sofern in der Shell nicht noch ein Programm läuft oder mit einer Fehlermeldung wartet, wird die Shell sofort korrekt beendet. Ansonsten werden Sie erst noch gefragt, ob Sie das laufende Programm wirklich terminieren wollen.

### **bash oder tcsh?**

In der UNIX-Welt gibt es einen Glaubenskrieg, wie man ihn sonst nur zwischen Windows- und Macintosh-Anwendern bezüglich des »besseren« Systems gewohnt ist. Bei UNIX geht es aber darum, welches die beste Shell ist ... Und auch MacOS X bringt einige davon mit, wie Sie mit `cat /etc/shells` überprüfen können.

Die bekanntesten sind sicherlich `bash` und `tcsh`, die Sie in einer beliebigen Shell jederzeit mit `/bin/bash` bzw. `/bin/tcsh` aufrufen (und mit `exit` wieder verlassen können).

Bis Mac OS X 10.2 war `tcsh` die Standardshell, und wenn Sie eine Upgrade-Installation auf 10.3 durchgeführt haben, ist das auch nach wie vor so konfiguriert. Wenn Sie aber Mac OS X 10.3 komplett neu installiert haben, ist nun `bash` die Standardshell. Da die Beispiele in diesem Buch mit der `tcsh` gezeigt werden (sofern wir später überhaupt noch das Terminal verwenden), möchten Sie vielleicht auch unter 10.3 `tcsh` als Standardshell einstellen. Dazu haben Sie zwei Möglichkeiten. Entweder rufen Sie die Terminal-Einstellungen auf und geben bei »Befehl ausführen« das Kommando `/bin/tcsh` ein (siehe Abbildung 1.9). Oder Sie geben in der `bash` folgende zwei Befehle ein:

```
export EDITOR=pico
chsh
```

Damit können Sie die Standardshell in einer Text-Konfigurationsdatei ändern, was dann für Ihr gesamtes Benutzerkonto (und nicht nur für die Terminal-Applikation) gilt. Profis, die als Editor lieber den `vi` verwenden, können den ersten Befehl natürlich weglassen (und wer aus Versehen im `vi`-Editor landet, kommt mit `:q` wieder raus).

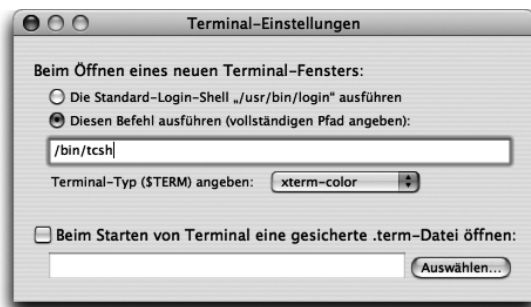


Abbildung 1.9 Terminal-Einstellungen für die `tcsh`-Shell

Als UNIX-Anwender werden Sie Ihre Shell schon in- und auswendig beherrschen, aber falls Sie bisher nur mit grafischen Oberflächen zu tun hatten, kommt Ihnen das alles vielleicht etwas kompliziert vor (seien Sie beruhigt, UNIX-Shell-Benutzern geht es mit grafischen Oberflächen meistens genauso ...). Die folgenden Kapitel betrachten vor allem grafische Entwicklungstools, aber es kann nicht schaden, sich etwas genauer mit den diversen Shell-Befehlen (`cd`, `ll` usw.) zu beschäftigen. Falls Ihnen die kurzen Beispiele in diesem Kapitel nicht

ausreichen, lesen Sie bei Bedarf bitte ein spezialisiertes Buch, z.B. »UNIX für Mac OS X-Anwender« von K. Surendorf.

Wenn der Befehl `ll` nicht gefunden wird, können Sie ihn in der `tcsh` aktivieren, indem Sie die folgenden drei Befehle ausführen:

```
echo "source /usr/share/tcsh/examples/rc" >> ~/.tcshrc
echo "source /usr/share/tcsh/examples/login" >> ~/.login
echo "source /usr/share/tcsh/examples/logout" >> ~/.logout
Eine ausführlichere Anleitung dazu erhalten Sie mit cat
/usr/share/tcsh/examples/README | more (weiterblättern mit )
```

Die Ausgabe des `HalloWelt`-Programms wurde direkt im Terminal angezeigt, denn dort hatten Sie das Programm ja auch gestartet. Java-Applikationen, die außerhalb einer Shell z.B. mit einem Doppelklick gestartet werden, schicken ihre Ausgaben dagegen an die **Konsole**, die Sie unter `/Programme/Dienstprogramme/Konsole` finden. Wenn Sie Mac OS X 10.3 verwenden, öffnen Sie doch einfach mal ein Finder-Fenster für ihr Home-Verzeichnis. Wenn Sie dort doppelt auf `HalloWelt.class` klicken, wird das Java-Programm gestartet und die Zeichenkette »HalloWelt!« landet in der Konsole. Einzelne Klassen auf diese Art zu starten, funktioniert aber nur, wenn keine benutzerdefinierten Bibliotheken außerhalb des Klassenpfads verwendet werden.

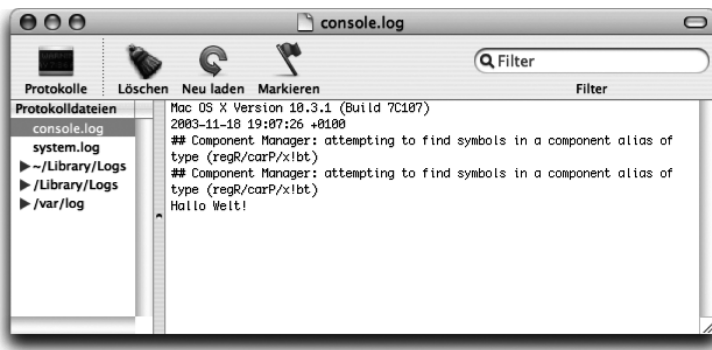


Abbildung 1.10 Java-Textausgabe in der Konsole

Die Konsole zeigt auch die so genannten Crash-Logs an, die auf der Festplatte abgelegt werden, falls eine Anwendung abstürzt. Sie finden die Crash-Logs in `~/Library/Logs/` im Verzeichnis `CrashReporter`. Dies betrifft aber eigentlich nur native Mac-Programme. Die Stack-Traces abstürzender Java-Applikationen werden im Terminal oder in der Konsole in `console.log` ausgegeben.

Wenn Sie ein bestimmtes Log-File immer im Blick haben wollen, ohne das Konsole-Fenster öffnen zu müssen, laden Sie sich doch das `GeekTool` von <http://projects.tynsoe.org/en/geektool/> herunter. Damit können Sie beliebige Textdateien im Desktop-Hintergrund anzeigen lassen.

### Hilfe in der Shell

Sie haben schon kurz das Mac-Hilfesystem kennen gelernt, aber auch in der Shell können Sie ein Hilfesystem nutzen, und zwar die UNIX-»man pages« (manual pages, Handbuchseiten). Diese werden mit dem Befehl `man` aufgerufen, beispielsweise `man javac`. Mit den Pfeiltasten bewegen Sie sich zeilenweise im Dokument, mit `→` geht es seitenweise vorwärts und mit `⌘` können Sie die Hilfe jederzeit verlassen.

## 1.5 Entwicklerwerkzeuge und -dokumentation

Nachdem Sie gerade gesehen haben, dass Java fester Bestandteil der Mac OS X-Standardinstallation ist, sollten Sie nun noch die Apple-Entwicklertools installieren, um Apples Entwicklungsumgebung (Project Builder bis Mac OS X 10.2, Xcode ab 10.3) und alle wichtigen Java-Dokumentationen nutzen zu können. Apples »Developer Tools« tragen seit Mac OS X 10.3 den offiziellen Namen »Xcode Tools« – einen Überblick darüber erhalten Sie auf der Seite <http://developer.apple.com/tools/macosxtools.html>.

Falls Sie bei der Systeminstallation also noch nicht die Software von der Entwickler-CD installiert haben, holen Sie dies nun bitte nach (siehe Abbildung 1.11). Wenn Sie gerade einen neuen Rechner mit vorinstalliertem Mac OS X gekauft haben, bekommen Sie die Entwickler-Tools nicht auf CD mitgeliefert, sondern finden sie auf Ihrer Festplatte im Verzeichnis `/Programme/Installer/Developer Tools/`. Sie können die Tools auch von Apples ADC-Webseite, die im folgenden Abschnitt vorgestellt wird, herunterladen.

Die Entwicklerwerkzeuge werden auf der ADC-Seite in zwei Varianten angeboten: einmal als eine einzige Datei, die dann mehrere hundert MByte groß ist, und einmal in Form von kleineren Segmenten. Wenn Sie letztere Variante wählen, laden Sie alle Segmente herunter und packen Sie sie im selben Verzeichnis aus – dort sollten sich dann eine `dmg`- und viele `dmgp`-Dateien befinden. Zum Aktivieren (»mounten«) des kompletten Volumens führen Sie dann einfach einen Doppelklick nur auf die `dmg`-Datei aus.



Abbildung 1.11 Xcode-Tools installieren

Nach jeder Installation einer neuen Xcode-Version müssen Sie zusätzlich die jeweils aktuellen »Java Developer Tools« installieren, die Sie von der ADC-Webseite herunterladen können. Apple stellt für jede Java-Version ein entsprechendes Entwicklerpaket bereit – beispielsweise gehört zum »Java 1.4.2 Update 1« das Entwicklerpaket »Java 1.4.2 Update 1 Developer Tools.«

Anschließend können Sie die bekannte JDK-API-Dokumentation von Sun installieren. Apple liefert sie als Archiv mit der Entwicklersoftware mit, Sie müssen zum Auspacken nur noch ein Shell-Skript ausführen. Geben Sie dazu bis Xcode 1.2 im Terminal folgenden Befehl ein:

```
sudo /Developer/Documentation/Java/scripts/
  unjarJavaDocumentation.sh
```

Die Entwickler-Software finden Sie nun im Verzeichnis `/Developer/`, die Java-Dokumentation im Verzeichnis `/Developer/Documentation/Java/` (die API-Dokumentation von Sun ist dort im Ordner `Reference` abgelegt). Ab Xcode 1.5 führen Sie stattdessen den Befehl

```
sudo /Developer/ADC\ Reference\ Library/documentation/Java/
  scripts/unjarJavaDocumentation.sh
```

aus, wodurch die Dokumentation im mittlerweile von Apple favorisierten Verzeichnis `/Developer/ADC Reference Library/documentation/Java` angelegt wird. Wenn Sie die »Java 1.4.2 Update 1 Developer Tools« installiert haben, ist keiner der beiden obigen Befehle nötig, da die Dokumentation dann

bereits komplett ausgepackt wurde – dies ist hoffentlich auch bei den künftigen Java-Entwicklerwerkzeugen der Fall.

Obwohl Sie den Großteil von Apples Entwicklerdokumentation nun bei sich auf der Festplatte im Verzeichnis `/Developer/Documentation` bzw. `/Developer/ADC Reference Library` finden, wird hier im Buch meistens mit Webadressen auf Apples Online-Dokumentation verwiesen. Zum einen haben damit auch Entwickler auf anderen Plattformen Zugang zur Apple-Dokumentation, zum anderen greifen Sie damit immer auf die aktuellsten Informationen zu.

Um die API-Dokumentation bequem zu durchstöbern, eignet sich der `JavaBrowser` im Verzeichnis `/Developer/Applications/Java Tools/`, der in Kapitel 8 genauer vorgestellt wird.

## 1.6 Up to date bleiben

Vielleicht ist MacOS X 10.3 »Panther« Ihr erster Kontakt mit einem Apple-Rechner und dem aktuellen Apple-Betriebssystem, vielleicht verwenden Sie Mac OS X aber auch schon seit Version 10.0. So oder so, Sie kennen sicherlich die Tatsache, dass Software (und damit auch ein Betriebssystem) ab einer gewissen Komplexität nie ganz fehlerfrei ist. Außerdem ist Software nie ganz »fertig«, die Entwickler erweitern sie ständig – nach den Anforderungen des Marktes, nach den Wünschen der Anwender, um kompatibel mit einem neuen Standard zu sein.

Dementsprechend stellt auch Apple regelmäßig Aktualisierungen (Updates) für das Betriebssystem und die zugehörigen Programme zur Verfügung. Dies können echte Neuerungen sein, beispielsweise eine neue Java-Version, oder aber Fehlerbereinigungen, die aufgedeckte Sicherheitslücken im System schließen. Am einfachsten können Sie solche Updates von der eingebauten Software-Aktualisierung installieren lassen. Öffnen Sie dazu im Apfel-Menü oder unten am Bildschirmrand im Dock die »Systemeinstellungen« und wählen Sie dort »Software-Aktualisierung« aus. Sie können nun konfigurieren, ob das System regelmäßig automatisch nach Updates suchen soll, und Sie können die Suche durch Anklicken von »Jetzt suchen« manuell starten. Wenn die **Software-Aktualisierung** Updates findet, bekommen Sie diese in einem Dialog angezeigt, wo Sie auswählen können, ob und welche Updates installiert werden sollen.

Ab Mac OS X 10.3 können Sie im Apfel-Menü direkt den Menüpunkt »Software aktualisieren...« auswählen, dann sucht das System sofort nach verfügbaren Updates – Sie sparen sich dadurch ein paar Klicks.



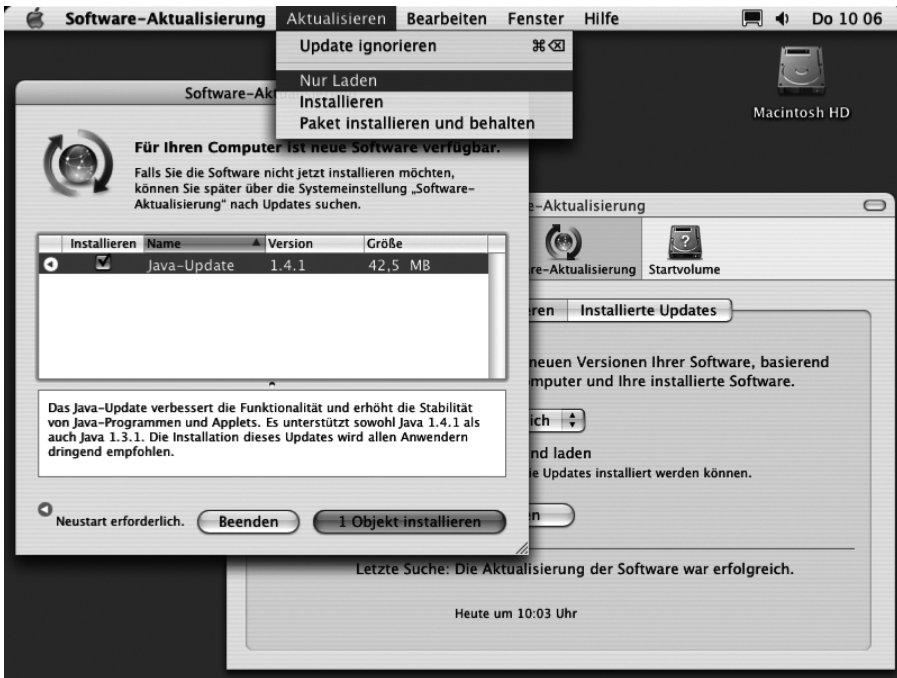


Abbildung 1.12 Mac OS X Software-Aktualisierung

Wenn Sie dagegen lieber die volle Kontrolle über die Downloads der System-Aktualisierungen haben und die Archive dafür manuell herunterladen möchten oder weitere Informationen dazu benötigen, können Sie auf <http://www.info.apple.com/> in Apples Support-Datenbank nachsehen. Dies ist auch der beste Einstiegspunkt, wenn Sie nicht genau wissen, wo Sie direkt Informationen zu bestimmten Apple-Produkten finden können.

Die Aktualisierungen werden also weitestgehend über eine Online-Verbindung ins Internet abgewickelt, ebenso stehen dort aktuelle Dokumentationen zur Verfügung (sofern sie sich nicht bereits auf der Festplatte im /Developer-Verzeichnis befinden). CD-ROMs mit Software-Updates veröffentlicht Apple nur noch sehr selten, und bei gedruckten Entwicklerdokumentationen verlässt sich Apple mittlerweile komplett auf Fachbuch-Verlage<sup>4</sup>.

Wenn Sie ein Java-Produkt entwickeln und verkaufen (oder kostenfrei an andere Anwender abgeben), finden Ihre Kunden unter <http://www.apple.com/java/> allgemeine Hinweise zu Apples Java-Implementierung. Idealerweise bekommen die Anwender Ihrer Software aber gar nichts davon mit, dass sie ein Java-Programm starten. In Kapitel 4, *Ausführbare Programme*, werden Sie

<sup>4</sup> Was einer der Gründe dafür ist, warum Sie dieses Buch aus dem Galileo-Verlag lesen ;-) )

sehen, wie Sie eine Java-Anwendung so »verpacken« können, dass sie wie eine normale Mac-Applikation aussieht. Das Einzige, was reine Anwender tun müssen, ist, ab und zu die Software-Aktualisierung zu starten und vorhandene Updates installieren zu lassen.

Für Sie als Programmierer dürften Apples Entwicklerseiten deutlich interessanter sein. Die Einstiegsseite <http://developer.apple.com/> fasst die wichtigsten Technologien zusammen und gibt einen schnellen Überblick über die letzten Änderungen und Neuerungen. Für die Java-Entwicklung sollten Sie sich die Unterseite <http://developer.apple.com/java/> als Lesezeichen in Ihrem Web-Browser definieren. Hier finden Sie nicht nur die aktuelle Java-Dokumentation, sondern auch Beispielcode und FAQs (Frequently Asked Questions: häufige Fragen und nützliche Antworten).

Damit Entwickler unter sich diskutieren können, hat Apple für Java-Programmierer die **Mailingliste java-dev** eingerichtet, zu der Sie sich unter <http://lists.apple.com/mailman/listinfo/java-dev> anmelden können. Obwohl Sie hier keinen offiziellen Apple-Support bekommen, lesen und schreiben in der Liste auch Apple-Mitarbeiter. Bei dringenden Problemen kann man hier oftmals schnell Hilfe finden. Aber Achtung, das Niveau der Liste ist recht hoch – Sie sollten sich also mit den grundlegenden Java-Problemen bereits selbst beschäftigt haben. Listensprache ist Englisch. Wenn Sie ein deutschsprachiges Forum bevorzugen und Sie sich mit dem Usenet auskennen, können Sie mit einem Newsreader (z.B. Mozilla<sup>5</sup> oder MacSOUP<sup>6</sup>) in den Gruppen *news:de.comp.lang.java* und *news:de.comp.sys.mac.misc* mitdiskutieren, je nachdem, ob Ihre Frage vor allem Java oder vor allem den Macintosh betrifft.

Die wichtigste Quelle für zukünftige, noch nicht öffentlich verfügbare Systemsoftware ist die **Apple Developer Connection (ADC)**. In einem geschützten Bereich können Sie Vorabversionen z.B. von Java-Erweiterungen herunterladen, um Ihre Software damit zu testen und rechtzeitig daran anzupassen. Um in den geschützten Bereich zu gelangen, benötigen Sie eine ADC-Mitgliedschaft. Drei Stufen werden angeboten: Online, Select und Premier. Während die Select- und die Premier-Mitgliedschaft 500 bzw. 3500 US-\$ jährlich kosten (und Ihnen dafür auch entsprechende Leistungen und Vergünstigungen bieten), ist die *Online-Mitgliedschaft kostenlos* und reicht zum Herunterladen der wichtigsten Vorabversionen vollkommen aus. Die ADC-Anmeldung und das Login in den geschützten Bereich geschieht unter <https://connect.apple.com/>.

