Kapitel **2**

Die Programmierumgebung

In diesem Kapitel lernen Sie, wie man das Java Software Development Kit (SDK) installiert und wie sich die verschiedenen Arten von Programmen – Konsolenprogramme, grafische Anwendungen und Applets – kompilieren und ausführen lassen. Die SDK-Tools rufen Sie über Befehle in einem Shell-Fenster auf. Viele Programmierer ziehen aber die Annehmlichkeiten einer integrierten Entwicklungsumgebung vor. Deshalb zeigen wir hier, wie Sie eine frei verfügbare Entwicklungsumgebung einsetzen, um Java-Programme zu kompilieren und auszuführen. Die integrierten Entwicklungsumgebungen sind zwar einfacher zu erlernen und einzusetzen, brauchen aber länger zum Laden und beanspruchen viele Ressourcen. Als goldenen Mittelweg könnte man einen Texteditor verwenden, der den Java-Compiler und -Interpreter aufrufen kann. Dieses Kapitel stellt eine Reihe von Texteditoren mit Java-Integration vor. Nachdem Sie die Verfahren in diesem Kapitel beherrschen und sich Ihre Entwicklungswerkzeuge herausgesucht haben, können Sie sich Kapitel 3 zuwenden. Dort beginnen Sie, sich mit der Programmiersprache Java bekannt zu machen.

2.1 Das Java Software Development Kit installieren

Die umfassendsten und neuesten Versionen der Java 2 Standard Edition (J2SE) sind von Sun Microsystems für Solaris, Linux und Windows verfügbar. Für den Macintosh und viele andere Plattformen gibt es Versionen in verschiedenen Entwicklungsphasen, wobei diese Versionen aber durch die Anbieter dieser Plattformen lizenziert und vertrieben werden.

Wenn Sie unter Solaris, Linux oder Windows arbeiten, können Sie sich das Java Software Development Kit von *http://java.sun.com/j2se* herunterladen. Die Installationshinweise unterscheiden sich für jede Plattform. Momentan finden Sie ihre Version unter:

- *http://java.sun.com/j2se/1.4/install-solaris.html*
- *http://java.sun.com/j2se/1.4/install-linux.html*
- *http://java.sun.com/j2se/1.4/install-windows.html*

Hinweis

Nur die Installations- und Kompilierungsanweisungen für Java sind systemabhängig. Nachdem Sie Java installiert haben und ausführen können, sollte alles andere in diesem Buch auf Ihr System zutreffen. Die Systemunabhängigkeit ist einer der großen Vorzüge von Java.

Hinweis

Das Setupprogramm bietet einen Standardpfad für das Installationsverzeichnis an, in dem die Versionsnummer des Java SDK enthalten ist, z.B. `j2sdk1.4.0`. Dieses Verzeichnis können Sie ohne weiteres in `jdk` ändern. Falls Sie als Java-Enthusiast die verschiedenen Versionen des Java SDK sammeln, können Sie auch die Standardeinstellung übernehmen. In diesem Buch beziehen wir uns auf das Installationsverzeichnis `jdk`. Ist zum Beispiel vom Verzeichnis `jdk/bin` die Rede, dann ist das Verzeichnis `bin` unter dem Installationsverzeichnis des Java SDK gemeint. Beachten Sie auch, dass hier die UNIX-Schreibweise für Verzeichnisnamen angegeben ist. Unter Windows verwenden Sie Laufwerksbuchstaben und den Backslash in Pfadbezeichnungen, zum Beispiel `c:\jdk\bin`.

2.1.1 Ausführungspfad festlegen

Nachdem Sie das Java SDK installiert haben, müssen Sie einen zusätzlichen Schritt durchführen: das Verzeichnis `jdk/bin` in den Ausführungspfad aufnehmen, d.h. in die Liste der Verzeichnisse, die das Betriebssystem bei der Suche nach ausführbaren Dateien durchläuft. Die Richtlinien für diesen Schritt unterscheiden sich ebenfalls bei den einzelnen Betriebssystemen.

- In UNIX (einschließlich Solaris oder Linux) hängt der Vorgang für die Bearbeitung des Ausführungspfades von der verwendeten *Shell* ab. Wenn Sie mit der C-Shell (der Solaris-Standardeinstellung) arbeiten, fügen Sie eine Zeile wie die folgende an das Ende Ihrer `~/.cshrc`-Datei an:

  ```
  set path=(/usr/local/jdk/bin $path)
  ```

  Bei der Bourne Again-Shell (dem Linux-Standard) fügen Sie folgende Zeile an das Ende der Datei `~/.bashrc` bzw. `~/.bash_profile` an:

  ```
  export PATH=/usr/local/jdk/bin:$PATH
  ```

  Für andere UNIX-Shells konsultieren Sie bitte die Dokumentation, um die entsprechenden Schritte durchzuführen.

- Unter Windows 95/98/ME schreiben Sie eine Zeile wie die folgende an das Ende der Datei `AUTOEXEC.BAT`:

  ```
  SET PATH=c:\jdk\bin;%PATH%
  ```

  Beachten Sie, dass weder vor noch nach dem Gleichheitszeichen Leerzeichen stehen. Damit die vorgenommene Änderung wirksam wird, müssen Sie den Computer neu starten.

- Unter Windows NT/2000/XP gehen Sie in die Systemsteuerung, wählen SYSTEM, gehen auf die Registerkarte ERWEITERT und klicken auf UMGEBUNGSVARIABLEN. Im Fenster BENUTZERVARIABLEN suchen Sie den Eintrag mit der Variablen `path` auf. Fügen Sie das Verzeichnis `jdk\bin` am Beginn des Pfades hinzu und schreiben Sie ein Semikolon, um diesen Eintrag vom nächsten zu trennen, zum Beispiel:

  ```
  c:\jdk\bin;andere Pfadeinträge
  ```

  Speichern Sie Ihre Einstellungen. Die neuen Pfadangaben sind für jedes Konsolenfenster gültig, das Sie ab jetzt öffnen.

Ob Sie die Schritte richtig ausgeführt haben, können Sie wie folgt testen:

Starten Sie ein Shell-Fenster. Die dazu erforderlichen Befehle hängen vom jeweiligen Betriebssystem ab. Geben Sie die Zeile

```
java -version
```

ein und drücken Sie die ↵-Taste. Daraufhin sollte etwa folgende Ausgabe erscheinen:

```
java version "1.4.0"
Java(TM) 2 Runtime Environment, Standard Edition
Java HotSpot(TM) Client VM
```

Wenn Sie stattdessen eine Meldung erhalten, die besagt, dass der Befehl oder Dateiname nicht gefunden oder erkannt wurde, sollten Sie die oben angegebenen Schritte wiederholen und dabei peinlichst genau auf Fehler achten.

2.1.2 Bibliotheksquellen und Dokumentation installieren

Die Bibliotheksquelldateien sind im SDK als komprimierte Datei `src.jar` enthalten. Diese Datei müssen Sie erst entpacken, bevor Sie auf den Quellcode zugreifen können. Diesen Schritt sollten Sie unbedingt ausführen. Gehen Sie wie folgt vor:

1. Vergewissern Sie sich, dass das Java SDK installiert und das Verzeichnis `jdk/bin` in der Pfadeinstellung angegeben ist.
2. Öffnen Sie eine Befehls-Shell.
3. Wechseln Sie zum Verzeichnis `jdk` (zum Beispiel `/usr/local/jdk` oder `C:\jdk`).
4. Führen Sie den folgenden Befehl aus:

   ```
   jar xvf src.jar
   ```

Tipp

Die Datei `src.jar` enthält den Quellcode für alle öffentlichen Bibliotheken. Weitere Quellen (für den Compiler, die virtuelle Maschine, die systemeigenen Methoden und die privaten Hilfsklassen) erhalten Sie auf der Website *http://www.sun.com/software/java2*.

