PEARSON Education

Markt+Technik

easy

Java 2

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-Entwicklungssystem (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.

Installation des JDK

Alles, was wir – abgesehen von einem Editor zum Aufsetzen der Programmquelltexte – zur Programmerstellung mit Java benötigen, ist im Java Development Kit, kurz JDK, zusammengestellt. Den JDK kann man sich kostenlos von der Java-Website *http://java.sun.com* herunterladen (er wird dort unter dem Namen *J2SDK Standard Edition* geführt). Die zum Zeitpunkt der Drucklegung dieses Buches aktuelle Version finden Sie aber auch auf der Buch-CD im Verzeichnis *JDK*.

Datei	Beschreibung
CDROM:\JDK\j2sdk1_4_2-windows-i586.exe	Setup-Datei für Windows
CDROM:\JDK\j2sdk-1_4_2-linux-i586.bin	Setup-Datei für Linux
CDROM:\JDK\j2sdk1_4_2-doc.zip	ZIP-Datei mit zusätzlicher Java-Referenz im HTML-Format

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

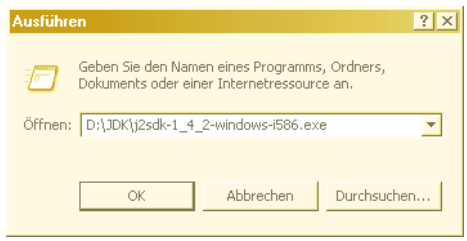
Zur Installation des JDK schließen Sie bitte zunächst alle laufenden Anwendungen. Installationsprogramme (auch Setup-Programme genannt) sind unter Windows häufig etwas empfindlich und könnten durch andere Anwendungen, die gerade ausgeführt werden, gestört werden.

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



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 *j2sdk-1_4_2-windows-i586.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\j2sdk-1_4_2-windows-i586.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.

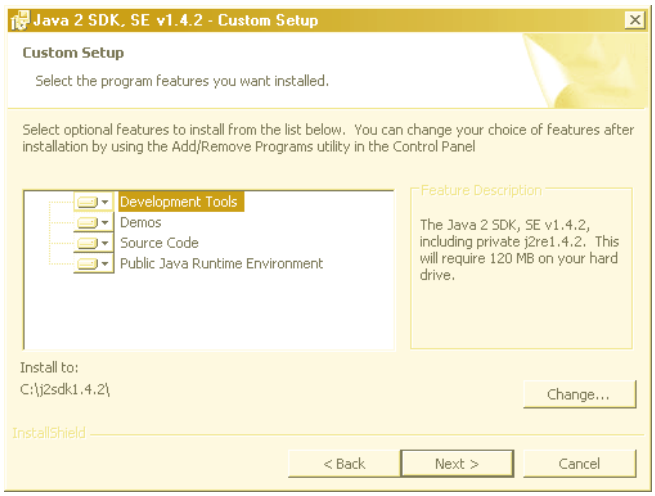
Zum Schluss wird durch Klicken auf den OK-Schalter das Installationsprogramm gestartet.

HINWEIS

Das Setup-Programm für Linux heißt j2sdk-1_4_2-linux-i586.bin. Kopieren Sie die Datei in das Verzeichnis, unter dem der 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

```
./j2sdk-1_4_2-linux-i586.bin
```

ein.



3 Beantworten Sie die Fragen des Installationsprogramms.

Es erscheinen jetzt nacheinander das Begrüßungsfenster des Installations-Assistenten, der Lizenzvertrag und die Abfrage von Umfang der Installation und gewünschtes Zielverzeichnis. Übernehmen Sie am besten alle Vorgaben und klicken Sie jeweils auf die bestätigende Auswahl (je nach Fenster YES, ACCEPT, NEXT oder FINISH). Bei den Lizenzbestimmungen müssen Sie auf I ACCEPT THE TERMS IN THE LICENCE AGREEMENT klicken, damit der NEXT-Button aktiviert wird. Die letzte Abfrage betrifft die Installation von Java Plugins für die gängigen Browser Internet Explorer und Netscape. Lassen Sie die vorgegebenen Häkchen aktiviert, damit Ihre Browser in der Lage sind, Applets nach dem neuesten Java-Standard auszuführen.

Nach Abschluss der Installation ist Ihre Festplatte um bis zu 120 MByte freien Speicherplatz ärmer, aber Sie um das J2SDK Standard Edition Version 1.4.2 (was für ein Name!) reicher.

Wenn Sie nun mit Hilfe des Windows Explorers auf Ihrer Festplatte nachschauen, werden Sie ein neues Verzeichnis *c:\j2sdk1.4.2* (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.

