Vorlesung
Softwareentwicklung II
Graphische Benutzeroberflächen mit Swing

Ulrike Hammerschall
Fakultät für Informatik und Mathematik
Sommersemester 2014

Grafische Benutzeroberflächen

- Häufiger als Graphical User Interface (GUI) bezeichnet.
- Bilden die Dialogschnittstelle zwischen Anwender und System:
  - Der Anwender macht Eingaben über die Schnittstelle.
  - Das System zeigt die Ergebnisse an der Schnittstelle an.
- Was gehört zur Dialogschnittstelle?
  - ein oder mehrere Dialogfenster (Masken).
  - die Ablaufsteuerung an der Dialogschnittstelle (Dialogsteuerung).

Abstract Window Toolkit (AWT)

- Erste API von Sun für den Aufbau von grafischen Benutzeroberflächen.
- Nutzt über Peer-Klassen die grafischen Komponenten der darunterliegenden Plattform.
- Vor- und Nachteile:
  - Look and Feel von AWT-Anwendungen entspricht dem aller Anwendungen auf dieser Plattform.
  - Peer-Klassen auf unterschiedlichen Plattformen können sich unterschiedlich verhalten.
  - Sehr kleiner gemeinsamer Nenner an gemeinsamen Peer-Klassen auf unterschiedlichen Plattformen.
- AWT Komponenten werden wegen dieser Kopplung auch als schwergewichtige Komponenten bezeichnet.

Die Java Foundation Classes

- Lösen 1997 die AWT Komponenten ab. Bestehen aus:
  - Swing Framework: GUI Elemente wie Buttons, Textfelder, Labels, Panels, Fenster, Menu-Leisten.
  - Pluggable Look and Feel Support: Erlaubt das einfache Umstellen des Look and Feel (z.B. Windows, Java, …).
  - Accessability API: Erlaubt assistierten Zugang (Braille Schrift, Lesen der Inhalte, …).
  - Java 2D API: Unterstützung für 2D Graphiken.
  - Internationalization: Unterstützung der Mehrsprachigkeit wie z.B. unterschiedliche Schriftzeichen oder Datumsformate.
  - Drag and Drop. Daten können leicht von einer Anwendung zur anderen übertragen werden.

Das Swing-Framework

- Dient zum Aufbau von beliebig komplexen Dialogen.
- Swing-Komponenten sind leichtgewichtig: verfügen über keinen Peer der unterliegenden Plattform.
- Alle Komponenten werden mit Zeichenoperationen gemalt.
- Vor- und Nachteile:
  - Hohe Flexibilität bei der Darstellung der Komponenten.
  - Plattformabhängige Komponenten stehen nicht unmittelbar zur Verfügung (z.B. Dateipicker).
  - Swing ist im Gegensatz zu AWT nicht Thread-safe. Entwickler muss manuell parallele Zugriffe koordinieren.
- Alle Klassen des Swing-Frameworks liegen im javax.swing Package oder darunter.

Swing-Fenster

- Fenster dienen als Grundlage für Dialoge. Alle weiteren GUI-Elemente werden innerhalb eines Fensters angezeigt.
- Swing kennt drei Fenstertypen, auch als top-level Container bezeichnet. Jeder der top-level Container erweitert eine AWT-Klasse:
  - javax.swing.JFrame erweitert java.awt.Frame.
  - javax.swing.JApplet abgeleitet von java.applet.Applet.
  - javax.swing.JDialog abgeleitet von java.awt.Dialog.

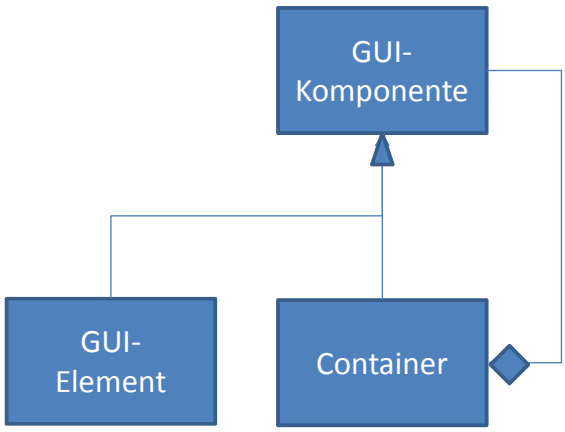
Fensterverhalten

- Standardinitialisierung:

```
JFrame f = new JFrame(„My Window");
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setSize(300,200);
f.setVisible(true);
```

- Die Konstante EXIT_ON_CLOSE stellt sicher, dass Anwendung und jvm mit System.exit() geschlossen werden.
- setVisible(true) stellt sicher, dass das Fenster angezeigt wird. Weitere Varianten: toBack(), toFront().