---

5 Download von <http://www.mozilla.org/>

6 Download von <http://home.snafu.de/stk/macsoup/>

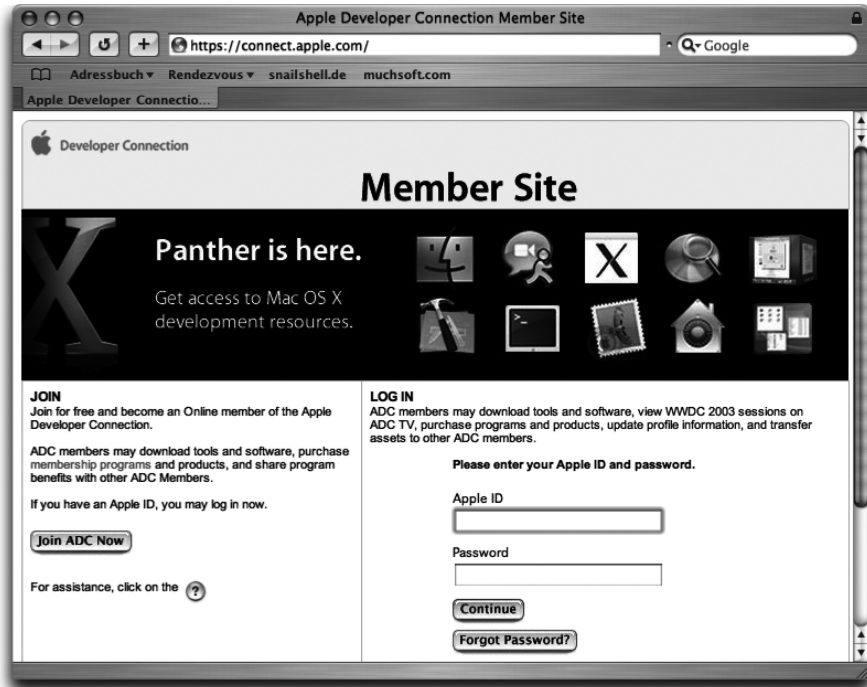


Abbildung 1.13 Apple Developer Connection (ADC): Anmeldung und Login

Sie kommen nun also problemlos an aktuelle Apple-Informationen heran, es fehlen nur noch die neuesten Informationen zu diesem Buch – denn auch das hier Geschriebene unterliegt mit der Software, die es beschreibt, ständigen Änderungen. Sollte es also Ergänzungen oder Korrekturen zu diesem Text geben, finden Sie diese auf <http://www.galileocomputing.de/> bei der Beschreibung dieses Buches als »BuchUpdate«. Wenn Sie sich bei MyGalileo anmelden, können Sie dieses Buch dort registrieren und haben dann alle Zusatzinformationen im einfachen Überblick.

Die Quelltexte der in diesem Buch vorgestellten Beispiele finden Sie auch unter <http://www.muchsoft.com/java/> in der jeweils aktuellen (und gegebenenfalls fehlerkorrigierten) Version.

## 1.7 Mac OS X erkennen

Wenn Sie Ihre Java-Applikation für eine bestimmte Plattform optimieren möchten, müssen Sie entweder für jede Plattform einen speziell angepassten Quelltext pflegen oder Sie pflegen nur einen einzigen Quelltext, in dem Sie zur Laufzeit feststellen, auf welcher Plattform die Software gerade läuft. Natürlich stellt sich die Frage, warum Sie eine Java-Applikation überhaupt an ein

# 7 Grafik und Multimedia

7.1	Java Advanced Imaging (JAI) und Java Image I/O (JIIO) .....	353
7.2	Java 3D .....	356
7.3	OpenGL/JOGL .....	358
7.4	Java Media Framework (JMF) .....	362
7.5	QuickTime for Java (QTJava) .....	365
7.6	Sound/Musik .....	374
7.7	Drucken .....	375
7.8	Literatur & Links .....	380

## 7 Grafik und Multimedia

*»Ich tätowier« dir meinen Namen in allen Farben, in allen Farben/  
Bemal mich, mach mich bunt/Bemal mich bunt!«  
(Etwas)*

Die Grafikausgabe ist mit der Java-Implementation von Mac OS X im Allgemeinen kein Problem. Die einfachen `java.awt.Graphics`-Methoden werden ebenso unterstützt wie Java 2D mit seinem `java.awt.Graphics2D`-Kontext. Diese Grundlagen werden daher im Folgenden nicht mehr speziell behandelt – bei den Beispielsprogrammen der vorangegangenen Kapitel kamen sie teilweise schon zum Einsatz. Wenn Sie bei der Java 2D-Ausgabe optische Feineinstellungen vornehmen wollen, sollten Sie sich im Anhang den Abschnitt »Grafikdarstellung« in Kapitel 16, *System-Properties*, ansehen.

In diesem Kapitel erhalten Sie einen Überblick über spezielle Themen der Grafik- und Multimediaverarbeitung. Zunächst werden zwei Standard-Erweiterungspakete für Bildmanipulation und 3D-Grafik vorgestellt, letzteres wird dann mit der Java-Implementierung des 3D-Standards OpenGL verglichen. Auch für Multimediadaten wird zunächst Suns Java-Standard besprochen, anschließend dann Apples QuickTime-Lösung. Ganz zum Schluss folgen noch Hinweise zur Sound-Ausgabe und zu den Druckmöglichkeiten unter Java.

### 7.1 Java Advanced Imaging (JAI) und Java Image I/O (JIIO)

Die Java Advanced Imaging API stellt im Paket `javax.media.jai` hochperformante, plattformunabhängige Schnittstellen zur Bildbearbeitung zur Verfügung, darunter diverse Bildoperationen wie z.B. Filter. JAI ist für die Bildbearbeitung im Netzwerk optimiert, unterstützt viele Bildformate und die Bilddaten können problemlos mit der Java 2D-Ausgabe gemischt werden. Auf der Seite <http://java.sun.com/products/java-media/jai/> hat Sun eine Liste der Dokumentationen und diverser JAI-Anwendungen zusammengestellt.

Apple hat für Mac OS X 10.3 und Java 1.4 die aktuelle JAI-Version 1.1.2 implementiert, allerdings muss diese separat heruntergeladen und installiert werden. Das Installationspaket für JAI und Java 3D finden Sie auf der Seite <http://docs.info.apple.com/article.html?artnum=120289>.

Die vielleicht am häufigsten genutzten Klassen von JAI dienen zur schnellen Ein- und Ausgabe von Bilddateien (»Image I/O«) und gehören zum Paket

`com.sun.media.jai.codec`. Wie Sie am Namen unschwer erkennen, ist dies kein Standardpaket, daher sollten Sie den Einsatz bei portablen Programmen vermeiden. Zum Glück gibt es ab Java 1.4 aber portablen Ersatz, die Java Image I/O API im Paket `javax.imageio`. Diese Klassen sind fester Bestandteil vom J2SE 1.4 und werden wie folgt genutzt:

```
//CD/examples/ch07/imageio/ImageIOTest/ImageIOTest.java
import java.io.*;
import java.net.*;
import java.awt.image.*;
import javax.imageio.*;
//...
try {
    URL url = new File("galileo_logo.gif").toURI().toURL();
    BufferedImage img = ImageIO.read( url );
    if (!ImageIO.write( img, "jpeg", new File("test.jpg") )) {
        System.exit(1);
    }
}
catch (Exception e) {
    e.printStackTrace();
}
```

Sie rufen also die statischen Methoden `read()` und `write()` in der Klasse `javax.imageio.ImageIO` auf – einfacher geht es kaum. Bei Java 1.4 können Sie sich darauf verlassen, dass GIF, JPEG und PNG geladen und JPEG und PNG gespeichert werden können. Unterstützung für weitere Bildformate (z.B. BMP, JPEG2000 und TIFF) erhalten Sie mit den »Java Advanced Imaging Image I/O Tools« von der Seite <http://java.sun.com/products/java-media/jai/downloads/download-iio.html>. Laden Sie dort das Linux-Archiv herunter und packen Sie das enthaltene Archiv `jai_imageio.jar` auf den Klassenpfad, z.B. in das Verzeichnis `/Library/Java/Extensions/`. Leider hat Apple die nativen JIIO-Codex noch nicht implementiert, so dass derzeit nur die portablen, aber relativ langsamen Java-Routinen verwendet werden.

Für die Ein-/Ausgabe von Bilddaten mit älteren Java-Versionen können Sie JIMI (Java Image Management Interface) verwenden, das Sun auf der Seite <http://java.sun.com/products/jimi/> zum Download anbietet. JIMI unterstützt recht viele Formate, und vor allem können Sie damit auch PICT-Bilder verarbeiten. PICT ist ein relativ altes Apple-Bildformat, das sowohl Bitmap- als auch Vektordaten speichern kann und das auch von Mac OS X noch verwendet wird. Der Quelltext zum Einlesen einer PICT-Datei ist kurz und übersichtlich:

```

import java.io.*;
import com.sun.jimi.core.*;
//...
    InputStream in = new FileInputStream( "mein.pict" );
    java.awt.Image img =
        Jimi.getImage( in, "image/pict", Jimi.VIRTUAL_MEMORY );

```

Wenn Sie Bilder möglichst performant darstellen wollen, sollten Sie die Bilddaten nach dem Laden in ein Format umwandeln, das von der Grafikhardware ohne großen Aufwand verarbeitet werden kann. Dazu laden Sie das Bild am besten wie oben beschrieben mit `ImageIO.read()` und geben die geladene `BufferedImage`-Referenz `img` dann in eine optimierte Bitmap aus. Die optimierten Bilddaten lassen Sie dann wie gewohnt innerhalb einer Komponente mit `Graphics.drawImage()` zeichnen:

```

import java.awt.*;
import java.awt.image.*;
//...
    BufferedImage img = javax.imageio.ImageIO.read( url );
    BufferedImage optimiert =
        GraphicsEnvironment.getLocalGraphicsEnvironment()
            .getDefaultScreenDevice().getDefaultConfiguration()
            .createCompatibleImage( img.getWidth(), img.getHeight() );
    optimiert.getGraphics().drawImage( img, 0,0, this );

```

Auch wenn der MIME-Typ für das PICT-Format mittlerweile `image/x-pict` lautet, sollten Sie bei JIMI noch die früher verwendete Zeichenkette angeben. Wenn Sie das Bild aus einer anderen Quelle als einer PICT-Datei einlesen, müssen Sie darauf achten, dass vor den eigentlichen Bilddaten ein 512 Bytes langer Header mit Nullbytes gesendet wird.

Leider besitzt JIMI beim Einlesen von PICTs einige Fehler, so dass PICT-Dateien mit der PackBits-Kompression nicht verwendet werden können. Auch beim Schreiben der Dateien enthält der Originalquelltext Fehler, für die es allerdings unter <http://www.amug.org/~glguerin/other/index.html#PICTWriter> eine Korrektur gibt. Eine andere Möglichkeit zum Bearbeiten von PICT-Bildern bietet QTJava, das weiter hinten in diesem Kapitel vorgestellt wird.

## 7.2 Java 3D

Java 3D ist ein Standard-Erweiterungspaket, das im Paket `javax.media.j3d` eine ausgefeilte objektorientierte Schnittstelle für die 3D-Grafik-Programmierung anbietet. Natürlich sollten alle Java-Bibliotheken mehr oder weniger objektorientiert sein, aber Sie werden im folgenden Abschnitt zu OpenGL sehen, dass es auch Grafikbibliotheken gibt, bei denen Sie sich deutlich mehr mit den Details beschäftigen müssen. Suns Portalseite für Java 3D ist <http://java.sun.com/products/java-media/3D/> – etwas versteckt befindet sich dort auch die Seite <http://java.sun.com/products/java-media/3D/collateral/> mit einer ausführlichen Anleitung und vielen Beispielen.

Java 3D wird von Apple zusammen mit JAI als Installationspaket angeboten, das Sie unter anderem von <http://www.apple.com/downloads/macosx/apple/java3dandjavaadvancedimagingupdate.html> herunterladen können. Wie bei JAI wird MacOS X 10.3 mit Java 1.4 unterstützt, die von Apple implementierte Java 3D-Version 1.3.1 ist die derzeit aktuelle.

In der Praxis wird häufig OpenGL als programmiersprachenübergreifender Standard eingesetzt. Dass man aber auch gut mit Java 3D entwickeln kann, zeigt folgendes Programm, das an ein Beispiel aus der oben erwähnten Java 3D-Anleitung angelehnt ist:

```
//CD/examples/ch07/java3d/Java3Dtest/Java3Dtest.java
import java.awt.*;
import javax.media.j3d.*;
import com.sun.j3d.utils.geometry.ColorCube;
import com.sun.j3d.utils.universe.SimpleUniverse;
```

Die Klassen aus den Unterpaketen von `com.sun.j3d` gehören zwar nicht direkt zum Java 3D-Standard, aber sie werden auch bei der MacOS X-Implementation mitgeliefert. Sie fassen häufige Anwendungsfälle zusammen und erleichtern die Programmierung mitunter deutlich.

```
public class Java3DTest extends Frame {
    public Java3DTest() {
        GraphicsConfiguration config =
            SimpleUniverse.getPreferredConfiguration();
        Canvas3D canvas = new Canvas3D( config );
        this.add( canvas, BorderLayout.CENTER );
        BranchGroup scene = this.createSceneGraph();
        scene.compile();
        SimpleUniverse universe = new SimpleUniverse( canvas );
```



```

        universe.getViewingPlatform().setNominalViewingTransform();
        universe.addBranchGraph( scene );
    }

```

Die Hilfsklasse `SimpleUniverse` erzeugt zunächst eine passende Standard-Grafikkonfiguration, damit Sie sich um die Details nicht kümmern müssen. Damit wird dann ein `Canvas3D`-Objekt erzeugt, eine Ausgabekomponente für Java 3D-Grafik. Als Nächstes wird ein so genannter Szene-Graph zusammengestellt, der die darzustellenden Elemente und deren Transformationen speichert – die Methode `createSceneGraph()` müssen Sie selbst implementieren, sie wird gleich noch ausführlicher vorgestellt. Die Szene wird dann noch kompiliert, damit die Anzeige einigermaßen schnell vonstatten geht. Am Ende wird dann ein `SimpleUniverse`-Objekt erzeugt, das die eigentlichen Ausgaben vornimmt und das dazu sowohl mit dem `Canvas3D`- als auch mit dem Szene-Objekt verknüpft wird. Außerdem wird der Beobachtungspunkt so im Raum verschoben, dass man einen möglichst guten Blick auf die Szene hat.

```

public BranchGroup createSceneGraph() {
    BranchGroup scene = new BranchGroup();
    Transform3D transformation = new Transform3D();
    transformation.rotX( Math.PI / 4.0 );
    TransformGroup rotation =
        new TransformGroup( transformation );
    rotation.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
    Alpha zeit = new Alpha( -1, 5000 );
    RotationInterpolator animation =
        new RotationInterpolator( zeit, rotation );
    BoundingSphere bounds = new BoundingSphere();
    animation.setSchedulingBounds( bounds );
    rotation.addChild( new ColorCube( 0.4 ) );
    rotation.addChild( animation );
    scene.addChild( rotation );
    return scene;
}
//...
}

```

In dieser Methode wird der Szene-Graph zusammengebaut, der als Wurzel ein `BranchGroup`-Objekt besitzt. Die Szene setzt sich nicht nur aus den darzustellenden Elementen zusammen, sondern auch aus Transformationen und Animationen. Hier wird zunächst eine 3D-Transformation definiert, die das später zugeordnete Element um 45 Grad ( $2\pi/8$ ) entgegen dem Uhrzeigersinn um die

X-Achse dreht. Dann wird eine zusätzliche `TransformationGroup` angelegt, die sicherstellt, dass die Werte nach der Transformation wieder in der Gruppe gespeichert werden dürfen (`ALLOW_TRANSFORM_WRITE`).

Für die Animation benötigen Sie eine `Alpha`-Zeitfunktion. Hier wird eine Funktion definiert, die in 5.000 Millisekunden vom Anfang bis zum Ende durchläuft und die dann endlos wieder von vorne startet (das legt die `-1` fest). Mit der Zeitfunktion und der Rotationstransformation erzeugen Sie nun ein `RotationInterpolator`-Animationsobjekt, dem Sie durch das `BoundingSphere`-Objekt sagen, innerhalb welchen Bereichs die Animation stattfindet (hier im Standardbereich; in einer Kugel mit Radius 1 um den Ursprung).

Endlich wird auch das darzustellende Element erzeugt – ein `ColorCube`-Objekt mit passender Skalierung, das automatisch unterschiedlich eingefärbte Seiten besitzt. Dieses Würfel-Objekt wird mit den Transformationen in umgekehrter Reihenfolge ineinander verschachtelt und schließlich zu der Szene hinzugefügt.

Wenn Sie das Programm nun übersetzen und ausführen, sollten Sie in etwa Abbildung 7.1 sehen. Das Bemerkenswerte dabei ist, dass die Animation nirgendwo explizit gestartet wird – als Bestandteil der Szene beginnt sie einfach automatisch, sobald die Szene angezeigt wird.

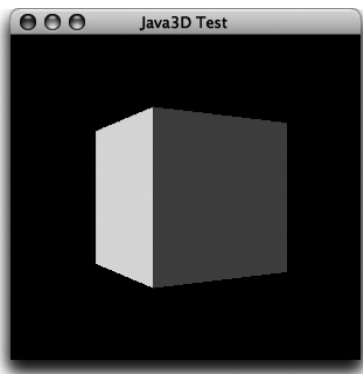


Abbildung 7.1 Drehender Würfel mit Java 3D

### 7.3 OpenGL/JOGL

Wenn Ihnen Java 3D schon recht mathematisch und aufwändig erschien, wird Ihnen die OpenGL-API erst recht etwas »roh« vorkommen – die Programmierschnittstellen sind sehr grundlegend, und es dauert deutlich länger als beispielsweise bei Java 3D, bis Sie eine komplexe Szene realisiert haben. Nichtsdestotrotz ist OpenGL seit Anfang der 1990er Jahre ein 2D- und 3D-Grafikstandard für viele Plattformen und Programmiersprachen, was vor allem

an der Geschwindigkeit, den guten Optimierungsmöglichkeiten und der vollen Kontrolle über viele Details der Darstellung liegt.

OpenGL ist fester Bestandteil von Mac OS X, Sie finden einige Hilfsprogramme (Profiler und ein Konstruktionsprogramm) im Order `/Developer/Applications/Graphics Tools/` auf Ihrer Festplatte. Auch aus Java heraus können Sie OpenGL nutzen, da diverse Anbindungen entwickelt wurden. Früher wurde vor allem GL4Java (<http://jausoft.com/gl4java/>) eingesetzt, mittlerweile hat sich »JOGL« als Quasi-Standard etabliert. Allgemeine Informationen zu JOGL finden Sie auf der Seite <https://jogl.dev.java.net/>, die Mac OS X-Portierung können Sie als Komplettpaket auch von <http://homepage.mac.com/gziemski/projects/> herunterladen.

Die heruntergeladenen Dateien installieren Sie am besten im Verzeichnis `/Library/Java/Extensions/`. Dort müssen sich mindestens die Bibliotheken `jogl.jar` und `libjogl.jnilib` befinden. Wenn Sie auch die drei Demo-JAR-Archive in dieses Verzeichnis kopieren, können Sie mit dem Befehl

```
java demos.vertexProgRefract.VertexProgRefract
```

ein schönes Beispiel für die Möglichkeiten von OpenGL starten. Die Beschreibungen aller verfügbaren Demoprogramme finden Sie auf der Seite <https://jogl-demos.dev.java.net/>. Aber wie können Sie nun selber eine kleine JOGL-Anwendung schreiben?

```
//CD/examples/ch07/jogl/JoglTest/JoglTest.java
import java.awt.*;
import net.java.games.jogl.*;
public class JoglTest extends Frame implements GLEventListener {
    public JoglTest() {
        super ("JOGL Test");
        GLCapabilities capabilities = new GLCapabilities();
        GLCanvas canvas =
            GLDrawableFactory.getFactory().createGLCanvas(capabilities);
        canvas.addGLEventListener(this);
        this.add( canvas, BorderLayout.CENTER );
    }
}
```

Das JOGL-Paket `net.java.games.jogl` deutet schon darauf hin, dass OpenGL häufig für Spiele verwendet wird, aber auch in Bereichen wie CAD, Simulation und Visualisierung kommt es zum Einsatz. Im Konstruktor erzeugen Sie zunächst ein `GLCapabilities`-Objekt, das die Standardvorgaben für die Grafikkonfiguration liefert. Damit lassen Sie dann ein `GLCanvas`-Objekt generieren, in dem die eigentliche Ausgabe der Grafikelemente stattfindet. Wenn

ein `GLCanvas` angezeigt wird, meldet er die vier Ereignisse `init()`, `display()`, `displayChanged()` und `reshaped()` an einen registrierten Listener, daher wird das entsprechende Interface `GLEventListener` von der Fensterklasse implementiert.

```
public void init(GLDrawable drawable) {
    GL gl = drawable.getGL();
    gl.glClearColor( 1.0F, 1.0F, 1.0F, 1.0F );
    gl.glColor3f( 0.0F, 0.0F, 1.0F );
}
```

Zur Initialisierung setzen Sie die Hintergrundfarbe mit `glClearColor()` auf Weiß (alle Farbkanäle sowie der Alpha-Kanal werden mit `1.0F` auf volle Intensität gebracht) und die Zeichenfarbe mit `glColor3f()` auf Blau. Die Methoden rufen Sie auf dem GL-Kontext auf – dies ist die grundlegende Schnittstelle zu OpenGL mit einer schier unglaublichen Anzahl von Grafikroutinen.

```
public void display(GLDrawable drawable) {
    GL gl = drawable.getGL();
    gl.glClear( GL.GL_COLOR_BUFFER_BIT );
    gl.glBegin( GL.GL_LINE_LOOP );
    gl.glVertex2i( 10, 20 );
    gl.glVertex2i( 80, 10 );
    gl.glVertex2i( 90, 80 );
    gl.glVertex2i( 20, 90 );
    gl.glEnd();
}
```

Wenn JOGL die `display()`-Methode aufruft, müssen Sie die Grafikelemente darstellen. Sie könnten die Ausgabe puffern, hier wird aber einfach bei jedem Aufruf die komplette Grafik neu gezeichnet. Zwischen `glBegin()` und `glEnd()` geben Sie dazu alle nötigen Informationen für das Grafikelement an – in diesem Fall werden Punkte für einen geschlossenen Linienzug (`GL_LINE_LOOP`) festgelegt. Diese Methode erinnert an die `paint()`-Methode und den `Graphics`-Kontext von AWT-Komponenten.

```
public void reshape(GLDrawable drawable,
                    int x, int y, int width, int height) {
    GL gl = drawable.getGL();
    GLU glu = drawable.getGLU();
    gl.glViewport( 0, 0, width, height );
    gl.glMatrixMode( GL.GL_PROJECTION );
```

```

gl.glLoadIdentity();
glu.gluOrtho2D( 0.0, 100.0, 0.0, 100.0 );
}

```

Wenn sich die Größe des GLCanvas-Objektes verändert, weil beispielsweise das Fenster breiter oder schmaler gemacht wurde, ruft JOGL die Methode `reshape()` auf. `glViewport()` setzt den Ausgabebereich auf die neue Breite und Höhe. Die Methode `gluOrtho2D()`, die zu den OpenGL-Hilfsroutinen gehört, legt das 2D-Koordinatensystem auf jeder Achse auf den Bereich 0 bis 100 fest.

```

public void displayChanged(GLDrawable drawable,
                           boolean modeChanged,
                           boolean deviceChanged) { }

```

Mit `displayChanged()` kann JOGL signalisieren, dass sich die Konfiguration des Ausgabegeräts verändert hat, beispielsweise die Farbtiefe oder die Bildschirmauflösung. Allerdings ist diese Funktionalität derzeit nicht in JOGL implementiert.

