Java-Swing-Einführung

Vorbereitungswoche für das SWP2001

von Andreas Lattner

1. Einleitung: AWT & Swing
2. Komponenten
3. Layoutmanager
4. Model-View-Controller
5. Events & Listener
6. Packages, Klassen & Interfaces
7. Zusammenfassung

Folien z.T. von Oliver Radfelder übernommen

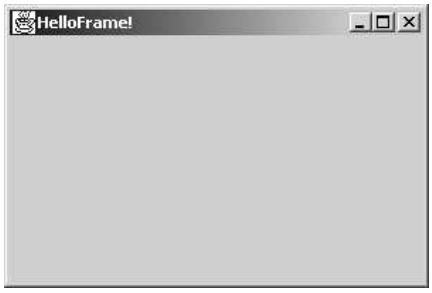
AWT und Swing

- AWT (Abstract Windowing Toolkit)
  - Rudimentäre GUI-Komponenten
  - Seit JDK1.0
  - Schnell und einfach zu bedienen
  - Kleinster gemeinsamer Nenner der Betriebssysteme
    - z.B. keine Tables, Trees
    - systemabhängiges Aussehen
  - Von allen gängigen Browsern unterstützt

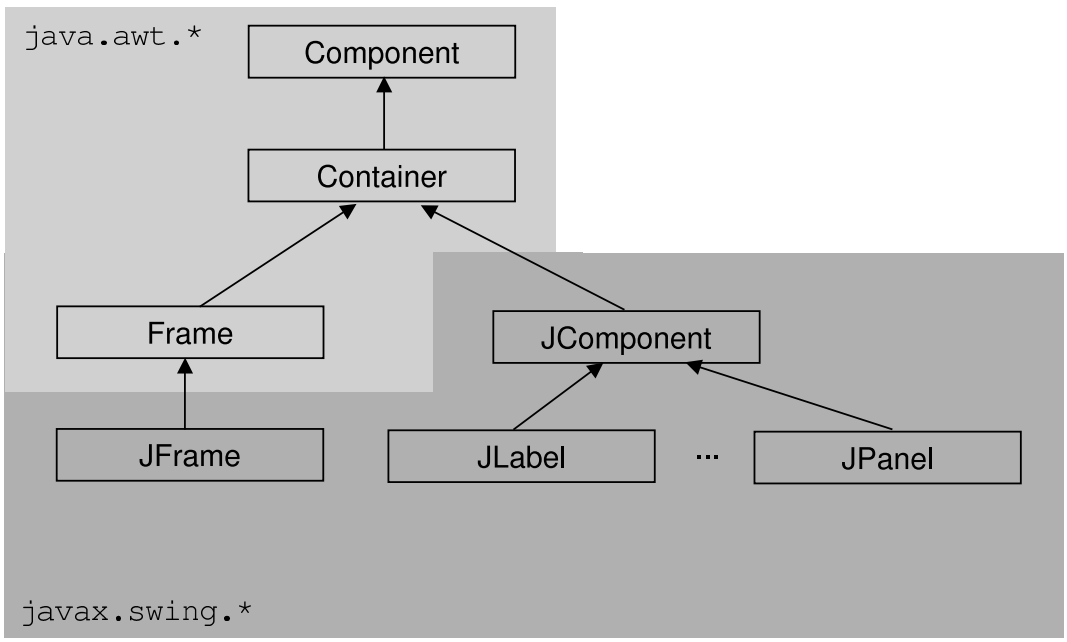
- Swing
  - Moderneres Toolkit, setzt auf AWT auf
  - Komplexer als AWT
  - Seit JDK1.2 enthalten
  - Reine Java-Implementierung (‚lightweight' GUI-Komponenten)
  - Möglichst nicht mit AWT-Komponenten mischen!

