# Inhalt

Erstellen eines Grundgerüsts mit dem Java-Editor ..... 2
Schritt 1: Erstellen des Grundgerüsts mit dem Java-Editor http://www.javaeditor.org ..... 2
Schritt 2: Anpassen des Codes in der Programmierumgebung, zum Beispiel BlueJ ..... 3
Wichtige Klassen und einige ihrer Methoden ..... 4
JButton und JLabel ..... 4
JTextField ..... 4
JTextArea ..... 4
JPanel (extends Container) ..... 4
JMenuBar ..... 4
JMenu ..... 4
JMenuItem ..... 4
JFrame ..... 4
Container – betrifft vor allem die contentPane des JFrames ..... 4
ActionListener ..... 5
Exkurs: Bilder als Hintergrund ..... 6
Menüs anlegen ..... 7
Wechselnde Oberflächen ..... 8

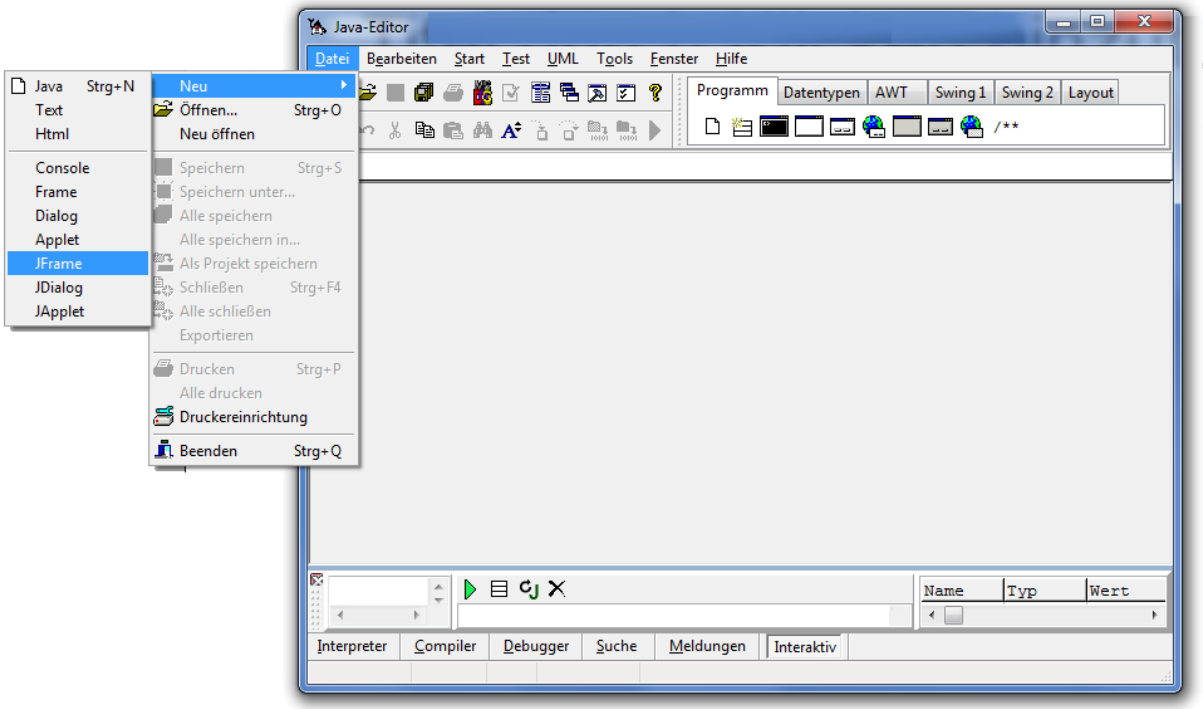
# Erstellen eines Grundgerüsts mit dem Java-Editor

*Zusammenfassung: Mit dem Java-Editor wird ein Grundgerüst erstellt, das danach manuell angepasst wird. Das angepasste Modell lässt sich nicht ohne weiteres wieder im Editor verändern – wenn einem danach einfällt, dass man gerne noch einen Button mehr hätte, macht man das besser im Code manuell oder fängt nach dem Prinzip "build one to throw one away" wieder ganz von vorne mit einer verbesserten Version an..*