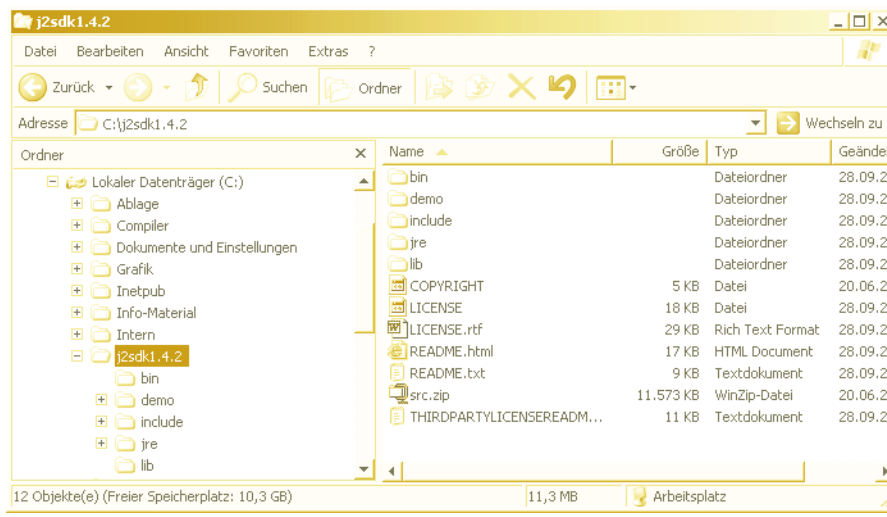


Abbildung 3.1: Das installierte JDK auf der Festplatte

Weitere Vorbereitungen

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

Ausführliche Informationen zur Einrichtung Ihres Rechners für die Arbeit mit dem JDK finden Sie auch auf der Support-Site zu diesem Buch, www.carpelibrum.de, und auf den Download-Webseiten des Sun-Servers.

4 Tragen Sie die Java-Entwicklungswerkzeuge in Ihren Systempfad 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 XP öffnet man die Konsole über den START-Menüeintrag ALLE PROGRAMME/ZUBEHÖR/EINGABEAUFFORDERUNG[1]. 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 3.2: Aufruf des javac-Compilers von der Windows-Konsole aus).

WAS IST DAS

Über die Konsole kann man Betriebssystembefehle eingeben und Programme ausführen, die – wie die Java-Entwicklungswerkzeuge – über keine grafische Benutzeroberfläche verfügen und nicht mit dem Fenstermanager des Window-Systems zusammenarbeiten.[2]

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

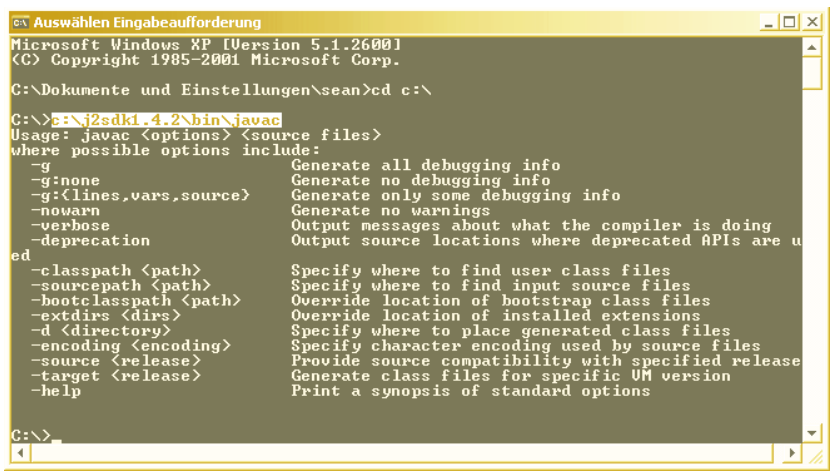


Abbildung 3.2: Aufruf des javac-Compilers von der Windows-Konsole aus

1. Windows 98: START/PROGRAMME/MS-DOS-EINGABEAUFFORDERUNG oder Windows 2000: START/PROGRAMME/ZUBEHÖR/EINGABEAUFFORDERUNG.
2. Die meisten Betriebssysteme erlauben dem Anwender, den Prompt nach seinen Vorstellungen zu konfigurieren.

