Yehezkel Sivan, Aaron Purger, Ahmet Özsoy
3BHIT

LIGHTSOUT

Protokoll

Inhaltsverzeichnis

Einführung

Aufgabe

Es ist ein Spiel LightsOut zu programmieren. Das Ziel des Spieles ist es, in einem 5x5 Gitter auf schwarze bzw. orange Felder zu klicken, die umliegenden umzuschalten und damit alle vorhandenen Felder auf Schwarz umzuschalten.

Team

Özsoy Ahmet

Purger Aaron

Sivan Yehezkel

Aufgabeneinteilung

Özsoy

Controller-Klasse

Purger

GUI –und Frame-Klasse

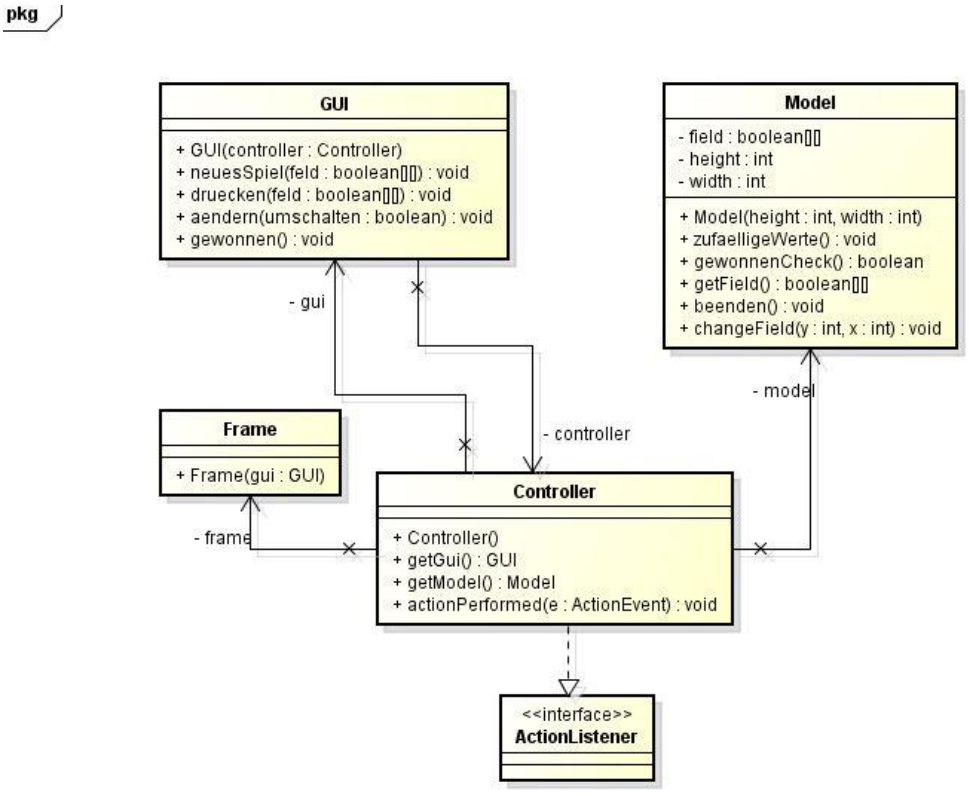
Sivan

Model –und Test-Klasse

Readme

Protokoll

UML-Diagramm von dem Spiel „ LIGHTS-OUT“



Die Klassen

Controller

```
public class Controller implements ActionListener {

    private GUI gui; // GUI
    private Frame frame; // Frame
    private Model model; // Model

    /**
     * Konstruktor
     */
    public Controller(){
        model = new Model(5, 5); // Neues Model erzeugen
        gui = new GUI(this); // Neues GUI erzeugen
        frame = new Frame(gui); // Neues Frame erzeugen
    }
```

Der Controller implementiert den ActionListener, somit hat die Klasse nur zwei wesentliche Aufgaben.
Die Daten zwischen Model und View auszutauschen und auf die geklickten Buttons zu reagieren.

```
/**
 * ActionListener zum reagieren auf das Druecken der Buttons
 */
public void actionPerformed(ActionEvent e) {
    switch (e.getActionCommand()) {
    case "Neues Spiel":
        gui.aendern(true);
        model.zufaelligeWerte();
        gui.neuesSpiel(model.getField());
        break;
    case "Spiel Beenden":
        model.beenden();
        break;
    default:
        int xk = Integer.parseInt(e.getActionCommand()) % 5;
        int yk = Integer.parseInt(e.getActionCommand()) / 5;
        model.changeField(xk, yk);
        gui.druecken(model.getField());
        if (model.gewonnenCheck() == true) {
            gui.aendern(false);
            gui.druecken(model.getField());
            gui.gewonnen();
        }
    }
}
```

