|  |
| --- |
|  |
| IHK Simulation der Programmierung eines Softwaresystems |
| Simulation der IHK Abschlussprüfung |

|  |
| --- |
| Markus Fassbender  11.04.2014 |

Inhaltsverzeichnis

[Aufgaben Analyse 2](#_Toc384716951)

[Analyse 2](#_Toc384716952)

[Eingabeformat 2](#_Toc384716953)

[Ausgabeformat 2](#_Toc384716954)

[Anforderung an das Gesamtsystem 2](#_Toc384716955)

[Sonderfälle 2](#_Toc384716956)

[Verfahrensbeschreibung 2](#_Toc384716957)

[Strategie von Spieler 1 2](#_Toc384716958)

[Gesamtsystem 3](#_Toc384716959)

# Aufgabenanalyse

## Analyse

Es soll die Simulation eines Zwei-Personen-Spiels mit dem Namen Nim umgesetzt werden. Dabei werden aus mehreren Reihen, aber mindestens 1 und maximal 9, Streichhölzer gezogen. Ein Zug darf immer nur 1-n Hölzer aus einer Reihe ziehen, die n Hölzer hat. Die Reihenfolge der vertikal angeordneten Reihen ist nicht relevant, da ja eine beliebige ausgewählt wird.

Der Spieler, welcher die letzten Hölzchen zieht, gewinnt. Daher muss eine Strategie für Spieler 1 entwickelt werden, die eine möglichst gute Ausgangslage schafft. Spieler 2 ist durch den Zufall gesteuert, solange er nicht sofort gewinnen kann. Berücksichtigt werden müssen auch Spielfehler, dass kein Hölzchen genommen wurde oder aus mehreren Reihen.

Wie eine möglichst gute Ausgangslage aussieht ist nicht näher definiert. Daher muss eine Strategie entworfen werden, die diese erzeugt bzw. findet. Ein Spiel wird 10mal durchgeführt und es gibt in jeder Runde einen Sieger. Am Ende wird eine kurze Statistik ausgeben.

Das Programm kann grob in ein Model, eine View und einen Controller unterteilt werden. Dabei sind alle Daten im Model gespeichert, die Ausgabe erfolgt über die View und der Controller nimmt alle Berechnungen vor.

## Eingabeformat

## Ausgabeformat

## Anforderung an das Gesamtsystem

## Sonderfälle

## Fehlerfälle

# Verfahrensbeschreibung

## Strategie von Spieler 1

Es ist sehr wichtig, die gute Ausgangslage zu finden. Daher ist meine Strategie in drei Schritte aufgeteilt. Zuerst werden alle möglichen Schritte erzeugt, wobei einige Einschränkungen gelten. Bei einer 1 in einer Reihenfolge kann nur 0 oder 1 rauskommen. Bei einer 2 kann nur 0 oder 1 rauskommen. Sonst kann 0, 1 oder 2 rauskommen. So generiere ich für jede Reihe, die ungleich 0 ist, 1-3 mögliche Entscheidungen.

Im zweiten Schritt werden die Möglicheiten nach folgendem Schema bewertet:

* a entspricht der Anzahl aller Reihen
* b entspricht der Anzahl der Reihen ungleich 0
* e entspricht der Anzahl der Reihen mit Einsen
* s ist eine Summe und zu beginn 0

wenn b gerade ist

s wird um a-b erhöht

s wird um e erhöht

sonst

s wird um (a-b-3) erhöht

s wird um e verringert

Dadurch erhalte ich eine Punktzahl, die bestimmt wie gut eine Möglichkeit ist. Eine ungerade Anzahl an Reihen ungleich 0 ist also sehr gut, insbesondere wenn darin Einsen sind. Eine unge Anzahl an Reihen ungleich 0 ist schlecht und noch schlimmer, wenn Einsen darin vorkommen.

Die beste Entscheidung wird im dritten Schritt ermittelt und ausgeführt. Bei mehreren gleichguten kann eine beliebige (ggf. die erste) ausgewählt werden.

## Gesamtsystem

// TODO

# Änderungen

## Strategie

1. Konstruktur zu Possibility hinzugefügt
2. Copy-Konstruktor zu GameState hinzugefügt
3. GameState hat toString() überschrieben