Es ist natürlich sehr lästig, immer den vollen Pfad für das Programm *javac* einzutippen. Daher sollten Sie das bin-Verzeichnis der JDK-Installation in den Systempfad (Umgebungsvariable PATH) eintragen. Der PATH legt fest, wo das Betriebssystem nach auszuführenden Programmen sucht. Je nach Betriebssystem ist die Vorgehensweise unterschiedlich. Wir nehmen wir für die folgenden Erklärungen an, dass Sie das JDK in das Verzeichnis *C:\j2sdk1.4.2* installiert haben.

Windows 98:

Laden Sie die Systemdatei *c:\autoexec.bat* in einen Texteditor (beispielsweise über START/PROGRAMME/ZUBEHÖR/EDITOR) oder von der Konsole aus mit dem Kommando *notepad c:\autoexec.bat* und suchen Sie dort nach einem PATH-Eintrag. Hängen Sie an das Ende ein Semikolon und das JDK-Bin-Verzeichnis an.

WAS IST DAS

Die Datei autoexec.bat, die im Verzeichnis C:\ stehen muss, ist eine Batch-Datei, die automatisch beim Hochfahren des Windows 98-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:\;c:\dos;c:\windows
```

so erweitern Sie diesen zu:

```
SET PATH=.;c:\;c:\dos;c:\windows;c:\j2sdk1.4.2\bin
```

HINWEIS

Wenn Sie gar keine PATH-Angabe finden, dann fügen Sie eine neue PATH-Anweisung hinzu, beispielsweise SET PATH=c:\j2sdk1.4.2\bin. Eventuell müssen Sie sogar die autoexec.bat-Datei neu anlegen.

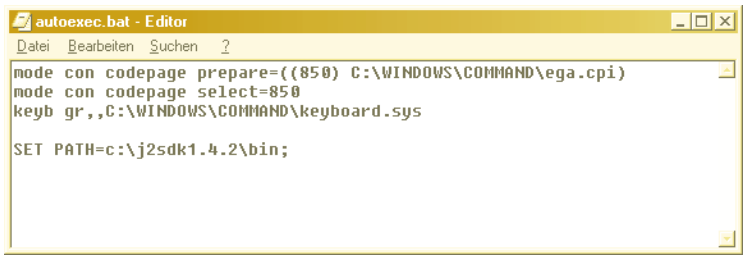


Abbildung 3.3: Die Datei autoexec.bat unter Windows 98

Starten Sie den Computer danach neu, damit die neue *autoexec.bat* aktiviert wird, öffnen Sie eine Konsole und testen Sie, ob Sie *javac* oder *java* von beliebigen Verzeichnissen aus aufrufen können.

