Aufgabe 3: Tobis Turner

Team-ID: 00325

Team-Name: pwned

Bearbeiter/-innen dieser Aufgabe:
Anton Ferdinand Althoff

25. Oktober 2020

Inhaltsverzeichnis

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.

```
PS [redacted]08_Informatik\BWINF\Aufgabe3\executable> java Main
Bitte Spielstaerke-Datei name angeben:
../beispieldaten/spielstaerken1.txt
Anzahl der Spieler: 8
Games per Round: nCr(8, 2) = 28, with 5000000 rounds
Spieler 0 mit Spielstaerke: 0 hat bei RoundLiga 0/5000000 (0.0%) Spielen gewonnen, bei K.O. 0/5000000 (0.0%) Spielen gewonnen(0.0 vs. 0.0)
Spieler 1 mit Spielstaerke: 10 hat bei RoundLiga 29524/5000000 (0.59047997%) Spielen gewonnen, bei K.O. 38393/5000000 (0.76786%) Spielen gewonnen(0.59047997 vs. 0.76786)
Spieler 2 mit Spielstaerke: 20 hat bei RoundLiga 202083/5000000 (4.04166%) Spielen gewonnen, bei K.O. 165545/5000000 (3.3109002%) Spielen gewonnen(4.04166 vs. 3.3109002)
Spieler 3 mit Spielstaerke: 30 hat bei RoundLiga 459232/5000000 (9.18464%) Spielen gewonnen, bei K.O. 350956/5000000 (7.0191197%) Spielen gewonnen(9.18464 vs. 7.0191197)
Spieler 4 mit Spielstaerke: 40 hat bei RoundLiga 698644/5000000 (13.97288%) Spielen gewonnen, bei K.O. 568645/5000000 (11.3729%) Spielen gewonnen(13.97288 vs. 11.3729)
Spieler 5 mit Spielstaerke: 50 hat bei RoundLiga 882489/5000000 (17.64978%) Spielen gewonnen, bei K.O. 805281/5000000 (16.105621%) Spielen gewonnen(17.64978 vs. 16.105621)
Spieler 6 mit Spielstaerke: 60 hat bei RoundLiga 1003788/5000000 (20.07576%) Spielen gewonnen, bei K.O. 1045760/5000000 (20.9152%) Spielen gewonnen(20.07576 vs. 20.9152)
Spieler 7 mit Spielstaerke: 100 hat bei RoundLiga 1724240/5000000 (34.484802%) Spielen gewonnen, bei K.O. 2025420/5000000 (40.5084%) Spielen gewonnen(34.484802 vs. 40.5084)
In dieser Aufstellung gewinnt der beste Spieler (Spielstaerke: 100 ) mit einer hoeheren Warscheinlichkeit in einem K.O. Spiel
PS [redacted]08_Informatik\BWINF\Aufgabe3\executable> java Main
Bitte Spielstaerke-Datei name angeben:
../beispieldaten/spielstaerken2.txt
Anzahl der Spieler: 8
Games per Round: nCr(8, 2) = 28, with 5000000 rounds
Spieler 0 mit Spielstaerke: 10 hat bei RoundLiga 15473/5000000 (0.30946%) Spielen gewonnen, bei K.O. 24713/5000000 (0.49426%) Spielen gewonnen(0.30946 vs. 0.49426)
Spieler 1 mit Spielstaerke: 10 hat bei RoundLiga 13836/5000000 (0.27672002%) Spielen gewonnen, bei K.O. 24699/5000000 (0.49398002%) Spielen gewonnen(0.27672002 vs. 0.49398002)
Spieler 2 mit Spielstaerke: 10 hat bei RoundLiga 12625/5000000 (0.2525%) Spielen gewonnen, bei K.O. 24683/5000000 (0.49366%) Spielen gewonnen(0.2525 vs. 0.49366)
Spieler 3 mit Spielstaerke: 10 hat bei RoundLiga 11474/5000000 (0.22948%) Spielen gewonnen, bei K.O. 24615/5000000 (0.4923%) Spielen gewonnen(0.22948 vs. 0.4923)
Spieler 4 mit Spielstaerke: 80 hat bei RoundLiga 1684655/5000000 (33.6931%) Spielen gewonnen, bei K.O. 1130426/5000000 (22.60852%) Spielen gewonnen(33.6931 vs. 22.60852)
Spieler 5 mit Spielstaerke: 80 hat bei RoundLiga 1257451/5000000 (25.149021%) Spielen gewonnen, bei K.O. 1129286/5000000 (22.58572%) Spielen gewonnen(25.149021 vs. 22.58572)
Spieler 6 mit Spielstaerke: 80 hat bei RoundLiga 964137/5000000 (19.28274%) Spielen gewonnen, bei K.O. 1130452/5000000 (22.60904%) Spielen gewonnen(19.28274 vs. 22.60904)