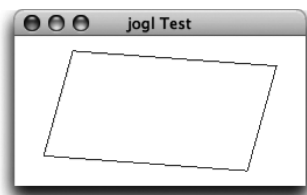


Abbildung 7.2 2D-Ausgabe mit OpenGL bzw. JOGL

Wenn Sie das Programm ausführen, sehen Sie ein Viereck, das sich durch die in `reshape()` definierten Matrix-Transformationen automatisch an die Größe des Fensters anpasst (siehe Abbildung 7.2). Alles in allem haben Sie – verglichen mit Java 3D – bei JOGL mehr Aufwand für weniger »Effekte« getrieben, aber gerade wenn Sie Spiele oder andere zeitkritische grafische Anwendungen schreiben wollen, lohnt sich der Aufwand meistens. Weitere JOGL-Einführungen finden Sie zahlreich im Internet, unter anderem auf den Seiten <http://today.java.net/pub/a/today/2003/09/11/jogl2d.html>, <http://today.java.net/pub/a/today/2004/03/18/jogl-2d-animation.html> und <http://www.genedavissoftware.com/books/jogl/>.

## 7.4 Java Media Framework (JMF)

Mit dem Java Media Framework aus dem Standard-Erweiterungspaket `javax.media` können Sie Audio-, Video- und andere zeitbasierte Mediendaten in Ihre Java-Applikationen integrieren. Die auf der Seite <http://java.sun.com/products/java-media/jmf/> beschriebenen Eigenschaften lesen sich gut, und die Liste der unterstützten Formate erscheint ausreichend lang. Leider jedoch können bei weitem nicht alle der verfügbaren AVI-, MPEG- und QuickTime-Dateien dekodiert werden, weshalb oft lieber das im folgenden Abschnitt vorgestellte QuickTime for Java verwendet wird. Dennoch bietet JMF einen unschlagbaren Vorteil: Es kann portabel auf jeder J2SE-Implementation eingesetzt werden.

Die aktuelle JMF-Version 2.1.1e können Sie von der Seite <http://java.sun.com/products/java-media/jmf/2.1.1/download.html> herunterladen. Für Mac OS X müssen Sie das »Cross-platform Java«-Archiv verwenden, es steht leider kein »Performance Pack« wie für andere Systeme zur Verfügung. Wenn Sie den ausgepackten Ordner in Ihr Benutzerverzeichnis legen, können Sie mit den Befehlen

```
setenv JMFHOME ~/JMF-2.1.1e
java -classpath $JMFHOME/lib/jmf.jar JMFRegistry
```

testen, ob die JMF-Installation auf Ihrem Rechner läuft. Es erscheint der »JMF Registry Editor« (siehe Abbildung 7.3), mit dem Sie unter anderem die installierten Format-Module verwalten können.

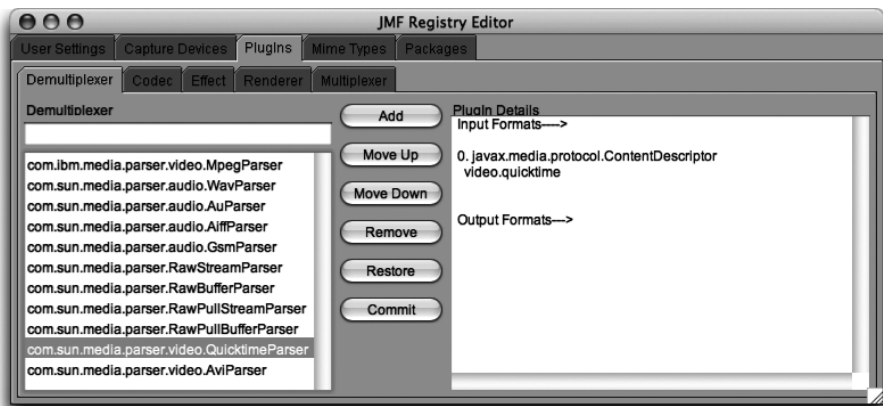


Abbildung 7.3 Registry Editor zur Konfiguration vom JMF

Für die einfachere Verwendung in Ihren Applikationen kopieren Sie aber am besten das Archiv `jmf.jar` in das Verzeichnis `/Library/Java/Extensions/`

(oder Sie packen das Archiv in das jeweilige Programmpaket). Damit können Sie dann das folgende Beispiel starten, ein einfaches Abspielprogramm sowohl für Video- als auch für Audiodaten:

```
//CD/examples/ch07/jmf/SimpleJMFPlayer/SimpleJMFPlayer.java
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import javax.media.*;
public class SimpleJMFPlayer extends Frame
    implements ControllerListener, WindowListener {
    private Player player = null;
    private Component visualComponent = null;
    private Component controlComponent = null;
```

Zunächst benötigen Sie ein `Player`-Objekt, das das Abspielen im Hintergrund steuert. Von diesem Objekt können Sie dann zwei Komponenten erfragen, die in die Benutzungsoberfläche integriert werden: Eine, die den Film selbst anzeigt, und eine für die Elemente, mit denen der Anwender den Film starten und stoppen kann. Die Kommunikation zwischen dem `Player` und der Oberfläche erfolgt über die im Interface `ControllerListener` definierte Methode `controllerUpdate()`.

```
public SimpleJMFPlayer(URL url) {
    try {
        player = Manager.createPlayer( url );
        player.addControllerListener(this);
        player.start();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    this.addWindowListener(this);
}
```

Der Konstruktor lässt am Anfang das nötige `Player`-Objekt erzeugen, wobei die URL der abzuspielenden Daten übergeben wird. Danach wird das Fenster-Objekt als Listener am `Player` registriert und die Wiedergabe gestartet. Hier fehlt zunächst noch die Verknüpfung mit der Oberfläche, diese wird in der folgenden Methode hergestellt.

```
public synchronized void controllerUpdate
    (ControllerEvent event) {
```

```

if (player == null) return;
if (event instanceof RealizeCompleteEvent) {
    visualComponent = player.getVisualComponent();
    controlComponent = player.getControlPanelComponent();
    if (visualComponent != null) {
        this.add( visualComponent, BorderLayout.CENTER );
    }
    if (controlComponent != null) {
        this.add( controlComponent, BorderLayout.SOUTH );
    }
    this.invalidate();
    this.pack();
}
else if (event instanceof EndOfMediaEvent) {
    player.setMediaTime( new Time(0) );
}
}

```

Für die verschiedenen Ereignisse (Fertig zur Anzeige, Ende der Wiedergabe usw.) ruft JMF die Methode `controllerUpdate()` auf und übergibt als Parameter geeignete `ControllerEvent`-Unterklassen. Am interessantesten ist der `RealizeCompleteEvent`, mit dem Sie aufgefordert werden, die Abspielkomponenten in die Oberfläche zu integrieren. Dazu fragen Sie die beiden Komponenten mit `getVisualComponent()` und `getControlPanelComponent()` ab, fügen sie zum `Frame` hinzu und lassen das Layout neu validieren – fertig.

Der zweite gezeigte `EndOfMediaEvent` tritt beim Ende der Wiedergabe auf. Hier wird in diesem Fall einfach nur der Film ganz »zurückgespult«, indem die Wiedergabezeit auf Null gesetzt wird.

```

public void stop() {
    if (player != null) {
        player.stop();
        player.deallocate();
    }
}

```

Wenn die Wiedergabe von außen abbrechbar sein soll, können Sie diese `stop()`-Methode zur Verfügung stellen. Dabei sollten Sie nicht nur das `Player`-Objekt stoppen, sondern mit `deallocate()` auch nicht mehr benötigten Speicher sofort freigeben.



Bevor das Programm beendet wird, rufen Sie `close()` auf, wodurch das `Player`-Objekt interne Aufräumarbeiten durchführen kann:

```
public void windowClosing(WindowEvent e) {
    if (player != null) {
        this.stop();
        player.close();
    }
    this.setVisible(false);
    this.dispose();
    System.exit(0);
}
//...
}
```

Nun können Sie das Abspielprogramm starten, und in dem Fenster wird nicht nur der Film an sich angezeigt, sondern auch die Steuerungselemente (siehe Abbildung 7.4) – beides hatten Sie ja in der Methode `controllerUpdate()` zum `Frame`-Layout hinzugefügt.



Abbildung 7.4 JMF spielt einen QuickTime-Film ab.

Um das Problem der schlecht unterstützten Medienformate zu umgehen, beschreibt der Artikel <http://www.onjava.com/pub/a/onjava/2002/12/23/jmf.html> die interessante Möglichkeit, QuickTime als JMF-Plugin für die Kodierungen und Dekodierungen zu verwenden. Wenn Sie allerdings für Ihre Filme die Standard-JMF-API nur zusammen mit einer systemabhängigen Bibliothek nutzen können, können Sie auch gleich QuickTime verwenden.

## 7.5 QuickTime for Java (QTJava)

QuickTime (QT) ist wohl eine der ältesten und ausgereiftesten Multimedia-bibliotheken, die es gibt. Seit Anfang der 1990er Jahre steht die Software für Mac OS und Windows zur Verfügung<sup>1</sup>, und bei Mac OS X ist sie fester Bestand-

<sup>1</sup> Siehe <http://david.egbert.name/work/newmedia/quicktime/history/>

teil seit der ersten Version dieses Betriebssystems. Am bekanntesten ist QuickTime sicherlich dafür, Filmdateien abzuspielen (beispielsweise im QuickTime-eigenen \*.mov-Format), aber es eignet sich genauso gut dazu, Musikdateien anzuhören, Bilddateien zu betrachten und zwischen einer Vielzahl von Formaten zu konvertieren, Audio- und Videodaten als Echtzeit-Datenstrom zu empfangen sowie 3D- und Virtual-Reality-Modelle zu bearbeiten. Mittlerweile ist QuickTime bei Version 6.5 angekommen, das aktuellste Installationspaket können Sie von der Seite <http://www.apple.com/quicktime/download/> herunterladen. Die Abspielsoftware QuickTime Player finden Sie im Verzeichnis /Programme/, die Konfiguration nehmen Sie über **Systemeinstellungen · QuickTime** vor.

Seit 1998 (damals war QuickTime 3.0 aktuell) gibt es eine Java-Schnittstelle zu dieser Technologie. Apple stellt mit »QuickTime for Java«, kurz »QTJava« oder »QTJ«, eine objektorientierte Zugriffsschicht auf die darunter liegende, prozedurale Programmierschnittstelle zur Verfügung. QTJava ist also kein reines Java und damit eigentlich nicht plattformunabhängig. Wenn Sie aber als Zielplattformen Mac OS (X) und Windows voraussetzen können, ist QuickTime die bessere Wahl gegenüber dem Java Media Framework, denn die Funktionalität und die Anzahl der unterstützten Formate sind deutlich höher. Linux-Anwender bleiben bei QuickTime derzeit leider komplett außen vor.

Mit QuickTime 6.4 hat Apple eine Veränderung an den QTJava-APIs vorgenommen, um die Schnittstellen zu vereinfachen und an Java 1.4 anzupassen – vieles, was QTJava vor einigen Jahren noch selbst mitbringen musste, ist mittlerweile Bestandteil der Standard-Klassenbibliothek. Insofern ist eine solche Schlankheitskur eigentlich zu begrüßen. Leider jedoch wurden die betreffenden Klassen und Methoden nicht nur einfach als veraltet (»deprecated«) markiert, einige wichtige alte Routinen verweigern nun mit Java 1.4 komplett ihren Dienst. Dies betrifft vor allem die Klassen aus dem Paket `quicktime.app.display`, die durch andere Klassen im Paket `quicktime.app.view` ersetzt wurden. Nahezu alle Java-Quelltexte müssen daran angepasst werden, wenn die Anwendungen auch mit einer aktuellen QuickTime-Version und Java 1.4 laufen sollen. Dass Sie QuickTime 6.4 benötigen, damit Java 1.4 unterstützt wird, betrifft übrigens nur Mac OS X – die Windows-Variante von QTJava kommt bereits seit QuickTime 6.0 mit dem JDK 1.4 klar.

Die Versionsnummern von QuickTime und von QTJava werden nicht synchron hochgezählt, auch wenn QTJava oft zusammen mit einer neuen QuickTime-Version aktualisiert wird. Zu QuickTime 6.4 und 6.5 gehört beispielsweise QTJava 6.1, zu QuickTime 6.3 die QTJava-Version 6.0. Die nötigen Java-Bibliotheken werden normalerweise vom QuickTime-Installationsprogramm auf die

Festplatte geschrieben. Einzige Ausnahme ist Mac OS X 10.2 – dort müssen Sie bei QuickTime 6.4 die Software-Aktualisierung bemühen, um QTJava nachträglich zu installieren.

Apples Portalseite für QTJava ist <http://developer.apple.com/quicktime/qtjava/>. Passen Sie aber auf, welche der Informationen Sie von dort verwenden, denn teilweise wurden die Texte noch nicht an MacOS X angepasst. Am besten laden Sie sich zunächst die Javadoc-Dokumentation von der Seite <http://developer.apple.com/quicktime/qtjava/javadocs.html> herunter. Sie finden dort zwei Versionen, QTJava 6.0 und QTJava 6.1. Da noch viele Quelltexte für die ältere QTJava-Version programmiert sind, wird im folgenden Beispiel ein einfacher Film-Abspieler für QTJava 6.0 entwickelt. Beachten Sie aber, dass Sie das Programm unter Mac OS X dann zwingend mit Java 1.3 ausführen müssen, sonst erhalten Sie Laufzeit-Fehlermeldungen.

### 7.5.1 QTJava 6.0 (QuickTime bis Version 6.3)

```
//CD/examples/ch07/qtjava/QTJava60Player/QTJava60Player.java
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import quicktime.*;
import quicktime.app.display.*;
import quicktime.app.QTFactory;
import quicktime.app.image.ImageDrawer;
```

Neben den typischen `import`-Anweisungen für eine AWT-Applikation mit Datei- und Netzwerkzugriffen wird vor allem das Paket `quicktime` mit einigen Unterpaketen eingebunden.

```
public class QTJava60Player extends Frame implements WindowListener {
    private Drawable myQTContent = null;
    private QTCanvas myQTCanvas = null;
```

Jedes QTJava60Player-Fenster hält zwei Referenzen. Das `Drawable`-Objekt verwaltet die darzustellenden Daten, beispielsweise einen Film. Der `QTCanvas` ist eine ganz gewöhnliche AWT-Komponente, mit der das `Drawable`-Objekt dann innerhalb der Benutzungsoberfläche angezeigt wird.

```
public QTJava60Player(String url) {
    try {
        QTSession.open();
```

```

myQTCanvas =
    new QTCanvas( QTCanvas.kInitialSize, 0.5F, 0.5F );
this.add ( myQTCanvas, BorderLayout.CENTER );
myQTContent = QTFactory.makeDrawable( url );
}

```

Bevor Sie irgendwelche QTJava-Objekte anlegen, müssen Sie zur Initialisierung von QuickTime ganz zu Anfang `QTSession.open()` aufrufen. Danach wird die `QTCanvas`-Komponente erzeugt, die dann in den `Frame` eingefügt wird. Mit `kInitialSize` wird die anfängliche Größe festgelegt, die beiden 0.5-Werte zentrieren den Film horizontal und vertikal innerhalb der Komponente. Schließlich wird mit `QTFactory.makeDrawable()` aus einer beliebigen URL ein darstellbares Multimedia-Objekt erzeugt. Was hier noch fehlt, ist die Verknüpfung zwischen dem `QTCanvas` und dem `Drawable`-Objekt.

```

catch (QTEException e) {
    if (QTSession.isInitialized()) {
        myQTContent = ImageDrawer.getQTLogo();
    }
    else {
        throw new RuntimeException( e.getMessage() );
    }
}

```

Aber zunächst prüfen Sie, ob ein Fehler aufgetreten ist. Wenn die `QTSession` initialisiert ist, wurde eventuell die Filmdatei nicht gefunden oder der Inhalt ist defekt. In diesem Fall wird einfach das QuickTime-Logo angezeigt, das sich mit einer `ImageDrawer`-Methode anfragen lässt. Ansonsten wird das Programm mit einer Fehlermeldung beendet.

```

if (myQTCanvas != null) {
    try {
        myQTCanvas.setClient( myQTContent, true );
    }
    catch (QTEException e) {
        throw new RuntimeException( e.getMessage() );
    }
}
this.addWindowListener(this);
}

```

Konnte QuickTime initialisiert und das `QTCanvas`-Objekt erzeugt werden, sagen Sie dem `QTCanvas`-Objekt nun mittels `setClient()`, welche Multi-

mediadaten in der Komponente angezeigt werden sollen. Der `true`-Parameter bewirkt, dass das Layout der AWT-Komponente mit den gesetzten Client-Daten neu berechnet wird.

Damit kann Ihr Frame bereits einen QuickTime-Film abspielen! Es bedarf keiner speziellen Methoden-Aufrufe zum Starten der Wiedergabe – sobald der `QTCanvas` angezeigt wird, werden auch automatisch verwaltete Steuerungselemente dargestellt, mit denen der Anwender das Abspielen kontrollieren kann. Was nun noch folgt, ist Code zum korrekten Beenden der Wiedergabe.

```
public void stop() {
    if (myQTCanvas != null) {
        myQTCanvas.removeClient();
        myQTCanvas = null;
    }
}
```

Die Methode `stop()` kann von außen aufgerufen werden, wenn die Wiedergabe abgebrochen werden soll – beispielsweise weil der Anwender einen entsprechenden Knopf angeklickt hat oder weil ein Applet-Dokument geschlossen wird. Dazu wird einfach die Verbindung zwischen dem `QTCanvas` und den Client-Daten mit der Methode `removeClient()` aufgelöst.

```
public void windowClosing(WindowEvent e) {
    this.stop();
    QTSession.close();
    this.setVisible(false);
    this.dispose();
    System.exit(0);
}
//...
}
```

Spätestens wenn Ihre Applikation beendet wird, sollten Sie die `QTSession` schließen, damit QuickTime Aufräumarbeiten durchführen kann. Anschließend können Sie dann – wie hier geschehen – das Fenster schließen und das Programm beenden.

Damit Sie das Programm kompilieren können, müssen Sie noch die `QTJava`-Klassen zum Suchpfad hinzufügen. In Xcode tragen Sie dazu das Archiv `/System/Library/Java/Extensions/QTJava.zip` in den Target-Einstellungen unter **Search Paths · Java Classes** ein (siehe Abbildung 7.5). Zum Ausführen müssen Sie dieses ZIP-Archiv später dann nicht mehr extra angeben, da es sich in einem der Standard-Erweiterungsverzeichnisse befindet.

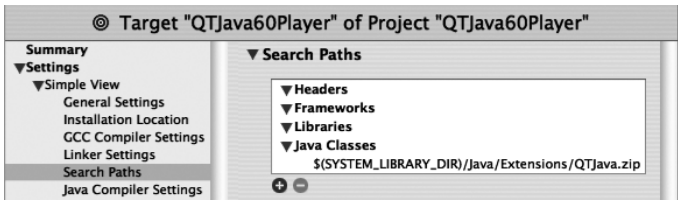


Abbildung 7.5 QTJava-Klassen zum Suchpfad hinzufügen

Da Sie hier mit veralteten Schnittstellen arbeiten, sollten Sie in den Xcode-Target-Einstellungen unter »Java Compiler Settings« den Eintrag »Show usage of deprecated API« aktivieren. So zeigt Ihnen Xcode nach dem Übersetzen neben dem Quelltext, welche Stellen kritisch für die Aufwärtskompatibilität sind. Wenn Sie `javac` direkt aufrufen, können Sie dafür die Option `-deprecation` setzen.