Die Dokumentation ist in einer komprimierten und vom Java SDK getrennten Datei enthalten und lässt sich von der Webseite *http://java.sun.com/docs* herunterladen. Es sind verschiedene Formate (`.zip`, `.gz` und `.Z`) verfügbar. Nehmen Sie einfach das Format, das Sie ohne weiteres dekomprimieren können. Im Zweifelsfall wählen Sie die ZIP-Datei, da Sie diese mit dem Programm `jar` dekomprimieren können, das zum Lieferumfang des Java SDK gehört. Beim Programm `jar` sind folgende Schritte auszuführen:

1. Vergewissern Sie sich, dass das Java SDK installiert und das Verzeichnis `jdk/bin` im Ausführungspfad angegeben ist.
2. Laden Sie die ZIP-Datei mit der Dokumentation herunter und verschieben Sie sie in das `jdk`-Verzeichnis. Die Datei heißt `j2sdk`*Version*`-doc.zip`, wobei *Version* in der Form `1_4_0` spezifiziert ist.
3. Öffnen Sie eine Befehls-Shell.
4. Wechseln Sie in das Verzeichnis `jdk`.
5. Führen Sie den Befehl

   ```
   jar xvf j2sdkVersion-doc.zip
   ```

   aus, wobei *Version* für die jeweilige Versionsnummer steht.

2.1.3 Programmbeispiele von Core Java installieren

Die Beispiele für die im Buch angegebenen Programme können Sie sowohl von der Begleit-CD (mit den deutschen Versionen) als auch von der Website *http://www.phptr.com/corejava* (amerikanische Originalversionen) herunterladen. Die Programme sind in der ZIP-Datei `corejava.zip` verpackt. Entpacken Sie diese Dateien in ein separates Verzeichnis – zum Beispiel `CoreJavaBook`. Zu diesem Zweck können Sie Dienstprogramme wie WinZip (*http://www.winzip.com*) oder `jar` (das zum Java SDK gehört) verwenden. Beim Programm `jar` führen Sie die folgenden Schritte aus:

1. Vergewissern Sie sich, dass das Java SDK installiert und das Verzeichnis `jdk/bin` im Ausführungspfad angegeben ist.
2. Erstellen Sie ein Verzeichnis `CoreJavaBook`.
3. Kopieren Sie die Datei `corejava.zip` in dieses Verzeichnis.
4. Öffnen Sie eine Befehls-Shell.
5. Wechseln Sie zum Verzeichnis `CoreJavaBook`.
6. Führen Sie den folgenden Befehl aus:

```
jar xvf corejava.zip
```

2.1.4 Die Java-Verzeichnisse

Bei Ihren Erkundungen in Java wollen Sie sicher gelegentlich auch einen Blick in die Java-Quelldateien werfen. Und natürlich werden Sie hin und wieder die Bibliotheksdokumentation konsultieren müssen. Tabelle 2.1: zeigt die Verzeichnisstruktur von Java. Die Anordnung kann abweichen, wenn Sie mit einer integrierten Entwicklungsumgebung arbeiten. Das Stammverzeichnis hängt von der installierten SDK-Version ab.

Verzeichnis	Inhalt
`\jdk\`	Stammverzeichnis des SDK (der Name kann auch anders lauten, beispielsweise `j2sdk1.4.0`)
`Docs`	Bibliotheksdokumentation im HTML-Format (nach Entpacken von `j2sdkVersion-doc.zip`)
`Bin`	Compiler und Tools
`Demo`	Beispieldateien
`Include`	Dateien für systemeigene Methoden (siehe Band 2)
`Lib`	Bibliotheksdateien
`src`	In den verschiedenen Unterverzeichnissen finden Sie den Quellcode der Bibliotheken (nachdem Sie `src.jar` entpackt haben).
`jre`	Dateien der Java-Laufzeitumgebung

Tabelle 2.1: Verzeichnisstruktur der Java-Installation

Die beiden wichtigsten Unterverzeichnisse in dieser Struktur sind `docs` und `src`. Das Verzeichnis `docs` enthält die Bibliotheksdokumentation zu Java im HTML-Format. Man kann sie mit jedem Webbrowser – zum Beispiel Netscape – ansehen.

Tipp

Setzen Sie in Ihrem Browser ein Lesezeichen auf die lokale Version von *docs/api/index.html*. Auf diese Seite werden Sie sich oft beziehen, wenn Sie die Java-Plattform erkunden.

Das Verzeichnis `src` enthält den Quellcode für den öffentlichen Teil der Java-Bibliotheken. Wenn Sie mit Java vertrauter werden, helfen Ihnen manchmal weder dieses Buch noch die Online-Informationen weiter. In diesem Fall stellt der Quellcode für Java einen guten Ausgangspunkt zur weiterführenden Suche dar. Manchmal ist es schon beruhigend, wenn man in den Quellcode eintauchen kann, um sich über die genaue Arbeitsweise einer Bibliotheksfunktion ein Bild zu machen. Wenn Sie sich beispielsweise für die inneren Abläufe in der Klasse `System` interessieren, können Sie sich den Quellcode von `src/java/lang/System.java` zu Gemüte führen.

2.2 Entwicklungsumgebungen

Wenn Sie Ihre Programmiererfahrungen mit Visual Basic oder Visual C++ gesammelt haben, sind Sie an eine Entwicklungsumgebung gewöhnt, in der Sie mit dem integrierten Editor Quelltexte erstellen, über Menübefehle ein Programm kompilieren und ausführen sowie auf einen integrierten Debugger zurückgreifen können. Das grundlegende Java SDK enthält nichts, was im Entferntesten daran erinnert. *Sämtliche* Aufgaben erledigt man, indem man Befehle in einem Shell-Fenster eintippt. In diesem Kapitel erfahren Sie, wie Sie das grundlegende Java SDK installieren und einsetzen, da wir festgestellt haben, dass sich ausgewachsene Entwicklungsumgebungen nicht unbedingt eignen, um Java zu erlernen – sie können sehr komplex sein und den Programmierer von den interessanten und wichtigen Details abschirmen.

Für einfache Programme erweisen sich integrierte Entwicklungsumgebungen manchmal als zu schwerfällig, weil sie langsamer sind, leistungsfähigere Computer voraussetzen und häufig eine relativ aufwändige Projekteinrichtung für das zu schreibende Programm verlangen. Diese Umgebungen eignen sich vor allem, wenn man größere Java-Programme schreibt, die aus vielen Quelldateien bestehen. Außerdem bieten diese Umgebungen Debugger, die für eine ernsthafte Entwicklung unerlässlich sind – der zum Java SDK gehörende kostenlose Befehlszeilendebugger ist extrem unhandlich. Wir zeigen, wie Sie mit der Sun ONE Studio Community Edition beginnen, einer frei verfügbaren Entwicklungsumgebung, die selbst in Java geschrieben ist. (Bevor die Übereifrigen aus der Marketing-Abteilung von Sun auf den Plan getreten sind, war das Programm als Forte bekannt.) Wenn Sie bereits über eine Entwicklungsumgebung wie JBuilder, Kawa, CodeWarrior oder Café verfügen, die die aktuelle Version von Java unterstützt, dann können Sie sie selbstverständlich auch für die im Buch gezeigten Beispiele einsetzen.

Für einfache Programme besteht ein brauchbarer Kompromiss zwischen Befehlszeilenwerkzeugen und einer integrierten Entwicklungsumgebung in einem Editor, der mit dem Java SDK integriert ist. Unter Linux bevorzugen wir Emacs. Für Windows bietet sich auch TextPad an, ein ausgezeichneter Shareware-Editor mit guter Java-Integration. Schließlich stellt JEdit eine hervorragende plattformübergreifende Alternative dar. Mit einem Editor, der mit dem Java SDK zusammenarbeitet, läuft die Entwicklung von Java-Programmen leicht und schnell ab. Die meisten Programme in diesem Buch sind auf diese Weise entwickelt und getestet worden. Da sich der Quellcode aus dem Editor heraus kompilieren und ausführen lässt, kann ein derartiges Tool zu Ihrer De-facto-Entwicklungsumgebung für die Beispiele im Buch avancieren.

