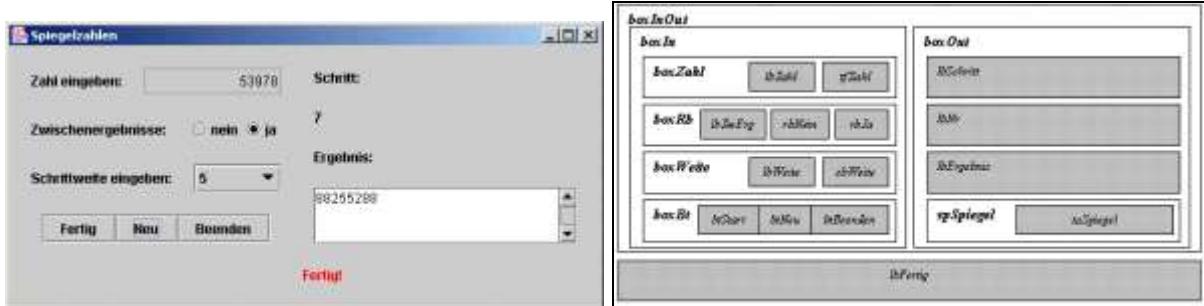


Inhalt

12 Grafische Benutzerschnittstellen.....	12-2
12.1 <i>Komponenten</i>	12-2
12.1.1 AWT und Swing.....	12-2
12.1.2 Beispiel „Hallo Welt!“	12-3
12.2 <i>Ereignisverarbeitung</i>	12-6
12.3 <i>Grafik</i>	12-10
12.3.1 Die Methoden <code>paint</code> und <code>repaint</code>	12-10
12.3.2 Die abstrakte Klasse <code>java.awt.Graphics</code>	12-10
12.3.3 Beispiel „Simple Grafik“	12-11

12 Grafische Benutzerschnittstellen

Grafische Benutzerschnittstellen (**GUI** *Graphical User Interface*) sollen die Bedienung eines Programms erleichtern. Der Aufbau erfolgt in **Containern** nach einem *hierarchischen Baukastensystem*. Aus einer Menge vorgegebener **Komponenten** werden die Oberflächen zusammengesetzt. Von diesen Komponenten können einige wieder als Container verwendet werden, d.h. diese können selbst Bausteine aufnehmen.



<i>box</i>	Box	Container
<i>lb</i>	Label	Text
<i>bt</i>	Button	Knopf
<i>rb</i>	RadioButton	Auswahlknöpfe
<i>cb</i>	ComboBox	Listen, Kombinationsfelder
<i>sp</i>	ScrollPane	Container mit Rollbalken
<i>tf</i>	TextField	Textzeile mit Ein- und Ausgabe
<i>ta</i>	TextArea	Text mit Ein- und Ausgabe

12.1 Komponenten

Die von Java bereitgestellten Klassen lassen sich grob in vier Gruppen unterteilen:

1. **Grundkomponenten**, diese beinhalten einfache *Oberflächenelemente*, wie **Label**, **Button**, **RadioButton**, **ComboBox**, **TextField**, **TextArea** u.a.m.
2. **Container**, diese können Komponenten aufnehmen. Dazu gehören Fenster **Frame**, einfache Darstellungsbereiche **Panel**, **Box**, **ScrollPane** u.a.m.
3. **Layout-Manager**, **Farben** und **Fonts** dienen der Anordnung und Gestaltung der einzelnen Komponenten, also dem *Design* im Container.
4. **Ereignisse** und **Ereigniswächter** sind für Aufnahme und Verarbeiten von Benutzereingaben, also zur *Interaktion* der Komponenten mit den Anwendern, notwendig.

12.1.1 AWT und Swing

Java bietet zwei Pakete **java.awt.*** und **javax.swing.***, mit deren Hilfe man grafische Benutzeroberflächen entwickeln kann.

Die Klasse **java.awt.Component** ist die Oberklasse aller Grafikkomponenten. Von dieser werden zahlreiche spezifische Komponentenklassen abgeleitet.

