JAVA

Alfred Nussbaumer

Java erfordert eine strenge objektorientierte Programmierung. In diesem Beitrag sollen daher Beispiele vorgestellt werden, die die Verwendung von Objekten bereits bei winzigen Anwendungen sinnvoll erscheinen lassen: Eigene Anwendungen werden häufig als Erweiterung bestehender, mächtiger Java-Klassen geschrieben (z.B. die Klasse "Frame" für Applikationen, die Klasse "Applet" für Applets).Nachdem in den ersten beiden Beiträgen (vgl. **PCNEWS**-72 und **PCNEWS**-73) ausschließlich Java-Applikationen realisiert wurden, sollen hier Java-Applets verwendet werden, die innerhalb der jeweiligen Browser-Umgebung ablaufen.

1. Applets

Applets werden als Teil einer Web-Seite vom jeweiligen Browser ausgeführt - falls das Ausführen von Java-Byte-Code nicht aus "Sicherheitsgründen" gesperrt wurde. Alle Applets sind Unterklassen der Klasse "Applet", die im Package "java.applet" enthalten ist. Wird also eine Unterklasse von "Applet" erzeugt, so werden automatisch alle Methoden vererbt, die notwendig sind, damit ein Java-Programm als Applet im Rahmen einer Web-Seite ausgeführt werden kann.

Applets werden also zunächst mit einem Editor erstellt, dann mit dem Javacompiler übersetzt und schließlich von der Java-Laufzeit-Umgebung eines Webbrowsers ausgeführt. Auch hier gilt: Der Name des Applets muss mit dem Namen der Quelldatei übereinstimmen.(Zum Entwickeln und Testen von Java-Applets ist es allerdings günstiger, den sogenannten "Appletviewer" zum Ausführen des erstellten Applets zu verwenden.)

Das fertige Applet wird mit dem Applet-Tag in ein HTML-Dokument eingebunden:

```
...
<applet code = "grafapp.class"
        height = "300"
        width = "300">
</applet>
...
```

Damit kann ungefähr folgende Vorgangsweise beim Erstellen von Applets (je nach Betriebssystem und Entwicklungsumgebung) gewählt werden:

1. **edit grafapp.java**
2. **javac grafapp.java**
3. **appletviewer grafapp..html** (enthält den Aufruf der erzeugten Klasse grafapp.class).

Erstellt und kompiliert man den Java-Code mit Hilfe des mächtigen Editors "Emacs", so muss eine Datei "index.html" angelegt werden, die den Aufruf des Applets enthält. Nach dem Kompilieren kann dann das Applet aus der "JDE".

gestartet werden.Aus Sicherheitsgründen gelten für (unsignierte) Applets Einschränkungen im Vergleich zu Applikationen - immerhin sollte jeglicher Java-Code, der auf einem PC vorliegt als sicher gelten, während jeder Java-Code, der über Netzwerke geladen wird, grundsätzlich als unsicher einzustufen ist. Nach dem so genannten »Java in a Sandbox«-Prinzip sind daher für Applets u.a. folgende Aktionen unzulässig:

1. Applets dürfen keine Dateien auf der Festplatte schreiben oder lesen (gilt auch für Verzeichnisse und Druckerausgabe).
2. Applets dürfen keine Netzwerksverbindungen (sockets) aufbauen.
3. Applets dürfen keine lokalen Prozesse starten (z.B. LinuX: **rm ***, **forking**, DOS: DLLs, **format.com** etc.)

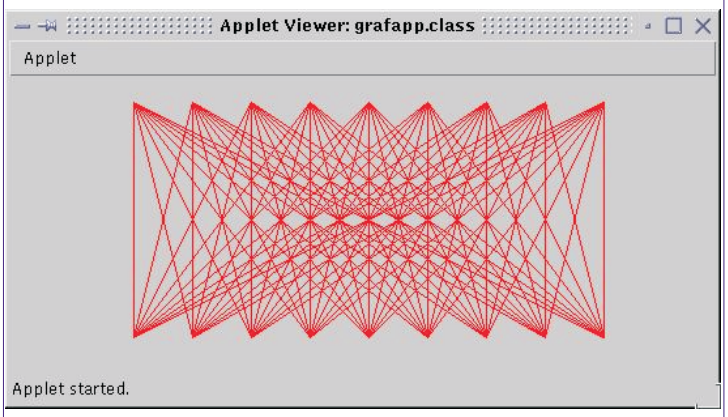
Applets werden als Erweiterungen der Klasse »Applet« codiert - der übrige Programmcode deckt sich weitgehend mit dem Code für Applikationen. Applets brauchen (als Erweiterung von `java.applet.Applet`) kein eigenenes Hauptprogramm mehr - bestimmte Methoden werden beim Laden des Applets in den Browser automatisch aufgerufen.

Beispiel: Fadengrafik

```
import java.awt.*;

public class grafapp
             extends java.applet.Applet {

  public void paint (Graphics g) {
    int i;
    int j;
    g.setColor(Color.red);
    for (i=1;i<10;i++) {
      for (j=1;j<10;j++) {
        g.drawLine(50+i*45,20,500-j*45,200);
      }
    }
  }
}
```



Vergleiche den Code dieses Applets mit dem Code des Beispiels von **PCNEWS** 72!

