PEARSON Education

Markt+Technik

easy

Java 5

Aufregend programmieren

DIRK LOUIS PETER MÜLLER

Markt+Technik

→leicht →klar →sofort

Kapitel 3

Wie erstellt man eigene Programme?

In diesem Kapitel geht es darum, dass Sie Ihre Programmierwerkzeuge kennen lernen. Wir beginnen mit der Installation des Java-Entwicklungssystems (JDK) und der Einrichtung Ihres Rechners für die Programmentwicklung mit dem JDK. Anschließend werden wir unser erstes Programm erstellen und sehen, wie sich die Programmdateien sinnvoll auf der Festplatte organisieren lassen.

Welche Entwicklungsumgebung darf es sein?

Unter einer Entwicklungsumgebung versteht man die Sammlung an Werkzeugen (Tools), die ein Programmierer zum Erstellen und Testen seiner Programme verwendet. Dies sind im Wesentlichen die folgenden Komponenten:

- ein Editor: hiermit wird der Quellcode geschrieben,
- ein Compiler: er übersetzt (kompiliert) den Quellcode in den bereits erwähnten Bytecode,
- ein Interpreter: er dient zur Ausführung eines fertigen Java-Programms,
- ein Debugger: ein besonderer Interpreter, der Hilfestellung bei der Fehlersuche in größeren Programmen gibt.

Ein erfahrener Programmierer wird in der Regel eine so genannte integrierte Entwicklungsumgebung (IDE = Integrated Development Environment) verwenden. Dies ist ein umfangreiches Programm, das alle oben erwähnten Komponenten und noch vieles mehr enthält. Bekannte Vertreter sind Borland JBuilder oder Eclipse, die für den Privatgebrauch auch als kostenlose Lizenzen erhältlich sind. Eine solche IDE ist für den erfahrenen Anwender sehr praktisch, da sie viel Arbeit abnehmen kann. Allerdings hat sie für einen absoluten Programmiereinsteiger gravierende Nachteile:

- Viele Arbeitsschritte während der Programmierung laufen unsichtbar ab, weil die IDE dies automatisch macht. Beim Erlernen einer Programmiersprache sollten Sie aber im Detail und bewusst nachvollziehen können, was warum passiert, um so die Sprache und den Programmierprozess zu verstehen.
- Eine IDE hat typischerweise mehrere interne Fenster und hundert oder mehr Menüeinträge, zudem eine so genannte Projektverwaltung für die Verwaltung verschiedener Programmierprojekte. Allein die korrekte Bedienung und Beherrschung all dieser Feature erfordert eine erhebliche Einarbeitung, die ein Anfänger besser auf das Erlernen der eigentlichen Programmsprache verwenden sollte.

Nach unserer Erfahrung ist es daher besser, als Einsteiger in die Programmierung zunächst ohne IDE zu arbeiten und alles von Hand zu Fuß zu erledigen, d.h. einen einfachen Editor (z.B. Notepad von Windows) und das Java-Entwicklungspaket JDK von Sun zu verwenden.

Installation des JDK

Alles, was wir – abgesehen von einem Editor zum Aufsetzen der Programmquelltexte – zur Programmerstellung mit Java benötigen, ist in einem Entwicklungspaket namens Java Development Kit, kurz JDK, zusammengestellt. Version 1.5 des JDK finden Sie auf der Buch-CD. Es kann im Übrigen aber auch kostenlos von der Java-Website `http://java.sun.com` heruntergeladen werden[1].

Datei	Beschreibung
CD:\Buchdaten\JDK\jdk-1_5_0_04-windows-i586-p.exe	Setup-Datei für Windows
CD:\Buchdaten\JDK\jdk-1_5_0_04-linux-i586.bin	Setup-Datei für Linux
CD:\Buchdaten\JDK\jdk-1_5_0-doc.zip	ZIP-Datei mit zusätzlicher Java-Referenz im HTML-Format

Tabelle 3.1: Inhalt des JDK-Verzeichnisses auf der Buch-CD

Zur Installation des JDK (Java Development Kit) schließen Sie bitte zunächst alle laufenden Anwendungen. Installationsprogramme (auch Setup-Programme genannt) sind unter Windows häufig etwas empfindlich und können durch andere Anwendungen gestört werden.

Das eigentliche Installieren ist für Microsoft Windows sehr einfach.

Hinweis

Unter Linux bedarf es zur Installation des JDK und zur Erstellung eigener Programme grundsätzlich der gleichen Schritte wie unter Windows – allerdings meist unter Verwendung anderer Dateien, Befehle etc. Achten Sie daher in den Schritt-für-Schritt-Anweisungen besonders auf die Linux-spezifischen Hinweise.

1 *Aufbau und Namen auf der Sun-Website sind ständigen Änderungen unterworfen. Zum Zeitpunkt der Drucklegung dieses Buches klicken Sie rechts im Bereich Popular Downloads auf den Link J2SE 5.0. Auf der Download-Seite selbst laden Sie dann das JDK 5.0 Update herunter. (Der Zusatz »Update« bedeutet in diesem Falle nur, dass es sich um eine überarbeitete Version des JDK handelt.)*

1 Legen Sie die Buch-CD in Ihr CD-ROM-Laufwerk ein.

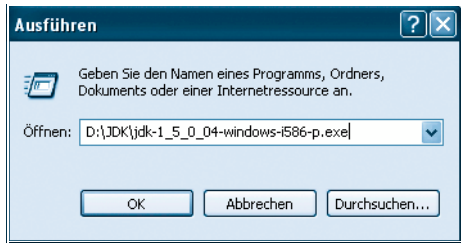

Abbildung 3.1: Setup-Programm für das JDK aufrufen

2 Führen Sie das Setup-Programm aus.

Öffnen Sie hierzu das *Start*-Menü von Windows und wählen Sie den Menüeintrag *Ausführen*. Das Setup-Programm für die Windows-Version selbst heißt *jdk-1_5_0_04-windows-i586-p.exe* und befindet sich auf der CD-ROM im Verzeichnis *JDK*. Wenn Ihr CD-Laufwerk z.B. als Laufwerk *D:* eingerichtet ist, würden Sie im Dialogfenster *D:\JDK\jdk-1_5_0_04-windows-i586-p.exe* eingeben.

Wenn Ihnen das Eintippen zu mühselig ist, klicken Sie auf die *Durchsuchen*-Schaltfläche und wählen Sie das Setup-Programm in dem Dialogfenster aus, das daraufhin erscheint.

Zum Schluss wird durch Klicken auf den *OK*-Schalter das Installationsprogramm gestartet.

> **Hinweis**
>
> Das Setup-Programm für Linux heißt *jdk-1_5_0_04-linux-i586.bin*. Kopieren Sie die Datei in das Verzeichnis, unter dem das JDK installiert werden soll (beispielsweise */home/ihrname/java*). Um das Programm zu starten, öffnen Sie eine Konsole (auch Terminalfenster genannt), wechseln in das Verzeichnis, in dem das Setup-Programm steht, und tippen
>
> ```
> ./jdk-1_5_0_04-linux-i586.bin
> ```
>
> ein.

