Swing Kompakt

Jan Krüger
jkrueger@techfak.uni-bielefeld.de

Übersicht

- ein Rückblick . . .
- Application vs. Applet
- LayoutManager : FlowLayout, GridLayout, BorderLayout, . . .
- Basiskomponenten: Panels, Buttons, TextElemente, Listen, Menüs, . . .
- Ereignissteuerung : ActionListener, WindowListener, . . .
- individuelles Zeichnen . . .

ein Rückblick . . .

- Java 1.0 : **AWT** (Abstract Window Toolkit) - package java.awt.*
- Java 1.1 : OO - Eventhandling - JFC/Swing als AddOn
- seit Java 1.2 : **JFC** (Java Foundation Classes) oder auch **Swing** als Standard GUI - API

AWT	**JFC**
- nicht OO	- vollständig OO und in Java
- minimales Set an Widgets	- einheitliches Look Feel
- benutzt die OS eigenen Widgets	- Look Feel durch Themes änderbar
- . . .	- vergleichsweise langsam

ein einfaches Applet

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

public class HelloWorldApplet extends JApplet {

    public void init() {
        getContentPane().add(new JLabel("Hello World"));
    }
}
```

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <applet code="HelloWorldApplet" width="200" height="100"></applet>
  </body>
</html>
```

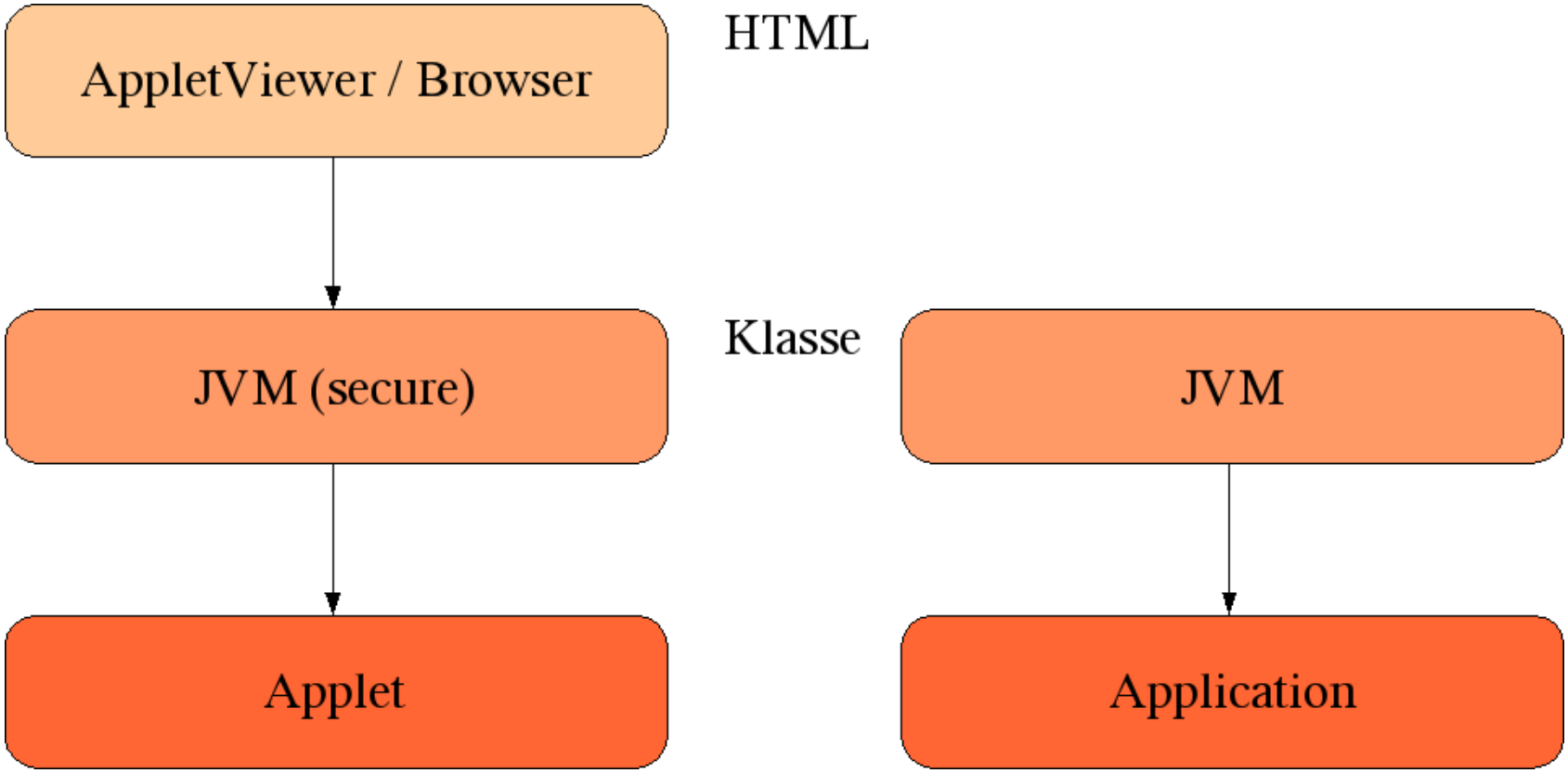
eine einfache Applikation (1)