Insgesamt können Sie unter drei Möglichkeiten für eine Entwicklungsumgebung wählen:

- Nehmen Sie das Java SDK und Ihren bevorzugten Editor. Kompilieren und starten Sie Programme in einer Befehls-Shell.
- Verwenden Sie das Java SDK und einen Editor, in den sich das Java SDK integrieren lässt. Emacs und TextPad besitzen diese Fähigkeit und es gibt daneben noch viele andere. Kompilieren und starten Sie Programme aus dem Editor heraus.
- Arbeiten Sie mit einer integrierten Entwicklungsumgebung wie zum Beispiel der kostenlos erhältlichen Sun ONE Studio Community Edition oder einer der vielen anderen frei verfügbaren oder kommerziell erhältlichen Umgebungen.

2.3 Befehlszeilenwerkzeuge

Ein Java-Programm lässt sich nach zwei Verfahren kompilieren und starten: von der Befehlszeile oder aus einem anderen Programm heraus, beispielsweise einer integrierten Entwicklungsumgebung oder einem Editor. Schlagen wir zunächst den schwereren Weg ein: von der Befehlszeile.

Öffnen Sie ein Shell- oder Terminalfenster. Wechseln Sie in das Verzeichnis `CoreJavaBook/v1ch2/Welcome` und geben Sie dann die folgenden Befehle ein:

```
javac Welcome.java
java Welcome
```

Daraufhin sollte die in Abbildung 2.1 dargestellte Meldung auf dem Bildschirm erscheinen.

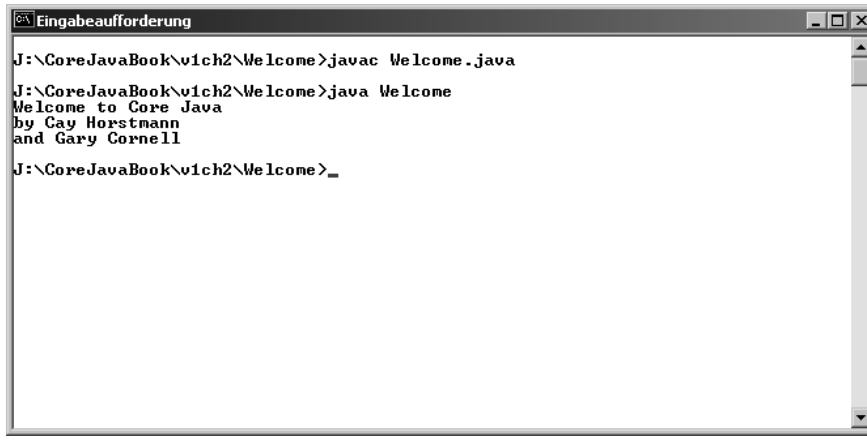

Abbildung 2.1: Welcome.java kompilieren und ausführen

Gratulation! Sie haben gerade Ihr erstes Java-Programm kompiliert und gestartet.

Was ist im Einzelnen passiert? Das Programm `javac` ist der Java-Compiler. Er kompiliert die Datei `Welcome.java` in die Datei `Welcome.class`. Beim Programm `java` handelt es sich um den Java-Interpreter. Er interpretiert die Bytecodes, die der Compiler in die Klassendatei geschrieben hat.

Das Programm `Welcome` ist extrem einfach. Es gibt lediglich eine Meldung auf der Konsole aus. Sehen Sie sich das in Beispiel 2.1 wiedergegebene Programm einmal näher an – eine Erläuterung zur Arbeitsweise folgt im nächsten Kapitel.

Beispiel 2.1: Welcome.java

```
1:  public class Welcome
2:  {
3:     public static void main(String[] args)
4:     {
5:        String[] greeting = new String[3];
6:        greeting[0] = "Welcome to Core Java";
7:        greeting[1] = "by Cay Horstmann";
8:        greeting[2] = "and Gary Cornell";
9:
10:       for (int i = 0; i < greeting.length; i++)
11:          System.out.println(greeting[i]);
12:    }
13: }
```

2.3.1 Hinweise zur Fehlersuche

Im Zeitalter der visuellen Entwicklungsumgebungen sind viele Programmierer nicht vertraut damit, Programme in einem Shell-Fenster auszuführen. Es gibt eine Reihe von Dingen, die dabei schief gehen können und zu frustrierenden Ergebnissen führen.

Achten Sie insbesondere auf folgende Punkte:

- Halten Sie sich genau an die Groß-/Kleinschreibung, wenn Sie das Programm manuell eintippen. Heißt eine Klasse zum Beispiel `Welcome`, ist das nicht dasselbe wie `welcome` oder `WELCOME`.
- Der Compiler verlangt einen *Dateinamen* im Format `Welcome.java` und der Interpreter erwartet einen *Klassennamen* der Form `Welcome` ohne eine Erweiterung `.java` oder `.class`.
- Wenn Sie eine Meldung erhalten, die sich auf einen fehlenden Befehl bezieht, sollten Sie Ihre Installation und hierbei insbesondere die Einstellungen für den Ausführungspfad überprüfen.
- Falls `javac` eine Meldung wie »error: cannot read: Welcome.java« liefert, müssen Sie prüfen, ob die Datei `Welcome.java` im aktuellen Verzeichnis vorhanden ist.

  Unter UNIX ist auch die Groß-/Kleinschreibung für den Dateinamen `Welcome.java` zu berücksichtigen.

  Zeigen Sie unter Windows eine Liste der Dateien über den Shell-Befehl `dir` und nicht mit dem Explorer an. Einige Editoren (insbesondere der Windows-Editor *Notepad.exe*) verlangen bei jeder Datei die Dateierweiterung `.txt`. Wenn Sie die Datei `Welcome.java` mit dem Windows-Editor bearbeiten, wird sie als `Welcome.java.txt` gespeichert. Gelten für Ihr System die Standardeinstellungen, unterdrückt der Explorer die Anzeige der Erweiterung `.txt`, weil diese zu den normalerweise nicht angezeigten »bekannten Dateitypen« gehört.

- Wenn `java` eine Fehlermeldung mit dem Inhalt »java.lang.NoClassDefFoundError« liefert, müssen Sie die Schreibweise der infrage kommenden Klasse genau prüfen.

  Beschwert sich der Interpreter über `welcome` (mit einem kleinen `w`), führen Sie den Befehl `java Welcome` erneut mit einem großen `W` aus. In Java spielt die Groß-/Kleinschreibung immer eine Rolle.

  Ist in der Fehlermeldung des Interpreters `Welcome` angegeben, dann wurde der Klassenpfad in Ihrem System geändert. Entweder entfernen Sie die Einstellung dieser Umgebungsvariablen oder Sie nehmen das aktuelle Verzeichnis (durch einen Punkt symbolisiert) in den Klassenpfad auf. Kapitel 4 geht näher darauf ein.
- Enthält Ihr Programm viele Fehler, dann huschen die jeweiligen Fehlermeldungen sehr schnell über den Bildschirm. Der `java`-Interpreter sendet die Fehlermeldungen an den Standardausgabestrom, sodass die Meldungen nur schwer zu erfassen sind, wenn sie mehr als einen Bildschirm einnehmen.
- Auf einem UNIX- oder Windows NT/2000/XP-System ist das kein großes Problem. Mit dem Shell-Operator `2>` können Sie die Fehler in eine Datei umleiten:

  ```
  javac MeinProg.java 2> fehler.txt
  ```

  Unter Windows 95/98/ME lässt sich der Standardfehlerstrom nicht von der Befehls-Shell aus umleiten. Von *http://www.horstmann.com/corejava/faq.html* können Sie aber das Programm `errout.exe` herunterladen und dann den Befehl

  ```
  errout javac MeinProg.java > fehler.txt
  ```

  ausführen.

Tipp

Unter *http://java.sun.com/docs/books/tutorial/getStarted/cupojava/* ist ein ausgezeichnetes Tutorial verfügbar, das tiefgründiger auf die »Fallstricke« eingeht, mit denen Einsteiger zu kämpfen haben.

2.4 Eine integrierte Entwicklungsumgebung

Dieser Abschnitt zeigt, wie Sie ein Programm mit der Sun ONE Studio Community Edition kompilieren. Dabei handelt es sich um eine frei von Sun Microsystems erhältliche Entwicklungsumgebung, die Sie sich über *http://www.sun.com/software/sundev/jde* herunterladen können. Sun ONE Studio ist in Java geschrieben und sollte sich auf jeder Plattform ausführen lassen, die über eine Java 2-Laufzeitumgebung verfügt. Es gibt vorkonfigurierte Versionen für Solaris, Linux und Windows. Eine ausgefeiltere Open Source-Version können Sie von *http://netbeans.org* beziehen.

