Übung 4

Interaktive Elemente, GUI-Programmierung

1. Schreiben Sie ein Programm (wahlweise als Applet oder Applikation, oder beides), das die folgenden interaktiven Elemente enthält:

   (a) Einen Button,

   (b) ein Eingabetextfeld,

   (c) ein Aufklapp-Menü mit mindestens zwei Elementen,

   (d) zwei Radio-Checkboxen,

   (e) zwei normale (unabhängig auswählbare) Checkboxen.

   Veränderungen in den Bedienelementen sollen (graphisch) im unteren Bereich des Fensters angezeigt werden.

   Bei Klick auf den „Schließen"-Knopf soll das Programm beendet werden.

   Prinzipiell könnten bei jedem Klick auf den Button alle Elemente ausgelesen werden, wenn sie als „globale Variablen" innerhalb der Klasse implementiert sind. Interessanter ist jedoch, die bei jedem Element auftretenen Events direkt darzustellen, dies wurde in der Musterlösung ***`MultiGUI.java`*** *so eingebaut.*

   MultiGUI.java:

```
// Dieses Programm soll verschiedene Java-Bedienelemente in einem Fenster
// sichtbar machen, und Aktionen ausgeben.

// In der Aufgabenstellung steht zwar, dass immer auf den BUTTON
// geklickt werden soll, um die anderen Elemente auszulesen, aber
// es ist interessanter, wenn bei jedem Element direkt bei dessen
// Veränderung eine Aktion ausgelöst wird.

import java.awt.*; // AWT-Grafikklassen importieren
import java.awt.event.*; // AWT-Eventklassen importieren
import java.applet.*; // Enthält die Definition der Applet-Klasse

// MultiGUI ist abgeleitet von Klasse Applet, und enthält damit
// auch Funktionen wie add(), init(), paint(Graphics g)...
// ActionListener ist für Buttons, ItemListener für Menüs
// und Checkboxen zuständig
public class MultiGUI extends Applet implements ActionListener,ItemListener {

 // Bei Applets werden die folgenden Funktionen aufgerufen:
 // paint(Graphics g) - immer, wenn das Browser-Fenster neu gezeichnet wird
 // start() - beim Start des Applet
 // init() - Kurz nach dem Laden des Applet
```

```
// stop() - Wenn auf eine andere Webseite verzweigt wird, wird das Applet
// angehalten.

// init(): GUI aufbauen, WENN als Applet gestartet.
public void init() { showGUI(); }

// main() wird in der Variante "Standalone-Anwendung" verwendet.
// Wenn das Programm als Applet läuft (im Browser) wird main() ignoriert.
public static void main(String[] args){
 Frame frame = new Frame("MultiGUI"); // Fenster
 frame.setSize(800,600);
 // Applet erzeugen
 MultiGUI applet = new MultiGUI();
 // in applet haben wir nun alle Methoden zur Verfügung
 // Applet mit Fenster verbinden
 frame.add(applet);
 // Wir wollen das Fenster auch schließen können.
 frame.addWindowListener(new WindowAdapter() {
  public void windowClosing(WindowEvent e){ System.exit(0); }
 });

 // Und jetzt sind wir da, wo wir mit Browser auch gewesen wären. :-)
 applet.init();  // Siehe oben.
 applet.start(); // Standard-Methode aus Applet, damit es losläuft.
 frame.setVisible(true); // Fenster sichtbar machen.
}

public void showGUI(){
 // Layout der Elemente: Positionierung pixelgenau per setBounds()
 setLayout(null); // Freies/manuelles Layout

 // Zeichensatz für diverse Elemente
 Font font = new Font("SansSerif", Font.BOLD, 16); // Schrift

 // Die Bedienelemente

 Button button = new Button("Klick"); // Ein Knopf-Objekt
 button.setFont(font); // Schriftart für Knopf setzen.
 button.setBounds(10,10,80,30); // Position und Größe des Knopfes
 add(button); // Und den Knopf dem Fenster oder Applet hinzufügen
 button.addActionListener(this); // Aktuelle Klasse enthält die
                                 // ActionListener-Funktionen

 TextField textfield = new TextField("Textfeld", 20); // Ein Textfeld
 textfield.setFont(font);
 textfield.setBounds(100,10, 280, 30);
 add(textfield); // Mit Applet verknüpfen
 textfield.addActionListener(this);

 Choice choice = new Choice(); // Ein Aufklapp-Menü
 choice.setFont(font);
 choice.setBounds(10,80,260,60);
 choice.add("erster Menüeintrag");
 choice.add("zweiter Menüeintrag");
 choice.add("dritter Menüeintrag");
```

```
  add(choice);
  choice.addItemListener(this);

  CheckboxGroup radio = new CheckboxGroup(); // "Radio"-Checkboxen
  Checkbox checkbox1 = new Checkbox("Eins", radio, true);
  Checkbox checkbox2 = new Checkbox("Zwei", radio, true);
  checkbox1.setFont(font); checkbox2.setFont(font);
  checkbox1.setBounds(10,150,200,60); checkbox2.setBounds(250,150,200,60);
  add(checkbox1); add(checkbox2);
  checkbox1.addItemListener(this); checkbox2.addItemListener(this);

  Checkbox c1 = new Checkbox("Checkbox 1"); // Unabhängige Checkboxen
  Checkbox c2 = new Checkbox("Checkbox 2");
  c1.setFont(font); c2.setFont(font);
  c1.setBounds(10,200,200,60);
  c2.setBounds(250,200,200,60);
  add(c1); add(c2);
  c1.addItemListener(this); c2.addItemListener(this);
 }

// Eine eigene Funktion, die an Stelle y des Fensters einen Text zeichnet
 public void printText(String text, int y) {
   Graphics g = getGraphics();
   g.clearRect(10,y-40,780,80); // Alten Text löschen (Rechteck)
   g.drawString(text, 10, y);
 }

// Hier stehen die Methoden aus "implements ActionListener". Es gibt
// nur eine.
 public void actionPerformed(ActionEvent e) {
  printText(e.toString(), 350);
 }

// Hier stehen die Methoden aus "implements ItemListener". Es gibt nur eine.
 public void itemStateChanged(ItemEvent e) {
  printText(e.toString(), 350);
 }

}
```

