GUI und Applet-Programmierung

In dieser Übersicht wird nur AWT, nicht das bessere Swing behandelt!

Das Paktet java.awt

Erzeugung eines Fensters mit einfachen Komponenten

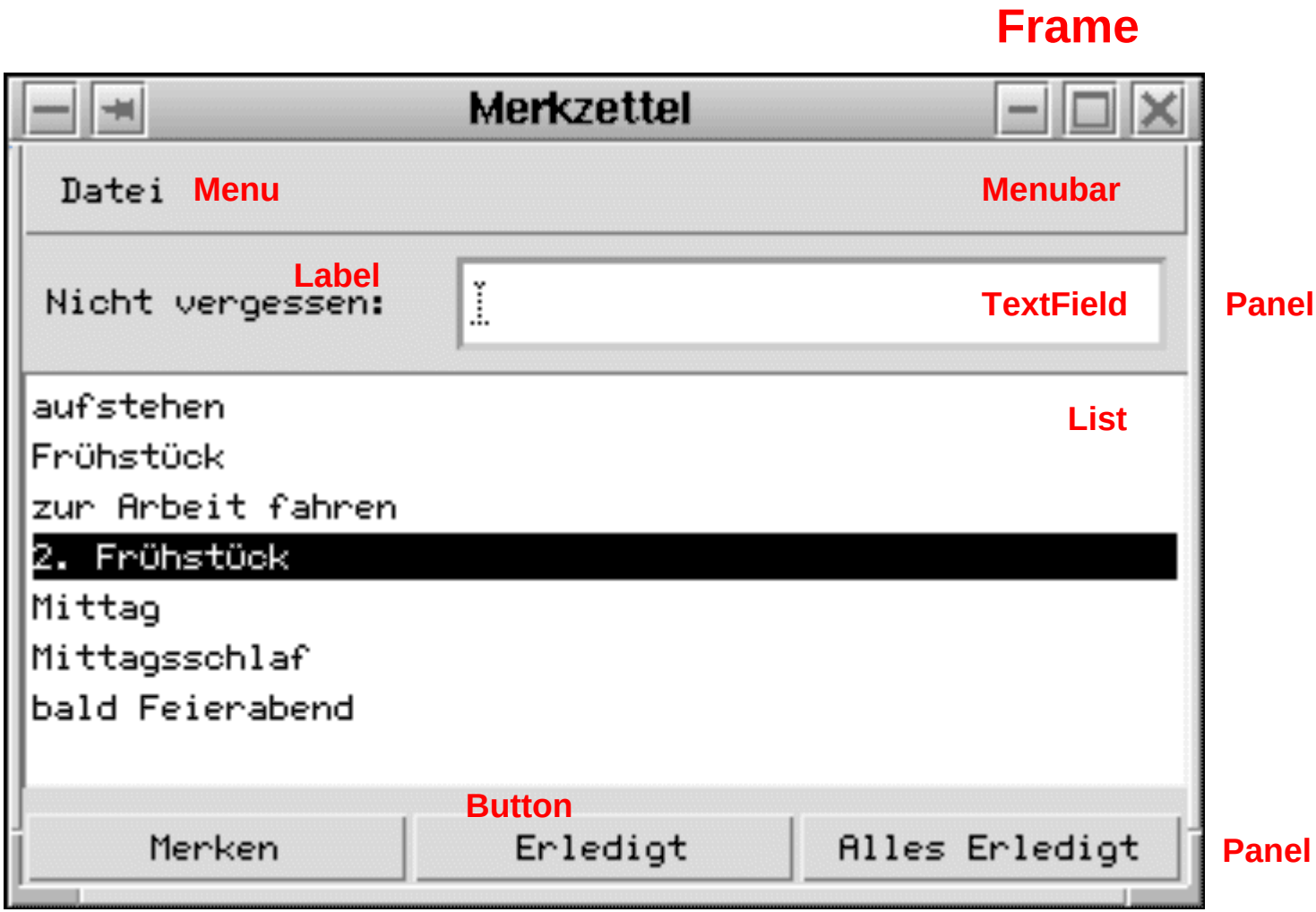
Das Ereignis-Konzept

Eine komplette Anwendung

Applets und HTML

Ein komplettes Beispiel

Komponenten einer GUI



Komponenten einer GUI(2)