Dialogaufbau



- Dialoge sind strikt hierarchisch aufgebaut.
- Container sind die Knoten im Baum. Können weitere Komponenten (Container oder GUI-Elemente) enthalten.
  - Beispiele: JPanel, JTabbedPane, JScrollPane, ….
- GUI-Elemente sind die Blätter im Baum.
  - JButton, JTextField, JTextArea, JRadioButton, ….

Dialogaufbau

- Der top-level Container erhält eine Content-Pane als Wurzel des Baums.
- Jede weitere Komponente wird im Baum unterhalb der Content-Pane eingefügt.
- Vorgehen zum Aufbau einer GUI:
  - Top-level Container mit Content-Pane erstellen.
  - GUI-Komponenten erstellen und in den Baum einfügen.
  - Pro GUI-Komponente die Eigenschaften setzen.

Bearbeitung von Ereignissen

- Graphische Benutzeroberflächen reagieren auf Ereignisse von außen.
- Das Ereignismodell von Java unterscheidet Ereignisquellen (Event Source) und Ereignissenken (Listener).
- Ablauf:
  - Ein Ereignis (Button-Click, Mouse-Event, etc.) wird von der Ereignisquelle registriert.
  - Daraus wird ein entsprechendes Ereignisobjekt erzeugt und
  - an den entsprechenden (registrierten) Ereignisempfänger zur Verarbeitung weitergeleitet.

Ereignisquellen und -senken

- Alle Swing-Komponenten können als Ereignisquellen agieren.
- Damit Sie auf ein Event entsprechend reagieren können, muss für das Event ein entsprechender Listener registriert sein.
- Alle Swing-Listener implementieren die Listener-Schnittstelle.
- Beispiel:
  - Ereignisquelle (Event Source): JButton
  - Ereignis, das ausgelöst wird (Event): ActionEvent
  - Wann: Bei Mouse-click auf die Button-Oberfläche
  - Ereignissenke (Listener): ActionListener

Beispiel ActionListener

- ActionEvent und ActionListener: Ereignisbearbeitung auf Schaltflächen.
- Implementierung des Interfaces ActionListener:

```
public void actionPerformed(ActionEvent e) {..}
```

- Registrieren des Listeners an GUI-Element:

```
addActionListener(ActionListener al);
```

Programmiermodelle für Listener

- Anonyme, innere oder lokale Klasse: geeignet für Listener mit wenig Code.
- Separate Klasse
  - pro Listener eine eigene Klasse oder
  - eine Klasse für alle Listener.
- JFrame implementiert selbst die Schnittstelle des EventListeners.

Layout-Manager

- Verantwortlich für die Positionierung von GUI-Komponenten (Container und GUI-Elemente) auf Containern.
- Passen abhängig vom Manager-Typ die Positionierung der Element bei Änderungen der Fenstergröße an.
- Layout-Manager werden den Container-Klassen (mit Ausnahme der top-level Container) im Swing-Framework zugeordnet.
- Kein Layout-Manager bedeutet: absolute Positionierung der Elemente.

