Mit diesem Aufgabenblatt erhalten Sie einen Einstieg in die GUI[1] -Programmierung mit Java AWT und Swing.

1 Eine Applikation mit GUI Erstellen

1.1 Hello World

Bis jetzt haben Sie Java Programme geschrieben, die mit der Kommandozeile ausgeführt werden. Nun soll eine Applikation erstellt werden, die in einem eigenen Fenster läuft. Erstellen Sie dazu eine Klasse, die von `JFrame` abgeleitet ist. In `main()` können Sie dann eine Instanz von dieser Klasse erstellen. Beachten Sie das `main()` eine Methode der selben Klasse sein kann:

```
import javax.swing.JFrame;
public class IhreKlasse extends JFrame {
  public static void main(String[] args) {
    JFrame f = new IhreKlasse();
  }
}
```

Nun können Sie verschieden Methoden auf dem JFrame-Objekt aufrufen, um das Fenster Ihren Bedürfnissen anzupassen. **Achtung:** das Fenster wird erst sichtbar, wenn Sie `setVisible(true)` aufrufen.

Tipp: Wenn Sie das Fenster schliessen, wird die Anwendung nicht automatisch beendet, was Sie aber mit `setDefaultCloseOperation(EXIT_ON_CLOSE)` ändern können. Die Fenstergrösse können Sie mit `setSize(600,400)` setzen. Des Weiteren können Sie diese Methoden auch von der Konstruktor-Methode aus aufrufen.

1.2 Komponenten Hinzufügen

Nun können Sie mittels der `add()`-Methode Komponenten zu Ihrem JFrame-Objekt hinzufügen. `add(new Button("Test"))` fügt z.B. einen Button mit der Anschrift "Test" hinzu, der das ganze Fenster ausfüllt.

Um Komponenten gezielter in das Fenster einzufügen, können Sie verschiedene Layouts wählen. `setLayout(new BorderLayout())` teilt z.B. das Fenster in fünf Bereiche auf {NORTH, EAST, SOUTH, WEST, CENTER}. `add(new Button("Test"), BorderLayout.NORTH)` würde nun den Button in den oberen Bereich des Fensters einfügen.

Wählen Sie das `BorderLayout` und fügen Sie in die Bereiche {NORTH, EAST, SOUTH, WEST} jeweils einen Button mit der Anschrift {"Kreis", "Dreieck", "Quadrat", "Ellipse"} ein. CENTER können Sie leer lassen.

1.3 Fensterbereiche weiter Aufteilen

Nun kommen wir zum `CENTER`-Bereich im `BorderLayout`. Diesen Bereich möchten wir weiter aufteilen. Zu diesem Zweck fügen Sie eine `JPanel` Komponente in den `CENTER`-Bereich ein.

[1] graphical user interface

Für die `JPanel` Komponente können Sie dann ein neues Layout wählen. Mit dieser Methode können Sie das Fenster beliebig aufteilen. Wählen Sie für JPanel ein `OverlayLayout`, das erlaubt Komponenten übereinander zu stapeln, was in späteren Teilaufgaben wichtig wird.

```
JPanel p = new JPanel();
p.setLayout(new OverlayLayout(p));
```

1.4 Objekte Zeichnen

Die Klasse `Canvas` erlaubt es Ihnen, geometrische Figuren zu zeichnen (java.awt.Canvas). Erstellen Sie eine Klasse, die von `Canvas` erbt und überschreiben Sie die Methode `paint(Graphics g)`. Mit `g.drawOval(...)` können Sie z.B. eine Ellipse zeichnen. Damit Sie die gezeichneten Objekte sehen könne, müssen Sie dass `Canvas` Objekt zu Ihrem `JPanel` Objekt via `add(...)` hinzufügen. Zeichnen Sie einen roten Kreis, der in der Mitte des Fensters angezeigt wird.

Tipp: Die Mitte des Fensters kann mit `getWidth()` und `getHeight()` ermittelt werden.

Unterlegen Sie das eben gezeichnete Objekt mit einem Hintergrundbild. Ein Bild kann wie folgt geladen werden:

```
ImageIO.read(new File("background.jpg"))
```

Das geladene Bild kann dann via `g.drawImage(...)` angezeigt werden. Achten Sie darauf, dass das Bild entsprechend der Fenstergrösse skaliert wird.