## Schritt 1: Erstellen des Grundgerüsts mit dem Java-Editor

http://www.javaeditor.org

1. Ein neues JFrame erzeugen (Menü: Datei/Neu/JFrame), Größe anpassen, Attributwerte im Objekt-Inspektor oder mit Drag-and-Drop verändern




2. Unter dem Reiter "Swing1" Elemente auswählen und auf dem JFrame platzieren
   - **Knöpfe: JButton, auswählen über das [OK]-Icon**
     - Attributwerte (Position, Größe, Beschriftung) mit Maus oder Objekt-Inspektor ändern
     - das Attribut "Name" ist dabei der Bezeichner für das Referenzattribut, mit dem der Knopf angesprochen wird; wenn Zeit ist, den gleich auf sinnvollen Wert ändern
     - zu jedem JButton wird automatisch eine Methode ergänzt, in die man später einträgt, was beim Drücken des Knopfes geschehen soll
   - **Beschriftungsfelder: JLabel, auswählen über das J-Icon**
     - Attributwerte (auch "Name" als Variablenbezeichner) nach Wunsch anpassen
     - später kann man mit `bezeichner.getText()` und `bezeichner.setText(String)` die Beschriftung des Feldes ändern oder auslesen
   - **Texteingabefelder: JTextField, auswählen über das Icon mit dem "a" darin**
     - ähnlich wie JLabel, nur dass der dort Benutzer manuell Text eintragen kann
     - auslesen und setzen mit `getText()` bzw. `setText(String)`
   - **JTextArea, auswählen über das Icon mit der Seite**

- Für größere Textmengen, kann Scrollbalken enthalten
- neben den setText- und getText.Methoden gibt es auch `append(String)` zum Hintenanfügen

## Schritt 2: Anpassen des Codes in der Programmierumgebung, zum Beispiel BlueJ

Im einfachsten Fall (und beim Arbeiten nach dem Model-View-Controller-Prinzip) macht die GUI nicht viel anderes, als Nachrichten an die mit ihr verbundene Steuerung zu schicken und auf empfange Nachrichten zu reagieren. Also muss man manuell im Code zwei Dinge ergänzen:

1. Ein Referenzattribut, z.B.: `Steuerung steuerung = new Steuerung();`
2. Eine Methode zum Senden von Nachrichten an die Steuerung:
```
void senden(String s) {
   steuerung.empfangen(s);
}
```

Für einen Knopf z.B. mit dem Variablenbezeichner jButton1 hat der Java-Editor bereits eine Methode erstellt, in die man schreibt, was beim Drücken des Knopfs geschieht:

3. 
```
public void jButton1_ActionPerformed(ActionEvent evt) {
   // TODO hier Quelltext einfügen
   senden("XXX");
} // end of jButton1_ActionPerformed
```

Und die Zeile mit dem //TODO ersetzt man durch `senden("X");` womit beim Knopfdruck automatisch ein "X" an die Steuerung gesendet wird.

# Wichtige Klassen und einige ihrer Methoden

## JButton und JLabel

- `void setText(String)`
- `String getText()`
- `void setIcon(ImageIcon)`
- `void setBackground(Color) (z.B. Color.BLUE)`
- `void addActionListener(ActionListener) //nur JButton`

## JTextField

- `void setText(String)`
- `String getText()`

## JTextArea

- `void setText(String)`
- `String getText()`
- `void append(String)`

## JPanel (extends Container)

- `void setBackground(Color) (z.B. Color.BLUE)`
- `setOpaque(boolean) //durchsichtig oder nicht, wichtig fuer Farbe`
- `void setText(String)`
- `String getText()`

## JMenuBar

- `JMenuBar() //Konstruktor`
- `add(JMenu) //Menüspalte hinzufuegen`

## JMenu

- `JMenu(String) //Konstruktor; der String gibt den Spaltentitel an`
- `add(JMenuItem)//Menüpunkt hinzufügen`