Verwendung von Swing: JFrame

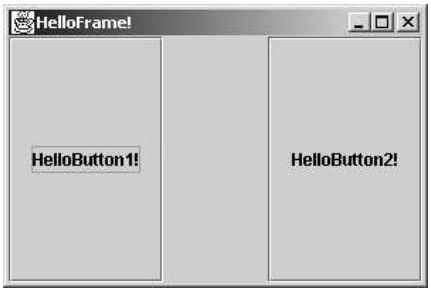
```
import javax.swing.*;

public class TestFrame extends JFrame
{
  public TestFrame(String title)//Konstruktor
  {
    super(title);
  }

  public static void main(String args[])
  {
    TestFrame myFrame = new TestFrame("HelloFrame!");//Instanz erzeugen
    myFrame.setSize(300,200);//Groesse setzen
    myFrame.show();
  }
}
```



Vererbungsbeziehungen zwischen AWT und Swing



Komponenten zu Containern hinzufügen



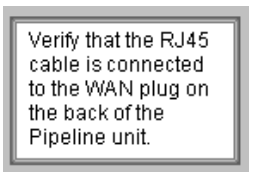

```
import javax.swing.*;
import java.awt.*;


public class TestFrame2 extends JFrame {
  public TestFrame2(String title) {
    super(title);
    this.getContentPane().add(new JButton("HelloButton1!"),BorderLayout.WEST);
    this.getContentPane().add(new JButton("HelloButton2!"),BorderLayout.EAST);
  }

  public static void main(String args[]) {
    TestFrame2 myFrame = new TestFrame2("HelloFrame!");//Instanz erzeugen
    myFrame.setSize(300,200);//Groesse setzen
    myFrame.show();
  }
}
```

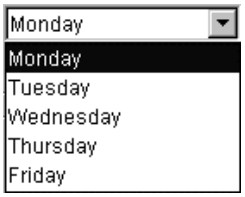

Swing-Komponenten

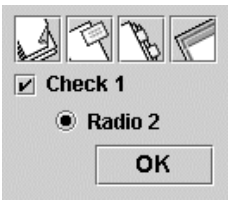
- JLabel
- JButton
- JTextField
- JPasswordField
- JTextArea
- JComboBox
- JCheckBox
- JRadioButton
- JList
- JTable
- JTree
- JScrollBar
- JMenu
- JMenuBar
- JMenuItem
- JSeparator
- JToolBar
- JToolTip
- JFileChooser
- ...

Swing-Komponenten (2)

- JLabel
  - einfacher Text
- JTextField
  - einzeiliges Eingabefeld
- JTextArea
  - mehrzeiliges Eingabefeld
- JPasswordField
  - Eingabefeld mit Sternchen statt Text
- JButton
  - Knopf
- JComboBox
  - Auswahlbox
- JCheckBox
  - Mehrfachselektion möglich
- JRadioButton
  - Alternative Selektion







Quelle: http://www.java.sun.com/docs/books/tutorial/

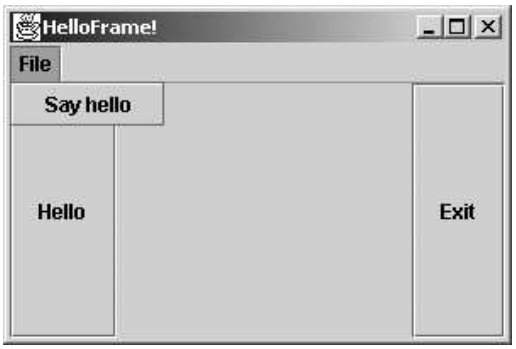
Swing-Komponenten (3)

- JList
  - Auswahlmenge mit möglicher Mehrfachselektion
- JTable
  - Tabellen-Darstellung von Daten
- JTree
  - Baumstruktur
  - auf- / zuklappbar
- JScrollPane
  - Pane mit Rollbalken
- JMenuBar
  - Menüzeile oben im Fenster
- JMenu
  - Menü mit mehreren Menüpunkten
- JMenuItem
  - Menüpunkt

Quelle: http://www.java.sun.com/docs/books/tutorial/

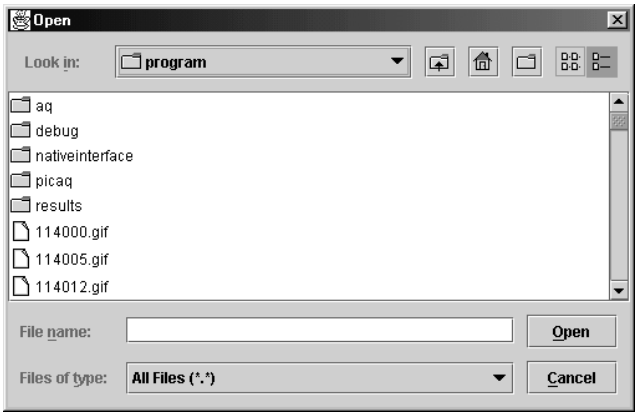
Menü-Beispiel

```
JMenuBar menuBar;
JMenu fileMenu;
JMenuItem helloItem;

menuBar = new JMenuBar();
fileMenu = new JMenu("File");
helloItem = new JMenuItem("Say hello");
fileMenu.add(helloItem);
menuBar.add(fileMenu);

helloItem.addActionListener(this);
this.setJMenuBar(menuBar);

public void actionPerformed(ActionEvent e)
{
  if(e.getSource()==helloItem)//Event
    System.out.println(„Hello“);
}
```