AWT (*Abstract Windowing Toolkit*) gehört seit JDK Version 1.0 zum Lieferumfang, ist in den aktuellen Java Versionen enthalten und umfasst *alle* oben genannten Gruppen von Klassen.

Swing ist eine Weiterentwicklung des AWT, ist *einfacher* zu programmieren und erzeugt *komfortablere* Komponenten als AWT. Deren Aussehen ist *plattformunabhängig* und während der Laufzeit *veränderbar*. Swing ersetzt AWT-Komponenten der *ersten beiden Gruppen*. Von AWT bereits bekannte Komponenten wurden von Swing übernommen. Sie behalten ihren alten Namen vor dem jedoch der Buchstabe „J“ steht.

Die *letzten beiden Gruppen* von Klassen werden auch weiterhin von **AWT** bedient.

In den Containern verwalten **Layout-Manager** die *Platzierung* der Komponenten. Es existieren verschiedene Layout-Manager.

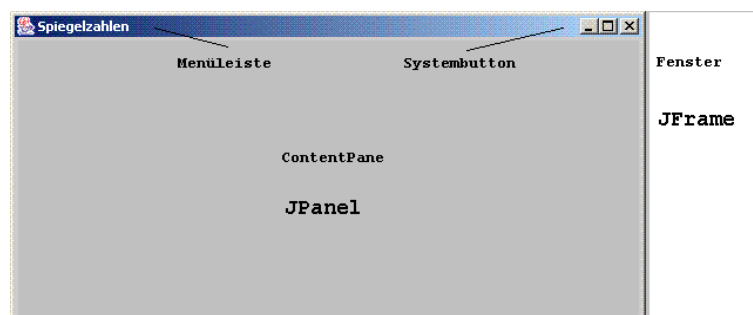
- Bei vielen der Vorlesungsbeispielen beschränken wir uns auf die Verwendung des Standardmanagers **FlowLayout**. Er ist voreingestellt und ordnet die Komponenten im Container *zentriert* an.
- Üblich ist es auch, *Boxen* als Container zu benutzen. Diese verwenden standardmäßig das **BoxLayout** und sind ebenfalls sehr leicht zu handhaben (Beispiel „Spiegelzahlen“, Beispiel „Bruchaddierer“ s. Übung).
- Wünscht man eine *matrixartige Anordnung*, so bietet sich das **GridLayout** an (Beispiel im Paket **Tools.Game**).

Die Verwendung anderer Manager soll hier nicht weiter erläutert werden.

12.1.2 Beispiel „Hallo Welt!“

In einem ersten Beispiel soll die typische Grundstruktur eines einfachen Fensters aus Komponenten des Pakets **javax.swing.*** aufgebaut werden:

Die Klasse **JFrame** stellt alle Funktionalitäten für ein *Fenster* zur Verfügung. Objekte dieser Klasse besitzen bereits das *Aussehen* und alle *Merkmale* eines Fensters auf der jeweiligen Betriebssystemplattform. Das Fenster besteht aus einem Rand (*Border*) mit Titelleiste und typischem Systemmenü und einem *Container* zur Darstellung seines Inhalts (*Content Pane*).



Da die nächste Anwendung in einem Fenster ablaufen soll, leiten wir diese von der Klasse **JFrame** ab.

```
public class HalloWeltGUI1 extends JFrame{ ... }
```

Die Klasse **JFrame** besitzt einen *Konstruktor*, welcher die *Titelleiste* mit Text versieht. Dieser wird zu genau dem Zweck im Konstruktor der Klasse **HalloWeltGUI1** aktiviert.

```
super( titel);
```

In einem *Container*, einem Objekt der Klasse **JPanel**, wird die grafische Benutzeroberfläche zusammengebaut. Sie soll mit einer Hintergrundfarbe versehen werden und einen Schriftzug beinhalten. Dazu wird ein *Label* der Klasse **JLabel** mit Text versehen und in den Container aufgenommen.

```
JPanel contentPane = new JPanel(); // Panel
contentPane.setBackground( farbe);

JLabel lbHallo = new JLabel( "Hallo Welt!"); // Label
contentPane.add( lbHallo);
```