3 Lesen Sie den Lizenzvertrag.

Es erscheint zunächst der Lizenzvertrag. Wenn Sie das JDK installieren möchten, müssen Sie den Vertrag akzeptieren, indem Sie die Option *I accept the terms ...* auswählen und den *Next*-Schalter drücken.

Hinweis

Linux-Anwender browsen durch Drücken der ↵-Taste zum Ende des Vertragstextes und beantworten die Frage *Do you agree to the above license terms?* mit `yes`. (Einfach in der Konsole eingeben und abschicken.)

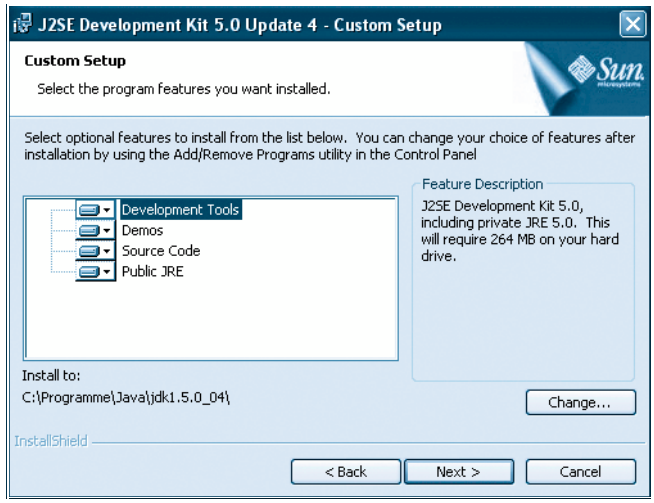

Abbildung 3.2: Das Setup-Programm sammelt die nötigen Informationen für die Installation.

4 Beantworten Sie die Fragen des Installationsprogramms.

In den nachfolgenden Fenstern übernehmen Sie am besten alle Vorgaben durch Anklicken des *Next*-Schalters. Es werden dann nacheinander das JDK und die JRE (Java-Laufzeitumgebung) installiert.

Wenn Sie knapp mit Festplattenspeicher sind (die vollständige Installation verschlingt etwa 500 Mbyte), können Sie im Dialogfeld *Custom Setup* die Optionen *Demos* und *Source Code* deaktivieren (spart ungefähr 90 Mbyte). Die *Public JRE* ist vor allem wichtig, wenn Sie in Ihren Browsern Java-Applets ausführen wollen (siehe *Kapitel 16, »Abschluss und Ausblick«*). Sollten Sie kein Interesse an der Applet-Entwicklung haben, können Sie auch diese Option deaktivieren.

Das Zielverzeichnis für die Installation wird im Dialogfeld *Custom Setup* links unten angezeigt. Zum Ändern drücken Sie auf die Schaltfläche *Change*.

Hinweis

Die Linux-Installation läuft vollautomatisch ab, ohne weitere Abfragen. Als Zielverzeichnis wird unter dem aktuellen Verzeichnis ein Unterverzeichnis *jdk1.5.0_04* angelegt.

Nach Abschluss der Installation ist Ihre Festplatte um etliche Mbyte ärmer, aber Sie um eine Java-Entwicklungsinstallation der Version 5 reicher.

Die JRE (Java-Laufzeitumgebung)

Das Wichtigste vorneweg: Das JDK ist für Leute, die selbst Java-Programme schreiben wollen; die JRE ist für Leute, die Java-Programme auf ihrem Rechner ausführen wollen.

Wie Sie bereits wissen, können Java-Programme nicht direkt, sondern nur mit Hilfe eines Interpreters ausgeführt werden. Neben dem Interpreter werden allerdings noch weitere Hilfsmittel benötigt, z.B. die Java-Standardbibliotheken oder Plugins, um Java-Applets in Browsern ausführen zu können. Das komplette Paket aller Hilfsmittel, die zum *Ausführen* von Java-Anwendungen und -Applets benötigt werden, nennt man *Java-Laufzeitumgebung* oder *JRE* (englisch für Java Runtime Environment).

Im JDK ist bereits eine private JRE-Installation enthalten, damit Sie alle Anwendungen und Applets, die Sie erstellen, auch testen und ausführen können.

Die Public JRE des JDK entspricht den JREs, die auf der Sun-Website zum kostenlosen Download angeboten werden. Sie sind für Anwender gedacht, die Java-Programme ausführen, jedoch nicht selbst schreiben wollen.

Das JDK auf der Festplatte

Wenn Sie nun mit Hilfe des Windows Explorers auf Ihrer Festplatte nachschauen, werden Sie ein neues Verzeichnis *c:\Programme\Java\jdk1.5.0_04* (bzw. das von Ihnen bei der Installation gewählte Verzeichnis) vorfinden, das wiederum mehrere Unterverzeichnisse enthält. Hier stehen alle notwendigen Dateien und Programme für die Programmerstellung mit Java.

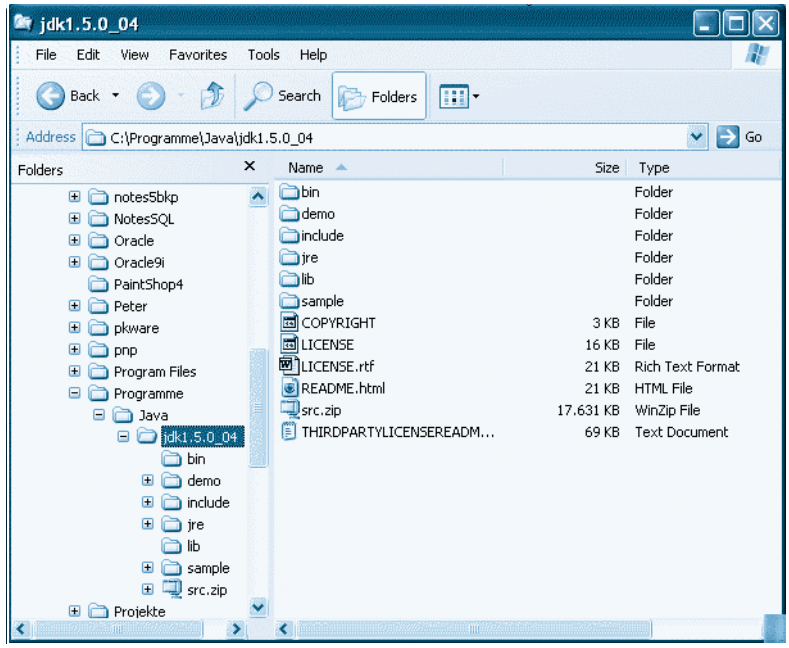

Abbildung 3.3: Das installierte JDK auf der Festplatte

Hinweis

Die Linux-Version wird in das Unterverzeichnis *jdk1.5.0_04* installiert.

