Java

Alfred Nussbaumer

Java scheint in der Informatikausbildung durch seine Plattformunabhängigkeit, seine streng vorgegebene objektorientierte Programmierung und seine leichte Erlernbarkeit immer wichtiger zu werden. Ein paar dieser Aspekte soll an Hand dieses Artikels vorgestellt werden.

1. Java in der Schule

Java wurde 1995 von der Firma Sun Microsystems als plattformunabhängige Programmiersprache veröffentlicht. In der Folge verbreiteten sich Java-basierte Applets rasch über zahlreiche Internet-Seiten. Java-Applikationen wurden als plattformunabhängige Entwicklungen für Netzwerkserver und für Clientrechner geschrieben. Von Java abgeleitete Scriptsprachen wurden als JavaScript (gemeinsam mit Netscape) oder als JScript (gemeinsam mit Microsoft) populär.

Basic, Makroprogrammierung, Pascal, Delphi, JavaScript, PHP - zahlreiche Script- und Compilersprachen erlauben, Grundkenntnisse im Programmieren zu erlangen. Warum soll nun eine neue Sprache eingesetzt werden? Dennoch hat sich in den letzten Jahren Java an bestimmten Standorten etablieren können. Dies mag verschiedene Ursachen haben:

1. Das Java-Developing-Kit JDK (bzw. SDK) steht allen Anwendern unter der Adresse *http://www.java.sun.com/products/jdk/1.2/* gratis zum Download zur Verfügung (seit der Version 1.2 wird das JDK mit SDK bezeichnet).
2. Java ist einfach zu erlernen.
3. Java wird mittlerweile an Universitäten zur Ausbildung verwendet - dies legt nahe, Maturantinnen und Maturanten mit den Grundzügen der Programmierung in Java vertraut zu machen.

Grundsätzlich erzeugt der Java-Compiler aus dem Quelltext einen so genannten »Bytecode«. Dieser Bytecode ist plattformunabhängig - er wird erst vom Bytecodeinterpreter ausgeführt, der dem jeweiligen Betriebssystem angepasst ist. Auf diese Weise lassen sich Java-Entwicklungen auf jedem System ausführen, auf dem das Java-Runtime-Environment JRE installiert ist. Dabei kann etwa zwischen **Java-Applikationen, Java-Applets** und **Java-Servlets** unterschieden werden.

Java-Applikationen sind eigenständige Computer-Programme, die vom Bytecode-Interpreter in der jeweiligen Betriebssystemumgebung ausgeführt werden. Sie haben abhängig von den Rechten des jeweiligen Benutzers Zugriffsrechte auf das System. Applikationen können somit auch Dateien anzeigen, ändern, löschen oder schreiben. Anwendungen in fast allen Wirtschaftsbereichen können auf der WebSeite der Firma Sun Microsystems nachgelesen werden.

Java-Applets sind Teil einer WebSite und werden auf den Clientrechner übertragen. Falls in den Sicherheitseinstellungen des Browsers das Ausführen von Java-Applets zugelassen wurde, laufen Java-Applets im Rahmen des Browsers ab. Um unerwünschte Zugriffe auf das System des Clientrechners zu unterbinden, sind Zugriffe auf das Dateisystem, das Aufbauen von Netzwerksverbindungen und der Start von lokalen Prozessen nicht zugelassen (»Java in a Sandbox«). Applets werden beispielsweise für Animationen oder Visualisierungen von Inhalten auf WebSeiten verwendet.

Java-Servlets sind gleichsam das Gegenstück zu Java-Applets am Server: Geeignete Erweiterungen des WebServers führen Servlets aus und reichen die Ergebnisse an den WebServer weiter, der sie über das Netzwerk an den Clientrechner ausliefert. In Zusammenhang mit dynamisch generierten Webseiten, serverseitigen Datenbanken und Applikationsservern haben Servlets in den letzten Jahren eine wichtige Bedeutung erhalten.

Im Anfangsunterricht bietet es sich zunächst an, einfache Applikationen zu erstellen. Später können Applets als Erweiterung der Java-Klasse “Applet” geschrieben und in entsprechende HTML-Dokumente eingebettet werden. Im fortgeschrittenen Unterricht können schließlich Servlets verwendet werden, wenn Grundlagen zu Netzwerk und WebServer bekannt sind.

