TEKO Olten

Techniker - Informatik

1.Jahr, 1. Semester

Semesterarbeit

Oware-Game

Oware-Game aus Afrika implementiert mit Java(FX)

Autor:	**Arn Yanick** Brügglisackerweg 1 4704 Niederbipp
Matrikelnummer:	20562
Gutachter:	Yanick Arn
Zweitgutachter:	Werner Naef
Abgabetermin:	17.03.2020

1.1 Kurzfassung

Für die Weiterbildung zum Techniker HF Informatik sollen wir im Modul „Lösungsalgorithmen entwerfen“ ein kleines Programm entwickeln.

Das Programm wird normalerweise mit der Software EOS geschrieben, in der man verschiedene Klassen hat, welche mehr oder weniger Formen sind, die man erstellen, vergrößern, färben und bewegen kann. Ziel der Arbeit soll es nun sein, etwas Kleines zu machen, um ein bisschen die Grundlagen der Softwareentwicklung kennenzulernen und zu zeigen, dass man es versteht.

Ich habe den Lehrer darum gebeten, eine einfache Java Applikation entwickeln zu können, bei der die Darstellung durch das Framework JavaFX übernommen wird. Dies habe ich gemacht, da ich eine langjährige Erfahrung mit der Programmiersprache Java habe und diese auch in der Firma benutze.

Ich habe mich dazu entschieden das sogenannte „Oware“-Spiel in JavaFX nachzubauen. Das Spiel kommt ursprünglich aus Afrika und ist ein typisches Eins-gegen-Eins Spiel. Von diesem Spiel gibt es verschiedene Varianten.

Inhaltsverzeichnis

Aufgabenstellung

Für das Modul „Lösungsalgorithmen entwerfen“ soll mit Java und JavaFX das Owware-Game entwickelt und dargestellt werden.

Im Spiel gibt es zwei Spieler. Beide Spieler haben dabei jeweils 6 Gruben, welche mit jeweils vier Samen gefüllt sind.

Nun wählen die Spieler abwechselnd Gruben auf ihrer Seite aus. Die Samen im jeweiligen Feld werden dann jeweils auf die nächsten Felder verteilt. Dabei gilt, dass jeder Samen in das nächste Feld vom vorherig abgelegten Samen gelegt wird.

Das erste Feld ist dabei jeweils das nächste Feld nach dem ausgewählten Startfeld. (Uhrzeigersinn)

Wenn es nun ein Spieler schafft einen Samen in ein Feld auf der Gegenseite zu legen, in dem vorher nur ein (1) Samen drin war, so darf er beide Samen (2) rausnehmen und sich die als Punkte anrechnen. (+2)

Es wird so lange gespielt bis nur noch zwei Samen übrig sind, da danach ein Endloszyklus entsteht.

Während der Entwicklung muss beachtet werden, dass es Status gibt, bei denen ein Spieler nichts machen kann, weil er keine Samen mehr auf seiner Spielseite hat. Ist dies der Fall, so wird dieser Spieler übersprungen und der Gegner kommt automatisch wieder an die Reihe. Prinzipiell bedeutet das, dass ein Spieler so oft am Zug ist, bis der andere Spieler die Möglichkeit hat einen Spielzug zu machen.

Den Spielern darf es zudem nicht möglich sein einen Spielzug auf der Seite des Gegners zu tätigen.

1.2 Detaillierte Aufgabenstellung

Für das Endprodukt wird `Java` und `JavaFX Version 8` benutzt, welches bereits gebundelt ist in der `Java JDK 8`. Die Entwicklung wird vollständig in der `Jetbrains IntelliJ Ultimate Edition` stattfinden.