Beim Ablauf eines Applets im WebBrowser werden der Reihe nach folgende Methoden abgearbeitet:

`init()`	beim Laden des Applets wird vom Browser diese Methode aufgerufen. Falls in einem Applet Initialisierungen notwendig sind, können sie hier codiert werden.
start()	wird jedesmal aufgerufen, wenn das Applet wieder sichtbar wird (z.B. neuer Bildschirmaufbau).
paint()	wird automatisch aufgerufen, um das Applet zu »zeichnen«. Beim Überschreiben dieser Methode wird im Wesentlichen das Applet »entwickelt«. Falls während des Programmlaufes etwas Neues gezeichnet werden soll, ist die Methode `repaint()` zu verwenden.
update()	wird vom Browser aufgerufen, wenn Änderungen auszugeben sind.
stop()	Beendet die Ausführung des Applets (etwa wenn die Ausgabefläche in den Hintergrund verschoben wird)
destroy()	wird beim Beenden des Applets ausgeführt; dabei wird im Allgemeinen ein sauberes »Aufräumen« des Hauptspeichers sicher gestellt.

Die meisten von der Klasse »Applet« zur Verfügung gestellten Methoden werden fertig in das eigene Applet übernommen. Nur die Methoden `init()` und `paint()` eigenen sich in besonderer Weise beim Erstellen eigener Applets.

Applets eigenen sich in besonderer Weise für animierte Teile einer Webseite, oder wenn eine Anwendung auf Benutzereingaben reagieren soll. Sollen Daten gedruckt, gespeichert oder während der Anwendung zum WebServer übertragen werden (etwa bei Homebanking-Anwendungen), sind kompliziertere Mechanismen zu realisieren (z. B. signierte Applets, Servlets).

2. Objektorientierte Programmierung

Bei der Durchsicht des letzten Beispiels fällt der objektorientierte Ansatz von Java sofort ins Auge:

```
public class grafapp
             extends java.applet.Applet {
```

Jede eigene Benutzeranwendung ist eine Klasse, die gegebenenfalls eine schon bestehende Klasse erweitert (in diesem Beispiel ist die Klasse Applet die Super-klasse der Klasse `grafapp`), die Klasse `grafapp` eine abgeleitete Klasse, die alle Methoden der Superklasse zur Verfügung hat.

```
  public void paint (Graphics g) {
    ...
    g.setColor(Color.red);
```

Das Applet verwendet hier die wichtige Methode `paint()`, mit der die Bildschirmausgabe geregelt wird. Die Methode `paint()` arbeitet mit dem Objekt `g` der Klasse `Graphics`, die alle wichtigen Grafikbefehle enthält - hier etwa das Setzen der Zeichenfarbe. Beachte die korrekte Referenzierung! Für die Grafik wird schließlich noch die Klasse `Color` verwendet, die die übliche RGB-Farbdarstellung zur Verfügung stellt (in diesem Beispiel die Farbkonstante `Color.red`).

3. Beispiel - Wurfbahn „interaktiv“

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

public class wurfbahnapp extends Applet
     implements ActionListener {

  TextField vein;
  TextField vwindein;
  Button ok;
  Button close;
  double v;
  double vwind;

  public void init() {
    setLayout(new BorderLayout());
    Panel panel = new Panel();
    panel.setLayout(new GridLayout(12,1,10,5));
    Label header = new Label("Schiefer Wurf");
    panel.add(header);
    Label text = new Label("Eingaben - ");
    panel.add(text);
    Label label1 = new
      Label("Abwurfgeschwindigkeit:");
    panel.add(label1);
    vein = new TextField("70",8);
    panel.add(vein);
    Label label2 = new
       Label("Windgeschwindigkeit:");
    panel.add(label2);
    vwindein = new TextField("20",8);
    panel.add(vwindein);
    ok = new Button("ok");
    panel.add(ok);
    ok.addActionListener(this);
    close = new Button("exit");
    panel.add(close);
    close.addActionListener(this);
    add("East",panel);
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == ok) {
      werteuebernehmen();
      repaint();
    }
    if (e.getSource() == close) {
      System.exit(0);
    }
  }

  public void werteuebernehmen() {
    v = Double.valueOf(vein.getText()).doubleValue();
    vwind = Double.valueOf(vwindein.getText()).doubleValue();
  }

  public void paint (Graphics g) {
    double xalt,yalt,x,y,vx,vy,ax,ay,dt;
    g.setColor(Color.white);
    g.fillRect(0,0,620,600);
    g.setColor(Color.black);
    g.drawLine(0,200,620,200);
    g.setColor(Color.red);
    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));
    }
  }
}
```

Bei einem Applet liegt kein Konstruktor wie bei Applikationen vor. Da die Methode `init()` beim Laden des Applets auf jeden Fall ausgeführt wird, eignet sie sich zum Festlegen des Layouts. Beachte, dass der so genannte „ActionListener“ mit den beiden Schaltflächen („Buttons“) verbunden werden muss.

Die beiden Schaltflächen ***„ok“*** und ***„exit“*** lösen einen so genannten ***„Event“*** aus, der mit Hilfe der Methode `actionPerformed()` abgearbeitet wird. Innerhalb dieser Methode wird mit Hilfe von `if`-Abfragen der entsprechende Programmteil aufgerufen.

Die Klasse `wurfbahnapp` verwendet das sogenannte „Interface“ `ActionListener`, das die Methoden festlegt, die für die Interaktion von Schaltflächen verwendet werden. Interfaces sind Klassen, deren Methoden nur „abstrakt“ definiert sind - die Implementierung muss der Anwender innerhalb der eigenen Klasse selbst vornehmen - vergleiche den Programmcode zur Methode `actionPerformed()`. Dieser Sachverhalt wird im Kopf der Klasse durch die Ergänzung `„implements ActionListener"` deutlich.

Wurfbahnen: Neben statischem Text im HTML-Code der Webseite erlaubt das Applet dem Benutzer Abwurf- und Gegenwindgeschwindigkeit einzugeben. Die Wurfbahnen werden nach dem Mausklick auf die Schaltfläche “ok” für sechs verschiedene Abwurfwinkel gezeichnet.