---

2. Schreiben Sie ein GUI, das

   (a) zwei Texteingabefelder mit Beschriftung „Zahl 1“ und „Zahl 2“,

   (b) einen Knopf mit Beschriftung „Addieren“

   enthält. Bei Druck auf den Knopf sollen die Bedienelemente verschwinden, Zahl 1 und Zahl 2 addiert und das Ergebnis im Fenster dargestellt werden.

   Das Programm kann als Standalone Java-Anwendung, als Applet, oder beides gleichzeitig realisiert werden.

   Es ist komplizierter, als es sich zunächst anhört, da die Eingabe-Knöpfe während der Darstellung des Ergebnisses „verschwinden“ sollen. Im Beispiel ***`Addieren.java`*** *wurde dies durch die Funktion* ***`setEnabled(false)`*** *gelöst, so dass keine Eingaben mehr stattfinden können, während das Ergebnisfenster noch auf dem Bildschirm ist.*

Ein Problem kann bei diesem Programm aber noch auftreten:
die Konvertierungsfunktion ***`Integer.valueOf(zahlenfeld.getText()).intValue()`***
kann „abstürzen“, wenn keine Zahl im Feld steht, dann erscheint das Ergebnisfenster gar nicht. Demnächst werden wir lernen, wie Fehler in Programmen abgefangen und selbst behandelt werden können.