Swing-Komponenten (4)

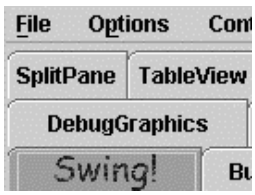
- JToolBar
  - Button-Zeile für schnellen Zugriff auf bestimmte Funktionen
- JToolTip
  - Information über GUI-Komponenten
- JFileChooser
  - vorhandene Komponente zur Dateiauswahl

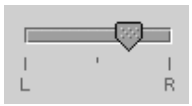


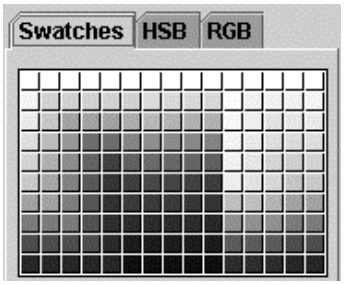
Quelle: http://www.java.sun.com/docs/books/tutorial/

Swing-Komponenten (5)

- JTabbedPane
  - Panel mit mehreren Karteireitern (‚Tabs')
  - vorderste ‚Karte' wird angezeigt
- JSplitPane
  - teilt einen Bereich horizontal oder vertikal ein
  - läßt sich verschieben
- JProgressBar
  - Fortschrittsbalken
- JSlider
- JColorChooser
  - Klasse für eine graphische Selektion von Farben
  - gibt ‚Color'-Objekt zurück







Quelle: http://www.java.sun.com/docs/books/tutorial/

JOptionPane für Dialoge

public static int showConfirmDialog(Component parentComponent, Object message, String title, int optionType);

public static String showInputDialog(Component parentComponent, Object message, String title, int messageType);

public static void showMessageDialog(Component parentComponent, Object message, String title, int messageType);

messageType:

ERROR_MESSAGE
INFORMATION_MESSAGE
WARNING_MESSAGE
QUESTION_MESSAGE
PLAIN_MESSAGE

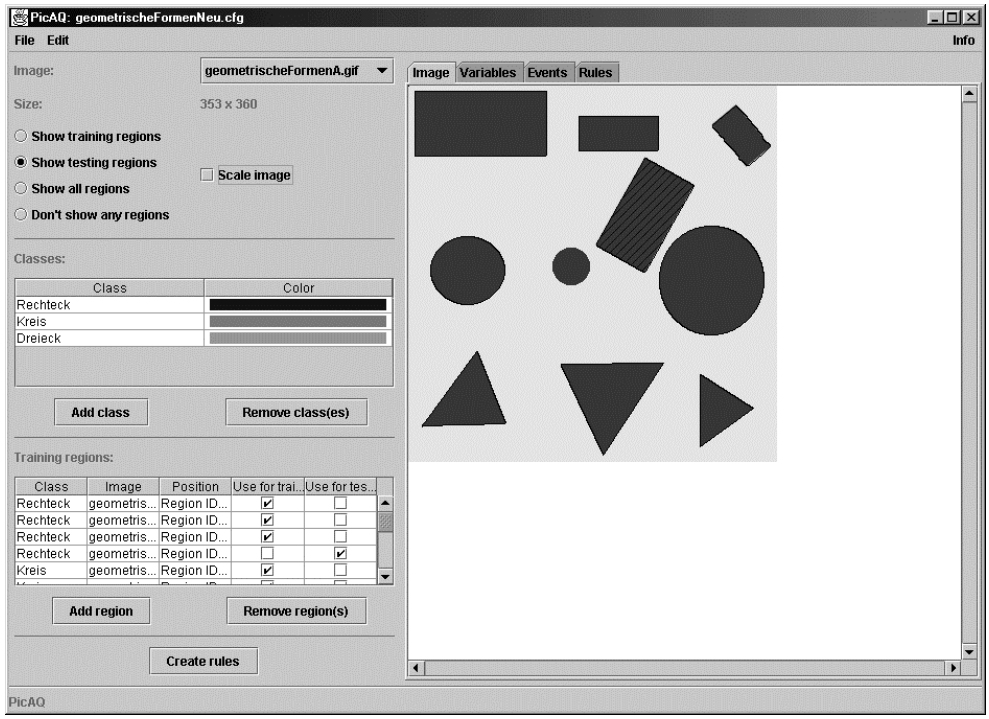
optionType:

DEFAULT_OPTION
YES_NO_OPTION
YES_NO_CANCEL_OPTION
OK_CANCEL_OPTION

Layoutmanager

- Anordnung der Komponenten in einem Container erfolgt über Layoutmanager
- Container bekommen Layoutmanager zugewiesen
- Beim Neuzeichnen (z.B. beim Ändern der Fenstergröße) einer Komponente werden die enthaltenen Komponenten durch den Layoutmanager neu ausgerichtet
- Der Layoutmanager legt fest
  - wo welche Komponenten stehen,
  - welche Ausrichtung die Komponenten haben,
  - ob und wie sich die Komponenten auf freien Platz ausdehnen.
- Layoutmanager befinden sich in den Packages
  - java.awt.*
  - javax.swing.*

GUI-Beispiel



Layoutmanager (2)

- **FlowLayout**
  - von links nach rechts, dann von oben nach unten
- **BorderLayout**
  - vier Himmelsrichtungen und Zentrum
- **CardLayout**
  - hintereinander mehrere Karten/Ebenen
- **GridLayout**
  - Zeilen und Spalten mit jeweils gleicher Größe
- **GridbagLayout**
  - komplexes Einteilen in Zeilen und Spalten, wobei eine Komponente sich über mehrere Zellen erstrecken kann
- **BoxLayout (javax.swing)**
  - horizontales oder vertikaltes Aufreihen, verschachtelt
- **OverlayLayout (javax.swing)**
  - überlagernde Komponenten

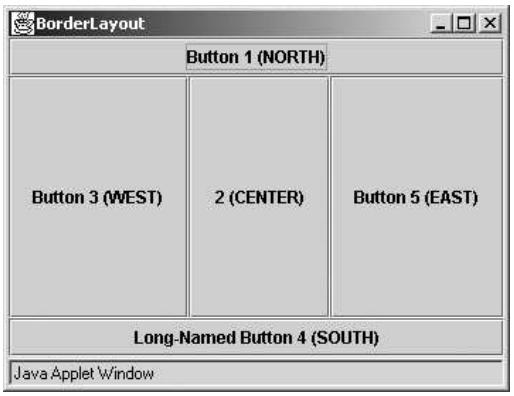
FlowLayout

Quelle: http://www.java.sun.com/docs/books/tutorial/

```
Container contentPane = getContentPane();
contentPane.setLayout(new FlowLayout());

contentPane.add(new JButton("Button 1"));
contentPane.add(new JButton("2"));
contentPane.add(new JButton("Button 3"));
contentPane.add(new JButton("Long-Named Button 4"));
contentPane.add(new JButton("Button 5"));
```

BorderLayout

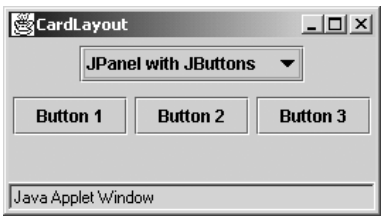
```
Container contentPane = getContentPane();

//unnötig, da default:
//contentPane.setLayout(new BorderLayout());

contentPane.add(new JButton("Button 1 (NORTH)"),
                BorderLayout.NORTH);
contentPane.add(new JButton("2 (CENTER)"),
                BorderLayout.CENTER);
contentPane.add(new JButton("Button 3 (WEST)"),
                BorderLayout.WEST);
contentPane.add(new JButton("Long-Named Button 4 (SOUTH)"),
                BorderLayout.SOUTH);
contentPane.add(new JButton("Button 5 (EAST)"),
                BorderLayout.EAST);
```

