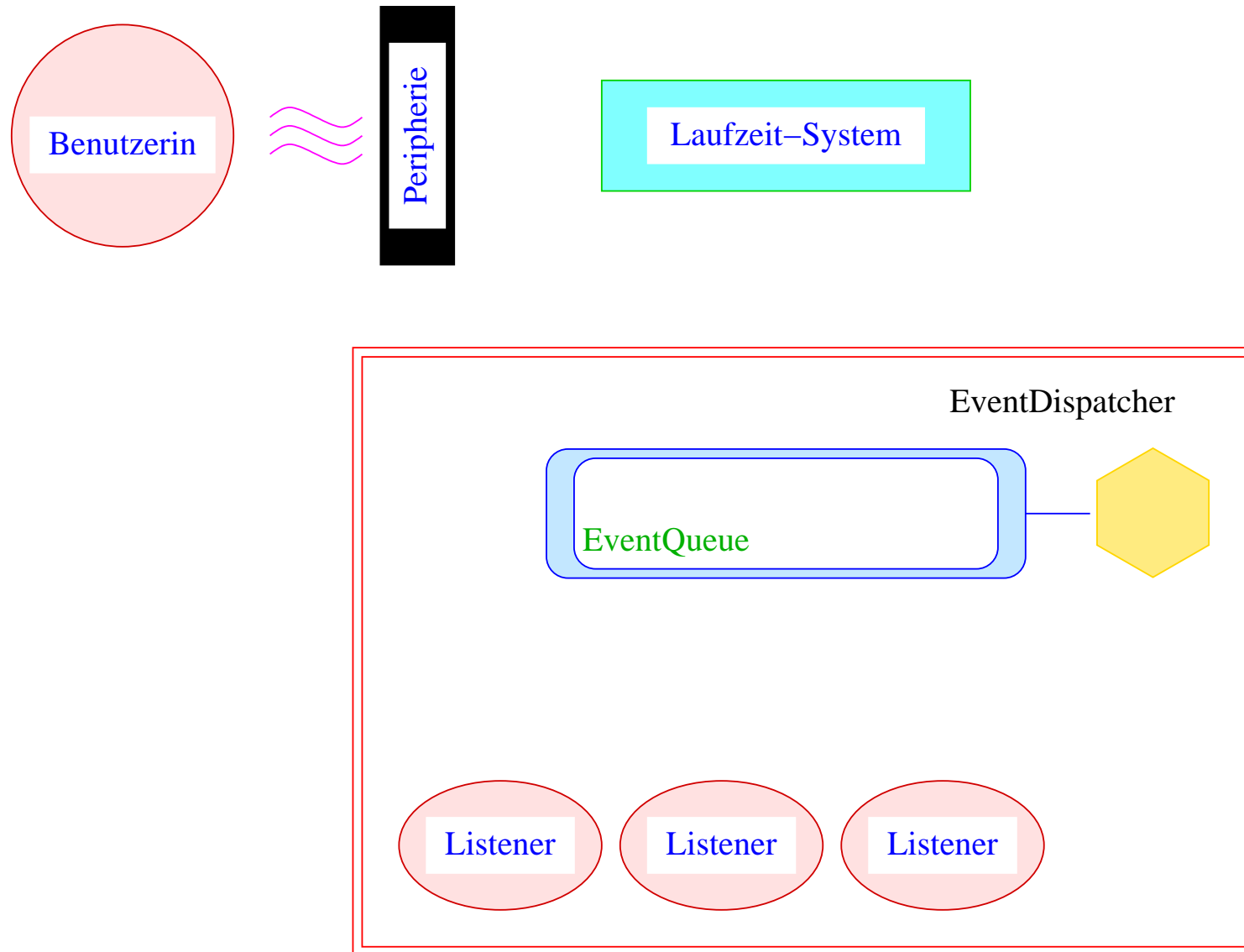


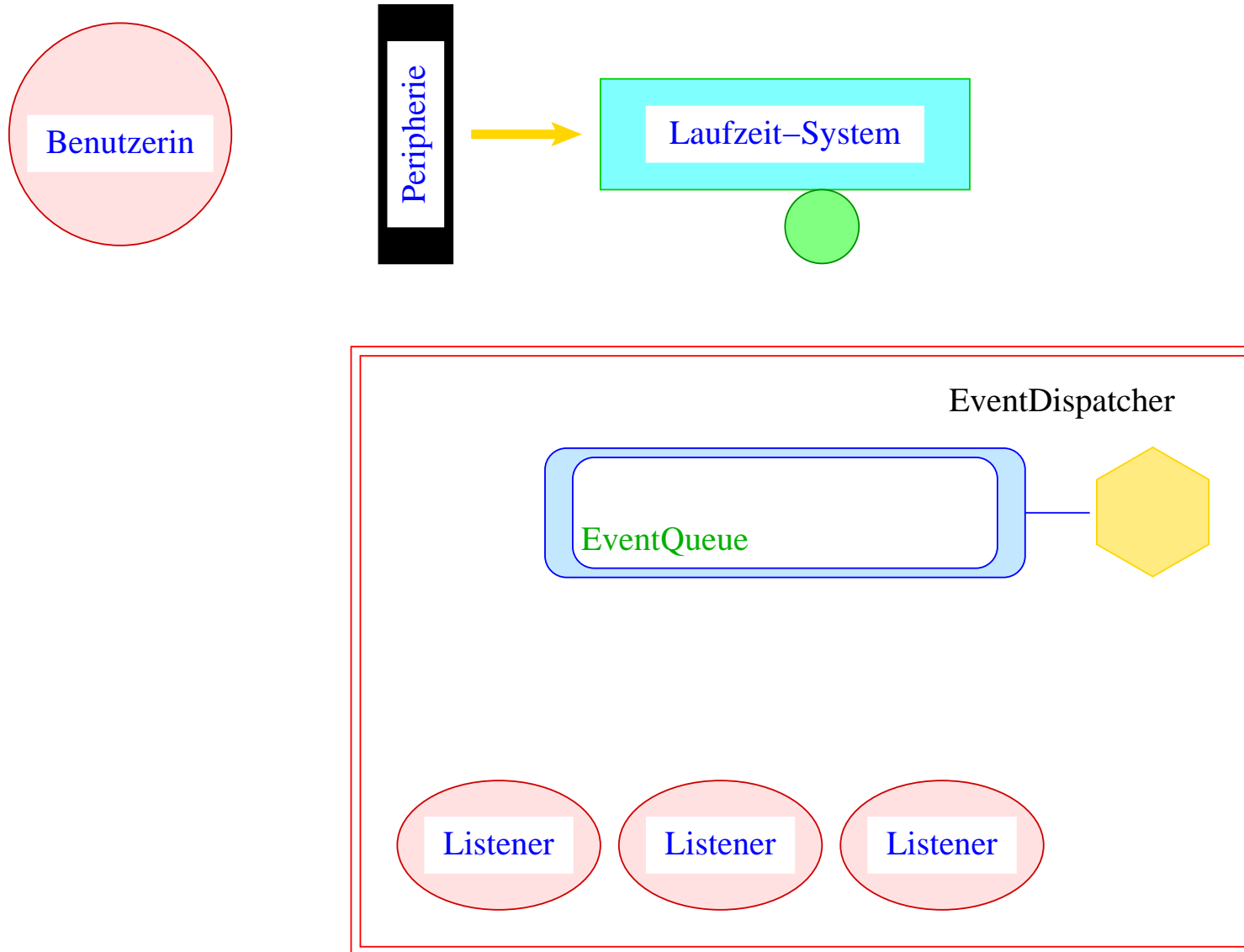
## 21 Graphische Benutzer-Oberflächen

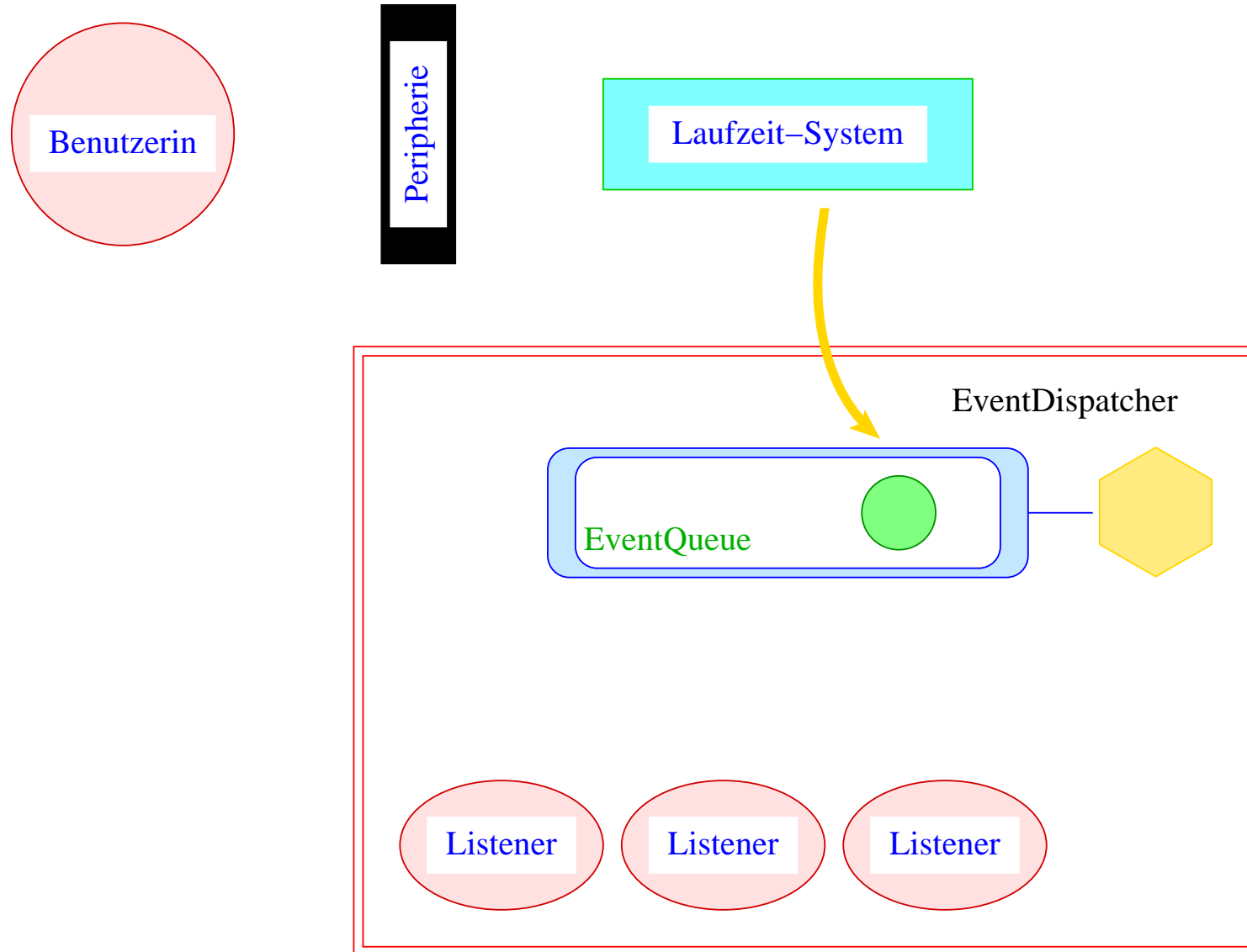
Eine graphische Benutzer-Oberfläche (GUI) ist i.a. aus mehreren Komponenten zusammen gesetzt, die einen (hoffentlich :-} intuitiven Dialog mit der Benutzerin ermöglichen sollen.