Nach dem Start von Sun ONE Studio erscheinen verschiedene Symbolleisten und Fenster (siehe Abbildung 2.2).

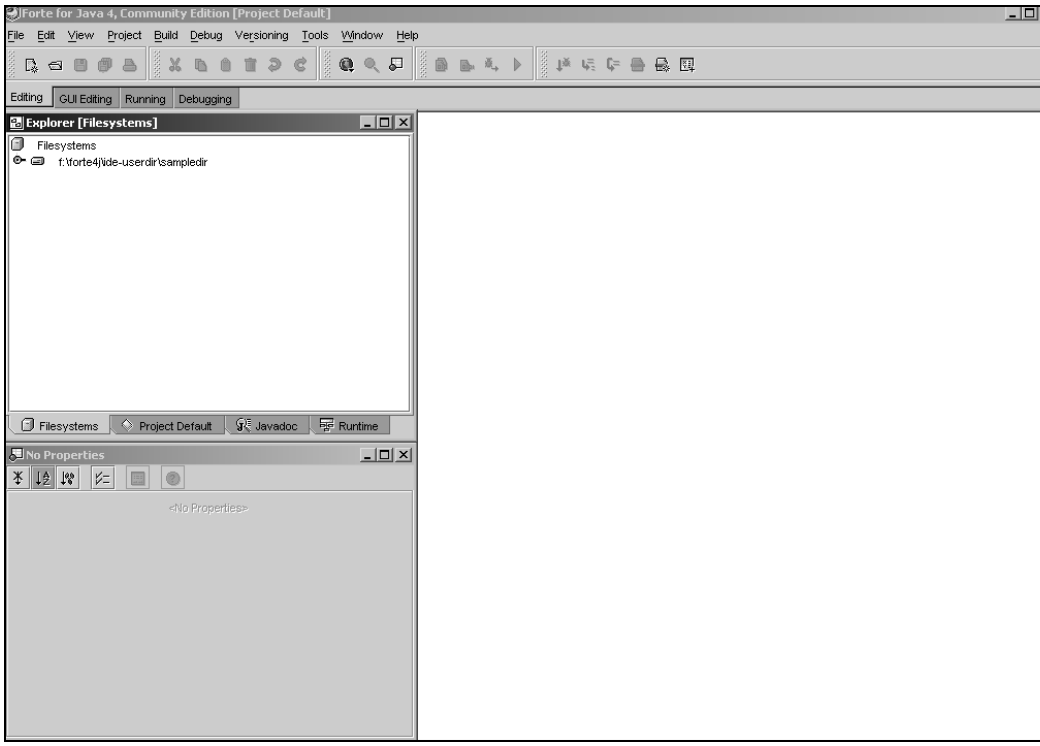

Abbildung 2.2: Sun ONE Studio nach dem Start

Wählen Sie den Menübefehl FILE / OPEN FILE und laden Sie die Datei `CoreJavaBook/v1ch2/Welcome/Welcome.java`. Es erscheint die Frage, ob Sie die Datei in das Standardpaket aufnehmen möchten (»The IDE has guessed that the file Welcome.java should be in the default package...«). Klicken Sie auf ACCEPT. (Kapitel 4 bringt weitere Details zu Paketen. Momentan nehmen Sie alle Ihre Programme in das Standardpaket auf.) Es sollte jetzt ein Fenster mit dem Programmcode erscheinen (siehe Abbildung 2.3).

Das Programm kompilieren Sie über den Menübefehl BUILD / COMPILE. Wenn es sich fehlerfrei kompilieren lässt, wählen Sie BUILD / EXECUTE, um das Programm auszuführen. Das Bearbeitungsfenster verschwindet und am unteren Rand des Bildschirms erscheint ein Ausgabefenster. Es zeigt die Ausgaben des Programms an (siehe Abbildung 2.4).

Um nach Beendigung des Programms zum Bearbeitungsfenster zurückzukehren, klicken Sie auf die Registerkarte EDITING am oberen Bildschirmrand.

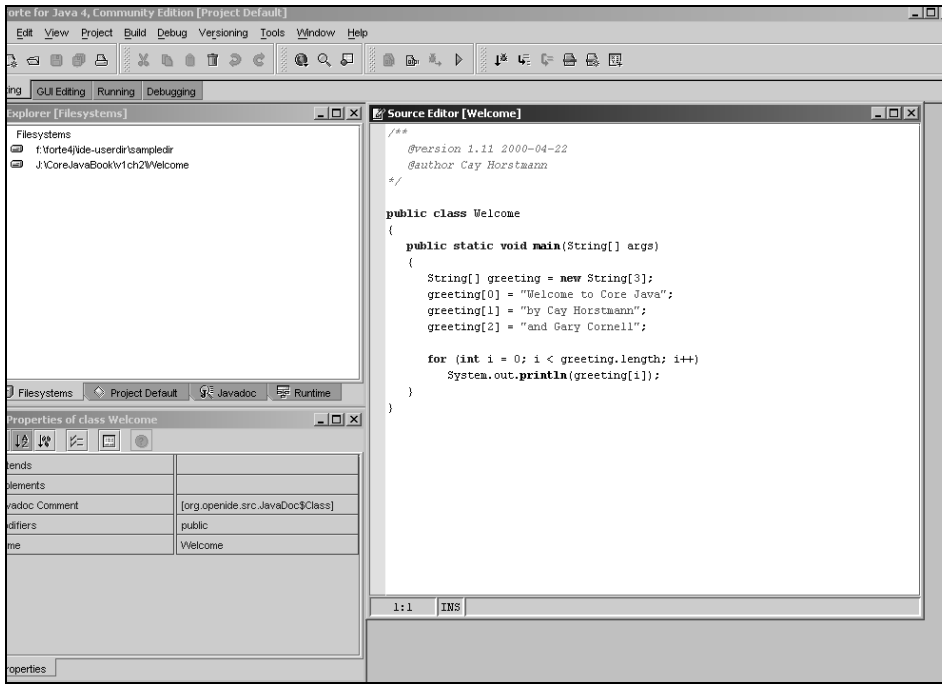

Abbildung 2.3: Das Bearbeitungsfenster von Sun ONE Studio

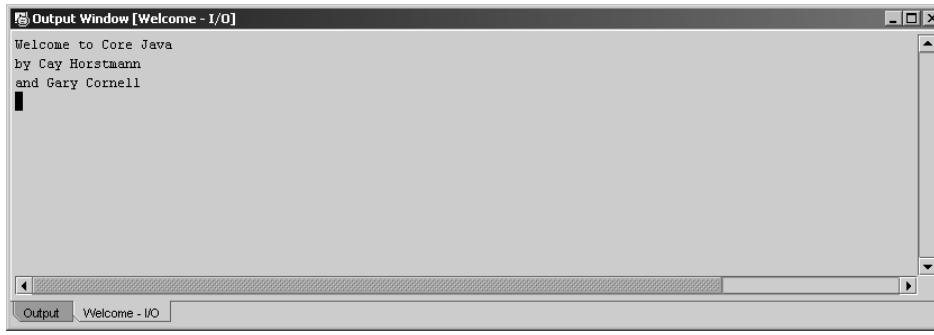

Abbildung 2.4: Das Ausgabefenster von Sun ONE Studio

2.4.1 Kompilierfehler aufspüren

Vermutlich hat unser Programm keine Tippfehler oder Bugs. (Schließlich ist es auch nur wenige Zeilen lang.) Nehmen wir einfach als Beispiel an, dass sich versehentlich ein Tippfehler (oder sogar ein Programmfehler) in den Code eingeschlichen hat. Probieren Sie es aus – verstümmeln Sie die Datei, indem Sie zum Beispiel die Großschreibung von `String` wie folgt ändern:

```
string[] greeting = new String[3];
```

Starten Sie nun den Compiler erneut – Sie erhalten Fehlermeldungen (siehe Abbildung 2.5). Die erste bemängelt einen unbekannten Typ `string`. Klicken Sie einfach auf die Fehlermeldung. Der Cursor geht zur zugehörigen Zeile im Bearbeitungsfenster und Sie können den Fehler korrigieren. Auf diese Weise lassen sich Fehler schnell beseitigen.

