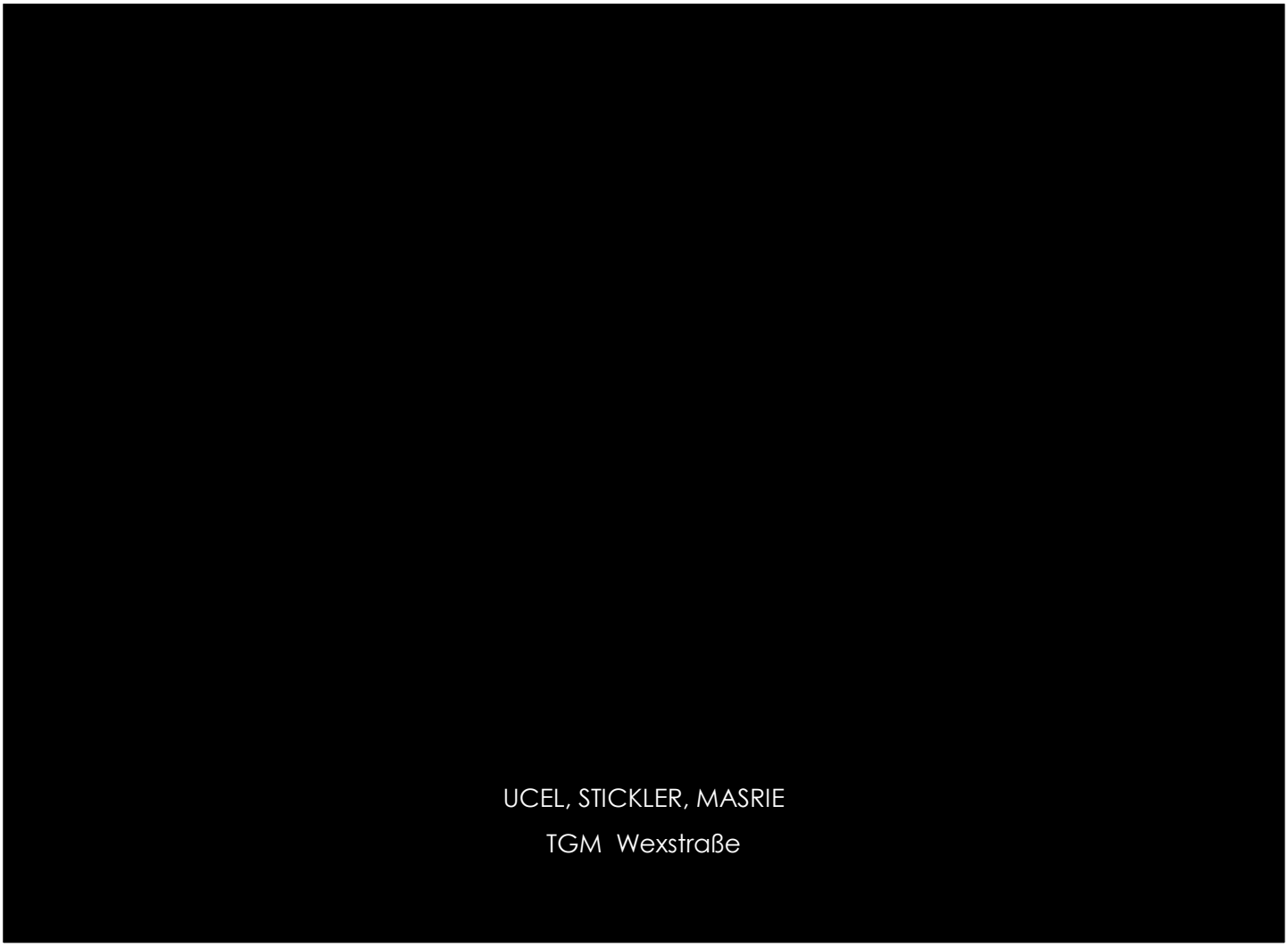
LIGHTS OUT



UCEL, STICKLER, MASRIE

TGM Wexstraße

Inhaltsverzeichnis

Aufgabe

Trainierte Kompetenzen

Die trainierten Kompetenzen beinhalten:

- das Arbeiten mit GitHub
- das Fördern der Teamarbeit
- MVC - Anwendung

Aufgabenstellung

Das Ziel war es ein Spiel mit dem Namen „*Lights Out*“ als Gruppe zu programmieren. Dabei wurden verschiedenen Teile des Programms sowie die Führung des Protokolls auf mehrere Schüler aufgeteilt. Diese mussten von Schülern auf ein Gruppen-Repository in Github hochgeladen werden um zusammen zu arbeiten.

