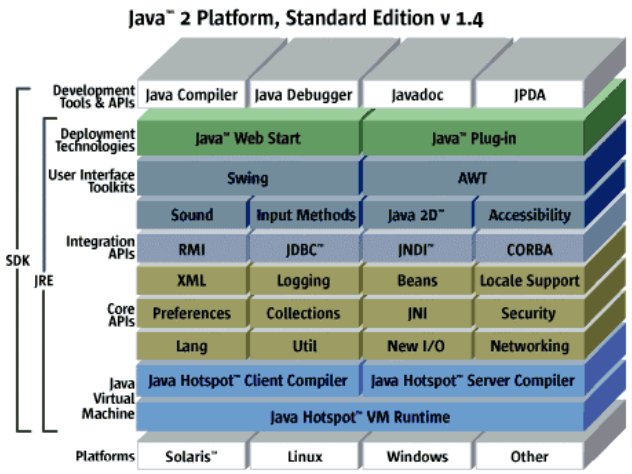
Java I – Vorlesung 11
Graphische Oberflächen mit Swing

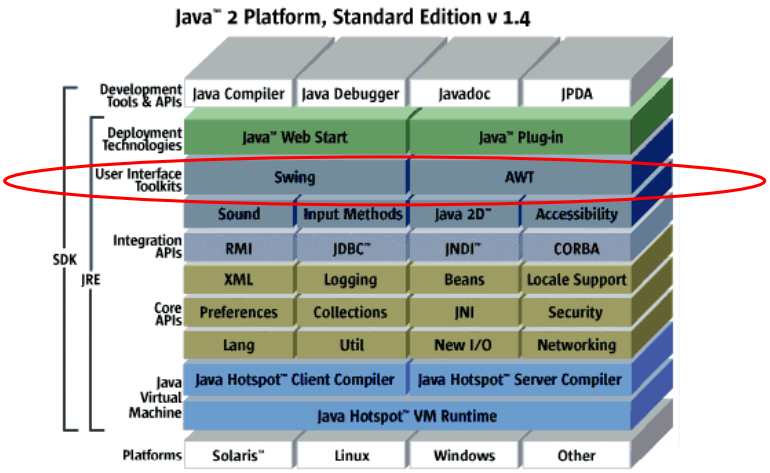
5.7.2004

Swing
Komponenten
Layout-Manager
Events
Swing und Threads

Die Java-Plattform



Die Java-Plattform



"Hello World" in Swing

HelloWorldS.java

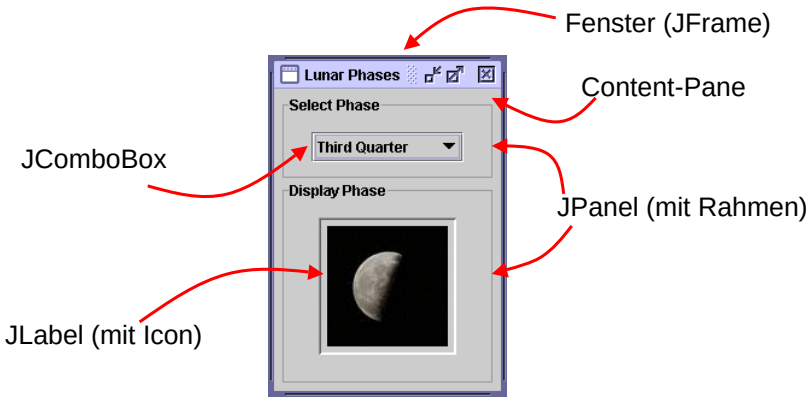
```
import javax.swing.*;


JFrame frame = new JFrame("Hello World");

JLabel label = new JLabel("Hello World!");
frame.add(label);

frame.pack();
frame.setVisible(true);
```

Anatomie eines Fensters



Swing-Komponenten

- Ein Swing-Fenster besteht aus Komponenten.
- Jede Komponente ist als ein Objekt repräsentiert.
- Es gibt drei Typen von Komponenten:
  - Top-Level-Container (JFrame, Applets): ein eigenes Fenster des Betriebssystems
  - Zwischen-Container (JPanel, JSplitPane): Enthalten andere Komponenten
  - Atomare Komponenten (JButton, JTextField)

Schachtelung von Komponenten

- Komponenten können in Container-Komponenten enthalten sein.
- Jede Container-Klasse hat eine Methode add(JComponent), mit der man Komponenten in sie hineinzeichnen kann.
- Alle Swing-Klassen, die mit J anfangen und nicht Top-Level-Container sind, sind von JComponent abgeleitet.

Content Panes

- Top-Level-Container enthalten genaugenommen nicht selbst andere Komponenten.
- Sie haben eine Content Pane (von Typ java.awt.Container), zu der man Komponenten hinzufügen kann.
- Methode add() von JFrame ruft Methode add() der Content-Pane auf, verhält sich also, als ob man direkt Komponenten reintun könnte.

Layout

- Man gibt in Swing normalerweise nicht direkt an, wo eine Komponente gezeichnet wird. Diese Aufgabe erledigen Layout-Manager.
- Vorteile:
  - automatische Reaktion auf Events wie z.B. Änderung der Fenstergröße
  - Portabilität
- Wenn es sein muss, kann man mit Layout-Manager `null` Komponenten absolut positionieren.

