Java

Alfred Nussbaumer

Java vereint einige für den Informatikunterricht nicht unwichtige Eigenschaften:

- Java ist relativ einfach zu erlernen,
- Java erfordert eine strenge objektorientierte Programmierung, und
- Java ist plattformunabhängig.

Nach einer ersten Vorstellung elementarer Java-Programme (PCNEWS-72) sollen in diesem Artikel **Datentypen, Klassen**, **Objekte** und **Programmstrukuren** an Hand von Beispielen aus der Physik dargestellt werden.

1. Datentypen

Zur Verfügung stehen 8 einfache Datentypen:

Typ	Größe	Wertebereich
byte	8 bit	-128 .. 127
short	16 bit	-32 768 .. 32767
int	32 bit	-2 147 483 648 .. 2 147 483 647
long	64 bit	-9.223.372.036.854.775.808 9.223.372.036.854.775.807
float	32 bit	3.40282347E+38 .. 3.40282347E+38 Auf 8 Stellen genau
double	64 bit	-1.79769313486231570E+308 .. 1.79769313486231570E+308 auf 17 Stellen genau
char	1 byte	1 Zeichen
boolean	1 byte	false / true

Durch das Voranstellen eines Typs kann die Umwandlung in einen Datentyp erzwungen werden ("Type-Casting"):

```
float realzahl;
int ganzzahl;
ganzzahl = (int) realzahl;
```

2. Klassen und Objekte, Zeichenketten

Objektorientierte Programmierung setzt die Verwendung von Klassen voraus. Darunter verstehen wir bestimmte Eigenschaften, die zu einem benutzerdefinierten Datentyp zusammengefasst werden. Im Grunde genommen bildet jedes Java-Programm eine eigene Klasse, die durch das Hinzufügen der Methode `main()` ausführbar wird.

Klassen enthalten somit eine Reihe von **Variablen** und eine Reihe **Methoden**, die die Eigenschaften dieser Variablen verändern. Das Zusammenfassen von Variablen und Methoden in einer Klasse („Kapselung") und die Möglichkeit, diese Methoden weiteren Klassen zur Verfügung zu stellen („Vererbung") sind Teil der Grundlage der objektorientierten Programmierung. Alle Variablen und Methoden einer Klasse werden den Objekten zur Verfügung gestellt, die zur Laufzeit (mit dem Schlüsselwort `new`) von der Klasse abgeleitet werden. Von einer Klasse abgeleitete Objekte haben gewissermaßen den gleichen „Aufbau" und ein gleiches „Verhalten". Sie eignen sich in besonderer Weise, komplexe Datentypen zu entwerfen, die den jeweiligen Erfordernissen optimal angepasst sind.

Alle Zeichenketten werden in Java als eigene Objekte der Klasse `java.lang.string` abgeleitet. Aus diesem Grund stehen zahlreiche Methoden zur Verfügung, mit denen Zeichenketten bearbeitet werden können.

```
zeile = "Dies ist eine Zeichenkette";
System.out.println(zeile);
System.out.println(zeilc.length());
```

3. Programmstrukturen

Der Ablauf jedes Programmes ist - außer durch die **Sequenz** (einfache Aufeinanderfolge von Befehlen) durch **Verzweigungen** , **Schleifen** und **Unterprogramme** gekennzeichnet. Neben Unterprogrammen stehen in Java folgende Strukturen zur Verfügung:

3.1 Verzweigung

Die `if`-Anweisung wird in der folgenden (allgemeinen) Form kodiert:

```
if (logischer Ausdruck)
  Anweisung1;
else
  Anweisung2;
```

Der `else`-Zweig kann auch fehlen, `if`-Verzweigungen können geschachtelt werden.

Logische Ausdrücke können logische Werte, logische Variable oder Ausdrücke mit logischen Operatoren sein. Abgefragt wird jeweils, ob der logische Ausdruck wahr oder falsch ist.