Der Button "Neues Spiel" erzeugt ein Feld mit neuen schwarzen und orangen Buttons. Der Button "Spiel beenden" beendet das Spiel und es wird zur Sicherheit nochmals eine Meldung ausgegeben, ob Sie das Spiel wirklich beenden wollen. Die restlichen "default"-Buttons sind die eigentlichen (schwarzen und orangen) Buttons, also jene die dem Spiellogik dienen.

GUI

GUI-Klasse ist für das JPanel das im JFrame ist zuständig.

```
/**
 * Die GUI Klasse zum Spiel LightsOut
 * @author Purger Aaron
 * @version 12.12.2014
 */
public class GUI extends JPanel{

    private JButton[][] buttons;
    private Controller controller;
    private JPanel model,panel;
    private JButton neuesSpiel,beenden;

    /**
     * Konstruktor
     */
    public GUI(Controller controller){
        this.controller = controller;
        this.setLayout(new BorderLayout()); // Ein BorderLayout setzen
        model = new JPanel(); // Neues Panel erzeugen (model)
        panel = new JPanel(); // Neues Panel erzuegen (panel)
        neuesSpiel = new JButton("Neues Spiel"); // Neues Button erzeugen und benennen
        neuesSpiel.setBackground(Color.GREEN); // Farbe des Buttons setzen
        beenden = new JButton("Spiel Beenden"); // Neues Button erzeugen und benennen
        beenden.setBackground(Color.RED);// Farbe des Buttons setzen
        buttons = new JButton[5][5];
        int i = 0; // Den Wert 0 auf die Variable i zuweisen
        for(int j = 0; j<buttons.length; j++){
            for(int h = 0; h<buttons.length; h++){
                buttons[j][h] = new JButton("");
                buttons[j][h].setBorder(BorderFactory.createLineBorder(Color.white));
                buttons[j][h].setActionCommand(i+"");
                buttons[j][h].addActionListener(controller);
                model.add(buttons[j][h]);

                i++;
```

Enthält Attribute: JButton-Array(2-Dimensional), 2 JPanel, 2 JButton, und 1 Controller der Controller-Klasse.

Im Konstruktor wird das Layout des Haupt-Panels gesetzt. Buttons und Panels werden definiert.

JButton-Array wird definiert: Größe des Arrays, Border-Farbe der Buttons, ActionCommand, und ActionListener aus der Controller-Klasse.

Weiteres wird dem Panel für die 2 Buttons neues Spiel und beenden ein Layout definiert und die Buttons gesetzt. Dem Haupt-Panel werden nun beide Panele (Spielfeld-panel, und für die zusätzlichen Buttons) hinzugefügt.

Folge Methoden sind enthalten:

neuesSpiel:

```
/**
 * Eine Methode das das Spiel neu startet
 * @param feld
 */
public void neuesSpiel(boolean[][] feld){
    for(int i = 0; i<buttons.length; i++){
        for(int j = 0; j<buttons.length; j++){
            if(feld[i][j] == true){
                buttons[i][j].setBackground(Color.orange);
            }
            else{
                buttons[i][j].setBackground(Color.black);
            }
        }
    }
```

druecken:

```
/**
 * Eine Methode zum wechseln des Zustands der Buttons
 * @param feld
 * Zu aktivierende felder
 * und zu deaktivierende felder
 */
public void druecken(boolean[][] feld) {
    for (int i = 0; i < feld.length; i++) {
        for (int j = 0; j < feld[i].length; j++) {
            buttons[i][j].setBackground(Color.black);
        }

    }
    for (int i = 0; i < feld.length; i++) {
        for (int j = 0; j < feld[i].length; j++) {
            if (feld[i][j] == true) {
                buttons[i][j].setBackground(Color.orange);
            }
        }

    }
}
```

aendern und gewonnen:

```
/**
 * Eine Methode zum umschalen der Buttons
 * @param status
 */
public void aendern(boolean umschalten) {
    for (int i = 0; i < buttons.length; i++) {
        for (int j = 0; j < buttons[i].length; j++) {
            buttons[i][j].setEnabled(umschalten);
        }
    }
}

/**
 * Diese Methode gibt eine Meldung wenn das Spiel gewonnen wurde
 */
public void gewonnen() {
    JOptionPane.showMessageDialog(null, "Sie haben GEWONNEN ! GRATULATION :D ");
}
```

Die Methode aendern sorgt dafür, dass sich die Farben der Buttons verändern.

