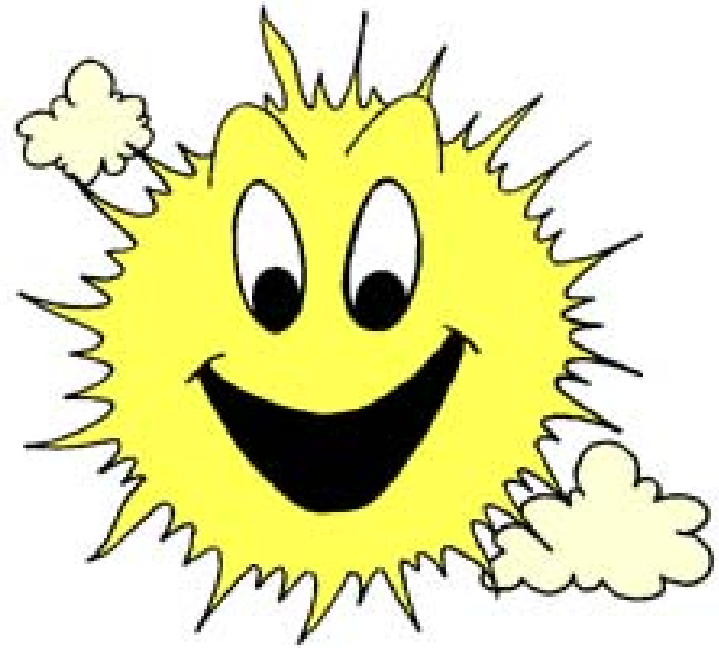
Wiederholung 3. Vorlesung



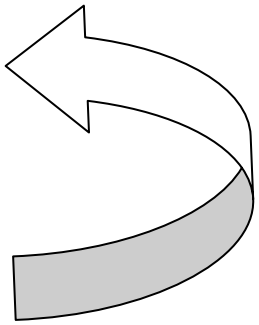
2. Entwicklung von Anwendungen (2)

Auch für das **Software Design** ist UML ein guter Ansatz:

- Use Case
- Klassen-Diagramm
- Sequenz-Diagramm
- Kollaborationsdiagramm
- Status-Diagramm

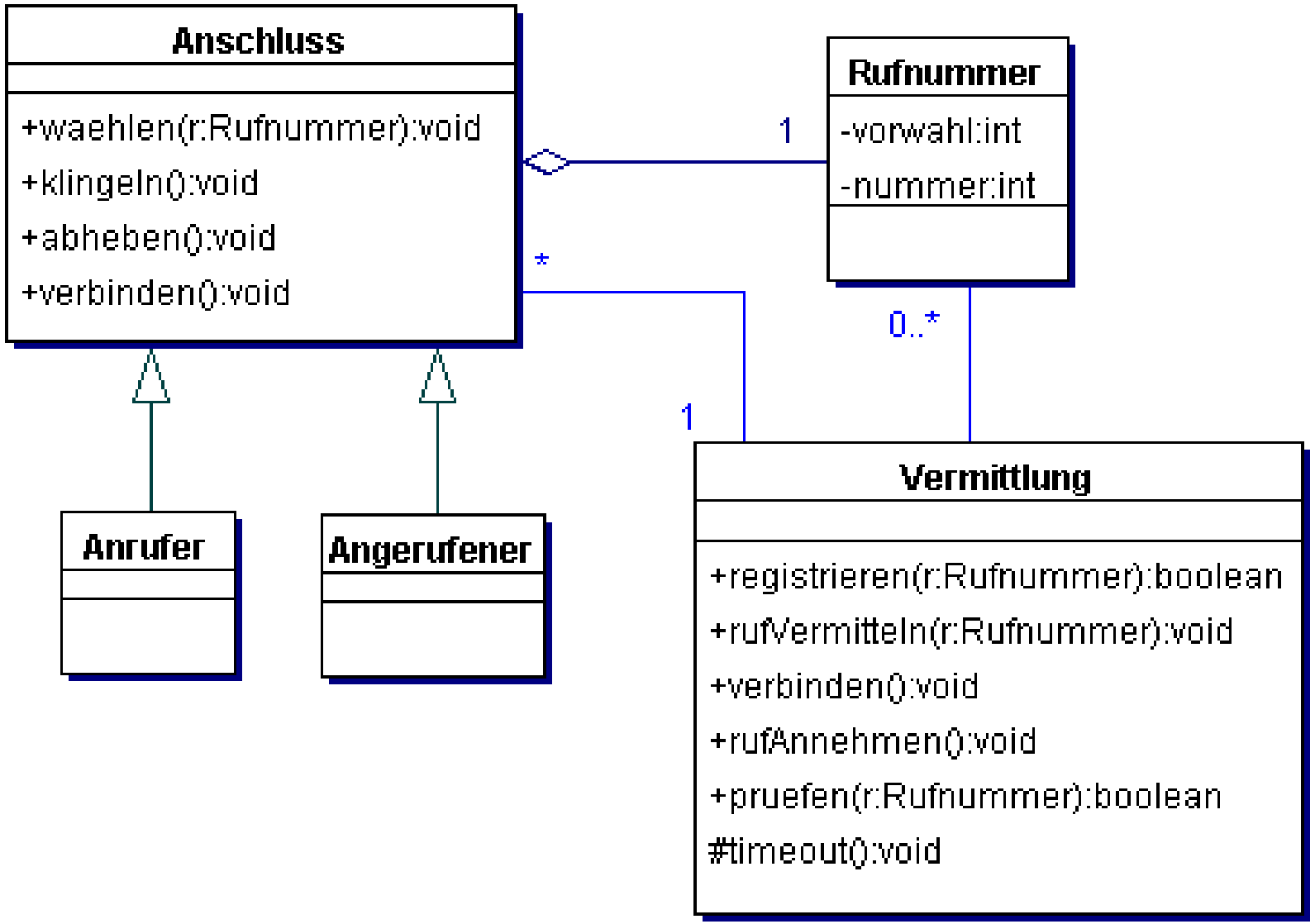
Entwicklung schrittweise vornehmen!