Tipps: Damit sie das Hintegrundbild und Ihre geometrische Figur sehen, zeichnen Sie zuerst das Hintergrundbild und erst dann die Figur. Bevor Sie mit Methoden wie `g.drawOval(...)` zeichnen, können versch. Parameter gesetzt werden, zum Beispiel:

- die Farbe: `g.setColor(Color.RED)`,
- die Strichbreite: `g.setStroke(new BasicStroke(3))`,

Achtung: Damit Ihnen `setStroke(...)` zur Verfügung steht, müssen Sie `g` vorher in ein `Graphics2D`-Objekt casten.

1.5 Action Listener

Als nächster Schritt sollen Sie dem Button, der mit "Kreis" gekennzeichnet, Funktionalität hinzufügen. Dazu müssen Sie einen sogenanntes ActionListener-Objekt mit der Methode `b.addActionListener(...)` hinzufügen, wobei `b` die Instanz des betreffenden Buttons ist.

Als ActionListener-Objekt soll das Objekt übergeben werden, das Sie in der vorherigen Teilaufgabe erstellt haben; also das Objekt der Klasse, welche von `Canvas` erbt und die geometrische Figur darstellt. Sie müssen aber erst das Interface `ActionListener` implementieren. Erweiteren Sie Ihre Klasse dazu wie folgt:

```
public class IhreGeomFig extends Canvas implements ActionListener {
```

und fügen Sie folgende Methode hinzu:

```
public void actionPerformed(ActionEvent arg0) {...}
```

Diese Methode wird jedes Mal aufgerufen, wenn der entsprechende Button geklickt wird. Programmieren Sie nun folgende Funktionalität: wenn der Button "Kreis" das erste Mal geklickt wird verschwindet die gezeichnete Figur, beim nächsten Klick kommt sie wieder zum Vorschein, dann verschwindet sie wieder, u.s.w.

Tipp: `setVisible(true/false)` kann auch von `actionPerformed(...)` aus aufgerufen werden.

1.6 Observer Pattern

Um Ihre GUI-Applikation abzurunden, programmieren Sie folgendes: Wenn einer der vier Buttons geklickt wird, dann soll die entsprechende geometrische Figur dargestellt werden. Das heisst, Sie müssen als erstes drei weitere Klassen für versch. Figuren erstellen, so dass Sie am Schluss die Figuren Kreis, Dreieck, Quadrat und Ellipse darstellen können.

Wenn nun ein Button geklickt wird, soll die entsprechende Figur in der Mitte dargestellt werden (`setVisible(true)`) und alle anderen Figuren sollen verschwinden (`setVisible(false)`). Dieses Muster lässt sich am einfachsten mit dem Observer-Pattern[2] implementieren.
Schreiben Sie dazu eine Klasse `CanvasModel` nach folgendem Muster:

```
public class CanvasModel implements ActionListener {

  private Collection<DrawLayer> listeners = new LinkedList<DrawLayer>();

  public void actionPerformed(ActionEvent arg0) {
    notify(arg0.getSource());
  }

  public void notify(Object o) {
    for(DrawLayer l : listeners) {
      l.update(o);
    }
  }

  public void addListener(DrawLayer c) {
    listeners.add(c);
  }

  public void removeListener(DrawLayer c) {
    listeners.remove(c);
  }
}
```

Mit `addListener(...)/removeListener(...)` können Sie Listener hinzufügen, bezw., wieder entfernen. Die Idee ist, das Sie hier die Objekte übergeben, welche die versch. geometrischen Figuren zeichnen. Wenn ein Button geklickt wird, dann wird `actionPerformed(...)` aufgerufen, und der Event (welcher Button geklickt wurde) wird an alle Listeners via `notify(...)` weitergeleitet, die dann entsprechend reagieren können.

Tipp: Damit alles wie oben beschrieben funktionieren kann, müssen sie die Methode `update(...)` für alle geometrischen Figuren implementieren.

[2] http://en.wikipedia.org/wiki/Observer_pattern