Die Methode gewonnen sorgt dafür, dass eine Meldung kommt sobald man gewinnt.

Frame

Diese Klasse ist für das Erstellen eines JFrame Fensters zuständig mit der Größe von 700x700 in der Mitte des Bildschirms.

```
/**
 * Frameklasse zum Spiel LightsOut
 * @author Purger Aaron
 * @version 12.12.2014
 *
 */
public class Frame extends JFrame{

    /**
     * Konstruktor für die Klasse
     */
    public Frame(GUI gui){
        super("Lights Out"); // Titel setzen
        this.add(gui);
        this.setSize(700, 700); // Größe setzen
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(3);
        this.setVisible(true);

    }
}
```

Model

```
import java.io.Closeable;
import java.util.Random;
import javax.swing.JOptionPane;


/**
 * Die Model-Klasse zum Spiel LightsOut
 * @author Yehezkel Sivan
 * @version 12.12.2014
 */
public class Model {
    private boolean field[][]; //Erstellt ein 2D Array
    private int height, width; // hoehe und breite

    /**
     * Standard-Konstruktor der Klasse Model
     * @param height
     * @param width
     */
    public Model(int height, int width){
        this.width = width;
        this.height = height;
        zufaelligeWerte();
    }
```

Diese Klasse ist für die Logik des Programms verantwortlich und wird durch ein 2D-Array welches über den boolean Wert bestimmt ob das Feld ein oder ausgeschalten ist verwaltet.

Die Attribute dieser Klasse:

boolean field [][], Dieses Attribut ist ein 2D-Array

int height und width

Im Standard Konstruktor wird die breite und höhe festgelegt.

In der Klasse Model gibt es folgende Methoden:

zufaelligeWerte:

```
/**
 * Diese Methode setzt zufällige Werte vom 2D Array auf true oder false
 */
public void zufaelligeWerte(){
    field = new boolean[height][width];
    Random r = new Random();
    for(int i=0; i<field.length; i++){
        for(int e=0; e<field[i].length; e++){
            field[i][e] = r.nextBoolean();
        }
```

gewonnenCheck:

```
/**
* Diese Methode ueberprueft ob alle Felder den richtigen Zustand haben
* @return false wenn ein Feld den Zustand true hat, true wenn alle Felder false sind
*/
public boolean gewonnenCheck() {
    for (int i = 0; i < field.length; i++) {
        for (int j = 0; j < field[i].length; j++) {
            if (field[i][j] == true) {
                return false;
            }
        }
    }
    return true;
}
```

getfield:

```
/**
* Gibt das Feld zurueck
* @return the field
*/
public boolean[][] getField() {
    return field;
}
```

beenden:

```
/**
 * Diese Methode beendet das Spiel
 */
public void beenden() {
    int result = JOptionPane.showConfirmDialog(null,"Möchten Sie das Programm beenden?", "Spiel beenden", JOptionPane.YES_NO_OPTION);
    switch(result) {
        case JOptionPane.YES_OPTION:
        System.exit(0); //Aktion(en) bei Klicken auf den "Ja-Button"
  }
}
```

changeField:

```
/**
 * Diese Methode verändert den Wert der gedrückten Buttons und die von seinen umliegenden Buttons.
 * @param y
 * @param x
 */
public void changeField(int y, int x) {
    field[x][y] = !field[x][y];
    if (x>=1) {
        field[x-1][y] = !field[x-1][y];
    }
    if (x<width-1) {
        field[x+1][y] = !field[x+1][y];
    }
    if (y>=1) {
        field[x][y-1] = !field[x][y-1];
    }
    if (y<height-1) {
        field[x][y+1] = !field[x][y+1];
    }
}
```

Test

Diese Klasse ist dafür zuständig um das Programm auszuführen und damit auch zu testen.

```
/**
 * Die Testklasse zum Spiel LightsOut
 * @author Yehezkel Sivan
 * @version 12.12.2014
 */
public class Test {

    /**
     * Testet die Klasse Controller
     * @param args
     */
    public static void main(String[] args) {
        new Controller();
    }
}
```