Spieler 7 mit Spielstaerke: 100 hat bei RoundLiga 1040349/5000000 (20.80698%) Spielen gewonnen, bei K.O. 1511126/5000000 (30.22252%) Spielen gewonnen(20.80698 vs. 30.22252)
In dieser Aufstellung gewinnt der beste Spieler (Spielstaerke: 100 ) mit einer hoeheren Warscheinlichkeit in einem K.O. Spiel
PS [redacted]08_Informatik\BWINF\Aufgabe3\executable> java Main
Bitte Spielstaerke-Datei name angeben:
../beispieldaten/spielstaerken3.txt
Anzahl der Spieler: 16
Games per Round: nCr(16, 2) = 120, with 5000000 rounds
Spieler 0 mit Spielstaerke: 22 hat bei RoundLiga 6574/5000000 (0.13148%) Spielen gewonnen, bei K.O. 31110/5000000 (0.6222%) Spielen gewonnen(0.13148 vs. 0.6222)
Spieler 1 mit Spielstaerke: 38 hat bei RoundLiga 104252/5000000 (2.0850399%) Spielen gewonnen, bei K.O. 134874/5000000 (2.69748%) Spielen gewonnen(2.0850399 vs. 2.69748)
Spieler 2 mit Spielstaerke: 66 hat bei RoundLiga 734026/5000000 (14.68052%) Spielen gewonnen, bei K.O. 451036/5000000 (9.02072%) Spielen gewonnen(14.68052 vs. 9.02072)
Spieler 3 mit Spielstaerke: 93 hat bei RoundLiga 1606846/5000000 (32.13692%) Spielen gewonnen, bei K.O. 838423/5000000 (16.76846%) Spielen gewonnen(32.13692 vs. 16.76846)
Spieler 4 mit Spielstaerke: 51 hat bei RoundLiga 213491/5000000 (4.26982%) Spielen gewonnen, bei K.O. 265623/5000000 (5.31246%) Spielen gewonnen(4.26982 vs. 5.31246)
Spieler 5 mit Spielstaerke: 51 hat bei RoundLiga 200555/5000000 (4.0111003%) Spielen gewonnen, bei K.O. 266015/5000000 (5.3203%) Spielen gewonnen(4.0111003 vs. 5.3203)
Spieler 6 mit Spielstaerke: 58 hat bei RoundLiga 303937/5000000 (6.0787396%) Spielen gewonnen, bei K.O. 348690/5000000 (6.9738%) Spielen gewonnen(6.0787396 vs. 6.9738)
Spieler 7 mit Spielstaerke: 67 hat bei RoundLiga 459921/5000000 (9.19842%) Spielen gewonnen, bei K.O. 464555/5000000 (9.2911%) Spielen gewonnen(9.19842 vs. 9.2911)
Spieler 8 mit Spielstaerke: 51 hat bei RoundLiga 156187/5000000 (3.12374%) Spielen gewonnen, bei K.O. 264987/5000000 (5.29974%) Spielen gewonnen(3.12374 vs. 5.29974)
Spieler 9 mit Spielstaerke: 57 hat bei RoundLiga 227230/5000000 (4.5446%) Spielen gewonnen, bei K.O. 334909/5000000 (6.69818%) Spielen gewonnen(4.5446 vs. 6.69818)
Spieler 10 mit Spielstaerke: 57 hat bei RoundLiga 213445/5000000 (4.2689%) Spielen gewonnen, bei K.O. 336810/5000000 (6.7362003%) Spielen gewonnen(4.2689 vs. 6.7362003)
Spieler 11 mit Spielstaerke: 60 hat bei RoundLiga 243014/5000000 (4.86028%) Spielen gewonnen, bei K.O. 374906/5000000 (7.49812%) Spielen gewonnen(4.86028 vs. 7.49812)
Spieler 12 mit Spielstaerke: 73 hat bei RoundLiga 449734/5000000 (8.994679%) Spielen gewonnen, bei K.O. 547184/5000000 (10.94368%) Spielen gewonnen(8.994679 vs. 10.94368)
Spieler 13 mit Spielstaerke: 13 hat bei RoundLiga 16/5000000 (3.2E-4%) Spielen gewonnen, bei K.O. 5854/5000000 (0.117079996%) Spielen gewonnen(3.2E-4 vs. 0.117079996)
Spieler 14 mit Spielstaerke: 41 hat bei RoundLiga 38654/5000000 (0.77308%) Spielen gewonnen, bei K.O. 163002/5000000 (3.2600398%) Spielen gewonnen(0.77308 vs. 3.2600398)
Spieler 15 mit Spielstaerke: 42 hat bei RoundLiga 42118/5000000 (0.84236%) Spielen gewonnen, bei K.O. 172022/5000000 (3.4404402%) Spielen gewonnen(0.84236 vs. 3.4404402)