Sie werden sehen, dass Sie das Interface `quicktime.app.display.Drawable`, die Klasse `quicktime.app.QTFactory` und vor allem die Klasse `quicktime.app.display.QTCanvas` bei neuen Projekten nicht mehr einsetzen sollten. Geeigneter Ersatz wird Ihnen im folgenden Abschnitt vorgestellt.

Vor dem Ausführen müssen Sie noch sicherstellen, dass die richtige Java-Version gestartet wird. Dazu ändern Sie beim Executable »java« den Programmpfad unter »Path to Executable« auf `/System/Library/Frameworks/JavaVM.framework/Versions/1.3.1/Commands/java` (siehe Abbildung 7.6).

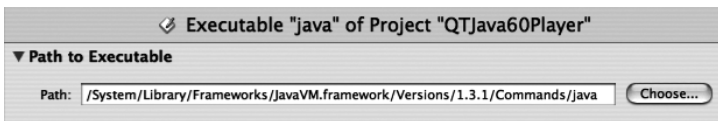


Abbildung 7.6 QTJava 6.0 mit Java 1.3 ausführen

Wenn Sie das Programm nun übersetzen und starten, öffnet sich ein Fenster, in dem neben dem Medien-Ausgabebereich automatisch auch eine Steuerungsleiste eingeblendet wird (siehe Abbildung 7.7). Mit dem Dreieck-Knopf können Sie die Wiedergabe starten (und auch wieder anhalten) – um die korrekte Behandlung der entsprechenden Ereignisse kümmert sich QTJava.



Abbildung 7.7 Filme mit QTJava 6.0 abspielen

### 7.5.2 QTJava 6.1 (QuickTime ab Version 6.4)

Um ein solches Abspielprogramm mit dem aktuellen QTJava 6.1 zu realisieren, müssen Sie vor allem den Konstruktor der Klasse anpassen – hier wird ja das Film-Objekt erzeugt und in die Oberfläche eingebunden. Wenn Sie Java 1.4 verwenden, sind diese Änderungen zwingend. Und da die vorgestellten Klassen auch mit Java 1.3 funktionieren, sollten Neuentwicklungen immer wie folgt aufgebaut sein:

```
//CD/examples/ch07/qtjava/QTJava61Player/QTJava61Player.java
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import quicktime.*;
import quicktime.app.view.*;
import quicktime.std.*;
import quicktime.std.movies.*;
import quicktime.std.movies.media.DataRef;
```

Es werden die typischen Pakete eingebunden, dazu das QTJava 6.1-Paket `quicktime.app.view` sowie das ältere (aber immer noch gültige) Paket `quicktime.std` für diverse Konstanten.

```
public QTJava61Player(String url) {
    try {
        QTSession.open();
        DataRef ref = new DataRef( url );
        Movie movie =
            Movie.fromDataRef( ref,
```

```

        StdQTConstants4.newMovieAsyncOK |
        StdQTConstants.newMovieActive );
MovieController ctrl =
    new MovieController( movie );
QTComponent component =
    QTFactory.makeQTComponent( ctrl );
this.add ( component.asComponent(),
          BorderLayout.CENTER );
}
catch (QTEException e) { /* ... */ }
this.addWindowListener(this);
}

```

Auch mit dem aktuellen QTJava muss zunächst eine `QTSession` geöffnet werden, um QuickTime zu initialisieren. Anschließend wird aus der übergebenen URL ein `DataRef`-Objekt erzeugt, mit dem in QuickTime die unterschiedlichsten Datenquellen referenziert werden.

**Achtung:** `DataRef` kommt mit den aus einem `File`-Objekt erzeugten URL-Zeichenketten nicht zurecht. Um Dateien zu referenzieren, müssen Sie die `file://`-URLs für den `DataRef`-Konstruktor von Hand zusammenbauen.

Mit der erhaltenen Daten-Referenz können Sie nun ein `Movie`-Objekt generieren lassen. Dazu übergeben Sie die Referenz sowie ein paar von Apple empfohlene Konstanten an `Movie.fromDataRef()`. Das `Movie`-Objekt alleine bietet zu wenig Funktionalität zum Abspielen, daher verpacken Sie es in einem `MovieController`. Aus diesem Steuerungsobjekt können Sie dann schließlich mit `QTFactory.makeQTComponent()` eine Komponente für die Benutzungsoberfläche erzeugen lassen.

Beachten Sie, dass es sich hier um die Klasse `quicktime.app.view.QTFactory` handelt und nicht um die gleichnamige, aber veraltete Klasse aus dem Paket `quicktime.app`!

Diese Komponente fügen Sie dann in die Oberfläche ein. Da `makeQTComponent()` eine `quicktime.app.view.QTComponent`-Referenz zurückgibt, müssen Sie das Objekt dafür entweder in eine `java.awt.Component` umwandeln (casten) oder – besser – eine passende Referenz mit `asComponent()` erfragen.



Wenn Sie sich die `quicktime.std.movies.MovieController`-Referenz als Objektvariable merken, können Sie das Abspielen außerhalb des Konstruktors beeinflussen. Beispielsweise können Sie der Methode `play()` einen Parameter für die Abspielgeschwindigkeit übergeben – mit `1.0F` wird der Film in normaler Geschwindigkeit angezeigt, mit `0.0F` wird der Film angehalten.

In der `windowClosing()`-Listener-Methode bleibt alles beim Alten: Unmittelbar vor dem Programmende wird die `QTSession` geschlossen:

```
public void windowClosing(WindowEvent e) {
    QTSession.close();
    this.setVisible(false);
    this.dispose();
    System.exit(0);
}
// ...
}
```

Die Programmierung mit QTJava 6.1 erfordert also minimal mehr Aufwand als mit Version 6.0. Dafür funktioniert die Lösung sowohl mit Java 1.3 als auch mit Java 1.4 – und für die flexiblen Steuerungsmöglichkeiten mit dem `MovieController` müssten Sie bei QTJava 6.0 sowieso denselben Aufwand betreiben.

Wenn Sie sich weitere Beispiele zu QTJava ansehen möchten, finden Sie auf <http://developer.apple.com/samplecode/Java/idxQuickTime-date.html> eine Liste mit Archiven zu diversen Themen. Aber Achtung: Die meisten davon laufen nur mit QTJava 6.0, Apple hat sie leider noch nicht an QTJava 6.1 angepasst. Von der Seite <http://developer.apple.com/sdk/qtjavasdk.html> können Sie ein QTJava-Entwicklungspaket (SDK) herunterladen, das alle diese Beispiele und die API-Dokumentation enthält – allerdings nur für Windows. Um die Situation für QTJava-Programmierer zu verbessern, hat sich gerade die Netzgemeinschaft »OpenQTJ« gegründet, die auf der Seite <https://open-qtj.dev.java.net/> aktualisierte Beispiele, Anleitungen und Hilfestellungen bei Problemen anbietet.

QuickTime bietet auch vielfältige Im- und Exportmöglichkeiten. Beispielsweise können Sie folgenden QTJava-Quelltext verwenden, um eine Bilddatei im Apple-eigenen PICT-Format in das JPEG-Format umzuwandeln:

```
import quicktime.io.QTFile;
import quicktime.qd.Pict;
```

```

import quicktime.std.image.GraphicsExporter;
import quicktime.util.QTUtils;
//...
Pict p = Pict.fromFile( new java.io.File("mein.pict") );
GraphicsExporter export =
    new GraphicsExporter( QTUtils.toOSType("JPEG") );
export.setInputPicture( p );
QTFile out = new QTFile( "mein.jpg" );
export.setOutputFile( out );
export.doExport();

```

Wenn Sie das PICT-Bild innerhalb des Java-Programms weiterverarbeiten möchten, können Sie vom QuickTime-`GraphicsImporter` ein Image-Objekt erzeugen lassen:

```

import java.awt.*;
import quicktime.app.view.*;
import quicktime.io.QTFile;
import quicktime.qd.QDRect;
import quicktime.std.image.GraphicsImporter;
// ...
GraphicsImporter import =
    new GraphicsImporter( QTUtils.toOSType("PICT") );
import.setDataFile( new QTFile("mein.pict") );
QDRect rect = import.getNaturalBounds();
GraphicsImporterDrawer drawer =
    new GraphicsImporterDrawer( import );
QTImageProducer producer =
    new QTImageProducer( drawer,
        new Dimension( rect.getWidth(), rect.getHeight() ) );
Image img = Toolkit.getDefaultToolkit().createImage( producer );

```

## 7.6 Sound/Musik

Zum Abspielen von Musik, beispielsweise im MP3-Format, setzen Sie am besten QuickTime for Java ein, wie es im vorangegangenen Abschnitt beschrieben wurde. Ob ein Film oder eine Sounddatei abgespielt wird, ändert an der QTJava-Programmstruktur nichts. Wenn Sie auf MP3-Dateien verzichten, können Sie alternativ auch das Java Media Framework verwenden.

Speziell zur Sound-Verarbeitung hat Sun die Java Sound API (JavaSound) entworfen, siehe <http://java.sun.com/products/java-media/sound/>. Diese portablen

Klassen stehen seit Java 1.3 im Paket `javax.sound` zur Verfügung – beachten Sie aber, dass die Audio-Eingaberoutinen beim Mac OS X-Java erst ab Java 1.4 implementiert sind. Und auch in Java 1.4 kommen für die Eingabe nur bestimmte Werte in Frage (44,1 kHz Sample-Frequenz, PCM-Kodierung, mono oder stereo, 8- oder 16-Bit-Samples). Liegt die Audio-Eingabedatei in einem anderen Format vor, müssen Sie sie vor der Verwendung passend umwandeln.

Was mit JavaSound unter Mac OS X leider überhaupt nicht funktioniert, ist die Ansteuerung von MIDI-Geräten. Sollten Sie dies benötigen, müssen Sie das Mac OS X-spezifische CoreAudio-Framework einsetzen, für das Apple Java-Hüllklassen bereitstellt. Die Klassen sind im Archiv `/System/Library/Java/Extensions/CoreAudio.jar` gespeichert und gehören zum Paket `com.apple.audio` (bzw. zu diversen Unterpaketen davon). Eine Übersicht über CoreAudio finden Sie auf den Seiten <http://developer.apple.com/audio/coreaudio.html> und <http://developer.apple.com/documentation/MusicAudio/Reference/CoreAudio/>, die API-Dokumentation der Java-Klassen liegt auf Ihrer Festplatte im Verzeichnis `/Developer/Documentation/CoreAudio/Java/`. Allerdings ist die Java-Dokumentation mehr als dürftig, und die Klassen wurden längere Zeit nicht weiterentwickelt. Insofern sollten Sie nach Möglichkeit auch hier QTJava einsetzen, das zudem wenigstens zwischen Windows und Mac OS X portabel ist.

## 7.7 Drucken

Die schlechte Nachricht zuerst: Die Java 1.4-Druck-API im Paket `javax.print` ist derzeit unter Mac OS X nicht nutzbar – Apple hat sie einfach noch nicht vollständig implementiert. Die gute Nachricht: Alle früheren Druck-APIs in den Paketen `java.awt` und `java.awt.print` sind voll funktional und können auch mit Java 1.4 genutzt werden. Falls Sie dennoch eine `javax.print`-ähnliche Schnittstelle benötigen, empfiehlt Apple, bis auf weiteres auf »J2PrinterWorks« von <http://www.wildcrest.com/> zurückzugreifen.

Damit Sie einen Eindruck bekommen, wie Mac OS X die vorhandenen Druck-Schnittstellen umsetzt, folgen zwei kurze Beispiele. Zunächst wird die `java.awt`-Druck-API vorgestellt, die mit Java 1.1 eingeführt und mit Java 1.3 erweitert wurde:

```
//CD/examples/ch07/print/Print11/Print11.java
import java.awt.*;
public class Print11 extends Frame {
    public void print() {
        JobAttributes jobAttr = new JobAttributes();
        PageAttributes pageAttr = new PageAttributes();
```

```

jobAttr.setDialog( JobAttributes.DialogType.NATIVE );
pageAttr.setColor( PageAttributes.ColorType.MONOCHROME );

PrintJob job = this.getToolkit().getPrintJob( this,
                                             "Java 1.1/1.3 DruckTest",
                                             jobAttr, pageAttr );

if (job == null) return;

```

Zentrale Klasse ist hier `java.awt.PrintJob`. Sie repräsentiert einen Druckauftrag, den Sie mit `JobAttributes`- und `PageAttributes`-Objekten konfigurieren können. In diesem Beispiel wird mit `DialogType.NATIVE` festgelegt, dass Java den System-Druckdialog anzeigen soll (dieser Aufruf ist eigentlich unnötig, da dies die Standardeinstellung ist). `ColorType.MONOCHROME` bestimmt entsprechend eine schwarz-weiße Ausgabe.

Das `PrintJob`-Objekt legen Sie nicht selbst an, sondern Sie lassen es von `Toolkit.getPrintJob()` erzeugen. Ein `Toolkit`-Objekt stellt die tatsächliche AWT-Implementierung des Systems dar, und die meisten Methoden darin rufen Sie nie direkt auf. Einige wenige nützliche Hilfsroutinen – beispielsweise zur Ermittlung der Bildschirmgröße oder eben zum Erzeugen eines Druckauftrags – benötigt man aber relativ häufig. Das passende `Toolkit`-Objekt erfragen Sie entweder mit `getToolkit()` von einer AWT-Komponente oder aber mit `Toolkit.getDefaultToolkit()`. Hier wird ersteres verwendet, da die Klasse von `Frame` erbt und damit selbst eine Komponente ist – außerdem benötigen Sie die `Frame`-Referenz für `getPrintJob()`, weil Sie den Druckauftrag damit dem `Frame` zuordnen.

Durch den Aufruf von `getPrintJob()` wird auch der Druckdialog angezeigt. Wenn der Benutzer darin auf »Abbrechen« klickt, erhalten Sie als Rückgabe die `null`-Referenz und können Ihrerseits die Druckausgabe beenden.

Der Methodenname `print()` ist übrigens beliebig wählbar und durch keine Schnittstelle fest vorgegeben.

```

Graphics g = job.getGraphics();
Dimension pageSize = job.getPageDimension();
g.drawOval( 0, 0, pageSize.width, pageSize.height );
g.drawString( "Breite " + pageSize.width + " Pixel, Höhe "
              + pageSize.height + " Pixel.",
              pageSize.width/2, pageSize.height/2 );
g.dispose();
job.end();
}

```

```
// ...  
}
```

Hat der Anwender »Drucken« angeklickt, können Sie vom `PrintJob`-Objekt einen `Graphics`-Kontext erfragen und darin ganz normal Grafik und Text ausgeben. Sobald Sie fertig sind, müssen Sie den `Graphics`-Kontext mit `dispose()` freigeben und den Druckauftrag mit `end()` abschließen – sonst wird nichts gedruckt!

Wenn Sie das Programm nun laufen lassen, zeigt das System zuerst einen Dialog zur Konfiguration des Papierformats an (siehe Abbildung 7.8). Ob dieser Dialog wirklich erscheint, hängt davon ab, ob das System aus dem übergebenen `PageAttributes`-Objekt ausreichend Informationen extrahieren kann oder nicht.

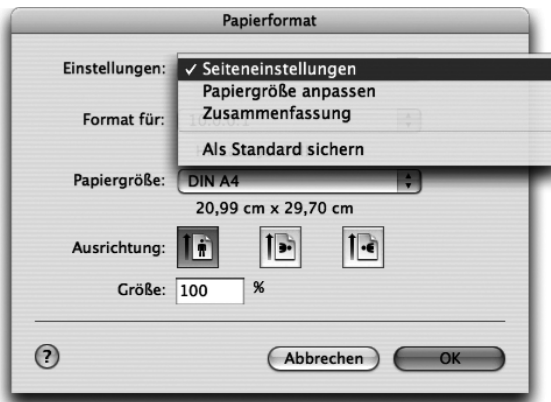


Abbildung 7.8 Konfiguration des Papierformats

Anschließend wird der eigentliche Druckdialog dargestellt, in dem Sie die Mac-typische Vielzahl von Einstellmöglichkeiten vorfinden (siehe Abbildung 7.9). Unten links im Dialog sehen Sie auch die Knöpfe, mit denen die Druckausgabe automatisch in eine PDF-Datei umgelenkt werden kann.

Neben diesem systemspezifischen Druckdialog bietet Java auch einen Dialog, der auf allen Systemen gleich aussieht (siehe Abbildung 7.10). Sie können diese Form auswählen, indem Sie bei `JobAttributes.setDialog()` den Wert `JobAttributes.DialogType.COMMON` übergeben. Überlegen Sie sich aber gut, ob Ihnen der Java-eigene Druckdialog irgendwelche Vorteile bringt, denn der Anwender wird ihn als Fremdkörper in einer ansonsten an das System angepassten Applikation empfinden.



Abbildung 7.9 Konfiguration des Druckauftrags

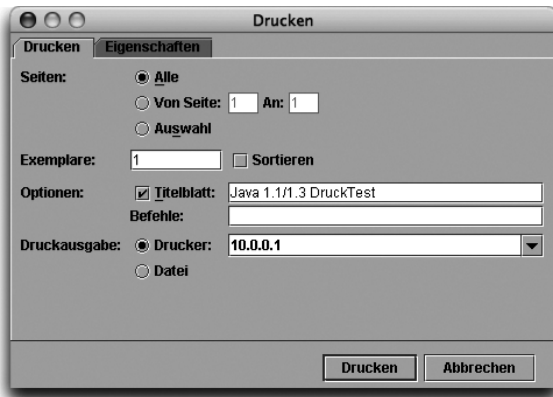


Abbildung 7.10 Javas systemübergreifender Druckdialog

Der große Nachteil der `java.awt`-Druckschnittstelle ist, dass Java 2D von ihr nicht unterstützt wird. Daher wurde mit Java 1.2 eine neue Druck-API im Paket `java.awt.print` eingeführt, die neben der Java 2D-Fähigkeit auch gute Konfigurationsmöglichkeiten für das Papierformat besitzt. Zudem benötigen Sie hierbei keine `Frame`-Referenz wie beim `java.awt.PrintJob`, daher ist die Ansteuerung beispielsweise auch aus einer `main()`-Methode heraus möglich:

```
//CD/examples/ch07/print/Print12/Print12.java
import java.awt.print.*;
// ...
```

```

PrinterJob job = PrinterJob.getPrinterJob();

PageFormat seitenformat = job.defaultPage();
seitenformat.setOrientation( PageFormat.PORTRAIT );

Printable daten = new Druckdaten();
job.setPrintable( daten, seitenformat );

if ( job.printDialog() ) {
    try {
        job.print();
    }
    catch (PrinterException e) {
        e.printStackTrace();
    }
}

```

Diesmal wird kein `java.awt.PrintJob`-Objekt verwendet, sondern ein `java.awt.print.PrinterJob`-Objekt. Dieses erzeugen Sie wiederum nicht direkt, sondern Sie ermitteln ein passendes Objekt mit `getPrinterJob()`. Mit `defaultPage()` erfragen Sie danach ein Standard-`PageFormat`-Objekt für das Papierformat, das dann noch konfiguriert werden kann – hier wird mit `setOrientation()` das Koordinatensystem eingestellt, das seinen Ursprung links oben hat.

Die Besonderheit der `java.awt.print`-API kommt jetzt: Die eigentliche Methode für die Druckausgabe wird in einer Klasse realisiert, die das `Printable`-Interface implementiert. Bei diesem Beispiel ist dies die Klasse `Druckdaten`, die Sie weiter unten aufgelistet finden. Dieses `Printable`-Objekt wird zusammen mit dem `PageFormat`-Objekt mittels `setPrintable()` an den Druckauftrag übergeben.

Nun können Sie die `print()`-Methode des Druckauftrags zum sofortigen Ausdrucken aufrufen. Oder aber Sie rufen vorher noch `printDialog()` auf, damit der Druckdialog angezeigt wird und der Anwender die voreingestellten Werte noch verändern kann. `printDialog()` gibt `false` zurück, wenn der Benutzer »Abbrechen« angeklickt hat.