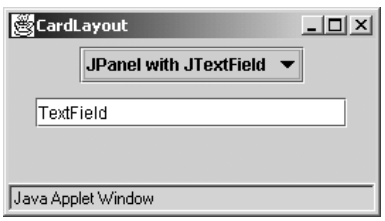


Quelle: http://www.java.sun.com/docs/books/tutorial/

CardLayout

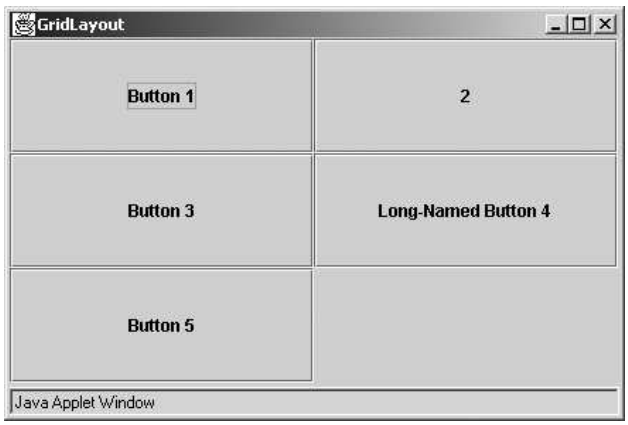
```
final static String BUTTONPANEL = "JPanel with JButtons";
final static String TEXTPANEL = "JPanel with JTextField";

JPanel cards = new JPanel();
cards.setLayout(new CardLayout());

JPanel p1 = new JPanel();
p1.add(new JButton("Button 1"));
p1.add(new JButton("Button 2"));
p1.add(new JButton("Button 3"));

JPanel p2 = new JPanel();
p2.add(new JTextField("TextField", 20));

cards.add(p1, BUTTONPANEL);
cards.add(p2, TEXTPANEL);
```





Quelle: http://www.java.sun.com/docs/books/tutorial/

GridLayout

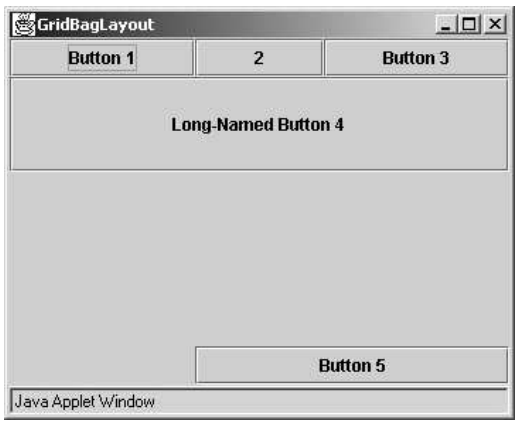


Quelle: http://www.java.sun.com/docs/books/tutorial/

```
Container contentPane = getContentPane();
contentPane.setLayout(new GridLayout(0,2));

contentPane.add(new JButton("Button 1"));
contentPane.add(new JButton("2"));
contentPane.add(new JButton("Button 3"));
contentPane.add(new JButton("Long-Named Button 4"));
contentPane.add(new JButton("Button 5"));
```


GridBagLayout

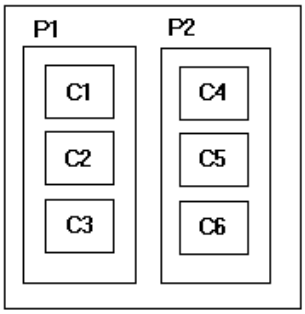
```
JButton button;
Container contentPane = getContentPane();
GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints c = new GridBagConstraints();
contentPane.setLayout(gridbag);

button = new JButton("Button 1");
c.fill = GridBagConstraints.HORIZONTAL;
c.weightx = 0.5;
c.gridx = 0;
c.gridy = 0;
gridbag.setConstraints(button, c);
contentPane.add(button);

button = new JButton("Long-Named Button 4");
//...
c.gridwidth = 3;
c.gridx = 0;
c.gridy = 1;
//...
```

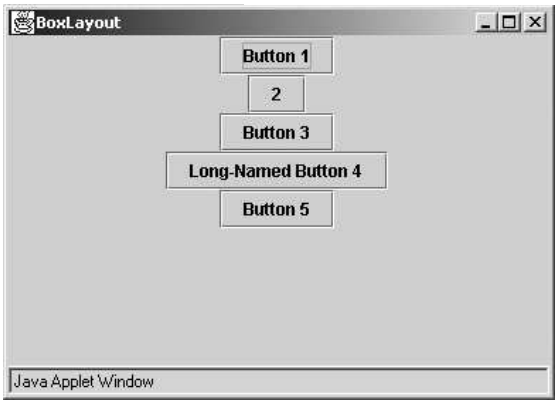


Quelle: http://www.java.sun.com/docs/books/tutorial/

BoxLayout

```
Container contentPane = getContentPane();
contentPane.setLayout(new BoxLayout(contentPane,
                                    BoxLayout.Y_AXIS));

JButton b1 = new JButton("Button 1");
b1.setAlignmentX(Component.CENTER_ALIGNMENT);
JButton b2 = ...
b2.setAlignmentX(...);
...
JButton b5 = ...;
b5.setAlignmentX(...);

contentPane.add (b1);
...
contentPane.add (b5);
```





