Auf einen Blick

Inhalt

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)[1]

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.

1 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-Programmierumgebungen 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 Mac OS 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 Mac OS 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 Mac OS 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 Mac OS 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 Mac OS 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 Mac OS 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 heraussuchen 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)	`/* Schrift mit fester Zeichenbreite */`

Java-Bezeichner (Variablen, Methoden, Klassen)	`Schrift mit fester Zeichenbreite:`
	`meinObjekt`
	`indexOf()`
	`String`

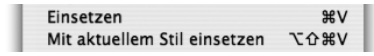
Dateinamen und Pfadangaben	`Schrift mit fester Zeichenbreite:` `/Programme/TextEdit`

Verwendete Schriftkonventionen

Textbestandteil	Darstellung
Befehle und Programmausgaben	`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)	*Kursivschrift:* *http://www.muchsoft.com/java/*
Menüs und Menüeinträge, Texte auf dem Bildschirm	**Menütitel · Menüeintrag · Untermenüeintrag** usw.: **Bearbeiten · Kopieren** **Ablage · Sichern unter...** Einzelne Menüeinträge und Texte auf dem Bildschirm werden auch in Anführungszeichen gesetzt.
Tastaturkürzel (Shortcuts)	Die Umschalttasten werden ausgeschrieben. Einzelne Buchstaben werden großgeschrieben, ⇧ ist dabei aber nur zu drücken, wenn dies auch in der Tastenkombination steht. Sonstige Aktionen werden normal dargestellt: ⌘+Q Ctrl+Klick 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 ⌘+V, darunter den Eintrag »Mit aktuellem Stil einsetzen« mit dem Kürzel ⌥+⇧+⌘+V. Die Umschalttaste Ctrl 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 Matteschеck 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* 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 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 Mac OS 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** (Codename »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 Mac OS 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-Sourcen 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 Benutzungsoberfläche (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 1.1 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 ⌘+⇥ 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 ⌘+⇥-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: ⌘ (auch Propeller- oder Befehlstaste genannt), ⌥ (Option- oder Wahltaste), Ctrl (Steuerung) und ⇧ (Umschalttaste). Mobile Rechner haben zusätzlich noch eine Fn-Taste, die aber für Programm-Shortcuts nicht relevant ist. Die wichtigsten Kürzel sind wohl ⌘+W zum Schließen des obersten (aktiven) Fensters und ⌘+Q zum Beenden der aktiven Applikation.

Vielleicht sind Sie es von Ihrem System gewohnt, dass Sie sich mit ⇥ 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 ⇥ 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 ⌘+X/C/V auf. Beachten Sie, dass Windows mangels ⌘-Taste stattdessen einfach 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 ⌘+O 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 Ein-Tasten-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 ⌥+⌘+Esc erreichen. Seit Mac OS X 10.3 können Sie die oberste Applikation direkt ohne Nachfrage mit ⇧+⌥+⌘+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 ⌥- und ⌘-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-Benutzungsoberfläche (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 C, um von einer eingelegten CD-ROM zu starten. Wenn Sie stattdessen ⌥ 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.