Die Klasse mit der eigentlichen Ausgabe-Methode ist wie folgt implementiert:

```

//CD/examples/ch07/print/Print12/Druckdaten.java
import java.awt.*;
import java.awt.print.*;
public class Druckdaten implements Printable {

```

```

public int print( Graphics graphics,
                 PageFormat pageFormat, int pageIndex ) {
    if (pageIndex > 0) return NO_SUCH_PAGE;

```

Durch das Printable-Interface muss die Klasse die Methode print() implementieren, der ein Graphics-Kontext, ein Seitenformat-Objekt und die aktuelle Seitenzahl übergeben wird. Java ruft die print()-Methode einfach immer wieder mit aufsteigender Seitenzahl auf – bis das Ende erreicht ist und print() den Wert Printable.NO\_SUCH\_PAGE zurückgibt. Hier kann nur eine einzige Seite mit dem Index 0 gedruckt werden.

```

    Graphics2D g2d = (Graphics2D)graphics;
    g2d.translate( pageFormat.getImageableX(),
                  pageFormat.getImageableY() );
    int breite = (int)pageFormat.getImageableWidth();
    int hoehe = (int)pageFormat.getImageableHeight();

    g2d.drawOval( 0, 0, breite, hoehe );
    g2d.drawString( "Breite " + breite + " Pixel, Höhe " + hoehe
                    + " Pixel.", breite/2, hoehe/2 );
    return PAGE_EXISTS;
}
}

```

Wie bei Java2D üblich können Sie den Graphics-Kontext in einen Graphics2D-Kontext umwandeln. Anschließend wird die Ausgabe mit translate() in den sichtbaren Bereich verschoben, dessen Startpunkt mit getImageableX() und getImageableY() aus dem Seitenformat ermittelt wird. Ebenso werden die Breite und Höhe des Ausgabebereichs abgefragt, wodurch die Seitenränder korrekt beachtet werden. Nach erfolgreicher Ausgabe einer Seite muss print() schließlich noch den Wert Printable.PAGE\_EXISTS zurückgeben.

## 7.8 Literatur & Links

### ► Java 2D

- J. B. Knudsen, »Java 2D Graphics«, O'Reilly 1999  
Dieses Buch ist zwar schon etwas älter, bietet aber eine hervorragende Programmieranleitung für 2D-Grafik mit Java.
- <http://java.sun.com/products/java-media/2D/>  
Auf dieser Seite listet Sun alle relevanten Dokumentationen zu Java 2D und verwandten APIs auf.



### ► Java 3D

- D. Selman, »Java 3D Programming«, Manning Publications 2002  
Dieses Buch deckt viele weiterführende Informationen zu Java 3D ab, die von der Standarddokumentation nicht oder nur unzureichend behandelt werden.

### ► OpenGL

- Shreiner/Woo/Neider, »OpenGL Programming Guide«, 4th ed., Addison-Wesley 2003  
Auch wenn in diesem Buch C und nicht Java als Programmiersprache eingesetzt wird, bietet es einen guten Überblick über die wesentlichen Ideen der OpenGL-Programmierung. Und Java-APIs wie JOGL sind eh recht nahe an den C-Routinen.
- <http://www.opengl.org/>  
Die OpenGL-Homepage enthält unter anderem eine Programmierführung und das Referenzhandbuch.

### ► QuickTime

- Vogt/Kastenholz, »QuickTime 6«, Galileo Press 2002  
Dieses Buch ist keine Programmieranleitung für QTJava, sondern eine ausführliche Anleitung für die Anwendung der QuickTime-Technologie – aber das Verständnis dafür ist unerlässlich, wenn Sie QuickTime ernsthaft programmieren wollen.

# Index

\$APP\_PACKAGE 197, 213, 214  
\$JAVAROOT 197, 212, 214  
.app 166  
.hotspotrc 566  
.login 65, 414  
.MacOSX 65  
.mov 366  
.pbHelpIndexerList 85  
.pbproj 79  
.pkg 166  
.webloc 596  
.xcode 79  
/etc/hostconfig 434, 439  
/usr/bin/cvs 96  
/usr/lib 320  
/usr/lib/java 317  
/usr/local/ 448  
/usr/local/bin 403  
/usr/local/mysql/ 433  
/usr/local/pgsql/ 438  
\n 294  
\r 294  
\_blank 273  
127.0.0.1 449  
3D-grafik 356  
68k 270, 531

## A

AAPL 199  
Abhängigkeiten von Targets 90, 314  
Ablage-Menü 28  
Abstand zum Fensterrand 129  
abstract 524  
Abstract Windowing Toolkit 119, 121  
abstrakte Klasse 524  
Abstürze 30  
ACTION 579  
Action 334, 336, 337  
action 456  
ACTION\_COPY 169  
ACTION\_LINK 169  
ACTION\_MOVE 169  
ActionEvent 126, 153  
ActionListener 126  
actionPerformed() 126

ADB 301  
ADC 46  
    Reference Library 43  
add() 126  
addActionListener() 126  
addApplicationListener() 155  
addDocumentListener() 159  
addMouseListener() 135  
addSeparator() 128, 149  
addWindowListener() 123  
Administrator-Rechte 67, 348  
Aktivitäts-Anzeige 32  
Aktualisierung 44, → s. Update  
Alias 31, 589, → s. Link,  
    → s. Verknüpfung  
Alpha-Kanal 189  
Alpha-Wert 134  
ALT\_MASK 150  
Alt-Taste 16  
always 166  
anonyme Klasse 126, 527  
anonymes Paket 511  
Ant 62, 77, 78, 385, 465, 468, 474  
    ear 475  
    Eclipse 394  
    Filter 395  
    jar 475  
    JAR-Archiv 391  
    jarbundler 396  
    Manifest 391  
    metainf 475  
    Programmpaket erzeugen 394  
    Property 389  
    Umgebungsvariablen 389  
    war 475  
    Xcode 388  
    Xcode 1.5 393  
    XDoclet 477  
ANT\_HOME 386  
AntiAliasedGraphicsOn 558  
AntiAliasedTextOn 558  
Antialiasing 556, 557, 589  
Ant-Task 387  
Anweisung 514  
Apache 242, 257, 445, 448, 503

- Ant 385
  - Maven 396
- Apfel-Menü 27
- Apfel-Taste 16
- API 589
- API-Dokumentation 43, 62, 512
- app 187
- appBundlesTraversable 166
- AppKit 324, 336
- APPL???? 192
- Apple 70
  - Applet Runner 59, 535
  - Application Server 453
  - Application Servers-Paket 465
  - Developer Connection 46
  - Gestaltungsrichtlinien 171
  - Human Interface Guidelines 171
  - J2EE 479
  - OSXAdapter 164
  - WebObjects 464
- apple.awt.antialiasing 557
- apple.awt.brushMetalLook 168, 552
- apple.awt.fakefullscreen 555
- apple.awt.fileDialogForDirectories 138, 552
- apple.awt.fractionalmetrics 557
- apple.awt.fullscreencapturealldisplays 555
- apple.awt.fullscreenhidecursor 555
- apple.awt.fullscreenusefade 555
- apple.awt.graphics.EnableLazyDrawing 558
- apple.awt.graphics.EnableLazyDrawingQueueSize 558
- apple.awt.graphics.EnableLazyPixelConversion 563
- apple.awt.graphics.OptimizeShapes 558
- apple.awt.graphics.RenderDrawArc 562
- apple.awt.graphics.RenderDrawOval 562
- apple.awt.graphics.RenderDrawPolygon 562
- apple.awt.graphics.RenderDrawRect 562
- apple.awt.graphics.RenderDrawRoundRect 562
- apple.awt.graphics.RenderDrawShape 562
- apple.awt.graphics.RenderFillArc 562
- apple.awt.graphics.RenderFillOval 562
- apple.awt.graphics.RenderFillPolygon 562
- apple.awt.graphics.RenderFillRect 562
- apple.awt.graphics.RenderFillRoundRect 562
- apple.awt.graphics.RenderFillShape 562
- apple.awt.graphics.RenderGlyphs 563
- apple.awt.graphics.RenderImage 562
- apple.awt.graphics.RenderLine 562
- apple.awt.graphics.RenderString 563
- apple.awt.graphics.RenderUnicode 563
- apple.awt.graphics.UseTwoImageLazyPixelConversion 563
- apple.awt.interpolation 557
- apple.awt.rendering 557
- apple.awt.showGrowBox 168, 551
- apple.awt.textantialiasing 557
- apple.awt.window.position.forceSafeCreation 556
- apple.awt.window.position.forceSafeProgrammaticPositioning 556
- apple.awt.window.position.forceSafeUserPositioning 556
- apple.cmm.usecolorsync 563
- apple.laf.AquaLookAndFeel 144
- apple.laf.useScreenMenuBar 162, 181, 551
- AppleEvent-Deskriptor 341
- AppleJavaExtensions 163, 329
- AppleScript 340, 350
  - Cocoa Java 340
  - osascript 342
- APPLET 254
- Applet 59, 60, 64, 241, 250, 536
  - Cache 260
  - Lebenszyklus 253
  - Sicherheit 262
- Applet Launcher 61, 257
- Applet Runner 535
- AppleTalk 302
- AppletContext 271, 281

- appletviewer 61, 257, 264, 273
- Application 154, 203, 289
- Application Kit 336
- Application Support 68
- application.xml 473, 475
- ApplicationEvent 155
- ApplicationListener 154, 205
- Applications 31, 225
- Applications (Mac OS 9) 533
- Applikation 187
  - aktive 27
- Applikations-Server 453, 463, 464
- Aqua 31, 56, 144, 589
  - Gestaltungsrichtlinien und Layout-Manager 166
  - Größe von Komponenten 166
  - Hintergrund 134
  - Human Interface Guidelines 332
- Aqua-Look & Feel 142, 144
- Arbeitsverzeichnis 213
- ARCHIVE 254, 255, 261
- Archivnamen 288
- ArgoUML 417
- Argumentliste, variable 545
- Arguments 213, 216, 550
- Array 545
- ASCII 589
- ASP 445
- assert 570, 580
- assertTrue 399
- Assoziation 522
- Attribute 327
- attributorientiertes Programmieren 477
- Audio 366, 374
- Audiodaten 362
- Auflösungsunabhängigkeit 120
- Aufrufhierarchie 88
- Aufzählungen 545
- Ausdruck 515
- ausführen 87
- Ausführungsbit → s. Execution-Bit
- Ausnahmefehler 528
- Aussehen 142, 144
  - im Programmcode setzen 145
- Auswerfen 221
- Authentisierung 463
- Autoboxing 544

- Auto-Commit 431
- automatische Variable 515
- Autorisation 348, 463
- Autorisation Toolkit 348
- Autounboxing 545
- AVI 362
- awakeFromNib() 337
- AWT 119, 121, 251, 291, 589
  - Anpassung an andere Systeme 128
  - Brushed Metal 167
  - Cocoa 339
  - Xcode 122

## **B**

- backgroundOnly 171, 553
- Back-in-Time-Debugging 106
- Backslash 590
- bash 39
- Basisklasse 520
- batch 576
- Batch-Datei 177
- BBEdit 36, 177
- Bea WebLogic 464
- Bean 463
- Bean Managed Persistence 468
- Bearbeiten-Menü 28
- beenden 176
- Beenden-Menüpunkt 132
- Befehl-Taste 16
- Benutzer 31, 432
  - Passwort ändern 433
- Benutzer-Prozesse 65
- Benutzerverwaltung 26, 347, 433
- Benutzer-Verzeichnis 37, → s. Home-Verzeichnis
- Benutzungsoberfläche 25, 119, 551
- Berechtigung 263
- Berkeley DB 441
- Betriebssystem 48, 297
- Bezeichner 510
- bg 435
- Bibliothek 51, 66, 182, 318
  - Abhängigkeiten 319
  - Installationspfad 321
  - Suchreihenfolge 68
- bicubic 557
- Big Endian 548
- BigDecimal 516

- BigInteger 516
- Bildbearbeitung 188, 353
- Bildlaufleiste 159
- bilinear 557
- bin 30, 188
- Bitmap 589
- Block 509
- BlueJ 113
- Bluetooth 497
  - MIDP 500
- Bluetooth Assistent 498
- Bluetooth Datenaustausch 498
- BMP 354, 468, 589
- Boolean 161
- bootclasspath 490, 568
- Booten 31
- Boot-Manager 31, 531
- bootstrap 543
- BorderLayout 120, 125, 274
- Borland
  - JBuilder 107
  - Together Control Center 108
  - Together Solo 108
- Bourne-Shell 177
- Boxing 544
- BoxLayout 120
- BranchGroup 356
- Breakpoint 88, 102
- BrowseListener 303
- Browser 60, 64, 234, 250, 255, 265, 299, 324, 325
  - AWT-Komponente 339
  - Java 1.4 266
- Browser → s. Web-Browser
- Browser Launcher 300
- Browser Opener 300
- BrowserControl 301
- Brushed Metal 167
- brushMetalLook 552
- BSD 56
- Buch-CD 595
- BuchUpdate 47
- Buch-Webseite 596
- BufferedImage 563
- Bug Parade 292
- Bug Reporter 548
- Build 86, 212, 391
  - Einstellungen 91
  - Phasen 92, 313, 491
  - Protokoll 86, 494
  - Prozess 385, 577
  - Settings 91
  - System 90
  - transcript 86
  - Variablen 92
- Build Settings 577
- Build Transcript 494
- build.xml 386, 474
- BUILD\_DIR 578
- buildDMG 222
- Buildfile 386, 474
- BUILT\_PRODUCTS\_DIR 578
- Bundle 187, 219, 319, 394, 582, 589
  - im Dateiauswahldialog 138
- Bundle Resources 92
- Bundle-Bit 207
- Business Logic 470
- Button 120, 125
- Bytecode 22, 38, 403, 404, 414, 512, 589
  - Viewer 406

## C

- C, C++ 24, 309, 519, 521
- Cache 238, 260
- CAD 359
- call 276
- CallStaticVoidMethod() 323
- Camino 259, 268
- Canvas 506
- Canvas3D 356
- Carbon 57, 589
  - Locking 326
- Carbon File Locking 573
- CarbonLock 326
- Carriage Return 294, 548
- CASE 108
- Cast 516, 589
- CATALINA\_HOME 449
- catch 528
- cc 318
- CD 595
- ceil() 516
- CENTER 125, 129
- Certificate Authority 247
- CFBundleAllowMixedLocalizations 210

CFBundleDevelopmentRegion 210  
 CFBundleDisplayName 210, 219  
 CFBundleDocumentTypes 216  
 CFBundleExecutable 210  
 CFBundleGetInfoHTML 210, 219  
 CFBundleGetInfoString 210, 219  
 CFBundleIconFile 211  
 CFBundleIdentifier 211  
 CFBundleInfoDictionaryVersion 211  
 CFBundleName 130, 211, 219, 552  
 CFBundlePackageType 192, 211  
 CFBundleShortVersionString 211, 219  
 CFBundleSignature 192, 211  
 CFBundleTypeExtensions 217  
 CFBundleTypeIconFile 217  
 CFBundleTypeMIMETypes 217  
 CFBundleTypeName 217  
 CFBundleTypeOSTypes 217  
 CFBundleTypeRole 217  
 CFBundleVersion 212  
 CGI 445, 589  
 check, jni 569  
 Checkbox 120, 125  
 CheckboxGroup 125  
 checkSpelling() 346  
 chmod 178, 207  
 chown 438  
 CHUD-Tools 409  
 CTime 576  
 Class 140  
 Class.forName() 140, 288, 289, 291,  
 323, 349  
 CLASS\_ARCHIVE 581  
 CLASS\_ARCHIVE\_SUFFIX 581  
 CLASS\_FILE\_DIR 491, 581  
 CLASS\_FILE\_LIST 582  
 ClassCastException 153  
 class-Datei 39, 509  
 Classes 63, 475  
 classes.jar 288, 405, 413  
 Classic 324, 589  
 Speicherprobleme 537  
 Umgebung 24, 533  
 ClassLoader 244, 289  
 ClassNotFoundException 289  
 CLASSPATH 51, 66, 68, 176, 180, 212,  
 216, 413, 550, 567, → s. cp  
 CLDC 483  
 Clean 390  
 Client-VM 566  
 close() 428, 429, 519  
 cmd 298  
 CMM 563  
 CMP 468, 589  
 Cocoa 57, 330, 349, 589  
 Action 334, 337  
 API 336, 339  
 Applikation Kit 336  
 AWT 339  
 Foundation 336  
 id 334  
 Java und Objective-C 339  
 NSObject 333  
 Outlet 334  
 Swing 339  
 Cocoa Java 300, 330  
 AppleScript 340  
 Bridge 339  
 Referenzen 335  
 Speech 344  
 Spelling 345  
 Xcode 331  
 Cocoa Java, Bridge 330  
 CocoaComponent 339  
 CocoaMySQL 437  
 Cocoon 299  
 CODE 255  
 Code Completion 83  
 Code Conventions 530  
 Code Sense 83  
 CODEBASE 239, 255, 261  
 Codec 354  
 CodeGuide 106  
 Code-Optimierung 405  
 Codepage 295  
 CodeWarrior 110  
 Collections API 22  
 Color 134, 523  
 ColorSync 563  
 com.apple.audio 375  
 com.apple.backgroundOnly 171, 553  
 com.apple.cocoa.  
 application.NSMenu 555  
 com.apple.dnssd 303  
 com.apple.eawt 146, 154  
 com.apple.eio 329

- com.apple.forcehwaccel 561
- com.apple.hwaccel 560
- com.apple.hwaccelextclude 560
- com.apple.hwaccelist 560
- com.apple.hwaccelnogrowbox 561
- com.apple.macos.smallTabs 553
- com.apple.macos.use-file-dialog-  
packages 137, 553
- com.apple.macos.useMDEF 553
- com.apple.macos.useScreenMenuBar  
162, 553
- com.apple.macosx.AntiAliased-  
GraphicsOn 558
- com.apple.macosx.AntiAliased-  
TextOn 558
- com.apple.mrj 121, 122, 328  
auf anderen Systemen 139
- com.apple.mrj.appli-  
cation.apple.menu.about.name  
130, 181, 552
- com.apple.mrj.application.classpath  
216, 550
- com.apple.mrj.appli-  
cation.growbox.intrudes 552
- com.apple.mrj.appli-  
cation.JVMVersion 216, 551
- com.apple.mrj.application.live-resize  
553
- com.apple.mrj.application.main 216,  
550
- com.apple.mrj.application.para-  
meters 216, 550
- com.apple.mrj.appli-  
cation.vm.options 216, 551
- com.apple.mrj.application.  
workingdirectory 216, 550
- com.apple.mrj.MRJAboutHandler 552
- com.apple.mrj.swing.MacLoo-  
kAndFeel 144
- com.apple.usedebugkeys 561
- com.muchsoft.util.Mac 139, 163
- com.sun.j3d.utils 356
- com.sun.jimi.core 355
- com.sun.media.jai.codec 354
- Combo Box 166
- Command 503
- command 178
- CommandListener 503
- Commands 63
- Command-Taste 16
- commit() 430
- Communicator 267
- Compiler 38, 412, 512, 589
- CompileThreshold 576
- Component 120
- Concurrent Versions System 94
- Connection 335, 428, 429, 430
- Connection-URL 428
- Connector/J 437
- Console 41
- Constructor 141
- Container 120
- Container Managed Persistence 468
- Content Pane 159
- Contents 190, 582
- context-root 473
- Controller 467
- Copy Files 93  
Programm 313
- Copyright 198, 211  
Zeichen 129
- CORBA 479, 589
- CoreAudio 375
- CoreFoundation 316
- CoreGraphics 559
- cp 180, 491
- Cp1252 296
- CR 294
- Crash-Log 41
- create() 470, 472
- createCompatibleImage() 355
- createdb 438
- CreateException 469
- createImage() 374
- createStatement() 428, 430
- Creator-Code 191, 199, 211, 296, 327
- Cross-Platform 305
- CTRL\_MASK 150
- Ctrl-Taste 16
- Current 64
- CurrentJDK 63
- currentTimeMillis() 301
- Custom Build Command 389
- CustomView 336
- CVS 94, 593  
aktivieren 96

- checkout 95
- commit 95
- Compare 97
- Diff 97
- import 95
- init 94
  - Projekt importieren 95
- CVSROOT 94
- cvs wrappers 94
- Cyberduck 244
  
- D**
- da 571
- Darstellungshinweis 556
- Darwin 56, 70, 318, 590
- Data Fork 188
- Database 423
- DataFlavor 169
- DataRef 371
  - file-URLs 372
- DateFormat 446
- Datei
  - Attribute 327
  - Information 198, 210
  - Metadaten 327
  - umbenennen 179
- Dateiauswahldialog 552, 553
  - AWT 136
  - Bundles / Pakete 138, 166
  - Swing 165
  - Verzeichnisse auswählen 138
- Dateiname 297, 537
  - Länge 294
  - Sonderzeichen 294
- Dateinamenerweiterung 328, 593
- Dateisystem 29, 69
- Dateitrennzeichen 293
- Dateiverwaltung 347
- Datenbank 423
  - Treiber laden 427
  - Verbindung abbauen 430
  - Verbindung aufbauen 428
- Datenbankentwurf 426
- Datenbank-System 424
- Datenbank-Verwaltungs-System 423
- Datenbasis 423
- Datensatz 424
- Datentyp 515
  