Quelle: http://www.java.sun.com/docs/books/tutorial/

Model-View-Controller

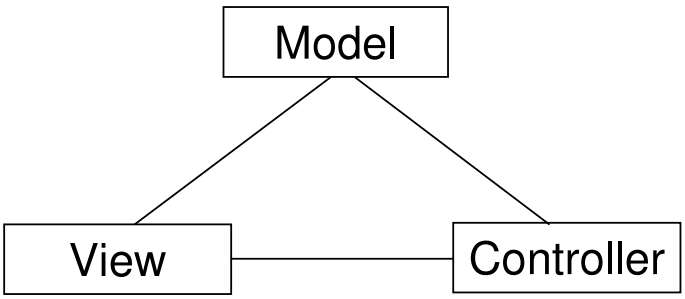
- Trennung von Inhalt, Form und Funktion

- **Model** hält den Zustand
- **View** stellt die Daten dar
- **Controller** verwaltet Interaktion mit dem Anwender

- saubere Trennung zwischen Inhalt und Darstellung
- konsistente Darstellung von Daten aus unterschiedlichen Sichten
- einheitlicher Zugriff über Controller
- Beispiel: Wertmenge als Liste, Balkengraphik und Kuchendiagramm anzeigen
  - keine Redundanz und keine Inkonsistenzen, da auf gleichen Daten gearbeitet wird

Model-View-Controller (2)

- benachrichtigt Views bei Änderungen



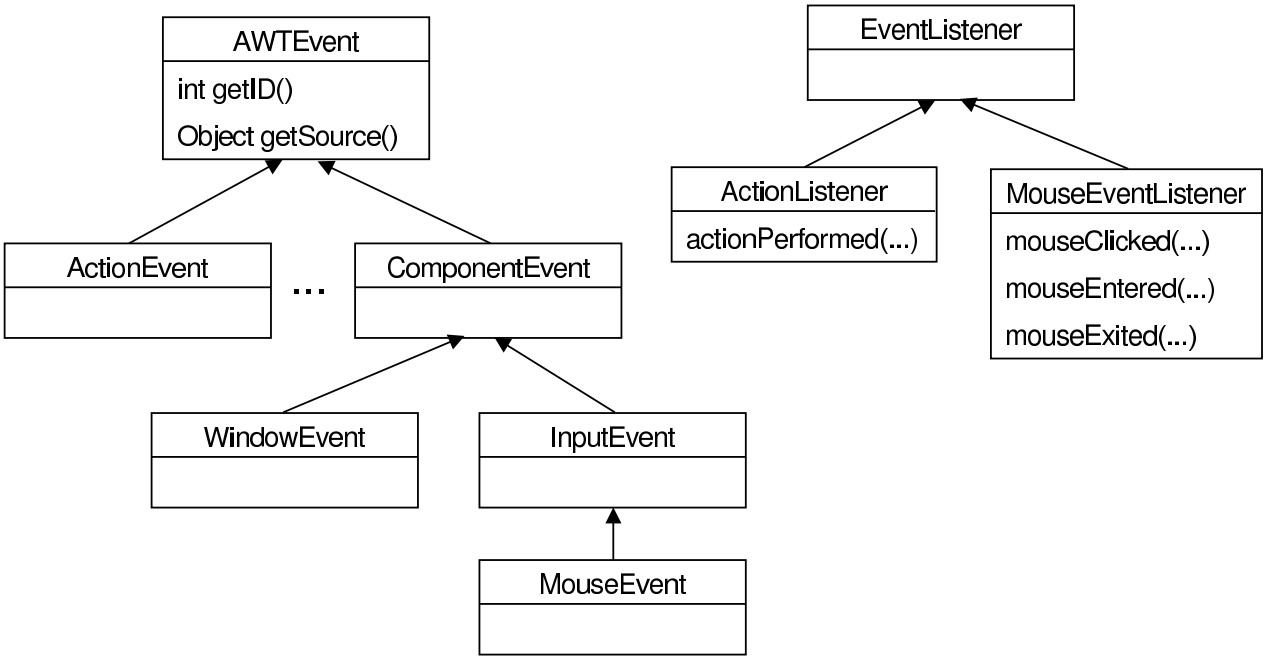
- muß korrekte Darstellung des Models sicherstellen