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

public class HelloWorldApplication extends JFrame {

    public HelloWorldApplication() {
       getContentPane().add(new JLabel("Hello World"));
    }

    public static void main(String [] args) {
        HelloWorldApplication helloworld = new HelloWorldApplication();
        HelloWorld.pack();
        HelloWorld.show();
    }
}
```

eine einfache Applikation (2)

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

public class HelloWorldApplication  {

    private JFrame jframe = new JFrame();

    public HelloWorldApplication() {
        jframe.getContentPane().add(new JLabel("Hello World"));
    }

    public void run() {
        jframe.pack();
        jframe.show();
    }

    public static void main(String [] args) {
        HelloWorldApplication helloworld = new HelloWorldApplication();
        helloworld.run();
    }
}
```

Applet vs. Application



ein Button und ...

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

public class MyButton extends JFrame {

    JButton button;

    public MyButton() {
        Container cp = getContentPane();
        button = new JButton("Button 1");
        cp.add(button);
    }

    public static void main (String [] args) {
        MyButton1 mybutton = new MyButton();
        mybutton.pack();
        mybutton.show();
    }
}
```

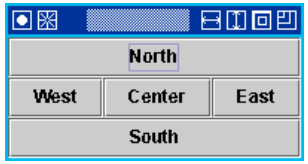
. . . eine einfache Ereignissteuerung

```
public class MyButton extends JFrame {

    JButton button;

    public MyButton() {
        Container cp = getContentPane();
        button  = new JButton("Button 1");
        button.addActionListener(new ActionLister(){
            public void actionPerformed(ActionEvent e) {
              System.out.println("Der Button wurde gedrückt!");
            }
        });
        cp.add(button);
    }

    public static void main(String [] args) {
        MyButton mybutton = new MyButton();
        mybutton.pack();
        mybutton.show();
    }
}
```

Layout

- JApplet, JFrame, JWindow und JDialog
  - produzieren mittels `getContentPane()` einen Container
  - einem Container kann mittels `setLayout(LayoutManager)` ein anderes Layout verpaßt werden
- JPanel und andere direkte Unterklassen von JComponent können direkt mittels `setLayout()` modifiziert werden.
- typische LayoutManager : BorderLayout, FlowLayout, GridLayout, GridBagLayout, BoxLayout

BorderLayout