Programmerstellung mit dem JDK

Das JDK ist nun installiert. Bevor wir jedoch richtig loslegen, sollten wir kurz prüfen, ob wirklich alle für die Programmierung benötigten Werkzeuge einsatzbereit sind. Im Einzelnen werden wir

- unser System so einrichten, dass wir die Java-Entwicklungswerkzeuge von jedem Verzeichnis aus aufrufen können,
- ein Verzeichnis für unsere Programme anlegen und
- einen passenden Editor auswählen.

Hinweis

Ausführlichere Informationen zur Einrichtung Ihres Rechners für die Arbeit mit dem JDK finden Sie auch auf den Support-Seiten zu diesem Buch unter `http://www.carpelibrum.de` sowie in den Installationshinweisen zum JDK unter `http://java.sun.com`.

5 Tragen Sie die Java-Entwicklungswerkzeuge in Ihren Systempfad (PATH) ein.

Für die Programmerstellung brauchen wir vor allem den Java-Compiler *javac* und den Java-Interpreter *java*, die bei der Installation des JDK in das JDK-Unterverzeichnis */bin* kopiert wurden.

Diese Programme werden von der Konsole aus aufgerufen. Unter Windows öffnen Sie die Konsole über den *Start*-Menüeintrag *Programme/Zubehör/Eingabeaufforderung*[2]. Unter Angabe des Programmnamens und des Pfades, der zu dem Programm führt, können Sie die Java-Entwicklungswerkzeuge von der Konsole aus aufrufen. Das Programm gibt dann eine Kurzreferenz zu seiner korrekten Verwendung aus (siehe Abbildung 3.4).

Was ist das?

Über die *Konsole* können Sie Betriebssystembefehle eingeben und Programme ausführen, die – wie die Java-Entwicklungswerkzeuge – über keine grafische Benutzeroberfläche verfügen und nicht mit dem Fenstermanager des Window-Systems zusammenarbeiten.

Mit der Konsole können Sie sich auch in der Verzeichnisstruktur des Rechners bewegen (den zugehörigen *cd*-Befehl werden wir später noch genauer vorstellen). Das Verzeichnis, in dem Sie sich gerade befinden, wird üblicherweise[3] am linken Rand der Eingabeaufforderung, des so genannten *Prompts*, angezeigt. (In welchem Verzeichnis Sie sich nach Aufruf der Konsole befinden, hängt vom Betriebssystem ab.)

2 *Je nach Windows-Version wird die Konsole auch über Start/Programme/Eingabeaufforderung oder Start/Programme/MS-DOS-Eingabeaufforderung aufgerufen.*

3 *Die meisten Betriebssysteme erlauben dem Anwender, den Prompt nach seinen Vorstellungen zu konfigurieren.*

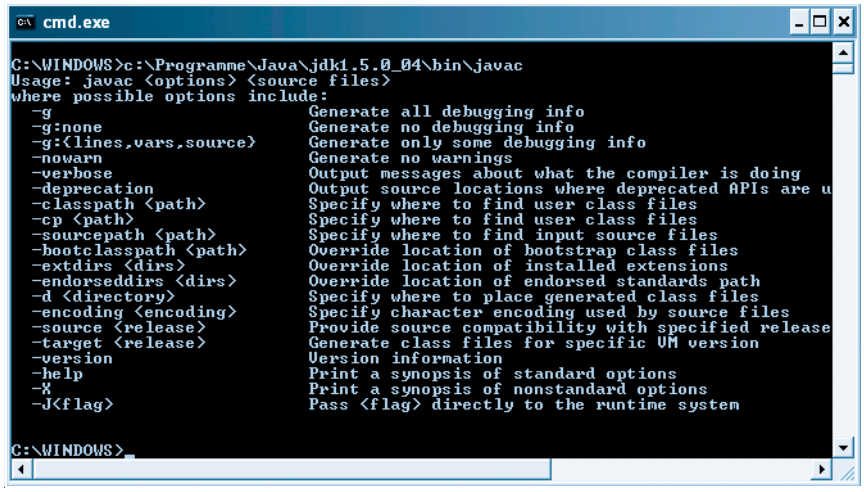

Abbildung 3.4: Aufruf des javac-Compilers von der Windows-Konsole aus

Das wiederholte Eintippen des Pfades ist allerdings recht lästig. Wir wollen daher unser System so einrichten, dass es die Java-Entwicklungswerkzeuge automatisch findet (und wir dann in der Konsole nur noch *javac* oder *java* eintippen müssen). Dazu müssen wir das JDK-Bin-Verzeichnis in den Suchpfad des Betriebssystems aufnehmen. Nehmen wir für die folgenden Erklärungen an, dass Sie das JDK in das Verzeichnis *c:\Programme\Java\jdk1.5.0_04* installiert haben.

Windows XP (Me/NT/2000...)

Unter Windows XP und den verwandten Windows-Versionen erfolgt das Anpassen der PATH-Variablen über die Systemsteuerung. Der Weg dorthin ist lang und von Betriebssystem zu Betriebssystem verschieden.

Unter Windows Me lautet er *Start*/*Programme*/*Zubehör*/*Systemtools*/*Systeminformationen*/Menü *Tools*/*System Konfiguration*/Register *Umgebung*.

Unter Windows 2000 und Windows XP rufen Sie über *Start* oder *Start*/*Einstellungen* die Systemsteuerung auf und gelangen über *System*[4], Register *Erweitert* zum Schalter *Umgebungsvariablen* und schließlich zum Ziel. Danach können Sie die Systemvariable `Path` auswählen, zum *Bearbeiten* laden und den Pfad zu den Java-Programmen anhängen (Semikolon an das Ende des PATH-Eintrags, dann den JDK-Bin-Pfad anhängen).

4 *Bei Windows XP und Verwendung der Kategorie-Ansicht müssen Sie erst noch Leistung und Wartung anklicken.*

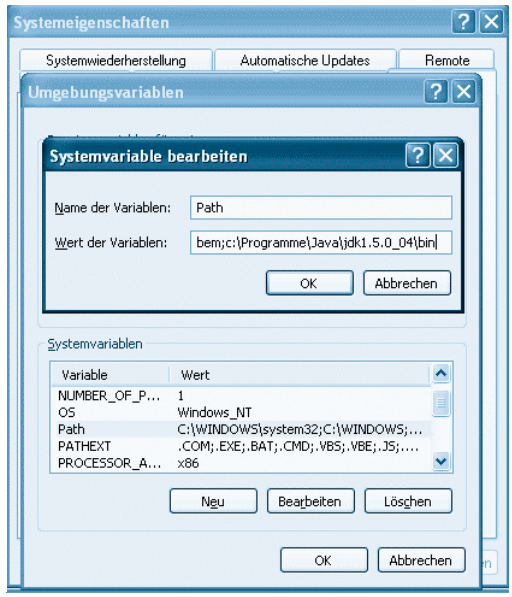

Abbildung 3.5: PATH-Umgebungsvariable unter Windows XP ergänzen

Windows 95/98

Bei Einsatz von Windows 95/98 laden Sie die Systemdatei *c:\autoexec.bat* in einen Texteditor (beispielsweise Notepad, *Start/Programme/Zubehör/Editor*) und suchen dort nach einem PATH-Eintrag. Hängen Sie diesem ein Semikolon und das JDK-Bin-Verzeichnis an.