In dieser Aufstellung gewinnt der beste Spieler (Spielstaerke: 93 ) mit einer hoeheren Warscheinlichkeit in einem RoundLiga Spiel
PS [redacted]08_Informatik\BWINF\Aufgabe3\executable> java Main
Bitte Spielstaerke-Datei name angeben:
../beispieldaten/spielstaerken4.txt
Anzahl der Spieler: 16
Games per Round: nCr(16, 2) = 120, with 5000000 rounds
Spieler 0 mit Spielstaerke: 100 hat bei RoundLiga 589109/5000000 (11.78218%) Spielen gewonnen, bei K.O. 346558/5000000 (6.9311595%) Spielen gewonnen(11.78218 vs. 6.9311595)
Spieler 1 mit Spielstaerke: 95 hat bei RoundLiga 453989/5000000 (9.07978%) Spielen gewonnen, bei K.O. 308842/5000000 (6.1768403%) Spielen gewonnen(9.07978 vs. 6.1768403)
Spieler 2 mit Spielstaerke: 95 hat bei RoundLiga 419618/5000000 (8.39236%) Spielen gewonnen, bei K.O. 310025/5000000 (6.2005%) Spielen gewonnen(8.39236 vs. 6.2005)
Spieler 3 mit Spielstaerke: 95 hat bei RoundLiga 388382/5000000 (7.76764%) Spielen gewonnen, bei K.O. 310579/5000000 (6.21158%) Spielen gewonnen(7.76764 vs. 6.21158)
Spieler 4 mit Spielstaerke: 95 hat bei RoundLiga 361859/5000000 (7.23718%) Spielen gewonnen, bei K.O. 310621/5000000 (6.21242%) Spielen gewonnen(7.23718 vs. 6.21242)
Spieler 5 mit Spielstaerke: 95 hat bei RoundLiga 336737/5000000 (6.73474%) Spielen gewonnen, bei K.O. 310788/5000000 (6.21576%) Spielen gewonnen(6.73474 vs. 6.21576)
Spieler 6 mit Spielstaerke: 95 hat bei RoundLiga 316041/5000000 (6.32082%) Spielen gewonnen, bei K.O. 309997/5000000 (6.1999397%) Spielen gewonnen(6.32082 vs. 6.1999397)
Spieler 7 mit Spielstaerke: 95 hat bei RoundLiga 296472/5000000 (5.92944%) Spielen gewonnen, bei K.O. 310915/5000000 (6.2183%) Spielen gewonnen(5.92944 vs. 6.2183)
Spieler 8 mit Spielstaerke: 95 hat bei RoundLiga 277561/5000000 (5.55122%) Spielen gewonnen, bei K.O. 310768/5000000 (6.21536%) Spielen gewonnen(5.55122 vs. 6.21536)
Spieler 9 mit Spielstaerke: 95 hat bei RoundLiga 262257/5000000 (5.2451396%) Spielen gewonnen, bei K.O. 310752/5000000 (6.21504%) Spielen gewonnen(5.2451396 vs. 6.21504)
Spieler 10 mit Spielstaerke: 95 hat bei RoundLiga 246990/5000000 (4.9398003%) Spielen gewonnen, bei K.O. 308654/5000000 (6.17308%) Spielen gewonnen(4.9398003 vs. 6.17308)
Spieler 11 mit Spielstaerke: 95 hat bei RoundLiga 233158/5000000 (4.66316%) Spielen gewonnen, bei K.O. 309715/5000000 (6.1942997%) Spielen gewonnen(4.66316 vs. 6.1942997)
Spieler 12 mit Spielstaerke: 95 hat bei RoundLiga 219773/5000000 (4.39546%) Spielen gewonnen, bei K.O. 310351/5000000 (6.20702%) Spielen gewonnen(4.39546 vs. 6.20702)
Spieler 13 mit Spielstaerke: 95 hat bei RoundLiga 209488/5000000 (4.1897597%) Spielen gewonnen, bei K.O. 310351/5000000 (6.20702%) Spielen gewonnen(4.1897597 vs. 6.20702)
Spieler 14 mit Spielstaerke: 95 hat bei RoundLiga 199175/5000000 (3.9834998%) Spielen gewonnen, bei K.O. 310613/5000000 (6.21226%) Spielen gewonnen(3.9834998 vs. 6.21226)
Spieler 15 mit Spielstaerke: 95 hat bei RoundLiga 189391/5000000 (3.78782%) Spielen gewonnen, bei K.O. 310471/5000000 (6.20942%) Spielen gewonnen(3.78782 vs. 6.20942)
In dieser Aufstellung gewinnt der beste Spieler (Spielstaerke: 100 ) mit einer hoeheren Warscheinlichkeit in einem RoundLiga Spiel
PS [redacted]\08_Informatik\BWINF\Aufgabe3\executable>
```

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

```java
import java.io.*;
import java.util.*;

public class RoundLiga
{
    public static final boolean DEBUG = false; //debug var

    public ArrayList<Player> players; /* Spieler-Array */
    public ArrayList<Game> games = new ArrayList<Game>(); /* Array, wo alle moeglichen Spiele gespeichert werden */


    /*
     * Konstruktor - Initialisiert das Array Games
     */
    public RoundLiga(ArrayList<Player> players)
    {
        this.players = players;

        /* Array mit allen moeglichen Spieler-Kombinationen erstellen */
        ArrayList<Integer[]> combinations = permutations(players.size());

        /* Fuer jede von diesen Kombinationen ein Spiel zwischen den Spielen erstellen */
        for(int i = 0; i < combinations.size(); i++){
            Integer[] tmp = combinations.get(i);
            games.add( new Game(players.get(tmp[0]-1) , players.get(tmp[1]-1)) ); // (Die -1 ist noetig, da das Players Array bei 0 anfängt, d.h. das [1,2] ein Spiel zwischen Spieler 0 und Spieler 1 erstellt)
        }
    }

    /*
     * Methode, die beliebig viele Runden Simuliert, ruft die Methode playRound() auf
     * - Gibt ein Array zurück, welches die Anzahl von Gewinnen pro Spieler enthaelt
     */
    public ArrayList<Integer> playNRounds(int n)
    {
        System.out.println("Games per Round: nCr(" + players.size() + ", 2) = " + (fact(players.size())/ (fact(2)*fact(players.size()-2))) + ", with " + n + " rounds"); /* INFO Nachicht */

        ArrayList<Integer> r = new ArrayList<Integer>(Collections.nCopies(players.size(),0));  /* Gewinn-Anzahl-Array */

        /*
         * n-mal eine Runde spielen, und den Gewinner notieren
         */
        for(int i = 0; i < n; i++){

            /* Runde Spielen */
            int winner = playRound();

            /* Der index von r entspricht dem Spieler, und der Wert von r[index] der Anzahl der Gewinne */
            r.set(winner, r.get(winner) + 1);
        }
        return r; /* Array mit der Anzahl der Gewinne zurueckgeben */
    }
```

```
    /* Spielt eine Runde */
    private int playRound()
    {

        /* Alle Wins zuruecksetzen */
        clearWins();

        /* Alle Games einmal durchfuhren */
        for(int i = 0; i < games.size(); i++){
            games.get(i).runGame(); //run all games
        }

        /* Den Sieger ermitteln (der mit den meisten wins) */
        int winner = 0;
        for(int i = 1; i < players.size(); i++){
            if(players.get(winner).wins < players.get(i).wins)winner = i;
        }
        return winner; /* winner zurueckgeben */
    }


    /* Setzt die temporaeren Gewinnzaehler von allen Spielen zurueck */
    private void clearWins()
    {
        for(int i = 0; i < players.size(); i++)players.get(i).wins = 0;
        if(DEBUG)System.out.println("----CLEARING SCORES ----");
    }

    /* Methode, um eine Beliebige Zahl zu Faktorisieren */
    private static Long fact(int i)
    {
            if(i <= 1) {
                    return 1;
            }
           return i * fact(i - 1);
    }

    /*
     * Methode, welche alle Mögliche Kombinationen von 1 bis zur uebergebenen Zahl (count) ermittelt und diese
     * in einem Array zurueckgibt - wird benutzt um Alle gegen Alle einmal spielen zu lassen
     */
    private ArrayList<Integer[]> permutations(int count)
    {
        /* Pool erstellen mit allen Zahlen */
        ArrayList<Integer> pool = new ArrayList<Integer>();
        for(int i = 1; i <= count; i++)
        {
            pool.add(i);
        }

        /* Alle Kombinationen durchgehen, und diese in das Array r einfuegen */
        ArrayList<Integer[]> r = new ArrayList<Integer[]>();
        for(int i = 1; i < count; i++)
        {
            for(int x = 1; x < pool.size(); x++)
            {
                Integer tmp[] = {i, pool.get(x)};
                r.add(tmp);
            }
            pool.remove(0);
        }
        return r;
    }
}
```

RoundKO Klasse

```java
import java.util.*;

public class RoundKO
{
    public static final boolean DEBUG = false; //debug var

    public ArrayList<Player> players;
    public ArrayList<Game> games = new ArrayList<Game>();
    Random rand = new Random();

    public RoundKO(ArrayList<Player> players)
    {
        this.players = players;
    }

    public ArrayList<Integer> playNRounds(int n)
    {
        ArrayList<Integer> r = new ArrayList<Integer>(Collections.nCopies(players.size(),0));
        for(int i = 0; i < n; i++)
        {
            RoundKO node = new RoundKO(players);
            node.initFirstRound();
            int winner = node.playRound();
            r.set(winner, r.get(winner) + 1);
        }
        return r;
    }

    private void initFirstRound()
    {
         ArrayList<Integer[]> combinations = permutations(players.size());
        for(int i = 0; i < combinations.size(); i++)
        {
            Integer[] tmp = combinations.get(i);
            games.add( new Game(players.get(tmp[0]-1) , players.get(tmp[1]-1)) );
        }
    }

    private void initNthRound()
    {
        for(int i = 0; i < players.size(); i = i + 2)
        {
            games.add( new Game(players.get(i), players.get(i+1)) );
        }
    }

    private int playRound()
    {
        if(players.size() == 2)
        {
            if(DEBUG)System.out.println("We have reached the Final Stage");
            Game finale = new Game(players.get(0), players.get(1));
            return finale.runGame().identifier;
        }
        ArrayList<Player>firstRoundWinners = new ArrayList<Player>();
        for(int i = 0; i < games.size(); i++){
            firstRoundWinners.add(games.get(i).runGame()); //run all games, add winners
        }
        if(DEBUG)System.out.println("Descending, round complete");
        RoundKO nextRound = new RoundKO(firstRoundWinners);
        nextRound.initNthRound();
        return nextRound.playRound();
    }
```

```
    private ArrayList<Integer[]> permutations(int count)
    {
        assert count%2 == 0 : "Invalid number of Players, must be a multiple of 2";

        ArrayList<Integer> pool = new ArrayList<Integer>();
        for(int i = 1; i <= count; i++)
        {
            pool.add(i);
        }

        ArrayList<Integer[]> r = new ArrayList<Integer[]>();
        while(pool.size() != 0)
        {
            int z = rand.nextInt(pool.size());
            int y = pool.get(z);
            pool.remove(z);

            z = rand.nextInt(pool.size());
            int x = pool.get(z);
            pool.remove(z);

            Integer[] tmp = { x,y };
            r.add(tmp);
        }
        return r;
    }
}
```