```
public class MyBorderLayout extends JFrame{

    public MyBorderLayout(){
        Container cp = getContentPane();
        cp.setLayout(new BorderLayout());
        cp.add(BorderLayout.NORTH,new JButton("North"));
        cp.add(BorderLayout.SOUTH,new JButton("South"));
        cp.add(BorderLayout.EAST,new JButton("East"));
        cp.add(BorderLayout.WEST,new JButton("West"));
        cp.add(BorderLayout.CENTER, new JButton("Center"));
    }

   public static void main (String [] args) {
        MyBorderLayout test = new MyBorderLayout();
        test.pack();
        test.show();
    }
}
```

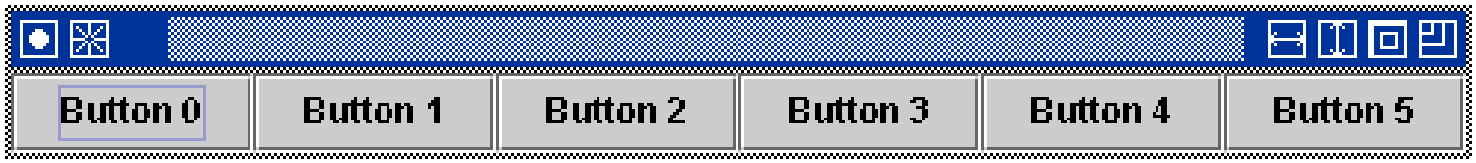
FlowLayout


```
public class MyFlowLayout extends JFrame{

    public MyFlowLayout(){
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        for (int i = 0; i < 5; i++){
            cp.add(new JButton("Button "+i));
        }
    }

 public static void main (String [] args) {
        MyFlowLayout test = new MyFlowLayout();
        test.pack();
        test.show();
    }
}
```

GridLayout


```
public class MyGridLayout extends JFrame{

    public MyGridLayout(){
        Container cp = getContentPane();
        cp.setLayout(new GridLayout(2,3));
        for (int i = 0; i < 6; i++){
            cp.add(new JButton("Button "+i));
        }
    }

    public static void main (String [] args) {
        MyGridLayout test = new MyGridLayout();
        test.pack();
        test.show();
    }
}
```

BoxLayout



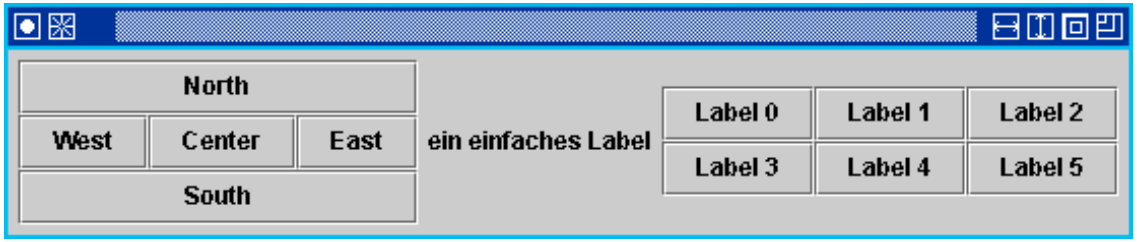

```
public class MyBoxLayout extends JFrame {

 public MyBoxLayout(){
  Container cp = getContentPane();
  cp.setLayout(new BoxLayout(cp,BoxLayout.Y_AXIS));
  for (int i = 0; i < 6; i++){
     cp.add(new JButton("Button "+i));
  }
 }