Eine GUI enthält:

1. Einfache Komponenten

    Label = unveränderlicher Text

    Button = Aktionsknopf

    TextField, TextArea = Textverarbeitungsfeld

    List = Auswahlliste

    Canvas = Zeichenfläche

2. Behälter-Komponenten

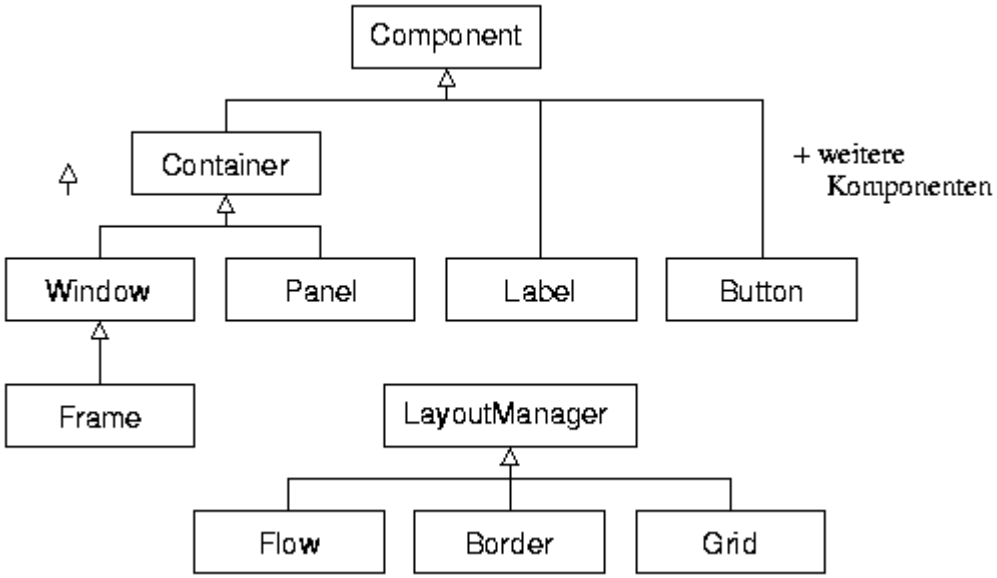
    Frame = Top-Level-Fenster

    Panel = Zusammenfassung von Teilkomponenten

3. Layoutregeln für Panels (wie werden die Komponenten angeordnet)

4. Aktionsobjekte, die den GUI-Aktionen zugeordnet sind.

Klassenhierarchie von java.awt



Frame, Panel: können andere Komponeten enthalten

Label, Button,... sind fertige Einzelkomponenten

FlowLayout:	Komponenten werden so wie es passt von links nach rechts und von oben nach unten angeordnet
BorderLayout::	Komponenten werden explizit an den Rändern plaziert (NORTH,,,,)
GridLayout:	Komponenten werden an vorgegebene Gitterpunkte gesetzt.

Einfachstes AWT-Fenster

```
import java.awt.*;

class SimpleGUI {
    SimpleGUI1(String message) {
        Frame f = new Frame("SimpleGUI");  // Fenster + Titel
        Label label1 = new Label(message); // Meldung im Fenster
        f.add(label1);                     // zum Fenster hinzufügen
        f.pack();                          // Fenster konfigurieren
        f.setVisible(true);                // Fenster darstellen
    }

    public static void main(String[] args) {
        new SimpleGUI1(args[0]);           // 1. GUI-Fenster anlegen
        new SimpleGUI1(args[1]);           // 2. GUI-Fenster anlegen
    }
}
```

Einfache Event-Behandlung (Fenster schließen)

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

class SimpleGUI2 {
    SimpleGUI2(String message) {
        Frame f = new Frame("SimpleGUI");
        f.addWindowListener(new WListener(f));
        Label label1 = new Label(message);
            f.add(label1);
        f.pack();
        f.setVisible(true);
    }
    ...
}
class WListener extends WindowAdapter { // implements WindowListener
    Frame f;                             // WindowAdapter hat Defaults
    WListener(Frame f) {                 // für den WindowListener
        this.f = f;
    }
    public void windowClosing(WindowEvent e) {
        f.dispose();
    }
}
```

Innere Klassen haben Zugriff auf Variable der umfassenden Klasse