- schreiben
- testen
- erweitern
- testen



zyklisch
und
iterativ

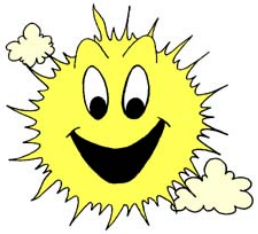
2.1 Beispiel Klassendiagramm



2.2 eine gute Anwendung hat/ist/bietet ... (Gruppenarbeit)

- intuitive GUI und Bedienung
- eindeutige/verständliche Dialoge/Befehle
- kontextsensitive Hilfe
- Ergonomie
  - keine unnötige Belastung
  - logischer Aufbau – Workflow
  - Schrift und Farbgebung
  - Shortcuts zu Befehlen
  - Macro/Batch-Fähigkeiten
  - Barrierefreiheit
  - zielgruppenorientiert
- Feedback über
  - Zustände / Status
  - Aktionen
- Fehlertoleranz
- einheitliches Look & Feel
- Internationalisierung / Lokalisierung
- Fehlerfreiheit
- Effizienz / Performanz
- Robustheit / Stabilität
- Sicherheit
- wartbar / erweiterbar
- modular
- portabel

Ende der Wiederholung



2.2 Benutzerzentriertes Engineering: Ziele

- Benutzerzentrierung
  - Benutzerführung
    - Fehlervermeidung (statt Fehlerbehebung)
    - Logische Abfolge – Workflow
    - Anordnung der Komponenten – relative Positionierung
  - Unterstützung des Benutzers
    - Erstellen – Modifizieren
    - Inspizieren – Prüfen
    - angemessene Präsentation und Interaktion
  - Navigation
    - auch in komplexen Situationen

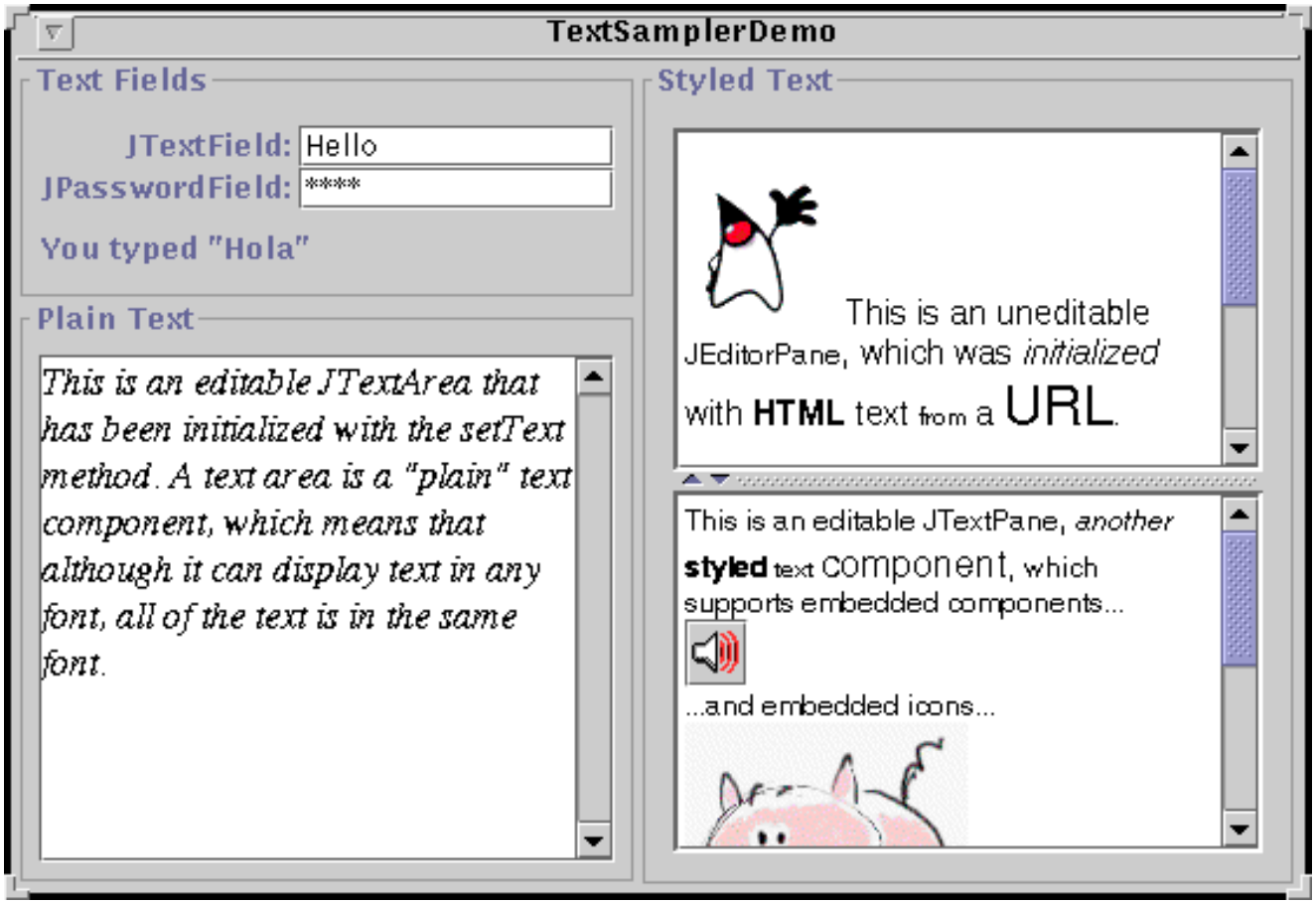
2.2 Benutzerzentriertes Engineering: Ziele

- Benutzerzentrierung (Fortsetzung)
  - Komplexität beherrschen
    - Information aufbereiten
    - selbsterklärend
    - kontextbezogene Hilfe
  - Realitätsnähe ←→ Abstraktion
- Modellierung
  - Objekthierarchien
  - Zustände
- Einheitliches Look & Feel