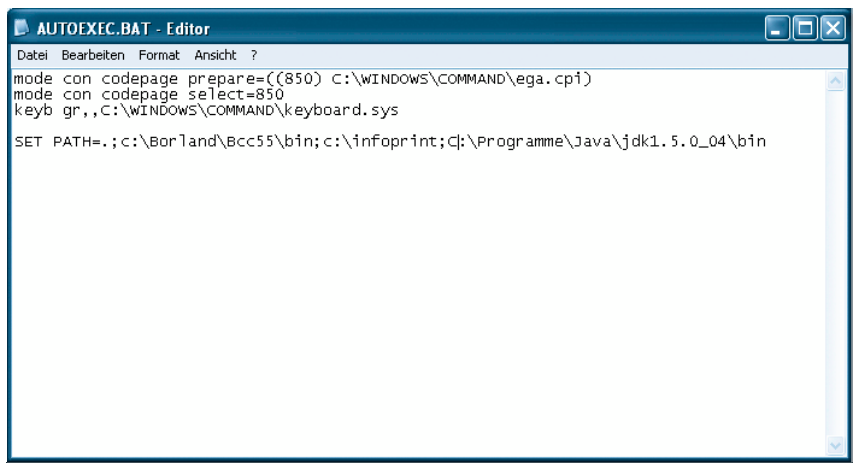

Abbildung 3.6: Systempfad anpassen für Windows 95/98

Was ist das?

Die Datei *autoexec.bat*, die im Verzeichnis *c:* stehen muss, ist eine Batch-Datei, die automatisch beim Hochfahren des Windows-Betriebssystems ausgeführt wird. In die *autoexec.bat* kann man bestimmte Befehle und Angaben zur Konfiguration des Systems schreiben.

Wenn Sie beispielsweise in Ihrer *autoexec.bat* folgenden Pfadeintrag finden:

```
SET PATH=.;c:\
```

und das JDK in das Verzeichnis *C:\Programme\Java\jdk1.5.0_04* installiert haben, so erweitern Sie den Pfadeintrag zu:

```
SET PATH=.;c:\;c:\Programme\Java\bin\jdk1.5.0_04\bin
```

Hinweis

Wenn Sie gar keine PATH-Angabe finden, dann fügen Sie eine neue PATH-Anweisung hinzu, beispielsweise

```
PATH=.;c:\Programme\Java\bin\jdk1.5.0_04\bin.
```

Eventuell müssen Sie sogar die *autoexec.bat*-Datei erst mal anlegen, falls in *c:* keine vorgegeben ist.

Starten Sie den Computer danach neu und testen Sie, ob Sie *javac* und *java* von beliebigen Verzeichnissen aus aufrufen können.

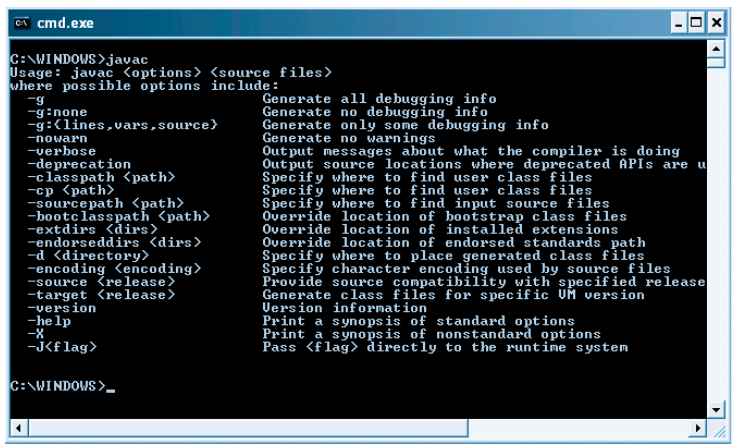

Abbildung 3.7: Aufruf des javac-Compilers ohne Pfadangabe

Linux

Unter Linux sollte eine manuelle Anpassung des Systempfades nicht nötig sein. Falls doch, gehen Sie analog vor. Suchen Sie die Pfadangabe `path` in der zuständigen ini-Datei (je nach Konfiguration *.login*, *.profile*, *.tcshrc* o.ä.) und fügen Sie das Java-Bin-Verzeichnis, beispielsweise */home/ihrname/jdk1.5.0_04/bin* in der nächsten Zeile nach der bisherigen Pfadangabe hinzu[5]:

```
set path = (/home/myname/jdk1.5.0_04/bin $path)
```

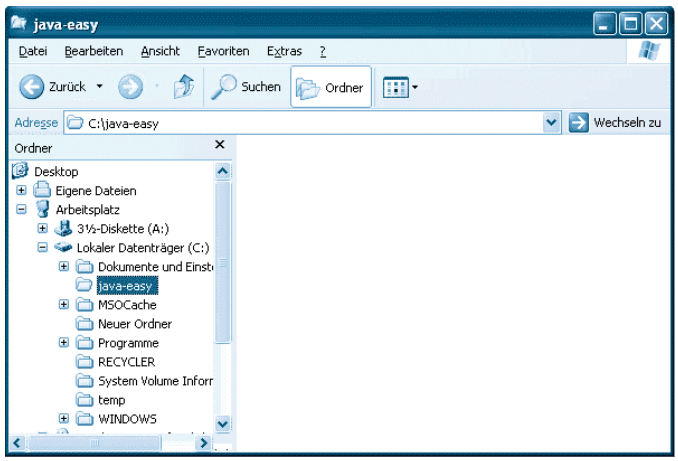

Abbildung 3.8: Projektverzeichnis für die Beispielprogramme anlegen

6 Legen Sie ein Verzeichnis für die Programme an.

Für das Erstellen eigener Programmen brauchen wir noch ein geeignetes Plätzchen, wo die zugehörigen Java-Dateien abgelegt werden sollen. Für den Rest des Buches gehen wir davon aus, dass es dazu ein Verzeichnis *c:\java-easy* gibt. Legen Sie daher bitte mit Hilfe des Windows Explorers (je nach Windows-Version *Start/Programme/Zubehör/Windows Explorer* oder *Start/Programme/Windows Explorer*) ein solches Verzeichnis an (Befehl *Datei/Neu/Ordner*).

> **Hinweis**
>
> Unter Linux können Sie das Verzeichnis mit einem passenden Dateimanager (unter KDE beispielsweise der Konqueror) anlegen oder Sie öffnen ein Konsolenfenster und verwenden den Shell-Befehl *mkdir verzeichnisname*.