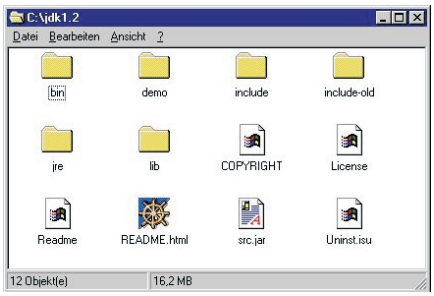
2. Systemanforderungen

In einfachen Varianten läuft Java auch auf älteren Systemen, beispielsweise auf Rechner mit einer 200MHz getakteten CPU, 32MB-RAM und Windows95. Damit sollte den Auszubildenden eine »Übungsmaschine« relativ leicht zur Verfügung stehen. Freilich steigt auch hier die Performance mit der Größe des Hauptspeichers und mit der Höhe des Prozessortaktes ;-). Entwicklungsumgebungen wie Forte4Java (*http://www.java.sun.com/*) oder JBuilder4 (*http://www.inprise.com/*) verlangen auf jeden Fall 64MB bzw. 128 MB RAM, hinreichend schnelle Prozessoren und genügend Speicherplatz auf den Platten. Da Java grundsätzlich plattformunabhängig ausgebildet wurde, können Java-Anwendungen auf Windows-, Linux- oder Solaris-Systemen in gleicher Weise entwickelt werden.

3. Erste Schritte

Um ohne große Systemanforderungen (einfache) Java-Anwendungen unter Windows schreiben zu können, installieren wir das JDK: Dazu kopieren wir beispielsweise die Datei **»jdk12-win32.exe«** aus dem Internet auf die lokale Platte. Nach dem Doppelklick auf das Icon wird die Datei entpackt und die Setup-Routine gestartet (die vorgeschlagenen Einstellungen können problemlos übernommen werden).

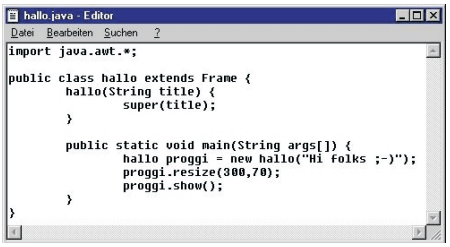
Nach der Installation sollte ein Verzeichnis **»jdk1.2.x«** auf der Festplatte vorhanden sein, in dem der Java-Compiler, der Java-Byte-Code-Interpreter und alle mitgelieferten Java-Klassen enthalten sind:



Für den Anfang mag es einfacher erscheinen, die Quelldateien und die kompilierten Klassen direkt ins Verzeichnis **C:\jdk1.2.x\bin** zu stellen. Besser ist es, die Umgebungsvariablen **PATH** und **CLASSPATH** in der Datei **AUTOEXEC.BAT** zu setzen (vgl. die Dokumentation zum JDK):

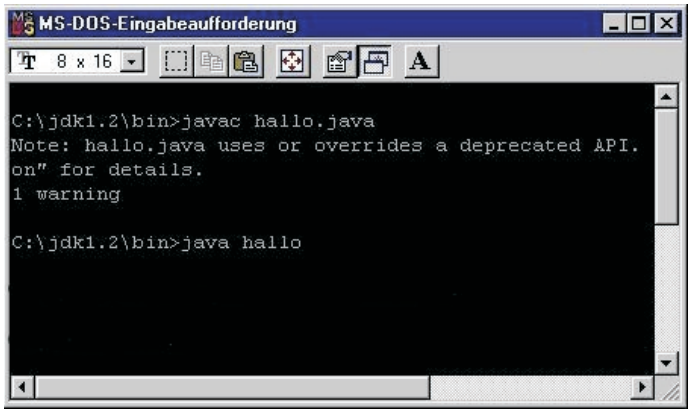
```
SET PATH = %PATH%; C:\jdk1.2.x\bin; C:\jdk1.2.x\lib
SET CLASSPATH = .; C:\jdk1.2.1\lib
```

Damit können wir unser erstes Java-Programm schreiben. Wir öffnen dazu den Editor des Betriebssystems, um den Quelltext zu erstellen:



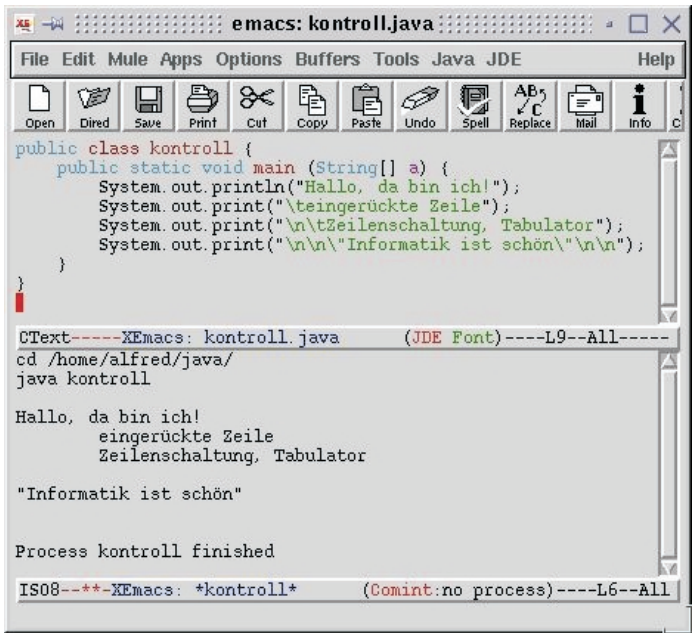
Die Quelltextdatei muss anschließend mit der Dateinamenergänzung **.java** gespeichert werden!