3.2 Mehrfachverzweigung

Eine Verzweigung auf mehrere Möglichkeiten wird mit der Switch-Anweisung realisiert...

```
switch (Ausdruck) {
   case Wert1: Anweisung1; break;
   case Wert2: Anweisung2; break;
   ...
   default: Anweisung;
 }
```

3.3 Schleifen

Zählschleifen

```
for (Initialisierung; Abbruchbedingung;Inkrement) {
   Anweisung(en);
}
```

(fußgesteuerte) do-while-Schleife

```
do { Anweisungen }
while (logischer Ausdruck);
```

(kopfgesteuerte) while-Schleife

```
while (logischer Ausdruck) {
    Anweisung(en);
}
```

4. Unterrichtsbeispiele

Simulation von „Wurfbahnen"

Die Verwendung von Rechenanlagen hat sich seit langem für die numerische Behandlung von (mehr oder weniger einfachen) Fragestellungen aus der Physik bewährt. Im Beispiel berechnen wir eine Wurfbahn schrittweise:

```
import java.awt.*;
import java.awt.event.*;

public class wurfbahn extends Frame {

  wurfbahn() {
    super("Schiefer Wurf");
  }

  public void paint (Graphics g) {
    double v, vwind;
    double xalt,yalt,x,y,vx,vy,ax,ay,dt;
    g.setColor(Color.white);
    g.fillRect(0,0,600,600);
    g.setColor(Color.black);
    g.drawLine(0,200,600,200);
    g.setColor(Color.red);
    v = 65;
    vwind = 15;

    for (int i=0;i<7;i++) {
```

```
        dt=0.1;
        xalt=0;
        yalt=200;
        vx = v*Math.cos(-15*i*3.14/180);
        vy = v*Math.sin(-15*i*3.14/180);
        x=0;
        y=200;
        ax=0;
        ay=9.81;

        do {
            vx=vx+(ax-vwind*vwind*0.01)*dt;
            vy=vy+ay*dt;
            x=x+vx*dt;
            y=y+vy*dt;
            g.drawLine((int)xalt,(int)yalt,(int)x,(int)y);
            xalt=x;
            yalt=y;
        } while ((x<600) && (y<600) && (x>=0) && (y>=0));
      }
    }

    public static void main (String[] arguments) {
      wurfbahn proggi = new wurfbahn();
      WindowListener wl = new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
          System.exit(0);
        }
      };
      proggi.addWindowListener(wl);
      proggi.setLocation(100,100);
      proggi.setSize(600,600);
      proggi.show();
    }
}
```

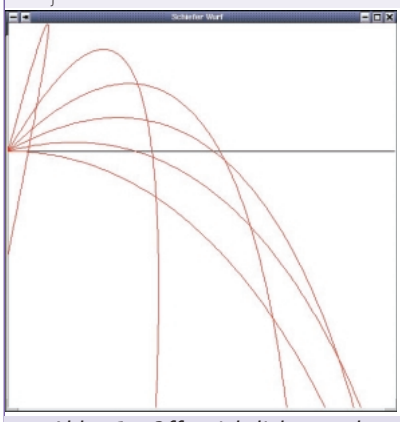

Abb. 1: Offensichtlich wurden Bahnen von Körpern berechnet, die aus einem Hochhaus unter verschiedenen Winkeln geworfen wurden. Der Gegenwind bewirkt „unsymmetrische" Bahnkurven...

Wegen der verwendeten mathematischen Funktionen (Sinus-, Kosinus- und Quadratwurzelfunktion aus dem Paket Math) verwenden wir innerhalb der Iteration Variable mit dem Datentyp „double". Da die Grafik-Methoden ganzzahlige Werte verlangen, werden die reellen Zahlen mittels Type-Casting in ganze Zahlen umgewandelt. Die Laufvariable für die Zählschleife ist vom Typ „integer". Lediglich der Titel des Rahmens ist eine Zeichenkette.

Die Zählschleife für die Wurfbahnen bei verschiedenen Anfangswinkeln enthält eine fußgesteuerte while-Schleife (Physik: vx = vx + a*dt, x = x + vx*dt). Das Berechnen und die Ausgabe der Wurfbahn wird abgebrochen, wenn die Werte außerhalb des Zeichenfensters liegen.

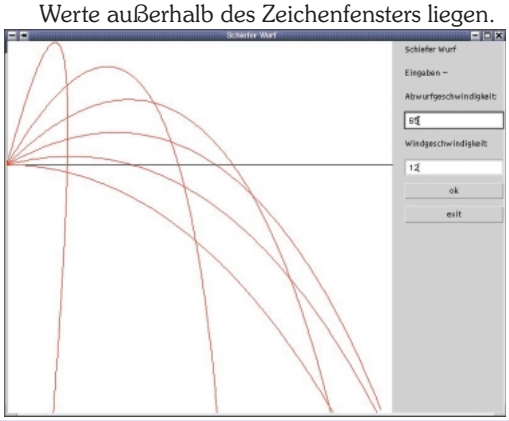

Abb. 2: Eingabefelder zur Wahl der Anfangswerte und Schaltflächen ergeben eine einfache Benutzerschnittstelle.

Bis auf die Möglichkeit, das Programm mit Hilfe der „Schließen-Methode" des Rahmens, bzw. mit der Tastenkombination + zu beenden, fehlt jede Möglichkeit der Interaktion durch den Benutzer. Dazu fehlen etwa Eingabefelder für die Anfangswerte und Schaltflächen zur Programmführung. Mit welchen Java-Klassen eine solche (einfache) Bedienungsoberfläche gestaltet werden kann soll in einer weiteren Folge dargestellt werden.

Unterrichtsbeispiel - Simulation einer Satellitenbahn

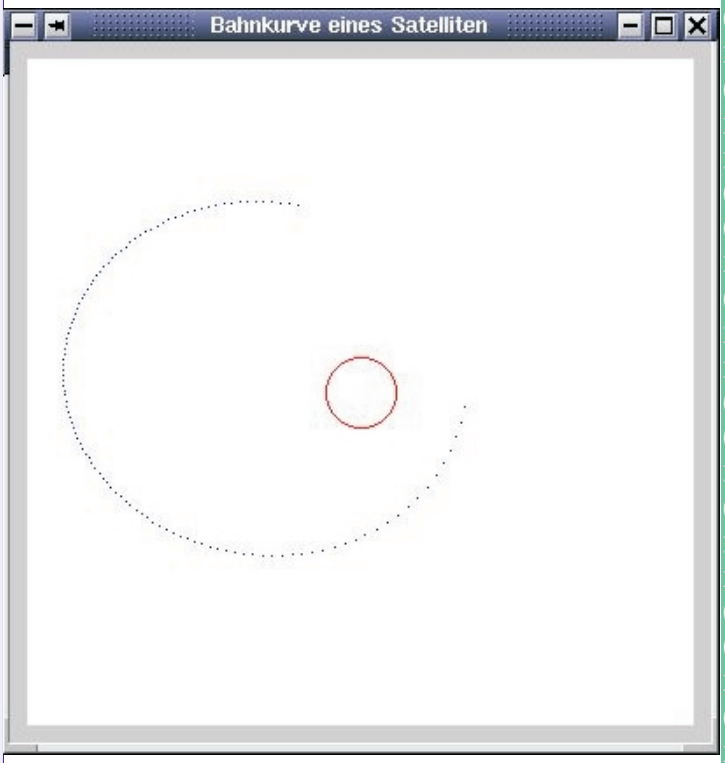

Abb. 3: Die Bahn eines Satelliten um die Erde wird punktweise berechnet und ausgegeben.

```
...
    public void paint (Graphics g) {
     double GM,r,r3,er;
       double x,y,vx,vy,ax,ay,dt,alpha;
     int zaehler;
     zaehler = 120;
     GM = 392000000000000.0;
     er = 6370000;
     alpha = 0;
     vx = 0;
     vy = 5500;
     dt = 500;
     x = 3*er;;
     y = 0;
     g.setColor(Color.white);
     g.fillRect(10,10,380,380);
     g.setColor(Color.red);
     g.drawOval(200 - (int)(er/(20*er)*400),
             200 - (int)(er/(20*er)*400),
             (int)(er/(20*er)*800),
             (int)(er/(20*er)*800));
     g.setColor(Color.blue);

     for (int i=1;i<zaehler;i++) {
         r3 = Math.sqrt(x*x+y*y);
         r=r3*r3*r3;
         ax=-GM*x/r;
         ay=-GM*y/r;
         vx=vx+ax*dt;
         vy=vy+ay*dt;
         x=x+vx*dt;
         y=y+vy*dt;
         g.fillRect (200+(int)(x/(20*er)*400),
                 200+(int)(y/(20*er)*400),1,1);
     }
    }
...
```

Die Variable "zaehler" legt den Abbruch fest: Nach 120 Rechenschritten endet die for-Schleife.

GM=392000000000000.0 - die Angabe der Dezimalzahl mit der Nachkommastelle "0" legt das Datenformat "double" fest (andernfalls erhält man einen Zahlenüberlauf).

Maßstab - die (in Vielfachen des Erdradius) berechneten Wegstrecken werden zunächst durch den 20-fachen Erdradius dividiert (= Pixel-Größe) und anschließend mit der Zahl der zur Verfügung stehenden Pixel multipliziert.

Der Nullpunkt des Koordinaten-Systems wurde in das Zentrum der Erde verlegt, die in der Mitte des Rahmens (an der Position x=200, y=200) gezeichnet wird.