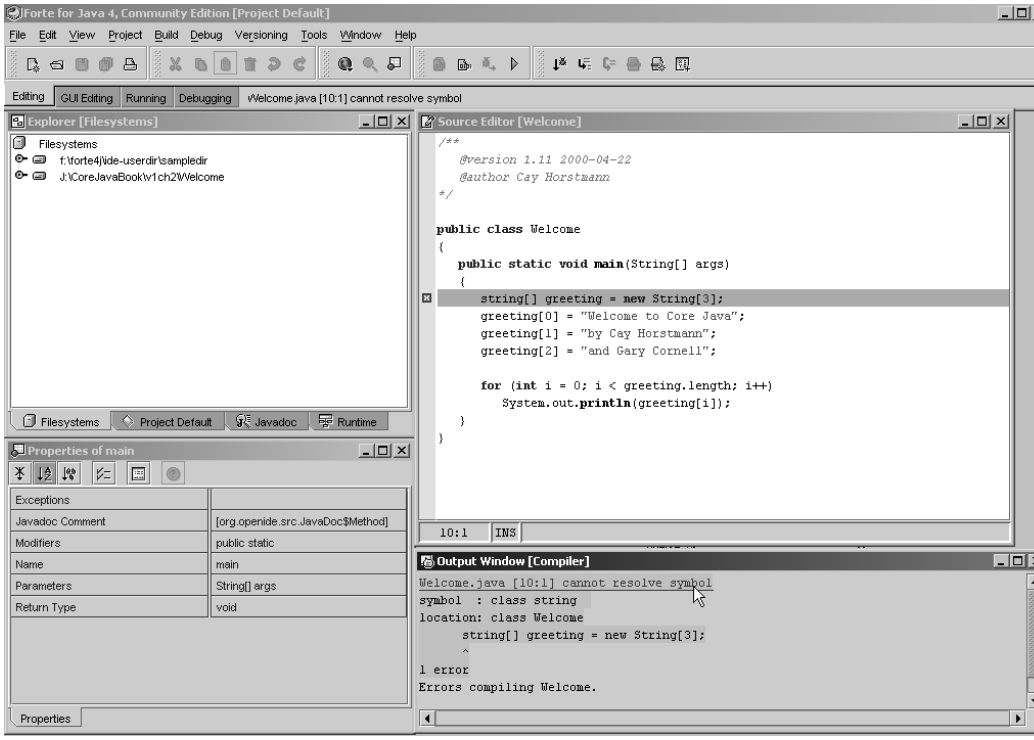

Abbildung 2.5: Fehlermeldungen in Sun ONE Studio

Um ein neues Programm mit Sun ONE Studio zu beginnen, wählen Sie den Menübefehl FILE / NEW FROM TEMPLATE. Im daraufhin erscheinenden Dialogfeld erweitern Sie den Eintrag `Classes`, indem Sie auf das Türgriffsymbol klicken. Markieren Sie `Empty` (siehe Abbildung 2.6) und klicken Sie dann auf die Schaltfläche NEXT.

Es erscheint die Frage, ob Sie diese Datei in das aktuelle Projekt aufnehmen möchten. Solange Sie noch nicht wirklich mit Projekten arbeiten, spielt es keine Rolle, ob Sie Ja (YES) oder Nein (NO) wählen. Jetzt sind Sie bereit, um die neue Datei zu bearbeiten, zu kompilieren und auszuführen.

Auf den Sun ONE Studio-Debugger geht Kapitel 11 näher ein.

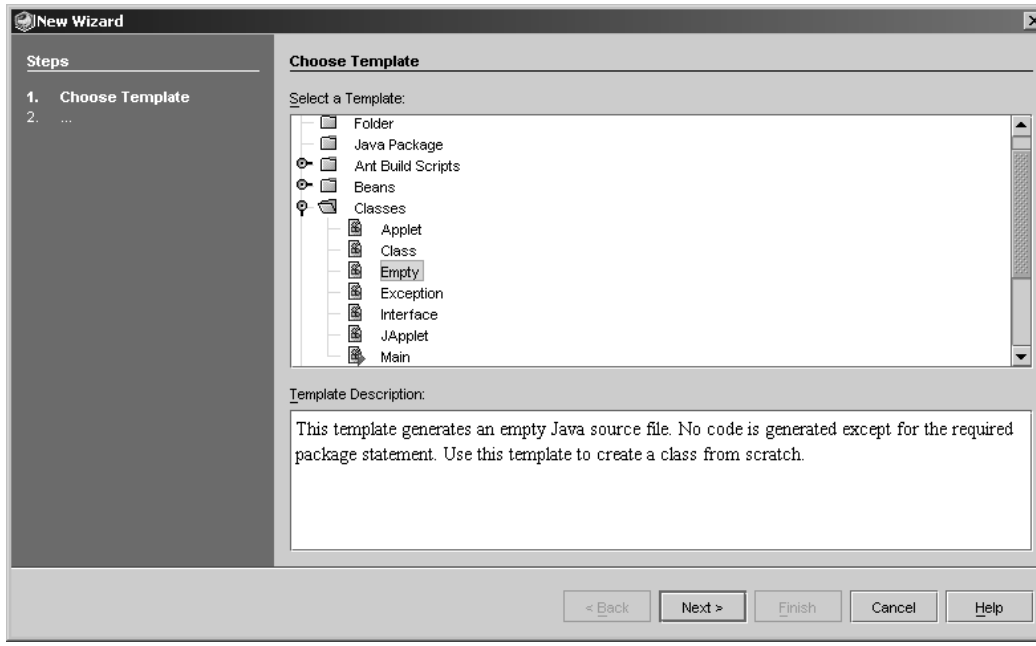

Abbildung 2.6: Ein neues Programm in Sun ONE Studio beginnen

2.5 Programme aus einem Editor heraus kompilieren und ausführen

Eine integrierte Entwicklungsumgebung wie Sun ONE Studio bietet reichlich Komfort, hat aber auch einige Nachteile. Insbesondere bei einfachen Programmen, die sich nicht über mehrere Quelldateien erstrecken, scheint es wie mit Kanonen auf Spatzen geschossen, wenn man eine Umgebung mit einer relativ langen Startphase und einer Menge überflüssiger Ausstattungsmerkmale einsetzt. Außerdem haben sich viele Programmierer an ihren bevorzugten Editor gewöhnt und lassen sich nur schwer zum Einsatz der Editoren bewegen, die Teil der integrierten Entwicklungsumgebungen sind. Zum Glück bieten viele Editoren die Möglichkeit, den Java-Compiler und -Interpreter zu starten sowie Fehlermeldungen und Programmausgaben zu übernehmen. Dieser Abschnitt stellt zwei typische Vertreter derartiger Editoren vor: Emacs und TextPad.

Hinweis

GNU Emacs ist über *http://www.gnu.org/software/emacs/* erhältlich. Für den Windows-Port von GNU Emacs gehen Sie auf die Seite *http://www.gnu.org/software/emacs/windows/ntemacs.html*. Stellen Sie sicher, dass das JDEE-Paket installiert ist, wenn Sie Emacs für die Java-Programmierung einsetzen. Das Paket können Sie von *http://jdee.sunsite.dk* herunterladen.

Abbildung 2.7 zeigt zum Beispiel wie der Emacs-Editor ein Java-Programm kompiliert. (Wählen Sie JDE / COMPILE aus dem Menü, um den Compiler zu starten.)