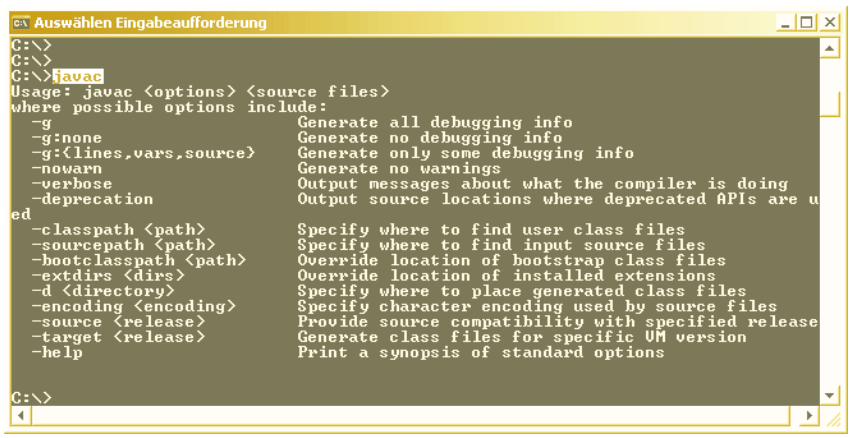


Abbildung 3.4: Aufruf des javac-Compilers ohne Pfadangabe

Windows 2000/XP:

Hier wird die Umgebungsvariable PATH über den Dialog der Systemeigenschaften verwaltet. Der Weg dorthin ist lang und von Betriebssystem zu Betriebssystem verschieden. Unter Windows XP rufen Sie über die START-Schaltfläche der Taskleiste (START/EINSTELLUNGEN unter Windows 2000) die SYSTEMSTEUERUNG auf und gelangen via SYSTEM, Register ERWEITERT, Schalter UMGEBUNGSVARIABLEN zum Ziel.

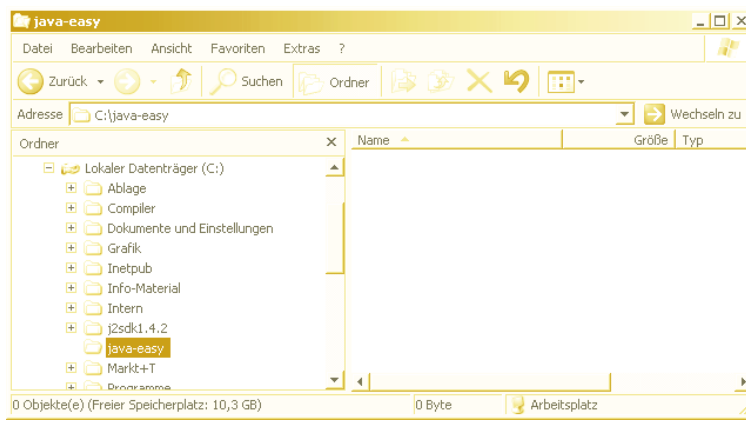
Dort markieren Sie in einem der Listenfelder für den aktuell angemeldeten Benutzer oder das gesamte System die Variable PATH und drücken die Schaltfläche BEARBEITEN. In dem aufspringenden Dialogfeld können Sie den PATH erweitern. Hängen Sie durch Semikolon getrennt die gewünschte Pfadangabe (in unserem Beispiel *c:\j2sdk1.4.2\bin*) am Ende der Zeile an und drücken Sie OK (oder SETZEN), um die Änderung zu aktivieren.

Falls es noch keinen PATH-Eintrag gibt, fügen Sie mit der Schaltfläche NEU einen hinzu und setzen ihn auf den Wert *c:\j2sdk1.4.2\bin*. Beachten Sie bitte, dass der neu gesetzte PATH nur für neu geöffnete Konsolenfenstern ausgewertet wird; Sie müssen also ein neues Konsolenfenster öffnen.[3]

HINWEIS

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/ihrUsername/jsdk1.4.2/bin in der nächsten Zeile nach der bisherigen Pfadangabe hinzu:*

