Einführung in GUI-Programmierung

javax.swing

GUI – Graphical User Interface („Grafische Benutzerschnittstelle“)

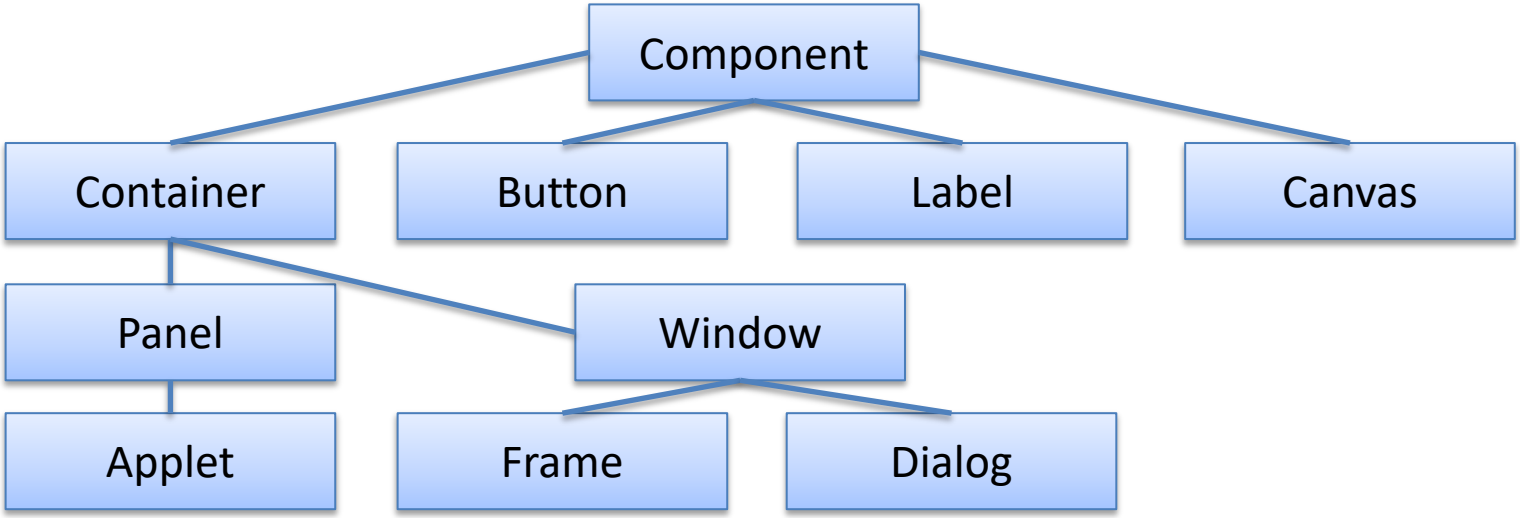
- Anschaulichere und leichtere Möglichkeit zur Dateneingabe und Kontrolle von Programmen
- Stellt Textfelder, Schaltknöpfe, Menüs, usw. zur Verfügung
- Java SE stellt Bibliotheken zur einfachen Implementierung von GUIs zur Verfügung
  - Swing:
    - Früher (Java 1.2 ca. 1998): Swing als fester Bestandteil der JRE
    - Entwickelt von Sun
    - Baut auf dem älteren Abstract Window Toolkit (AWT) auf.
      - Erweiterungen: Drag & Drop, neue Panels und Layouts, weitere Komponenten
  - JavaFX:
    - Heute(Seit Java SE 7 Update 6): JavaFX als fester Bestandteil von Java SE x86
    - Entwickelt von Oracle
    - Soll Swing ablösen
    - Schnell erstellbare neue UI-Komponenten (per CSS gestaltbar).

Heute: Kurze Einführung in **Swing**

- Java Swing-API ist umfangreich und sehr flexibel
- Ist im Package `javax.swing` verankert
  - Bietet ca. 18 weitere Unterpakete:
    - `border, event, table, text, tree,` usw.

Grundlage für die grafische Entwicklung ist AWT (Abstract Window Toolkit).

Ein Auszug aus der Klassenhierarchie (`java.awt`):

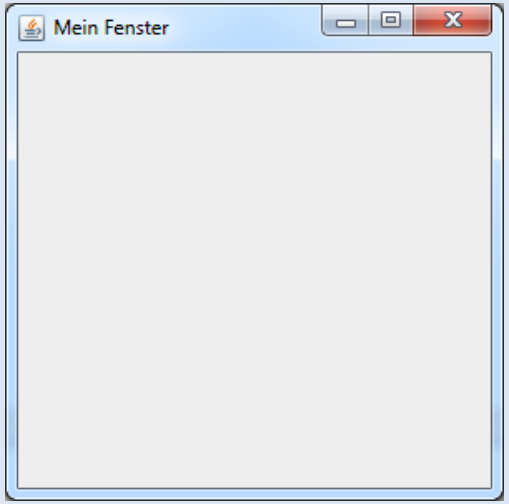


Fenster erstellen mit Swing

Wir wollen nun eine einfache Nutzerschnittstelle entwerfen:

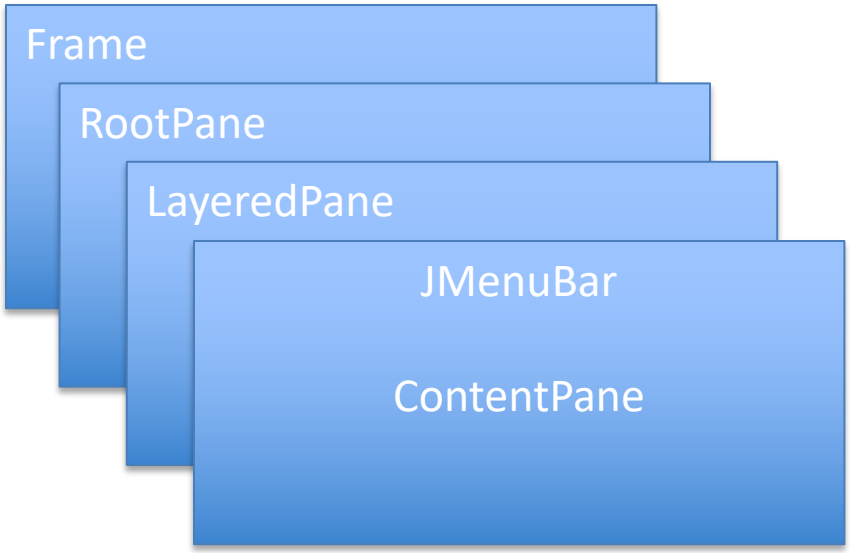
- Mit Hilfe der Klasse `JFrame` aus dem Packet `javax.swing` können wir sehr leicht ein einfaches bewegliches plattformabhängiges Fenster erstellen.
- Die Klasse bringt sehr viele Methoden und Eigenschaften mit, bspw.:
  - `setSize(width, height); // Legt Fenstergröße fest`
  - `setTitle(String title); // Legt Fenstertitel fest`
- Daher am besten von `JFrame` erben:

```
// Beispielcode:
import javax.swing.JFrame;

public class Gui extends JFrame{

   public Gui(){
      this.setTitle(„Mein Fenster");
      this.setSize(300, 300);

      this.setVisible(true);
   }
}
```



Fenster mit Elementen füllen

Allgemeiner Aufbau:



Unser `JFrame` beinhaltet das `JRootPane` als einziges Kind

- Stellt die `ContentPane` zur Verfügung
  - Ist die Basis-Komponente für alle Unterkomponenten

Unserem `JFrame` können nun direkt Elemente (`Components`) hinzugefügt werden:

- `this.add(new JButton(„OK“)) // Fügt einen OK-Button hinzu`

Oder man definiert seine eigene `ContentPane` und gibt diese dem `JFrame`:

- `JPanel contentPane = new JPanel();`
- `setContentPane(contentPane);`

Komponenten in Swing

Als Komponenten stehen sämtliche Bedienelemente zur Verfügung:

- `JButton, JTextField, JLabel`, usw.
- Aber auch neue `JPanel` (Füllwänd)

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JTextField;

public class Gui extends JFrame{

   private JButton ok_btn;
   private JTextField txt_field;

   public Gui(){

      this.setTitle("Mein Fenster");
      this.setSize(300, 300);

      this.ok_btn = new JButton("OK");
      this.txt_field = new JTextField();

      this.add(this.txt_field);
      this.add(this.ok_btn);

      this.setVisible(true);
   }
}
```

Die Komponenten überlagern sich.
=> Wir brauchen also ein Layout

Layouts

Um die Elemente auf eine bestimmte Art anzuordnen, brauchen wir ein Layout.

- Das Packet `java.awt` hält verschiedene Layouts bereit:
  - `BorderLayout, CardLayout, FlowLayout, GridBagLayout, GridLayout`
- Oder auch `javax.swing`:
  - `BoxLayout, Overlayout, GroupLayout`

- Einen guten Überblick über die verschiedenen Layouts gibt das Oracle Java Tutorial: https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

Hier als Beispiel:

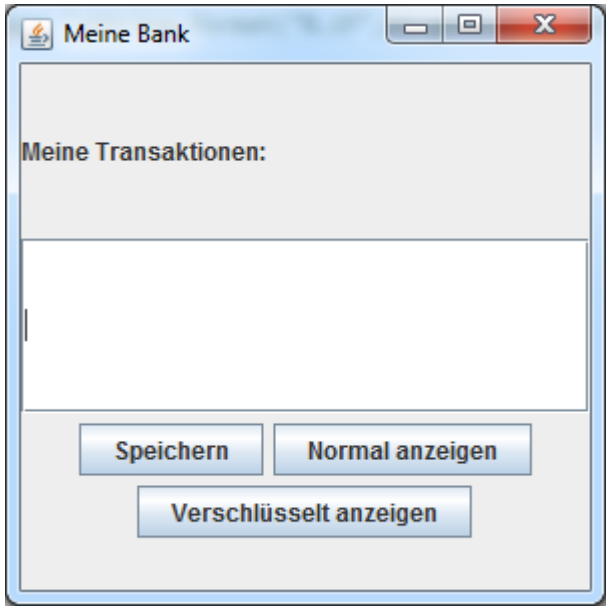
```
…
this.setLayout(new GridLayout(2, 1));
this.add(this.txt_field);
this.add(this.ok_btn);
```

Programmieraufgabe: GUI

Implementieren Sie nun für Ihre Bankkonto-Anwendung eine Einfache GUI,

- die ein Textfeld für die Ein- und Auszahlungen bereithält
- Zudem einen Button „Speichern“
- Und 2 Button „Normal anzeigen“ „Entschlüsselt anzeigen“

Ihre GUI kann beispielsweise wie folgt aussehen:



Button mit Leben füllen

Um nun auf Button-Klicks reagieren zu können, müssen wir das Interface `ActionListener` aus dem Packet java.awt.event implementieren:

- Das geht sehr einfach über eine anonyme innere Klasse
- Oder seit Java 8 über Lambda-Ausdrücke

```
// Einfaches Beispiel:

    this.txt_field = new JTextField();
    this.ok_btn = new JButton("OK");
    this.ok_btn.addActionListener(new ActionListener() {

      @Override
      public void actionPerformed(ActionEvent e) {

        txt_field.setText("OK");
      }
    });

// Per Lambda-Ausdruck:
this.ok_btn.addActionListener((ActionEvent e)-> txt_field.setText("OK"));
```

Hausaufgabe

Versuchen Sie nun Ihre eben erstellte GUI mit Leben zu füllen, indem Sie beim Klick des entsprechenden Buttons die folgenden Aktionen durchführen:

- Speichern: Speichert den Text im Textfeld in eine Datei
- Normal anzeigen: Zeigt den Text von der Datei im Klartext an
- Verschlüsselt anzeigen: Zeigt den Text aus der Datei verschlüsselt an
  - Nehmen Sie hierzu den FilterReader aus der vorherigen Aufgabe