Werte, die sich innerhalb des Spiels verändern, sollen in einzelnen Instanzen von Klassen abgelegt werden, welche wie folgt benennt sind:

- Spielfeld
- Grube
- Spieler

Die Gruben sind jeweils die sechs Einkerbungen auf dem Spielfeld, welche am Anfang des Spiels jeweils vier (4) *seeds* (Samen) beinhalten. Die Darstellung des ganzen Spiels wird über JavaFX Klassen übernommen. Diese werden in der Main-Klasse instanziert und mutiert. Die genaue Darstellung hat hierbei keine genauen Regeln, ausser:

- die Punktzahl der Spieler wird dargestellt
- die Anzahl an (verbleibenden) Samen wird in jeder Grube dargestellt
- der Spieler, welcher an der Reihe ist, wird hervorgehoben

Für das Einfangen von Aktionen sollen sogenannte «*EventHandler*» benutzt werden. Ob diese global (über die ganze Applikation) angewendet werden oder einzeln spielt dabei keine Rolle. Bevor das Spiel startet, soll der Spieler auswählen können, ob er gegen einen Computer spielen will oder gegen einen anderen menschlichen Spieler. Nach der Auswahl soll das Spielfeld erscheinen und das Spiel losgehen.

Daten wie *Gruben*, *Punktzahl*, Anzahl verbleibender *Seeds* und *Reichweite* des Spielers (Spielerseite des Spielers), werden über die jeweiligen Instanzen über Getter Methoden geholt und auf der Oberfläche aktualisiert. Versucht der Spieler eine Grube anzuklicken, die nicht ihm gehört, so soll eine Meldung kommen, die den Spieler darauf hinweist, dass die angewählte Grube sich nicht auf seiner Spielseite befindet.

2 Bedienungsanleitung

Das ist eine Bedienungsanleitung für das Starten der Applikation. Um herauszufinden, wie das Spiel funktioniert, wenden sie sich bitte an die Aufgabenstellung.

2.1 Java installieren & JAR herunterladen

Falls Java noch nicht installiert ist, muss es über https://www.java.com/de/download heruntergeladen und installiert werden.

Danach kann das JAR-File über OneNote runtergeladen werden.

Abschlussarbeit

Tuesday, 28 January 2020 12:29

Auftrag:
Programmiere mithilfe von Java und JavaFX (Oberfläche) das aus Afrika stammende Oware.
Programmiere hierbei die Variante, in der jeder Spieler 6 Felder hat.

Entwickle dazu einen einfachen Bot, der gegen dich spielt.
Es soll auch möglich sein, dass Spieler gegen Spieler spielt.

AYB

Wie funktioniert das Oware-Spiel?:
In Oware gibt es zwei Spieler. Beide Spieler haben dabei jeweils 6 Felder (Pits), welche mit jeweils 4 Samen gefühlt sind.

Nun wählen die Spieler abwechselnd Felder auf ihrer Seite aus. Die Seeds im jeweiligen Feld werden dann jeweils auf die nächsten Felder verteilt. Dabei gilt, dass jeder Samen in das nächste Feld vom vorherig abgelegten Samen gelegt wird.

Das erste Feld ist dabei jeweils das nächste Feld nach dem ausgewählten Startfeld. (Uhrzeigersinn)

Wenn es nun ein Spieler schafft einen Samen in ein Feld auf der Gegenerseite zu legen, in dem vorher nur ein Samen drin war, so darf er beide Samen rausnehmen und sich die als Punkte anrechnen. (+2)

Es wird so lange gespielt bis nur noch zwei Samen übrig sind, da sonst ein Endloszyklus entsteht.

Hat ein Spieler in seinem Spielzug keine Samen auf seiner Seite, so wird er übersprungen.

AYB

Downloads:

JAR

OwareGame

2.3 Starten der Applikation

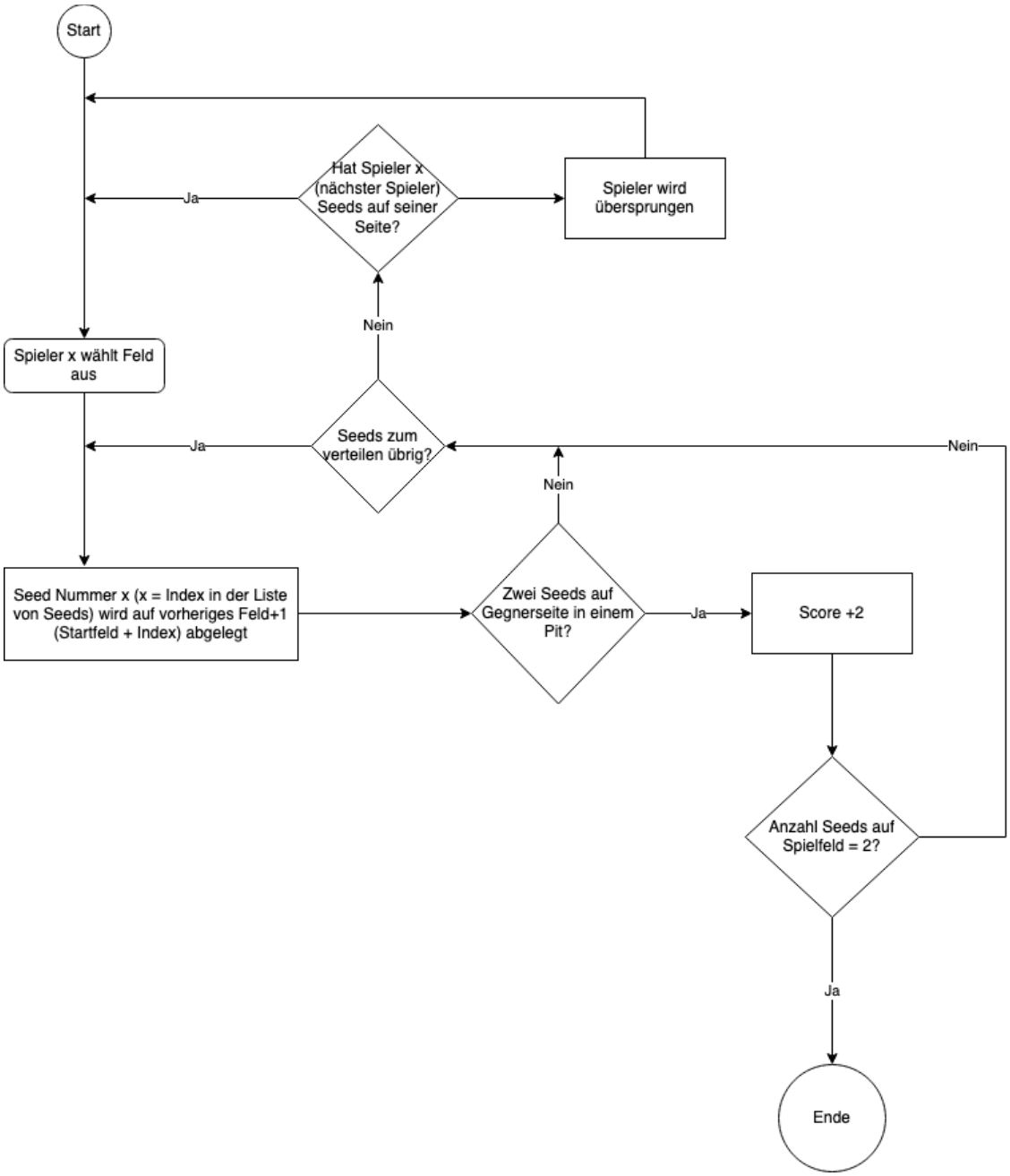
Um die Applikation zu starten, macht man jetzt einen Doppelklick auf das JAR-File oder macht einen Rechtsklick darauf und drückt «*Öffnen*». Es sollte folgendes Fenster erscheinen.

Hier kann der Spieler nun wählen, ob er gegen einen Computer oder ob er gegen einen anderen menschlichen Spieler spielen will. Hat der Spieler die Auswahl getroffen, so erscheint das Spielfeld. Spieler 1 fängt dabei immer an und beginnt, indem er auf ein Feld mit einer Zahl klickt. In der Aufgabenstellung ist beschrieben, wie genau das Spiel funktioniert.