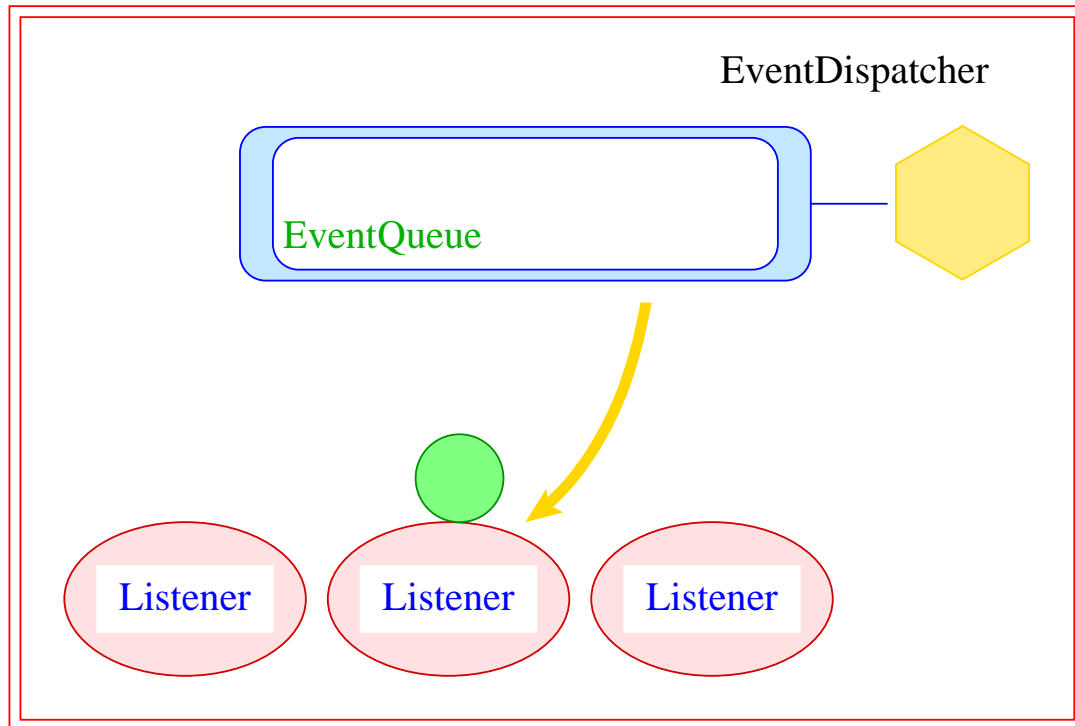
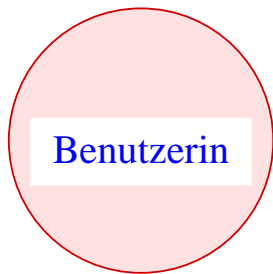
### Idee:

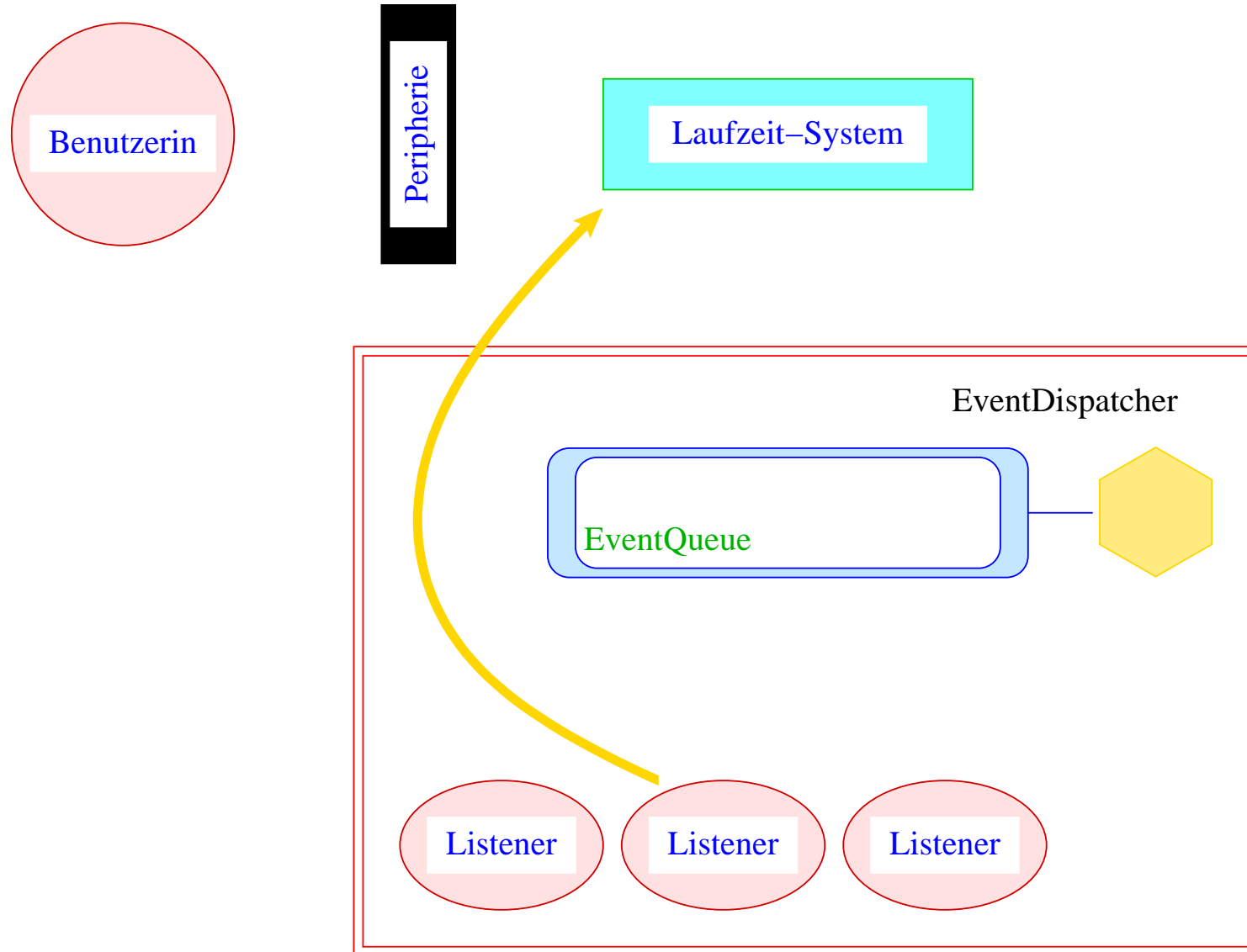
- Einzelne Komponenten bieten der Benutzerin Aktionen an.
- Ausführen der Aktionen erzeugt Ereignisse.
- Ereignisse werden an die dafür zuständigen Listener-Objekte weiter gereicht    ↑ Ereignis-basiertes Programmieren.

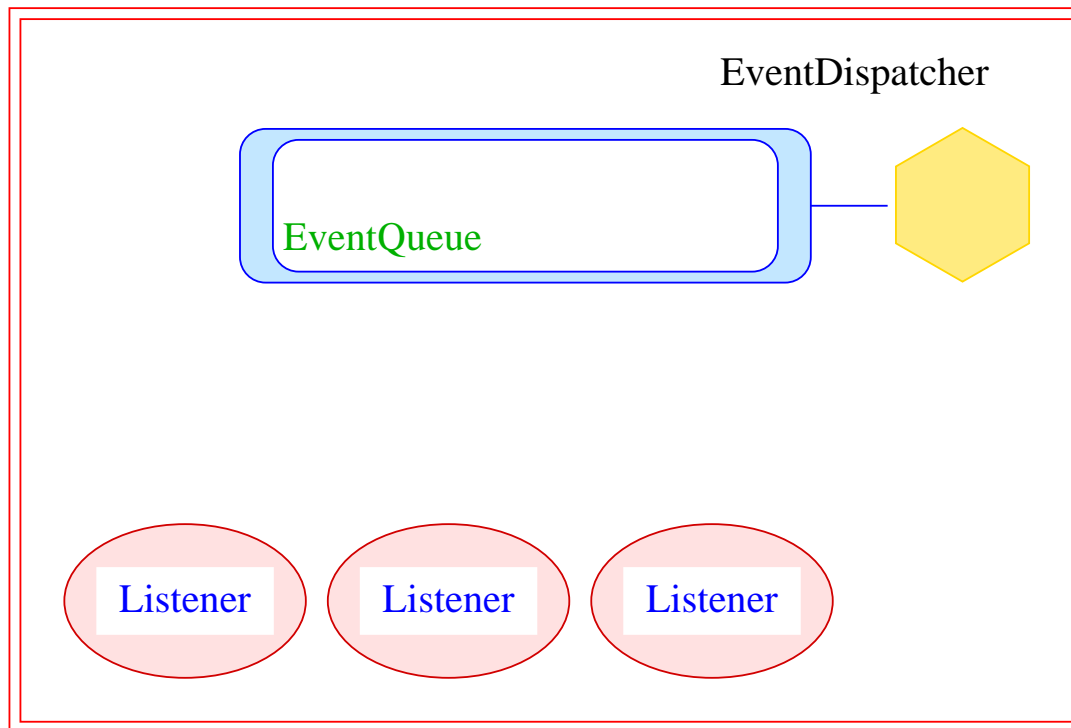
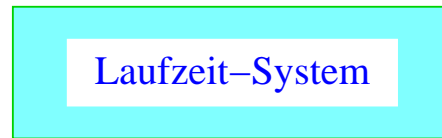
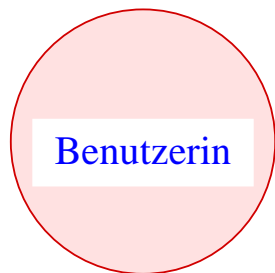