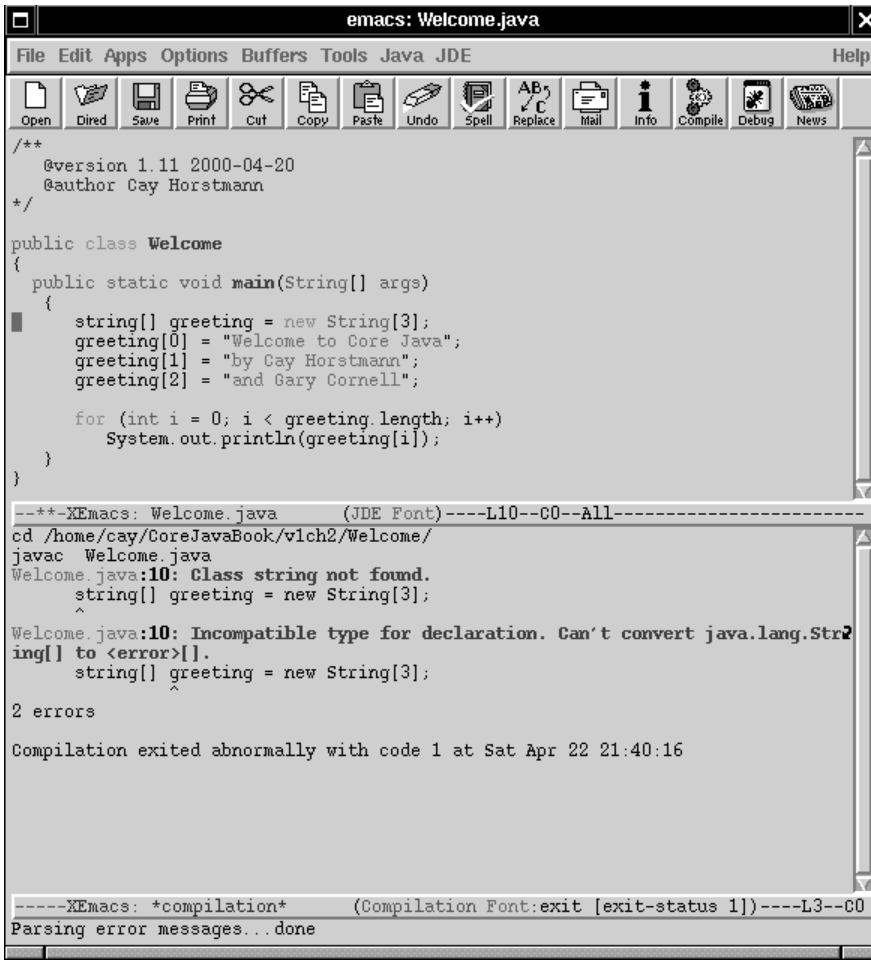

Abbildung 2.7: Ein Programm mit Emacs kompilieren

Die Fehlermeldungen erscheinen im unteren Teil des Bildschirms. Wenn Sie den Cursor auf eine Fehlermeldung setzen und die ↵-Taste drücken, wird der Cursor auf die entsprechende Quelltextzeile verschoben.

Nachdem Sie alle Fehler beseitigt haben, können Sie das Programm über JDE / RUN APP aus dem Menü starten. Die Ausgabe erscheint innerhalb eines Editorfensters (siehe Abbildung 2.8).

Abbildung 2.8: Ein Programm aus Emacs heraus ausführen

Emacs ist ein ausgezeichneter Editor, der kostenlos für UNIX, Linux, Windows und Mac OS X verfügbar ist. Allerdings erscheint vielen Programmierern die Lernkurve als zu steil. Für diese Programmierer empfiehlt sich TextPad. Im Gegensatz zu Emacs hält sich TextPad an die Standardkonventionen von Windows. TextPad finden Sie auf der Website *http://www.textpad.com*. Beachten Sie, dass TextPad Shareware ist.

Hinweis

Auf der Website *http://www.textpad.com* ist TextPad in mehreren Sprachen verfügbar. Die Übersetzung dieses Buches bezieht sich auf die deutsche Version von TextPad.

Um ein Programm zu kompilieren, wählen Sie den Menübefehl EXTRAS / JAVA KOMPILIEREN oder drücken die Tastenkombination Strg + 1.

Hinweis

Falls kein derartiger Menübefehl vorhanden ist, wählen Sie KONFIGURATION / EINSTELLUNGEN und erweitern den Eintrag EXTRAS in der Struktur auf der linken Seite. Auf der rechten Seite klicken Sie auf die Schaltfläche HINZUFÜGEN. Daraufhin öffnet sich eine Liste, die den Eintrag *JDK-Befehle* enthält. Markieren Sie diesen Eintrag und klicken Sie dann auf OK. Die JDK-Befehle werden nun in das Menü EXTRAS aufgenommen.

Kompilierungsfehler zeigt TextPad in einem separaten Fenster an (siehe Abbildung 2.9).

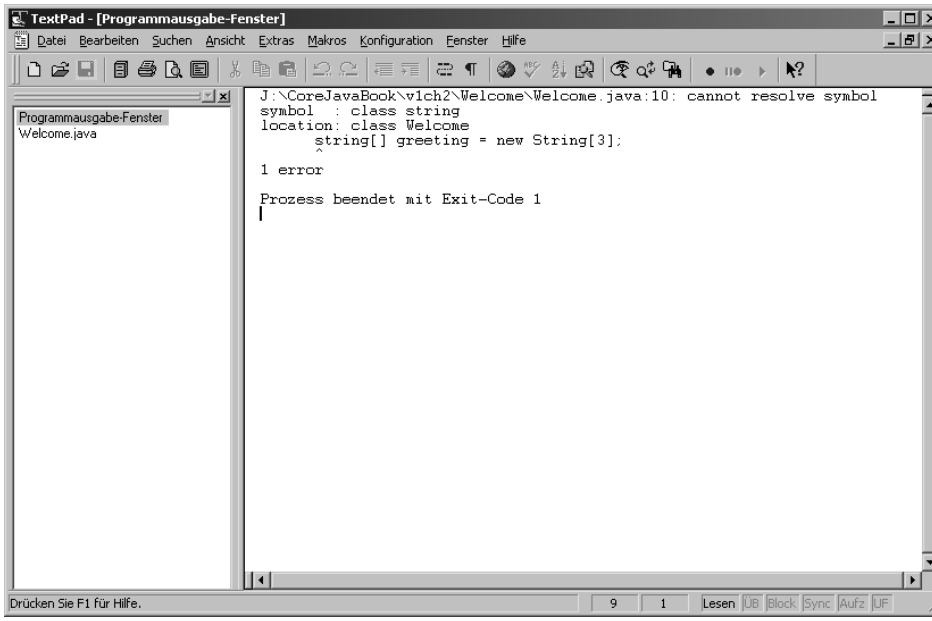

Abbildung 2.9: Kompilierungsfehler in TextPad aufsuchen

Verschieben Sie den Cursor auf die erste Zeile einer Fehlermeldung und drücken Sie ↵, um zur entsprechenden Stelle in der Datei zu gelangen. Mit dem Befehl SUCHEN / ZUM NÄCHSTEN SPRINGEN (oder der Taste F4) können Sie sich durch die restlichen Fehlermeldungen bewegen.

Um ein Programm auszuführen, wählen Sie den Menübefehl EXTRAS / JAVA-PROGRAMM STARTEN oder drücken die Tastenkombination Strg + 2. Das Programm startet in einem separaten Shell-Fenster. Abbildung 2.10 zeigt ein Java-Programm, das über TextPad gestartet wurde.

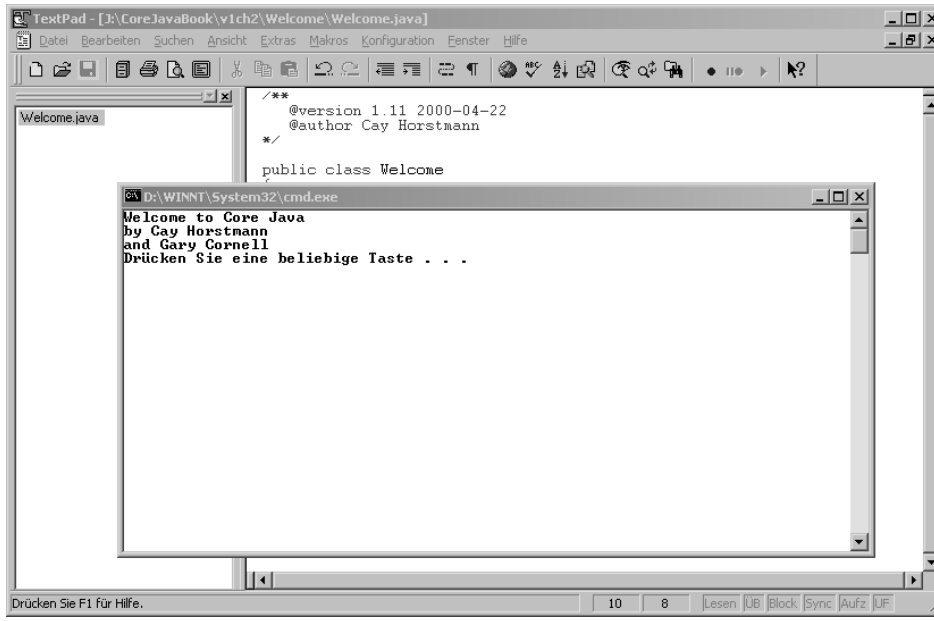

Abbildung 2.10: Ein Java-Programm aus TextPad heraus ausführen

Wenn das Programm beendet ist, müssen Sie eine Taste drücken, um fortzusetzen und das Shell-Fenster zu schließen.