5 *Beachten Sie bitte, dass das Setzen von Umgebungsvariablen unter Linux stark von der verwendeten Shell abhängt. Konsultieren Sie daher im Zweifelsfall am besten das Handbuch Ihrer Linux-Installation.*

7 Setzen Sie gegebenenfalls die CLASSPATH-Umgebungsvariable.

Eine weitere wichtige Umgebungsvariable für die Ausführung von Java-Programmen ist CLASSPATH. In dieser Variable wird festgelegt, wo der Java-Compiler und der Java-Interpreter nach ausführbaren Java-Class-Dateien suchen sollen.

Wenn die CLASSPATH-Variable auf Ihrem System noch *nicht* gesetzt ist, brauchen Sie selbst auch nichts weiter zu tun. Im anderen Fall erweitern Sie den CLASSPATH um das aktuelle Verzeichnis (repräsentiert durch den Punkt .) und das im vorangehenden Schritt angelegte *c:\java-easy*-Verzeichnis.

> **Hinweis**
>
> Das Setzen bzw. Anlegen der CLASSPATH-Variable erfolgt analog zu den oben beschriebenen Schritten für das Setzen der PATH-Umgebungsvariable. Weitere Informationen finden Sie auf den Webseiten zu diesem Buch unter `www.carpelibrum.de`.

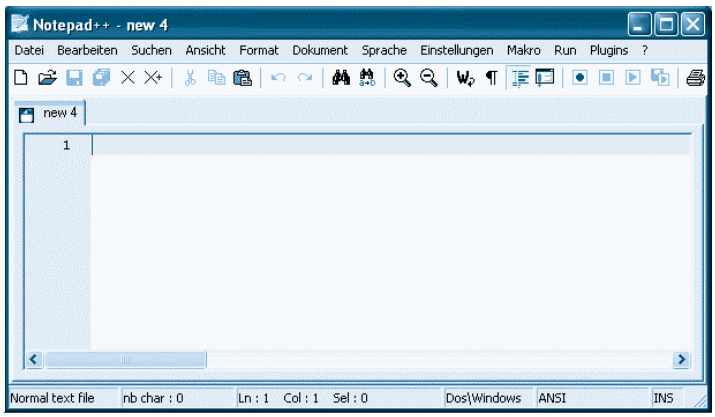

Abbildung 3.9: Java-Programme können mit jedem beliebigen Texteditor geschrieben werden. Empfehlenswert: die Freeware Notepad++.

8 Wählen Sie einen Editor für das Aufsetzen der Programmquelltexte aus.

Zu guter Letzt brauchen wir noch ein Textverarbeitungsprogramm zum Erstellen der Java-Quellcodedateien. Hierzu gibt es spezielle *Editoren*, aber für die Zwecke dieses Buches reicht auch der simple Windows-Editor *Notepad*, den Sie über *Start/Programme/Zubehör/Editor* aufrufen können. Oder laden Sie sich aus dem Internet *Notepad++* herunter (z.B. von `www.chip.de`), ein kostenloser und sehr guter Editor. Natürlich können Sie auch ein ausgewachsenes Textverarbeitungssystem wie Word oder StarOffice verwenden. Wichtig ist nur,

dass Sie dann beim Speichern die Funktion *Speichern unter* wählen und als Dateityp *Nur Text* auswählen.

> **Hinweis**
>
> Unter Linux können Sie beispielsweise den vi oder KEdit/KWrite (falls Sie mit der KDE-Oberfläche arbeiten) verwenden.

Damit sind auch schon alle wesentlichen Vorbereitungen getroffen und es kann mit dem ersten Programm losgehen!

Das erste Programm

Nun wird es ernst. Wir beginnen damit, den Java-Quelltext aufzusetzen.

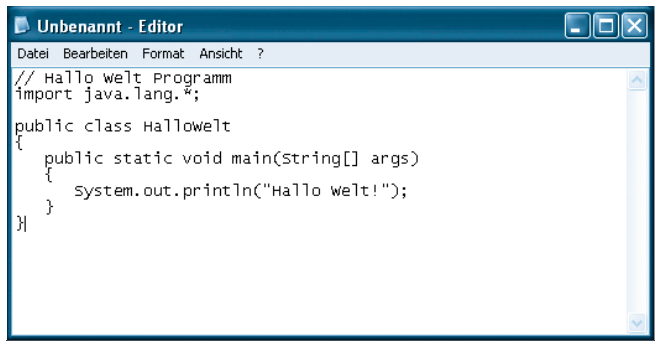

Abbildung 3.10: Das erste Programm im Editor Notepad

1 Legen Sie in Ihrem Editor eine neue Datei an und tippen Sie den folgenden Programmtext ein.

```
// Hallo Welt Programm
import java.lang.*;

public class HalloWelt
{
   public static void main(String[] args)
   {
      System.out.println("Hallo Welt!");
   }
}
```

Geben Sie diesen Quellcode bitte genau so ein wie hier abgedruckt. Wichtig ist insbesondere das Semikolon am Ende der `println`-Zeile.

Achtung

In Java wird zwischen Groß- und Kleinschreibung unterschieden. Wenn Sie also beispielsweise `Main()` statt `main()` eintippen, ist dies ein Fehler!

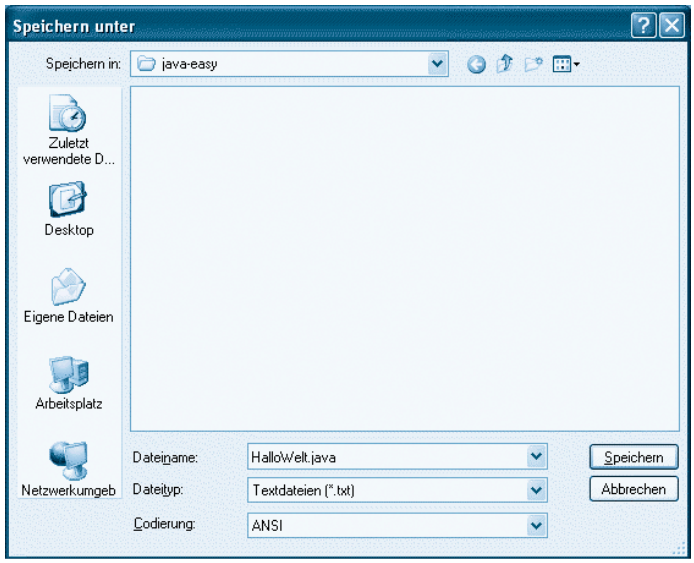

Abbildung 3.11: Quelltext speichern

2 Den Quellcode speichern.

Nun wird der Quellcode gespeichert. Verwenden Sie als Dateinamen *HalloWelt.java* (Groß-/Kleinschreibung beachten!) und als Verzeichnis das oben vorgeschlagene *c:\java-easy*.