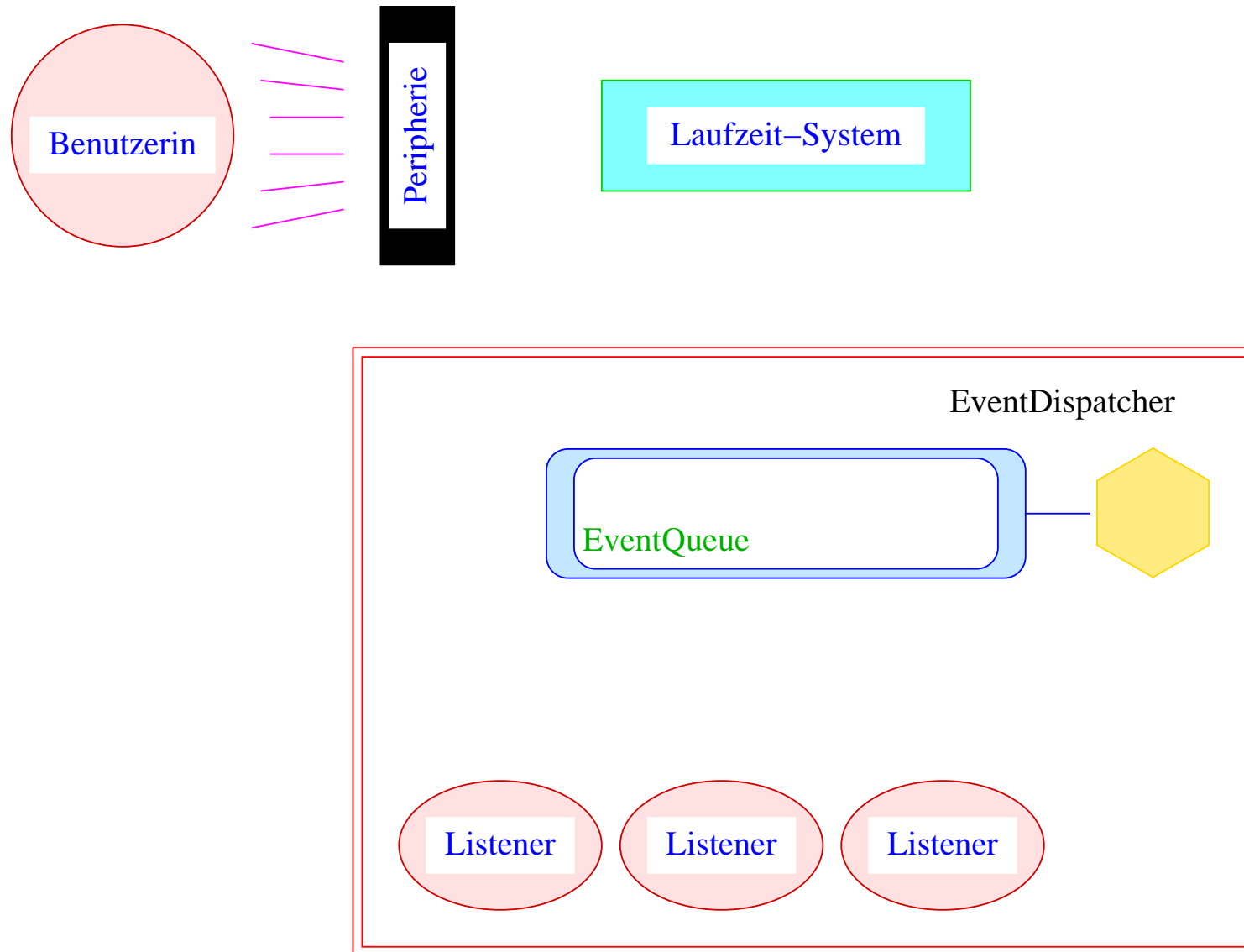














- Maus-Bewegungen und -Klicks, Tastatur-Eingaben etc. werden von der Peripherie registriert und an das  $\uparrow$  **Betriebssystem** weitergeleitet.
- Das **Java**-Laufzeit-System nimmt die Signale vom Betriebssystem entgegen und erzeugt dafür AWTEvent-Objekte.
- Diese Objekte werden in eine AWTEventQueue eingetragen  $\implies$  **Producer!**
- Die Ereignis-Schlange verwaltet die Ereignisse in der Reihenfolge, in der sie entstanden sind, kann aber auch mehrere ähnliche Ereignisse zusammenfassen ...
- Der AWTEvent-Dispatcher ist ein weiterer Thread, der die Ereignis-Schlange abarbeitet  $\implies$  **Consumer!**

- Abarbeiten eines Ereignisses bedeutet:
  1. Weiterleiten des `AWTEvent`-Objekts an das Listener-Objekt, das vorher zur Bearbeitung solcher Ereignisse **angemeldet** wurde;
  2. Aufrufen einer speziellen Methode des Listener-Objekts.
- Die Objekt-Methode des Listener-Objekts hat für die Reaktion des Applets zu sorgen.



- Ereignisse können jederzeit eintreten.
- Ihre Abarbeitung erfolgt sequentiell.