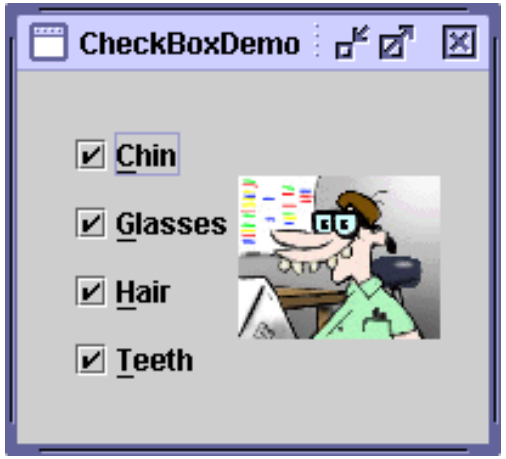
4. Grafische Grundkomponenten

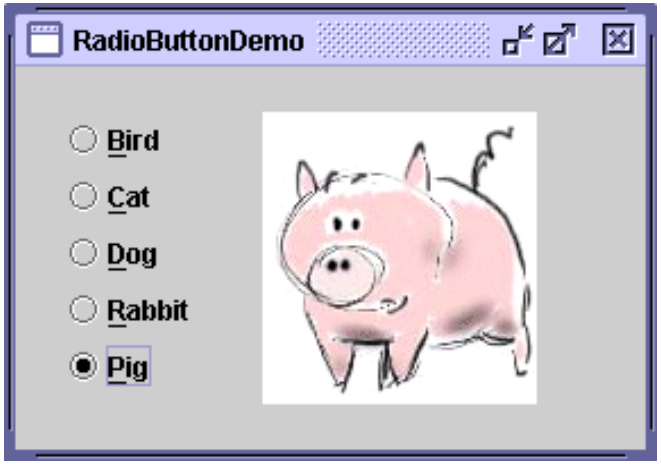
- Label
- Text-Field (Textfeld)
- Text-Area (Textbereich)
- Button (Schaltfläche
  - Simple Button
  - Toggle Button
  - Checkbox
  - Radio Button
- Auswahlboxen
  - Combo Box
  - List Box
  - Spin Button
  - Tabellen
- Slider (Schieberegler
- Progress Bar
- Menu
- Toolbar
- Tooltip
- Dialogboxen
  - Information
  - Eingabe
- ...