Hinweis

Etwas lästig ist beim Einsatz mancher Textverarbeitungsprogramme, dass sie unter Umständen an die zu speichernde Datei die Endung *.txt* anhängen. Falls dies passiert, müssen Sie über den Windows Explorer von Hand den Dateinamen korrigieren und das *.txt* wieder beseitigen (Datei auswählen, dann rechte Maustaste klicken und *Umbenennen* wählen). Bei Einsatz des Windows-Editors *Notepad* können Sie sich auch dadurch behelfen, dass Sie bei der Angabe des Dateinamens den Namen in Anführungszeichen setzen (also z.B. "HalloWelt.java").

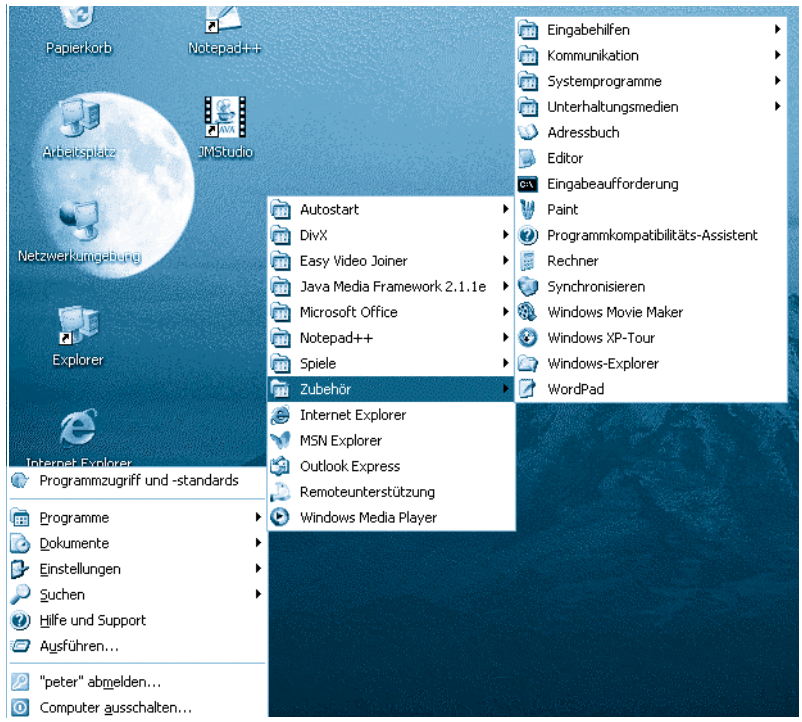

Abbildung 3.12: Konsole öffnen

3 Eine Konsole aufmachen.

Der nächste Schritt im Entwicklungsprozess ist das Kompilieren des Java-Quelltextes in den Zwischencode (Bytecode). Hierfür brauchen wir den Java-Compiler. Leider ist er keine grafische Anwendung und ohne Fenster. Wir müssen daher zunächst die Windows-Konsole (auch Eingabeaufforderung genannt) aufmachen – beispielsweise über das Start-Menü (*Start*/*Programme*/*Zubehör*/*Eingabeaufforderung*[6]) oder über *START*/*Ausführen* und die Eingabe von *cmd*.

6 *Je nach Windows-Version wird die Konsole auch über Start/Programme/Eingabeaufforderung oder Start/Programme/MS-DOS Eingabeaufforderung aufgerufen.*

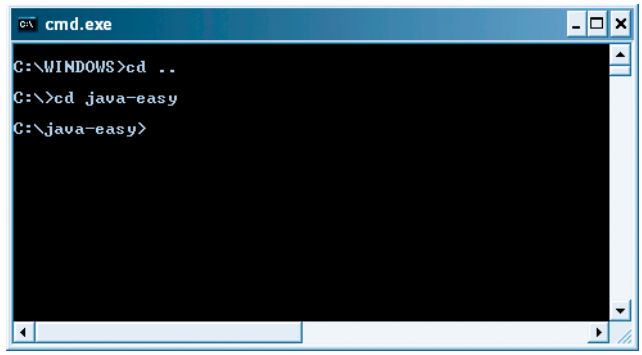

Abbildung 3.13: Das Verzeichnis wechseln

Wenn die Konsole erscheint, müssen Sie mit dem *cd*-Befehl (*change directory*) in das gewünschte Arbeitsverzeichnis wechseln. In unserem Beispiel wäre das *c:\java-easy*. Tippen Sie also `cd java-easy` ein und drücken Sie die ↵-Taste.

Wenn Sie in ein Unterverzeichnis wechseln wollen, müssen Sie nicht den ganzen Pfad angeben, die Angabe des Verzeichnisses reicht. Um also beispielsweise von *c:* in das Verzeichnis *c:\java-easy* zu wechseln, genügt der Befehl:

```
cd java-easy
```

Wenn Sie in ein übergeordnetes Verzeichnis wechseln wollen, tippen Sie

```
cd ..
```

ein.

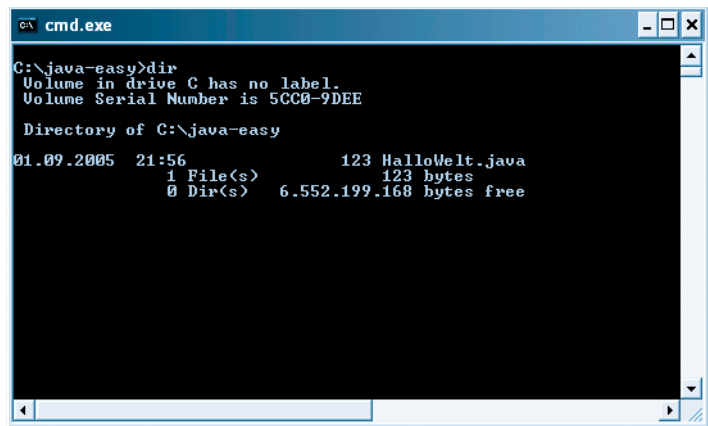

Abbildung 3.14: Inhalt eines Verzeichnisses auflisten lassen

Ein weiterer wichtiger Befehl, den Sie oft brauchen werden, ist *dir* (*directory*). Er zeigt den Inhalt des aktuellen Verzeichnisses an. Wenn Sie die Dateien und Unterverzeichnisse des aktuellen Verzeichnisses seitenweise auflisten lassen wollen, geben Sie den Befehl *dir /p* ein.

> **Hinweis**
>
> Unter Linux lauten die Konsolenbefehle zum Wechseln und Auflisten der Verzeichnisse *cd* und *ls*.

