SWING
Steuerelemente

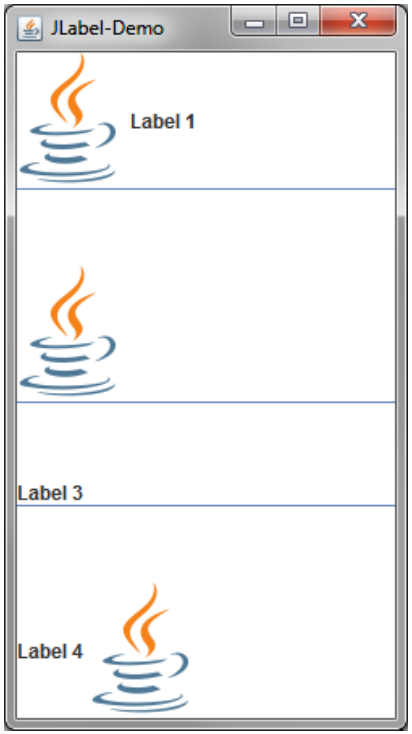
Steuerelemente

- alle Steuerelemente sind von **`java.awt.Component`** bzw. **`javax.swing.JComponent`** abgeleitet

- haben alle folgende Eigenschaften:
  - rechteckige Form bestimmter Größe
  - Rahmen, der durch ein Objekt vom Typ **`javax.swing.Border`** beschrieben wird
  - relative Position zum enthaltenden Container (immer linke obere Ecke)
  - Referenz auf den enthaltenden (übergeordneten) Container
  - Hinter- und Vordergrundfarbe vom Typ **`java.awt.Color`**
  - Schriftart vom Typ **`java.awt.Font`**
  - Darstellung des Mauszeigers für MouseOver vom Typ **`java.awt.Cursor`**
  - Tooltip (kurze Zeichenkette, die bei längerem MouseOver erscheint)
  - Zustandsinformationen, ob Steuerelement aktiv und sichtbar ist
  - Fähigkeit, Fokus anzunehmen (immer nur eine Komponente zur gleichen Zeit)
  - Methoden zum Registrieren an Listener (**`addXXXListener`**)
  - Grafikkontext zum Zeichnen in der Komponente (Typ **`java.awt.Graphics`**)

- **`https://docs.oracle.com/javase/tutorial/uiswing/components/jcomponent.html`**

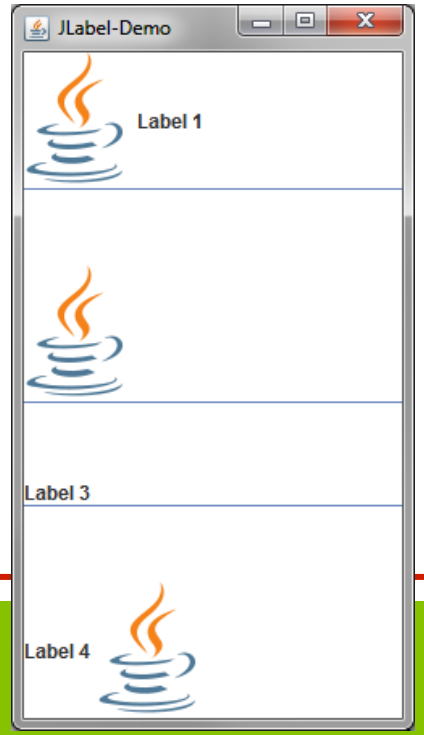
JLabel



- JLabel sind Beschriftungsfelder zur Anzeige von einzeiligem Text und/oder einem Bild
- Konstruktoren:
  - `JLabel()`
  - `JLabel(Icon image)`
  - `JLabel(Icon image, int horizontalAlignment)`
  - `JLabel(String text)`
  - `JLabel(String text, Icon icon, int horizontalAlignment)`
  - `JLabel(String text, int horizontalAlignment)`
- implementiert die Schnittstelle SwingConstants mit folgenden Konstanten:
  - `SwingConstants.LEFT` (nach links ausgerichtet)
  - `SwingConstants.RIGHT` (nach rechts ausgerichtet)
  - `SwingConstants.CENTER` (zentriert)
  - `SwingConstants.LEADING` (Text vor Symbol)
  - `SwingConstants.TRAILING` (Text nach Symbol)