 public static void main (String [] args) {
   MyBoxLayout test = new MyBoxLayout();
    test.pack();
    test.show();
   }
}
```

JPanel

- Behälter für andere JComponenten (JPanel, JButtons, JTextField, ...)
- ein JPanel zeichnet nur seinen Hintergrund (default)
- benutzt einen LayoutManager zum JComponenten im JPanel anzuordnen (default : `FlowLayout()`)

$\rightarrow$ häufigste Anwendung : Positionieren von JCompents in JFrame, JApplet, ...

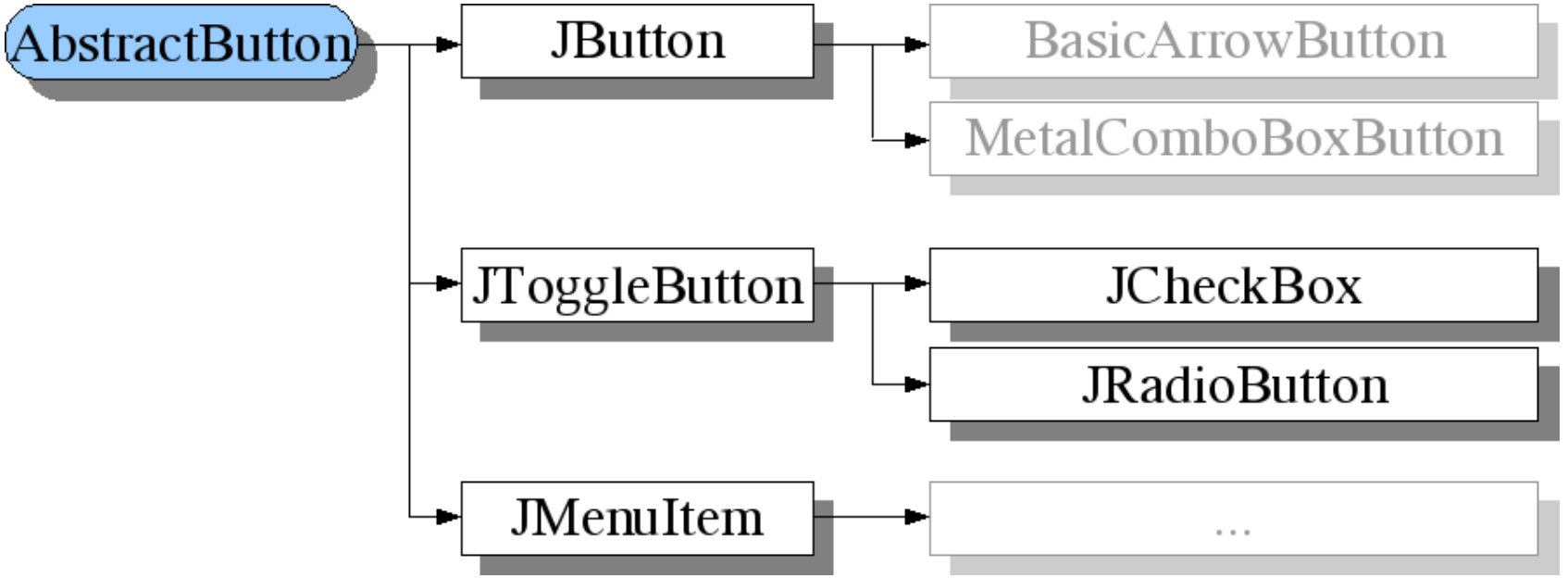
JPanel - Beispiel




```
JPanel first = new JPanel();
JPanel second = new JPanel();
public MyPanel() {
  Container cp = getContentPane();
  cp.setLayout(new FlowLayout());
  first.setLayout(new BorderLayout());
  first.add(BorderLayout.NORTH,new JButton("North"));
  ...
  first.add(BorderLayout.CENTER,new JButton("Center"));
  cp.add(first);
  cp.add(new JLabel("ein einfaches Label"));
  second.setLayout(new GridLayout(2,3));
  for (int i = 0; i < 6; ++i){
      second.add(new JButton("Label "+i));
  }
  cp.add(second);
...
```

Button

- Basistyp : `AbstractButton`
- Layout : Text, Icons, `...`
- Klassenabhängigkeiten:



- Ereignis : `ActionEvent`, Interface : `ActionListener`

JButton


```
public class MyJButton extends JFrame {

    public ImageIcon icon;

    public MyJButton() {
        // Icon
        icon = new ImageIcon(getClass.getResource("images/back.png"));
        // JButton
        JButton a = new JButton("back");
        JButton b = new JButton(icon);
        JButton c = new JButton("back",icon);
        // ----------------------------
        Container cp = getContentPane();
        //cp.setLayout(new FLowLayout());
        cp.add(BorderLayout.WEST,a);
        cp.add(BorderLayout.CENTER,b);
        cp.add(BorderLayout.EAST,c);
    }
...
```

JToggleButton


