Pflichtenheft
Roos & Hüfner

Gruppenmitglieder

- Alexander Roos
- Kevin Hüfner

Beschreibung des Projekts / Spielprinzip

Aufgebaut ist das Spiel „Bott ärgere dich nicht“, ähnlich dem besser bekannten Spiel „Mensch ärgere dich nicht“.
Ziel des Spiels ist, alle seiner vier Spielfiguren (Token) in die eigenen Zielfelder zu bewegen. Dazu müssen sich die Spielfiguren eine Runde durch das gesamte Spielfeld bewegt haben. Dabei treffen diese auf die Spielfiguren der Gegenspieler, welche die eigenen Spielfiguren (Token) wieder an den Anfang werfen können. Damit eine Spielfigur (Token) das Spielfeld betreten kann, muss eine 6 gewürfelt werden. Um wie viele Spielfelder sich einer der vier Spielfiguren (Token) bewegt, wird mittels eines virtuellen Würfels (1-6) bestimmt. Alle Spieler sind der Reihe nach dran, um zu würfeln und dann eines ihrer Spielfiguren zu bewegen. Die maximale Spieleranzahl sind 4 Personen. Damit ein Spiel gestartet werden kann, müssen mindestens 2 Spieler vorhanden sein.
Der Eingabe zur Steuerung des Spiels geschieht mittels Eingabe vorgegebener Buchstaben/Zahlen (Key), welche über die Tastatur ausgewählt werden müssen.

Grundfunktionen des Spiels

- Das Spiel läuft rundenbasiert ab (aus dem Lastenheft)
    - Jeder Spieler ist der Reihe nach dran. Dieser Vorgang wird wiederholt, bis einer der Spieler alle seine 4 Spielfiguren ins Ziel gebracht hat, oder das Spiel abgebrochen wird.
- Der Spielverlauf soll auf der Konsole ausgegeben werden (aus dem Lastenheft)
    - Das Spielfeld wird auf der Konsole inkl. der Auswahlmenüs ausgegeben. Nur mögliche Spielzüge werden dargestellt. Nach jedem Zug eines Spielers wird das Spielfeld aktualisiert und neu auf der Konsole ausgegeben. Um dem Spiel „Mensch ärgere dich nicht“ so nahezukommen wie möglich, und die einzelnen Spielfiguren (Token) auseinanderhalten zu können, werden die Spielfiguren (Token) jeweils in der Farbe dargestellt, die sich der jeweilige Spieler am Anfang des Spiels ausgewählt hat.

- Es soll ein Zustand existieren, an dem das Spiel gewonnen oder verloren ist (aus dem Lastenheft)
    - Sobald einer der teilnehmenden Spieler alle seine 4 Spielfiguren (Token) im Ziel hat, hat dieser gewonnen und alle anderen Mitspieler haben automatisch verloren. Das Spiel endet mit dem Sieg eines Spielers.
- Das Spiel soll ein Menü enthalten (aus dem Lastenheft)
    - Zur Verwaltung des gesamten Spiels enthält das Spiel mehrere Menüs bzw. Untermenüs. Darstellung und Aufbau aller Menüs sind im Anhang zu finden.
- Das Spiel muss sinnvoll auf mehrere Klassen aufgeteilt sein, also objektorientiert programmiert werden (aus dem Lastenheft)
    - Siehe UML im Anhang

Erweiterte Funktionen

- Ein Mehrspielermodus soll möglich sein. (An einem PC)
    - Das Spiel ist für 2-4 Spieler konzipiert. Es besteht die Möglichkeit gegen einen oder mehrere reellen Spieler zu spielen, als auch gegen einen oder mehrere Bots. Die jeweiligen Spielzüge der Spieler werden über die Tastatur gesteuert.

- Der Spielstand soll speicherbar sein
    - Die Spielstände können über das Menü gespeichert und wieder aufgerufen werden. Ebenso werden die Spieler abgespeichert, welche verwendet wurden. Diese können dann für zukünftige Spiele wieder ausgewählt werden.
    - Es ist nicht möglich, die Spieler mittendrin zu wechseln.

