Konfigurierbare Grafiken

Um eine Grafik einstellbar zu machen, muss man zuerst entscheiden, was überhaupt einstellbar sein soll.

Beispiel: Erstelle eine Grafik, die ein Rechteck in der Mitte des Panels zeichnen. Die Breite des Rechtecks soll der halben Panel-Breite entsprechen und die Höhe der halben Panel-Höhe. Die Farbe des Rechtecks soll mit Hilfe von drei Buttons (rot, grün, blau) auswählbar sein.

Im obigen Beispiel muss sich die Grafik an verschiedene Gegebenheiten anpassen:

Anpassungen, mit Hilfe von inneren Eigenschaften:

- Position, immer in der Mitte → abhängig von der Breite bzw. Höhe des Grafik-Panels
- Größe des Rechteckes → abhängig von der Breite bzw. Höhe des Grafik-Panels

Anpassungen, die von außen vorgenommen werden:

- Farbe des Rechteckes (mit Hilfe der Buttons einstellbar)

Übung 1:

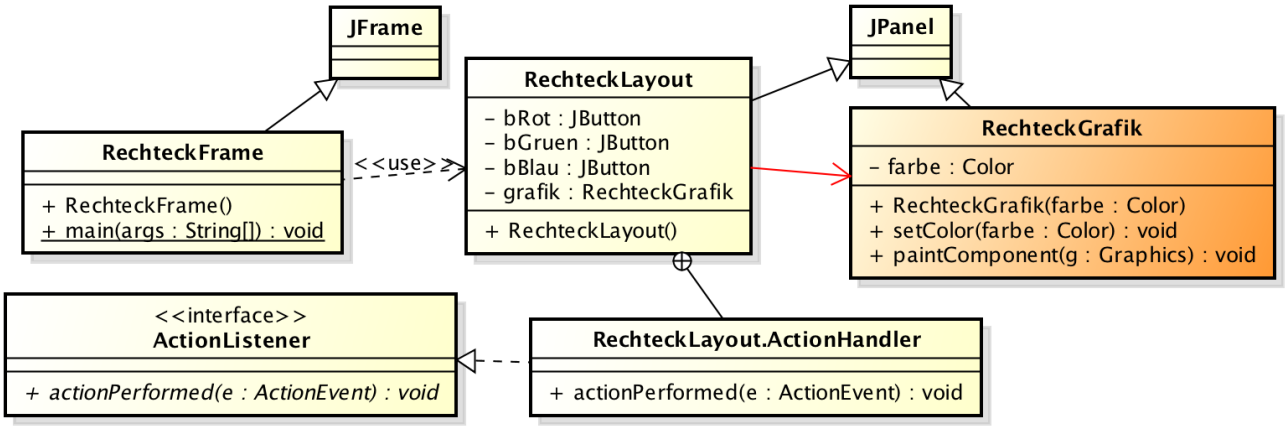
Analysiere die unten gegebene Aufgabenstellung bezüglich Anpassungen der Grafik, die auf Grund von inneren Eigenschaften erfolgen, bzw. Anpassungen, die von außen ermöglicht werden sollen:

Erstelle einen Grafik, die einen Kreis in der Mitte des Panels zeichnet. Die Größe des Kreises soll mit Hilfe von 2 Buttons verändert werden können (ein Plus-Button, der den Kreis vergrößert, und ein Minus-Button, der den Kreis verkleinert). Die Farbe soll sich je nach Größe ändern: solange die Größe kleiner als ein Viertel der Panel-Breite ist, soll der Kreis rot sein, bis zur Hälfte dann gelb, bis zu Dreiviertel der Panel-Breite soll der Kreis grün sein und ab dann blau.

Attribute als Konfigurationsvariablen

Damit ein Objekt (und auch unsere Grafik ist ein Objekt) seinen Zustand von außen veränderbar machen kann, muss es entsprechende Methoden bereitstellen. Der Zustand eines Objektes wird in Attributen gespeichert. Das Wort Zustand bedeutet in der OOP nichts anderes als "Werte der Attribute". Für die inneren Anpassungen müssen weder Attribute noch Methoden zur Verfügung gestellt werden. Das Objekt hat schon alle Informationen, die es dazu benötigt (jede GUI-Komponente "weiß" über ihre Breite und Höhe Bescheid und kann sie mit den Methoden getHeight() und getWidth() abfragen). Damit ergibt sich folgende Klassenstruktur für das Beispiel:

In der Klasse RechteckGrafik ist nun farbe als Attribut gesetzt und dazu die Methode setFarbe, damit dieses Attribut auch von außen änderbar ist. Die Methode paintComponent(Graphics g) wird überschrieben. In dieser findet nach wie vor das eigentliche Zeichnen statt. Weiters besitzt die Klasse noch einen Konstruktor, mit dem man über den Parameter die Anfangsfarbe setzen kann.



