Klausurvorbereitung - Medientechnik

Aufgabe 1: MVC/Swing

a) Erklären Sie das Entwurfsmuster *Adapter*! Wozu wird es benötigt? Konstruieren Sie ein Beispiel.

b) Gegeben sei untenstehende Klasse. Analysieren Sie den Quellcode und skizzieren Sie das GUI-Layout der Anwendung.

```
public class AufgabeSwing extends javax.swing.JFrame {
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton jButton1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JList jList1;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JToggleButton jToggleButton1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JMenuBar jMenuBar1;

    public AufgabeSwing() {
        initComponents();
    }

    private void initComponents() {
        jScrollPane1 = new javax.swing.JScrollPane();
        jButton1 = new javax.swing.JButton();
        jTextField1 = new javax.swing.JTextField();
        jToggleButton1 = new javax.swing.JToggleButton();
        jList1 = new javax.swing.JList();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                System.exit(0);
            }
        });

        jButton1.setText("jButton1");
        getContentPane().add(jButton1, java.awt.BorderLayout.NORTH);

        jTextField1.setText("jTextField1");
        getContentPane().add(jTextField1, java.awt.BorderLayout.SOUTH);

        jToggleButton1.setText("jToggleButton1");
        getContentPane().add(jToggleButton1, java.awt.BorderLayout.WEST);

        jList1.setModel(new javax.swing.AbstractListModel() {
            String[] strings = { "Item1", "Item2" };
            public int getSize() { return strings.length; }
            public Object getElementAt(int i) { return strings[i]; }
        });
        getContentPane().add(jList1, java.awt.BorderLayout.EAST);

        jScrollPane2.setHorizontalScrollBarPolicy
```

```
        (javax.swing.JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        jScrollPane2.setVerticalScrollBarPolicy
        (javax.swing.JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
        jScrollPane2.setViewportView(jTextArea1);

        getContentPane().add(jScrollPane2, java.awt.BorderLayout.CENTER);

        jMenu1.setText("Menu");
        jMenuBar1.add(jMenu1);

        setJMenuBar(jMenuBar1);

        setSize(640, 480);
    }


    public static void main(String args[]) {
        new AufgabeSwing().show();
    }

}
```

c) Gegeben sei untenstehende Klasse. Analysieren Sie den Quellcode und skizzieren Sie das GUI-Layout der Anwendung.
d) Erklären Sie kurz die prinzipielle Funktionsweise der Anwendung.
e) Reorganisieren Sie den Quellcode der Klasse *AufgabeMVC*. Unterteilen Sie sie in 4 Klassen – eine Klasse für das *Model*, eine für den *View*, eine für den *Controller* und eine Hauptklasse zum Starten der Anwendung. Implementieren Sie dabei die Schnittstelle *Observer*. Der Inhalt der Methode *initComponents()* ändert sich nicht und soll daher nicht noch einmal mit notiert werden.

```
public class AufgabeMVC extends javax.swing.JFrame {
    private int zahl = 0;
    private javax.swing.JButton jButton1;
    private javax.swing.JTextField jTextField1;


    public AufgabeMVC() {
        initComponents();
        zahl=zahl*zahl;
        jTextField1.setText(new Integer(zahl).toString());
    }

    private void initComponents() {
        jTextField1 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                System.exit(0);
            }
        });

        jTextField1.setEditable(false);
        getContentPane().add(jTextField1, java.awt.BorderLayout.NORTH);

        jButton1.setText("Calculate");
        addActionListenerToButton();

        getContentPane().add(jButton1, java.awt.BorderLayout.CENTER);
```

```
        pack();
    }

    private void addActionListenerToButton() {
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        zahl +=1;
        zahl=zahl*zahl;
        jTextField1.setText(new Integer(zahl).toString());
    }



    public static void main(String args[]) {
        new AufgabeMVC().show();
    }

}
```

Aufgabe 2: Java 2D

a) Was bedeutet *Antialiasing* und wozu wird es benötigt?
b) Implementieren Sie die Methode *paint(…)* im untenstehenden Quellcode.

```
import …

public class AufgabeJava2D extends Frame {
    public static void main(String[] args) {
        new AufgabeJava2D();
    }

    public AufgabeJava2D() {
        …
        setVisible(true);
    }

    public void paint(Graphics g) {
        //Graphics2D-Objekt erzeugen;


        //Quadrat „myShape“ erzeugen (Position: 200, 200; Seitenlänge: 100)


        //Antialiasing ausschalten


        // “myShape” um Faktor 0.5 skalieren


        //Zeichenfarbe auf blau setzen


        //Strichstärke auf 5.0 setzen


        //“myShape“ zeichnen
```

```
        //Füllfarbe  Rot setzen

        //"myShape" füllen

    }
}
```

Aufgabe 3: VRML
Gegeben sei ein Würfel mit der Seitenlänge 1, der im Koordinatenursprung liegt. Die Würfel soll bei diffuser Beleuchtung mit weißem Licht rein Rot erscheinen. Wenn der Mauszeiger über dem Würfel positioniert wird, soll der Würfel innerhalb von 5 Sekunden kontinuierlich ganz transparent werden und innerhalb von weiteren 5 Sekunden wieder völlig undurchsichtig. Der Vorgang soll sich so lang wiederholen, wie sich der der Mauszeiger über der Kugel befindet. Notieren Sie den VRML-Code für das Beispiel.

Aufgabe 4: Java3D
Gegeben sei folgende Szene:
Eine Lichtquelle kreist im Abstand von 1 vom Koordinatenursprung in der x-y-Ebene um die z-Achse. Eine Umkreisung soll 5 Sekunden dauern und sich ständig wiederholen. Die Änderung des Drehwinkels sei positiv. Im Ursprung rotiert eine Kugel mit dem Radius 0.2 um die z-Achse. Eine Umdrehung soll ebenfalls 5 Sekunden dauern und sich wiederholen. Die Änderung des Drehwinkels der Kugel sei ebenfalls positiv. Die Lichtquelle soll ein rein weißes Punktlicht ohne Abschwächung sein. Das Material der Kugel soll so beschaffen sein, dass es bei diffuser Beleuchtung rein blaues Licht reflektiert.

a) Wie werden in Java3D bei der Definition von Flächen die Vorder- und Rückseiten der Flächen festgelegt?
b) Skizzieren den Inhaltszweig (*content branch*) des virtuellen Universums entsprechend der oben beschriebenen Szene in der Java3D-üblichen Szenengrafnotation. Notieren Sie neben den Knoten aussagekräftige Namen (z. B. „Licht“ oder „Lichttranslation“).
c) Schreiben Sie eine Methode *createSceneGraph(),* die einen *BranchGroup*-Knoten mit dem Inhaltszweig entsprechend Ihrer Überlegungen aus Aufgabe a) zurückgibt. Halten Sie sich dabei nach Möglichkeit an folgende Schrittfolge:
   - Transformknoten definieren/*Capabilities* setzen
   - *Appearances* definieren
   - Primitive Objekte definieren
   - Lichtquelle definieren
   - Interpolator definieren
   - Szenengraf zusammenbauen
   - Wurzelknoten zurückgeben