17.12.2014

LightsOut - Game

CEYLAN Muhammed | KOMON Patrick | STÖGER Michael
3BHIT 14/15

Inhalt

1. Aufgabenstellung

Das Programm

Eigenes Programm erstellen mit dem Namen *Lights Out*.

LightsOutGame basiert auf einem einfachen Konzept. Durch Klicken auf eine Zelle schaltet die Zelle das Licht aus und jedem seiner nächsten Nachbarn. Das Ziel ist, alle Lichter auszuschalten, idealerweise mit einer minimalen Anzahl von Klicks. Funktioniert mit einfachem Binärprinzip.

Aufgaben

Unsere Aufgabe besteht darin dies mit der Programmiersprache Java zu realisieren. Entsprechend wird das durch das Programm Eclipse geschrieben und verwirklicht.

Das Spiel soll über eine interaktive GUI (Graphical User Interface) durchgeführt werden. Eigene Modell-, View- und Controller- Klassen sind enthalten. Verschiedene Klassen haben verschiedene Eigenschaften und Funktionalitäten. Programme basieren auf dem MVC – Prinzip.

Programme sind ausführlich zu kommentieren, dies beinhaltet, dass alle Methoden und manche Codezeilen notwendigerweise kommentiert werden müssen.

Test – Klasse zum Testen der anderen Klassen wurde erstellt.

2. Teams

Alle Teams bestehen aus 3 – 4 Mitgliedern. Alle Teammitglieder haben beim Projekt zu mitarbeiten, da dies eine Teamaufgabe ist und es wurden gerechter Weise die Aufgaben auf die jeweiligen Teammitglieder verteilt. Die verschiedenen Veränderungen wurden, können auf GitHub verfolgt werden.

3. Entwurf

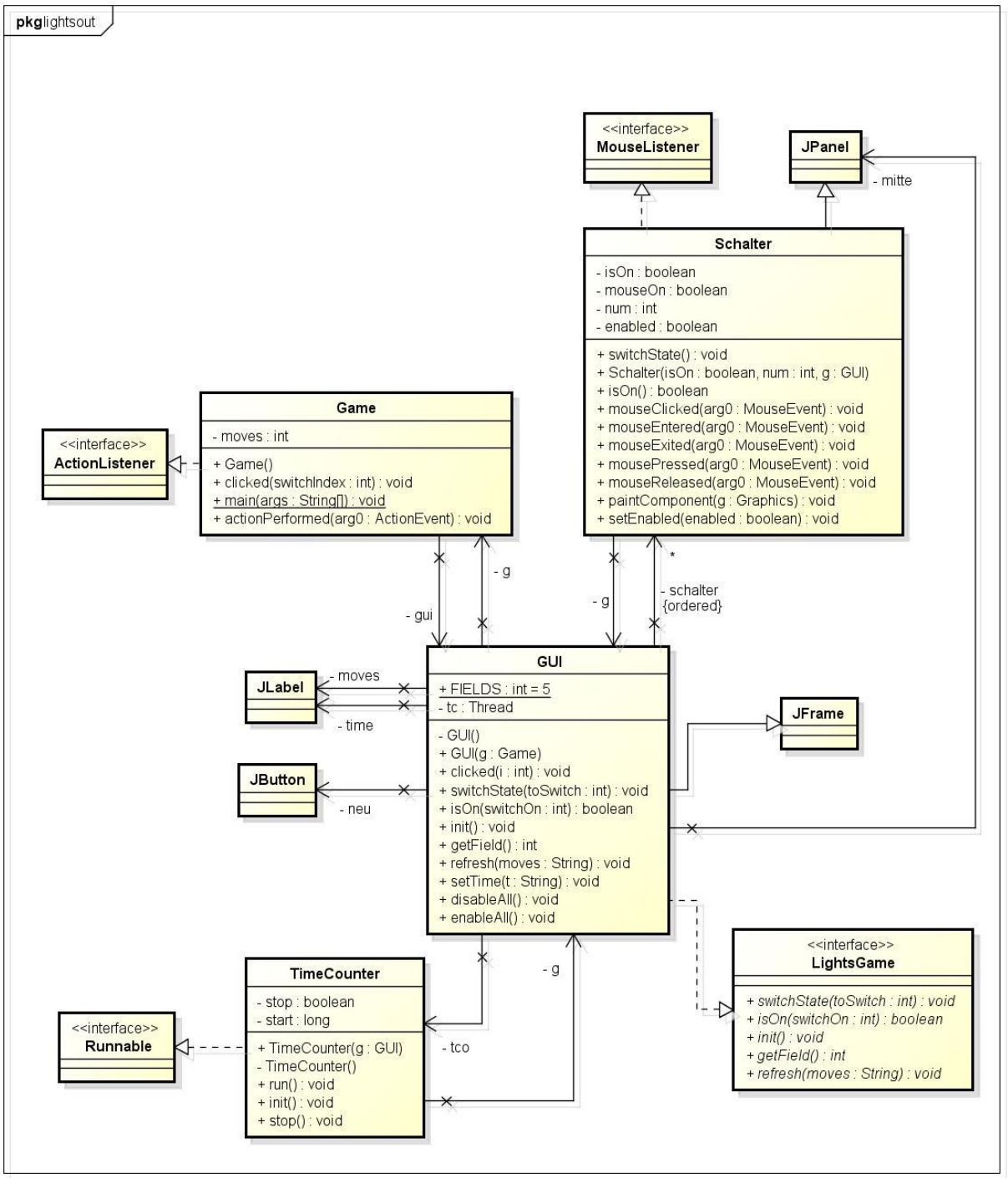
Der Entwurf der Applikation erfolgt nach dem MVC-Patters. Alle, für dieses Programm, notwendige Klassen werden in 3 verschiedene „Zuständigkeitsbereiche“ eingeteilt. Diese Zuständigkeitsbereiche sind Model (zuständig für die Logik der Anwendung selbst), View (zuständig für die Anzeige des Programms, für den Benutzer sichtbare Teil der Anwendung, GUI) und Controller (zuständig für das Zusammensetzen der anderen Teile, für das Reagieren auf Benutzereingaben und für die Ablauflogik der Anwendung).