JLabel Beispiel (JLabelDemo.java)

```
ImageIcon icon   = new ImageIcon("java_logo.png"); // in den Projektordner
JLabel label1    = new JLabel("Label 1", icon, SwingConstants.RIGHT);
JLabel label2    = new JLabel();
JLabel label3    = new JLabel("Label 3");
JLabel label4    = new JLabel("Label 4", icon, SwingConstants.LEFT);

label2.setIcon(icon);
label4.setHorizontalTextPosition(SwingConstants.LEADING);

JPanel panel = new JPanel();
panel.setBackground(Color.WHITE);
panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
panel.add(label1);
panel.add(new JSeparator());
panel.add(label2);

    ...
```



Schaltflächen

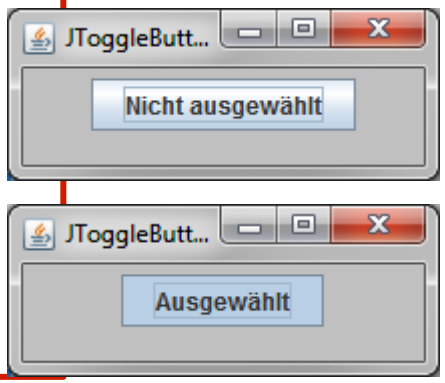
- von AbstractButton abgeleitet:
  - Schaltflächen (JButton, JToggleButton)
  - Kontrollkästchen (JCheckBox)
  - Optionsfelder (JRadioButton)
  - Menükomponenten (JMenu, JMenuItem, JRadioButtonItem, JCheckBoxMenuItem)

- einige Methoden:
  - addActionListener(ActionListener l)
  - addChangeListener(ChangeListener l)
  - addItemListener(ItemListener l)
  - doClick() – simuliert per Programm ein Klick auf die Schaltfläche
  - String getActionCommand() – gibt Aktionsbefehl der Schaltfläche zurück
  - boolean isSelected() – gibt Zustand der Schaltfläche zurück
  - setIcon(Icon symbol) – legt Bild auf der Schaltfläche fest
  - …

- JButton zeigen wir hier nicht … kennen wir ja auch schon, aber …

JToggleButton Beispiel (JToggleButtonDemo.java)

```
// Komponenten einfügen
      toggleButton = new JToggleButton("Nicht ausgewählt");
      toggleButton.addItemListener(new ButtonListener());
    ...

// Ereignisbehandlungsmethoden für Komponenten
   class ButtonListener implements ItemListener
   {
      public void itemStateChanged(ItemEvent e) {
         int zustand = e.getStateChange();

         if(zustand == ItemEvent.SELECTED)
           toggleButton.setText("Ausgewählt");
         else
           toggleButton.setText("Nicht ausgewählt");
      }
   }
```

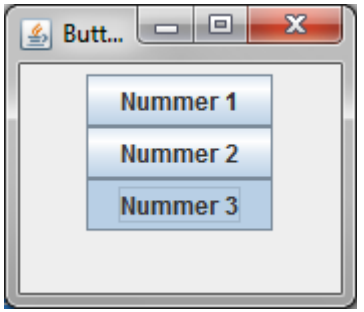


ButtonGroup

- mehrere Schaltflächen werden zu einer Gruppe zusammengefasst
- nur jeweils eine Schaltfläche darin kann aktiv sein
- besonders geeignet für JRadioButton- und JToggleButton-Gruppen
- ausgewählte Methoden:
  - ButtonGroup() - Konstruktor; erzeugt eine Schaltflächengruppe
  - add(AbstractButton b) – fügt b in Schaltflächengruppe ein
  - int getButtonCount() – gibt Nummer der Schaltfläche in der Gruppe zurück
  - remove(AbstractButton b) – entfernt b aus Schaltflächengruppe
- klickt der Nutzer auf eine Schaltfläche, bleibt genau diese als einzige Schaltfläche innerhalb der Gruppe aktiv bis eine andere ausgewählt wird – dann ist nur diese aktiv

ButtonGroup Beispiel (ButtonGroupDemo.java)

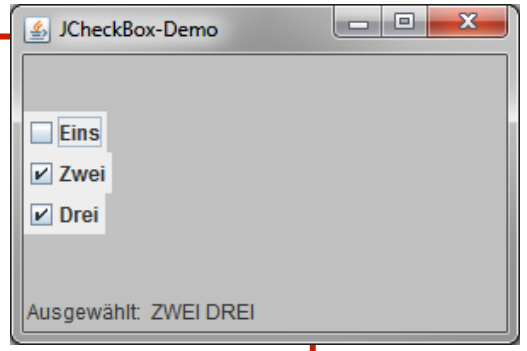
```
// ButtonGroup erzeugen
     ButtonGroup gruppe = new ButtonGroup();


// Wechselschalter erzeugen und in Gruppe einfügen
// Der erste Schalter wird anfangs ausgewählt
        JToggleButton eins, zwei, drei;

        eins = new JToggleButton("Nummer 1", true);
        zwei = new JToggleButton("Nummer 2");
        drei = new JToggleButton("Nummer 3");
        gruppe.add(eins);
        gruppe.add(zwei);
        gruppe.add(drei);
```



