Einstieg in die Informatik mit Java

Ereignisverwaltung und Layout-Typen

Gerd Bohlender

Institut für Angewandte und Numerische Mathematik

Gliederung

Gliederung

Ereignis-Verwaltung

Eine *Komponente* wie ein Button, Scrollbar, Textfield kann mit Hilfe der Methode

```
add()
```

der Umgebung (z.B. dem Applet) hinzugefügt werden.

Diese Komponente erzeugt *Ereignisse* (Mausereignisse werden vom Applet erzeugt), die über den Browser oder den `appletviewer` ausgelöst werden.

Eine selbstdefinierte Klasse implementiert den entsprechenden *Listener* bzw. ist abgeleitet von entsprechenden *Adaptern* (s.u.), d.h. die Klasse implementiert z.B. den `ActionListener` und muss deswegen alle Methoden dieser Schnittstelle implementieren. Das bedeutet im Falle des `MouseListeners`, dass fünf Methoden überschrieben werden müssen.

Ereignis-Verwaltung

Abhilfe

Es werden spezielle Adapterklassen mit leeren Implementierungen definiert, z. B. `MouseAdapter`, von diesen können mit Hilfe der Vererbung eigene Klassen abgeleitet werden, wobei nur die nötigen Methoden überschrieben werden müssen.

Die selbstdefinierte Klasse muss per

```
addXYZListener(...);
```

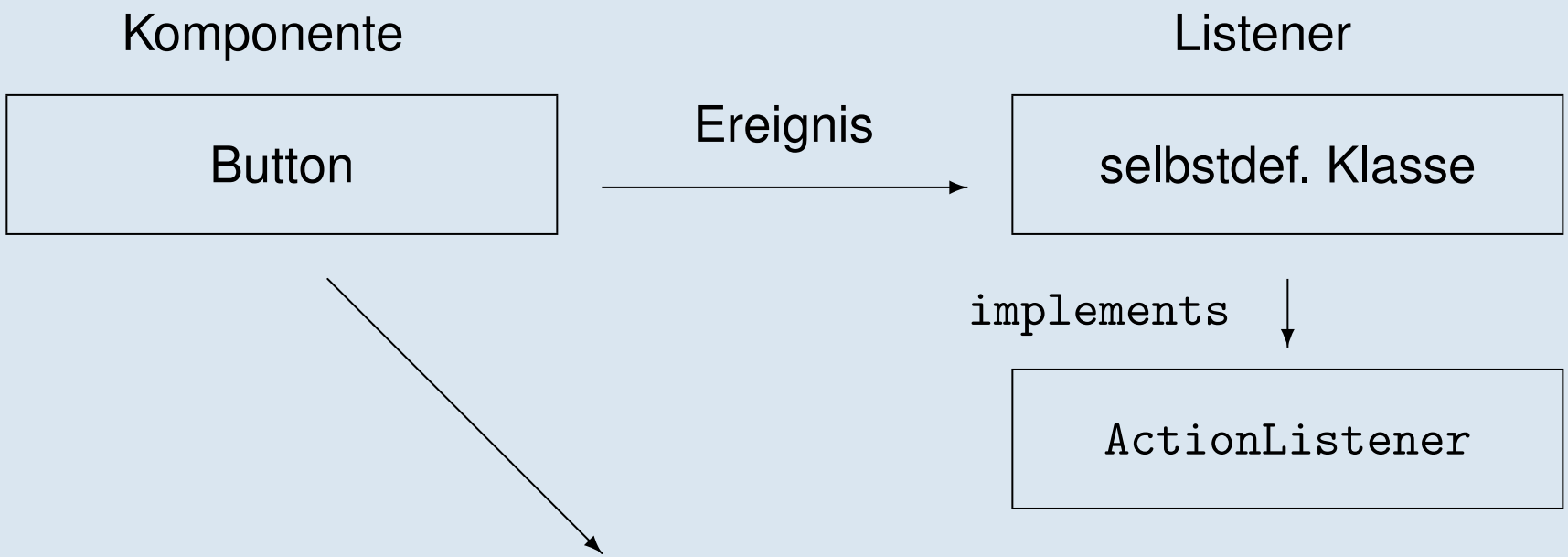
bei der Komponente registriert werden, damit sie von Ereignissen benachrichtigt wird, d.h. damit die entsprechenden Methoden in ihr aufgerufen werden.

