Programmieren II

Dr. Klaus Höppner

Hochschule Darmstadt – Sommersemester 2010

Einführung: Grafische Benutzeroberflächen

Kleine Anwendung

Einführung: GUI

Fast alle Programme besitzen mittlerweile eine grafische Benutzeroberfläche (engl: *Graphical User Interface*, GUI).

Ziele sind:

- einfache Bedienung mit der Maus
- einheitliches *Look & Feel* der Anwendungen
- Stichwort: *Usability*

Typische Elemente von GUIs sind:

- Menüs
- Symbolleisten (*tool bar*)
- Schaltfächen
- Auswahlfelder (*button*, *radio button*, *check box*, *pull-down menu*)
- Texteingabefelder

Java im Vergleich zu anderen Sprachen

Java enthält eine direkte Unterstützung für grafische Oberflächen, die Teil der Laufzeitumgebung ist.

Daher liegt Java insbesondere hier gegenüber anderen Sprachen im Vorteil. Java ist heute *der* Weg, grafische Anwendung über Systemgrenzen hinweg portabel zu schreiben.

Für die meisten anderen Programmiersprachen gibt es zwar GUI-Zusatzpakete, diese stehen aber oft zueinander in Konkurrenz und sind nur auf wenigen Systemen verfügbar, z. B.

> Microsoft Foundation Classes (Visual Basic, Visual C++), Windows Forms (Visual C#/Basic/C++ .NET), Qt (C++, Python auf Unix und Windows), GTK (C++, Python auf Unix), Tk

AWT

Bereits beim ersten Java-Release 1995 war ein *Toolkit* zum Erstellen grafischer Oberflächen enthalten:

AWT – Abstract Window Toolkit

Features:

- *heavyweight*: AWT definiert eine Abstraktionsebene über den Betriebssystembibliotheken für Oberflächen.

$\Rightarrow$ Auslagerung der Arbeit an das Betriebssystem

$\Rightarrow$ Betriebssystem-typisches Look & Feel

$\Rightarrow$ aber uneinheitliches Aussehen der selben Anwendung auf verschiedenen Plattformen

$\Rightarrow$ Beschränkung auf kleinsten Nenner

Einfache AWT-Anwendung

```
import java.awt.*;

public class AwtApp {
    public static void main(String[] args) {
        Frame myFrame = new Frame("Erstes Fenster");
        myFrame.setSize(400, 200);
        myFrame.add(new Label("Hallo"));
        myFrame.setVisible(true);
    }
}
```

Swing

Swing ist Teil der *Java Foundation Classes*, die ursprünglich von Netscape als *Internet Foundation Classes* entwickelt wurden und dann in Kooperation von Sun und Netscape in Java 1.2 integriert wurden.

Wesentliche Eigenschaften:

- *lightweight*, die Widget-Klassen sind in Java selber implementiert und setzen nur auf wenig Ressourcen des unterliegenden Betriebssystems auf
- einheitliches Aussehen der Anwendungen auf verschiedenen Plattformen
- entspricht nicht unbedingt dem Look & Feel des Betriebssystems
- erweiterbares Komponentenmodell

Einfache Swing-Anwendung

```
import javax.swing.*;

public class SwingApp extends JFrame {
    public SwingApp(String title) {
        super(title);
        setSize(400, 200);
        add(new JLabel("Hallo"));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    public static void main(String[] args) {
        SwingApp myApp = new SwingApp("Erstes Fenster");
    }
}
```

Grundsätzliche Entwicklungsschritte

Der Entwurf einer grafischen Anwendung zerfällt in zwei Schritte:

1. Entwurf der grafischen Komponenten und deren Anordnung
2. Implementierung der Eventbehandlung (z. B. Eingabe von Text, Klicken auf Schaltflächen etc.)

Toplevel-Elemente

Swing kennt insgesamt drei *Toplevel*-Elemente:

JFrame ein Fenster mit Rahmen und Titel
Selbst geschriebene Anwendungen mit grafischer Oberfläche befinden sich meist innerhalb eines `JFrame`. Daher erbt die Hauptklasse einer Anwendung oft von `JFrame`.

JDialog für Meldungsfenster

JApplet für Java-Applets, die innerhalb eines (Java-fähigen) Webbrowsers laufen.

Innerhalb eines Toplevel-Fensters können beliebige Komponenten (außer anderen Toplevel-Fenstern) angeordnet werden.

Der Layout-Manager

Der *Layout-Manager* ist für die Platzierung der Komponenten innerhalb eines Containers (z. B. ein Toplevel-Element) verantwortlich.

Swing kennt im Wesentlichen:

FlowLayout ordnet Komponenten von links nach rechts an

BorderLayout Anordnung nach „Himmelsrichtungen“

GridLayout Anordnung in Zeilen und Spalten

GridBagLayout komplexeres GridLayout

Beispiel

Als einfaches Beispiel soll folgende Anwendung realisiert werden:

- Titelzeile oben,
- In der Mitte eine 2×2-Tabelle mit Label und einem Texteingabefeld in der ersten Zeile und Label und Ergebnisfeld in der zweiten Zeile,
- Anwendungsschaltfläche unten

Beim Klicken auf „Anwenden“ soll im eingegebenen Text die Anzahl des Buchstaben „e“ gezählt und im Ergebnisfeld ausgegeben werden.