```
set path = (/home/myname/j2sdk1.4.2/bin $path)
```



5 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/WINDOWS EXPLORER oder START/PROGRAMME/ZUBEHÖR/WINDOWS EXPLORER) ein solches Verzeichnis an (Befehl DATEI/NEU/ORDNER).

3. Beachten Sie bitte, dass das Setzen von Umgebungsvariablen unter Linux stark von der verwendeten Shell abhängt. Lesen Sie im Handbuch Ihrer Linux-Installation nach, was Sie tun müssen.

HINWEIS

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

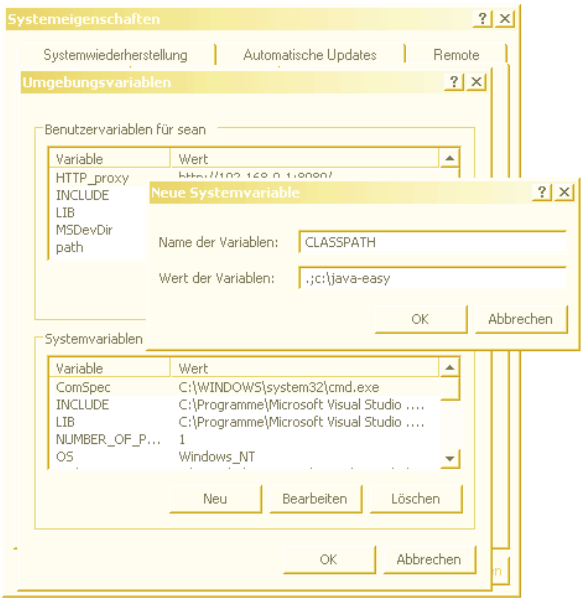
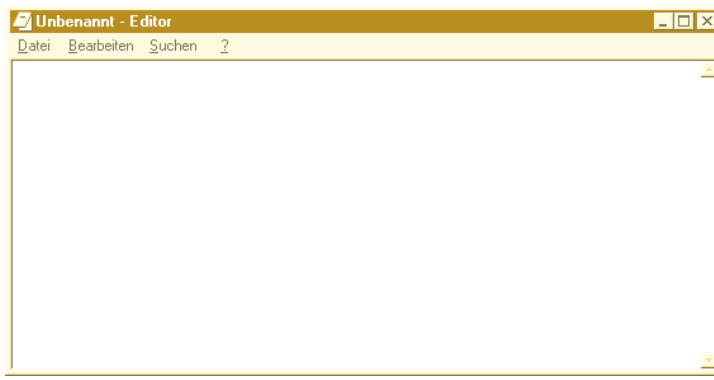


Abbildung 3.5: Das Setzen der CLASSPATH-Umgebungsvariablen (unter Windows 2000/XP)

6 Setzen Sie 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. Eventuell ist die CLASSPATH-Variable auf Ihrem System schon gesetzt. Für das Durcharbeiten des Buches empfehlen wir, den CLASSPATH-Eintrag um das aktuelle Verzeichnis (repräsentiert durch den Punkt .) und das im vorangehenden Schritt angelegte *c:\java-easy*-Verzeichnis zu erweitern.

Das Setzen bzw. Anlegen der CLASSPATH-Variable erfolgt analog zu den oben beschriebenen Schritten für das Setzen der PATH-Umgebungsvariablen (siehe Abbildung 3.5: Das Setzen der CLASSPATH-Umgebungsvariablen (unter Windows 2000/XP) für die Einrichtung unter Windows 2000/XP).



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

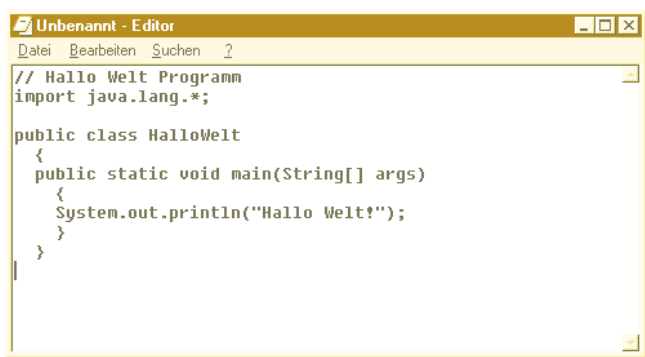
Zu guter Letzt brauchen wir noch ein Textverarbeitungsprogramm zum Erstellen der Java-Quellcode-Dateien. Hierzu gibt es spezielle *Editoren*, aber für die Zwecke dieses Buches reicht auch der simple Windows-Editor *notepad*, den Sie über START/PROGRAMME/ZUBEHÖR/EDITOR aufrufen können. Natürlich können Sie auch ein ausgewachsenes Textverarbeitungssystem wie Word oder StarOffice verwenden. Wichtig ist nur, dass Sie beim Speichern die Funktion SPEICHERN UNTER wählen und dann als Dateityp NUR TEXT auswählen.

Unter Linux können Sie beispielsweise den vi oder KEdit (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.



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

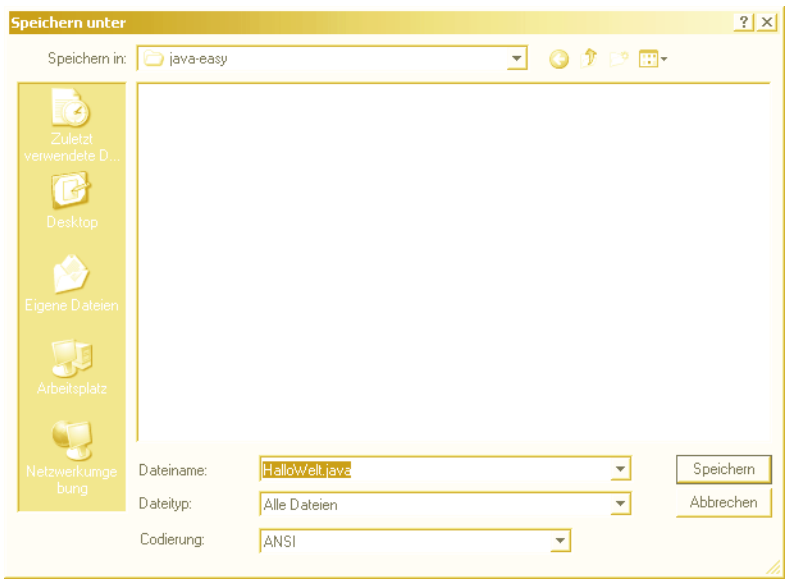
```
// Hallo Welt Programm
import java.lang.*;