4.1 Grundkomponenten: Label und Text

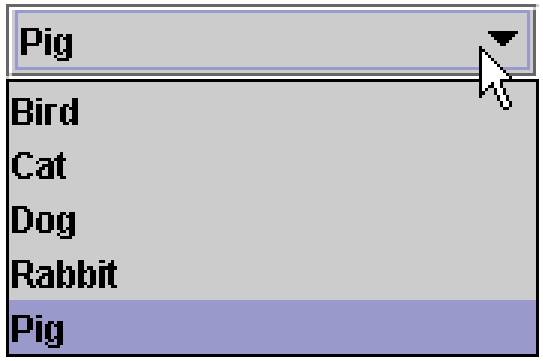


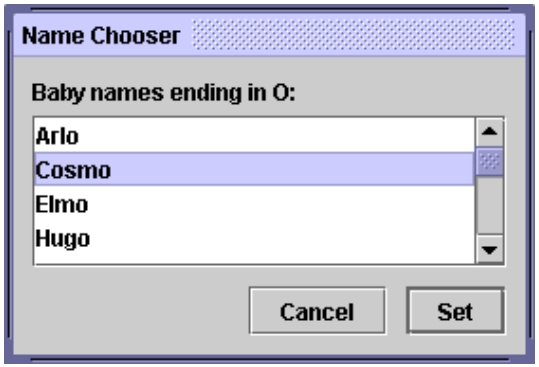
4.1 Grundkomponenten: Button

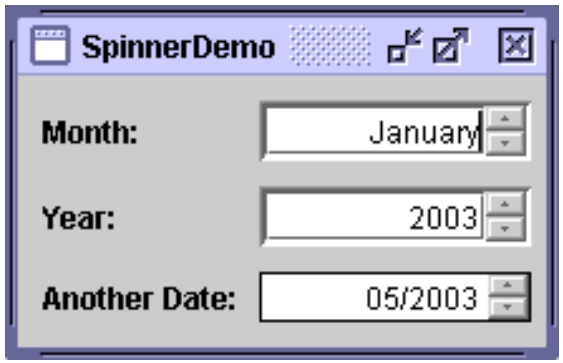


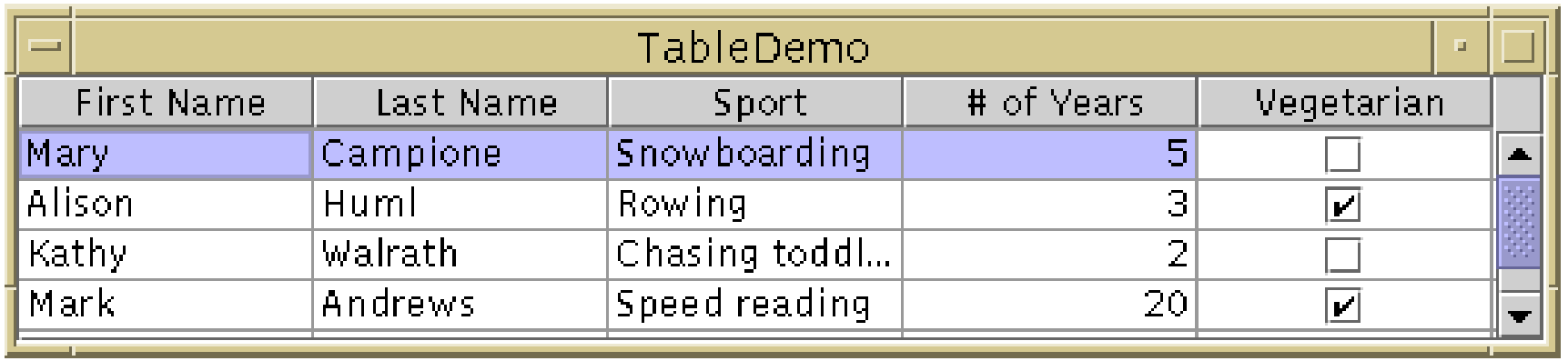


4.1 Grundkomponenten: Auswahlboxen



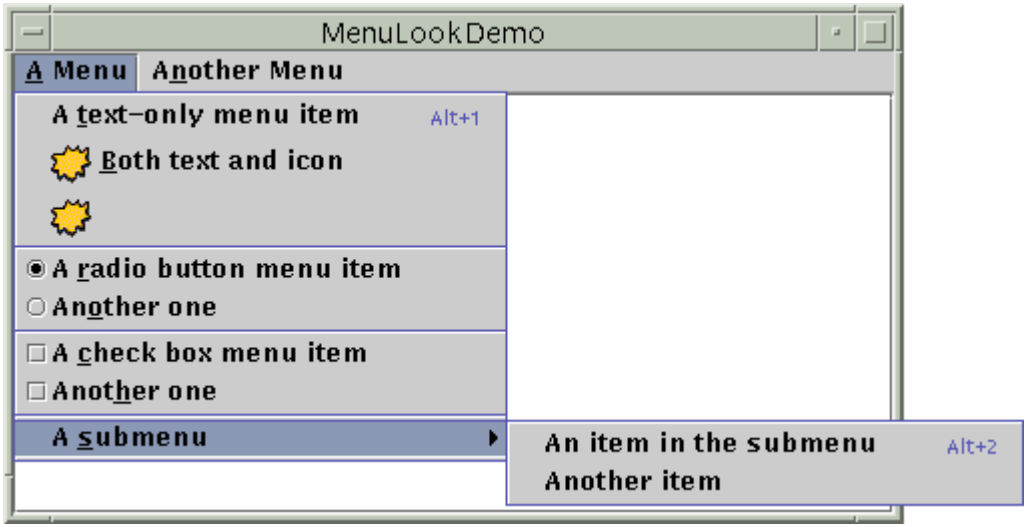


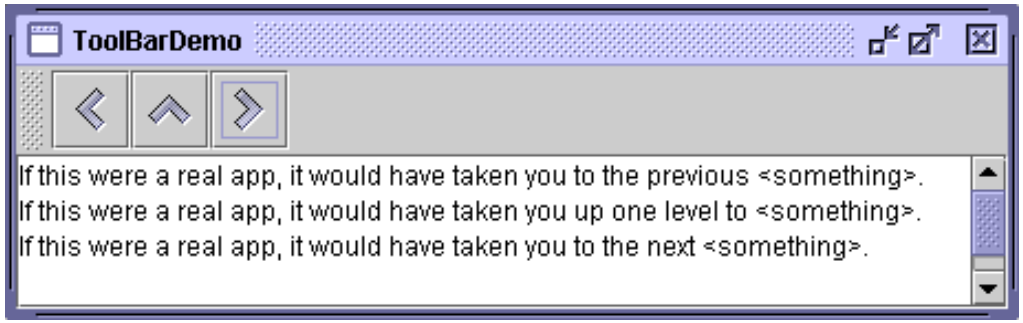




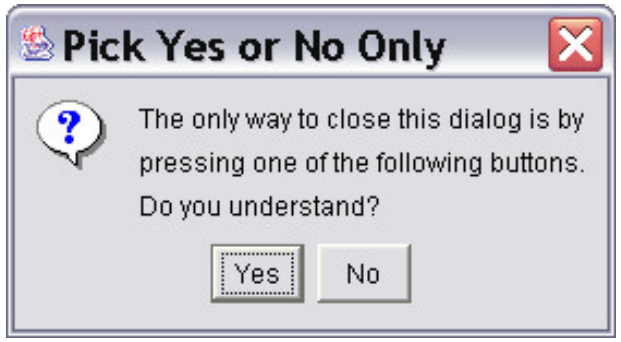
4.1 Grundkomponenten: Slider / Progress

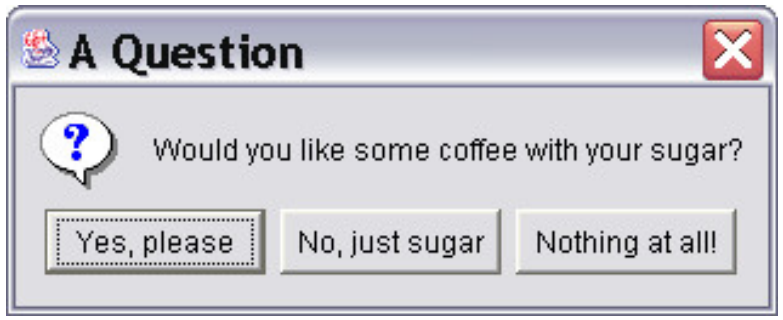
4.1 Grundkomponenten: Menu / Toolbar

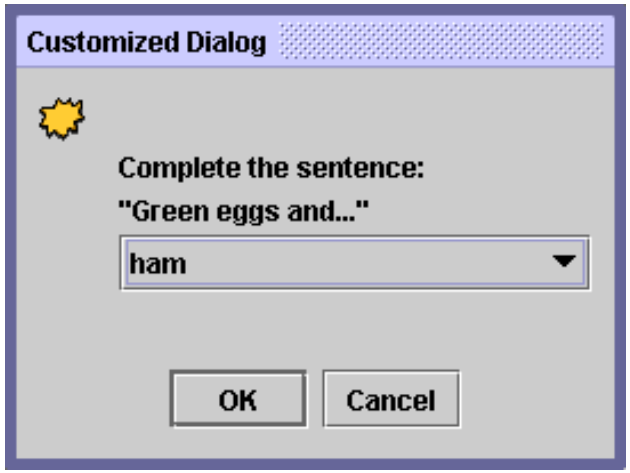


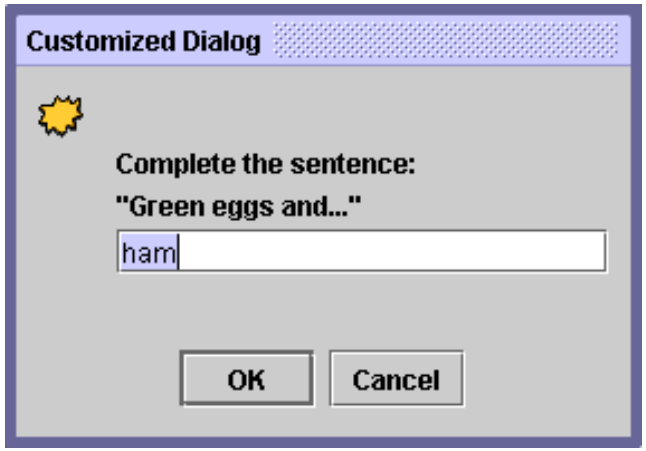


4.1 Grundkomponenten: Dialogboxen