Der Darstellungsbereich *Content Pane* des Fensters wird durch den neu erzeugten Container ersetzt.

```
setContentPane( contentPane);
```

Die *Größe* des Fensters wird dem *Darstellungsinhalt* angepasst. Die Funktionalität des Fensterschließens wird dem *X-Button* der Menüleiste übergeben und das Fenster sichtbar gemacht.

```
pack();
setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
setVisible( true);
```

Das komplette Programm hat folgendes Aussehen, nicht erläuterte Programmzeilen sind selbsterklärend.

HalloWeltGUI1.java

```
// HalloWeltGUI1.java MM 2014

import java.awt.*; // Color
import javax.swing.*; //JFrame JPanel JLabel

/**
 * Erzeugen eines Fensters mit Text.
 */
public class HalloWeltGUI1 extends JFrame
{
/* ----- */
/**
```

```

* Konstruktor, baut das Fenster auf.
* @param titel Titelleistentext
* @param farbe Fensterhintergrundfarbe
*/
public HalloWeltGUI1( String titel, Color farbe)
{
// Titelleiste
    super( titel);

// Darstellungsbereich
    JPanel contentPane = new JPanel();           // Panel
    contentPane.setBackground( farbe);

    JLabel lbHallo = new JLabel( "Hallo Welt!"); // Label
    contentPane.add( lbHallo);

// Fenster
                                // Fensterinhalt uebernehmen
    setContentPane( contentPane);
    pack();                       // Anpassen der Fenstergroesse
                                // Beenden x-Button
    setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
    setVisible( true);           // Sichtbarmachen
}

/* ----- */
/**
 * Aufruf des Programms.
 */
public static void main( String args[])
{
    HalloWeltGUI1 hallo1 =
        new HalloWeltGUI1( "Hallo Welt", Color.green);

    HalloWeltGUI1 hallo2 =
        new HalloWeltGUI1( "Hallo Welt", Color.yellow);

    HalloWeltGUI1 hallo3 =
        new HalloWeltGUI1( "Hallo Welt", Color.red);
}
}

```



12.2 Ereignisverarbeitung

Im letzten Beispiel wurde eine einfache grafische Oberfläche aufgebaut. Diese soll nun so erweitert werden, dass das Programm auf Nutzereingaben (**Ereignisse**) reagiert. Die **Ereignisüberwachung** wird intern durch eine *Endlosschleife* realisiert, welche auf Ereignisse wartet, eingetretene Ereignisse registriert und, durch Benachrichtigen des betreffenden Programms, weiterverarbeitet. Verschiedene Klassen der Ereignisse (*Events*) und ihre Empfänger (*Listener*) des Pakets `java.awt.event.*` stellen die notwendigen Funktionalitäten bereit.

Ein *Ereignis* kann durch *Mausklick auf einen Button* ausgelöst werden. Dabei wird ein Objekt der Ereignisklasse **ActionEvent** erzeugt.

Zu jeder Ereignisklasse gibt es einen Listener, ein Interface, dessen Methoden eingetretene Ereignisse der Klasse weiterverarbeiten.

Zu **ActionEvent** gehört **ActionListener**, welches nur eine zu implementierende Methode, die Methode `actionPerformed`, enthält. Diese Methode wird aufgerufen, wenn ein Button betätigt wurde.

Wir erweitern unser Programm, indem wir zwei Button einrichten, einen zum Ändern der Hintergrundfarbe unseres Fensters und einen zum Beenden des Programms.