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. Beachten Sie ferner, dass `println` von »`print line`« abstammt. Nach dem Buchstaben `t` folgt daher der Buchstabe `l` und nicht die Zahl `1`!

ACHTUNG

In Java wird zwischen Groß- und Kleinschreibung unterschieden. Wenn Sie also beispielsweise `Main()` *statt* `main()` *eintippen, ist dies ein Fehler, da die Java-Syntax an dieser Stelle eine Methode namens* `main()` *vorschreibt und eben nicht* `Main()`*!*



2 Den Quellcode speichern.

Nun wird der Quellcode gespeichert. Als Dateinamen müssen Sie den Namen der Klasse verwenden, d.h. in unserem Beispiel lautet der Dateiname *HalloWelt.java* (Groß-/Kleinschreibung beachten!). Wählen Sie als Verzeichnis das oben vorgeschlagene *c:\java-easy*.

HINWEIS

Etwas lästig ist beim Einsatz mancher Textverarbeitungsprogrammen, dass sie unter Umständen an die zu speichernde Datei die Endung .txt automatisch anhängen.[4]

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").*

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 graphische Anwendung und ohne Fenster. Wir müssen daher zunächst die Windows-Konsole (die MS-DOS Eingabeaufforderung) aufmachen. Sie können sie wieder über die Start-Leiste finden (Windows XP: START/ALLE PROGRAMME/ZUBEHÖR/EINGABEAUFFORDERUNG).[5]

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

4. Tückisch ist hierbei, dass eventuell Ihr Windows so konfiguriert ist, dass Dateiendungen gar nicht angezeigt werden. Um dies zu ändern, öffnen Sie den Dateimanager (Windows Explorer) und deaktivieren dann über sein Menü EXTRAS/ORDNEROPTIONEN/ANSICHT den Eintrag DATEINAMENERWEITERUNG AUSBLENDEN.
5. Windows 98: START/PROGRAMME/MS-DOS-EINGABEAUFFORDERUNG oder Windows 2000: START/PROGRAMME/ZUBEHÖR/EINGABEAUFFORDERUNG.

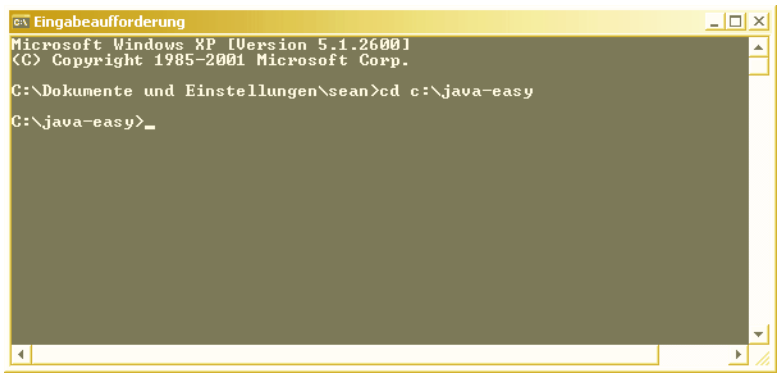


Abbildung 3.6: Das Verzeichnis wechseln

Wenn Sie in ein Unterverzeichnis wechseln wollen, müssen Sie nicht den ganzen Pfad anzugeben, die Angabe des Verzeichnisses reicht. Um also beispielsweise von *C:* aus 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.

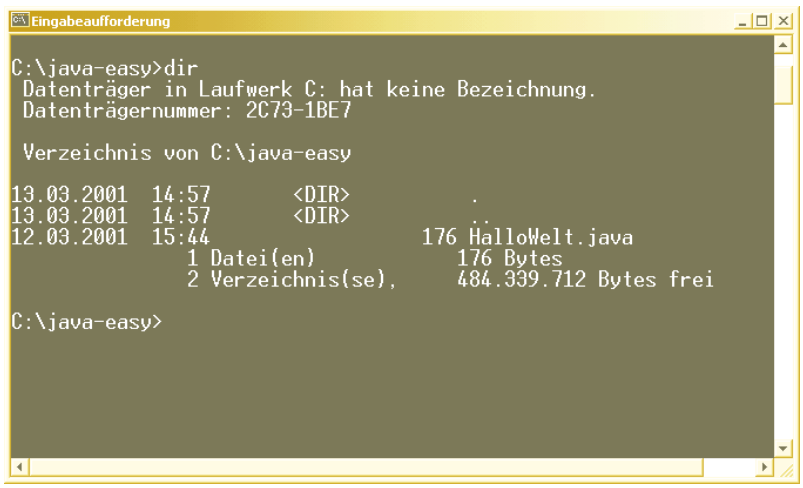
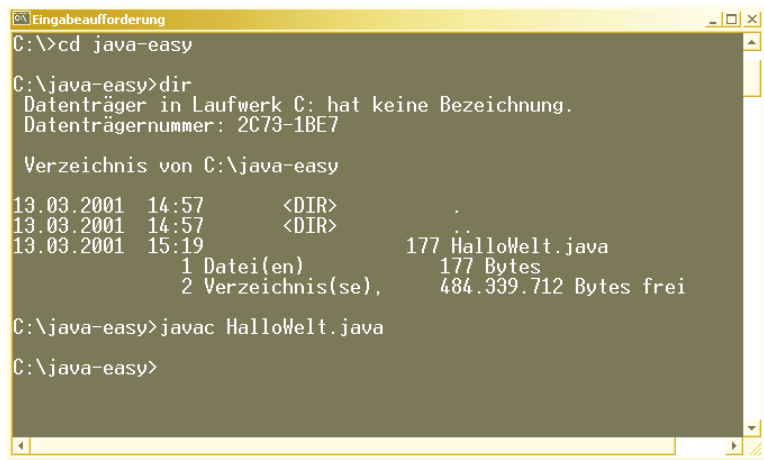