JCheckBox (JCheckBoxDemo.java)

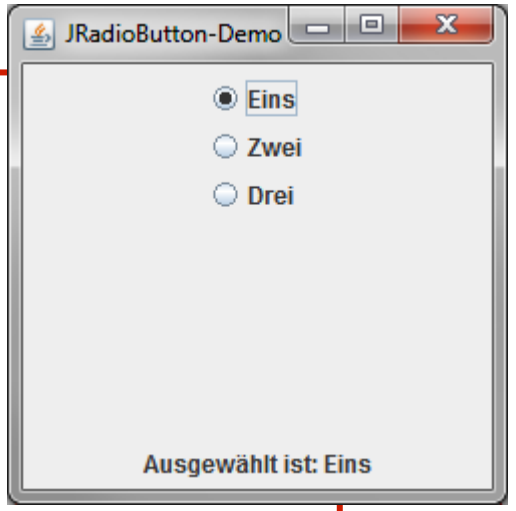
- Schaltflächen, die zwei Zustände annehmen können (von `JToggleButton` abgeleitet)
- lösen Aktionsereignisse (`ActionEvent`) und Änderungsereignisse (`ItemEvent`) aus

```
JCheckBox eins, zwei, drei;
eins = new JCheckBox("Eins");
zwei = new JCheckBox("Zwei");
drei = new JCheckBox("Drei");


public void itemStateChanged(ItemEvent e) {
        String s = "Ausgewählt: ";

        // Feststellen, welche Kontrollkästchen ausgewählt sind
        if (eins.isSelected()) s = s + " EINS";
        if (zwei.isSelected()) s = s + " ZWEI";
        if (drei.isSelected()) s = s + " DREI";

        // Ergebnis in Beschriftungsfeld anzeigen
        label.setText(s);
}
```



JRadioButton (JRadioButtonDemo.java)

- ebenfalls Schaltflächen, die zwei Zustände annehmen können
- werden meistens einer Gruppe zugeordnet → dann immer nur genau ein RadioButton ausgewählt




```
JRadioButton eins = new JRadioButton("Eins");
eins.setSelected(true);
JRadioButton zwei = new JRadioButton("Zwei");
JRadioButton drei = new JRadioButton("Drei");

ButtonGroup gruppe = new ButtonGroup();
gruppe.add(eins);
gruppe.add(zwei);
gruppe.add(drei);

public void actionPerformed(ActionEvent e) {
        JRadioButton button = (JRadioButton) e.getSource();
        String ausgewaehlt = button.getActionCommand();
        anzeigeLabel.setText("Ausgewählt ist: " + ausgewaehlt);
}
```

JList (JListDemo.java)

- Auswahl einer oder mehrerer Alternativen (Verwendung von Shift- und Strg-Taste)
- Anlegen einer Liste, z.B.:


```
JList<String> liste = new JList<String>();
String[] elemente = {"Eins", "Zwei", "Drei", "Vier", "Fünf", "Sechs",
                     "Sieben", "Acht", "Neun", "Zehn", "Elf", "Zwölf",
                     "Dreizehn", "Vierzehn", "Fünfzehn"};
liste.setListData(elemente);
```

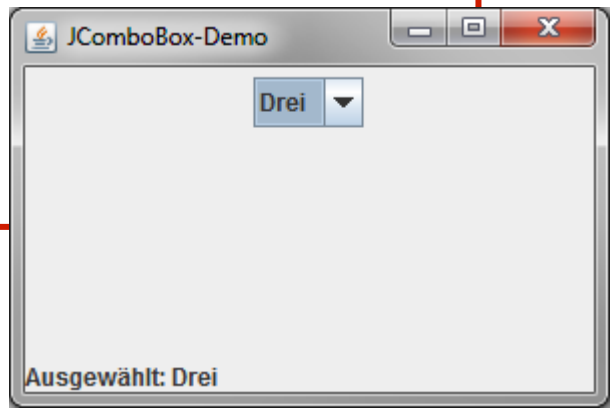
- üblicherweise zusammen mit `JScrollPane` (Scrollbalken) verwendet
- jeder Eintrag kann im Programm über den Index ausgewählt werden; es sollte dann auch noch darauf geachtet werden, dass das ausgewählte Element im sichtbaren Bereich ist

```
liste.setSelectedIndex(11);          // Auswahl der „Zwölf“
liste.ensureIndexIsVisible(11);      // „Zwölf“ im sichtbaren Bereich
```