- DB 423, 590
- DBMS 424, 590
- dclj 530
- de.comp.lang.java 530
- debug 569
- Debugger 88, 102
- Debugging 88
  - Back-in-Time 106
  - Grafikausgabe 562
- Debug-Information 409
- DecimalFormat 514
- Decompiler 403, 404
- Decoration 134
- DefaultClasspath 68
- Deinstallieren 187
- Deklaration 515
- Deployment 476
- DEPLOYMENT\_LOCATION 578
- Deployment-Deskriptor 458, 471, 473, 477
- deprecated 328, 370
- Desktop 29, 590
- destroy 253
- DestroyJavaVM() 324
- Destruktor 519
- Developer Tools 42, 76
- DEVELOPMENT\_PLIST\_FILE 582
- Dialog 120
  - Elemente 28
  - zentrieren 129
- Dictionary 209
  - Java 212, 237
- Dienst 450
- Dimension 129
- disableassertions 571
- DisableExplicitGC 575
- disablesystemassertions 572
- Disassembler 406
- Disk Copy 220
- Disk Image 220, 231
- Display 490
- display-name 471, 473
- displayURL() 300
- dispose() 124, 519
- DLL 311
- DMG 220, 590
  - Hintergrundbild 222
  - Nur lesen 223



- Segmente 42
- DMG Maker 222
- DMG Tool 222
- dmgpart 42
- DnD 169, 590
- DnDConstants 169
- DNS 303, 590
- dns\_sd.jar 303
- DNSSD 303
- DNSSDService 303
- do..while 517
- Dock 26, 535, 590
- dock
  - icon 572
  - name 572
  - Symbol 180
  - Symbol anklicken 156
- Document Object Model 279
- DocumentListener 159
- doGet() 446, 472
- Dokumentation 43, 46, 410, 512
- Dokumentsymbol 217
- Dokumenttyp 199, 204, 209, 216
- Dokumenttypen 131
- DOM 279
- Dongle 404
- doPost() 456
- Doppelpunkt 69
- DOS 293, 295
- Drag & Drop 29, 51, 169, 220, 590
- DragSourceDragEvent 170
- Drawable 367
- Drei-Schicht-Architektur 463
- Driver/Manager 428, 430
- DrJava 115
- Druckdialog 299, 377
- Drucken 375
- dsa 572
- DSTROOT 579
- DTD 214
- Duke 535
- dylid 320
- DYLD\_FALLBACK\_FRAMEWORK\_PATH 320
- DYLD\_FALLBACK\_LIBRARY\_PATH 320
- DYLD\_LIBRARY\_PATH 320

- Dylib 318, 319, 326, 590
  - Suchpfade 320
- Dynamic Shared Library 318
- dynamischer Methodenaufruf 524
- dynamisches Laden von Klassen 289

## E

- ea 571
- EAR 385, 474
- ear 475, 581
- Early Access 541
- EAST 125
- Eclipse 98, 205
  - Ant 394
  - Debugger 102
  - Eingabevollständigung 100
  - Fehler 100
  - JUnit 402
  - kompilieren 100
  - SDK 98
  - Versionen 2.1 und 3.0 99
  - XML-Plugin 417
- Eden-Generation 566, 574
- Editor 204, 217
- einfacher Datentyp 515
- Eingabeaufforderung 38
- Eingabevollständigung 83, 100
- Einstellungen...-Menüpunkt 52, 131, 155, 164
- eio 329
- EJB 78, 463, 467, 479, 590
  - Container 463, 468, 471
  - Deployment-Deskriptor 471
  - Komponente 463, 477
- ejb 473
- ejbActivate() 470
- ejb-class 471
- ejbCreate() 470
- EJBHome 470
- ejb-jar.xml 471, 473, 475
- ejb-name 471, 473
- EJBObject 469
- ejbPassivate() 470
- ejbRemove() 470
- Element 254
- else 516
- emacs 37

- E-Mail 295
- EMBED 255
- Embedding Framework 325
- emulator 494
- enableassertions 571
- EnableLazyDrawing 558
- EnableLazyDrawingQueueSize 558
- EnableLazyPixelConversion 563
- enablesystemassertions 572
- Ende-Tag 209, 254
- Endorsed Directory 568
- English.lproj 192, 218
- Enterprise JavaBeans 463, 467
- Enterprise-Applikation 78, 468
- Enterprise-Applikations-Archiv 474
- Enterprise-Applikations-Deployment-Deskriptor 473
- Entity-Bean 467
- Entwicklerpaket 591
- Entwicklertools 42
- Entwicklungsumgebung 75
- Entwurfsmuster 591
- Enumeration 152, 545
- environment.plist 65
- Environment-Variable 94
- erben 520
- Ereignis-Behandlung 120
- Ereignis-Verarbeitungs-Thread 157, 274
- Ergebnistabelle 426, 429
- Erweiterungs-Bibliotheken, Suchreihenfolge 68
- Erweiterungsverzeichnisse 66
- esa 572
- Escape-Sequenz 514, 590
- etc 30, 53
- Ethernet-Hardware-Adresse 310
- Euro-Symbol 590
- eval 277
- Event Handling 120
- EventObject 164
- EventQueue 291
- examples 595
- Exception 528
- ExceptionDescribe() 324
- ExceptionOccurred() 324
- exec() 297, 342
- execPrivileged() 349

- Executable 80
  - ausführen 87
- executeQuery() 428
- executeUpdate() 430
- Execution-Bit 178
- Expert View 92, 577
- Explorer 27
- Exposé 30, 590
- Expression 515
- ext 67
- extends 122, 520, 526
- Extension 593
- Extensions 66, 131, 401
- extern C 315
- External Target 90, 389
- Extreme Programming 398

## **F**

- fail 400
- fakefullscreen 555
- Fallunterscheidung 516
- FALSE 161
- FAQ 46
- Farbverwaltung 563
- Feld 424, 429
- Fenster 120, 535
  - Änderungssymbol 160
  - Hintergrund 134
  - unsichtbar 155
  - zentrieren 129
- Fenster-Menü 29, 148
- Fensterrand 129
- Festplatte 31
- Festplatten-Dienstprogramm 32, 220
- Festschreiben 431
- File 293, 297
  - pathSeparator 294
  - pathSeparatorChar 294
  - separator 293, 294
  - separatorChar 294
- File Transfer Protocol 244
- file.encoding 296, 549
- file.separator 294, 549
- FileDialog 136, 137, 165, 552, 553
- FileManager 296, 300, 329, 330
- FileMerge 97
- FilenameFilter 138

- FileStorm 231
- FileType 199
- Filme 366
- Filter 138
- final 522
- finally 528
- FindClass() 323
- Finder 27, 590
- findFirstMisspelledWord() 347
- findFolder() 330
- findMisspelledWordInString() 347
- Firebird 439
- Firefox 259, 269
- Fix and Continue 89
- floor() 516
- FlowLayout 120, 125, 253
- for 517, 544
- for each 544
- forcehwaccel 561
- forceSafeCreation 556
- forceSafeProgrammaticPositioning 556
- forceSafeUserPositioning 556
- FORM 456
- Form 488
- formale Parameter 513
- forName() 140, 289
- Foundation (Cocoa) 336
- Foundation Classes 119
- fractionalmetrics 557
- Frame 120, 122
  - Rahmen 134
- Frame.ICONIFIED 128, 153
- Frame.NORMAL 154
- Framework 57, 62, 316, 590
- Frameworks & Libraries 92
- FreeBSD 56, 302
- freie Software 590
- Fremdschlüssel 425
- FTP 244
- Full-Screen Exclusive Mode 554
- fullscreencapturealldisplays 555
- fullscreenhidecursor 555
- fullscreenusefade 555
- FullScreenWindow 556
- fullversion 568
- Funktion 513
- future 568

## G

- G3, G4, G5 548
- Garbage Collection 326, 519, 566, 574
- gcc 312, 318, 412
- gcc3 318
- GDB 89
- gebürstetes Metall 167
- GeekTool 42
- Generics 24, 544
- GenericServlet 446
- Generische Typen 544
- German.lproj 218
- Geschäftslogik 467, 470
- Geschwindigkeit 13
- Gestaltungsrichtlinien 171
- GET 456
- getAppletContext 273
- getBundle() 157
- getBytes() 296
- getConnection() 428, 430
- getConstructor() 141, 290
- getContentPane 274
- getContentPane() 159
- getCrossPlatformLookAndFeelClassName() 145
- getDefault() 157
- getDefaultToolkit() 149, 374, 376
- getDirectory() 137
- getDouble() 428
- getExtendedState() 153
- getFile() 137
- getFileCreator() 328
- GetFileInfo 199, 327
- getFileType() 328
- getGraphics() 376
- getInsets() 129
- getKeyStroke() 149
- getMember 277
- getMenuComponentCount() 152
- getMenuShortcutKeyMask() 149
- getMethod() 289
- getMouseLocationOnScreen() 289
- getParameter 255
- getParameter() 456
- getPrinterJob() 379
- getPrintJob() 376
- getProperty 547
- getRequestURI() 456

getRootPane() 134, 160  
getScreenSize() 129  
getSize() 129  
getSource() 153  
GetStaticMethodID() 323  
getString() 148, 428  
getSystemLookAndFeelClassName()  
    145  
Getter 519  
getTitle() 153  
getToolkit() 129, 376  
getUserAction() 170  
getWindow 276  
GIF 189, 354, 492  
GL4Java 359  
GLCanvas 359  
glconfigurations.properties 560  
globaler Code 510  
GNU 590  
    GPL 300  
gnumake 310  
Goal 397  
Gosling, James 11  
GPL 300, 590  
Grafikausgabe 353  
Grafik-Hardware-Beschleunigung 196,  
    558  
Grafikkarte 559  
Grafik-Primitiv 557  
grafische Benutzeroberfläche 119  
grant 263  
Graphic User Interface → s. GUI  
GraphicConverter 188, 327  
Graphics 253, 353, 376, 380  
    dispose() 376  
Graphics2D 253, 353, 380  
GraphicsEnvironment 355  
GraphicsExporter 374  
GraphicsImporter 374  
GraphicsImporterDrawer 374  
GridLayout 125  
Groovy 414  
groovy 415  
Groovy, Compiler 415  
GROOVY\_HOME 414  
groovyc 415  
groovyConsole 415  
groovysh 414

Growbox 551, 552  
Grundlagen 509  
Gruppenverwaltung 347  
GUI 25, 48, 119, 340, 551, 590

## H

Häkchen vor einem Menüpunkt 153  
Handheld 483, 500  
handleAbout() 130, 140, 155, 164  
handleOpenApplication() 131, 140,  
    156, 164  
handleOpenFile() 131, 132, 140, 156,  
    164, 205, 218  
handlePreferences() 156  
handlePrefs() 130, 140, 164  
handlePrintFile() 140, 156, 164  
handleQuit() 132, 133, 140, 156, 164  
handleReOpenApplication() 156, 164  
Handy 483, 497  
Hardware-Beschleunigung 196, 558  
Hardware-Kopierschutz 404  
Hardware-Zugriff 309  
Hauptklasse 512  
Hauptversion 195, 259  
hdiutil 222  
Header 312, 315  
Headers (Verzeichnis) 325  
Headless AWT 553  
HeadlessException 170  
Headless-Modus 170  
Heap 573  
HFS+ 69, 297, 590  
Hilfe 29, 33, 42  
Hintergrund 134  
Hintergrundbild 222  
HiRes Timer 301  
Hit Mask 189  
Hochauflösende Zeitmessung 301  
Home 37, 53, 63, 543  
home 471  
Home-Interface 470  
HORIZONTAL\_SCROLLBAR\_ALWAYS  
    159  
Host 590  
hostconfig 434, 439  
Hot Deployment 476  
HotSpot 22, 566  
Hotsync Manager 500

- hqx 188
- HSQLDB 440
- HTML 234, 241, 250, 253, 265, 274, 278,
  - 445, 453, 590
  - anzeigen 299
  - Entity 296
  - in Swing-Komponenten 161
  - Validator 271
- HTML Converter 256
- HTTP 446, 456, 591
- httpd 242
- HTTP-Proxy 259
- HttpServlet 446, 472
- HttpServletRequest 446
- HttpServletResponse 446
- HTTPS-Proxy 259
- Hüllklasse 276, 311, 330, 343
- Hüllobjekt 544
- Human Interface Guidelines 31, 48,
  - 171, 332
- Humble Narrator 344
- hwaccel 560
- hwaccel\_info\_tool 560
- hwaccelextclude 560
- hwaccelextclude.properties 561
- hwacclist 560
- hwaccelnogrowbox 561
- Hypersonic SQL 440

**I**

- IB 331, 591
- IBAction 336
- IBOutlet 336
- iCab 270, 281
- ICC\_ColorSpace 563
- icns Browser 189
- icns-Datei 180, 188
- Icon Composer 188
- Icon-Datei 28, 188, 191, 207, 217
- ICONIFIED 128, 153
- id 334
- ID → s. Schlüssel
- IDE 75, 591
- IDEA 105
- Identifier 200, 211
- Identifikation 348
- if 516
- IllegalStateException 132
- Im Dock ablegen 128
- Image 374
- Image I/O 353
- Immortal-Generation 576
- implements 122, 525
- import 288, 511
- import static 545
- incgc 575
- Index 83
- Index Templates 84
- Info.plist 130, 171, 191, 207, 208, 214,
  - 216, 237, 339, 395, 550, 565, 582
- INFO\_PLIST\_FILE 582
- InfoPlist.strings 192, 219
- Information Property List 208
- Inhalts-Ebene 159
- init 252
- initdb 438
- InitialContext 473
- Inkrement 517
- innere Klasse 124, 527
- Input Files 93
- InputEvent 149
- InputStreamReader 296
- INSERT INTO 430
- Insets 129
- INSTALL\_DIR 579
- INSTALL\_GROUP 579
- INSTALL\_MODE\_FLAG 579
- install\_name\_tool 321
- INSTALL\_OWNER 579
- INSTALL\_PATH 579
- install\_templates 84
- install4j 229
- InstallAnywhere 227
- Installationspaket 166, 223, 592
- Installationsprogramm 192, 224
- InstallerMaker 232
- InstallShield 228
- int 575
- Integrated Development Environment
  - s. IDE
- Intellij IDEA 105
- InterBase 439
- Interface 123, 290, 525
- interface 525
- Interface Builder 331, 340
  - Action 336

- Connection 335
- Klassen ableiten 333
- Objekte erzeugen 335
- Outlet 335
- Quelltext generieren 336
- Swing 340
- Target/Action 336
- internalversion 569
- Internet Explorer 267
- Internet-Protokolle 295
- interpolation 557
- Interpreter 22
- invoke() 289
- invokeAndWait() 274, 291
- invokeLater() 133, 274, 291
- Invoker-Servlet 452
- iODBC 441
- IOKit 316
- IP 590, 591
- IPv4 445
- IPv6 445
- isPopupTrigger() 135
- iText 299
- IzPack 233

## J

- J/Direct 324
- J2EE 78, 447, 463, 591
  - Apple 479
  - Applikations-Server 463
  - SDK 463
  - Server 463
- J2ME 483, 591
  - Konfiguration 483
  - MIDlets 485
  - MIDP 484
  - Profil 484
  - System-Properties 488
  - Tipps 508
  - Wireless Toolkit 485
- J2SE 447, 591
  - Version 5.0 24, 541
- j2se 240
- jad 403
- JAD-Datei 493, 502
- Jaguar 23, 591
- JAI 24, 353
  - Image I/O Tools 354
- jai\_imageio.jar 354
- Jakarta Struts 459
- Jakarta Tomcat 448
- Jam 90
- JApplet 272
- jar 182, 475, 567
  - Option 185
- Jar Bundler 62, 193, 214
- Jar Bundler Ant Task 396
- JAR Caching 260
- Jar Launcher 185
- JAR-Archiv 51, 67, 182, 197, 207, 241, 244, 288, 385, 491
  - in Xcode 186
  - per Doppelklick starten 184
  - statisch einbinden 92
- jarsigner 247
- Java 57
  - 2D 353, 378, 380
  - 3D 24, 356, 381
  - Advanced Imaging 24, 353
  - API-Dokumentation 410, 512
  - Application Stub 191, 582
  - Archive Settings 92
  - Bytecode 22, 38
  - Code Conventions 530
  - Collection Framework 544
  - Communications API 301
  - Compiler 38, 412, 512
  - Compiler Settings 91
  - Data Objects 423, 591
  - Debugger 89
  - Developer Package 312
  - Developer Tools 43, 76
  - Development Kit 23
  - Einführung 530
  - Embedding API 536
  - Embedding Framework 64, 258, 325
  - Embedding Plugin 259
  - Enterprise Edition 463
  - Erweiterungen 66
  - Erweiterungsverzeichnis 401
  - Foundation Classes 119
  - Grundlagen 509
  - Home 65
  - Image I/O 354
  - Image Management Interface 354
  - Kompatibilität 292

- Konsole 258
- Language Specification 412
- Laufzeitparameter 259
- Laufzeitumgebung 321, 513
- Look & Feel 142, 146
- Media Framework 362, 374
- Micro Edition 483
- Native Interface 309
- Network Launching Protocol 234
- Plugin 60, 255, 258, 266, 277, 325, 536
- Plugin Einstellungen 259
- Preferences API 54
- Programmierrichtlinien 510, 530
- Programmiersprache 22
- Quelltext 411
- QuickTime 366
- Reflection 139, 163
- Resources 92
- Runtime Environment 23, 176
- Runtime-Version 586
- Scripting 280
- Server Faces 104
- Servlets 445, 446
- Shared Archive 566, 576
- Sound 374
- Speech Framework 343
- Spelling Framework 345
- Studio Creator 104
- User Group 70
- Version 63, 194, 292, 551, 585, 586
- Version 1.1 91
- Version 1.5 24, 541
- Version der Apple-Implementierung 586
- Versionsproblem 407
- Virtual Machine 22, 64, 321, 536, 565
- VM 22, 64
- VM-Framework 63
- Web Start 60, 112, 234
- java 39, 63, 176, 263, 513, 543, 565
- Java 1.3.1 Plugin Einstellungen 259
- Java 1.4.2 Plugin Einstellungen 259
- Java 2 Plattform 22
- Java Applet Plugin Enabler 268
- Java Applet.plugin 266
- Java Web Start 236
- java.applet 251
- java.awt 121, 375
- java.awt.color.ICC\_ColorSpace 563
- java.awt.datatransfer 169
- java.awt.dnd 169
- java.awt.event 124
- java.awt.FileDialog 552
- java.awt.headless 171
- java.awt.image.BufferedImage 563
- java.awt.print 375, 378
- java.awt.RenderingHints 557
- java.class.path 288
- java.endorsed.dirs 568
- java.ext.dirs 68
- java.home 66
- java.io.File 297
- java.io.FileNameFilter 138
- java.io.Serializable 169
- java.lang.reflect 141
- java.library.path 294, 318
- java.math 516
- java.net.preferIPv4Stack 445
- java.policy 263
- java.rmi 469
- java.runtime.version 548, 586
- java.sql 426
- java.text 514
- java.vendor 548
- java.vendor.url 548
- java.vendor.url.bug 548
- java.version 547, 585
- java.vm.vendor 548
- java.vm.version 547
- JAVA\_APP\_STUB 582
- JAVA\_ARCHIVE\_CLASSES 581
- JAVA\_ARCHIVE\_COMPRESSION 581
- JAVA\_ARCHIVE\_TYPE 581
- JAVA\_CLASS\_SEARCH\_PATHS 581
- JAVA\_COMPILER 580
- JAVA\_COMPILER\_DEBUGGING\_SYMBOLS 580
- JAVA\_COMPILER\_DEPRECATED\_WARNINGS 580
- JAVA\_COMPILER\_DISABLE\_WARNINGS 580
- JAVA\_COMPILER\_FLAGS 580
- JAVA\_COMPILER\_SOURCE\_VERSION 580