Abbildung 3.7: Inhalt eines Verzeichnisses auflisten lassen

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.



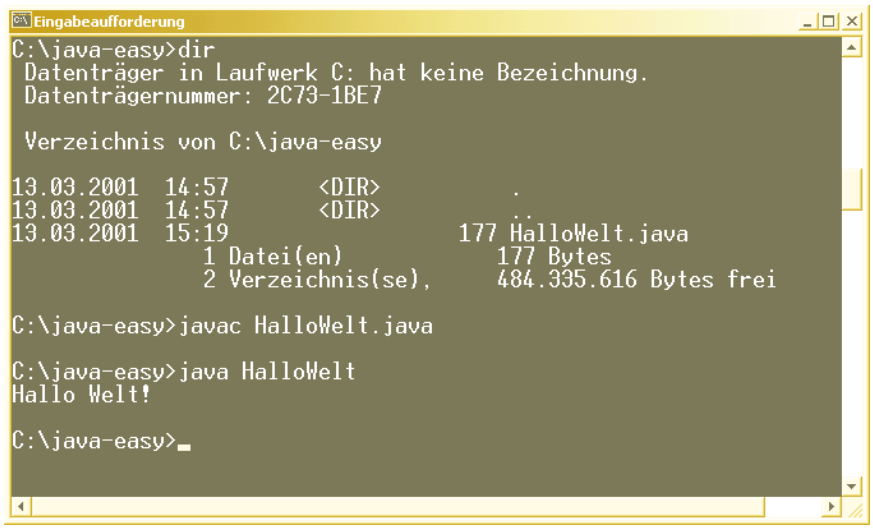
4 Den Quelltext kompilieren.

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 im Dateinamen!



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 Ihres ersten Java-Programms, das erfolgreich ausgeführt worden ist!

Ein häufig auftretender Fehler in diesem Schritt ist eine Fehlermeldung der Art `java.lang.NoClassDefFoundError`. Prüfen Sie dann bitte folgende Punkte:

- Haben Sie Groß-/Kleinschreibung beim Aufruf von *HalloWelt* beachtet?
- Sind Sie in der Konsole im Verzeichnis *c:\java-easy*?
- Ist die CLASSPATH-Variable richtig gesetzt? Sie sollte einen Eintrag auf *c:\java-easy* oder das aktuelle Verzeichnis (repräsentiert durch den Punkt .) enthalten.

Syntaxfehler beheben

Nur selten kommt es vor, dass man beim Aufsetzen des Quelltextes keinen Fehler macht. Oft sind es ganz unnötige 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 aber auch das Weglassen des Semikolons, das prinzipiell nach jeder Anweisung stehen muss[6].

Syntaxfehler und die aus ihnen resultierenden Fehlermeldungen beim Kompilieren sind der Alltag und daher wollen wir hier schon mal 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 MS-DOS-Eingabeaufforderung durch den Aufruf von *javac HalloWelt.java* einen neuen Kompilierdurchgang. Sie werden das Resultat aus Abbildung 3.8: Der Compiler ist auf einen Fehler gestoßen erhalten:

javac hat erkannt, dass hinter der schließenden Klammer der `println()`-Anweisung das Semikolon fehlt. Dies zeigt er Ihnen auch durch ein ^ in seiner Fehlermeldung an. Sehr hilfreich ist auch die Angabe *HalloWelt.java:8*, die angibt, dass der Fehler in Zeile 8 von Quellcode-Datei *HalloWelt.java* aufgetreten ist. Beseitigen Sie nun aufgrund dieser Informationen den Fehler und kompilieren Sie dann hoffentlich[7] fehlerfrei.