- bietet Möglichkeiten, mit denen das UI auf Eingaben reagieren kann

Events und Listener

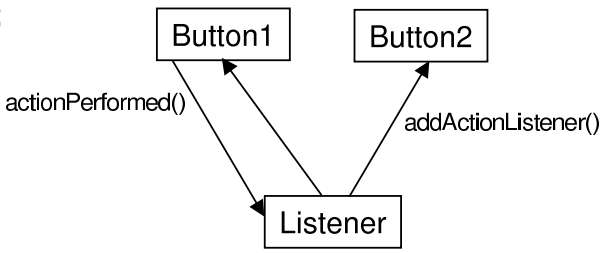
- Modell zur Ereignisverarbeitung ist im JDK enthalten
- Events werden durch Methodenaufruf verschickt
- Listener registrieren sich bei Komponenten als Interessenten
- Listener-Klassen müssen bestimmtes Listener-Interface implementieren um
  - sich als Listener anmelden zu können und
  - auf Events durch entsprechende Methoden reagieren zu können.
- z.B. bei Button als ActionListener anmelden
  - Implementieren der Methode beim Listener:
    void actionPerformed(ActionEvent e);
  - Hinzufügen als Listener:
    button.addActionListener(myListener);

Events und Listener (2)



ActionListener

```
class MyListenerExample implements ActionListener
{
    JButton myExitButton = new JButton(„Exit“);

    public MyListenerExample()
    {
        myExitButton.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource()==myExitButton) //Bedingungen...
        {
            System.exit(0); //Und weg!
        }
    }
}
```



Beispiele für Listener

Komponente	Listener	Methoden im Interface
JButton	ActionListener	actionPerformed(ActionEvent)
JFrame	WindowListener	windowClosed(WindowEvent) windowIconified(WindowEvent) ...
	MouseMotionListener	mouseMoved(MouseEvent) mouseDragged(MouseEvent)
JTextArea	KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyTyped)
...		

Wichtige Packages, Klassen und Interfaces

- javax.swing.*
  - JButton, JCheckBox, JComboBox, JDialog, JFileChooser, JFrame, JLabel, JList, JMenu (etc.), JOptionPane, JPanel, JRadioButton, JScrollPane, JSeparator, JSplitPane, JTabbedPane, JTable, JTextArea, JTextField, JTree, ...
- java.awt.*
  - LayoutManager, CardLayout, FlowLayout, Graphics2D, GridBagConstraints, GridBagLayout, GridLayout, Insets, ...
- java.awt.event.*
  - ActionListener, KeyListener, MouseListener, MouseMotionListener, WindowListener, ActionEvent, KeyEvent, MouseEvent, WindowEvent, ...

Zusammenfassung

- Swing ist eine komplexe Klassenbibliothek innerhalb des JDK zur Erstellung von GUIs
- setzt auf dem AWT auf, besteht jedoch nur aus ‚leightweight' Komponenten (rein Java)
- viele vorhanden GUI-Elemente, die in Containern zusammengefaßt werden können:
  - Menüs, Buttons, Listen, Tabellen, Auswahlboxen, Textfelder, ...
- Ausrichtung der Komponenten durch LayoutManager
- Möglichst saubere Trennung durch Model-View-Controller-Konzept
- Event-Handling für Maus- und Tastaturinteraktion
  - ActionListener, MouseListener, ...

Sonstiges

- Diese Folien und das zweite Übungsblatt sind über die SWP2001-Seite erreichbar:
  - http://www.db.informatik.uni-bremen.de/teaching/courses/ss2001_swp/
- dort liegt auch eine Beispiellösung für Teile des ersten Übungsblatts, die zur Bearbeitung des zweiten Übungsblatts verwendet werden kann
- die JDK-API sowie ein gutes Java-Tutorial (auch zu Swing) liegen unter
  - http://www.java.sun.com/j2se/1.3/docs/api/index.html
  - http://www.java.sun.com/docs/books/tutorial/
- die Übung geht von 13 – 16 (?) Uhr