Grundgerüst

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

public class MyApp extends JFrame {
    private JTextField eingabe;
    private JLabel result;

    public MyApp(String title) {
        super(title);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(400, 200);
        this.setLayout(new BorderLayout());
```

Grundgerüst (Forts.)

```
        JLabel top = new JLabel("Bitte einen Text eingeben");
        top.setHorizontalTextPosition(SwingConstants.CENTER);
        this.add(top, BorderLayout.PAGE_START);

        JButton applyButton = new JButton("Anwenden");
        this.add(applyButton, BorderLayout.PAGE_END);
    }

    public static void main(String[] args) {
        MyApp app = new MyApp("e-Anwendung");
        app.setVisible(true);
    }
}
```

Aktueller Stand

Durch dieses Grundgerüst existieren das Hauptfenster mit zentrierter Titelzeile oben und einer Schaltfläche „Anwenden“ unten.

Die Tabelle im Format 2×2 in der Mitte soll im GridLayout gesetzt werden. Hierfür wird im Zentrum ein Container, nämlich ein `JPanel` eingefügt, der das Grid enthält.

`JPanel` ist weitgehend identisch zu `JFrame`, aber natürlich ohne Rahmen und Fenstertitel.

Neuer Konstruktor von MyApp

```
public MyApp(String title) {
    super(title);

    ...

    JPanel mid = new JPanel();
    mid.setLayout(new GridLayout(2,2,10,10));
    this.add(mid,BorderLayout.CENTER);
    eingabe = new JTextField();
    result = new JLabel();
    mid.add(new JLabel("Eingabe"));
    mid.add(eingabe);
    mid.add(new JLabel("Antwort"));
    mid.add(result);
}
```

GridLayout

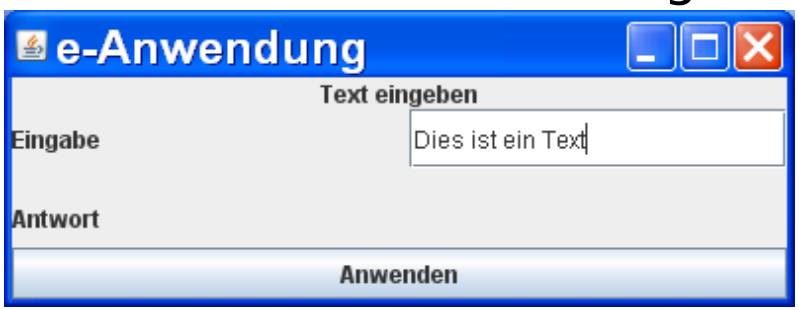
Die Layout-Klasse `GridLayout` kennt drei Konstruktoren:

`GridLayout()` Standardkonstruktor,

`GridLayout(int rows, int cols)` Grid mit Angabe von Zahl der Zeilen und Spalten,

`GridLayout(int rows, int cols, int hgap, int vgap)` wie zuvor, aber mit Angabe von horizontalem (hgap) und vertikalem (vgap) Abstand zwischen den Zellen.

Nun sieht die Anwendung etwa so aus:



Aktionen ausführen

Noch passiert beim Klicken auf „Anwenden“ nichts. Hierfür muss bei der Aktion „Klicken auf die Schaltfläche“ eine entsprechende Aktion ausgelöst werden.

Für diese Aktionen existiert das Interface `ActionListener`, das die abstrakte Methode `actionPerformed` enthält.

Klassen, die `ActionListener` implementieren, können nun mit der Methode `actionPerformed` von `JButton` in die Liste der auszuführenden Aktionen aufgenommen werden.

Quelltext

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

public class MyApp extends JFrame implements ActionListener {
    private JTextField eingabe;
    private JLabel result;

    public MyApp(String title) {
        ...
        applyButton.addActionListener(this);
    }
    public void actionPerformed(ActionEvent event) {
        int count = 0;
        String text = eingabe.getText();
        for (int i=0; i<text.length(); i++) {
            if (text.charAt(i)=='e') count++;
        }
        result.setText(String.valueOf(count));
    }
}
```

Kompletter Quelltext

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

public class MyApp extends JFrame implements ActionListener {
    private JTextField eingabe;
    private JLabel result;

    public MyApp(String title) {
        super(title);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(400, 150);
        this.setLayout(new BorderLayout());

        JLabel top = new JLabel("Text eingeben",JLabel.CENTER);
        top.setHorizontalTextPosition(SwingConstants.CENTER);
        this.add(top, BorderLayout.PAGE_START);
```

Kompletter Quelltext (Forts.)

```
        JButton applyButton = new JButton("Anwenden");
        this.add(applyButton, BorderLayout.PAGE_END);
        applyButton.addActionListener(this);

        JPanel mid = new JPanel();
        mid.setLayout(new GridLayout(2,2,10,10));
        this.add(mid,BorderLayout.CENTER);
        eingabe = new JTextField();
        result = new JLabel();
        mid.add(new JLabel("Eingabe"));
        mid.add(eingabe);
        mid.add(new JLabel("Antwort"));
        mid.add(result);
    }
```

Kompletter Quelltext (Forts.)

```
    public void actionPerformed(ActionEvent event) {
        int count = 0;
        String text = eingabe.getText();
        for (int i=0; i<text.length(); i++) {
            if (text.charAt(i)=='e') {
                count++;
            }
        }
        result.setText(String.valueOf(count));
    }

    public static void main(String[] args) {
        MyApp app = new MyApp("e-Anwendung");
        app.setVisible(true);
    }
}
```