Aufgabe 3: Tobis Turner

Team-ID: 00325

Team-Name: pwned

Bearbeiter/-innen dieser Aufgabe:   
Anton Ferdinand Althoff

23. Oktober 2020

Inhaltsverzeichnis

[Lösungsidee 1](#_Toc54558000)

[Umsetzung 1](#_Toc54558001)

[Beispiele 3](#_Toc54558002)

[Quellcode 4](#_Toc54558003)

[RoundLiga Klasse 4](#_Toc54558004)

[4](#_Toc54558005)

[RoundKO Klasse 5](#_Toc54558006)

[7](#_Toc54558007)

[Game Klasse 8](#_Toc54558008)

[Player Klasse 9](#_Toc54558009)

[SolverInput Klasse 10](#_Toc54558010)

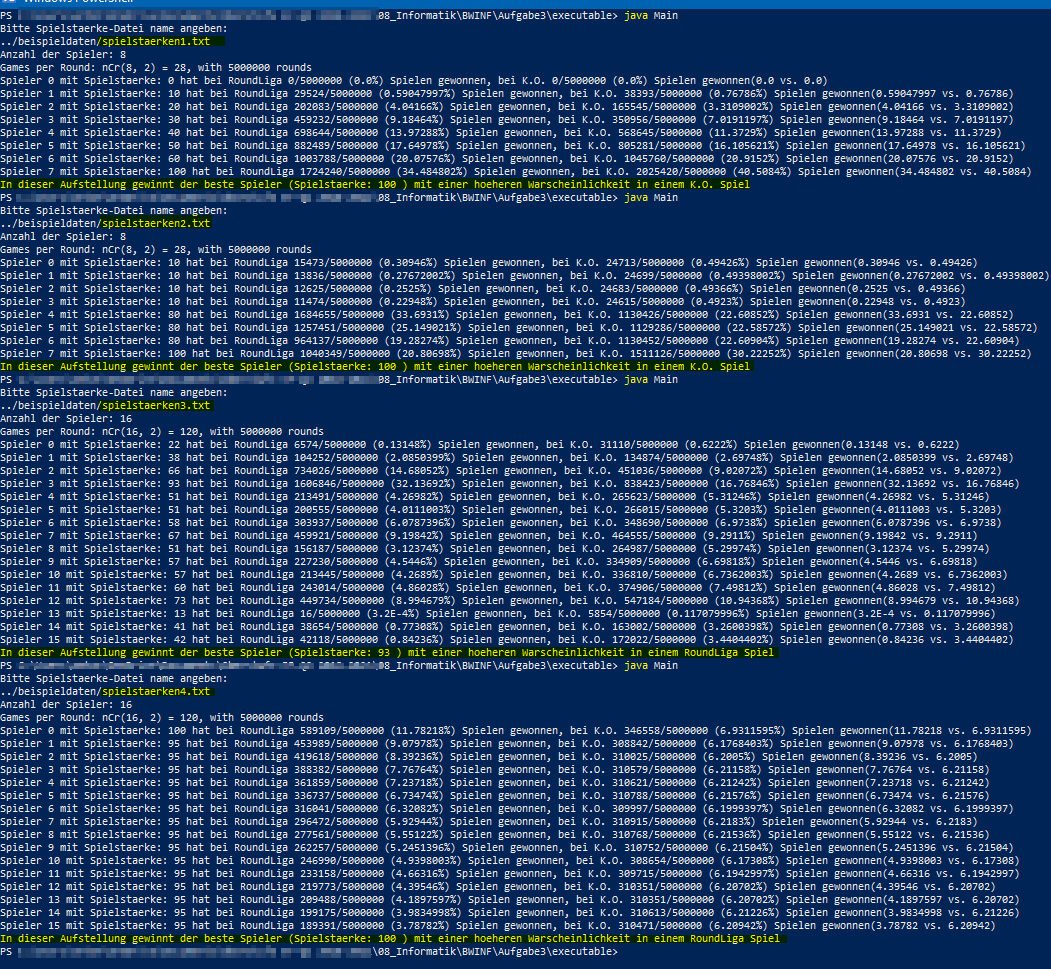
# Lösungsidee

Um herauszufinden, welche Spielform (Liga oder K.O.) sich besser eignet, um den insgesamt besten Spieler herauszustellen und somit Tobis Frage zu beantworten, simulieren wir beide Varianten mit einer zufälligen Spielerverteilung und notieren bei beiden Varianten, welcher Spieler gewinnt. Dies machen wir mehrere Millionen Mal, um die durchschnittliche Gewinnwahrscheinlichkeit für jeden Spieler in beiden Spielformen ermitteln zu können. Dann suchen wir uns den insgesamt stärksten Spieler raus, und vergleichen beide Varianten, und ermitteln in welcher von beiden Spielformen seine/ihre Gewinnwahrscheinlichkeit am höchsten ist. Diese Spielform empfehlen wir Tobi.

# Umsetzung

Für die Umsetzung habe ich Java genommen, da Java leicht lesbar ist, und OOP unterstützt. Aufgeteilt ist das Programm in 6 Klassen: Main, SolverInput, Player, Game, RoundLiga, und RoundKO, wobei die beiden Round Klassen beide Spielformen (Liga und K.O) simulieren. Hierzu benötigen sie die Helfer-Klasse Player, welche einen Spieler simuliert, und die Klasse Game, welche ein einzelnes Spiel zwischen zwei Spielern simuliert. Die RoundLiga Klasse bekommt eine Spielerliste angegeben, wodrauf sie alle möglichen Kombinationen von Spielen zwischen zwei Spielen mithilfe der Methode permutations() berechnet, und Spiele(Game-Objekte) erstellt. Die Methode playRound() spielt alle diese Spiele einmal durch, und gibt den Gewinner (mit den am meisten gewonnenen Spielen) zurück. Diese Methode ist