JAVA\_COMPILER\_TARGET\_VM\_VERSION 580  
 JAVA\_FORCE\_FILE\_LIST 582  
 JAVA\_HOME 65, 414, 543  
 JAVA\_LAUNCHER\_VERBOSE 207  
 JAVA\_MANIFEST\_FILE 581  
 JAVA\_SOURCE\_FILE\_ENCODING 580  
 JAVA\_SOURCE\_PATH 582  
 JAVA\_SOURCE\_SUBDIR 582  
 JAVA\_USE\_DEPENDENCIES 582  
 Java13Adapter 140  
 Java13Handler 139, 140  
 Java14Adapter 163  
 Java14Handler 163  
 Java2D 556  
 JavaApplet.h 325  
 JavaApplicationStub 206, 210, 395  
 JavaBrowser 62, 410  
 javac 38, 63, 370, 512, 543, 580  
 JAVAC\_DEFAULT\_FLAGS 580  
 JavaClasses 581  
 JavaComm 301  
 javaconfig 68  
 JavaConfig.plist 68  
 JavaConnect 281  
 JavaControl.h 325  
 Javadoc 410, 477  
 JavaEmbedding.h 325  
 javah 310, 312  
 JavaMonitorsInStackTrace 569  
 javap 406  
 JavaPluginCocoa.bundle 266  
 JavaScript 242, 256, 274  
     globales Objekt 276  
     window 276  
 JavaScriptCore 266  
 JavaScript-URL 281  
 JavaServer Pages 445, 453  
 JavaSound 374  
     Audio-Eingabe 375  
 JavaSpeechFramework.jar 343  
 JavaSpellingFramework.jar 345  
 JavaVM 322  
 JavaVM.framework 312  
 JavaVMInitArgs 322  
 JavaVMOption 322  
 javaw.exe 186  
 javax.ejb 469  
 javax.imageio 354  
 javax.jnlp 248  
 javax.media 362  
 javax.media.j3d 356  
 javax.media.jai 353  
 javax.microedition 485  
 javax.naming 472  
 javax.print 375  
 javax.rmi 472  
 javax.servlet 446  
 javax.servlet.http 446  
 javax.sound 375  
 javax.sql 426  
 javax.swing 147, 251, 272  
 javax.swing.plaf.metal.  
     MetalLookAndFeel 144  
 JAWS 234, 591  
     Desktop-Integration 236  
     Programm-Manager 235  
     Programmname 239  
     Sicherheit 245  
 JayBird 440  
 JBindery 537, 539  
 JBoss 78, 464  
     beenden 467  
     server/default/deploy 476  
     starten 465  
     Workbook 478  
 jboss-j2ee.jar 474, 475  
 JBuilder 107  
     Versionen 6,7, X und 2005 107  
     Versionen 8 und 9 108  
 JCavaj 403  
 JCheckBoxMenuItem 153  
 jclasslib 407  
 JComboBox 167  
 JComponent 120  
 JDBC 423, 426, 428, 591  
     Adresse 428  
     Treiber 427, 437, 439  
     URL 428  
 JDBC-ODBC-Bridge 441  
 JdbcOdbcDriver 441  
 JDesktopPane 143  
 JDeveloper 108  
 JDialog 120  
     Menüzeile 163



- JDirect 324, 325, 349, 536
  - Version 2 und 3 326
- JDirect\_MacOSX 325
- JDK 23, 57, 58, 531, 591
- JDKClasses.zip 91
- JDO 423, 591
- jEdit 111
- Jelly 396
- Jetty 445, 465
- Jext 112
- JFC 119, 537
- JFileChooser 165
- JFileChooser.appBundlesTraversable 166
- JFileChooser.packagelsTraversable 166
- JFrame 120, 147
  - Rahmen 134
- JGoodies 171
- JIO 354
- Jikes 412, 580, 581
  - Xcode 413
- JKES\_DEFAULT\_FLAGS 581
- JKESPATH 413
- JIMI 354
- JInternalFrame 143
- JIT-Compiler 22, 575
- JJEdit 113
- JLabel 161
- JManager 324, 325, 536
- JmDNS 305
- JMenu 147, 148
- JMenuBar 148
- JMenuItem 147, 149, 153
- JMF 362
- jmf.jar 362
- JMFRegistry 362
- JMX 465
- JNDI 591
- JNI 309, 339, 349, 536, 569
  - Abhängigkeiten von Bibliotheken 319
  - Bibliothek 67
  - Bundle 319
  - Mac OS X 10.0 319
  - Pfad für Bibliotheken 317
  - Suchpfade 317
  - Xcode 310
- JNI, Bibliothek 246
- jni.h 312, 316, 322
- JNI\_CreateJavaVM() 322
- JNI\_GetDefaultJavaVMInitArgs() 322
- JNI\_VERSION\_1\_2 322
- JNI\_VERSION\_1\_4 322
- JNIDirect 326
- JNIEnv 316, 322
- jnilib 311
- JNLP 234
  - Developer's Pack 248
- jnlp.jar 248
- JNLP-API 248
- JNLP-Datei 238
- JobAttributes 375
- JOGL 359
- jogl.jar 359
- JOptionPane 120, 133
- JPanel 120
- JPEG 354
- JPEG2000 354
- JProfiler 408
- JRE 23, 58, 176, 249, 260, 591
- JRendezvous 305
- JRootPane 134
- JRun 464
- JSA 566, 576
- JScrollPane 159
- JSEException 276
- JSubject 275
  - call 276
  - eval 277
  - getMember 277
  - getWindow 276
- JSP 445, 453
  - Ausdruck 454
  - Direktive 454
  - Element-Bibliothek 455
  - Expression 454
  - Scriptlet 454
  - Standard Tag Library 455
  - Tag Library 455
- JSP-Container 445, 454
- JSR 541
- JSTL 455
- JTA 591
- JTabbedPane 166
- JTextArea 159
- JTS 591

JTxtCmpontDrvr 345  
JUG 70  
JUnit 398  
    Eclipse 402  
    Xcode 401  
Just-in-time-Compiler 22  
JVM 22, 38, 64, 321, 536  
    Optionen 565  
    Version 194  
JVMVersion 212, 216, 551  
JWS 60

## K

Kapselung 519  
Karteireiter 166, 553  
KDE 266  
Kennung 559  
Kernel 591  
KEY\_ALIASING 557  
KEY\_FRACTIONALMETRICS 557  
KEY\_INTERPOLATION 557  
KEY\_RENDERING 557  
KEY\_TEXT\_ALIASING 557  
Key4J 404  
KeyEvent 127, 149  
Keystore 246  
KeyStroke 149  
keytool 246  
KHTML 266  
Kilobyte Virtual Machine 484  
KJS 266  
Klasse 275, 509  
    abstrakte 524  
    anonyme 126, 527  
    dynamisch laden 289  
    innere 124, 527  
    lokale 527  
    Typ 289  
Klassenbibliothek 22  
Klassendatei 39, 512  
Klassendatei → s. class-Datei  
Klassendiagramm 416  
Klassenhierarchie 62  
Klassenlader 139, 289  
Klassenmethode 513  
Klassenname 510  
    voll qualifiziert 176

Klassenpfad 51, 66, 68, 92, 176, 197,  
    212, 244, 288, 387, 401, 413, 567,  
    → s. Classpath  
KodakCMM 563  
Kodierung 209, 549  
Kommandozeile 38, 176  
Kommandozeilenparameter 514  
Kommentar 510  
Kompatibilität 292  
Komponente 120  
Komponentenmodell 463  
Konfiguration 483  
Konfigurationsdatei 52  
Konsole 41, 87, 185, 258, → s. Console  
Konstante 522  
Konstruktor 519  
Konstruktorverkettung 521  
Kontakt 17  
Kontext 474  
Kontext-Klick 136  
Kontext-Popup-Menü 80, 135  
Kontrollfelder 532  
Kontrollkästchen 120  
kooperativ 537  
Kopierschutz 404  
kTemporaryFolderType 330  
KVM 484, 490

## L

L&F 142  
Label 129  
LAF 142, 591  
Landeskennung 157  
Länge von Dateinamen 294  
Laufwerk 29  
    virtuelles 220  
Laufzeitfehler 88  
Laufzeitparameter 259  
Laufzeitumgebung 22, 321, 513  
Laufzeitverhalten 408  
LaunchServices 202  
Layout-Manager 120, 253, 274  
    Positionsangabe 126  
ld 318  
learnWord() 347  
Legacy-Target 90  
LF 294  
LGPL 590

- lib 67, 311
- LIBRARY\_STYLE 319
- line.separator 294, 548
- LineBreak 177
- Linefeed 294, 548
- Link 31, 591
- LINKED\_CLASS\_ARCHIVES 581
- Linker 318, 325
- Linkliste 14
- Linux 23, 24, 55, 58, 302
- Listener 121, 124
- LiteSwitch 28
- LiveConnect 274
  - Verfügbarkeit 281
- live-resize 553
- ll 41
- LOAD 137
- loadLibrary() 310
- LOCAL\_ADMIN\_APPS\_DIR 583
- LOCAL\_APPS\_DIR 584
- LOCAL\_DEVELOPER\_DIR 584
- LOCAL\_LIBRARY\_DIR 584
- Locale 157
- localhost 239, 449
- log4j 62, 465
- loggc 575
- lokale Klasse 527
- lokale Variable 515
- Lokalisierung 149, 157, 192, 210, 218, 591
- Look & Feel 56, 144
  - im Programmcode setzen 145
- lookup() 472
- LSBackgroundOnly 171
- LShasLocalizedDisplayName 219
- lsMac 199

**M**

- Mac 591
- Mac OS
  - Extended Format 69
  - Java 59
  - Updates 538
  - Versionen 8 und 9 23, 49, 531
- Mac OS X 23
  - Application Bundle 187
  - Applikation 187
  - Emulation 13
  - Java-Struktur 60
  - Systemarchitektur 55
  - Version 10.1 35
  - Versionen 586
- Mac OS X Server 24, 464
  - /etc/hostconfig und MYSQL 435
  - Dokumentation 479
  - MySQL 432
  - MySQL-Startuptem 434
- MAC-Adresse 310, 591
- Mach-O 318
- MACINTOSH 580
- Macintosh Look & Feel 142
- Macintosh Runtime for Java 23
- MacLookAndFeel 144
- MacRoman 296, 549
- MacSOUP 46
- MagicDraw 417
- Mailingliste, java-dev 46
- main() 514
  - Argument-Array 132
- main.m 331, 337
- Main-Class 184, 238
- MainClass 212, 216, 550
- MainMenu.nib 331
- make 310, 385, 412
- Makefile 310
- makeKeyAndOrderFront() 337
- man pages 42
- Manifest 183, 238, 391, 492
- MANIFEST.MF 183
- markMisspelledWords() 346
- Maske 188
- Math 516
- Maustasten 30
- Maven 396
- MAVEN\_HOME 397
- MaxFDLimit 569
- MaxHeapFreeRatio 574
- MaxNewSize 574
- MaxPermSize 575
- MAYSCRIPT 279
- MBean 465
- MDI 29, 143, 591
- ME4SE 496
- Media Access Control 310
- Mehrfachvererbung 521
- Mehrprozessorsystem 575

- Menu 127
- MenuBar 127
- MenuItem 127
- Menüpunkt mit Häkchen 153
- MenuShortcut 127
- Menüzeile 27, 148, 181, 535, 551, 553
  - AWT 127, 134
  - JDialog 163
  - ohne offene Fenster 148
  - Swing 143, 162
- Merge 92
- Meta Package 227
- META\_MASK 150
- Metadaten 327, 477
- META-INF 183, 475
- metainf 475
- Metal 142, 146
- MetalLookAndFeel 144
- Meta-Programmierung 290
- METHOD 456
- Method 289
- Methode 513
  - veraltet 328
- Metrowerks CodeWarrior 110
- MicroEmulator 496
- Microsoft 267
- Microsoft Office 11
- MIDI 375
- MIDlets 485
  - beenden 489
  - Formular anzeigen 489
  - implementieren 488
  - JAR-Archiv 491
  - Zustand 489
- MIDlet-Suite 492
- MIDP 484, 591
  - API 486
  - Applikation 484
  - Bluetooth 500
  - Emulator 486, 494, 507
  - Ereignisbehandlung 507
  - Farbe 506
  - Game Action 507
  - Grafik 506
  - Referenz-Implementation 485, 496
  - RMS 493
  - Unicode 504
- MIME-Typ 204, 217, 243, 447, 502
- MinHeapFreeRatio 574
- Minimal.app 596
- MisFox 200
- MisspelledWord 347
- mixed 576
- mn 573
- Mnemonics 165
- modaler Dialog 120
- Model, View, Controller 121
- module 473
- more 184
- MouseAdapter 135
- MouseEvent 135
- MouseListener 136
- mousePressed() 135
- mouseReleased() 135
- Movie 362, 366, 371
- MovieController 372
  - play() 373
- Mozilla 46, 259, 268, 281
- MP3 374
- MPEG 362
- mpkg 227
- MRJ 23, 49, 59, 91, 121, 328, 531, 586, 591
  - FAQ 536
  - SDK 539
- MRJ Software Development Kit 324
- mrj.version 49, 548, 586
- MRJAboutHandler 122, 130, 552
- MRJAdapter 131, 147
  - openURL() 300
- MRJApp.properties 192, 215, 550
- MRJAppBuilder 193
- MRJApplicationUtils 123
- MRJClasses 59, 91
- MRJFileUtils 300, 328, 330
- MRJOpenApplicationHandler 122, 131
- MRJOpenDocumentHandler 123, 131, 205
- MRJOSType 328
- MRJPluginCarbon 281
- MRJPrefsHandler 123, 131
- MRJPrintDocumentHandler 123
- MRJQuitHandler 123, 132
- MRJToolkit 23, 536
- MRJToolkit.jar 328
- MRJToolkitStubs 139, 329

ms 573  
Multimedia 365  
Multiple Document Interface 143  
Multitasking 537  
Musik 366, 374  
mv 179  
MVC 121, 333, 591  
mx 573  
MySQL 432, 459  
    beenden 436  
    grafische Oberflächen 437  
    JDBC-Treiber 437  
    starten 435  
mysql 436  
MySQL Manager 432  
mysqladmin 435  
MYSQLCOM 434  
mysqld\_safe 435  
mysqldump 437  
mysqlimport 437

## **N**

NAME 279  
Namensdienst 463  
nanoTime() 302  
NanoTimer 302  
narrow() 472  
native Routinen 309, 325  
Navigator 267, 274  
nearestneighbor 557  
net.java.games.jogl 359  
NetBeans 103, 206  
    Launcher 104  
NetInfo 433, 591  
NetInfo Manager 432, 438  
Netscape 267, 274, 281  
netscape.jar 277  
netscape.javascript 275  
Netzwerkkarte 310  
Netzwerkkommunikation 295  
Netzwerkkonfiguration 302  
Netzwerkverbindung 240  
never 166  
newInstance() 141, 290, 291  
Newline 548  
NewObjectArray() 323  
NewRatio 574

newrt.jar 543  
Newsgroup 530  
NewSize 574  
NewSizeThreadIncrease 576  
Newsreader 46  
NeXT 57, 269  
next() 428  
NeXTSTEP 331, 592  
NeXTSTEP-Konfigurationsdatei 450  
NFS 297  
nib4j 340  
NIB-Datei 331, 337, 338, 592  
niutil 433  
nm 315  
noclassgc 575  
NORMAL 154  
Normalisierung 426  
NORTH 125  
NPAPI 325  
NSAppleEventDescriptor 341  
NSAppleScript 340  
NSApplicationMain() 337  
NSJavaNeeded 339  
NSJavaPath 339  
NSJavaRoot 339  
NSMenu 555  
NSMutableDictionary 341  
NSObject 333  
NSSpeechRecognizer 344  
NSSpeechSynthesizer 344  
NSSpellChecker 345  
NSWorkspace 300  
NumberFormat 514

## **O**

Oberflächendesign 142  
Oberklasse 520  
Obfuscator 404, 508  
OBJECT 255  
Objective-C 57, 330, 333, 589  
    id 334  
Objekt 518  
Objektdatei 318  
objektorientierte Datenbank 423  
Objektvariable 519  
OBJROOT 579  
ODBC 441, 592

- ODBC Administrator 441
- ODBMS 592
- Office 11
- offline-allowed 240
- Öffnen mit 179
- OmniCore CodeGuide 106
- OmniWeb 269
- OO 592
- OOA 592
- OOD 592
- OODB 423
- OOP 592
- open 297, 298, 300
- Open Database Connectivity 441
- Open Source 56, 592
- OpenGL 56, 358, 381, 558
- OpenQTJ 373
- OpenTalk 302
- openURL() 300
- Opera 270
- operationFailed() 304
- Optimierung 405
- OptimizeShapes 558
- Option-Taste 16
- Oracle 440, 442, 539
  - JDeveloper 108
- Oracle 10g 440
- Ordner 30, 592
- Organizer 483, 500
- os.arch 548
- os.name 48, 548, 585
- os.version 548
- OSA 342
- osalang 342
- osascript
  - AppleScript 342
  - Sprachausgabe 344
- OSXAdapter 164
- osxutils 199
- OTHER\_JAVA\_CLASS\_PATH 581
- otool 321
- Outlet 333, 335, 338
  - Java 337
- OutOfMemoryError 538
- Output Files 93
- OutputStreamWriter 296
- <oXygen> 417

**P**

- pack() 123
- Package 67, 254, 510, 592
- PACKAGE\_TYPE 582
- packageIsTraversable 166
- PackageMaker 192, 223
- PageAttributes 375
- PageFormat 379, 380
- paint 253
- Paket 67, 510, 592
  - anonymes 511
  - Installation 223
  - Java 187, 254
  - Programm 187, 190
- Paket → s. Package
- Paket-Bit 187
- Paketinhalt zeigen 190
- Paketname 176
- Palm Desktop 500
- PalmOS 500
- Panel 120, 125
- Panther 24, 592
- Papierformat 377
- parallele Schnittstelle 301
- PARAM 254, 261
- Parameter 550
  - formale 513
- Partition 31, 592
- passwd 433
- PATH 69, 386, 414
- path.separator 294, 548
- PB 76, 592
- pbdevelopment.plist 582
- pbhelpindexer 85
- pbxbuild 87, 577
- PDA 483, 500
- PDF 56, 190, 377
  - anzeigen 298
  - erzeugen 299
  - Plugin 299
  - sichern als 299
- PearPC 13
- Perf 301
- Perforce 94
- Performance Pack 362
- Performanz 408
- Permanente Generation 575
- Permission 263

Persistenz 463, 467, 592  
 Personal Web Sharing 242, 257, 304  
 Petstore 62  
 Pfadangaben 35, 51  
     im Buch 596  
 Pfadtrennzeichen 294  
 pg\_ctl 438  
 Photoshop 188  
 PHP 445  
 pico 37  
 PICT 354, 373, 374  
 Pixie 189  
 pkg 224  
 PkgInfo 191, 207, 211, 582  
 PKGINFO\_FILE 582  
 PLAF 142, 592  
 PLAIN\_DIALOG 134  
 Plattformunabhängigkeit 287, 309  
 Player 363  
 plist 54, 208, 214, 450  
 Pluggable Look & Feel 142  
 Plugin 59, 60, 65, 255, 258, 277, 325,  
     536  
     deaktivieren 266  
     Einstellungen 259  
     Java Embedding 259  
 PNG 189, 354, 492  
 Policy 263  
 policytool 264  
 Polymorphismus 523  
 POP3 295  
 Pop-up-Menü 135  
 Popup-Trigger 136  
 Port, 8080 449  
 Portabilität 287, 309  
 PortableRemoteObject 472  
 Poseidon 417  
 POSIX 347, 592  
 PosixSuites 347  
 POST 456  
 POSTGRES 439  
 PostgreSQL 438  
     beenden 439  
     JDBC-Treiber 439  
     starten 438  
     StartupItem 439  
 Post-Inkrement 517  
 PowerBook 11  
 PowerPC 548  
 PPC 23, 531, 548  
 Pramati Server 464  
 PRC Converter 500  
 PRC-Datei 500  
 Prebinding 319  
 Preferences 52  
 Prefix.h 331  
 preverify 490, 491, 494  
 Primärschlüssel 425  
 Primary Key 425  
 primitiver Datentyp 515  
 print() 295, 379, 380, 514  
 Printable 379  
 PrintCompilation 576  
 printenv 69  
 PrinterJob 379  
 printf 546  
 PrintJavaStackAtFatalState 569  
 PrintJob 376  
     end() 376  
 println() 295, 514  
 PrintSharedSpaces 566, 576  
 printStackTrace() 472  
 PrintStream 295  
 PrintTenuringDistribution 575  
 PrintWriter 295, 446  
 private 514  
 Process 343  
 Processing 348  
 processMouseEvent() 136  
 PRODUCT\_NAME 389, 392, 579  
 prof 569  
 Profil 484  
 Profiler 408  
 Programm 509  
     deinstallieren 187  
     löschen 187  
 Programmabbruch 529  
 Programmcode 191  
 Programme 31  
     Ordner 220, 225  
 Programmiereditor 111  
 Programmierrichtlinien 510, 530  
 Programm-Manager 235  
 Programm-Menü 27, 130, 180, 211

- Programmname 180, 239, 552
- Programmpaket 166, 187, 190, 236, 317, 338, 394, 582, 592, 596
  - im Dateiauswahl-Dialog 138
  - von Hand erzeugen 206
- Programmsymbol 180, 188, 211, 236
- ProGuard 405
- Project Builder 24, 42, 62, 76, 90, 203, 577
- Project Rave 104
- project.xml 397
- PROJECT\_NAME 578
- Projekt 90, 387
  - Gruppe 80
  - Index 83
  - Struktur 79
  - Template 77
  - Typ 77
- Projektplanung 416
- Projektverwaltung 75
- Prolog 209
- Propeller-Taste 16
- Properties 214, 293
- properties, Umlaute und Sonderzeichen 158
- properties-Datei 158
- Property 547
- Property List 208, 219, 592
- Property List Editor 214
- protected 513
- Protokoll 525
- Prototyp 275
- Proxy
  - HTTP 259
  - HTTPS 259
- Proxy-Einstellung 250
- Prozedur 513
- prozedurale Programmierung 513
- Prozess 33, 533, 537
- Prozessverwaltung 347, 535
- ps 33, 533
- psmp-Datei 225
- psql 438
- pthread 324
- public 513
- put() 166
- putClientProperty() 160, 166
- PWD 579

- pwd 53
- Python 112
- Q**
- QDRect 374
- QT 365
- QTCanvas 367
- QTException 368
- QTFactory 368, 372
- QTFile 374
- QTImageProducer 374
- QTJ 366
- QTJava 366, 374, 556
  - Versionen 366
  - Xcode 369
- QTJava.zip 369
- QTSession 367, 369, 371, 373
- QTUtils 374
- quality 557
- Quaqua-Look&Feel 172
- Quartz 56
- Quartz Extreme 196, 559
- Quelltext 411
- Quelltext-Datei 510
- Quelltextkodierung 83
- Quelltextverwaltung 94
- Query 424
- QuickTime 56, 362, 365, 381
  - Im- und Export 373
  - Versionen 366
- QuickTime for Java 366
- QuickTime Player 366
- quicktime.app.display 366, 367
- quicktime.app.display.FullScreen-Window 556
- quicktime.app.view 366, 371
- quicktime.std 371

- R**
- Radio-Button 125
- Rational Rose 417
- RCEEnvironment 94
- RDBMS 592
- Receipts 227
- Rechnerstart 31, 531, → s. Booten
- Rechtschreibprüfung 36, 345
- Rechtsklick 80, 135
- ReduceSignalUsage 570



Referenzdatentyp 516  
 Reflection 139, 163, 289, 290  
 registerAboutHandler() 123, 130, 552  
 registerJava13Handler() 140  
 registerJava14Handler() 163  
 registerOpenApplicationHandle 123  
 registerOpenDocumentHandle 123  
 registerPrefsHandle 123  
 registerPrintDocumentHandle 123  
 registerQuitHandle 123  
 reguläre Ausdrücke 22  
 Relation 425  
 relationale Datenbank 423  
 remote 471  
 Remote Method Invocation 469  
 RemoteException 469  
 Remote-Interface 469  
 RenderDrawArc 562  
 RenderDrawOval 562  
 RenderDrawPolygon 562  
 RenderDrawRect 562  
 RenderDrawRoundRect 562  
 RenderDrawShape 562  
 RenderFillArc 562  
 RenderFillOval 562  
 RenderFillPolygon 562  
 RenderFillRect 562  
 RenderFillRoundRect 562  
 RenderFillShape 562  
 RenderGlyphs 563  
 RenderImage 562  
 Rendering Hint 556  
 RenderLine 562  
 RenderString 563  
 RenderUnicode 563  
 Rendezvous 302, 592  
 Repository 94  
 ReservedCodeCacheSize 570  
 reservierte Schlüsselwörter 509  
 Resource Fork 188  
 ResourceBundle 148, 157, 192  
 Ressourcen, Zugriff 244  
 RestartService() 450  
 ResultSet 428, 429  
 return 514  
 Reverse Engineering 108  
 Richtlinientool 264  
 RIGHT 125  
 RMI 469  
 Role 204, 217  
 rollback() 430  
 ROOT 452, 455, 457  
 root 30, 579, 592  
 Root Pane 134, 161  
 ROOT-Kontext 452, 457  
 Round Trip Engineering 108, 417  
 round() 516  
 rs 570  
 RS232-Schnittstelle 301  
 rt.jar 288, 405, 543  
 Rückgabety 513  
 Ruhezustand 533  
 Run 87  
 run() 130, 527  
 Rundung 516  
 runhprof 569  
 Runnable 133, 527  
 Runtime.exec() 297, 298, 299, 342  
 RXTX 301

## S

Safari 266, 281  
 safe\_mysql 435  
 Sandbox 235, 245, 262  
 SAVE 137  
 Schaltfläche 120  
   Reihenfolge beim Mac 126  
 Schleife 517  
 Schlüssel 425  
 Schlüsselwort 509  
 Schnittstelle 123, 525  
 Schrägstrich 69  
 Schreibtisch 590  
 SCM 94, 592  
 Scriptlet 454  
 Scrollbar 159  
 SDK 463, 593  
 Search Paths 91  
 Security Policy 263  
 Security-Manager 246  
 Segmente 42  
 Selbst-Referenz 520  
 SELECT FROM 425, 428  
 Separator 293  
 Serializable 169  
 serielle Schnittstelle 301

- Server 24, 234, 242
- Server → s. Web-Server
- Server-Administration 464
- Server-Anwendungen 170
- Server-VM 566
- Service 450
- serviceFound() 303
- serviceLost() 304
- servlet.jar 475
- Servlet-Container 445, 448, 465
- ServletException 446
- Servlet-Kontext 457
- Servlet-Mapping 458
- Servlets 445, 446, 467, 472
  - Xcode 447
- Session 454, 467
- SessionBean 78, 467, 470
- ServletContext 470
- session-type 471
- setAccelerator() 149, 165
- setAutoCommit() 430
- setBackground() 134, 155
- setBounds() 155
- setContentType() 446
- setDirectory() 137
- setEnabled() 151
- setEnabledPreferencesMenu() 155
- setEnabledPrefs() 163
- setenv 65
- setExtendedState() 128, 154
- SetFile 199, 207, 223, 328
- setFile() 138
- setFilenameFilter() 138
- setFullScreenWindow() 554
- setHandled() 156, 164
- setHelpMenu() 127
- setJMenuBar() 149
- setLayout() 125
- setLocation() 123, 129
- setLookAndFeel() 145
- setMenuBar() 127
- setMenuBarVisible 555
- setMnemonic() 165
- setMode() 138
- setProperty() 549
- setReadOnly() 428, 430
- setRenderingHint() 556
- setResizable() 123
- setSessionContext() 470
- setState() 128, 153
- Setter 519
- setUndecorated() 134, 155, 554
- setUp() 400
- setVisible() 124, 133
- setWindowDecorationStyle() 134
- sh 177
- Sharing 242
- SharingMenu 243
- Shark 409
- Sheets 38
- Shell 38, 40, 51, 64, 93, 176, 536, 593
- Shell Script Files 93
- Shell Script Target 90
- SHIFT\_MASK 149
- Shift-Taste 16
- Shortcut 16, 28
- show() 134, 135
- showConfirmDialog() 133
- showDocument() 248, 272, 281
- showGrowBox 551
- showStatus() 271
- Sicherheit 463
- Sicherheitslücke 11
- Sicherheitsrichtlinie 263
- SIGHUP 570
- SIGINT 570
- Signale 570
- Signatur 211, 323, 525
- Signature 199
- Signieren 246, 262
- SIGQUIT 570
- SIGTERM 570
- SIGUSR1 570
- SIGUSR2 570
- SimpleDateFormat 514
- SimpleUniverse 356
- Sites 242
- Sitzung 467
- SKIP\_INSTALL 578
- Skript 176, 593
- Skripteditor 340
- Skriptsprache 340, 413
- smallTabs 553
- Smart Group 80
  - anlegen 82
- SMB 297

- Soft Button 503
- Software-Aktualisierung 44, 249
- Softwareprozess 416
- Solaris 23, 24, 55, 302, 311
- Sonderzeichen 38
- Sound 374
- Source Control Management 94
- sourcepath 582
- Sources 92
- SOUTH 125
- Speech Framework 343
- speed 557
- Speicherauslastung 33
- Speicherschutz 535
- Speicherverbrauch 408
- Speicherverwaltung 519
- Spelling Framework 345
- SpellingChecker 347
- Spiele 359
- Splash-Screen 240
- Sprachausgabe 343
- SQL 424, 593
  - Abfrage 425
- SQL/J 423, 593
- SQLException 429
- src.jar 411
- SRCROOT 579
- ss 573
- Stack 573
- Stack Trace 41, 88, 569
- staff 579
- Standard-Klassenpfad 68
- Standardverzeichnisse 583
- Stapelverarbeitungs-Datei 177
- start() 130, 253
- Startpunkt 514
- startRealtimeChecking() 346
- StartService() 449
- Start-Tag 209, 254
- StartupItem 434, 439, 449
  - MySQL 434
  - PostgreSQL 439
- StartupParameters.plist 450
- Startvolume 31, 532, 593
- Stateful Session-Bean 468
- Stateless Session-Bean 468
- Statement 424, 428, 429
- static 513
- statische Imports 545
- statischer Initialisierungsblock 311
- Steuerungsobjekt 121, 467
- Steuerung-Taste 16
- stop() 253
- StopService() 450
- String, getBytes() 296
- stringFlavor 169
- StringTokenizer 288
- Struts 459
- Stub 191
- Stub-Klasse 139, 163, 329
- su 438
- Subklasse 520
- Subroutine 513
- Subversion 94, 593
- Suchreihenfolge von Erweiterungs-  
Bibliotheken 68
- sudo 592
- Suffix 593
- suggestGuessesForWord() 347
- Suite 347, 400
- Suite/P Toolkit 347
- Sun 22, 70
  - Bug Parade 292
- Sun Java Studio Creator 104
- sun.boot.class.path 288
- sun.cpu.endian 548
- sun.jdbc.odbc.JdbcOdbcDriver 441
- sun.misc.Perf 301
- super 521
- super() 521
- Superklasse 520
- Superuser 592, 593
- Superuser-Rechte 348
- SurvivorRatio 574
- Swing 22, 56, 119, 142, 251, 272, 291, 537, 593
  - Anpassungen für andere Systeme 150
  - Brushed Metal 168
  - Cocoa 339
  - Performanz 291
  - Threads 274, 291
  - Xcode 146
- swing.defaultlaf 144
- swing.properties 144
- SwingSet2 142

- SwingUtilities 133, 145, 274
  - invokeAndWait() 291
  - invokeLater() 291
- switch 517, 545
- Switcher 12, 371
- Sybase ASE 441
- symbolischer Link 589
- SYMROOT 578
- Synthesizer 344
- Sys 50, 140, 163, 585, 596
  - getHomeDirectory 53
  - getJavaHome 66
  - getLocalPrefsDirectory 53
  - getPrefsDirectory 53
  - getPrefsDirectory() 293
  - getWorkingDirectory 53
  - isAMac 50
  - isLinux 50
  - isMacOS 50
  - isMacOSX 50
  - isOS2 50
  - isWindows 50
- System
  - currentTimeMillis() 301
  - Kodierung 296
  - nanoTime() 302
  - Programmierumgebungen 57
  - Property 48, 52, 214, 241, 246, 488, 547, 567
- System.exit 246, 262
- System.exit() 132
- System.gc() 575
- System.getProperty() 293
- System.loadLibrary() 311
- SYSTEM\_ADMIN\_APPS\_DIR 583
- SYSTEM\_APPS\_DIR 583
- SYSTEM\_CORE\_SERVICES\_DIR 583
- SYSTEM\_DEMOS\_DIR 583
- SYSTEM\_DEVELOPER\_APPS\_DIR 583
- SYSTEM\_DEVELOPER\_DEMOS\_DIR 583
- SYSTEM\_DEVELOPER\_DIR 583
- SYSTEM\_DEVELOPER\_DOC\_DIR 583
- SYSTEM\_DEVELOPER\_GRAPHICS\_TOOLS\_DIR 583
- SYSTEM\_DEVELOPER\_JAVA\_TOOLS\_DIR 583

- SYSTEM\_DEVELOPER\_PERFORMANCE\_TOOLS\_DIR 583
- SYSTEM\_DEVELOPER\_UTILITIES\_DIR 583
- SYSTEM\_DOCUMENTATION\_DIR 583
- SYSTEM\_LIBRARY\_DIR 583
- Systemabhängigkeit 309
- Systemanpassungen 47
- Systemarchitektur 55
- Systemeinstellungen 26, 532
  - Sprache 344
- Systemerweiterungen 534
- Systeminstallation 32, 42
- Systemordner 533
- SystemStarter 451
- Szene-Graph 357

## T

- Tabelle 423, 424
- Tabs 166, 553
- Tabulator 590
- Tag 209, 254
- Tag Library 455
- Taglib 455
- Target 80, 90, 387
  - Abhängigkeiten 90, 314
  - aktives 80
  - Cocoa Java Specific 339
  - Cocoa Specific 339
  - Legacy 90
  - Pure Java Specific 338
  - Typ 90
- TARGET\_BUILD\_DIR 578
- TARGET\_NAME 579
- TargetSurvivorRatio 574
- Task 387
- Tastatureingaben 27
- Tastaturkürzel 16, 28
  - AWT-Menüs 128
  - Swing-Menüs 149
- Tastaturkürzel → s. Shortcut
- TCP 303
- tcsh 39, 65, 177, 414
- tearDown 400
- TEMP\_DIR 491, 579
- TEMP\_FILES\_DIR 579
- Template 24, 77

temporärer Speicher 537  
 temporäres Verzeichnis 330  
 Tenured-Generation 574  
 Terminal 38, 51, 176, 593  
 Test First-Ansatz 398  
 TestCase 398  
 Testen 398  
 Testfall 401  
 TestRunner 400, 402  
 TestSuite 398  
 TEXT 328  
 text/html 204  
 text/plain 204  
 textantialiasing 557  
 Textausgabe 536  
 TextEdit 35, 296, 298  
 Textkodierung, Xcode 158  
 Text-to-speech 343  
 this 520  
 Thread 130, 527, 528, 537, 576  
 ThreadStackSize 576  
 Thread-Synchronisation 569  
 throw 529  
 Ticker 505  
 Ticks 326  
 TIFF 354  
 Tiger 24, 541, 593  
 tiger 543  
 tigerc 543  
 Tilde 38  
 title 239  
 TKPLAFUtility 144  
 TLAB 576  
 toFront() 154  
 Together Control Center 108  
 Together Solo 108  
 Tomcat 445, 448, 465  
 Tomcat Monitor 452  
 Tool 203  
 Toolkit 149, 374, 376  
 tools.jar 288  
 top 33  
 toString() 520, 521  
 tr 177  
 transaction-type 471  
 Transaktion 431, 591  
 Transaktionsverwaltung 463  
 Transferable 169  
 Transform3D 357  
 TransformGroup 357  
 Transmit 244  
 Transparenz 189  
 Treiber, JDBC 427  
 Trennzeichen  
     Dateien 293  
     Pfadangaben 294  
 Trojaner 11  
 TruBluEnvironment 533  
 TRUE 161  
 try 528  
 TTS 343  
 Typ einer Klasse 289  
 TYPE\_INT\_ARGB 563  
 Typecast 516, 589  
 Type-Code 191, 199, 217, 296, 327  
 Typumwandlung 516, 589

**U**

Über...-Menüpunkt 130  
 Überladen 519  
 Überschreiben 521  
 Übersetzer 512, 589  
 Übersetzung 149, 157, 192, 591,  
     → s. Compiler  
 UDP 303  
 UFS 297  
 UI 119, 593  
 ui.jar 288, 328, 329  
 UIManager 145  
     put() 166  
 Umgebungsvariable 65, 69, 94, 176  
 UML 108, 113, 416, 593  
 Umlaute 83  
 Umschalttasten 28  
 Unicode 192, 209, 219, 295  
     Escape-Sequenz 129, 158  
 UninstalledProducts 578  
 Unit-Test 398  
 UNIX 11, 293, 295, 593  
 UnsatisfiedLinkError 314  
 unsichtbares Fenster 155  
 UnsupportedOperationException 145  
 Unterbrechungspunkt 88, 102  
 Unterklasse 123, 520  
 Unterpaket 512

- Unterprogramm 513
- Update 44
- updateComponentTreeUI() 145
- URI 456, 593
- URL 299, 593, 596
- URL-Clip 596
- USB 301
- USB-Docking-Station 500
- USB-Maus 80
- UseAltSigs 570
- usecolorsync 563
- UseConcMarkSweepGC 575
- usedebugkeys 561
- use-file-dialog-packages 553
- UseFileLocking 573
- useMDEF 553
- Usenet 46
- UseParallelGC 575
- User Interface 119
- user.dir 53
- user.home 52, 53
- USER\_APPS\_DIR 584
- USER\_LIBRARY\_DIR 584
- Users 31
- useScreenMenuBar 551, 553
- UseSharedGen 573
- UseSharedSpaces 576
- UseTLAB 576
- UseTwoImageLazyPixelConversion 563
- usr 30
- UTF-16 192, 219, 296
- UTF-8 209, 239, 296

## V

- Variable 515
- variable Argumentlisten 545
- Vector 147
- veraltete Methode 328
- verbose 567
- Vererbung 520, 526
  - Klassenbasiert 275
  - Prototyp-basiert 275
- Verknüpfung 31
- version.plist 192
- Versionierung 94
  - Applet 261
  - Installationspaket 227

- Java Web Start 235
- Versions (Verzeichnis) 63, 198, 211
- Versionsangabe 194, 240
- Versionsverwaltung 94
- VERTICAL\_SCROLLBAR\_ALWAYS 159
- vi 37, 40
- Videodaten 362, 366
- Viewer 204, 217
- Virtual Machine 22
- Virtual Reality 366
- Virus 11
- VISE 230
- VisualParadigm 417
- VK\_XXX 127, 149
- VM 22
- VMOptions 213, 216, 551, 565
- void 513
- Vollbildmodus 554
- vollqualifizierter Klassenname 511
- Volume 29, 220, 533, 593,
  - s. Laufwerk
  - deaktivieren 221
- Vorabversionen 46
- Vorschau 190
- VR 366

## W

- W3C 255
- Wahl-Taste 16
- WAP 502
- WAR 385, 475
- war 459, 473, 475, 581
- web 473
- Web Sharing 304
- web.xml 452, 457, 458, 473, 475
- Webapplikation 78, 455, 457, 474
- Webapplikationsarchiv 459, 473
- webapps 452, 455, 457
- Web-Browser 60, 64, 234, 265, 299, 325
- WebCore 266
- WEB-INF 452, 457, 475
- WebKit 266, 269, 339
- Weblog 459
- WebLogic Server 464
- WebObjects 464, 479
- Web-Server 234, 302, 445
- Web-Sites 239, 242

- web-uri 473
- Weichzeichnen 556, 589
- WEST 125
- while 517
- WHITE 134
- Window 124
- window 276
- WindowAdapter 124
- windowClosing() 124, 365
- WindowEvent 124
- WindowListener 124, 363
- windowModified 160
- Windows 23, 24, 55, 57, 186, 267, 293, 295, 296, 302, 304, 311
- Wireless Toolkit 485, 494
- Word-Dokumente 35
- WorkingDirectory 213, 216, 550
- Wrapper 311
- Wurm 11
- Wurzel-Ebene 161
- Wurzelelement 209
- Wurzelverzeichnis 30
- Wurzelverzeichnis → s. root

**X**

- X Window System 486
- X<sub>11</sub> 486, 495
  - FAQ 508
- Xcode 24, 42, 62, 76, 186, 203, 245, 577, 593
  - Ant 77, 388
  - Ant-Projekttypen 393
  - Applet 257
  - AWT 122
  - Build 86
  - Cocoa Java 331
  - Custom Build Command 389
  - Debugger 88
  - Dokumentation 84
  - Editor teilen 82
  - Eingabevervollständigung 83
  - Executable 392, 401
  - External Target 389
  - Fehler und Warnungen 86
  - Index neu erstellen 83
  - Index Templates 84
  - Java-API-Dokumentation 85
  - Jikes 413
  - JNI 310
  - JUnit 401
  - Klassenhierarchie statt JAR-Archiv 186
  - Konsole 87
  - Lesezeichen 82
  - LiveConnect 277
  - Maven 397
  - MIDlets 487
  - QTJava 369
  - Servlets 447
  - Shark 409
  - Suchen 83
  - Swing 146
  - Target löschen 389
  - Tastenfunktionen 82
  - Textkodierung 158
  - Umlaute im Quelltext 83
  - Unterbrechungspunkt 88
  - Version 2.0 76
  - Versionen 1.2 und 1.5 76
  - XDoclet 477
- Xcode Tools 42, 386
- xcodebuild 87, 577
- xcodeindex 84
- Xdock 180
- XDoclet 62, 78, 465, 477, 478
  - Xcode 477
- XML 22, 54, 208, 238, 279, 593
  - Dokument 214
  - Editor 417
  - Prolog 209
  - Transformation 417
- Xnoclassgc 409
- XP 398
- Xrun 569
- XrunShark 409
- XSL-FO 417
- XSLT 417

**Y**

- YourSQL 437

**Z**

- Zeichen, Kodierung 295
- Zeichen, zusammengesetzte 297

Zeichenkette 514  
Zeilenende 184, 294  
    umwandeln 177  
Zeilenumbruch 177, 294, 548  
Zeitmessung 301  
zentrieren auf dem Bildschirm 129  
Zeroconf 302

Zertifikat 246  
    selbst-signiert 247  
ZIP-Archiv 183  
Zugriffsrechte 32, 53, 178  
Zusicherung 570  
Zwischenablage 248