## JMenuItem

- `JMenuItem(String) //Konstruktor, der String gibt den Text an`
- `void addActionListener(ActionListener)`

## JFrame

- `void setResizable(boolean)`
- `void setVisible(boolean)`
- `setJMenuBar(JMenuBar) //Menüzeile festlegen`
- `Container getContentPane() //gibt die oberste Ebene zurück; //wichtig, weil alle anderen darzustellenden Elemente zu dieser //Ebene hinzugefüht werden`

## Container – betrifft vor allem die contentPane des JFrames

- `add(Component) //JButtons, JLabels, JPanels etc.`
- `remove(Component) //JButtons, JLabels, JPanels etc.`
- `void validate()`
- `void repaint()`

# ActionListener

Knöpfe und Menüeinträge lassen sich relativ leicht erstellen. Damit auch etwas geschieht, wenn man darauf klickt, muss jeder von ihnen einen zugeordneten ActionListener erhalten, der überwacht, ob jemand gerade darauf klickt und dementsprechend Aktionen auslöst.

Der Java-Editor erzeugt den Code zum Erstellen der ActionListener gleicht mit, vom Benutzer muss dann nur an der markierten Stelle der eigene Code ergänzt werden. Meist werden dabei nur Nachrichten an die assoziierte Steuerugn geschickt.

# Exkurs: Bilder als Hintergrund

JButtons und JLabels können als Hintergrundgrafik ein Bild erhalten. Oft sieht man das so:

```
ImageIcon jButton1Icon = new ImageIcon("bildname.png");
jButton1.setIcon(jButton1Icon);
```

Das funktioniert auch, allerdings kann diese Art der Pfadangabe Probleme machen, wenn der Java-Code von einem unerwarteten Verzeichnis aus aufgerufen wird. Alle derartigen Pfadangaben sind nämlich relativ zum aktuellen Java-Verzeichnis und *nicht* relativ zu der Datei, in der der Code steht. Siehe auch: http://www.bluej.org/help/archive.html#tip10

Also verwendet man besser eine Variante folgender Hilfsmethode, die aus dem Pfad zur Bilddatei eine URL erzeugt, und erst aus dieser das Bild generiert.

```
public ImageIcon bildErzeugen(String bildname) {
  java.net.URL bildURL;
  bildURL = getClass().getClassLoader().getResource(bildname);
  return new ImageIcon(bildURL);
}
```

Damit sieht der Code zum Erzeugen von Hintergrundbildern so aus:

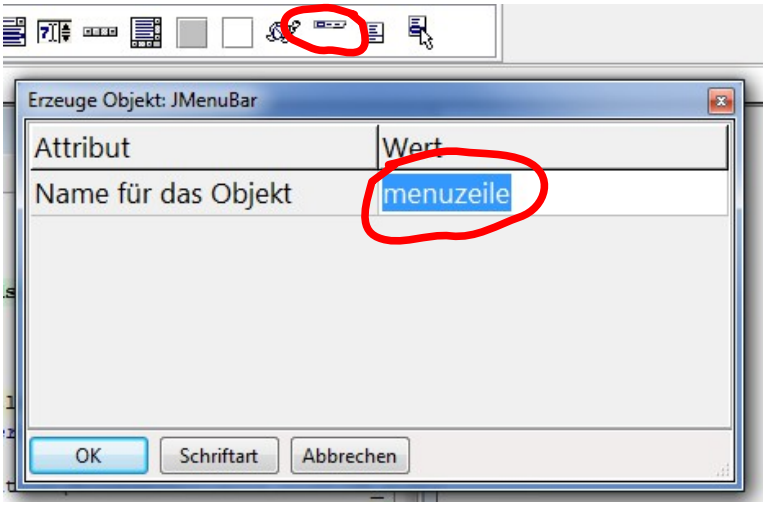
```
ImageIcon jButton1Icon = bildErzeugen("bildname.png");
jButton1.setIcon(jButton1Icon);
```

Hintergründe für JPanels gibt es leider keine – in diesem Fall kann man schnell ein großes JLabel erzeugen und über das Label platzieren.