Addieren.java:

```
import java.awt.*; // AWT-Grafikklassen importieren
import java.awt.event.*; // AWT-Eventklassen importieren
import java.applet.*; // Enthält die Definition der Applet-Klasse

public class Addieren extends Applet {

 // Da wir auf die Textfelder und Button von mehreren Methoden aus
 // zugreifen wollen, definieren wir sie direkt hier in der Klasse.
 private final static Label label1 = new Label("Zahl 1:");
 private final static Label label2 = new Label("Zahl 2:");
 private final static TextField zahl1 = new TextField("", 20); // 1. Textfeld
 private final static TextField zahl2 = new TextField("", 20); // 2. Textfeld
 private final static Button addieren = new Button("Addieren"); // Knopf-Objekt
 private final static Font font = new Font("SansSerif", Font.BOLD, 16);

 // init(): GUI aufbauen, WENN als Applet gestartet.
 public void init() {
  // Grid Layout der Elemente
  setLayout(new GridLayout(3,2));
  // Zeichensatz
  setFont(font);
  add(label1); add(zahl1);
  add(label2); add(zahl2);
  add(addieren);
  addieren.addMouseListener(new MouseAdapter(){
   public void mouseClicked(MouseEvent e){
    // Ergebnis zeigen
    showResult();
   }});
 }

 // main() wird in der Variante "Standalone-Anwendung" verwendet.
 // Wenn das Programm als Applet läuft (im Browser) wird main() ignoriert.
 public static void main(String[] args){
  Frame frame = new Frame("Addieren"); // Fenster
  frame.setSize(400,150);
  // Applet erzeugen
  Addieren applet = new Addieren();
  // Applet mit Fenster verbinden
  frame.add(applet);
  // Wir wollen das Fenster auch schließen können.
  frame.addWindowListener(new WindowAdapter() {
   public void windowClosing(WindowEvent e){ System.exit(0); }
  });
  applet.init();
  applet.start();
  frame.setVisible(true); // Fenster sichtbar machen.
```

```
 }

// Eine eigene Funktion, die das Ergebnis in einem Fenster anzeigt.
 public void showResult() {
   // Achtung: Das Umwandeln von String nach int kann schiefgehen. Wir lernen
   // später eine Möglichkeit kennen, Fehler abzufragen und selbst zu behandeln.
   int result = Integer.valueOf(zahl1.getText()).intValue() +
                Integer.valueOf(zahl2.getText()).intValue();
   // Erst mal die Knöpfe im Hauptfenster deaktivieren.
   zahl1.setEnabled(false); zahl2.setEnabled(false); addieren.setEnabled(false);
   final Frame frame = new Frame("Ergebnis");
   frame.setSize(200,100);
   frame.setFont(font);
   Button ergebnis = new Button(Integer.toString(result));
   frame.add(ergebnis);
   // Bei Klick auf Button, zurück zum Hauptfenster.
   ergebnis.addMouseListener(new MouseAdapter(){
    public void mouseClicked(MouseEvent e){
     frame.dispose(); // Fenster schließen
     // Die Knöpfe im GUI wieder aktivieren.
     zahl1.setEnabled(true); zahl2.setEnabled(true); addieren.setEnabled(true);
    }});
   // Bei Klick auf Schließen, zurück zum Hauptfenster.
   frame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e){
     frame.dispose(); // Fenster schließen
     // Die Knöpfe im GUI wieder aktivieren.
     zahl1.setEnabled(true); zahl2.setEnabled(true); addieren.setEnabled(true);
    } });
   frame.setVisible(true);
 }

}
```

---

3. Bilden Sie (nur!) die Oberfläche aus der vorigen Aufgabe mit HTML ab. Wie könnten hierbei die Aktionen zur Auswertung der Bedienelemente implementiert werden? (→ Web-Teil der Vorlesung)

Addieren-in-HTML.html:

```
<HTML>
 <HEAD><TITLE>Ein "Additions"-Programm in HTML</TITLE></HEAD>
 <BODY>

 <FORM NAME=Formular>

  Geben Sie bitte zwei Zahlen ein:

  <input type=text size=4 NAME=Eingabe1> +
  <input type=text size=4 NAME=Eingabe2>

  <!-- Bei Klick passiert ... noch nichts. -->
  <input type=button value=" = ">

 </FORM>
```

```
 </BODY>
</HTML>
```