- Es soll eine Bestenliste existieren (sortiert) und über das Menü aufrufbar sein (Dauerhaft gespeichert)
    - Hinter Menüpunkt „highscore“ befindet sich die Bestenliste aller abgespeicherten Spieler. Hier wird zu jedem Spieler die Anzahl an gewonnenen Spielen, Anzahl an gewürfelten 6en und die Anzahl an gespielten Spielen. Geordnet wird diese Liste nach dem Erstellzeitpunkt des Spielers.
      (Anhang_2)

Funktionen der verwendeten Klassen

- Klasse „Game“
    - In der Klasse Game befindet sich die Haupt-Logik des Spiels
    - Wird sowohl zur Auswahl der benötigten UI-Elemente als auch für die Spielsteuerung verwendet
    - Methode Run() startet das Programm
- Klasse „Menu“
    - Werden die jeweiligen UI-Elemente anhand der aktuellen Spiellage in ArrayLists<MenuItem> zusammengebaut
    - Ebenso befinden sich in der Klasse die Methoden zur Darstellung der UI-Elemente
- Klasse „Board“
    - Hier befindet sich der gesamte Aufbau des Spielfeldes inkl. der „Koordinaten“, welche für den Spielablauf benötigt werden. Das Spielfeld template ist als String[][] dargestellt (Anhang_1). Die Auflistung der „Koordinaten“ jeweils in einem int[][]. Ebenso befindet sich in der Klasse die Methode zur Darstellung des gesamten Spielfeldes inklusive der UI-Elemente
- Klasse „Score“
    - Wird das Spielfeld und deren Spieler gespeichert.
    - Methoden zur Manipulation des Spielstandes. Darunter zählt unteranderem: bewegen einer Spielfigur (Token), Überprüfung ob eine Bewegung mit einer Spielfigur (Token) möglich ist und das Würfeln im Spiel
- Klasse „Player“
    - Name und Statistik eines Spielers abgelegt
    - Der Name des Spielers dient ebenso als Identifikationsmerkmal des Spielers
- Klasse „PlayerList“
    - Beinhaltet eine ArrayList<Player> und zusätzlich die Methoden, um Spieler hinzuzufügen und deren Name zu ändern
    - Ebenso wird diese Klasse zum Abspeichern der Spieler-Objekte verwendet
- Klasse „ColorItem“
    - Ist eine Hilfsklasse der Klasse Color
    - In ihr werden die Color-Codes und die Bezeichnungen der jeweiligen Farben abgelegt
- Klasse „Color“
    - In ihr werden 5 Objekte der Klasse „ColorItem“ erstellt
    - Diese Objekte repräsentieren die Farben „Rot, Blau, Grün, Gelb und Reset“
    - Der Farbcode von „Reset“ stellt die Ursprungs-Formatierung der Konsole wieder her.

- Klasse „MenuItem“
    - Ist eine Hilfsklasse von Menu
    - In dieser wird der Anzeigetext, der Such-Schlüssel (Key) und der dazugehörige Rückgabewert abgelegt
- Klasse „Main“
    - Aufrufen der Methode „Run()“ zum Starten des Programms