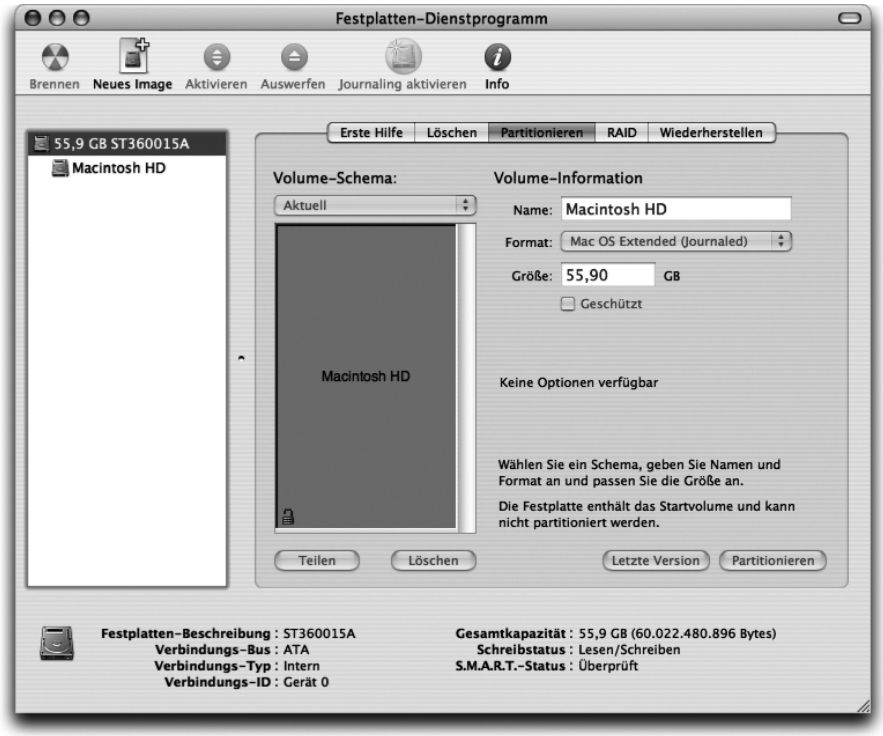


Abbildung 1.4 Festplatten-Dienstprogramm

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.

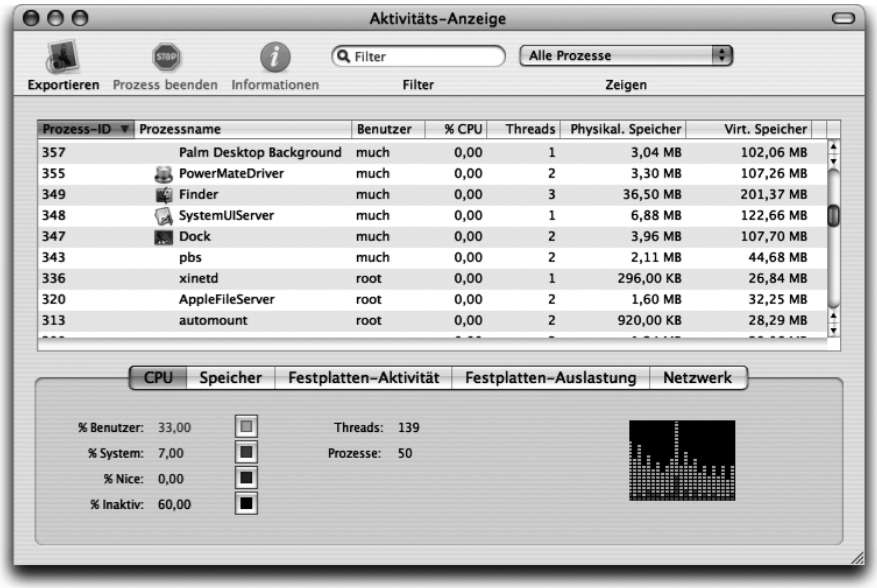


Abbildung 1.5 Aktivitäts-Anzeige

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
⌘+A	Alles auswählen
⌘+C	Markiertes Element in die Zwischenablage kopieren
⌘+F/⌘+G	Suchen/Weitersuchen

Tabelle 1.1 Die wichtigsten Mac OS X-Tastenkürzel[1]

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
Ctrl + ⏏	Dialog »Computer jetzt ausschalten?« öffnen. Hier können Sie mit ↵ den Rechner ausschalten, mit R (»Reboot«) neu starten und mit S (»Sleep«) in den Ruhezustand versetzen. Esc bricht den Dialog ab.
Fn + ←	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 Startvolume 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.