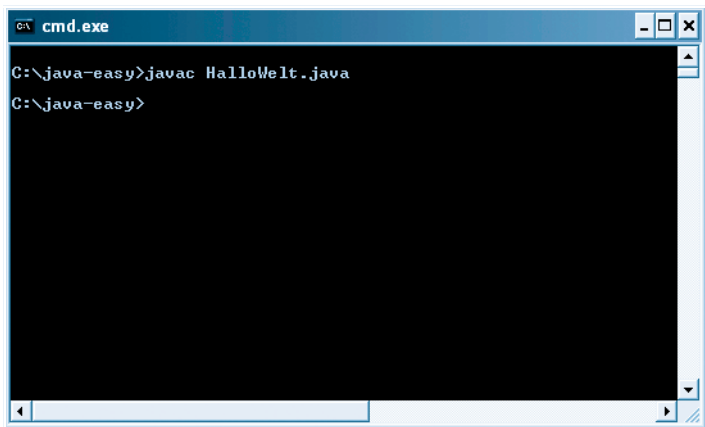

Abbildung 3.15: Quelltext kompilieren

4 Den Quelltext kompilieren.

Der nächste Schritt im Entwicklungsprozess ist das Kompilieren des Java-Quelltextes. Der Java-Compiler hat den unauffälligen Namen *javac* und wird über die Konsole aufgerufen. Als Parameter erwartet er den Namen der zu kompilierenden Java-Datei. Das Kommando lautet also in unserem Beispiel `javac HalloWelt.java`.

Wenn Sie nun wieder mit dem *dir*-Befehl den Inhalt des Verzeichnisses überprüfen, werden Sie feststellen, dass eine neue Datei mit dem Namen *HalloWelt.class* erzeugt worden ist. Sie enthält den Bytecode.

> **Achtung**
>
> Achten Sie auch hier auf die Groß- und Kleinschreibung!

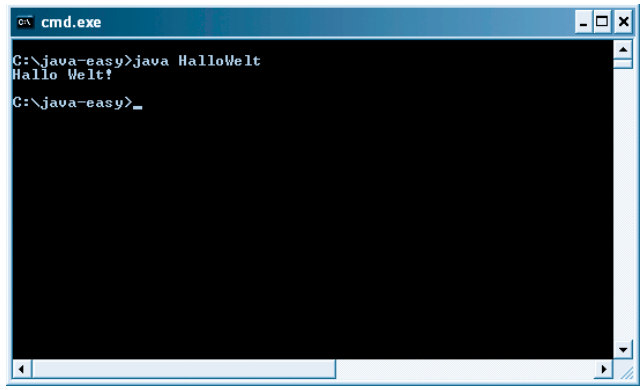

Abbildung 3.16: Programm ausführen

5 Das Programm ausführen.

Der Bytecode kann nun dem Java-Interpreter zur Ausführung übergeben werden. Der Java-Interpreter ist ebenfalls ein Konsolenprogramm und heißt schlicht *java*. Er benötigt als Parameter den Namen der auszuführenden *.class*-Datei, allerdings ohne die Endung *.class*! Das Ausführen unseres ersten Programms erfolgt also als `java HalloWelt`.

Nach dem Abschicken von `java HalloWelt` und Drücken der ↵-Taste wird nach wenigen Sekunden die Meldung `Hallo Welt!` in der Konsole erscheinen. Sie ist das Lebenszeichen von Ihrem ersten Java-Programm, das erfolgreich ausgeführt worden ist!

Syntaxfehler beheben

Nur selten kommt es vor, dass man beim Aufsetzen des Quelltextes keinen Fehler macht. Oft sind es ganz banale Tippfehler, die dem Java-Novizen das Lernen zur Hölle machen. Dabei sind diese so genannten »syntaktischen Fehler« noch recht harmlos, denn sie werden – im Gegensatz zu den logischen Fehlern – vom Compiler entdeckt und meist recht gut lokalisiert.

Syntaktische Fehler liegen vor, wenn der Quelltext nicht den Regeln der Programmiersprache – also hier Java – entspricht. Im Gegensatz hierzu gibt es noch logische Fehler, die sich dadurch bemerkbar machen, dass das Programm zwar fehlerfrei kompiliert werden kann, aber bei seiner Ausführung nicht das macht, was der Programmierer eigentlich wollte.

Während diese logische Fehler zum Teil sehr schwierig zu finden sind, ist die Lage bei Syntaxfehlern deutlich besser, da der Java-Compiler dabei hilft. Sobald ihm beim Kompilieren ein Verstoß gegen die Java-Syntax auffällt, gibt er

eine entsprechende Fehlermeldung aus und bricht das Kompilieren ab. Viele Syntaxfehler sind schlichte Tippfehler. Sehr beliebt ist zum Beispiel das Semikolon, das prinzipiell nach jeder Anweisung stehen muss[7].

Syntaxfehler und dadurch Fehlermeldungen beim Kompilieren sind der Alltag und daher wollen wir hier schon einmal diese Situation simulieren. Manipulieren Sie deshalb den Quelltext aus dem vorigen Abschnitt, indem Sie beispielsweise das Semikolon am Ende der `println()`-Anweisung entfernen. Speichern Sie dann die Datei wieder neu ab und starten Sie in der Konsole durch den Aufruf von `javac HalloWelt.java` einen neuen Kompilierdurchgang.

Sie werden folgendes Resultat erhalten:

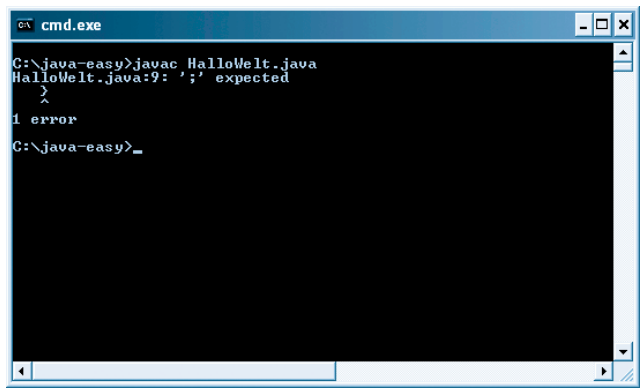

Abbildung 3.17: Der Compiler ist auf einen Fehler gestoßen.

javac hat erkannt, dass im Code ein Semikolon fehlt, und eine entsprechende Fehlermeldung ausgegeben. Die Fehlermeldung beginnt mit dem Namen der betroffenen Datei, gefolgt von der Zeilennummer und dem eigentlichen Meldungstext. Die Stelle im Code, wo der Fehler aufgetreten ist, wird durch ein ^ in der Fehlermeldung angezeigt. Etwas verwirrend ist, dass der Compiler das Semikolon erst vor der geschweiften Klammer (in der Zeile unter der `println()`-Anweisung) vermisst. Dies liegt daran, dass der Compiler überflüssige Leerzeichen, die der Programmierer setzt, um seinen Quellcode übersichtlicher zu gestalten (Umbrüche zur Aufteilung in mehrere Zeilen, Tabulatoren oder Leerzeichen zum Einrücken), schlichtweg ignoriert. Am Ende der `println()`-Anweisung erwartet der Compiler also ein Semikolon, aber es muss sich nicht notwendigerweise direkt anschließen. Erst als der Compiler in der folgenden Zeile auf die schließende geschweifte Klammern stößt, ist ihm klar, dass die `println()`-Anweisung nicht mehr mit Semikolon abgeschlossen wird, und er gibt die Fehlermeldung aus.