Weiteres muss jede Gruppe:

- Ein ReadMe – File mit der Aufgabenstellung bzw. der Team- und Arbeitsaufteilung erstellen
- Die Aufgabenstellung im Team umsetzen
- Ein Protokoll erstellen

Erweiterte Aufgabenstellung

Da auf Hardcoding verzichtet wurde, ist das Spielfeld anpassbar. So kann der Benutzer verschiedene „Schwierigkeitsstufen“ (wie z.B. 8x8) auswählen. Sowohl die Check – Methoden im Model, als auch die Button – Generierung im View ist anpassbar

Arbeitsaufteilung

Mittels MVC kann das Programm in Model, View und Controller aufgeteilt werden. Weiteres wird dann das Zusammenspiel aller Klassen geprüft und aufgetretene Fehler korrigiert.

Johannes Ucel	Model
Matthias Stickler	GUI
Oliver Masrie	Controller ,Protokoll

Funktion des Spiels

Das Spiel besteht aus einer 5*5 großen Fläche von Lichtern, welche entweder an oder aus sind. Wenn eines der Lichtfelder angeklickt wird ändert sich der An/Aus Zustand des geklickten Feldes, sowie der benachbarten Felder.

Beim Start soll eine zufällige Kombination oder ein vorgegebenes Muster von eingeschalteten Feldern vorkommen. Ziel des Spiels ist es, alle Flächen im ausgeschalteten Zustand zu haben.