## 21.1 Schon mal ein Anfang ...

Beispiel: Ein Knopf :-)

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

public class FirstButton extends Applet
    implements ActionListener {
    Label label;
    Button button;
    ...
}
```

```
public void init() {  
    setBackground(Color.orange);  
    setFont(new Font("SansSerif", Font.BOLD, 18));  
    label = new Label("This is my first button :-)");  
    add(label);  
    button = new Button("Knopf");  
    button.setBackground(Color.white);  
    button.addActionListener(this);  
    add(button);  
}
```

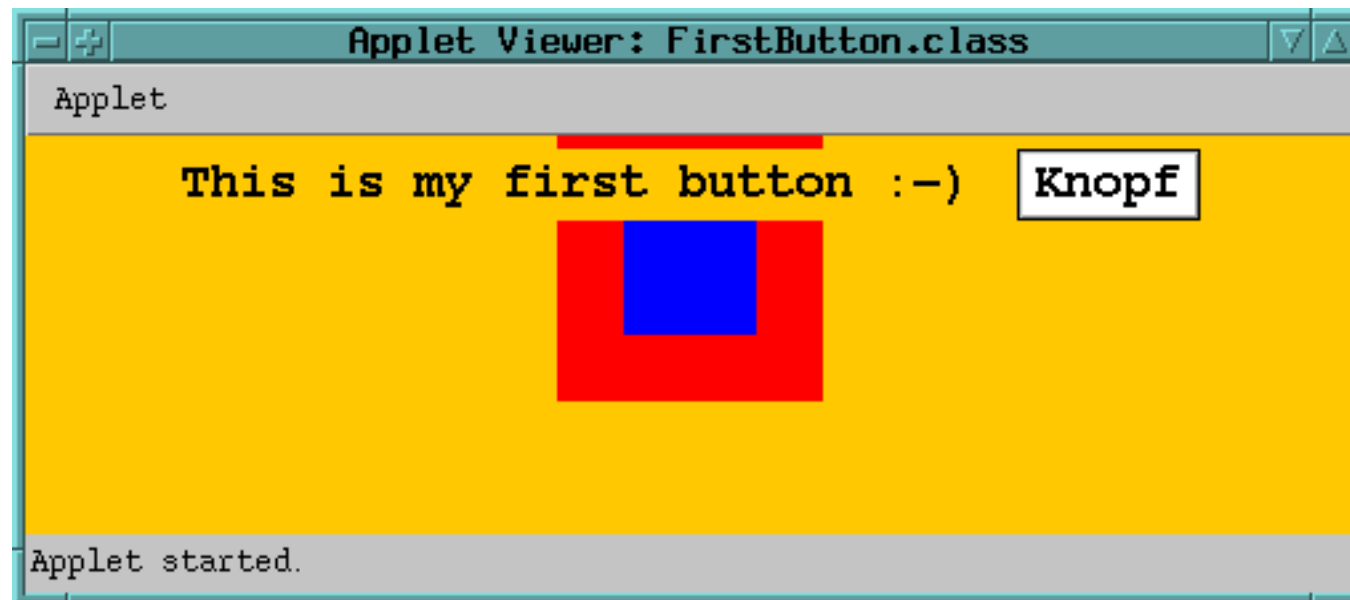
...

- Das Applet enthält zwei weitere Komponenten:  
(1) ein Label; (2) einen Button.
- Objekte dieser Klassen besitzen eine Aufschrift :-)

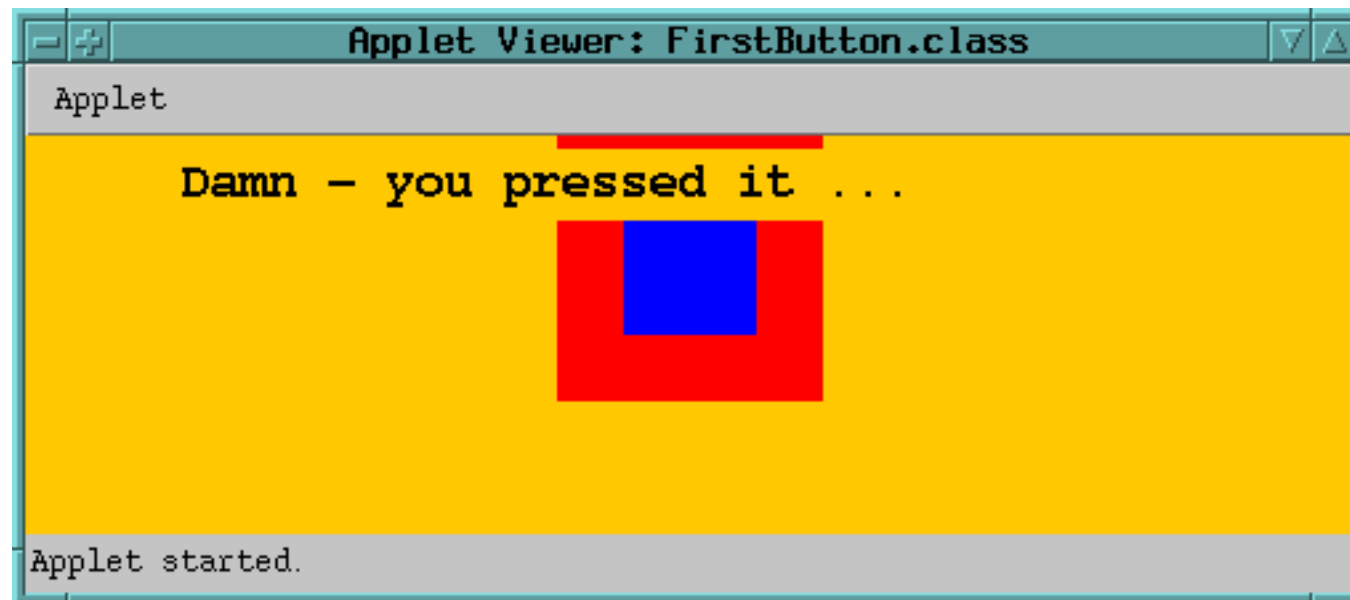
- Wie bei Applet-Objekten kann der Hintergrund gesetzt werden.
- `public void addActionListener(ActionListener listener)` registriert ein Objekt `listener` als dasjenige, das die von der Komponente ausgelösten `ActionEvent`-Objekte behandelt, hier: das Applet selber :-).
- `ActionListener` ist ein **Interface**. Zu seiner Implementierung muss die Methode `void actionPerformed(ActionEvent e)` bereitgestellt werden.
- Die Objekt-Methoden:
  - `void add(Component c)`
  - `void add(Component c, int i)`... fügen die Komponente `c` zum Applet hinten (bzw. an der Stelle `i`) hinzu.

