AWT UND SWING

AWT = abstract windowing toolkit

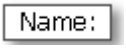
Bausteine	

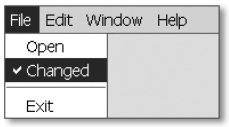
Eigenes Fenster
Programmcode steht nicht im Main, sondern in einer separaten Klasse
`extends JFrame`
	`Fenster f = new Fenster();`
	`f.setSize(400,200);`
	`f.setVisible(true);`
	`f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`
	`f.setTitle("Titel");`

JWindow
selbstständiges Fenster, ohne Rahmen und Titel
	`JWindow win = new JWindow();`
	`win.setVisible(true);`
	`win.setSize(500,300);`

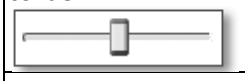
JFrame („Standardfenster“)
Window + Rahmen + Titel
Kann Menüleisten enthalten
	`JFrame frame = new JFrame("JFrame 1");`
	`frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`
	`frame.setVisible(true);`
	`frame.setSize(500,300);`

Container
Für Verschachtelungen
Diverse Layouts
	`Container cp = this.getContentPane(); //Setzt Container in Window`
	`cp.setLayout(new BorderLayout());`
	`cp.add(new JLabel("Text"),BorderLayout.CENTER);`
	`cp.add(new JButton("Oben"),"North");`

JPanel Ähnlich wie Container Von Container abgeleitet	`JPanel p = new JPanel();` `p.add(new JButton("Button"));`
JLabel Textanzeige 	`JLabel jlbl = new JLabel("Label");`
JButton Knopf 	`JButton b1 = new JButton("Links");` `b1.addActionListener(this);`

JMenuBar Menuleiste
JMenu
JMenuItem
 
	`JMenuBar jmb = new JMenuBar();`
	`JMenu jm_file = new JMenu("Datei");`
	`JMenuItem jmi_file_new = new JMenuItem("Neu");`
	`jm_file.add(jmi_file_new);`
	`jm_file.addSeparator();`
	`jmb.add(jm_file);`
	`this.setJMenuBar(jmb); //Setzt JMenuBar in Window`

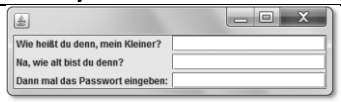
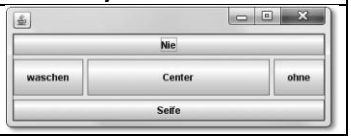
JFileChooser
Dialogfenster
	`JFileChooser fc_save = new JFileChooser();`
	`int returnVal = fc_save.showSaveDialog(fc_save);`
	`if (returnVal == JFileChooser.APPROVE_OPTION) { }`

JSlider
 
	`js = new JSlider(JSlider.VERTICAL);`
	`js.addChangeListener(this);`
	`cp.add(js,BorderLayout.EAST);`

JTextArea	`JTextArea jta = new JTextArea();`
	`jta.setSize(200,300);`
	`jta.append("Aktion 1" + '\n');`
	`jta.setForeground (Color.blue);`

JScrollPane 	`JScrollPane scrollPane = new JScrollPane(jta);` `this.add(scrollPane);`
Canvas	`Canvas myCanvas = new Canvas();` `this.getContentPane().add(myCanvas);`

Layouts

FlowLayout	**GridLayout**	**CardLayout**	**BorderLayout**
			
Die Komponenten werden nacheinander in den Container gelegt	Die Komponenten werden in einem Gitter angeordnet	Die Kompoenten werden wie Registerkarten übereinandergelegt	Die Komponenten werden den fünf Regionen zugeordnet

Ereignisse

ActionEvent
Irgendeine Aktion wurde durchgeführt
z.B. ein Button wurde gedrückt

`implements ActionListener`

```
b1.addActionListener(new ButtonListener());
```

```
class MeinButtonListener implements ActionListener{
   public void actionPerformed(ActionEvent e) {
      if(e.getActionCommand().equals("Knopf"))
         ...
   }
}
```

AdjustementEvents
Es wurde etwas verändert
z.B. der Schieber einer Scrollbar wurde betätigt

`implements ChangeListener`

```
js.addChangeListener(new ChangeListener());
```

```
public void stateChanged(ChangeEvent e) { }
```

KeyEvent
Es wurde eine Taste gedrückt
in einem Textfeld oder in einem Fenster

`implements KeyListener`

```
jta.addKeyListener(this KeyListener());
```

```
public void keyPressed(KeyEvent e) { }
public void keyReleased(KeyEvent arg0) { }
public void keyTyped(KeyEvent arg0) { }
```

MouseEvent
Die Maus wurde bewegt,
oder eine der Maustasten wurde betätigt

`implements MouseListener`
oder
`implements MouseMotionListener`

```
myCanvas.addMouseMotionListener(new MyListener(myCanvas));
```

```
public void mouseDragged(MouseEvent e) {}
public void mouseMoved(MouseEvent e) {}
```

WindowEvent
Irgendetwas wurde mit dem Fenster angestellt
Minimieren, wiederherstellen, schliessen

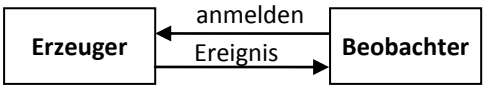
`implements WindowListener`

```
this.addWindowListener(new WindowListener());
```

```
class MeinWindowListener implements WindowListener{
   public void windowClosing(WindowEvent e) {
      System.exit(0);
   }
   public void windowActivated(WindowEvent e){}
   public void windowClosed(WindowEvent e){}
   public void windowDeactivated(WindowEvent e){}
   public void windowDeiconified(WindowEvent e){}
   public void windowIconified(WindowEvent e){}
   public void windowOpened(WindowEvent e){}
}
```

ItemEvent
In einer Liste oder in einer Checkbox
wurde ein Eintrag (Item) ausgewählt

TextEvent
Der Text in einem Textfeld wurde verändert



Aufruf

Direkt	`jbut.addActionListener(new ButtonListener());`
Indirekt	`ButtonListener ButList = new ButtonListener();` `jbut.addActionListener(ButList);`
Mit Konstruktor	`ButtonListener ButList = new ButtonListener(jbut);` `jbut.addActionListener(ButList);`

Listener

Ohne Unterscheidung

```
class MyListener implements ActionListener {
   public void actionPerformed(ActionEvent e) {
      System.out.print("+");
   }
}
```

Mit Unterscheidung

```
class MyListener implements ActionListener {
   public void actionPerformed(ActionEvent e) {
      if (e.getActionCommand().equals("But")) ... ;
      //oder
      if (e.getActionCommand == "But") ... ;
   }
}
```

Mit Konstruktor

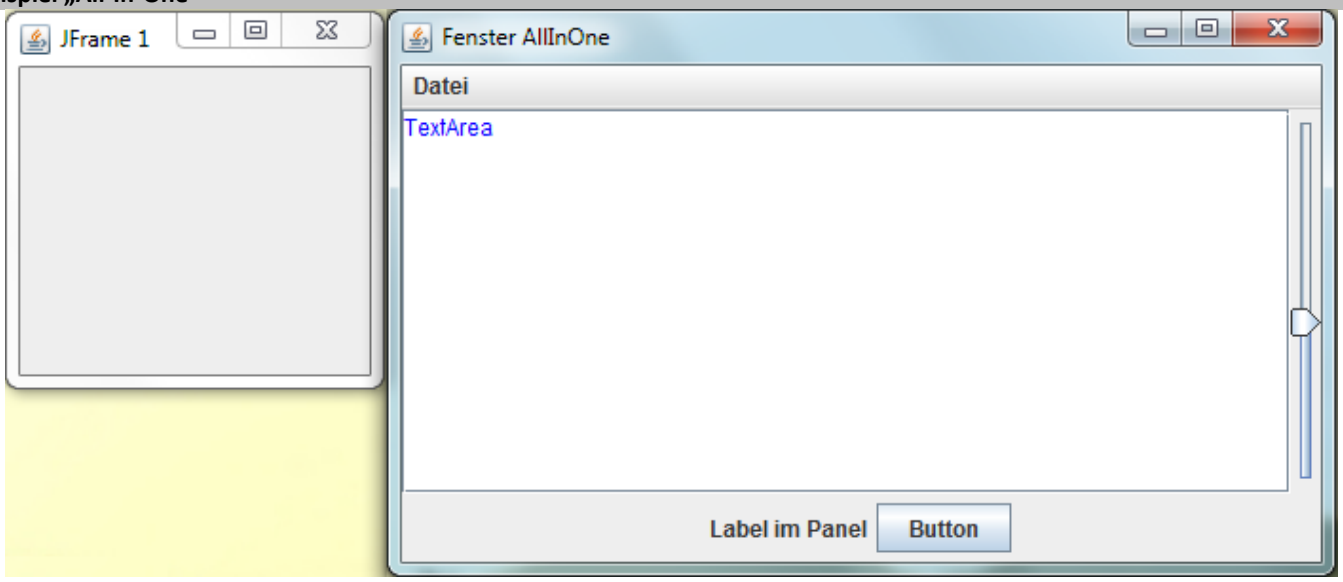
```
class ButtonListener implements ActionListener {
   JButton But = new JButton();                          //Deklaration
   ButtonListener(JButton But){ this.But = But;  }       //Konstruktor
   public void actionPerformed(ActionEvent e) {          //
      if (e.getSource() == But) System.out.print("+");
   }
}
```