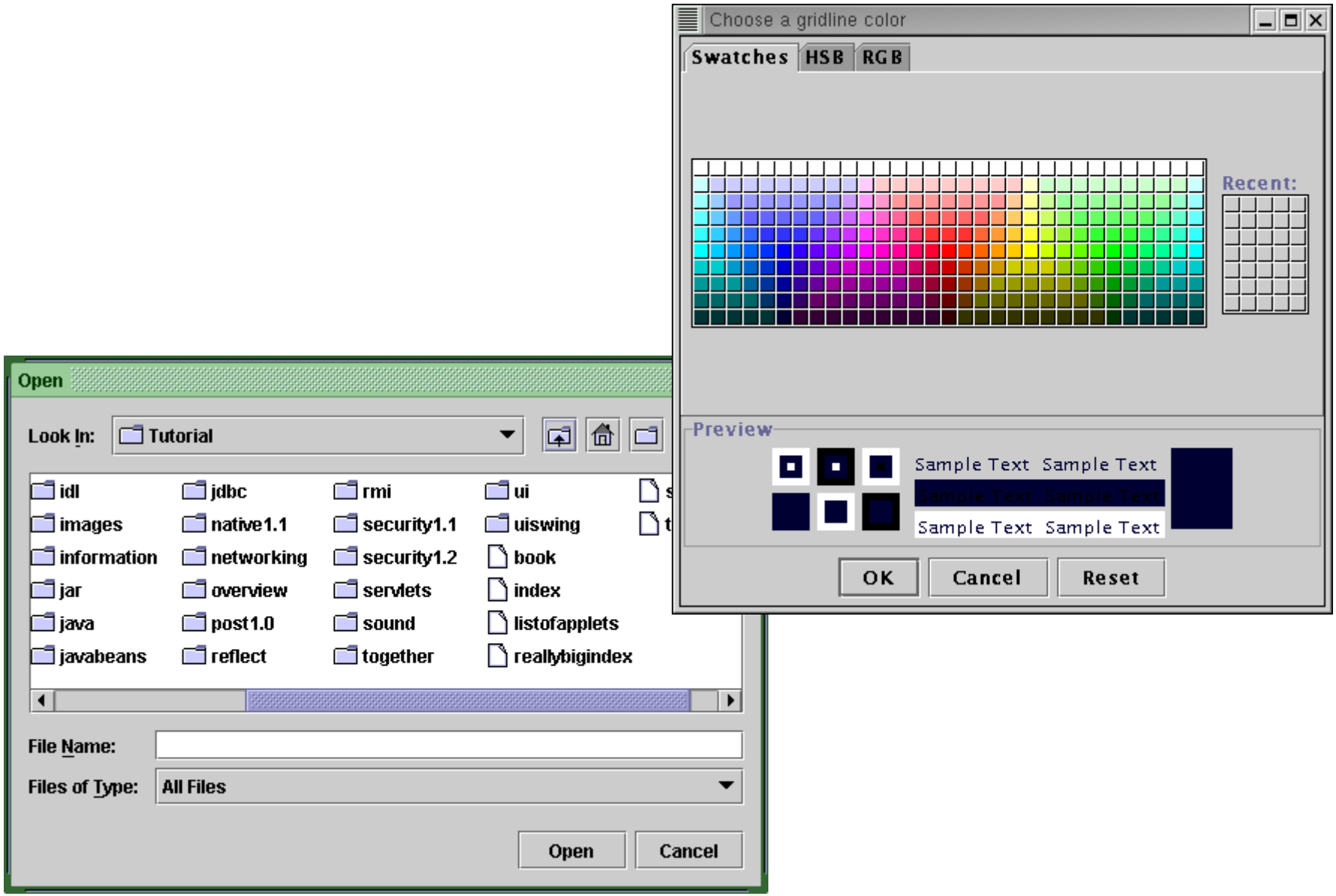




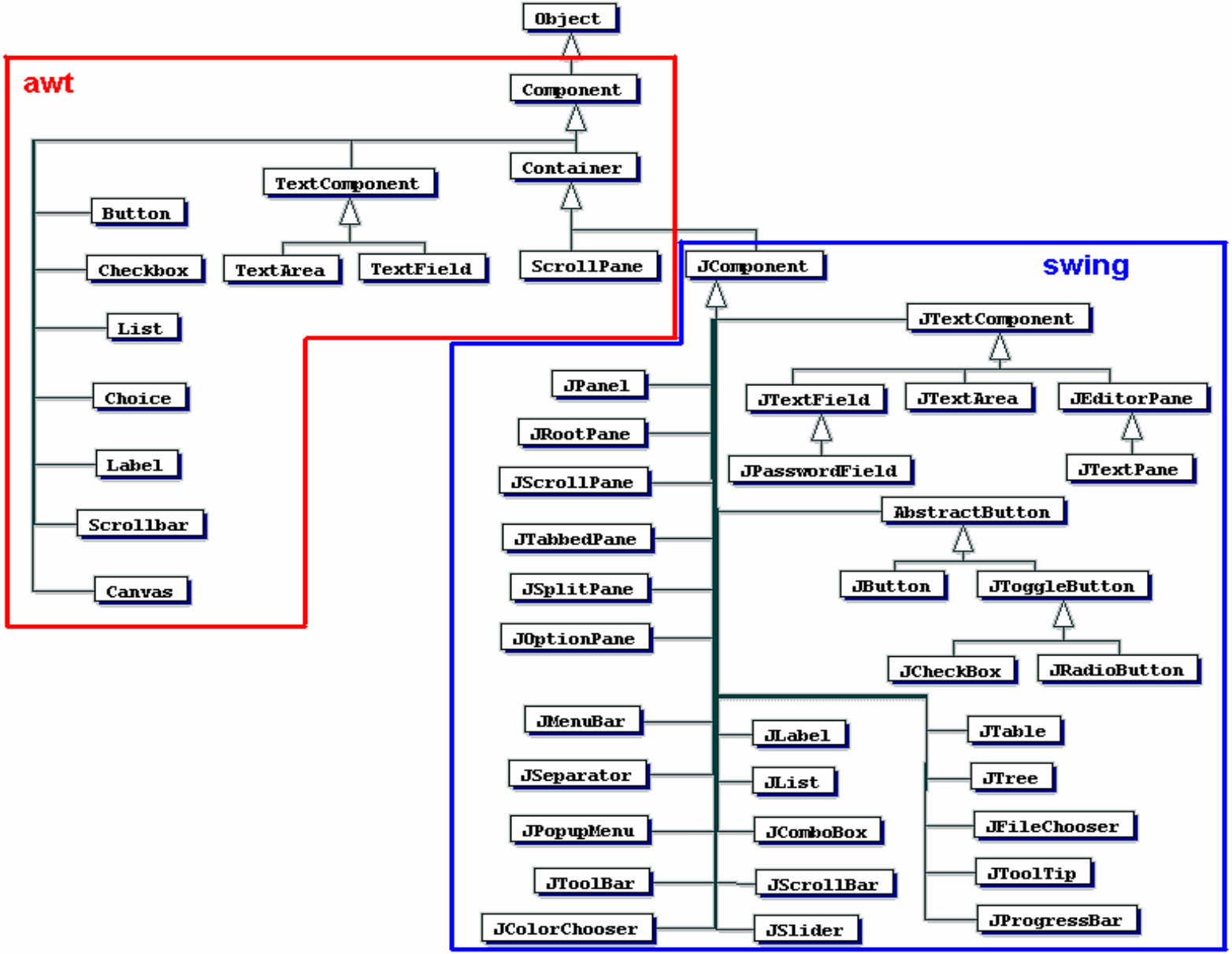


4.1 Grundkomponenten: Icons

4.1 Grundkomp.: Dialog-Auswahlboxen



4.1 Objekthierarchie: AWT und Swing

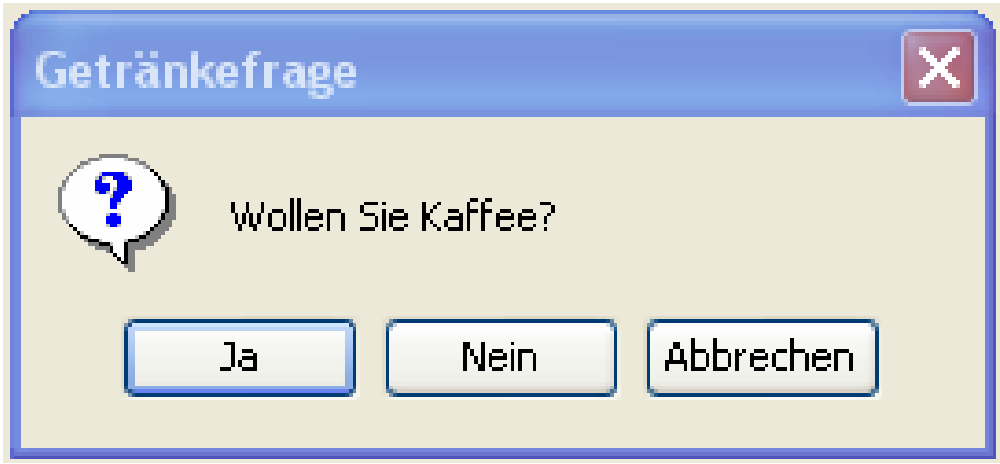


4.1 Dialogboxen: JOptionPane: Message

```
JOptionPane.showMessageDialog(
    this,
    "Dies ist eine\nWICHTIGE Meldung",
    "Information",
    JOptionPane.INFORMATION_MESSAGE );
```

4.1 Dialogboxen: JOptionPane: Confirmation

```
int option = JOptionPane.showConfirmDialog(
    this,
    "Wollen Sie Kaffee?",
    "Getränkefrage",
    JOptionPane.YES_NO_CANCEL_OPTION,
    JOptionPane.QUESTION_MESSAGE );