Spielablauf

1. Nachdem im Startmenü ausgewählt wurde, dass ein neues Spiel gestartet werden soll, muss die Anzahl der teilnehmenden Spieler ausgewählt werden.
2. Es muss nun eine Spielerauswahl getroffen werden. Sollten Spieler bereits gespeichert sein, so werden diese aufgelistet und können ausgewählt werden. Alternativ muss über das Startmenü in das User-Menu gegangen werden und dort ein neuer Spieler erstellt werden. Danach stehen die neu erstellten Spieler zum Auswählen zur Verfügung.
3. Mitspieler müssen sich eine Farbe für ihre Spielfiguren aussuchen
    ↳ Sollte ein Bot an dem Spiel teilnehmen, so muss dessen Farbe von einem reellen Mitspieler ausgewählt werden.
    ↳ Eine Farbe kann nur einmal ausgewählt werden.
4. Es wird automatisch für alle Mitspieler einmal gewürfelt. Der Spieler mit der höchsten gewürfelten Zahl beginnt das Spiel.
5. Spieler würfelt.
6. Sobald einer der Spieler eine 6 gewürfelt hat, kann dieser einen seiner Spielfiguren (Token) auf das Spielfeld bewegen.
7. Sobald ein Spieler mindestens einer seiner Spielfiguren (Token) auf dem Spielfeld hat, kann dieser jedes Mal, wenn er gewürfelt hat, auswählen, welcher seiner möglichen Spielfiguren (Token) er bewegen möchte.
    ↳ Im Spiel werden dem Spieler nur die Spielfiguren zum Auswählen angezeigt, welche, anhand der gewürfelten Zahl, bewegt werden können.
8. Nun kommen alle Spieler der Reihe nach dran, um zu würfeln. Nach jedem Würfeln kann der Spieler wieder auswählen, welchen seiner verfügbaren Spielfiguren (Token) er bewegen möchte. Sollte es keine Möglichkeit geben einer der Spielfiguren (Token) zu bewegen, so wird direkt an den nächsten Spieler übergeben.
9. Das Spiel endet, sobald ein Spieler alle seiner 4 Spielfiguren in das Ziel befördert, hat.

Abspeichern der Spieler und Spielstände

Beim Starten des Spieles, werden die notwendigen Dateipfade, falls diese nicht schon Existieren angelegt. Spieler in „./save/palyers“ und Spielstände in „./save/scores“.

Spieler:

> Die Spieler werden in einer ArrayList<Player> abgelegt. Diese wird über die Klasse PlayerList verwaltet. Beim Beenden des Spieles (quit), wird diese Liste in eine Json Datei abgelegt. Diese wird beim erneuten Starten des Spieles wieder geladen und steht dann im Spiel wieder zur Verfügung.

Spielstände:

> Die Spielstände können jederzeit während des Spieles im Hauptmenü abgespeichert werden. Ein Spielstand kann nur dann nicht abgespeichert werden, wenn alle Spieler von „Minions“ belegt wurden, dann muss das Spiel erst bis zum Ende gespielt werden bevor der Spielstand abgespeichert werden kann. Wenn ein Spielstand abgespeichert wird, dann wird das Erstellzeitpunkt genutzt, um den Namen der Datei zu generieren. Bei einem Erneutem Speichern wird der vorherige Spielstand überschrieben. Jeder Gespeicherte Spielstand wird in einer eigenen Json Datei abgelegt.

Installation, benötigte Bibliotheken und Einstellungen

Um das Spiel mit allen Funktionen auszuführen zu können, muss ein Add-on und 3 zusätzliche Bibliotheken hinzugefügt werden. Das benötigte Add-on ist: „**ANSI Escape in Console**“, welches im Eclipse Marketplace zu erhalten ist. Zusätzlich muss in den Einstellungen der Konsole das **„\r“ aktiviert** werden. Die weiteren Bibliotheken sind Bibliotheken, um die Json-Datei Handhabung zu realisieren. Diese Bibliotheken sind:

- **Jackson-core-2.13.3.jar**
- **Jackson-databind-2.13.3.jar**
- **Jackson-annotations-2.13.3.jar**

Erläuterung der Anhänge

- Anhang_1
  - ↳ Aufbau des Spielfeldes inkl. Adressierungen der Spielfelder
- Anhang_2
  - ↳ Darstellung aller Benutzeroberflächen und deren Zusammenhänge
- Anhang_3
  - ↳ Flussdiagramm der Methode „checkMove()“ der Klasse „Score“
- Anhang_4
  - ↳ Flussdiagramm der Methode „move()“ der Klasse „Score“
- Anhang_5
  - ↳ UML_Digramm des Programms
- Anhang_6
  - ↳ Stark vereinfachtes Flussdiagramm der Methode „run()“ der Klasse „Game“