Painting / Animation

- Jede „Component“ hat einen Grafikkontext: **graphic**: Graphics
- In diesem Kontext kann gezeichnet werden
  ```
  graphic.drawLine(...);
  graphic.drawOval(...);
  graphic.fillOval(...);
  ```
- Jede Component hat eine Methode, die die Komponente zeichnet
  ```
  void paint(Graphics g);
  ```
- **`paint()`** wird immer dann vom System aufgerufen, wenn es nötig erscheint
- Mit **`repaint()`** kann der Programmierer ein Neuzeichnen veranlassen

Link Swing Components

Beispiel „All-In-One“




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

public class AllInOne extends JFrame {
   public AllInOne(){ //Eigenes Fenster
      //Container
      Container cp = this.getContentPane();
      cp.setLayout(new BorderLayout());
      //JButton
      JButton JBut = new JButton("Button");
      //Panel und Panel
      JPanel p = new JPanel();
      p.setLayout(new FlowLayout());
      p.add(new JLabel("Label im Panel"));
      p.add(JBut);
      cp.add(p,"South");
      //JSlider
      JSlider JSlid = new JSlider(JSlider.VERTICAL);
      cp.add(JSlid,BorderLayout.EAST);
      //JMenuBar, JMenu, JMenuItem, Separator
      JMenuBar jmb = new JMenuBar();
      JMenu jm_file = new JMenu("Datei");
      JMenuItem jmi_file_save = new JMenuItem("Speichern");
      JMenuItem jmi_file_paint = new JMenuItem("Zeichnen");
      jm_file.add(jmi_file_save);  jm_file.addSeparator();
      jm_file.add(jmi_file_paint);
      jmb.add(jm_file);
      this.setJMenuBar(jmb);
      //JTextArea
      JTextArea jta = new JTextArea();
      jta.setSize(200,300);
      jta.append("TextArea" + '\n');
      jta.setForeground(Color.blue);
      cp.add(jta,"Center");
      //JScrollPane
      JScrollPane scrollPane = new JScrollPane(jta);
      this.add(scrollPane);
      //Listener
      MyListener myl = new MyListener(JBut, jmi_file_save, jmi_file_paint, JSlid);
      JBut.addActionListener(myl);
      jmi_file_paint.addActionListener(myl); jmi_file_save.addActionListener(myl);
      JSlid.addChangeListener(myl);
      jta.addKeyListener(myl);
   }
```

```
   public static void ShowCanvas(){
      //JFrame
      JFrame frame = new JFrame("JFrame 1");
      frame.setSize(200,200); frame.setVisible(true);
      //Canvas
      Canvas myCanvas = new Canvas();
      myCanvas.addMouseMotionListener(new MyMouseListener(myCanvas));
      frame.getContentPane().add(myCanvas);
   }
   public static void main(String[] args) {
      AllInOne win1 = new AllInOne();
      win1.setSize(500,300); win1.setVisible(true);
      win1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      win1.setTitle("Fenster AllInOne");
//    JWindow:
//    JWindow win = new JWindow();
//    win.setSize(100,100);
//    win.setVisible(true);
   }
}

class MyListener implements ActionListener, ChangeListener, KeyListener{
   JButton JBut; JMenuItem jmi_file_save; JMenuItem jmi_file_paint; JSlider JSlid;
   public MyListener(JButton JBut, JMenuItem jmi_file_save, JMenuItem jmi_file_paint,
          JSlider JSlid){
      this.JBut = JBut;
      this.jmi_file_save = jmi_file_save; this.jmi_file_paint = jmi_file_paint;
      this.JSlid = JSlid;
   }
   public void actionPerformed(ActionEvent e) {                          //ActionListener
      if(e.getSource() == JBut) System.out.println("Button");
      if(e.getSource() == jmi_file_save){
         //JFileChooser
         JFileChooser fc_save = new JFileChooser();
         int returnVal = fc_save.showSaveDialog(fc_save);
         if (returnVal == JFileChooser.APPROVE_OPTION) { }
      }
      if(e.getSource() == jmi_file_paint)     AllInOne.ShowCanvas();
   }
   public void stateChanged(ChangeEvent e) {                             //ChangeListener
      if(e.getSource() == JSlid)   System.out.println(JSlid.getValue());
   }
   public void keyPressed(KeyEvent e) { }                                //KeyListener
   public void keyReleased(KeyEvent e) { }
   public void keyTyped(KeyEvent e) {
      System.out.print(e.getKeyChar());
   }
}

class MyMouseListener implements MouseMotionListener {                   //MouseListener
   Canvas canvas;
   MyMouseListener(Canvas canvas) { this.canvas = canvas; }
   public void mouseDragged(MouseEvent e) {
      int x = e.getPoint().x;
      int y = e.getPoint().y;
      Graphics g = canvas.getGraphics();
      g.fillOval(x, y, 5, 5);
   }
   public void mouseMoved(MouseEvent arg0) { }
}
```