Die Containerklasse selbst wird als Listenerklasse realisiert, indem die von **JFrame** abgeleitete Klasse **HalloWeltGUI2** das Interface **ActionListener** implementiert.

```
public class HalloWeltGUI2
    extends JFrame
        implements ActionListener{ ... }
```

In unserem Beispiel werden zwei Aktionen betrachtet und durch Klassenkonstanten charakterisiert:

```
                // Aendern der Hintergrundfarbe
public static final String ACTION_SET = "Set";
                // Beenden des Programms
public static final String ACTION_QUIT = "Quit";
```

Als Instanzvariable ist der Darstellungsbereich bereitzustellen, da später bei der Änderung seiner Hintergrundfarbe auf ihn zugegriffen wird:

```
private JPanel contentPane;
```

Jeder Button wird eingerichtet, im Darstellungsbereich aufgenommen und dessen Ereignisüberwachung installiert.

```
                // Button Set mit Listener
JButton btSet = new JButton( ACTION_SET);
contentPane.add( btSet);
btSet.addActionListener( this);
                // Button Quit mit Listener
JButton btQuit = new JButton( ACTION_QUIT);
contentPane.add( btQuit);
btQuit.addActionListener( this);
```

Schließlich wird die Methode `actionPerformed` des Interfaces `ActionListener` implementiert, um die Weiterverarbeitung empfangener Ereignisse `ActionEvent` zu veranlassen. Dabei wird der aktivierte Button durch seine Aufschrift identifiziert.

```

public void actionPerformed( ActionEvent ae)
{
    String command = ae.getActionCommand();

    // Farbwechsel durch Button Set
    if( command.equals( ACTION_SET))
    {
        float dummy1 = ( float)Math.random();
        float dummy2 = ( float)Math.random();
        float dummy3 = ( float)Math.random();
        Color farbe = new Color( dummy1, dummy2, dummy3);

        contentPane.setBackground( farbe);
    }

    // Programmabbruch durch Button Quit
    if( command.equals( ACTION_QUIT))
    {
        System.exit( 0);
    }
}

```

Das vollständige Programm hat folgendes Aussehen:

HalloWeltGUI2.java

```

// HalloWeltGUI2.java
// MM 2014

import java.awt.*; // Color
import javax.swing.*; // JFrame JPanel JLabel JButton
import java.awt.event.*; // Listener

/**
 * Erzeugen eines Fensters mit Text und Button.
 */
public class HalloWeltGUI2
    extends JFrame
    implements ActionListener
{
    /* ----- */
    // View

    /**
     * Konstante, Aendern der Hintergrundfarbe.
     */
    public static final String ACTION_SET = "Set";

    /**
     * Konstante, Beenden des Programms.
     */
}

```

```
*/
public static final String ACTION_QUIT = "Quit";

/**
 * Content Pane, Darstellungsbereich des Fensters.
 */
private JPanel contentPane;

/**
 * Konstruktor, baut das Fenster auf.
 * @param titel Titelleistentext
 * @param farbe Fensterhintergrundfarbe
 */
public HalloWeltGUI2( String titel, Color farbe)
{
// Titelleiste
    super( titel);

// Darstellungsbereich
    contentPane = new JPanel();
    contentPane.setBackground( farbe);
                                // Button Set mit Listener
    JButton btSet = new JButton( ACTION_SET);
    contentPane.add( btSet);
    btSet.addActionListener( this);
                                // Label Hallo
    JLabel lbHallo = new JLabel( "Hallo Welt!");
    contentPane.add( lbHallo);
                                // Button Quit mit Listener
    JButton btQuit = new JButton( ACTION_QUIT);
    contentPane.add( btQuit);
    btQuit.addActionListener( this);

// Fenster
                                // Fensterinhalt uebernehmen
    setContentPane( contentPane);
    pack();                                // Anpassen der Fenstergroesse
                                // Beenden x-Button
    setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
    setVisible( true);                                // Sichtbarmachen
}

/* ----- */
                                // Controller

/**
 * Ereignisverarbeitung, ActionListener,
 * Betaetigen eines Button.
 */
public void actionPerformed((ActionEvent ae)
{
    String command = ae.getActionCommand();
}
```