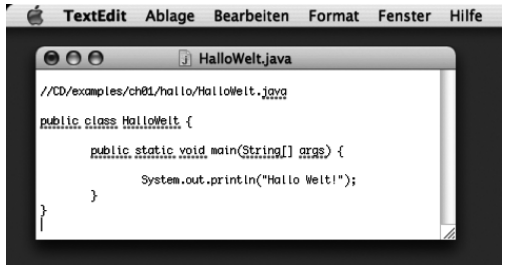


Abbildung 1.6 »Hallo Welt« in TextEdit

Wenn Sie die roten Unterstreichungen stören, die in einem Programmquelltext 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

herunterladen.[2] 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«. Mac OS 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[3], 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 Mac OS 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 ⌥+N 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

3 Der Rechnername lautet in diesem Fall »Straylight«. Sie können den Namen Ihres Rechners in den Systemeinstellungen unter Sharing • Gerätename 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.

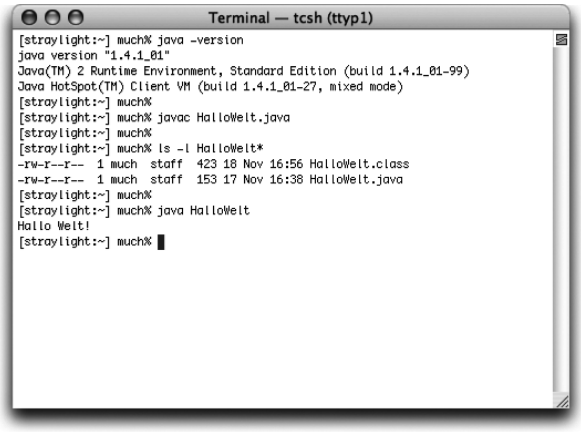


Abbildung 1.8 Die tcsh-Shell im Terminal

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 Mac OS 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).

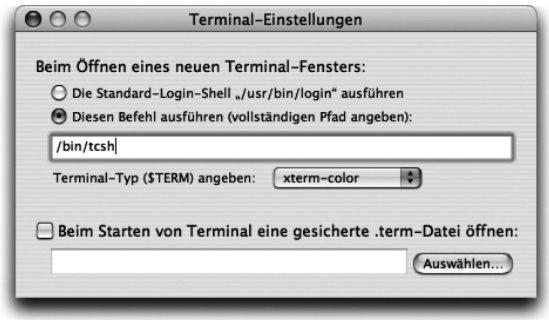


Abbildung 1.9 Terminal-Einstellungen für die tcsh-Shell

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.

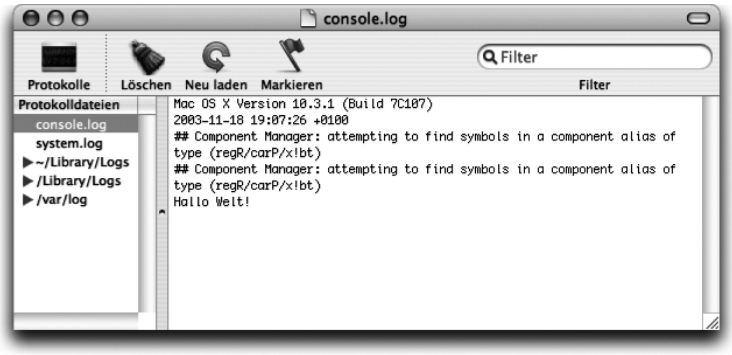


Abbildung 1.10 Java-Textausgabe in der Konsole

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 [Q] 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 `dmgpart`-Dateien befinden. Zum Aktivieren (»mounten«) des kompletten Volumes führen Sie dann einfach einen Doppelklick nur auf die `dmg`-Datei aus.