3 Flussdiagramm

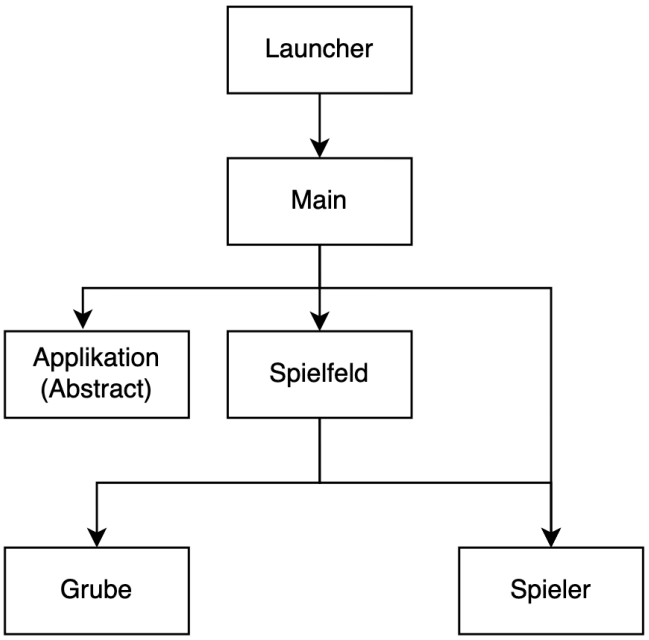
Ich habe mich für dieses Projekt für ein Flussdiagramm entschieden, da es meiner Meinung nach am meisten sind macht, für diese Art von Applikation. Das Flussdiagramm beschreibt mehr oder weniger den gesamten Spielvorgang und wurde einfach gehalten. Es ist auch im OneNote vorzufinden.

Die Variable "x" referenziert hier jeweils auf den momentanen Spieler (oder den nächsten falls es steht).
Im Falle des *Seeds* referenziert die Variable "x" auf *Seed* Nummer "x" in der Liste (Array).



4 Struktur

Die Struktur meine Software ist wie folgendermassen aufgebaut:



4.1 Launcher

Da JavaFX Schwierigkeiten hat eine Applikation aus einem JAR zu starten, wenn die *Main*-Klasse die *Application*-Klasse erweitert, muss eine Klasse erstellt werde, die lediglich die *main*-Methode in der *Main*-Klasse aufruft. Dafür ist die *Launcher*-Klasse hier.

4.2 Main

Die *Main*-Klasse erstellt, aktualisiert und verwaltet das *Spielfeld*. Das alles geschieht alles über Methoden in der *Spielfeld*-Klasse. Dazu gehören vor allem Getter- und Setter Methoden. Diese Klasse überprüft zudem, ob ein *Spieler* das Recht hat einen Spielzug zu machen oder ob sich dieser auf der falschen Spielfeldhälfte befindet.

Sie erweitert ausserdem die JavaFX Klasse *Application*, welche den Startpunkt einer JavaFX Applikation darstellt.

4.3 Spielfeld

Die Spielfeld Klasse enthält alle Informationen über das Spielfeld und verändert diese schlussendlich auch. Es enthält eine Liste von allen *Gruben*, und somit deren Anzahl an *Seeds*. Ausserdem ruft das Spielfeld die *Spieler*-Klasse auf und erhöht die *Punktzahl* des jeweiligen *Spielers*.

4.4 Grube

Diese Klasse enthält die Informationen darüber, wie viele *Seeds* in einer *Grube* verbleibend sind. Sie hat keine weitere Funktion ausser die *Seeds* zu verwalten.

4.5 Spieler

Die *Spieler* Klasse enthält die *Punktzahl* des jeweiligen *Spieler*s und Informationen wie die *Reichweite* des *Spieler*s ist. Die *Reichweite* ist lediglich, die genaue Position der *Grube*n auf seiner Spielfeldseite.

5 Schlusswort

Ich habe in diesem Projekt das erste Mal richtig mit JavaFX entwickelt. Da ich bereits relativ viel Erfahrung in der Entwicklung mit Java habe, fiel es mir jedoch nicht besonders schwer mit dem Spiel anzufangen oder es im Grossen und Ganzen zu entwickeln. Ich konnte jedoch trotzdem ein bisschen lernen, wie man mit JavaFX umgeht.

5.1 Verbesserungen

Für das nächste Projekt sollte ich mich unbedingt vor dem Beginn, oder unter anderem während der Planung darüber informieren, wie die angewandten Technologien funktionieren. Grund dafür ist, dass es für mich eine Herausforderung war JavaFX so zu konfigurieren, dass es funktioniert und ich schlussendlich auf eine Windows Maschine ausweichen musste.

Eidesstattliche Versicherung

Hiermit erkläre ich, Yanick Arn, an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.“

Niederbipp, 15.03.20,