```
public class MyJToggleButton extends JFrame {

    public ImageIcon icon;

    public MyJToggleButton() {
        // Icon
        icon = new ImageIcon(getClass().getResource("images/back.png"));
        // JToggleButton
        JToggleButton a = new JToggleButton("back");
        JToggleButton b = new JToggleButton(icon);
        JToggleButton c = new JToggleButton("back",icon);
        JToggleButton d = new JToggleButton("back",true);
        JToggleButton e = new JToggleButton(icon,true);
        JToggleButton f = new JToggleButton("back",icon,true);
        // ----------------------------
...
```

JCheckBox

```
public MyJCheckBox() {
     // Icon
     icon[0] = new ImageIcon(getClass().getResource("images/back.png"));
     icon[1] = new ImageIcon(getClass().getResource("images/back_over.png"));
     icon[2] = new ImageIcon(getClass().getResource("images/back_press.png"));

     // JRadioButton
     JCheckBox a = new JCheckBox("back");
     JCheckBox b = new JCheckBox(icon[0]);
     b.setRolloverIcon(icon[1]);
     b.setSelectedIcon(icon[2]);
     JCheckBox c = new JCheckBox("back",icon[0]);
     c.setRolloverIcon(icon[1]);
     c.setSelectedIcon(icon[2]);
     // ----------------------------
...
```

JRadioButton

```
...
public MyJRadioButton() {
   // Icons
   icon[0] = new ImageIcon(getClass().getResource("images/back.png"));
   icon[1] = new ImageIcon(getClass().getResource("images/back_over.png"));
   icon[2] = new ImageIcon(getClass().getResource("images/back_press.png"));
   // JRadioButton
   JRadioButton a = new JRadioButton("back");
   JRadioButton b = new JRadioButton(icon[0]);
   b.setRolloverIcon(icon[1]);
   b.setSelectedIcon(icon[2]);
   JRadioButton c = new JRadioButton("back",icon[0]);
   c.setRolloverIcon(icon[1]);
   c.setSelectedIcon(icon[2]);
   // JButtonGroup
   ButtonGroup bg = new ButtonGroup();
   bg.add(a);
   bg.add(b);
   bg.add(c);
...
```

Text Elemente

- JLabel - durch den Benutzer unveränderlich
- JTextField - einzeiliges Textfeld
- JTextArea - mehrzeiliges Textfeld
- ähnliche Methoden zum Verändern des Inhalts :
  - `setText(String) : void`
  - `getText(void) : String`

JLabel


```
public class MyJLabel extends JFrame{

    JLabel label = new JLabel("");

    public MyJLabel (){
        Container cp = getContentPane();
        cp.add(label);
        label.setText("Dies ist ein Label ...");
        String text = label.getText();
    }

    public static void main(String [] args) {
        MyJLabel myjlabel = new MyJLabel();
        myjlabel.pack();
        myjlabel.show();
    }
}
```

JTextField


```
public class MyJTextField extends JFrame{

    JTextField textfield = new JTextField(30);

    public MyJTextField (){
        Container cp = getContentPane();
        cp.add(textfield);
        textfield.setText("Dies ist ein JTextField ...");
        String text = textfield.getText();
    }

    public static void main(String [] args) {
        MyJTextField myjtf = new MyJTextField();
        myjtf.pack();
        myjtf.show();
    }
}
```

JComboBox


```
public class MyJComboBox extends JFrame {

    JComboBox jcombobox = new JComboBox();
    String [] elements = {"eins", "zwei", "drei", "vier", "..."};

    public MyJComboBox() {
        Container cp = getContentPane();
        for (int i = 0; i < elements.length; ++i) {
            jcombobox.addItem(elements[i]);
        }
        // jcombobox.setEditable(true);
        cp.add(jcombobox);
    }

    public static void main (String [] args) {
        MyJComboBox myjb = new MyJComboBox();
        myjb.pack();
        myjb.show();
    }
}
```

JList

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

public class MyJList extends JFrame {

    private String [] elements = {"eins", "zwei", "drei", "vier", "..."};
    private JList jlist = new JList(elements);

    public MyJList() {
        Container cp = getContentPane();
        cp.add(jlist);

    }

    public static void main (String [] args) {
        MyJList myjl = new MyJList();
        myjl.pack();
        myjl.show();
    }

}
```