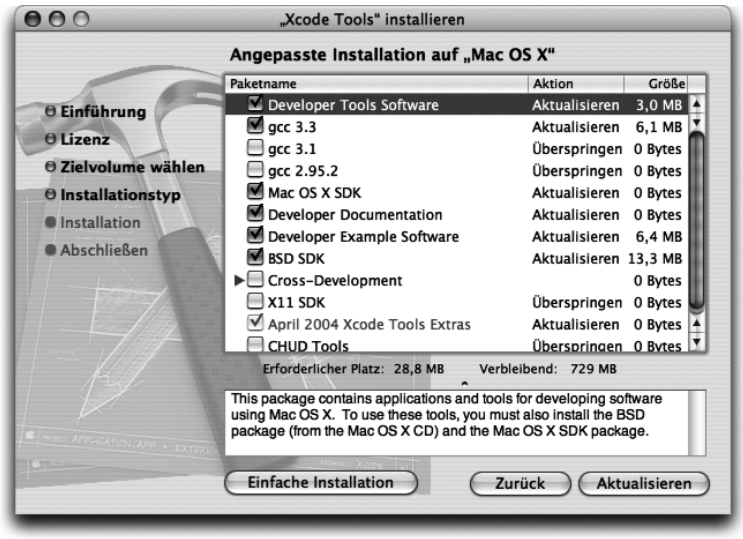


Abbildung 1.11 Xcode-Tools installieren

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 Mac OS 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[4].

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

4 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[5] oder MacSOUP[6]) 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/*

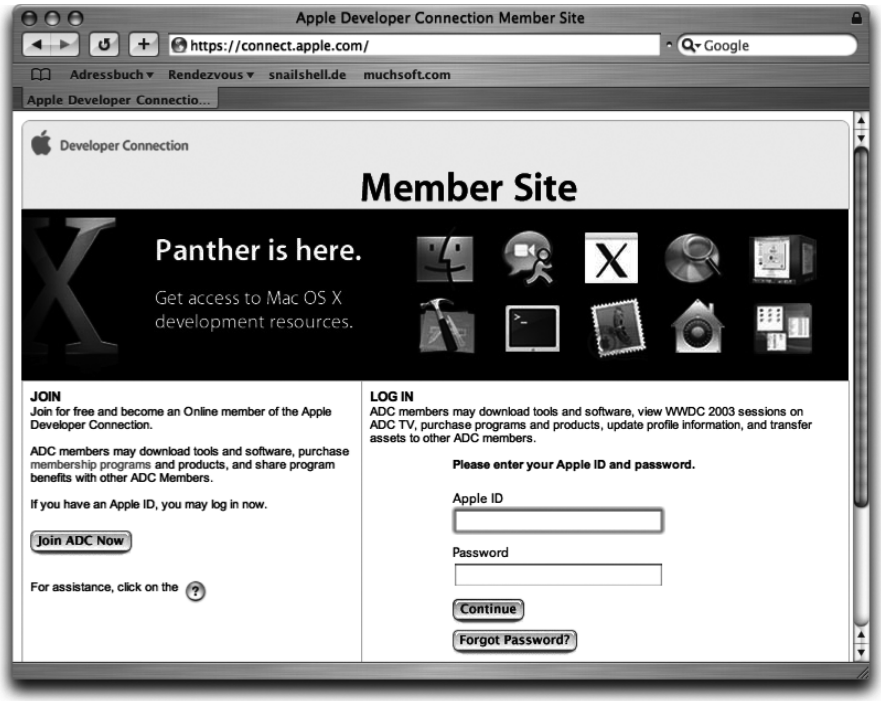


Abbildung 1.13 Apple Developer Connection (ADC): Anmeldung und Login

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 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-Codecs 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 Mac OS 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 Mac OS 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 szene = this.createSceneGraph();
        szene.compile();
        SimpleUniverse universe = new SimpleUniverse( canvas );
```

```
        universe.getViewingPlatform().setNominalViewingTransform();
        universe.addBranchGraph( szene );
    }
```

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 szene = 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 );
        szene.addChild( rotation );
        return szene;
    }
//...
}
```

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.