2.6 Grafikanwendungen

Das Programm `Welcome` war nicht gerade berauschend. Als Nächstes wollen wir eine Grafikanwendung starten. Es handelt sich um einen sehr einfachen Betrachter für GIF-Dateien. Er lädt eine GIF-Datei und zeigt sie an. Auch hier kompilieren und starten wir das Programm zunächst von der Befehlszeile.

1. Öffnen Sie ein Shell-Fenster.
2. Gehen Sie ins Verzeichnis `/CoreJavaBook/v1ch2/ImageViewer`.
3. Tippen Sie folgende Befehle ein:

```
javac ImageViewer.java
java ImageViewer
```

Daraufhin öffnet sich ein neues Programmfenster mit der `ImageViewer`-Anwendung (siehe Abbildung 2.11).

Wählen Sie nun DATEI / ÖFFNEN und suchen Sie sich eine GIF-Datei, die Sie öffnen können. (Im selben Verzeichnis haben wir Beispieldateien bereitgestellt.)

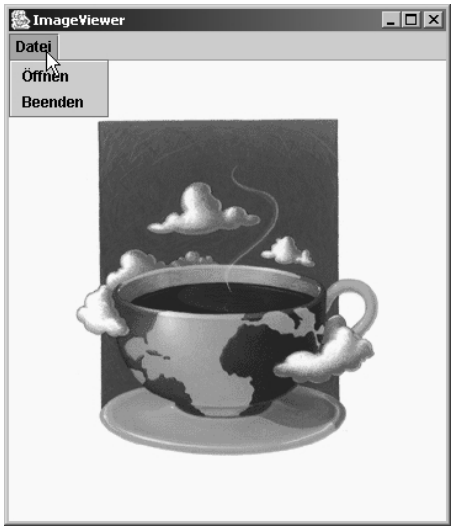

Abbildung 2.11: Die Anwendung ImageViewer ausführen

Um das Programm zu beenden, klicken Sie auf das SCHLIEßEN-Feld in der Titelleiste oder öffnen das Systemmenü und wählen den Befehl SCHLIEßEN. (In TextPad oder einer Entwicklungsumgebung kompilieren und starten Sie das Programm ebenfalls wie oben beschrieben. Zum Beispiel wählen Sie bei Emacs JDE / COMPILE und danach JDE / RUN APP.)

Wir hoffen, dass Sie dieses Programm interessant und nützlich finden. Sehen Sie sich kurz den Quellcode an. Das Programm ist wesentlich länger als das erste, dennoch aber nicht übermäßig kompliziert, wenn man in Betracht zieht, welcher Aufwand in C oder C++ für eine ähnliche Anwendung erforderlich wäre. In Visual Basic dagegen ist es recht einfach, ein derartiges Programm zu schreiben, oder besser gesagt, per Drag&Drop zu erstellen – man muss nur ein paar Codezeilen hinzufügen, damit es funktioniert. Das JDK hat keine visuellen Editoren für die Benutzeroberfläche, sodass man für alles und jedes Code schreiben muss, wie es das Beispiel 2.2 zeigt. In den Kapiteln 7 bis 9 erfahren Sie, wie man derartige Grafikprogramme schreibt.

Warnung

Wenn Sie dieses Programm mit einer Version des Java SDK vor 1.4 ausführen, erhalten Sie einen Compilerfehler auf der Zeile

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

In diesem Fall kommentieren Sie die Zeile aus und kompilieren das Programm neu. Dann wird das Programm aber nicht beendet, wenn Sie den Rahmen schließen. Wählen Sie stattdessen den Menübefehl DATEI / BEENDEN. Kapitel 7 gibt weitere Informationen zu diesem Problem.

Beispiel 2.2: ImageViewer.java

```
1:  import java.awt.*;
2:  import java.awt.event.*;
3:  import java.io.*;
4:  import javax.swing.*;
5:
6:  /**
7:     Ein Programm zur Anzeige von Bildern.
8:  */
9:  public class ImageViewer
10: {
11:    public static void main(String[] args)
12:    {
13:       JFrame frame = new ImageViewerFrame();
14:       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15:       frame.show();
16:    }
17: }
18:
19: /**
20:    Ein Rahmen mit einem Bezeichnungsfeld, um ein Bild anzuzeigen.
21: */
22: class ImageViewerFrame extends JFrame
23: {
24:    public ImageViewerFrame()
25:    {
26:       setTitle("ImageViewer");
27:       setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
28:
29:       // Ein Bezeichnungsfeld verwenden, um die Bilder anzuzeigen
30:       label = new JLabel();
31:       Container contentPane = getContentPane();
32:       contentPane.add(label);
33:
34:       // Dateiauswahl einrichten
35:       chooser = new JFileChooser();
36:       chooser.setCurrentDirectory(new File("."));
37:
38:       // Menüleiste einrichten
39:       JMenuBar menuBar = new JMenuBar();
40:       setJMenuBar(menuBar);
41:
42:       JMenu menu = new JMenu("Datei");
43:       menuBar.add(menu);
44:
45:       JMenuItem openItem = new JMenuItem("Öffnen");
46:       menu.add(openItem);
47:       openItem.addActionListener(new
48:          ActionListener()
```

```
49:          {
50:             public void actionPerformed(ActionEvent evt)
51:             {
52:                // Dateiauswahldialogfeld anzeigen
53:                int r = chooser.showOpenDialog(null);
54:
55:                // Wenn Datei ausgewählt, als Symbol des Bezeichnungsfeldes festlegen
56:                if(r == JFileChooser.APPROVE_OPTION)
57:                {
58:                   String name
59:                       = chooser.getSelectedFile().getPath();
60:                   label.setIcon(new ImageIcon(name));
61:                }
62:             }
63:          });
64:
65:       JMenuItem exitItem = new JMenuItem("Beenden");
66:       menu.add(exitItem);
67:       exitItem.addActionListener(new
68:          ActionListener()
69:          {
70:             public void actionPerformed(ActionEvent event)
71:             {
72:                System.exit(0);
73:             }
74:          });
75:    }
76:
77:    private JLabel label;
78:    private JFileChooser chooser;
79:    private static final int DEFAULT_WIDTH = 300;
80:    private static final int DEFAULT_HEIGHT = 400;
81: }
```

2.7 Applets

Die beiden ersten in diesem Buch präsentierten Programme sind Java-*Anwendungen*, eigenständige Programme wie jedes »richtige« Programm. Wie bereits im letzten Kapitel erwähnt, resultiert aber die Java-Euphorie aus der Möglichkeit, Applets in einem Webbrowser ausführen zu können. Wir werden zeigen, wie man ein Applet erstellt und von der Befehlszeile ausführt. Schließlich laden wir das Applet in den Applet-Viewer, der zum SDK gehört, und zeigen es in einem Webbrowser an.

Gehen Sie zuerst in das Verzeichnis `/CoreJavaBook/v1ch2/WelcomeApplet` und geben Sie dann die folgenden Befehle ein:

```
javac WelcomeApplet.java
appletviewer WelcomeApplet.html
```

Abbildung 2.12 zeigt, was im Fenster des Applet-Viewers erscheint.

