Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 2 SoSe 2024
Fakultät für Angewandte Informatik
Institut für Informatik
Prof R. Lorenz, L. Petrak, J. Linne, V. Le Claire, P. Schalk 07.06.2024

Lösungsvorschlag zu Übungsblatt 6

Abgabe: 17.06.2024, 10:00 Uhr (im Digicampus via VIPS: .java-Dateien für Code, .uxf für UML, .pdf für alles andere)

- Dieses Übungsblatt muss im Team abgegeben werden (Einzelabgaben sind nicht erlaubt!).
- Die **Zeitangaben** geben zur Orientierung an, wie viel Zeit für eine Aufgabe später in der Klausur vorgesehen wäre; gehen Sie davon aus, dass Sie zum jetzigen Zeitpunkt wesentlich länger brauchen und die angegebene Zeit erst nach ausreichender Übung erreichen.

* leichte Aufgabe / ** mittelschwere Aufgabe / *** schwere Aufgabe

Aufgabe 21 ** *(Fenster ohne Ereignisbehandlung, 30 Minuten)*

In dieser Aufgabe sollen Sie die grundlegenden Elemente von Fenstern in Java näher kennen lernen. Fügen Sie jeder Fensterklasse auch eine `main`-Methode hinzu, mit der eine Instanz des Fensters erzeugt wird.

a) *(*, Eingabe mit Textfeld, 8 Minuten)*
Implementieren Sie ein Hauptanwendungsfenster nach folgenden Vorgaben:

- Der Titel des Fensters soll `My First Java Frame` sein.
- Beim Klicken des Fensterkreuzes soll das Fenster geschlossen und das Programm beendet werden.
- Das Fenster soll 400×100 Pixel groß sein.
- Im Zentrum des Fensters soll ein Eingabefeld sein, das zu Beginn den Inhalt `"Text..."` hat.
- Links und rechts von diesem Eingabefeld sollen jeweils Buttons mit dem Text `Left` bzw. `Right` sein. Unterhalb des Eingabefelds soll ein unveränderbarer Text `South` stehen, der horizontal zentriert ist.

Lösung:

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import java.awt.BorderLayout;

public class MyJavaWindow extends JFrame {
    public MyJavaWindow() {
        super("My First Java Frame");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(400, 100);

        add(new JButton("Left"), BorderLayout.WEST);
        add(new JButton("Right"), BorderLayout.EAST);
```

```
        add(new JTextField("Text..."), BorderLayout.CENTER);
        JLabel label = new JLabel("South");
        label.setHorizontalAlignment(SwingConstants.CENTER);
        add(label, BorderLayout.SOUTH);

        setVisible(true);
    }

    public static void main(String[] args) {
        new MyJavaWindow();
    }
}
```

b) *(**, Farben, 8 Minuten)*
Implementieren Sie ein Hauptanwendungsfenster nach folgenden Vorgaben:

- Der Titel des Fensters soll `Grey Color Scheme` sein.
- Beim Klicken des Fensterkreuzes soll das Fenster geschlossen und das Programm beendet werden.
- Das Fenster soll in einem 16×16 Gitter Panel der Größe 20×20 Pixel enthalten.
- Die Panel sollen einen gleichmäßigen Farbverlauf von Schwarz zu Weiß darstellen.
- Die Panel sollen in jeder Zeile von links nach rechts heller werden.
- Das erste Panel in der ersten Zeile soll den Grauwert 0 (schwarz) haben, das erste Panel in der zweiten Zeile den Grauwert 16 usw., bis das letzte Panel in der letzten Zeile den Grauwert 255 (weiß) hat.

Hinweis: Um die Größe von Komponenten in einem Container mit LayoutManager anzupassen, muss die Methode `setPreferredSize` statt `setSize` verwendet werden.

Lösung:

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridLayout;

public class ColorSchemeFrame extends JFrame {
    public ColorSchemeFrame() {
        super("Grey Color Scheme");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new GridLayout(16, 16));
        for (int i = 0; i < 256; i++) {
            JPanel panel = new JPanel();
            panel.setPreferredSize(new Dimension(20, 20));
            panel.setBackground(new Color(i, i, i));
            add(panel);
        }
        pack();
        setVisible(true);
    }

    public static void main(String[] args) {
        new ColorSchemeFrame();
    }
}
```

c) *(**, Fenster mit Menü, 14 Minuten)*
Implementieren Sie ein Hauptanwendungsfenster nach folgenden Vorgaben:

- Der Titel des Fensters soll `Java Menus` sein.
- Beim Klicken des Fensterkreuzes soll das Fenster geschlossen und das Programm beendet werden.
- Das Fenster soll zwei Menüs haben: `File` und `Edit`.
- Das Menü `File` soll folgende Einträge haben:
  - zwei Einträge `New` und `Save`
  - eine Trennlinie
  - vier Radio-Buttons, von denen immer nur einer ausgewählt werden kann, die den Namen `Option 1` bis `Option 4` haben
- Das Menü `Edit` soll einen Eintrag `Undo` haben.
- Das Fenster soll ein mehrzeiliges Textfeld mit 100 Spalten und 4 Zeilen beinhalten.
- Das Fenster soll genau so groß sein, dass alle enthaltenen Komponenten dargestellt werden.

Hinweis: Recherchieren Sie im API-Eintrag der Klasse für Radio Buttons in einem Menü eine Möglichkeit, wie mehrere Buttons zusammengefasst werden können, sodass immer nur einer ausgewählt sein kann.

Lösung:

```
import javax.swing.ButtonGroup;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.JTextArea;
import java.awt.BorderLayout;

public class JavaMenu extends JFrame {
    public JavaMenu() {
        super("Java Menus");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        JMenuBar menuBar = new JMenuBar();
        JMenu menu = new JMenu("File");
        JMenuItem menuItem = new JMenuItem("New");
        menu.add(menuItem);
        menuItem = new JMenuItem("Save");
        menu.add(menuItem);
        menu.addSeparator();
        ButtonGroup buttonGroup = new ButtonGroup();
        for(int i = 1; i < 5; i++) {
            JRadioButtonMenuItem rbMenuItem = new JRadioButtonMenuItem("Option " + i);
            buttonGroup.add(rbMenuItem);
            menu.add(rbMenuItem);
        }
        menuBar.add(menu);
        menu = new JMenu("Edit");
        menuItem = new JMenuItem("Undo");
        menu.add(menuItem);
        menuBar.add(menu);
        setJMenuBar(menuBar);
        add(new JTextArea(4, 100), BorderLayout.CENTER);
        pack();
        setVisible(true);
    }

    public static void main(String[] args) {
        new JavaMenu();
    }
}
```

Aufgabe 22 * *(Fenster mit Ereignisbehandlung (`ActionEvents`), 32 Minuten)*

Fügen Sie jeder Fensterklasse auch eine `main`-Methode hinzu, mit der eine Instanz des Fensters erzeugt wird.

a) *(*, 9 Minuten)*
Implementieren Sie ein Hauptanwendungsfenster nach folgenden Vorgaben:

- Der Titel des Fensters soll `Color Switcher` sein.
- Beim Klicken des Fensterkreuzes soll das Fenster geschlossen und das Programm beendet werden.
- Das Fenster soll ein Panel enthalten.
- Das Panel enthält zwei Buttons mit den Texten `Green` bzw. `Red`.
- Wird ein Button geklickt, soll sich die Hintergrundfarbe des Panels auf die entsprechende Farbe ändern.
- Verwenden Sie Lambda-Ausdrücke für die Ereignisabhörer.

Lösung:

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Color;

public class ColorSwitcher extends JFrame {
    public ColorSwitcher() {
        super("Color Switcher");
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        JPanel panel = new JPanel();
        JButton button = new JButton("Green");
        button.addActionListener(e -> {
            panel.setBackground(Color.GREEN);
        });
        panel.add(button);
        button = new JButton("Red");
        button.addActionListener(e -> {
            panel.setBackground(Color.RED);
        });
        panel.add(button);
        add(panel);

        pack();
        setVisible(true);
    }

    public static void main(String[] args) {
        new ColorSwitcher();
    }
}
```

b) *(*, 13 Minuten)*
Implementieren Sie ein Hauptanwendungsfenster das wie folgt aussieht:

Das Fenster soll folgendes Verhalten implementieren

- Der Titel des Fensters soll zu Beginn `Counter: 0` sein.
- Beim Klicken des Fensterkreuzes soll das Fenster geschlossen und das Programm beendet werden.
- Wenn der Button `+1` bzw. `+10` geklickt wird, soll die Zähler im Titel des Fensters um 1 bzw. 10 erhöht werden.
- Wenn der Button `Reset` geklickt wird, soll der Zähler auf 0 zurückgesetzt werden.
- Implementieren Sie Ihre Fensterklasse als Ereignisabhörer.

Lösung:

```
import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class CounterWindow extends JFrame implements ActionListener {
    private int count = 0;
    private JButton plusOne;
    private JButton plusTen;
    private JButton reset;

    public CounterWindow() {
        super("Counter: 0");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        plusOne = new JButton("+1");
        plusOne.addActionListener(this);
        add(plusOne);
        plusTen = new JButton("+10");
        plusTen.addActionListener(this);
        add(plusTen);
        reset = new JButton("Reset");
        reset.addActionListener(this);
        add(reset);

        pack();
        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource().equals(plusOne)) {
            count++;
        } else if (e.getSource().equals(plusTen)) {
            count += 10;
        } else if (e.getSource().equals(reset)) {
            count = 0;
        }
        setTitle("Counter: " + count);
    }

    public static void main(String[] args) {
        new CounterWindow();
    }
}
```

c) *(*, 10 Minuten)*
Implementieren Sie ein Hauptanwendungsfenster nach folgenden Vorgaben:

- Der Titel des Fensters soll zu Beginn `Please select a title` sein.
- Beim Klicken des Fensterkreuzes soll das Fenster geschlossen und das Programm beendet werden.
- Das Fenster soll 400×100 Pixel groß sein.
- Das Fenster soll eine editierbare Dropdown-Box enthalten.
- Der Inhalt der Dropdown-Box soll dem Fenster beim Erstellen als Konstruktorparameter in Form eines `String`-Arrays übergeben werden können.
- Wird ein Eintrag ausgewählt oder eingegeben und mit Enter bestätigt, soll der Titel des Fensters entsprechend angepasst werden.
- Implementieren Sie eine eigene Abhörer-Klasse, die sowohl das Fenster als auch die Dropdown-Box als Parameter im Konstruktoraufruf erhält und das beschriebene Verhalten umsetzt.

Lösung:
TitleSelector.java:

```java
import javax.swing.JComboBox;
import javax.swing.JFrame;

public class TitleSelector extends JFrame {

    public TitleSelector(String[] titles) {
        super("Please select a title");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(400, 100);
        JComboBox<String> titleBox = new JComboBox<>(titles);
        titleBox.setEditable(true);
        titleBox.addActionListener(new TitleSelectorListener(this, titleBox));
        add(titleBox);

        setVisible(true);
    }


    public static void main(String[] args) {
        new TitleSelector(new String[]{"Cool Title", "Boring Title", "Exciting title"});
    }
}
```

TitleSelectorListener.java:

```java
import javax.swing.JComboBox;
import javax.swing.JFrame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class TitleSelectorListener implements ActionListener {
    private JFrame parent;
    private JComboBox<String> titleBox;
    public TitleSelectorListener(JFrame parent, JComboBox<String> titleBox) {
        this.parent = parent;
        this.titleBox = titleBox;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource().equals(titleBox)) {
            parent.setTitle(titleBox.getSelectedItem().toString());
        }
    }
}
```

Aufgabe 23 ** *(Fenster mit Ereignisbehandlung (`WindowEvents`, `KeyEvents`, `JOptionPane`, 15 Minuten)*

Implementieren Sie ein Hauptanwendungsfenster nach folgenden Vorgaben. Fügen Sie der Klasse auch eine `main`-Methode hinzu, mit der eine Instanz des Fensters erzeugt wird.

a) *(**, 10 Minuten)*

- Es soll den Titel `Window Event Log` haben, 500 Pixel breit und 500 Pixel hoch sein.
- Im Zentrum soll es ein deaktiviertes mehrzeiliges Textfeld mit 50 Spalten, 5 Zeilen und mit Scrollleiste anzeigen.
- Löst der Benutzer irgendein Fensterereignis aus, so sollen Zeitpunkt und Beschreibung des Ereignisses als Zeichenkette an den Inhalt des Textfelds in einer neuen Zeile angehangen werden.
  Hinweis: Implementieren Sie einen Ereignisabhörer Ihrer Wahl.

b) *(**, 5 Minuten)*

- Zusätzlich soll sich beim Klicken auf das Fensterkreuz ein einfacher modaler Informationsdialog mit der Nachricht `Hit the 'X' key to close the application!` öffnen.
  Hinweis: Benutzen Sie die Klasse `JOptionPane` für den Informationsdialog und implementieren Sie einen Ereignisabhörer Ihrer Wahl.
- Drückt der Benutzer die Taste `X` auf seiner Tastatur, soll die Anwendung beendet werden.
  Hinweis: Implementieren Sie einen Ereignisabhörer Ihrer Wahl.

Lösung:

```
import java.awt.BorderLayout;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.time.LocalDateTime;

import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;
import javax.swing.JOptionPane;

@SuppressWarnings("serial")
public class WindowEventLog extends JFrame implements WindowListener {
    private JTextArea taLog;

    public WindowEventLog() {
        super("Window Event Log");
        setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
        taLog = new JTextArea(5, 50);
        taLog.setEnabled(false);
        add(new JScrollPane(taLog), BorderLayout.CENTER);
        addKeyListener(new KeyAdapter() {
            @Override
            public void keyReleased(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_X) {
                    dispose();
                }
            }
        });
        addWindowListener(this);

        setSize(500, 500);
        setVisible(true);
    }
```

```
    @Override
    public void windowClosing(WindowEvent e) {
        JOptionPane.showMessageDialog(this, "Hit the 'X' key to close the application!");
        add(e);
    }

    @Override
    public void windowActivated(WindowEvent e) {
        add(e);
    }

    @Override
    public void windowDeactivated(WindowEvent e) {
        add(e);
    }

    @Override
    public void windowOpened(WindowEvent e) {
        add(e);
    }

    @Override
    public void windowClosed(WindowEvent e) {
        add(e);
    }

    @Override
    public void windowIconified(WindowEvent e) {
        add(e);
    }

    @Override
    public void windowDeiconified(WindowEvent e) {
        add(e);
    }

    private void add(WindowEvent e) {
        taLog.append(LocalDateTime.now() + ": " + e.paramString() + "\n");
    }

    public static void main(String[] args) {
        new WindowEventLog();
    }
}
```

Aufgabe 24 ** *(Zeichnen in Java, 20 Minuten)*

a) *(**, 17 Minuten)*
Implementieren Sie eine Zeichenfläche der Größe 500 × 500 Pixel, in der ein Rechteck gezeichnet werden kann:

- Das Rechteck soll mit der Maus mit gedrückter Maustaste „aufgezogen“ werden können.
- Der Punkt, an dem zuerst die Maustaste gedrückt worde, ist eine Ecke des Rechtecks.
- Der Punkt, an dem die Maus mit gedrückter Maustaste gerade ist, soll die gegenüberliegende Ecke des Rechtecks sein.
- Das Rechteck soll laufend aktualisiert werden, während die Maus mit gedrückter Maustaste bewegt wird.
- Zusätzlich sollen die Länge und Breite des Rechtecks etwa in der Mitte der oberen bzw. rechten Kante als Zahl angezeigt werden.
- Das Rechteck soll solange bestehen bleiben, bis ein neues Rechteck begonnen wird, dann soll das alte Rechteck verschwinden.
- Achten Sie auch darauf, dass man auch von rechts nach links und von unten nach oben ein Rechteck aufziehen kann!

Hinweis: Um die Größe von Komponenten in einem Container mit LayoutManager anzupassen, muss die Methode `setPreferredSize` statt `setSize` verwendet werden.
Lösung:

```
import javax.swing.JPanel;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class RectanglePanel extends JPanel {
    private int x1, x2, y1, y2;

    public RectanglePanel() {
        super();
        setPreferredSize(new Dimension(500, 500));
        addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                x1 = e.getX();
                y1 = e.getY();
                x2 = x1;
                y2 = y1;
                repaint();
            }
        });

        addMouseMotionListener(new MouseAdapter() {
            @Override
            public void mouseDragged(MouseEvent e) {
                x2 = e.getX();
                y2 = e.getY();
                repaint();
            }
        });
    }
```

```
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        int width = Math.abs(x1 - x2);
        int height = Math.abs(y1 - y2);
        int x = Math.min(x1, x2);
        int y = Math.min(y1, y2);
        g.drawRect(x, y, width, height);
        if (width > 0) {
            g.drawString(Integer.toString(width), x + width / 2, y - 1);
        }
        if (height > 0) {
            g.drawString(Integer.toString(height), x + width + 1, y + height / 2);
        }
    }
}
```

b) *(**, 3 Minuten)*
Implementieren Sie ein Hauptanwendungsfenster, das die Zeichenfläche der vorherigen Teilaufgabe beinhaltet. Beim Klicken des Fensterkreuzes soll das Fenster geschlossen und das Programm beendet werden. Fügen Sie der Klasse auch eine `main`-Methode hinzu, mit der eine Instanz des Fensters erzeugt werden kann.

Lösung:

```
import javax.swing.JFrame;

public class RectangleFrame extends JFrame {
public RectangleFrame() {
        super("Graphic Editor");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        add(new RectanglePanel());
        pack();
        setVisible(true);
    }

    public static void main(String[] args) {
        new RectangleFrame();
    }
}
```