Game Klasse

```java
import java.util.*;

/*
 * Klasse, welche ein Spiel zwischen zwei Spielern simuliert
 */
public class Game
{
    public static final boolean DEBUG = false;
    Player t1, t2;

    /* Konstruktor */
    public Game(Player t1, Player t2)
    {
        this.t1 = t1;
        this.t2 = t2;
    }

    /* Zufallszahl wuerfeln */
    private int getRandomNumber(int min, int max)
    {
        return (int) ((Math.random() * (max - min)) + min);
    }

    /* Methode, die einen Gewinner zufällig ermittelt (Urnenprinzip) */
    private Player chooseWinner()
    {
        if(t1.strength >= getRandomNumber(1, t1.strength + t2.strength))return t1;
        return t2;
    }

    /*
     * Spiel Simulieren
     */
    public Player runGame()
    {
        Player winner = chooseWinner(); /* Gewinner auslosen */

        if(DEBUG)System.out.println("Game beween strength " + t1.strength + " and " + t2.strength + ", WINNER " + winner.strength);

        winner.addWin(); /* einen win hinzufuegen - wichtig fuer liga */
        return winner; /* gewinner zurueckgeben - wichtig fuer K.O. */
    }
}
```

Player Klasse

```java
/* Klasse, um einen Spieler zu Simulieren */
public class Player
{
    public static final boolean DEBUG = false;
    public int wins = 0; /* Temporärer Win-Counter */
    public int strength = 0; /* Spielerstaerke */
    public int identifier = 0; /* Einzigartiger Identifier */

    public Player(int strength, int wins, int identifier) /* Konstruktor */
    {
        this.strength = strength;
        this.wins = wins;
        this.identifier = identifier;
        if(DEBUG)System.out.println("Created Team with Strength: " + strength + " and wins " + wins);
    }

    public void addWin() /* wins = wins + 1 */
    {
        wins++;
        if(DEBUG)System.out.println("Added Win to Team with Strength: " + strength + " now at " + wins);
    }
}
```

SolverInput Klasse

```java
public class SolverInput
{
    public static final int n = 5000000; //Anzahl der Runden (Am besten eine große Zahl)
    private Scanner stdin = new Scanner(System.in); //Scanner to get File name from user
    ArrayList<Player> players = new ArrayList<Player>(); // Array um Spieler und ihre Staerke festzuhalten

    public String readInput(String prepend) throws IOException //Input von STDIN einlesen
    {
        System.out.println(prepend);
        return stdin.nextLine();
    }

    public Scanner getFileScannerHandle(String filename) //Open File handle
    {
        Scanner reader = null;
        try
        {
            reader = new Scanner(new FileInputStream(filename), "utf-8");
        }
        catch(FileNotFoundException e)
        {
            System.out.println("Cannot open File, maybe doesnt exist?");
            System.exit(1); //wenn wir die Datei nicht oeffenen koennen, dann programm terminieren
        }
        return reader;
    }

    /*
     * Konstruktor, liest eine Datei mit den Spielerstaerken ein, und erstellt aus diesem Spieler, die in das Player Array eingefuegt werden
     */
    public SolverInput() throws Exception
    {
        Scanner reader = getFileScannerHandle(readInput("Bitte Spielstaerke-Datei name angeben:"));
        int player_count = reader.nextInt();
        System.out.println("Anzahl der Spieler: " + player_count);

        for(int i = 0; i < player_count; i++)
        {
            players.add(new Player(reader.nextInt(),0,i));
        }
    }

    /* Ausfuehrung */
    public void run()
    {

        /* RoundLiga N-mal spielen */
        RoundLiga x = new RoundLiga(players);
        ArrayList<Integer> liga_result = x.playNRounds(n);

        /* K.O. N-mal spielen */
        RoundKO y = new RoundKO(players);
        ArrayList<Integer> ko_result = y.playNRounds(n);


        int best_player = 0;
        for(int i = 0; i < players.size(); i++)
        {
            /* Merken des Allgemein Besten Spielers (hoechste Spielstaerke) */
            if(players.get(best_player).strength < players.get(i).strength)best_player = i;

            /* Ausgeben, wie jeder Spieler Abgeschnitten hat in RoundLiga und K.O. */
            System.out.println("Spieler " + players.get(i).identifier +" mit Spielstaerke: " + players.get(i).strength + " hat bei RoundLiga " + liga_result.get(i) +  "/" + n + " (" + (float) liga_result.get(i)
        }

        /* Empfehlung an Tobi geben, welche Variante mit einer hoeheren Warscheinlichkeit den Besten Spieler hervorbringt */
        if(liga_result.get(best_player) > ko_result.get(best_player)){
            System.out.println("In dieser Aufstellung gewinnt der beste Spieler (Spielstaerke: " + players.get(best_player).strength + " ) mit einer hoeheren Warscheinlichkeit in einem RoundLiga Spiel");
        } else {
            System.out.println("In dieser Aufstellung gewinnt der beste Spieler (Spielstaerke: " + players.get(best_player).strength + " ) mit einer hoeheren Warscheinlichkeit in einem K.O. Spiel");
        }
```