Menüs

```
public class MyJMenu extends JFrame{
    private JMenuBar jmenubar = new JMenuBar();
    private JMenu jmenu1 = new JMenu("Menue 1");
    private JMenu jmenu2 = new JMenu("Menue 2");
    private JMenuItem item1 = new JMenuItem("MenueItem 1");
    private JMenuItem item2 = new JMenuItem("MenueItem 2");
    public MyJMenu() {
        Container cp = getContentPane();
        jmenu1.add(item1);
        jmenu1.add(item2);
        jmenubar.add(jmenu1);
        jmenubar.add(jmenu2);
        cp.add(jmenubar);
    }
    public static void main (String [] args) {
        MyJMenu test = new MyJMenu();
        test.pack();
        test.show();
    }
}
```

Ereignissteuerung

- eine Komponente kann ein Ereignis (Event) auslösen
- jedes Event wird durch eine Klasse repräsentiert (z.B. `ActionListener`
- eine ausgelöstes Event kann durch ein oder mehrere "Listener" verarbeitet werden
- $\rightarrow$ strikte Trennung von Quelle des Events und Ziel der Verarbeitung ist möglich (und häufig sinnvoll)
- ähnliche Methoden zum Hinzufügen und Entfernen von Listenern : **addXXXListener(XXXListener)** und **removeXXXListener()**

Listener als anonyme Klasse

```
public class MyButton extends JFrame {

    JButton button;

    public MyButton() {
        Container cp = getContentPane();
        button  = new JButton("Button 1");
        button.addActionListener(new ActionLister(){
            public void actionPerformed(ActionEvent e) {
              System.out.println("Der Button wurde gedrückt!");
            }
        });
        cp.add(button);
    }

    public static void main(String [] args) {
        MyButton mybutton = new MyButton();
        mybutton.pack();
        mybutton.show();
    }
}
```

. . . externe Klasse

```
public class MyActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Der Button wurde gedrückt!");
    }
}

public class MyButton extends JFrame {

    JButton button;

    public MyButton() {
        Container cp = getContentPane();
        button  = new JButton("Button 1");
        button.addActionListener(new MyActionLister());
        cp.add(button);
    }

    public static void main(String [] args) {
        MyButton mybutton = new MyButton();
        mybutton.pack();
        mybutton.show();
    }
}
```

. . . eigene interne Klasse

```
public class MyButton extends JFrame {

    JButton button;

    class MyActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            System.out.println("Der Button wurde gedrückt!");
        }
    }

    public MyButton() {
        Container cp = getContentPane();
        button  = new JButton("Button 1");
        button.addActionListener(new MyActionLister());
        cp.add(button);
    }

    public static void main(String [] args) {
        MyButton mybutton = new MyButton();
        mybutton.pack();
        mybutton.show();
    }
}
```

. . . in der Klasse

```
public class MyButton extends JFrame implements ActionListener{

    JButton button;

    public MyButton() {
        Container cp = getContentPane();
        button  = new JButton("Button 1");
        button.addActionListener(new MyActionLister());
        cp.add(button);
    }

    public void actionPerformed(ActionEvent e) {
        System.out.println("Der Button wurde gedrückt!");
    }

    public static void main(String [] args) {
        MyButton mybutton = new MyButton();
        mybutton.pack();
        mybutton.show();
    }
}
```

Events und Listener Typen

ActionEvent, ActionListener, addActionListener(ActionListener), removeActionListener()	AbstractButton und alle Abkömmlinge (JButton, JRadioButton, JCheckboxButton, JMenuItem, . . .
ItemEvent, ItemListener, addItemListener(ItemListener), removeItemListener()	JCheckBox, JComboBox, JList, . . .
MouseEvent, MouseListener, addMouseListener(MouseListener), removeMouseListener()	Components und Abkömmlinge
TextEvent, TextListener, addTextListener(TextListener), removeTextListener()	alle Abkömmlinge von JTextComponent (JTextArea, JTextField . . .)
WindowEvent, WindowListener, addWindowListener(WindowListener), removeWindowListener()	Window und Abkömmlinge (JFrame, JDialog, JFileDialog)

+ AdjustmentListener, ComponentListener, ContainerListener, KeyListener, MousMotionListener, . . .