JComboBox (JComboBoxDemo.java)

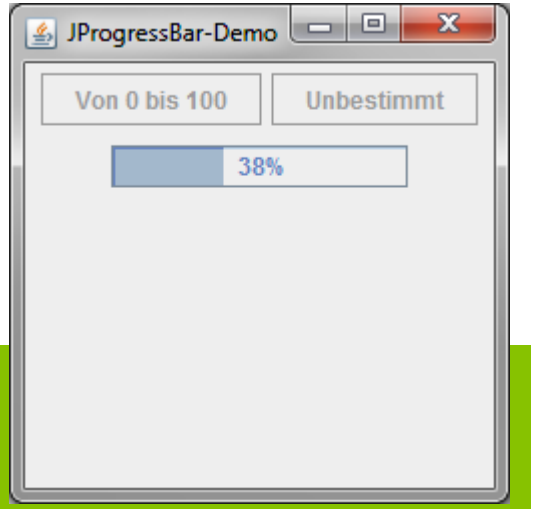
- Auswahl genau eines Elementes aus einer Liste

```
String[] elemente = new String[]{"Null", "Eins", "Zwei", "Drei", "Vier",
"Fünf"};
JComboBox<String> combo = new JComboBox<String>(elemente);
int anzahl      = combo.getItemCount();
combo.setSelectedIndex(anzahl-1); // letzten Eintrag selektieren
combo.addItemListener(this);

public void itemStateChanged(ItemEvent e) {
     int zustand = e.getStateChange();
     String s = "";
     if(zustand==ItemEvent.SELECTED)
     s="Ausgewählt: "+e.getItem();
     anzeigeLabel.setText(s);
}
```



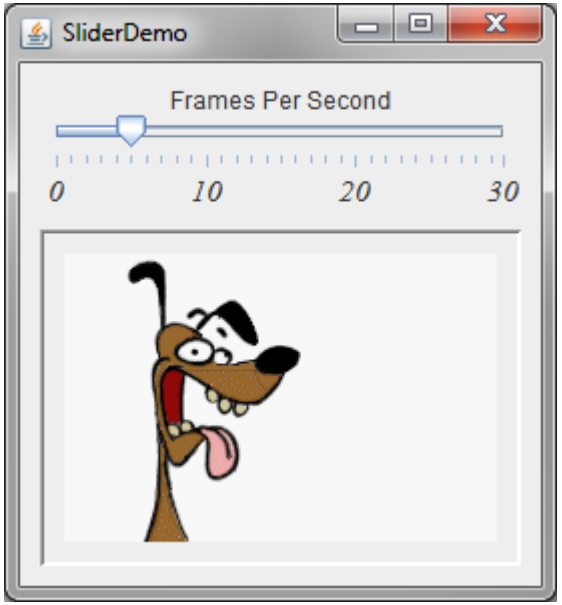
JProgressBar(JProgressBarDemo.java)

- Fortschrittsanzeige (von-bis oder unbestimmt; vertikal oder horizontal)
- einige Methoden:
  - `JProgressBar()` horizontale Fortschrittsanzeige mit Rahmen, aber ohne Fortschrittstext
  - `JProgressBar(int orientation)` `JProgressBar.VERTICAL` `JProgressBar.HORIZONTAL`
  - `JProgressBar(int min, int max)` kleinster und größter Wert
  - `setIntermediate(boolean wert)` true: ohne genauen Fortschrittswert
  - `setString(String s)` legt den anzuzeigenden Fortschrittstext fest
  - `setStringPainted(boolean show)` true: aktiviert Anzeige des Fortschritts-textes
  - `setValue(int n)` setzt die Fortschrittsanzeige auf Wert n



JSlider (`SliderDemo.java`)

- zur einfachen Eingabe numerischer Werte aus einem bestimmten Bereich
- siehe http://docs.oracle.com/javase/tutorial/uiswing/components/slider.html
- SliderDemo.java aus http://docs.oracle.com/javase/tutorial/uiswing/examples/components/index.html#SliderDemo



Aufgabe 5: TicTacToe

- Implementieren Sie das Spiel TicTacToe!
- durch Drücken der Buttons „Rot beginnt“ bzw. „Schwarz beginnt“ wird ein neues Spiel gestartet
- es muss sofort erkannt werden, wenn ein Spieler gewonnen hat bzw. wenn unentschieden ist → dann ist nur noch der Neustart des Spiels möglich
- ein einmal gestztes X bzw. O darf nicht mehr überschrieben werden können