Im nächsten Schritt wird der Java-Quelltext kompiliert. Dazu startet man beispielsweise auf der MS-DOS-Eingabezeile (**command.com**) den Java-Compiler.



Der Compiler erstellt dabei die Bytecode-Datei mit der Dateinamenergänzung **.class** - diese wird schließlich mit Hilfe des Java-Interpreters ausgeführt:

Der Standardeditor **»xemacs«** stellt unter Linux eine hinreichend komfortable Umgebung zur Entwicklung von Javacode zur Verfügung: Beim Erstellen überprüft der Editor den Quellcode bereits auf syntaktische Fehler. Der Compiler und der Bytecodeinterpreter werden aus dem Menü **»JDE«** aufgerufen. Compilermeldungen, Laufzeitfehlermeldungen und Mitteilungen auf der Standardausgabe werden in einem zweiten Rahmen angezeigt. In



dem dargestellten Quelltext kommt lediglich die Methode »main« vor, die jede Applikation enthalten muss. Diese Methode enthält nur vier Schreibebefehle auf die Standardausgabe: In den Zeichenketten sind entsprechende Steuerbefehle für Tabulator und Zeilenschaltungen eingebettet. Im unteren Rahmen ist die daraus folgende Ausgabe erkennbar.

4. Unterrichtsbeispiel - Ausgeben einer »Fadengrafik«

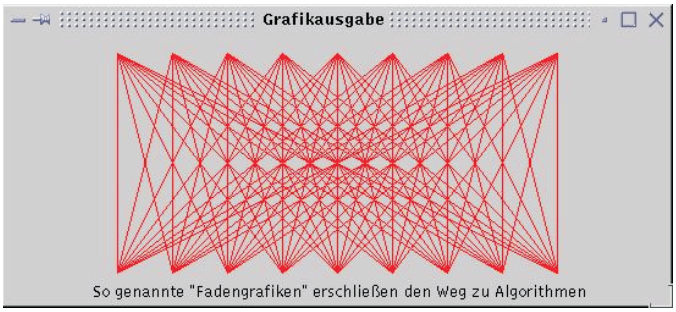
Sobald die grundlegenden Datenstrukturen und Algorithmen für Java bekannt sind, lassen sich einfache Beispiele zur Grafikausgabe realisieren. So genannte »Fadengrafiken« erlauben ein weites Feld ansprechender computergenerierter Grafiken - oft enthalten sie bereits interessante Algorithmen.

```
import java.awt.*;
public class grafik extends Frame {

    public static void main (String args[]) {
        grafik app = new grafik();
        app.resize(550,250);
        app.show();
    }

    grafik() {
      super("Grafikausgabe");
    }

    public void paint (Graphics g) {
      int i;
      int j;
      g.drawString("So genannte \"Fadengrafiken\" erschließen den
    Weg zu Algorithmen",75,240);
      g.setColor(Color.red);
      for (i=1;i<10;i++)
        for (j=1;j<10;j++)
          g.drawLine(50+i*45,40,500-j*45,220);
    }
}
```



Die eigene Anwendung ist eine Unterklasse der Klasse Frame - damit stellt das Package »Abstract Window Toolkit« von Java die betriebssystemüblichen Rahmen zur Darstellung eines Bildschirmfensters zur Verfügung. Es wird durch den import-Befehl zu Beginn des Quellcodes geladen. Im Hauptprogramm **main** wird lediglich ein Objekt **app** vom Typ der Klasse **grafik** erzeugt und gemäß dem Konstruktor **grafik()** initialisiert. In der Klasse grafik wird im Wesentlichen die Methode **paint()** aufgerufen, in der eine Zeichenkette und eine Reihe von geraden Linien auf das Grafikobjekt **g** ausgegeben werden.

Quellen

Java-Dokumentation von http://www.java.sun.com/

Laura Lemay, Rogers Cadenhead, »Teach Yourself Java 1.2 in 21 Days«, SAMS, ISBN 1-57521-390-7

Florian Hawlitzek, »Java 2«, Addison-Wesley (aus der Reihe »Nitty Gritty«), ISBN 3-8273-1671-5

Michael Kunzinger, Andreas Ulovec, Skriptum zur Vorlesung »Einführung in das Programmieren I« (Universität Wien, Institut für Mathematik)

Andreas Eberhart, Stefan Fischer, »Java-Bausteine für E-Commerce-Anwendungen«, Hanser, ISBN 3-446-21372-4

David Flanagan, »Java in a Nutshell«, O'Reilly, ISBN 3-89721-190-4 (deutsche Ausgabe)

David Flanagan, »Java Examples in a Nutshell«, O'Reilly, ISBN 3-89721-112-2 (deutsche Ausgabe)

Patrick Niemeyer, Jonathan Knudsen, »Learning Java«, O'Reilly, ISBN 1-56592-271-9