```
...
public void paint(Graphics page) {
    page.setColor(Color.red);
    page.fillRect(200,0,100,100);
    page.setColor(Color.blue);
    page.fillRect(225,25,50,50);
}
public void actionPerformed(ActionEvent e) {
    label.setText("Damn - you pressed it ...");
    System.out.println(e);
    remove(button);
}
} // end of Applet FirstButton
```

- Die Methode `public void actionPerformed(ActionEvent e)` setzt den Text des Labels auf einen neuen Wert und beseitigt anschließend den Knopf mithilfe der Objekt-Methode `public void remove(Component c);`
- Beim Drücken des Knopfs passiert das Folgende:
  1. ein `ActionEvent`-Objekt `action` wird erzeugt und in die Ereignis-Schlange eingefügt.
  2. Der `AWTEvent-Dispatcher` holt `action` wieder aus der Schlange. Er identifiziert das Applet `app` selbst als das für `action` zuständige Listener-Objekt. Darum ruft er `app.actionPerformed( action );` auf.
- Wären `mehrere` Objekte als `listener` registriert worden, würden sukzessive auch für diese entsprechende Aufrufe abgearbeitet werden.







## Beachte:

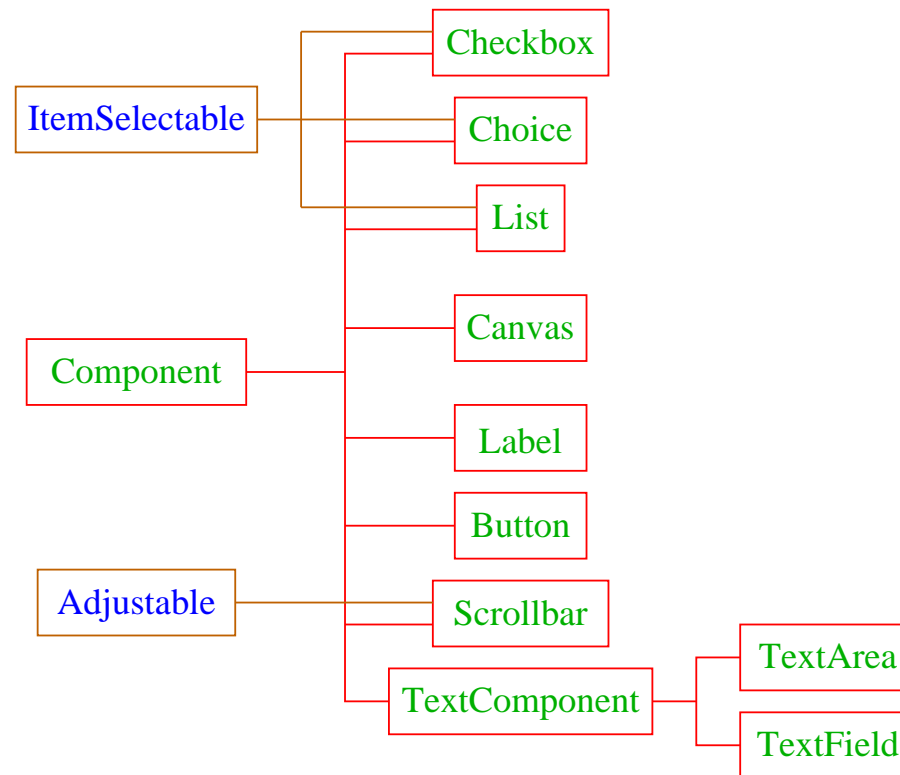
- Die in den Labels verwendete Schriftart richtet sich nach der des umgebenden Applets (zumindest in der Größe :-).
- Die Objekt-Methoden:

```
public String getText();  
public void setText(String text);
```

... gestatten den Zugriff auf den Text eines Label- (Button- oder TextComponent) Objekts.
- Die hinzugefügten Komponenten liegen im Applet-Feld **vor** der graphischen Darstellung, die die `paint()`-Methode erzeugt.
- Jede Komponente weiss, wie sie neu gemalt werden muss ...

## 21.2 Einfache AWT-Komponenten

Eine Übersicht ...



**Canvas** eine Fläche zum Malen ...

**Label** zeigt eine Text-Zeile.

**Button** einzelner Knopf, um eine Aktion auszulösen.

**Scrollbar** Schieber zur Eingabe von (kleinen) `int`-Zahlen.

**Canvas** eine Fläche zum Malen ...

**Label** zeigt eine Text-Zeile.

**Button** einzelner Knopf, um eine Aktion auszulösen.

**Scrollbar** Schieber zur Eingabe von (kleinen) `int`-Zahlen.

## Beispiel:

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class ScalingFun extends Applet
    implements AdjustmentListener {
    private Scrollbar sHeight, sWidth;
    private Image image;
    ...
}
```

```
private int width = 600; private int height = 500;
public void init() {
    setBackground(Color.white);
    image = getImage(getDocumentBase(),"df991230.jpg");
    setLayout(new BorderLayout());
    sHeight = new Scrollbar(Scrollbar.VERTICAL,
                           500,50,0,550);
    sHeight.addAdjustmentListener(this);
    add(sHeight,"West");
    sWidth = new Scrollbar(Scrollbar.HORIZONTAL,
                           600,50,0,650);
    sWidth.addAdjustmentListener(this);
    add(sWidth,"South");
}
...
```

- Ein Scrollbar-Objekt erzeugt AdjustmentEvent-Ereignisse.
- Entsprechende Listener-Objekte müssen das Interface AdjustmentListener implementieren.
- Dieses Interface verlangt die Implementierung einer Methode `public void adjustmentValueChanged(AdjustmentEvent e);`
- Die `init()`-Methode des Applets legt zwei Scrollbar-Objekte an, eines horizontal, eines vertikal.