Ausblick

- Text Widgets : JPasswordField, JFormatedTextField, ...
- Button Widgets : JRadioButtonMenuItem, JCheckboxMenuItem, JSpinner ...
- JDialogs : JFileChooser, JColorChooser, JDialog, ...
- JPanes : JTabbedPane, JSplitPane, JDesktopPane, JEditorPane, JTextPane ...
- JTooltip, JSlider, JProgressBar, ...

Individuelles Zeichnen

- individuelles Zeichnen - einfach ?
- zwei Ansätze : **Graphics** und **Graphics2d**

`getGraphics()`

oder

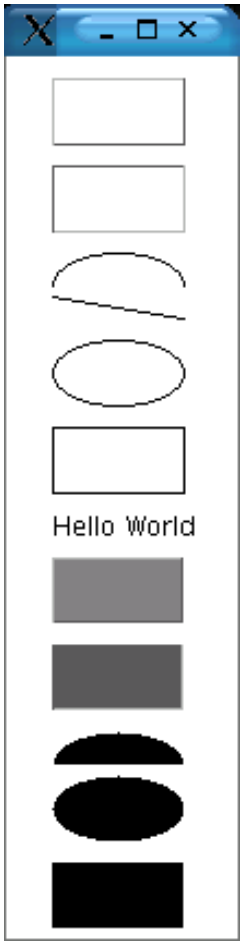
`paintComponent` und `repaint()`

Überschreiben von `paintComponent`

```
public class ModifiedPanel extends JPanel {

    public ModifiedPanel(){
      setBackground(Color.WHITE);
    }

    public void paintComponent(Graphics g) {
       super.paintComponent(g);
       ...
    }

    public Dimension getMinimumSize(){
       return new Dimension(100,410);
    }

    public Dimension getPreferredSize(){
       return getMinimumSize();
    }
}
```

einige Funktionen von Graphics

```
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  g.setColor(Color.GRAY);
  g.draw3DRect(20,10,60,30,true);
  g.draw3DRect(20,50,60,30,false);
  g.setColor(Color.BLACK);
  g.drawArc(20,90,60,30,0,180);
  g.drawLine(20,110,80,120);
  g.drawOval(20,130,60,30);
  g.drawRect(20,170,60,30);
  g.drawString("Hello World",20,220);
  g.setColor(Color.GRAY);
  g.fill3DRect(20,230,60,30,true);
  g.fill3DRect(20,270,60,30,false);
  g.fillArc(20,310,60,30,0,180);
  g.fillOval(20,330,60,30);
  g.fillRect(20,370,60,30);
}
```