```



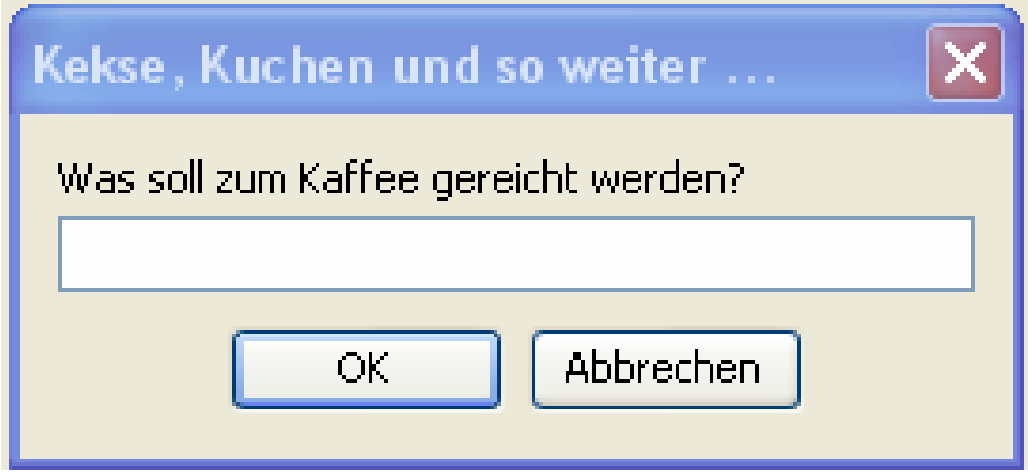
4.1 Dialogboxen: JOptionPane: Confirmation (2)

```
int option = JOptionPane.showConfirmDialog(
   this,
   "Wollen Sie Kaffee?",
   "Getränkefrage",
   JOptionPane.YES_NO_CANCEL_OPTION,
   JOptionPane.QUESTION_MESSAGE );

if ( option == JOptionPane.YES_OPTION )
   System.out.println( "Bringen Sie bitte Kaffee" );
else if ( option == JOptionPane.NO_OPTION )
   System.out.println( "Danke, nein" );
else if ( option == JOptionPane.CANCEL_OPTION )
   System.out.println( "Keine Antwort" );
else if ( option == JOptionPane.CLOSED_OPTION )
   System.out.println( "so was, abgebrochen..." );
```