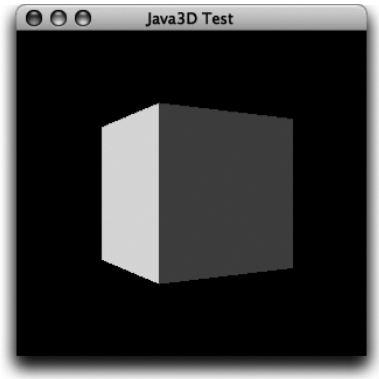


Abbildung 7.1 Drehender Würfel mit Java 3D

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.

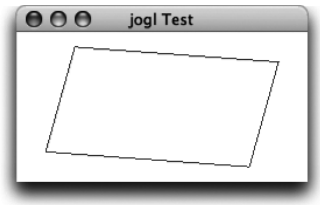


Abbildung 7.2 2D-Ausgabe mit OpenGL bzw. JOGL

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.

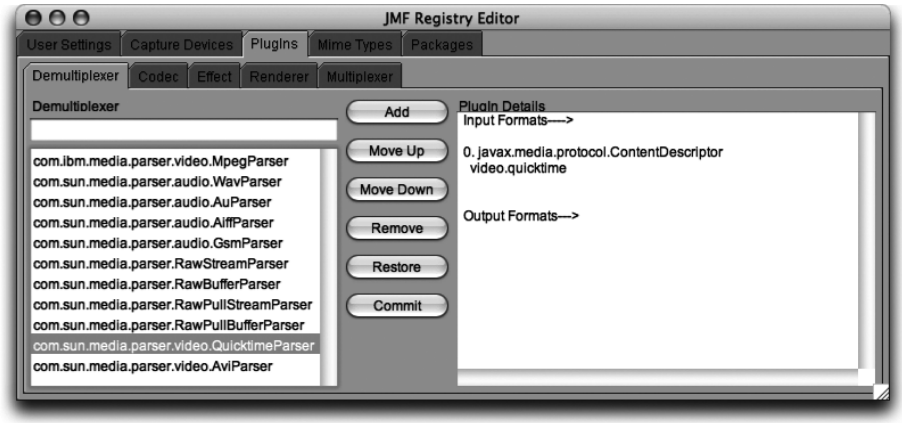


Abbildung 7.3 Registry Editor zur Konfiguration vom JMF

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 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 Multimediabibliotheken, die es gibt. Seit Anfang der 1990er Jahre steht die Software für Mac OS und Windows zur Verfügung[1], und bei Mac OS X ist sie fester Bestand-

1 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 Mac OS 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 (QTException 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 (QTException 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.

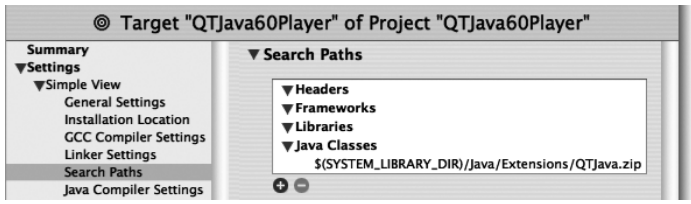


Abbildung 7.5 QTJava-Klassen zum Suchpfad hinzufügen

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 (QTException 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://openqtj.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.setOuputFile( 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

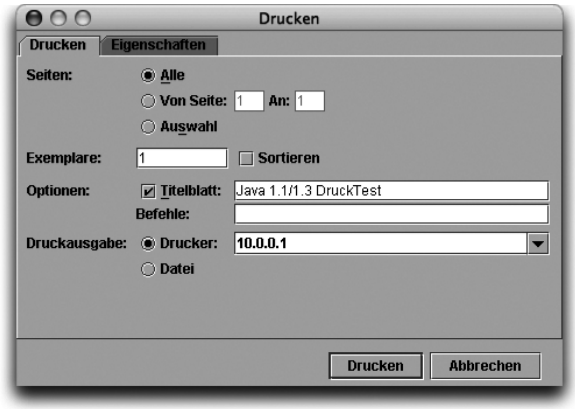


Abbildung 7.10 Javas systemübergreifender Druckdialog

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 Programmiereinführung 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

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z