Aus dieser, möglichst losen, Koppelung der Komponenten entsteht eine gewisse Übersichtlichkeit für den Entwickler, sowie auch eine (bei richtiger Umsetzung) leichter Erweiterbarkeit.

Framework

Wir haben uns für das Java-built-in Framework „Swing“ entschieden, da alle Teammitglieder bereits einige eigene Erfahrungen im Umgang mit Swing gemacht haben.

4. UML-Entwurf



5. Durchführung

Model

Aufgrund dessen, das dieses Spiel über (fast) keinerlei eigene Spiellogik verfügt, wurde beim Entwurf (wie man dem Klassendiagramm entnehmen kann) auf eine eigene Model-Klasse verzichtet. Die Entscheidung dafür ist durch mehrere Punkte gegeben:

Das Spiel basiert nur auf einigen Schaltflächen, welche nur einen bestimmten Zustand haben können (an oder aus). Das heißt also, dass die GUI, um zu wissen in welcher Farbe ein Feld gezeichnet wird (um den Zustand von Feldern zu unterscheiden, muss der Zustand durch eine bestimmte Farbe gekennzeichnet werden), muss jedes Feld wissen, ob es gerade an oder aus ist. Das macht ein eigenes Model, welches dafür zuständig wäre, überflüssig.

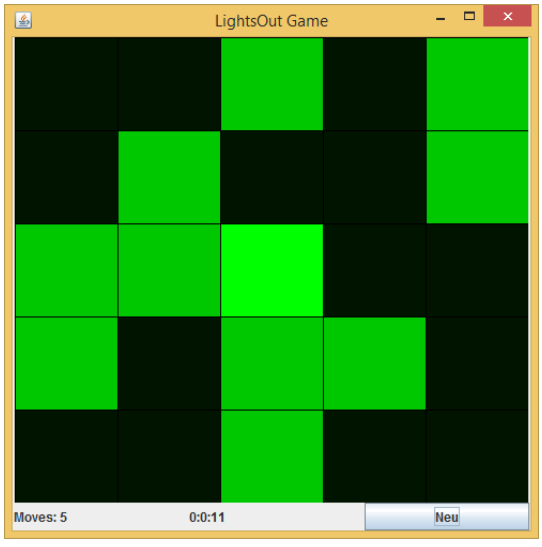
View/GUI

Die View-Komponente ist bei diesem Spiel sehr von Bedeutung, schließlich ist sie das, was der Spieler schließlich zu sehen bekommt und mit dem er interagieren muss.