Die Klasse RechteckLayout und insbesondere ihre innere Ereignissteuerungsklasse ActionHandler benutzen nun die Klasse RechteckGrafik, um immer bei einem Button-Klick die Farbe zu ändern. Dazu wird ein Objekt der Klasse RechteckGrafik als Attribut im Layout gespeichert und kann damit auch gezielt platziert werden.

Übung 2:

Erstelle für die Aufgabenstellung aus Übung 1 ein ähnliches UML-Klassendiagramm.

Implementierung der Grafik

Um die Grafik zu implementieren, muss man nun abgesehen vom Attribut und der setter-Methode noch 2 Sachen beachten:

- Die Grafik, die in der paintComponent-Methode gezeichnet wird, benutzt nun das Attribut und die inneren Eigenschaften, um je nach Werten ein anderes Aussehen zu realisieren.
- Bei Änderung eines Attributes in einer setter-Methode muss die Grafik aufgefordert werden sich neu zu zeichnen. Das darf niemals direkt durch das Aufrufen der paintComponent-Methode geschehen. Statt dessen wird mit der geerbten Methode repaint() die Komponente (= das Panel) aufgefordert, das nochmalige Aufrufen der paintComponent-Methode zu veranlassen.

```
import javax.swing.JPanel;
import java.awt.*;

public class RechteckGrafik extends JPanel {
  private Color farbe;

  public RechteckGrafik(Color farbe) {
    this.farbe = farbe;
  }

  public void setFarbe(Color farbe) {
    this.farbe = farbe;
    this.repaint();
  }

  @Override
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    int breite = this.getWidth();
    int hoehe = this.getHeight();

    g.setColor(this.farbe);

    g.fillRect(breite/4, hoehe/4, breite/2, hoehe/2);
  }
}
```

Attribut farbe zum Speichern der Farbinformation

Methode zum Ändern der Farbinformation

Aufforderung zum neuerlichen Aufruf von paintComponent

Abfragen der "inneren" Eigenschaften

Verwenden des Attributes in der Grafik

Verwenden der "inneren" Eigenschaften zum Zeichnen der Grafik

Übung 3:

Implementiere die Grafik-Klasse aus Übung 2. Zum Zeichnen des Kreises kannst du folgende Methode der Klasse Graphics verwenden:

`void`	`fillOval(int x, int y, int width, int height)` Fills an oval bounded by the specified rectangle with the current color.

Verwenden der Grafik-Klasse im Layout-Panel

Die Grafik-Klasse kann nun wie jede andere GUI-Komponente im Layout-Panel verwendet werden. Die Methode `setFarbe` zum Konfigurieren der Grafik wird innerhalb der Ereignissteuerung verwendet.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class RechteckLayout extends JPanel {
  private JButton bRot, bGruen, bBlau;
  private RechteckGrafik grafik;

  public RechteckLayout() {
    this.setLayout(new BorderLayout());

    grafik = new RechteckGrafik(Color.RED);
    this.add(grafik);

    JPanel unten = new JPanel(new GridLayout(1,3));
    bRot = new JButton("Rot");
    unten.add(bRot);
    bGruen = new JButton("Grün");
    unten.add(bGruen);
    bBlau = new JButton("Blau");
    unten.add(bBlau);
    this.add(unten, BorderLayout.PAGE_END);

    ActionHandler ah = new ActionHandler();
    bRot.addActionListener(ah);
    bGruen.addActionListener(ah);
    bBlau.addActionListener(ah);
  }

  private class ActionHandler implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
      if(e.getSource() == bRot) {
        grafik.setFarbe(Color.RED);
      } else if(e.getSource() == bGruen) {
        grafik.setFarbe(Color.GREEN);
      } else if(e.getSource() == bBlau) {
        grafik.setFarbe(Color.BLUE);
      }
    }
  }
}
```

Erstellen des Grafik-Objektes und Hinzufügen zum Layout

Konfigurieren der Grafik mit Hilfe der setter-Methode

Übung 4:

Implementiere die Layout-Klasse für die Aufgabenstellung aus Übung 1 unter Verwendung der Grafik-Klasse aus Übung 3.