Lebenszyklus eines Java-Applets

Jedes Java-Applet besitzt einen eigenen Lebenszyklus. Das Applet muss zunächst erstellt werden, danach wird es initialisiert, gestartet, gestoppt und schließlich wieder zerstört. Für jede dieser Phasen gibt es eine Methode, in der man Code ausführen kann:

- Der Konstruktor ist für die Erzeugung des Applets verantwortlich.
- Während der Initialisierung wird die Methode `public void init()` aufgerufen. In ihr kann man z.B. Variablen mit einem Wert belegen oder Objekte erzeugen. Außerdem kann man Parameter von der HTML-Seite einlesen, auf der sich das Applet befindet.
- In der Startphase wird die Methode `start()` aufgerufen. In dieser Methode können resourcenhungrige Vorgänge gestartet werden, z.B. Musikstücke oder einen Timer, der regelmäßig etwas zeichnet.
- Das Applet wird gestoppt (Methode `stop()` aufgerufen), wenn im Browser die Webseite nicht mehr sichtbar ist, z.B. wenn eine neue Webseite aufgerufen wird. Das Applet wird aber noch nicht zerstört.
  In der Methode `stop()` sollten die in der Methode `start()` begonnenen Vorgänge wieder angehalten werden. Wenn man dann auf die Webseite mit dem Applet zurückkehrt, wird nur noch die Methode start() aufgerufen. Ein schönes Beispiel ist das Abspielen eines Musikstücks: Wenn man eine Webseite betritt, wird die Musik gestartet und läuft so lange, bis man sie verlässt.
- Kurz bevor das Applet zerstört wird, wird die Methode `destroy()` aufgerufen, in der man noch irgendwelche Resourcen freigeben kann. Normalerweise wird diese Methode aber nicht benötigt. Bei vielen der heutigen Browser wird ein Applet erst zerstört, wenn der Browser beendet wird.

Neben diesen Methoden, gibt es noch die `paint()`-Methode, die immer dann aufgerufen wird, wenn das Applet neu gezeichnet werden muss, z.B. wenn die Größe des Browsers geändert wird oder wenn das Applet durch andere Fenster verdeckt wird.

Ereignisse in Java

Wenn man Programme in Java mit grafischen Benutzeroberflächen erstellt, z.B. Applets, muss sich an eine andere Denkweise wie bisher gewöhnen. Statt linearer Programmabläufe muss man jetzt auf sogenannte Ereignisse (Events) reagieren.

Wenn eine Taste, z. B. „x“, gedrückt wird, erhält man in Java mehrere Ereignisse: „Taste x heruntergedrückt“, „Taste x getippt“ und „Taste x losgelassen“. Ganz ähnlich sieht es aus, wenn eine Maustaste gedrückt wird. Man erhält die folgenden Ereignisse: „Maustaste heruntergedrückt“, „Maustaste losgelassen“.

Java hat für die Verarbeitung von Ereignissen ein spezielles Konzept, und zwar das der „Listener“ bzw. auf Deutsch „Zuhörer“. Wenn man an einem Ereignis interessiert, muss hierfür ein Listener zur Verfügung gestellt werden. Ohne Zuhörer wird das Auftreten des Ereignisses gar nicht erst gemeldet, da sich offensichtlich sowieso niemand dafür interessiert.

Insbesondere bei Applets findet man im Internet auch noch andere Möglichkeiten, auf Ereignisse zu reagieren. Diese sind jedoch veraltet und sollten nicht mehr verwendet werden, auch wenn sie auf den ersten Blick einfacher erscheinen. Das Konzept der Listener wird in den neueren Java-Versionen (ab 1.1) konsequent durchgehalten und man kann es auf allen anderen grafischen Komponenten, wie Schaltflächen übertragen.

Am einfachsten ist es, wenn man sich ein kleines Beispiel für die Ereignisbehandlung ansieht:


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

public class EventsDemo extends JApplet {

  // Applet konstruieren
  public EventsDemo() {
    // um in Java auf Ereignisse zu reagieren, muss ein Listener zur Klasse
    // hinzugefügt werden
    this.addMouseListener(new EventsDemo_mouseAdapter(this));  // (1)
  }

  // Ereignisbehandlung für das Drücken einer Maustaste
  public void mousePressed(MouseEvent e) {
      JOptionPane.showMessageDialog(this,
        "Maustaste wurde gedrückt an (" + e.getX() +"|" + e.getY() + ")");
  }
}

// Hilfsklasse zum Reagieren auf Mausereignisse
class EventsDemo_mouseAdapter extends MouseAdapter {
  EventsDemo Referenz;  // (2)

  EventsDemo_mouseAdapter(EventsDemo Referenz) {
    this.Referenz = Referenz;
  }

  public void mousePressed(MouseEvent e) {
    Referenz.mousePressed(e);  // (3)
  }
}
```

An der Stelle (1) wird ein Maus-Listener hinzugefügt. Der Maus-Listener wird durch die unten definierte Hilfsklasse EventsDemo_mouseAdapter zur Verfügung gestellt. Dem Konstruktor der Hilfsklasse wird eine Referenz auf die aktuelle Klasse übergeben und in der Eigenschaft Referenz (siehe (2)) gespeichert, damit man später in der Hilfsklasse auf die eigentliche Klasse zurückgreifen kann (siehe (3)).

Wenn jetzt eine Maustaste gedrückt wird, ruft Java den entsprechenden Listener auf, hier die Hilfsklasse `EventsDemo_mouseAdapter`. In dieser Klasse wird eine Methode `public void mousePressed(MouseEvent e)` gesucht und schließlich aufgerufen. In unserem Beispiel wird in dieser Methode die Methode `mousePressed(MouseEvent e)` des Applets aufgerufen. Natürlich hätte die Bearbeiten des Ereignisses auch direkt in der Hilfsklasse erfolgen können, wenn keine Eigenschaften des Applets gebraucht werden.

In der Methode `public void mousePressed(MouseEvent e)` wird mit Hilfe von `JOptionPane.showMessageDialog` ein Meldungsfenster angezeigt. Als 1. Parameter wird eine Referenz auf das Applet übergeben, als zweiter der auszugebende Text.

Dabei werden Methoden des Objektes `e` der Klasse `MouseEvent` verwendet, um die Koordinaten des Mausklicks festzustellen. Neben den Koordinaten werden noch weitere nützliche Informationen über das Ereignis übertragen, die man in der Hilfe nachlesen kann.

Es gibt auch noch weitere Methoden Listener zu erzeugen. Auf die wird aber erst später eingegangen.

Aufgaben:

1. Verändere das Applet EventsDemo so, dass es nur noch auf den Tastendruck reagiert, wenn die linke Maustaste gedrückt wird. (siehe Java-Hilfe!)
2. Erstelle ein Applet, in dem Man mit der Maus zeichnen kann. Ein Beispiel, wie ein solches Applet funktionieren kann, findest du auf der Webseite zum Kurs.