7 *Die Details hierzu erfahren Sie in den folgenden Kapiteln.*

Beseitigen Sie nun aufgrund dieser Informationen den Fehler und kompilieren Sie dann hoffentlich[8] fehlerfrei.

Hinweis

Auch wenn Sie zwischen einer Anweisung und dem abschließenden Semikolon beliebig viele Leerzeichen einfügen können, sollten Sie dies nicht tun. Setzen Sie das Semikolon immer direkt ans Ende der Anweisung!

Tipp

Manchmal erzeugt ein Fehler mehrere Fehlermeldungen. Korrigieren Sie daher zuerst immer nur den ersten Fehler und lassen Sie den Quelltext neu kompilieren. Manchmal verschwinden die Nachfolgefehler dann automatisch.

Quelltexte verwalten

Da Sie im Laufe des Buches etliche Java-Programme erzeugen werden, ist es auf die Dauer unhandlich, alle *.java*- und *.class*-Dateien im gleichen Verzeichnis abzuspeichern. Daher möchten wir Sie an dieser Stelle dazu anregen, für jedes Programm ein eigenes Unterverzeichnis im *java-easy*-Verzeichnis anzulegen.

Für das obige Beispiel wollen wir nun das Unterverzeichnis *HalloWelt* anlegen und dorthin die schon vorhandenen Dateien *HalloWelt.java* und *HalloWelt.class* verschieben. Für diese Aktionen können Sie entweder den Windows Explorer verwenden (dies werden Sie sicherlich schon können) oder auch mit Hilfe von Kommandos, die in der Konsole eingegeben werden müssen. Da Sie sowieso mit der Konsole arbeiten müssen, werden wir die notwendigen Schritte kurz erwähnen:

1 Wechseln Sie mit dem *cd*-Kommando in das Verzeichnis, wo ein Unterverzeichnis angelegt werden soll.

Da wir uns schon im Verzeichnis *java-easy* befinden, kann dieser Schritt hier ausnahmsweise entfallen.

2 Erzeugen Sie das neue Verzeichnis mit dem Kommando *mkdir*.

Dieser Befehl erwartet als Parameter den Namen des anzulegenden Verzeichnisses; in unserem Fall ist also zu tippen: `mkdir HalloWelt`.

8 *Ein weiterer beliebter Fehler ist das Einbauen neuer Fehler beim Beseitigen eines alten Fehlers!*

3 Verschieben Sie die Dateien *HalloWelt.java* und *HalloWelt.class* in das Unterverzeichnis mit dem Befehl *move*.

Der Befehl *move* erwartet als ersten Parameter den Namen der zu verschiebenden Datei und als zweiten Parameter den Namen des Zielverzeichnisses, also `move HalloWelt.java HalloWelt` und `move HalloWelt.class HalloWelt`.

Tipp

Wenn Sie einen bereits in der Konsole eingetippten Befehl nochmals ausführen lassen wollen, drücken Sie einfach so oft die ↑-Taste, bis der Befehl in der Konsole erscheint. Mit Hilfe der ←- und der →-Taste können Sie sich in der Befehlszeile bewegen und den Befehl vor dem Abschicken mit der ↵-Taste editieren.[9]

Hinweis

Wenn Sie nicht verschieben, sondern kopieren wollen, verwenden Sie das Kommando *copy* statt *move*. Die Parameter bleiben gleich: zuerst die »Quelle«, dann das »Ziel«.

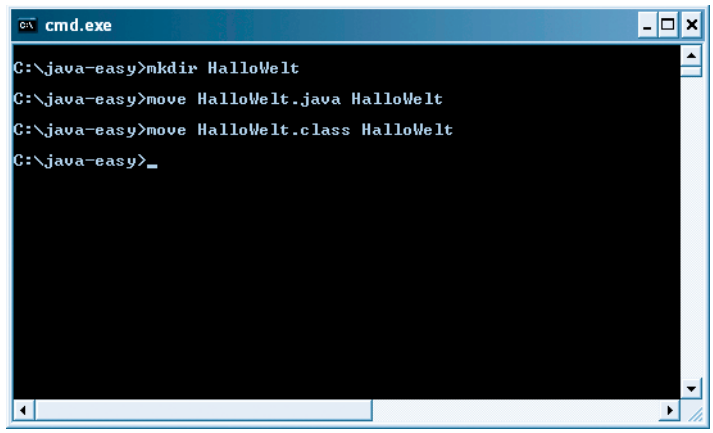

Abbildung 3.18: Das Anlegen eines Verzeichnisses und das Verschieben der bisherigen Dateien

9 *Unter Windows 95/98 müssen Sie die Aufzeichnung der eingegebenen Kommandozeilenbefehle explizit aktivieren. Geben Sie dazu nach Start der Konsole den Befehl doskey am Prompt ein oder schreiben Sie den Befehl doskey direkt in die autoexec.bat.*

Wenn Sie nun mit dem *dir*-Befehl den Inhalt des aktuellen Verzeichnisses *java-easy* prüfen, werden Sie sehen, dass die Dateien verschwunden sind und dafür ein neues Verzeichnis *HalloWelt* vorhanden ist. Wechseln Sie daher mit `cd HalloWelt` in dieses Unterverzeichnis und lassen Sie sich mit *dir* den Inhalt anzeigen!

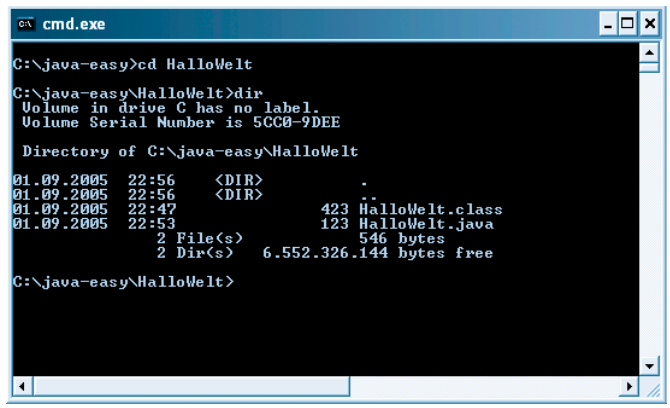

Abbildung 3.19: Das neu angelegte Verzeichnis mit den kopierten Dateien.

Hinweis

Unter Linux lauten die entsprechenden Befehle *mv* (für *move*) und *cp* (für *copy*).

Hätten Sie gedacht ...

... dass auch während der Ausführung eines Java-Programms (mit dem Interpreter *java*) unter Umständen eine Nachkompilierung stattfindet?

Die Java Virtual Machine überprüft beim Abarbeiten laufend, welche Codeteile des Programms besonders oft durchlaufen werden. Diese Bereiche werden als »Hot Spots« bezeichnet, da sie die Gesamtlaufzeit des Programms dominieren. Der Java-Interpreter übersetzt solche Codebereiche während der Abarbeitung nach und nach in den nativen Maschinencode der jeweiligen Plattform, so dass sie deutlich schneller abgearbeitet werden können.