— Übungsblatt 2 (T) —

Aufgabe 6 (T) (Frames)

Als Entwurf für ein einfaches Noteneingabesystem sei folgende `JFrame`-Klasse gegeben:

```
import java.awt.*;
import javax.swing.*;
public class NotenEingabe extends JFrame {
  Container c;
  public NotenEingabe() {
    c = getContentPane();
    c.setLayout(new GridLayout(5, 1));
    c.add(new JCheckBox("sehr gut"));
    c.add(new JCheckBox("gut"));
    c.add(new JCheckBox("befriedigend"));
    c.add(new JCheckBox("ausreichend", true));
    c.add(new JCheckBox("ungenuegend"));
  }
}
```

Schreiben Sie eine Klasse `NotenEingabeTest`, die in ihrer `main`-Methode ein Objekt der Klasse `NotenEingabe` als Frame mit Breite 150 und Höhe 200 Pixel erzeugt und anzeigt.

Geben Sie außerdem eine alternative Klasse `NotenEingabeNeu` an für Frames, die beim Aufruf prinzipiell die gleiche Gestalt wie `NotenEingabe`-Objekte haben, jedoch sicherstellen, dass bei der Eingabe der Noten nur eine Box, also jeweils genau eine Note markiert (angeschaltet) werden kann. Außerdem soll unterhalb der Noten-Boxen eine zusätzliche Box platziert werden, die es erlaubt, mittels ihrer Markierung zu kennzeichnen, dass die Note aus einer Wiederholungsprüfung stammt.

Auch ein Objekt dieser Klasse soll schließlich im Programm `NotenEingabeTest` erzeugt und angezeigt werden.

Aufgabe 7 (T) (Frames)

Skizzieren Sie jeweils das Erscheinungsbild der vier Frames, die vom Programm

```
import java.awt.*;
import javax.swing.*;
public class VierButtonFrame extends JFrame {
  Container c;

  public VierButtonFrame(int i) {
    c = getContentPane();
    if (i==1)
      c.setLayout(new FlowLayout());
    else if (i==2)
      c.setLayout(new BorderLayout());
    else if (i==3)
      c.setLayout(new GridLayout());
    else
      c.setLayout(new GridLayout(0,1));

    c.add(new JButton("A"));
    c.add(new JButton("B"));
    c.add(new JButton("C"));
    c.add(new JButton("D"));
  }

  public static void main(String[] args) {
    VierButtonFrame[] fenster = new VierButtonFrame[4];
    for (int i=0; i<4; i++) {
      fenster[i] = new VierButtonFrame(i+1);
      fenster[i].setTitle("Fenster " + (i+1));
      fenster[i].setSize(200,200);
      fenster[i].setLocation(i*200,0);
      fenster[i].setVisible(true);
      fenster[i].setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
  }
}
```

erzeugt werden.

Aufgabe 8 (T) (Frames)

Ändern Sie das Programm `VierButtonFrame` so ab, dass jedes `JButton`-Objekt mit einer zufällig gewählten Hintergrundfarbe versehen wird. Die RGB-Werte der Farben sollen als Tooltips erscheinen, wenn man mit dem Mauszeiger über den entsprechenden Button fährt. Verwenden Sie dazu keine normalen `JButton`-Objekte, sondern Objekte vom Typ `ColorButton`, einer selbst zu implementierenden Klasse, deren Konstruktor dafür sorgt, dass die `ColorButton`-Objekte mit Zufalls-Farbe und Tooltips ausgestattet sind.

Aufgabe 9 (T) (Zeichnen)