Abbildung 2.12: Das Applet WelcomeApplet im Applet-Viewer

Der erste Befehl ist der mittlerweile bekannte Befehl zum Aufruf des Java-Compilers. Dieser kompiliert die Quelle `WelcomeApplet.java` in die Bytecode-Datei `WelcomeApplet.class`.

Dieses Mal starten wir aber nicht den Java-Interpreter, sondern verwenden das Programm `appletviewer`. Dieses Programm ist ein spezielles Werkzeug, das zum SDK gehört und das schnelle Testen eines Applets gestattet. Dem Programm ist statt dem Namen einer Java-Klassendatei eine HTML-Datei zu übergeben. Der Inhalt der Datei `WelcomeApplet.html` ist in Beispiel 2.3 wiedergegeben.

Beispiel 2.3: WelcomeApplet.html

```
1:  <html>
2:  <title>WelcomeApplet</title>
3:  <body>
4:  <hr>
5:  <p>
6:  This applet is from the book
7:  <a href="http://www.horstmann.com/corejava.html">
8:  Core Java</a> by <em>Cay Horstmann</em> and
9:  <em>Gary Cornell</em>,published by Sun Microsystems Press.
10: </p>
11: <applet code="WelcomeApplet.class" width="400" height="200">
12: <param name="greeting" value ="Welcome to Core Java!"/>
13: </applet>
14: <hr>
15: <p><a href="WelcomeApplet.java">The source.</a></p>
16: </body>
17: </html>
```

Wenn Sie mit HTML vertraut sind, erkennen Sie die Standard-HTML-Anweisungen und das `applet`-Tag, das den Applet-Viewer anweist, das Applet zu laden, dessen Code in `WelcomeApplet.class` gespeichert ist. Der Applet-Viewer ignoriert alle anderen Tags in dieser Datei.

Die anderen HTML-Tags erscheinen, wenn Sie die HTML-Datei in einem Java 2-fähigen Browser anzeigen. Allerdings ist die Browser-Situation etwas undurchsichtig.

- Netscape ab Version 6, Mozilla und Opera unterstützen Java 2 sowohl unter Windows als auch unter Linux, auch wenn man die Java-Unterstützung nicht unbedingt herunterladen und installieren muss. Um mit Applets zu experimentieren, laden Sie einfach die neueste Version herunter und stellen sicher, dass Java aktiviert ist.
- Manche Versionen von Internet Explorer bieten überhaupt keine Unterstützung für Java; andere unterstützen nur die veraltete Microsoft Java Virtual Machine. Wenn Sie Internet Explorer unter Windows ausführen, gehen Sie auf *http://java.sun.com/getjava* und laden das Java Plug-In herunter. Es bindet die Java 2-Funktionalität in Internet Explorer ein.
- Wenn Sie OS X auf einem Macintosh ausführen, ist Internet Explorer mit der Macintosh Java-Implementierung integriert, die momentan J2SE in der Version 1.3 unterstützt. OS 9 unterstützt nur die veraltete Version 1.1.
- Unter Netscape 4 können Sie die Java 2-Unterstützung mit dem Sun Java Plug-In aktivieren, wobei aber der Browser weiterhin auf seine veraltete Java Virtual Machine für Applets, die mit dem `applet`-Tag geladen werden, zurückgreift. Deshalb müssen Sie die HTML-Datei neu schreiben und eine ziemlich umständliche Konstruktion mit dem `embed`-Tag verwenden. Diese Lösung wird nicht mehr empfohlen.

Vorausgesetzt, dass Sie einen Browser mit Java 2-Unterstützung haben, können Sie versuchen, das Applet in den Browser zu laden:

1. Starten Sie Ihren Browser.
2. Wählen Sie DATEI / ÖFFNEN (oder einen äquivalenten Befehl).
3. Gehen Sie in das Verzeichnis `/CoreJavaBook/v1ch2/WelcomeApplet`.

Im Dateidialog sollte nun die Datei `WelcomeApplet.html` zu sehen sein. Öffnen Sie diese Datei. Der Browser lädt nun das Applet einschließlich dem umgebenden Text. Abbildung 2.13 gibt das Ganze wieder.

Wie Sie sehen, ist diese Anwendung tatsächlich »lebendig« und bereit, mit dem Internet zu interagieren. Klicken Sie auf die Schaltfläche CAY HORSTMANN. Das Applet veranlasst nun, dass der Browser die Webseite von Cay anzeigt. Klicken Sie auf die Schaltfläche GARY CORNELL. Das Applet bewirkt, dass der Browser ein Mail-Fenster öffnet, in dem die E-Mailadresse von Gary bereits eingetragen ist.

Beachten Sie, dass beide Schaltflächen nicht im Applet-Viewer funktionieren. Der Applet-Viewer verfügt nicht über die Fähigkeiten, Mail zu senden oder eine Webseite anzuzeigen, und ignoriert einfach diesbezügliche Anforderungen. Mit dem Applet-Viewer lassen sich in erster Linie Applets in der Isolierung testen; man muss aber Applets in einen Browser bringen, um sich davon zu überzeugen, ob sie mit dem Browser und dem Internet zusammenarbeiten.

Tipp

Applets können Sie auch aus Ihrem Editor oder einer integrierten Entwicklungsumgebung testen. In Emacs führen Sie das Applet über den Menübefehl JDE / RUN APPLET aus. In TextPad wählen Sie EXTRAS / JAVA-APPLET STARTEN oder drücken die Tastenkombination Strg + 3. Daraufhin erscheint ein Dialogfeld, das alle HTML-Dateien im aktuellen Verzeichnis auflistet. Wenn Sie ESC drücken, erstellt TextPad automatisch eine HTML-Minimaldatei. In Sun ONE Studio laden Sie einfach die HTML-Seite mit den Applet-Tags. Sun ONE Studio enthält einen einfachen Browser, der das laufende Applet in einer Webseite zeigt. Alternativ können Sie mit der rechten Maustaste auf die Quelldatei klicken und den Wert der Eigenschaft *Executor* in der Registerkarte EXECUTION auf »Applet Execution« setzen.

Abbildung 2.13: Das Applet WelcomeApplet in einem Browser ausführen

Abschließend zeigt Beispiel 2.4 den Code für das `Welcome`-Applet. Momentan sollten Sie lediglich einen Blick darauf werfen. In Kapitel 10 erfahren Sie mehr darüber, wie man Applets schreibt.

In diesem Kapitel haben Sie gelernt, wie Sie Java-Programme kompilieren und ausführen. Damit sind Sie bereit, beginnend mit Kapitel 3 in die Grundlagen der Sprache Java einzusteigen.

Beispiel 2.4: WelcomeApplet.java

```
1:  import javax.swing.*;
2:  import java.awt.*;
3:  import java.awt.event.*;
4:  import java.net.*;
5:
6:  public class WelcomeApplet extends JApplet
7:  {
8:     public void init()
9:     {
10:       Container contentPane = getContentPane();
```

```
11:       contentPane.setLayout(new BorderLayout());
12:
13:       JLabel label = new JLabel(getParameter("greeting"),
14:          SwingConstants.CENTER);
15:       label.setFont(new Font("Serif", Font.BOLD, 18));
16:       contentPane.add(label, BorderLayout.CENTER);
17:
18:       JPanel panel = new JPanel();
19:
20:       JButton cayButton = new JButton("Cay Horstmann");
21:       cayButton.addActionListener(makeURLActionListener(
22:          "http://www.horstmann.com"));
23:       panel.add(cayButton);
24:
25:       JButton garyButton = new JButton("Gary Cornell");
26:       garyButton.addActionListener(makeURLActionListener(
27:          "mailto:gary@thecornells.com"));
28:       panel.add(garyButton);
29:
30:       contentPane.add(panel, BorderLayout.SOUTH);
31:    }
32:
33:    private ActionListener makeURLActionListener(final String u)
34:    {
35:       return new
36:          ActionListener()
37:          {
38:             public void actionPerformed(ActionEvent event)
39:             {
40:                try
41:                {
42:                   getAppletContext().showDocument(new URL(u));
43:                }
44:                catch(MalformedURLException e)
45:                {
46:                   e.printStackTrace();
47:                }
48:             }
49:          };
50:    }
51: }
```