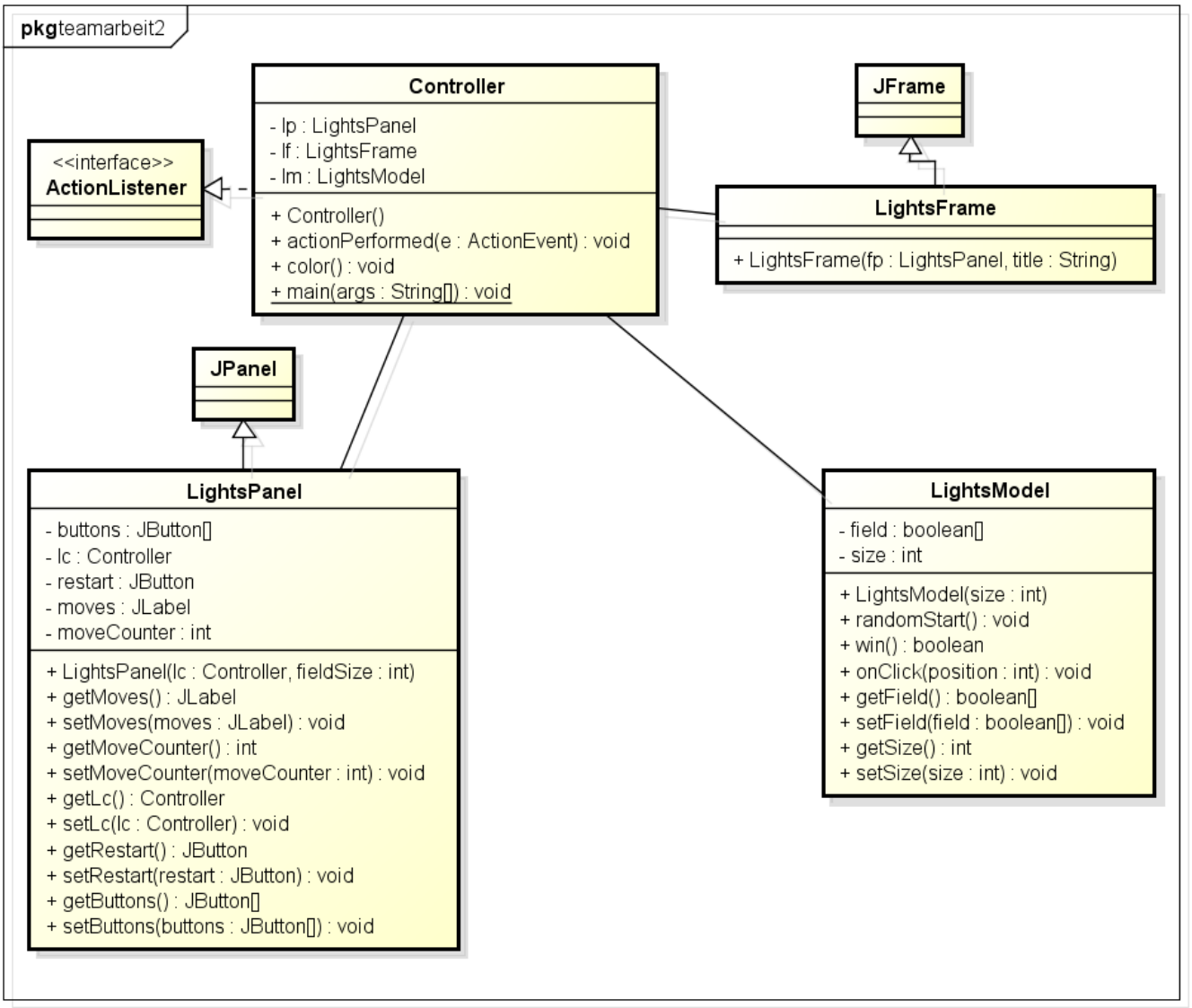
Aufwandsschätzung

	Ucel	Masrie	Stickler	Gesamtstunden
View	0	0	2	2
Model	2	0	0	2
Controller	0	1,5	0	1,5
Testen	1	1	1	3
Protokoll	1	1	1	3

Total: 11,5

Entwurf

Der Entwurf des Programmes wurde als UML-Klassendiagramm erstellt.



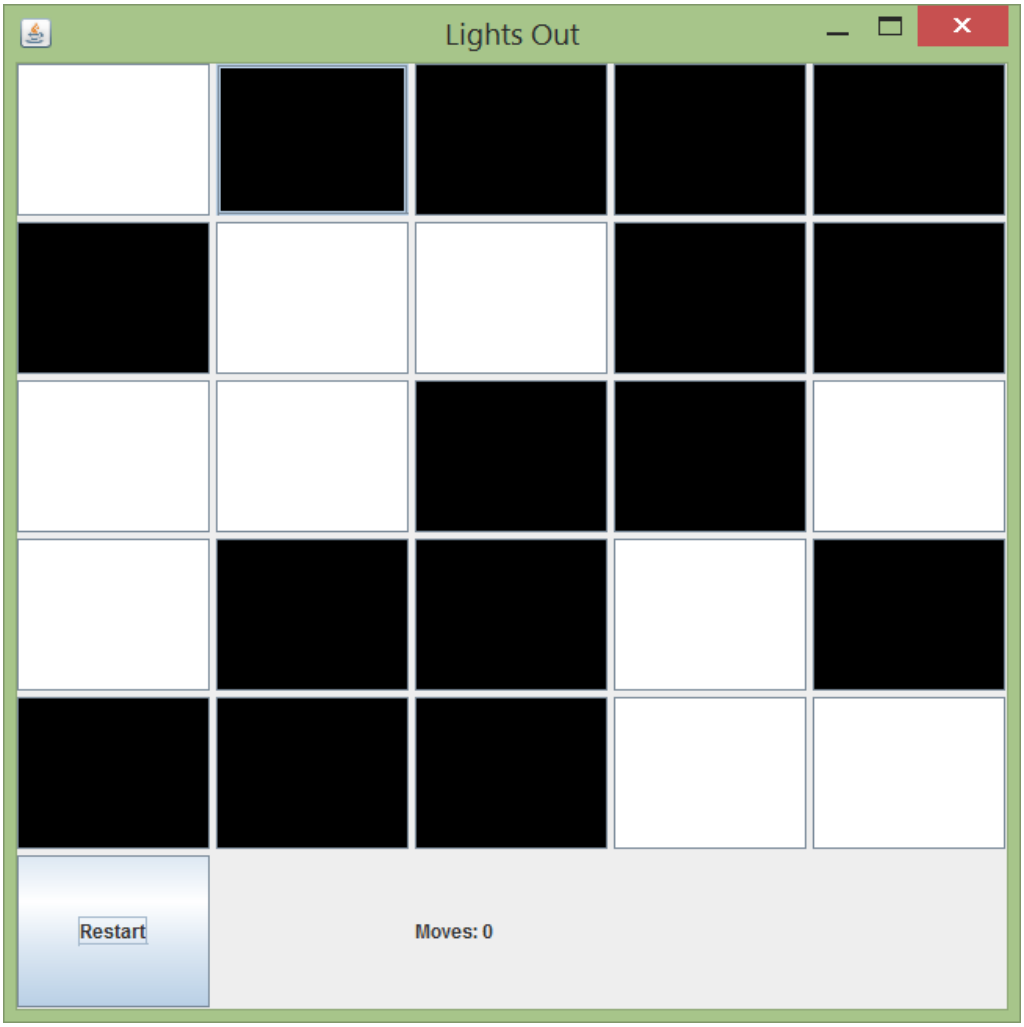
Implementierung

Codiert wurde das Spiel mit der Programmiersprache Java, nach dem Model-View-Controller - Prinzip auch als MVC bekannt. Deswegen wurde das Programm in 4 Teile aufgeteilt

Controller; LightsFrame; LightsModel; LightsPanel

View

Im View wird die grafische Oberfläche für den User realisiert. Hier werden die Buttons (=Spielfeld) erzeugt und das Layout angepasst.