Es gibt zwei Klassen die für die Darstellung zuständig sind: GUI und Schalter. Die Schalter-Klasse erbt von der Swing-Klasse JPanel und repräsentiert ein Feld welches entweder an oder aus sein kann. Sie implementiert einen MouseListener einerseits eine Art MouseOver-Effekt (ändert die Farbe) zu implementieren und andererseits um auf Klicks selbständig reagieren zu können und die clicked(indexDesFeldes)-Methode des Controller (die Game-Klasse) aufzurufen.

Die GUI-Klasse erbt von der Swing-Klasse JFrame und repräsentiert das gesamte Fenster der Applikation. Sie zeigt alle Schalter in der Mitte des Fensters an. Es werden an der unteren Seite ebenfalls die Anzahl der Züge die der Spieler gemacht hat (mittels Swing-Klasse JLabel), die Zeit, wie lange der Spieler schon spielt (ebenfalls mittels JLabel, näheres zur Zeitangabe folgt). Zusätzlich gibt es auch einen Button, zum Neustarten des Spieles (ruft die init()-Methode der GUI auf, der Zeitzähler wird zurückgesetzt und alle Schalter zufällig ein oder ausgeschaltet).

Screenshot der GUI, über mittlerem Feld war der Cursor (heller) :



Die Darstellung der aktuellen Spielzeit ist nicht so simpel, wir der Rest der GUI. Für die richtige Darstellung gibt es die Klasse TimeCounter. Diese Klasse speichert die Systemzeit (in Millisekunden) zum Start des Timers und zählt abhängig davon in einem eigenen Thread die vergangene Zeit, seit dem Start des Zählers (Differenz der jetzigen Zeit und der Startzeit). Zusätzliche hat der TimeCounter auch noch eine Methode zum Stoppen und Starten.

Controller

Der Controller selbst erzeugt nur die GUI und reagiert auf einen Klick des Buttons für das Starten eines neuen Spieles. Zusätzlich wird auch die Anzahl der Züge gespeichert und überprüft, ob der Spieler gewonnen hat (nach jedem Spielzug). Er ist ebenfalls auch der Startpunkt der Applikation (main-Methode).

Testing

Für das Testen wurden Ausgaben in der Konsole verwendet. Es kam zu keinen Fehlern, das Programm hat nach dem Hochladen ins Repository (nach dem Hinzufügen der mindestens benötigten Funktionen und Attributen) auf Anhieb funktioniert.

Der einzige Bug, welcher behoben werden musst, war die korrekte Umschaltung der umliegenden Felder. Wenn man ein Feld am linken unteren Rand umschalten wollte, kam es zu einer IndexOutOfBounds-Exception. Der Grund dafür wurde schnell gefunden: In der clicked(indexDesSchalters) der Klasse Game wurde als Feld-Breite immer direkt die Konstante FIELD der GUI-Klasse verwendet. Jedoch ist der höchste Index in dem Array nicht die Breite des Feldes sondern die Breite weniger 1 (denn die Indizes eines Arrays beginnen bekannter Weise mit 0).

6. Installation

Voraussetzung zum Starten des Programmes ist eine Installation von Java. Danach müssen die Source-Dateien (.java-Files) kompiliert werden und können über die Konsole gestartet werden. Für das Kompilieren und Starten kann die über die Konsole genauso wie (automatisiert) über eine IDE (wir verwendeten bei der Entwicklung die IDE „Eclipse“) erfolgen.

7. Referenzen

1. Java (verwendete Programmiersprache): https://www.java.com
2. Eclipse (verwendete IDE) : https://eclipse.org/