jedoch private, da ein einzelnes Spiel nicht die Unregelmäßigkeiten aussortiert: wir müssen mehrere Tausende Spiele spielen, um einen Durchschnitt zu bekommen. Hierzu dient die Methode playNRounds(). Die führt die Methode playRounds n-mal durch, und notiert die Gewinner, wodurch man die durchschnittliche Gewinnwahrscheinlichkeit berechnen kann. Die Klasse RoundKO fungiert nach demselben Prinzip, mit dem Unterschied, dass für jede Runde zufällige Spielkombinationen gewürfelt werden müssen, damit nicht immer dieselben Spieler gegeneinander antreten. Die Gewinner der ersten Etappe treten wieder gegeneinander an etc.. , bis nur noch ein Player übrig ist. Dieser gewinnt die Runde. Die Klasse SolverInput dient als User-Interface: Sie liest die Spielstaerken-Datei ein, kreiert die RoundLiga und RoundKO Objekte, berechnet die Prozentuale Gewinnwarscheinlichkeit für jeden Spieler anhand der zurückgegebenen Daten, und empfielt Tobi eine Spielform, die den insgesamt besten Spieler am häufigsten hervorbringt. Die Main Methode dient hier nur als Entry-Point, die ein SolverInput Objekt erstellt.

# Beispiele

Hier sind alle angegebenen Beispieldaten durchgerechnet, und die Ergebnisse markiert.

Die Empfehlungen lauten:

1. (**../beispieldaten/spielstaerken1.txt)** In dieser Aufstellung gewinnt der beste Spieler (Spielstärke: 100 ) mit einer höheren Wahrscheinlichkeit in einem K.O. Spiel
2. (**../beispieldaten/spielstaerken2.txt)** In dieser Aufstellung gewinnt der beste Spieler (Spielstärke: 100 ) mit einer höheren Wahrscheinlichkeit in einem K.O. Spiel
3. (**../beispieldaten/spielstaerken3.txt)** In dieser Aufstellung gewinnt der beste Spieler (Spielstärke: 93 ) mit einer höheren Wahrscheinlichkeit in einem Liga Spiel
4. (**../beispieldaten/spielstaerken4.txt)** In dieser Aufstellung gewinnt der beste Spieler (Spielstärke: 100 ) mit einer höheren Wahrscheinlichkeit in einem Liga Spiel

# Quellcode

## RoundLiga Klasse

## 

## RoundKO Klasse



## 

## Game Klasse

## Player Klasse

## SolverInput Klasse