Einige Layout-Manager

`LayoutTest.java`

- BorderLayout (Standard für Content-Panes): Lege Komponenten oben/unten/rechts/links ab.
- FlowLayout (Standard für JPanel): Lege Komponenten übereinander ab.
- BoxLayout: Lege Komponenten über- oder nebeneinander ab (flexibler als FlowLayout).
- GridLayout: Lege gleich große Komponenten tabellenförmig ab.
- GridBagLayout: Tabelle mit verschieden großen Einträgen (flexibler als GridLayout).

Hinweise für Layout-Manager

- Manche Layout-Manager respektieren Hinweise über Größe und Position von Komponenten.
- Größe kann man über setPreferredSize() angeben (geht auch minimale, maximale Größe).
- Position:
  - zweites Argument für add()
  - setAlignmentX/Y() (bei BoxLayout)
  - setBounds() (bei absoluter Positionierung)

Layout: Wann?

- Layout für Container und alle enthaltenen Komponenten wird angestoßen mit Aufruf von pack() vor dem ersten Zeichnen.
- Wenn man nach dem Zeichnen neue Komponenten hinzufügt oder alte verschiebt, muss man revalidate() und dann repaint() aufrufen. Damit wird Layout neu berechnet und Fenster neu gezeichnet.
- Bezieht sich jeweils auch auf alle enthaltenen Komponenten.

Einige Komponenten

http://java.sun.com/docs/books/tutorial/uiswing/components/components.html

- Atomare Komponenten:
  - JLabel: Nicht-interaktives Beschriftungsfeld
  - JButton: Button
  - JTextArea: Eingabe von Text
  - Radio- und Checkbuttons
- Container:
  - JFrame: Fenster
  - JPanel: Unsichtbarer Container
  - JSplitPane: hat linken und rechten Teil

JPanel

- Ein JPanel ist einfach ein Rechteck in der Hintergrundfarbe, in das man andere Komponenten legen kann.
- JPanels können Rahmen haben (Methode setBorder).
- JPanels sind essentiell, wenn man mit BoxLayout o.ä. vernünftig aussehende GUIs bauen will.

JScrollPane

`ScrollpaneTest.java`

- Aufgabe von JScrollPane ist es, irgendeine andere Komponente `comp` mit Scrollbalken zu versehen.
- Man legt dann das JScrollPane-Objekt in den Container statt `comp`.
- JScrollPane unterscheidet sich von anderen Containern dadurch, dass enthaltene Komponente im Konstruktor (und nicht in add-Methode) übergeben wird.

Events

- Die wesentliche offene Frage ist jetzt, wie wir darauf reagieren, wenn jemand
  - auf einen Button klickt
  - einen Eintrag in einer Liste auswählt
  - die Fenstergröße ändert
  - auf eine Taste drückt
  - usw.
- Das ist alles unter dem Begriff Ereignis (Event) zusammengefasst.

Event-Listeners

- Wenn ein Event auftritt, erzeugt Swing ein Objekt, das dieses Ereignis darstellt.
- Jedes Ereignis hat eine eindeutige Quelle (meistens Komponente), in der es erzeugt wurde.
- Kann Listener-Interface mit Callback-Methode implementieren und bei der Komponente anmelden.
- Diese Methode wird für jedes Ereignis aufgerufen, die von dieser Komponente erzeugt wird.

Beispiel: Event-Listeners

ListenerTest.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*; // definiert Listener-Interfaces


class MyProgram implements ActionListener {
  private JButton button;

  void drawStuff() {
    ...
    button = new JButton("Klick");
    button.addActionListener(this);
  }

  public void actionPerformed(ActionEvent e) {
    if( e.getSource() == button ) {
      System.err.println("Button geklickt!");
    }
  }
}
```

Andere Events

- Verschiedene Komponente können in Reaktion auf verschiedene Ereignisse verschiedene Events erzeugen.
- Zum Beispiel: ListSelectionListener kann auf Auswahl in JList reagieren.
- Siehe Doku der jeweiligen Komponente.

Swing und Threads

- Swing verwendet einen eigenen Thread (den Event Dispatching Thread, EDT), um zu zeichnen und Events zu verarbeiten.
- Event-Callbacks müssen schnell fertig sein, damit GUI auf weitere Events reagieren kann!! Notfalls neuen Thread erzeugen.
- Swing ist in weiten Teilen nicht thread-safe.
- Sobald eine Komponente gezeichnet wurde, sollte man nur aus dem EDT heraus darauf zugreifen (d.h. verändern oder auslesen).

Swing und Threads

`TextFieldTest.java`

- Insbesondere soll man keine Werte von Komponenten aus dem Haupt-Thread heraus ablesen!
- Typische Verfahren zum Auslesen von Werten:
  - in einem Listener (wird automatisch im EDT ausgeführt)
  - mit invokeAndWait()

Swing und Threads

- Eine Methode in EDT ausführen:
  - SwingUtilities.invokeLater(Runnable)
  - SwingUtilities.invokeAndWait(Runnable)
  - "Runnable" ist Interface, das von Klasse Thread implementiert wird: Methode run().
- Threadsichere Methoden:
  - repaint()
  - revalidate()
  - add...Listener(), remove...Listener()

Zusammenfassung

- Zur Programmierung von GUIs in Java verwendet man Swing (und einige Reste von AWT).
- Swing repräsentiert Fensterkomponenten als Objekte und verwendet eine Menge OOP.
- Layout-Manager kümmern sich um eigentliches Layout.
- In eigenem Thread werden Events erzeugt und an Callback-Methoden übergeben.