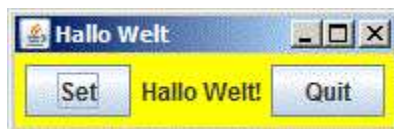
```
// Farbwechsel durch Button 'Set'
    if( command.equals( ACTION_SET)
    {
        float dummy1 = ( float)Math.random();
        float dummy2 = ( float)Math.random();
        float dummy3 = ( float)Math.random();
        Color farbe = new Color( dummy1, dummy2, dummy3);

        contentPane.setBackground( farbe);
    }

// Programmabbruch durch Button 'Quit'
    if( command.equals( ACTION_QUIT)
    {
        System.exit( 0);           // Beendet das Programm
    }
}

/* ----- */

/**
 * Aufruf des Programms.
 */
public static void main(String args[])
{
    HalloWeltGUI2 hallo =
        new HalloWeltGUI2( "Hallo Welt", Color.yellow);
}
}
```



12.3 Grafik

12.3.1 Die Methoden `paint` und `repaint`

In diesem Kapitel wird die Darstellung von Grafiken in Applikationen behandelt. In Applets haben wir dazu bereits die Methoden `paint` und `repaint` verwendet (Beispiele siehe Kapitel 3: `HalloWeltApplet.java`, `AmpelApplet.java`).

Ein **Repaint-Manager** ist für die Aktualisierung der Benutzeroberfläche beim *erstmaligen Aufbau* und bei *Änderungen des Erscheinungsbilds* zuständig. In diesen Fällen wird die Methode `repaint` aufgerufen, welche die Aktualisierung der Komponenten eines Fensters durch eine Methode `paint` veranlasst. Beide Methoden sind in der Klasse `java.awt.Component`, der Oberklasse aller Grafikkomponenten, definiert. Jede abgeleitete Komponente überschreibt `paint` entsprechend deren grafischen Darstellung. Für eigene Komponenten muss `paint` implementiert werden.

`paint` bekommt ein Objekt der Klasse `Graphics` übergeben. Man kann sich dieses als ein Zeichenblatt vorstellen, auf welchem eine Grafik entwickelt wird. Die Klasse bietet Methoden zur Erzeugung von Text-, Linien- und Füllelementen, verwaltet Zeichenfarben, Fonts zur Ausgabe von Schrift u. a. m.

`Container`, zum Beispiel ein `JPanel`-Objekt, können Grafiken aufnehmen. Durch Überschreiben der Methode `paint` ist eine individuelle Gestaltung einer Grafik möglich.

12.3.2 Die abstrakte Klasse `java.awt.Graphics`

In der Klasse `Graphics` werden zahlreiche Methoden zur Verfügung gestellt, um im Grafikkordinatensystem eines Containers zu zeichnen. Das Grafikkordinatensystem ist ein Pixel-Koordinatensystem mit dem Ursprung (0, 0) in der linken oberen Ecke des Containers.

Methoden (Auswahl)

Name	Parameteranzahl	Parameter-typ	Ergebnis-typ	Beschreibung
<code>drawLine</code>	4	<code>int, int, int, int</code>		zeichnet Linie vom Pixel nach Pixel
<code>drawRect</code>	4	<code>int, int, int, int</code>		zeichnet Rechteck ausgehend vom Pixel in der linken oberen Ecke mit gegebener Breite und Höhe
<code>drawOval</code>	4	<code>int, int, int, int</code>		zeichnet Ellipse in einem Rechteck ausgehend vom Pixel in der linken oberen Ecke mit gegebener Breite und Höhe
<code>drawString</code>	3	<code>String, int, int</code>		zeichnet String beginnend am gegebenen Pixel
<code>fillPolygon</code>	3	<code>int[], int[], int</code>		zeichnet gefülltes geschlossenes Polygon durch die angegebenen x- und y-Pixelkoordinaten und der angegebenen Eckenzahl
<code>setColor</code>	1	<code>Color</code>		setzt Zeichenfarbe
<code>setFont</code>	1	<code>Font</code>		setzt Schreibfont

12.3.3 Beispiel „Simple Grafik“

Eine Klasse **SimplePanel** wird von der Klasse **JPanel** abgeleitet. Bei der Gestaltung der Grafik werden die obigen Methoden verwendet. Ergänzend wird ein Bild in die Grafik aufgenommen und ein Rechteck durch affine Transformation gedreht dargestellt. Auf diese Details soll hier aber nicht mehr eingegangen werden.