Das Abmelden funktioniert analog mit

```
removeXYZListener(...);
```

Ereignis-Verwaltung

Trennung von graphischer Schnittstelle und Ereignisverarbeitung



Beispiel

Erstelle ein Zeichenprogramm, das Linien darstellt. Die Eingabe der Linien soll mit der Maus erfolgen.

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

public class LinienMaus extends Applet {
     private int beginx, beginy, endx, endy;
     public void init () {
          MausEreignis erg = new MausEreignis ();
          addMouseListener (erg);
          addMouseMotionListener (erg);
          setBackground (Color.lightGray);
          setForeground (Color.red);
     }
     public void paint (Graphics g) {
          g.drawLine (beginx, beginy, endx, endy);
     }

     class MausEreignis extends MouseAdapter
          implements MouseMotionListener {
               public void mousePressed (MouseEvent e) {
                    beginx = e.getX (); // Methode 1
                    beginy = e.getY ();
               }
               public void mouseDragged (MouseEvent e) {
                    endx = e.getX (); // Methode 2
                    endy = e.getY ();
                    repaint ();
               }
               public void mouseMoved (MouseEvent e) {
                    // Methode 3
               }
     }
}
```

Ereignis-Verwaltung

Achtung

Die Variante mit einer anonymer Klasse geht hier schlecht, weil der Name `erg` zweimal benötigt wird, d.h. Aufspalten in:

```
addMouseListener (
new MouseAdapter() { Methode 1 } );
addMouseMotionListener (
new MouseMotionListener() { Methode 2+3} );
addMouseMotionListener (
new MouseMotionAdapter() { Methode 3} );
```

Gliederung

Layout-Typen

Komponenten wie Buttons, Scrollbars usw. können durch den Methodenaufruf

```
setLayout(LayoutManager);
```

im Applet angeordnet werden. Im folgenden werden beispielhaft die Layoutmanager `GridLayout`, `FlowLayout` und `null` vorgestellt.

GridLayout

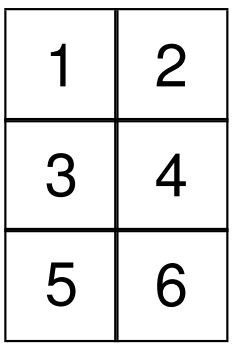
Beim Layoutmanager `GridLayout` werden die Komponenten eines Applets fortlaufend in einem rechteckigen Gitter angeordnet, wobei jedes Rechteck des Gitters genau eine Komponente erhält.

Beispiel

Anordnung von sechs Buttons in drei Reihen und zwei Spalten.

```
import java.awt.*;
import java.applet.Applet;
public class ButtonGrid extends Applet {
     public void init () {
          setLayout (new GridLayout(3,2));
          add (new Button ("1"));
          add (new Button ("2"));
          add (new Button ("3"));
          add (new Button ("4"));
          add (new Button ("5"));
          add (new Button ("6"));
     }
}
```

Erscheinungsbild im Applet



FlowLayout

Beim Layoutmanager `FlowLayout` werden die Komponenten fließend von links nach rechts angeordnet. Sobald eine Komponente platzmäßig nicht mehr in eine Zeile passt, wird diese umgebrochen und die Komponente wird in die nächste Zeile gesetzt. Dabei werden die Komponenten in jeder Zeile zentriert.

Achtung

Dieser Layoutmanager ist das Standardlayout jedes Applets, d.h. es braucht nicht explizit durch `setLayout()` gesetzt werden!

`null` Layout

Durch Setzen eines `null` Layouts müssen die Positionen und die Größe der Komponenten manuell durch die Methode

```
setBounds(int x, int y, int breite, int hoehe);
```

gesetzt werden. Damit erhält jede Komponente eine feste Position in der Appletumgebung, d.h. selbst bei Änderung der Fenstergröße des Applets bleiben die Komponenten an ihren festen Positionen.

Achtung

Im Gegensatz zum `FlowLayout` werden die Komponenten beim `null`-Layout nicht umgebrochen!

null Layout

Beispiel

```
import java.applet.*;
public class LayBsp extends Applet {
    Button b;
    public void init () {
        setLayout (null);
        add (b=new Button("Test"));
        b.setBounds (10, 10, 100, 50);
    }
}
```