Zusätzlich wird dem User die Möglichkeit gegeben, das Spiel mithilfe des Restart – Buttons neu zu starten. Weiteres gibt es einen Move – Zähler, welcher die Anzahl der Züge mitzählt.

Model

Da hier die erweiterte Aufgabenstellung berücksichtigt wurde, erhält das Model vom Controller die Spielfeldgröße und erstellt dementsprechend ein boolean – Array.

```
/**
 * Standardkonstruktor
 *
 */
public Controller() {
    this.lp = new LightsPanel(this, 5);
```

```
/**
 * Standardkonstruktor
 */
public LightsModel(int size) {
    this.size = size;
    field = new boolean[size];
}
```

Die randomStart – Methode wird verwendet, um bei Programmstart eine gewisse Anzahl an Buttons bzw. Lights auszuschalten. Die Anzahl und die Position dieser Felder wird zufällig mittels Math.random() gewählt.

Bei jedem Klick des Benutzers wird im Model die onClick – Methode aufgerufen. Hier wird die aktuelle Position des Buttons vom Controller übergeben und dementsprechend der Status der Randsteine angepasst (Gegenteil der voherigen Farbe).

Um hier das Überprüfen der Randbereiche auszulassen, wird jeder Zug mit einem try-catch – Block umhüllt.

```
try {
    if (position % tmp != 0)
        field[position - 1] = !field[position - 1];
} catch (ArrayIndexOutOfBoundsException a1) {

}
```

Um einen Gewinner zu ermitteln, gibt es noch die win – Methode, wo überprüft wird, ob alle Felder (bzw. Lichter) aus sind.

Controller

Die Aufgabe des Controllers ist das Verbinden von Model und View. Hier wird mithilfe des ActionListeners auf die Button – Klicks reagiert und die Methoden im Model aufgerufen bzw. die dazugehörigen Aktionen im View durchgeführt. Beim Start des Programs werden Objekte der Model-,Frame und Panel – Klasse erzeugt. Anschließend wird die randomStart – Methode im Model aufgerufen, um zu Beginn eine zufällige Anzahl an Buttons auf den ausgeschalteten Zustand zu stellen.

Installation

Verwendung

Um das Programm zu verwenden, muss auf dem Rechner die JRE funktionsfähig sein. Dann kann die auf GitHub veröffentlichte .jar Datei heruntergeladen werden. Anschließend kann die .jar – Datei gestartet werden und das Spiel gespielt werden

Entwicklungsumgebung

Das Betriebssystem von allen Mitgliedern ist Windows 8.1 und als Entwicklungsumgebung dient Eclipse Luna (4.4)

Tatsächlicher Aufwand

	Ucel	Masrie	Stickler	Gesamtstunden
View	0,1	0	1	1,1
Model	3,5	0	0	3,5
Controller	0,2	2	0	2,2
Testen	1	0,5	0,5	2
Protokoll	1	1	1	3

Total: 11,8

Verwendung von Github

Die individuell geschriebenen Klassen wurden auf Github mittels eines Repository geteilt. Repositories (dt. Verzeichnisse) auf Github können von einzelnen Usern erstellt, geupdated und geteilt werden.

Was ist Github?

GitHub ist ein webbasierter Hosting-Dienst für Software-Entwicklungsprojekte. Hosting Dienste erlauben es Usern ihre Dateien uploaden und andere downloaden, sowie Online-Speicher

Warum Github?

Im Gegensatz zu anderen Open-Source-Hostern wie SourceForge ist auf GitHub nicht das Projekt als Sammlung von Quellcode zentral, sondern der Nutzer mit seinen *Repositories*

Wichtige Commands

Erstellen Sie ein neues Verzeichnis, öffnen Sie es und führen **git init** aus, um ein neues git-Repository anzulegen.

Eine Arbeitskopie kann man erstellen, indem man folgenden Befehl ausführt:

git clone /pfad

Falls ein entferntes Repository verwendet wird, benützt man:

git clone benutzername@host:/pfad/zum/repository

Änderungen kann man vorschlagen oder zum Index hinzufügen mit:

git add <dateiname> bzw **git add ***

Um das lokale Repository mit den neuesten Änderungen zu aktualisieren, verwendet man:

git pull