SimplePanel.java

```
// SimplePanel.java                                MM 2014
//                                                  AZ 2004

import java.awt.*;                                // Color Graphics
import java.awt.geom.*;                          // AffineTransform
import javax.swing.*;                             // JPanel

/**
 * Darstellung einer simplen Grafik
 */
public class SimplePanel
    extends JPanel
{
    /**
     * Konstruktor, legt bevorzugte Groesse des
     * Darstellungsbereiches fest.
     * @param dimX Breite
     * @param dimY Hoehe
     */
    SimplePanel( int dimX, int dimY)
    {
        setPreferredSize( new Dimension( dimX, dimY));
    }

    /**
     * Zeichnen von Grafikobjekten.
     */
    public void paint( Graphics g)
    {
        // Leeren des Darstellungsbereich
        g.setColor( getBackground());
        g.fillRect( 0, 0, getWidth(), getHeight());

        // String-Ausgabe
        g.setColor( Color.red);
        g.setFont( new Font( "serif", Font.ITALIC, 30));
        g.drawString( "Simple Graphic", 20, 80);

        // Rechteck
        g.setColor( Color.green);
        g.drawRect( 20, 90, 170, 150);

        // Dreieck (gefüllt)
        int[] x= { 100, 150, 200};
```

```

    int[] y= { 270, 180, 250};
    g.setColor( Color.red);
    g.fillPolygon( x, y, 3);

// Kreis
    g.setColor( Color.blue);
    g.drawOval( 200, 170, 90, 90);

// Linie
    g.setColor( Color.magenta);
    g.drawLine( 10, 160, 290, 140);

// Bild
    MediaTracker tracker = new MediaTracker( this);
    Image bild = Toolkit.getDefaultToolkit().getImage
        ( "unilogo.gif");

    tracker.addImage( bild, 0);
    try
    {
        tracker.waitForAll();
    }
    catch( InterruptedException e)
    {
        throw new Error( "Konnte Bild nicht laden!");
    }
    g.drawImage( bild, 100, 0, null);

// Rechteck (gedreht) mit 2D Grafik
    g.setColor( Color.blue);
    Graphics2D g2d = (Graphics2D)g;

                                                                    // Drehung
    AffineTransform at = g2d.getTransform();
    g2d.setTransform
    ( AffineTransform.getRotateInstance
      ( Math.PI/3, 150, 150));
    g2d.draw( new Rectangle( 20, 90, 170, 150));
                                                                    // Drehung aufheben
    g2d.setTransform( at);
}
}

```

Eine Klasse **SimpleGrafik** bereitet ein Fenster vor, welches die Grafik aufnimmt. Diese Klasse ist im Aufbau dem zuerst besprochenen „HalloWelt“ - Beispiel sehr ähnlich und muss deshalb nicht noch einmal erläutert werden.

SimpleGrafik.java

```

// SimpleGrafik.java
//
                                                                    MM 2014
                                                                    AZ 2004

import java.awt.*;
                                                                    // Color

```

```

import javax.swing.*;                                // JFrame JPanel

/**
 * Simple 2D Grafik
 */
public class SimpleGrafik
    extends JFrame
{
    /**
     * Konstruktor, baut Fenster auf.
     * @param titel Text in Titelleiste
     * @param farbe Fensterhintergrundfarbe
     */
    public SimpleGrafik( String titel, Color farbe)
    {
        // Titelleiste
        super( titel);

        // Darstellungsbereich
        JPanel contentPane = new SimplePanel( 300, 300);
        contentPane.setBackground( farbe);

        // Fenster
        setContentPane( contentPane);
        pack();
        setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
        setVisible( true);
    }

    /* ----- */

    /**
     * Aufruf des Programms.
     */
    public static void main(String[] args)
    {
        SimpleGrafik zeichnen =
            new SimpleGrafik( "Grafik", Color.yellow);
    }
}

```