4.1 Dialogboxen: JOptionPane: Input

```
String antwort =
  JOptionPane.showInputDialog(
    this,
    "Was soll zum Kaffee gereicht werden?",
    "Kekse, Kuchen und so weiter ...",
    JOptionPane.PLAIN_MESSAGE );
System.out.println( antwort +
    " gibt\'s zum Kaffee" );
```



4.2 Ereignisbehandlung in Java

Wir wissen bereits, wie wir Komponenten in eine grafische Oberfläche einbinden können.

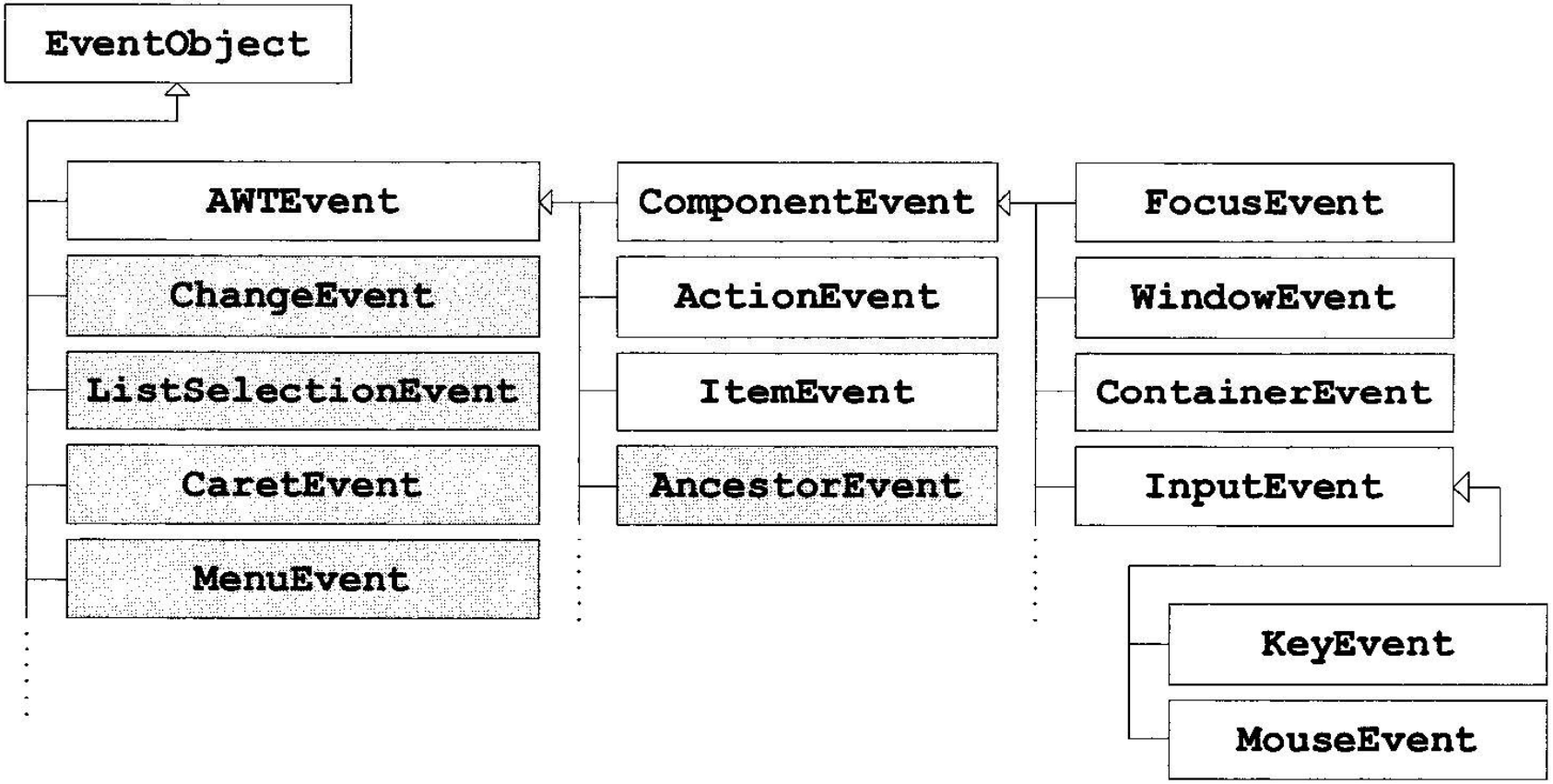
- es gibt passive Komponenten (JLabel, ...) und
- aktive Komponenten (Button, Menu, ...)
- wie kann ein Programm **auf eine bestimmte aktive Komponente** reagieren**?**