Layout-Manager

Layout Manager	Anordnung
FlowLayout	Ordnet Komponenten von links nach rechts an.
BoxLayout	Ordnet Komponenten horizontal oder vertikal an.
GridLayout	Setzt Komponenten in ein Raster. Jedes Element besitzt die gleichen Maße.
BorderLayout	Setzt Komponenten in den vier Himmelsrichtungen, sowie im Zentrum
GridBagLayout	Sehr flexibler aber auch komplexer Manager als Erweiterung des GridLayouts.
CardLayout	Verwaltet Komponenten wie auf einem Stapel, so dass nur eine Komponente sichtbar ist.
GroupLayout	Zur Verwendung mit GUI-Buildern
SpringLayout	Zur Verwendung mit GUI-Buildern

Border-Layout

- Komponenten werden entsprechend der Himmels-richtungen angeordnet
  - NORTH => Page_Start
  - OST => Line_End
  - WEST => Line_Start
  - SOUTH => Page_End
  - CENTER => Center

```
JPanel panel = new JPanel();
panel.setLayout(new BorderLayout());
panel.add(new JButton(),BorderLayout.NORTH);
```

Border-Layout Konstruktoren

- Unterstützt zwei Konstruktoren:
  - new BorderLayout()
  - new BorderLayout(int hpad, int vpad)
- Vorsicht: Methode preferredSize() wird nur in Teilen berücksichtigt:

	Breite	Höhe
NORTH, SOUTH	gesamte Breite des Containers	bevorzugte Höhe der Komponente
WEST, EAST	bevorzugte Breite der Komponente	verbleibender Platz zwischen NORTH und SOUTH
CENTER	verbleibender Platz	verbleibender Platz

Flow-Layout

- Ordnet Komponenten entsprechend der Breite des Darstellungsbereichs fließend hintereinander an.
- Methode perferredSize() wird vollständig berücksichtigt.

Flow-Layout Konstruktoren

- new FlowLayout():
  - zentrierte Darstellung
- new FlowLayout(int align)
  - Darstellung wie definiert: FlowLayout.LEFT, FlowLayout.RIGHT, FlowLayout.CENTER
- new FlowLayout(int align,int hgap, int vgap)
  - Darstellung wie definiert: FlowLayout.LEFT, FlowLayout.RIGHT, FlowLayout.CENTER
  - hgap: horizontaler Abstand zwischen Komponenten
  - vgap: vertikaler Abstand zwischen Komponenten

Grid-Layout

- Ordnet die Elemente in einem Netz von Zellen an. Anzahl von Reihen und Spalten wird im Konstruktor definiert. Einer der Werte darf 0 sein => Anzahl nicht vorgegeben.
- Jede Zelle hat die gleiche Größe.
- Jedes Element füllt seinen Platz komplett aus.
- Konstruktoren:
  - new GridLayout(int rows, int cols)
  - new GridLayout(int rows, int cols, int hgap, int vgap)

Gridbag-Layout

- Erweiterung des GridLayouts. Bietet größte Flexibilität bei der Entwicklung von GUIs.
- Plaziert GUI Komponenten in einem Netz von Zellen. Anzahl von Zeilen und Spalten nicht fest vorgegeben.
- Für jede Komponente wird ein Constraint-Objekt angelegt, welches die Eigenschaften für die Platzierung des Elements definiert.

```
JPanel pane = new JPanel(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();
c.gridx = 1; c.gridy = 1;
pane.add(theComponent, c);
```


Thread-sicheres Starten

- Swing ist im Gegensatz zu AWT nicht thread-sicher. Parallele Ereignisse können zu unvorhergesehenen Ergebnissen führen.
- Zur Vermeidung des Problems, wird der Start der GUI als eigener Thread für den Event Dispatching Thread von AWT eingereiht.
- Die Ausführung erfolgt asynchron, der Aufrufer wartet nicht auf die Rückkehr der Methode.
- Aufruf über statische Methode invokeLater() der Klasse
  - java.awt.EventQueue oder alternativ
  - java.swing.SwingUtilities

Beispielhafter Start

```
public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
                public void run() {
                        try {
                                MyGUI frame = new MyGUI();
                                frame.setVisible(true);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        });
}
```


Weiterführende Links

- Java Swing Tutorial:
- http://download.oracle.com/javase/tutorial/uiswing/index.html
- Einige Beispiele zu Swing
- http://download.java.net/javadesktop/swingset3/SwingSet3.jnlp
- Galileo Open Book: Java ist auch eine Insel
- http://openbook.galileocomputing.de/javainsel9/javainsel_19_001.htm
- GUI-Builder für Eclipse: Window Builder Pro (über Eclipse Marketplace installieren)