6. Die Details hierzu erfahren Sie in den folgenden Kapiteln.
7. Ein weiterer beliebter Fehler ist das Einbauen neuer Fehler beim Beseitigen eines alten Fehlers!

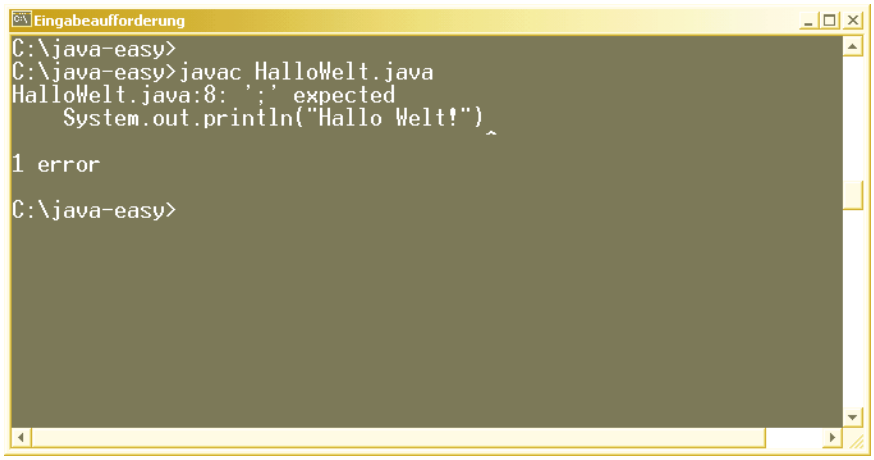


Abbildung 3.8: Der Compiler ist auf einen Fehler gestoßen

TIPP

Manchmal erzeugt ein Fehler mehrere Fehlermeldungen. Korrigieren Sie daher zuerst immer nur den ersten Fehler und lassen Sie dann 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 kann man 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 auflisten:

1 Wechseln Sie mit dem *cd*-Kommando in das Verzeichnis, unter dem 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*.

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.*[8]

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«.

8. 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*.

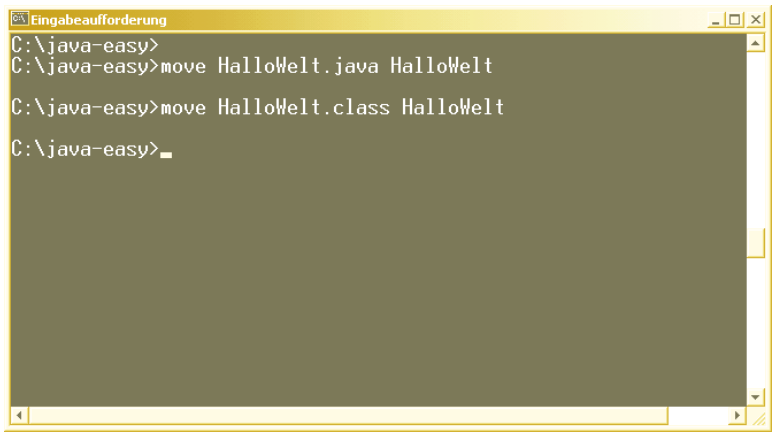


Abbildung 3.9: Das Anlegen eines Verzeichnisses und das Verschieben der bisherigen Dateien.

Wenn Sie nun mit dem *dir*-Befehl den Inhalt vom aktuellen Verzeichnis *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.

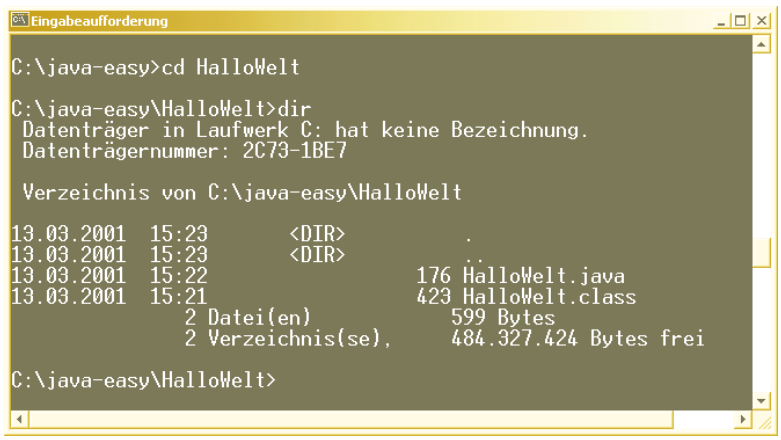


Abbildung 3.10: Das neu angelegte Verzeichnis mit den kopierten Dateien.

HINWEIS

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