Fenster in Java

AWT: Abstract Window Toolkit

Für den Programmierer:

Unabhängig von Betriebssystem und Fenstersystem.

Für den Benutzer:

Im wesentlichen: eigenes „look-and-feel“

Geschichte:

In Java 1.0: erste Version.

In Java 1.1: grundlegend geändert.

In Java 1.2: durch Swing ersetzt.

Ereignisgesteuerte Programmierung

Fensterkomponenten produzieren „Ereignisse“:

z. B.: Anklicken eines Buttons, Mausbewegungen, Tastendrücke.

Benutzer bestimmt, welche Ereignisse in welcher Reihenfolge stattfinden.

$\Rightarrow$ Programmierer hat weniger Kontrolle über Programmablauf.

$\Rightarrow$ unter Umständen: Programmkonzept anpassen.

Fensterkomponenten (I)

```
import java.awt.*;

...
Frame f;
Button b;
Checkbox c;
TextArea t;

f = new Frame();
b = new Button("Button");
c = new Checkbox("Checkbox", true);
t = new TextArea("TextArea",3,20);

f.setLayout(new FlowLayout());
f.add(b);
f.add(c);
f.add(t);
f.setSize(300,200);
f.setVisible(true);
```


Fensterkomponenten (II)

```
import java.awt.*;

...
Frame f;
Button b;
Checkbox c;
TextArea t;

f = new Frame();
b = new Button("Button");
c = new Checkbox("Checkbox", true);
t = new TextArea("TextArea",3,20);

f.setLayout(new BorderLayout());
f.add(b,"North");
f.add(c,"South");
f.add(t,"Center");
f.setSize(300,200);
f.setVisible(true);
```

Fensterkomponenten (III)

```
Frame f = new Frame();
Panel p = new Panel();
Label l = new Label("L");
Button b = new Button("B");
Checkbox c = new Checkbox("C", true);
TextArea t = new TextArea("TA",3,20);

f.setLayout(new BorderLayout());
p.setLayout(new FlowLayout
                   (FlowLayout.LEFT));
p.add(l);
p.add(b);
f.add(p,"North");
f.add(c,"South");
f.add(t,"Center");
f.setSize(300,200);
f.setVisible(true);
```


Fensterkomponenten (IV)

```
Frame f;
CheckboxGroup cg;
Checkbox c1, c2, c3;

f = new Frame();
cg = new CheckboxGroup();
c1 = new Checkbox("C1", true, cg);
c2 = new Checkbox("C2", false, cg);
c3 = new Checkbox("C3", false, cg);

f.setLayout(new FlowLayout());
f.add(c1);
f.add(c2);
f.add(c3);
f.setSize(300,200);
f.setVisible(true);
```


Fensterkomponenten (V)

```
Frame f;
Choice c;           // oder: List c;

f = new Frame();
c = new Choice();   // oder: new List();

c.add("Rot");
c.add("Gelb");
c.add("Grün");

f.setLayout(new FlowLayout());
f.add(c);
f.setSize(300,200);
f.setVisible(true);
```


Fensterkomponenten (VI)

```
Frame f;
TextArea t;

f = new Frame();
t = new TextArea("TextArea", 3, 20);

t.setForeground(new Color(255, 64, 0));
t.setBackground(Color.white);

t.setFont(new Font
        ("SansSerif", Font.PLAIN, 14));

f.setLayout(new FlowLayout());
f.add(t);
f.setSize(300,200);
f.setVisible(true);
```

Fensterkomponenten (VII)

Sonstiges für `Component`:

- `setSize`, `getSize`
- `paint`, `update`, `repaint`
- `setEnabled`
- …

spezielle Methoden für einzelne Komponenten:

- `dispose`
- `getText`, `setText`
- `getFile`
- `setEditable`, `isEditable`
- …

Ereignisse (I)

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


...
Button b1 = new Button("B1");
b1.addActionListener(new B1L());
...


public class B1L
             implements ActionListener {
  public void
      actionPerformed (ActionEvent e) {
     ...
  }
}
```


Ereignisse (II)

```
public class MyFrame extends Frame {
  Button b1 = new Button("B1");

  public MyFrame() {
    setLayout(new FlowLayout());
    b1.addActionListener(new B1L());
    add(b1);
  }

  public class B1L implements ActionListener {
    public void actionPerformed (ActionEvent e) {
      System.out.println("B1 gedrückt");
    }
  }

  public static void main(String[] args) {
    Frame f = new MyFrame();
    f.setSize(300,200);
    f.setVisible(true);
  }
}
```


Ereignisse (III)

```
public static class WL
           implements WindowListener {

  public void windowClosing (WindowEvent e) {
      System.out.println("Ende");
      System.exit(0);
  }

  public void windowClosed (WindowEvent e) {}
  public void windowOpened (WindowEvent e) {}
  public void windowIconified (WindowEvent e) {}
  public void windowDeiconified (WindowEvent e) {}
  public void windowActivated (WindowEvent e) {}
  public void windowDeactivated (WindowEvent e) {}
}

...
f.addWindowListener(new WL());
...
```

Ereignisse (IV)

```
public static class WL
               extends WindowAdapter {
  public void windowClosing
                    (WindowEvent e) {
      System.out.println("Ende");
      System.exit(0);
  }
}


...
f.addWindowListener(new WL());
...
```


Ereignisse (V)

```
...
f.addWindowListener(
               new WindowAdapter() {
  public void windowClosing
                  (WindowEvent e) {
    System.out.println("Ende");
    System.exit(0);
  }
});
...
```


Ereignisse (VI)

Alternativ, statt jeder Komponente ihre eigene Listener-Klasse zuzuordnen:

```
  public class BL
            implements ActionListener {
    public void actionPerformed
                    (ActionEvent e) {
        ... e.getSource() ...
      // liefert das Objekt, in dem
      // das Ereignis ausgelöst wurde.
    }
  }
```

(nur verwenden, wenn gleichartige Funktionalität)

Threads (I)

Thread (Faden):

vom Rest des Programmes unabhängig laufender Programmteil.

im Gegensatz zu Prozeß:
kein eigener Adreßraum.

Threads (II)

```
public class MyThr extends Thread {
  private int limit;
  public MyThr(int i) { lim = i; }
  public void run() {
    for (int i = 0;  i < lim;  i++) {
      System.out.println(i+"->"+lim);
    }
  }
}

...
public static void main(String[] args){
  MyThr t1 = new MyThr(10);
  MyThr t2 = new MyThr(30);
  t1.start();
  t2.start();
}
...
```


Threads (III)

```
public class MyThr extends Thread {
  private int limit;
  private boolean stop;
  public MyThr(int i) { lim = i; }
  public void terminate() {stop = true;}
  public void run() {
    int i;
    stop = false;
    for (i = 0; i < lim && !stop; i++) {
      try {
        sleep(100); // Millisekunden
      } catch (InterruptedException e){}
      System.out.println(i+"->"+lim);
    }
  }
}
```


Threads (IV)

```
...
public static void main(String[] args) {
  MyThr t1 = new MyThr(10);
  MyThr t2 = new MyThr(30);
  t1.start();
  t2.start();
  try {
    t1.join();
  } catch (InterruptedException e) {}
  System.out.println("t1 beendet");
  t2.terminate();
  System.out.println("t2 gestoppt");
}
...
```