Dafür gibt es in der Klasse Scrollbar die int-Konstanten HORIZONTAL und VERTICAL.

- Der Konstruktor `public Scrollbar(int dir,int init, int slide, int min, int max);` erzeugt ein Scrollbar der Ausrichtung `dir` mit Anfangsstellung `init`, Dicke des Schiebers `slide`, minimalem Wert `min` und maximalem Wert `max`.

Aufgrund der Dicke des Schiebers ist der [wirkliche](#) Maximal-Wert `max - slide`.

- `void addAdjustmentListener(AdjustmentListener adj);`  
registriert das `AdjustmentListener`-Objekt als Listener für die `AdjustmentEvent`-Objekte des Scrollbars.
- Das selbe Objekt kann mehrmals als Listener auftreten ;-)

Bleibt, das Geheimnis um `Layout` und `West` bzw. `South` zu lüften ...

- Jeder Container, in den man weitere Komponenten schachteln möchte, muss über eine Vorschrift verfügen, wie die Komponenten anzuordnen sind.

- Diese Vorschrift heißt `Layout`.

Zur Festlegung des Layouts stellt `Java` das Interface `LayoutManager` zur Verfügung sowie nützliche implementierende Klassen ...



- Eine davon ist das BorderLayout.

- Mithilfe der String-Argumente:

`BorderLayout.NORTH = "North",`

`BorderLayout.SOUTH = "South",`

`BorderLayout.WEST = "West",`

`BorderLayout.EAST = "East" und`

`BorderLayout.CENTER = "Center"`

kann man **genau eine** Komponente am bezeichneten Rand bzw. der Mitte positionieren.

```
public void paint(Graphics page) {
    page.drawImage(image,0,0,width,height,this);
}

public void adjustmentValueChanged(AdjustmentEvent e) {
    Adjustable s = e.getAdjustable();
    int value = e.getValue();
    if (s == sHeight) height = value;
    else width = value;
    repaint();
}
} // end of Applet ScalingFun
```

- Um AdjustmentEvent-Objekte behandeln zu können, implementieren wir die Methode  
`AdjustmentValueChanged(AdjustmentEvent e);`
- Jedes AdjustmentEvent-Objekt verfügt über die Objekt-Methoden:  

```
public AdjustmentListener getAdjustable();  
public int getValue();
```


... mit denen das auslösende Objekt sowie der eingestellte int-Wert abgefragt werden kann.
- Dann sieht das Applet so aus ...

Applet Viewer: ScalingFun.class

Applet

# DOCTOR FUN

30 Dec 99



Copyright © 1999 David Farley, d-farley@metalab.unc.edu  
<http://metalab.unc.edu/Dave/drfun.html>  
This cartoon is made available on the Internet for personal viewing only. Opinions expressed herein are solely those of the author.

Another unexpected Y2K rollover problem

Start: applet not initialized.

Applet Viewer: ScalingFun.class

Applet

# DOCTOR FUN

30 Dec 99

Copyright © 1999 David Farley, d-farley@metabab.unc.edu  
<http://metabab.unc.edu/Dave/drfun.html>  
This cartoon is made available on the Internet for personal viewing only. Opinions expressed herein are solely those of the author.

Another unexpected Y2K rollover problem

Start: applet not initialized.

## Texteingabe:

**TextField** zeigt eine Text-Zeile, die vom Benutzer modifiziert werden kann.

**TextArea** zeigt mehrere modifizierbare Text-Zeilen.

## Auswahl aus mehreren Alternativen:

**List** scrollbare Liste wählbarer Items;

**Choice** analog List – nur mit Anzeige des ausgewählten Items.

**Checkbox** kann nur die Werte `true` oder `false` annehmen. Mehrere davon können jedoch in einer `CheckBoxGroup` zusammengefasst werden.

## Beispiel: Eine Choice-Liste ...

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class FirstChoice extends Applet
                               implements ItemListener {
    private Color color = Color.white;
    private Choice colorChooser;
    ...
}
```

```
public void init() {
    setFont(new Font("Serif", Font.PLAIN, 18));
    colorChooser = new Choice();
    colorChooser.setBackground(Color.white);
    colorChooser.add("white");
    colorChooser.add("red");
    colorChooser.add("green");
    colorChooser.add("blue");
    colorChooser.addItemListener(this);
    add(colorChooser);
    setBackground(Color.orange);
}
...
```



- `public Choice();` legt ein neues `Choice`-Objekt an;
- Zu diesem Objekt können beliebig viele Items hinzugefügt werden. Dazu dient die ObjektMethode:  
`public void add(String str);`
- `public void addItemListener(ItemListener listener);` registriert das Objekt `listener` für die erzeugten `ItemEvent`-Objekte.
- `ItemListener` ist ein Interface ähnlich wie `ActionListener`.
- Wieder fügen wir die neue Komponente mithilfe von `void add(Component comp)` dem Applet hinzu.