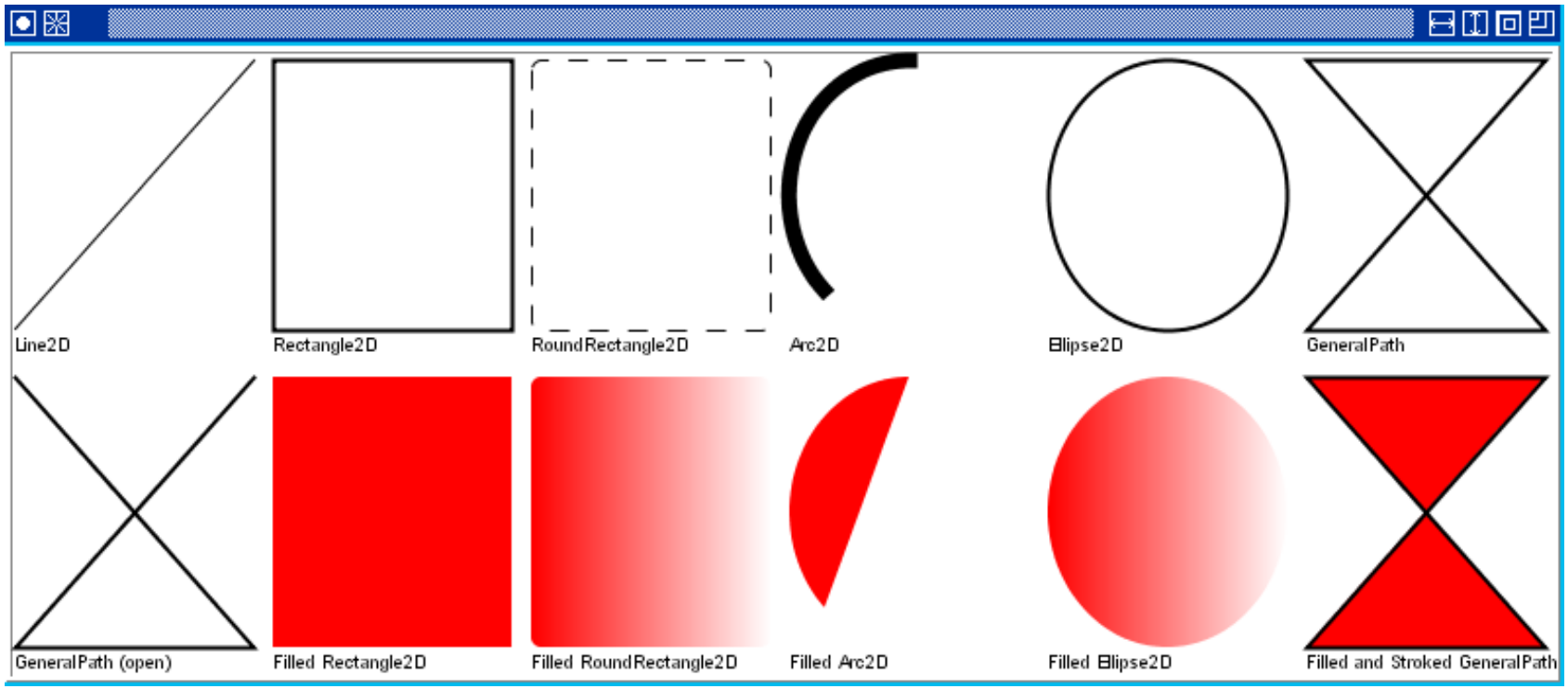
Graphics2D `extends` **Graphics**

- seit Java 1.2
- einheitliches Rendering Modell für verschiedene Ausgabegeräte
- grosse Auswahl von geometrischen „Primitiven"
- Mechanismen um Kollision (Überlappung) zu verarbeiten
- erweitertes Farbmodell
- (HW-Beschleunigung)

Graphics2D

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

...

   public void paintComponent(Graphics g) {
       super.paintComponent(g);
       Graphics2D g2 = (Graphics2D) g;
       g2.setPaint(Color.lightgray);
       g2.draw3DRect(0, 0, d.width - 1, d.height - 1, true);
       g2.draw3DRect(3, 3, d.width - 7, d.height - 7, false);
       ...
    }

...
```

einige Funktionen von Graphics2



vgl. MyGraphics2.java

Literatur

- Folien und Beispiele:
  http://www.techfak.uni-bielefeld.de/ jkrueger/education
- Java Dokumentation (Package javax.swing) :
  file:///vol/jdk/docs/api/javax/swing/package-summary.html
  http://java.sun.com/j2se/1.4.2/docs/api/
- The Swing Tutorial :
  file:///vol/java/share/doc/tutorial/uiswing/index.html
  http://java.sun.com/docs/books/tutorial/uiswing/index.html
- Thinking in Java, 3^{rd} Edition :
  http://www.mindview.net/Books/TIJ/

ENDE