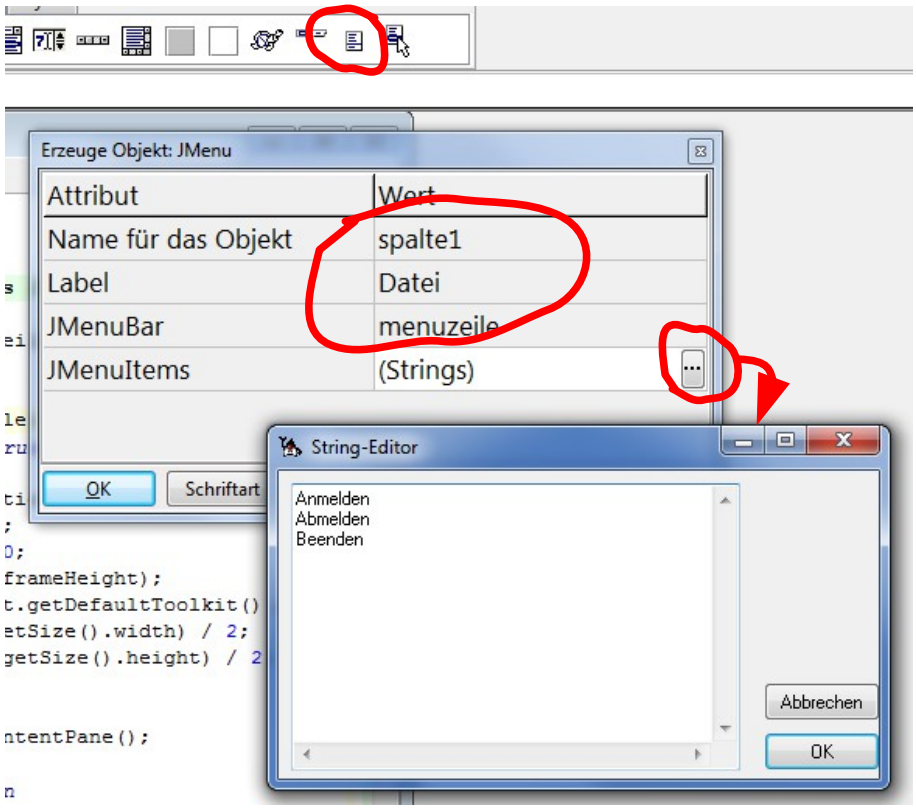
# Menüs anlegen

GUI erstellen mit Java-Editor: http://www.javaeditor.org

1. JMenuBar im Reiter "Swing 1" auswählen . Einen geeigneten Bezeichner wählen.




2. Ebenfalls unter "Swing 1" das Icon für "JMenu" auswählen. Im Menü dann angebe:
a) Variablenbezeichner (z.B. spalte1)
b) Beschriftung des Menüs (z.B. Datei)
c) zu welcher Menüzeile das Menü gehört (sinnvollerweise der eben zuvor angelegten)
d) welche Menüeinträge in das Menü sollen (einfach untereinanderschreiben, hier z.B. Anmelden/Abmelden/Beenden)



## Wechselnde Oberflächen

GUI erstellen mit Java-Editor: http://www.javaeditor.org

Manchmal soll sich die ganze Oberfläche (bis auf die Menüs) auf Anweisung der Steuerung verändern. Dafür gibt es mehrere Möglichkeiten:

- Man schaltet manuell die Elemente, die man haben will, `setVisible(true)` und die anderen `setVisible(false)`. Das geht aber eigentlich nur, wenn es nur um wenige Elemente geht.
- Mit der Klasse CardLayout – für Schule eher unnötig.
  http://docs.oracle.com/javase/tutorial/uiswing/layout/card.html
- Empfohlen: Man legt mehrere JPanels an, am besten durch selbst erstellte Unterklassen davon. Diese fügt man dann je nach Wunsch der contentPane-Ebene des JFrames hinzu oder nimmt sie weg.
  ```
  Container cp = this.getContentPane();
  cp.add(jPanelA);
  cp.remove(JPanelB);
  cp.repaint();
  ```
- Siehe Projekt "GUI/Komponente Panelwechsler".