4.2 Ereignisbehandlung in Java

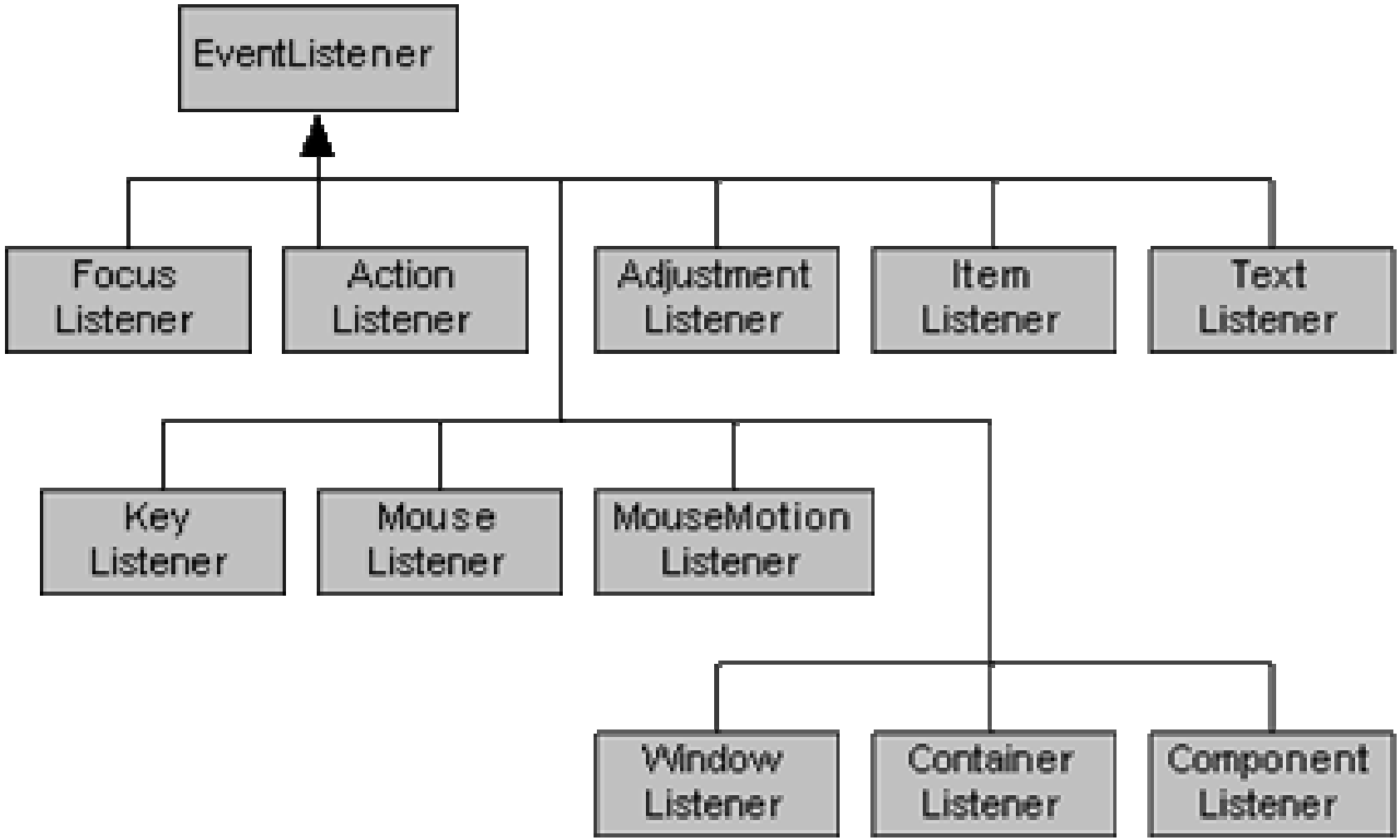
- eine Komponente kann ein **Ereignis** erzeugen
- ein Objekt muss **auf Ereignisse lauschen** (**Listener**)
- der aktive Komponenten (Button, Menu, ...) muss mitgeteilt werden, **wer auf Ereignisse** lauscht.
- Dem Listener muss mitgeteilt werden, **was zu tun ist, wenn das Ereignis eintritt (Aktion)**.

Listener (Wache)

4.2.1 Ereignisklassen in Java (Auszug)



4.2.2 Listener-Klassen in Java



4.2.2 Listener-Klassen in Java

- Listener-Klassen in Java sind in der Regel **Interfaces** oder **abstrakte Klassen**
- sie geben die **Grundstruktur** vor und vereinbaren **Schnittstellen** (Methoden)
- **was** aber bei einem Ereignis zu tun ist, muss der Programmierer festlegen.
- daher **muss** die **Aktion**, die auf ein Ereignis folgen soll **implementiert werden**
- dies erfolgt durch die
  - **Implementierung** des **Interfaces** oder
  - **Ableitung** und Implementierung der **abstrakten Klasse**

4.2.2 Listener Beispiel (Erweiterung Hello2)

Informatik

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

public class LDialog extends JFrame
{
    Container c;
    JPanel pCenter = new JPanel();
    JPanel pBottom = new JPanel();
    JLabel beschrift = null;
    JButton bRot  = new JButton( "Rot" );
    JButton bBlau = new JButton( "Blau" );
```

4.2.2 Listener Beispiel (2) Layout

```
public LDialog() {
      c = getContentPane();
      c.setLayout( new BorderLayout() );
      beschrift = new JLabel( "Text mit Farbe" );

      pCenter.add( beschrift );
      pCenter.setBackground( Color.GREEN );
      c.add( pCenter, BorderLayout.CENTER );

      pBottom.add( bRot );
      pBottom.add( bBlau );
      c.add( pBottom, BorderLayout.SOUTH );

      ButtonListener bl = new ButtonListener();
      bRot.addActionListener( bl );
      bBlau.addActionListener( bl );
}
```

4.2.2 Listener Beispiel (3) ActionListener

```
/* innere Klasse */
class ButtonListener implements ActionListener
{
   /* Action-Methode des Interface implementieren */
   public void actionPerformed( ActionEvent e )
   {
      if (e.getSource() == bRot )
         pCenter.setBackground( Color.RED );
      else if (e.getSource() == bBlau )
         pCenter.setBackground( Color.BLUE );
   }
}
```

4.2.2 Listener Beispiel (4) Listener hinzufügen

```
public LDialog() {
        c = getContentPane();
        c.setLayout( new BorderLayout() );
        beschrift = new JLabel( "Text mit Farbe" );

        pCenter.add( beschrift );
        pCenter.setBackground( Color.GREEN );
        c.add( pCenter, BorderLayout.CENTER );

        pBottom.add( bRot );
        pBottom.add( bBlau );
        c.add( pBottom, BorderLayout.SOUTH );

      ButtonListener bl = new ButtonListener();
      bRot.addActionListener( bl );
      bBlau.addActionListener( bl );
}
```