```
class TopLevelClass {
    static Object staticVariable;
    Object instanceVariable;

    static class NestedClass {
        // Zugriff: staticVariable
    }

    class InnerClass {
        // Zugriff: staticVariable, instanceVariable
    }

    void Methode() {
        Object localVariable;
        final Object finalLocalVariable
        ...
        class LocalInnerClass (
            // Zugriff: staticVariable, instanceVariable,
            //          finalLocalVariable
        }
        ...
    }
```

Anonyme Klasse

Syntax: new *SuperKlasse* **() {** *Klassendefinition* **}**

Bedeutung: definiert eine Klasse, die von SuperKlasse abgeleitet ist (extends oder implements) und erzeugt ein Objekt.

Beispiel:

```
package java.awt.event;
interface ActionListener() {
    public void actionPerformed(ActionEvent e);
}

Button b = new Button("Quit");
b.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            ...
        }
    }
);
```

eine anonyme Klasse ist gleichzeitig eine innere Klasse.

GUI mit anonymen Klassen

```
import java.awt.*;
import java.awt.event.*;
class SimpleGUI3 {
    private static int nFrames = 0;
    SimpleGUI3(String message) {
        final Frame f = new Frame("SimpleGUI");
        f.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e) {
                f.dispose();
                if (--nFrames==0) System.exit(0);
            }
        });
        Label label1 = new Label(message);
            f.add(label1);
        Button b = new Button("Quit");
            b.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    f.dispose();
                    if (--nFrames==0) System.exit(0);
                }
            });
            f.add(b);
        f.pack();
        f.setVisible(true);
    }
}
```

Panel = Gruppierung von Komponenten
Layout = Anordnung von Komponenten

```
SimpleGUI4(String message) {
       final Frame f = new Frame("SimpleGUI");
       ...

       Panel p = new Panel(new BorderLayout());

       Label label1 = new Label(message);
           p.add(label1, BorderLayout.NORTH);

       Button b = new Button("Quit");
           b.addActionListener(new ActionListener() {
               public void actionPerformed(ActionEvent e) {
                   f.dispose();
                   if (--nFrames==0) System.exit(0);
               }
           });
           p.add(b, BorderLayout.SOUTH);

       f.add(p);
       f.pack();
       f.setVisible(true);
   }
```

Applets

1. Applets werden in HTML-Seiten dargestellt.
2. Applets werden durch HTML-Syntax erzeugt und mit Parametern versorgt
3. Applets werden in einem Browser (oder appletviewer) ausgeführt.
4. Applets werden von dem Web-Host geladen.
5. Applets unterliegen besonderen Sicherheitsanforderungen.
6. Die Klasse `java.lang.Applet` ist abgeleitet von `java.awt.Panel.`
7. Eigene Applets werden abgleitet von der Klasse `java.applet.Applet`.
8. Applets müssen besondere Methoden implementieren:
    1. **`init()`** wird aufgerufen, wenn das Applet erzeugt wird
    2. **`start()`** wird aufgerufen, wenn das Applet sichtbar wird
    3. **`stop()`** wird aufgerufen, wenn das Applet nicht mehr sichtbar ist
    4. **`destroy()`** wird aufgerufen, wenn das Applet verschwindet

Aufruf und Definition eines Applets

```
<applet code="MyApplet.class" width=300 height=300>
<param name="nachricht" value ="hallo">
<param name="wert" value=“18">
</applet>

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

class MyApplet extends Applet {    // das Applet ist ein Panel !
    private String name;
    private int wert;

    public void init() {
        name = getParameter("nachricht");
        wert = Integer.parseInt(getParameter("wert"));

        setLayoutManager(new GridLayout(2,2));
        Label lbl = new Label(name);
        add(lbl);
        ...
    }
```

Anwendungen von Applets:

1. Ausführen von Programmen auf der Client-Machine

   Entlastung des Servers, Kopie der Class-Files, Sicherheit

2. Komplexe Interaktion mit Server-Komponenten

   Alternative: Javascript u.ä.: Applets sind mächtiger

3. Applets werden vom Brower ausgeführt

   eigene Applets müssen (möglichst) mit allen Browsern klar kommen.