Betrachten Sie die folgende Klasse:

```
import java.awt.*;
import javax.swing.*;
public class GraphZeichnen extends JFrame {

  public GraphZeichnen(int[] x, int[] y) {
    Container c = getContentPane();
    Zeichnung z = new Zeichnung(x, y);
    c.add(z);
  }

  public static void main(String[] args) {
     // Funktion festlegen:
     String t;      // Bezeichnung der Funktion
     Funktion f;    // Funktions-Objekt

     t = "Identitaet";       // fuer 'f(x) = x'
     f = new Identitaet();   // fuer 'f(x) = x'

     // Werte-Tabelle berechnen:
     int anzahl = 500;
     // x- und y-Werte
     double[] x = new double[anzahl];
     double[] y = new double[anzahl];
     // x- und y-Werte in Pixel-Koordinaten
     int[] ix = new int[anzahl];
     int[] iy = new int[anzahl];

     for (int i=0; i < anzahl; i++) {
       x[i] = i * 0.01;          // X-Wert festlegen
       y[i] = f.rechne(x[i]);    // Funktionswert berechnen
                                 // umd in Pixelkoordinaten umrechnen
       ix[i] = i;
       iy[i] = - (int) (100 * y[i] - anzahl);
     }

     // Zeichnen:
     GraphZeichnen fenster = new GraphZeichnen(ix, iy);
     fenster.setTitle(t);
     fenster.setSize(anzahl+15,anzahl+40);
     fenster.setVisible(true);
     fenster.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

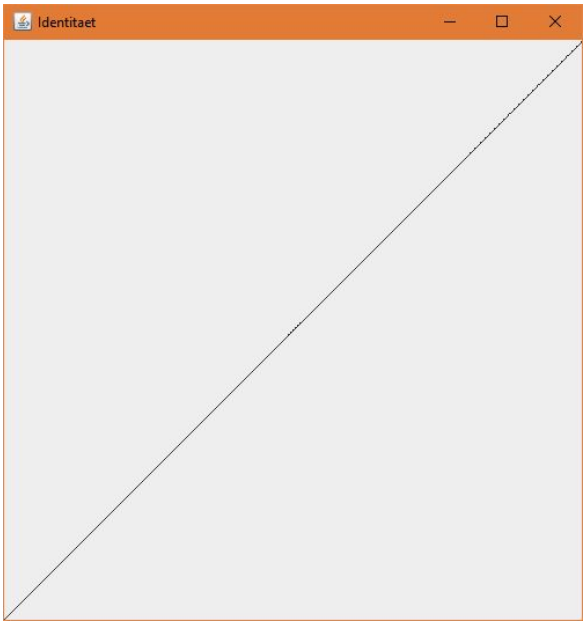
Die Klasse `GraphZeichnen` dient dazu bei ihrem Start eine graphische Oberfläche zu erzeugen, auf der ein Funktions-Graph gezeichnet wird. Ihre `main`-Methode arbeitet wie folgt:

- Zunächst wird der Name der Funktion auf `"Identitaet"` (für $f(x) = x$) als String-Objekt festgelegt.
- Danach wird das passende Objekt vom Typ `Funktion` erzeugt, mit dem später die Funktionswerte ausgerechnet werden können. Auch hier wird mit der Identität ($f(x) = x$) als Funktion gearbeitet.
- Das Interface `Funktion` und die Klasse `Identitaet` sind dazu wie folgt vorgegeben:

```
public interface Funktion {
  public double rechne (double x);
}

public class Identitaet implements Funktion {
  public double rechne (double x) {
    return x;
  }
}
```

- Danach wird die Werte-Tabelle für 500 Funktionswerte für $0 \leq x < 5$ erstellt, indem jeweils die Methode `rechne` angewandt wird, und anschließend die entsprechenden x- und y-Werte in Pixelkoordinaten umgerechnet.
- Schließlich wird ein Objekt der Klasse `GraphZeichnen` erzeugt, wobei dem Konstruktor die beiden Arrays mit den Pixelkoordinaten übergeben werden.
- Das eigentliche Zeichnen des Graphen übernimmt schließlich die grafische Oberflächenkomponente vom Typ `Zeichnung`. Diese Klasse fehlt noch.
- Mit funktionsfähiger Klasse `Zeichnung` hat die Oberfläche folgende Gestalt:



a) Schreiben Sie ein Klasse `Zeichnung`, die von `JPanel` erbt. Statten Sie die Klasse wie folgt aus:

- Definieren Sie zwei private Instanzvariablen `x` und `y` vom Typ `int[]`.
- Definieren Sie einen Konstruktor, der die beiden Instanzvariablen mit seinen Parametern initialisiert.
- Überschreiben Sie die Methode `paintComponent`, sodass sie mithilfe der `Graphics`-Methode `drawPolyline` den durch die Arrays `x` und `y` festgelegten Streckenzug darstellt.

b) Definieren Sie weitere Klassen (z. B. `Quadrat` oder `Wurzel`), die das Interface `Funktion` geeignet implementieren.

c) Bauen Sie dann die `main`-Methode der Klasse `GraphZeichnen` so aus, dass in einer `switch`-Anweisung abhängig vom Programm-Parameter (`args[0]`) oder einer Benutzer-Eingabe die Funktions-Bezeichnung `t` festgelegt und ein entsprechendes Funktions-Objekt `f` erzeugt werden kann, für das dann der Graph gezeichnet werden kann.

Beispiele: