Aufgabe 4: Würfelglück

Team-ID: 00070

Team: Gaußgamz

Bearbeiter/-innen dieser Aufgabe:
Manuel Pielka, Benedict Weis, Tina Balaban, Sinan Ayhan

26. Oktober 2021

Inhaltsverzeichnis

Lösungsidee

Das Programm soll aus einer vorgegebenen Liste von Würfeln einen Würfel bestimmen, welcher sich am besten für das Gewinnen des Spiels “Mensch ärgere dich nicht” eignet. Dafür werden mehrere Partien des Spiels simuliert, wobei jeder Würfel gegen jeden antritt, um so zu vergleichen, welcher am meisten gewinnt.

Umsetzung

Um eine Spielsimulation durchführen zu können, gibt es die Klasse “Player”, welche die Grundmechaniken von “Mensch ärgere dich nicht” ausführt, Würfel ohne und nur mit Sechsen aussortiert, da es mit diesen Würfeln nicht möglich ist das Spiel zu gewinnen, und Gewinner bestimmt. Die Spieler spielen hierbei nicht auf dem selben Spielbrett, sondern auf ihrem eigenen. Um zu überprüfen, ob ein Spieler seinen Gegner geschlagen hat, werden ihre Positionen mit der Variable “delta” verglichen. Eine Spielsimulation besteht aus zwei Runden, in der ersten fängt einer der Spieler an, in der zweiten der Andere.

In der Klasse “App” werden die Spieler bzw. die Würfel ihren Gegnern zugeordnet und die Ergebnisse, also die Anzahl gewonnener Partien, der einzelnen Würfel in einem Array gesammelt und bei Spielende zusammen mit dem Würfel, der am besten abgeschnitten hat ausgegeben.

Beispiele

Nehmen wir die Daten von wuerfel0.txt als Beispiel:

Das Programm ordnet Spieler 1 den ersten Würfel aus der Liste zu und Spieler 2 den zweiten Würfel. Der erste Würfel hat die Augenzahlen 1,2,3,4,5 und 6, der zweite hat 1,1,1,6,6,6.

Nun werden Spieler 1 und 2 sich als Gegner zugeteilt und 500 Spielsimulationen ausgeführt, also insgesamt 1000 Partien. Die Anzahl der gewonnenen Partien jedes Würfels werden im Array „wins“ gesammelt.

Dann wird Spieler 2 der dritte Würfel zugeteilt, welcher die Augenzahlen 1,2,3 und 4.

Da dieser Würfel nicht die Augenzahl 6 besitzt, wird er aussortiert, weil es nicht möglich ist mit ihm zu gewinnen.

Sobald der erste Würfel gegen alle anderen gespielt hat, spielt der dritte Würfel gegen die restlichen usw.

Da am Ende ausgegebene Statistik lautet:

dice 1: 3878

dice 2: 0

dice 3: 1682

dice 4: 1826

dice 5: 784

best dice: 1

Die weiteren Beispieldateien werden mit der selben Vorgehensweise bearbeitet, daher hier nur ihre Ausgaben:

wuerfel1.txt:

dice 0: 1767

dice 1: 3864

dice 2: 0

dice 3: 1672

dice 4: 1881

dice 5: 816

best dice: 1

wuerfel2.txt:

dice 0: 2

dice 1: 1023

dice 2: 2015

dice 3: 2995

dice 4: 3965

best dice: 4

wuerfel3.txt:

dice 0: 3426

dice 1: 2424

dice 2: 3058

dice 3: 2307

dice 4: 2636

dice 5: 1149

best dice: 0

Quellcode

Klasse Player:

```
public class Player {
  int[] die;
  int[] pieces;
  int delta;
  Player opponent;

  public Player(int[] die, int delta) {
    this.die = die;
    this.pieces = new int[] { 0, 0, 0, 0 };
    this.delta = delta;
  }
[...]
public boolean takeTurn() {
    int number = rollDice();
    int farthestPiece = -1;
    int farthestPieceIndex = 0;
    for (int i = 0; i < pieces.length; i++) {
```

```
      if (number == 6 && pieces[i] == 1 && takenPlace(pieces[i] + number) == -1) {
        pieces[i] += number;
        kickOutOpponent(pieces[i]);
        takeTurn();
        return false;
      }
      if (number == 6 && pieces[i] == 0 && takenPlace(1) == -1) {
        pieces[i] = 1;
        kickOutOpponent(pieces[i]);
        takeTurn();
        return false;
      }
      if (pieces[i] > farthestPiece && pieces[i] + number < 53 && takenPlace(pieces[i] + number) == -
1) {
        farthestPiece = pieces[i];
        farthestPieceIndex = i;
      }
    }
    if (farthestPiece > 0) {
      pieces[farthestPieceIndex] += number;
      kickOutOpponent(pieces[farthestPieceIndex]);
    }
    if (number == 6 && !gameWon())
      takeTurn();
    return gameWon();
}
[...]
  public void setOpponent(Player opponent) {
    this.opponent = opponent;
  }
  public boolean faultyDice() {
    boolean containsSix = false;
    for (int i : die) {
      if (i == 6)
        containsSix = true;
    }
    if (!containsSix)
      return true;
    for (int i : die) {
      if (i != 6)
        return false;
    }
    return true;
  }
  public boolean possibleWin() {
    for (int piece : pieces) {
      for (int value : die) {
```

```
      if (piece + value < 44 && takenPlace(piece + value) == -1) {
        return true;
      }
    }
  }
  return false;
 }
}
```

Klasse App:

```
public class App {
 static ArrayList<int[]> dice = new ArrayList<int[]>();
 public static void main(String[] args) {
  [...]
  for (int i = 1; i < lines.size(); i++) {
   String[] faces = lines.get(i).split(" ");
   int[] die = new int[faces.length - 1];
   for (int j = 1; j < faces.length; j++) {
    die[j - 1] = Integer.parseInt(faces[j]);
   }
   dice.add(die);
  }
  int[] wins = new int[dice.size()];
  for (int j = 0; j < dice.size(); j++) {
   for (int i = j + 1; i < dice.size(); i++) {
    playRound(j, i, wins, 500);
   }
  }
  HashMap<Integer, Integer> sameWins = new HashMap<Integer, Integer>();
  boolean inProgress = true;
  while (inProgress) {
   sameWins.clear();
   for (int i = 0; i < dice.size(); i++) {
    if (sameWins.containsKey(i))
     playRound(i, sameWins.get(i), wins, 2);
   }
   for (int i = 0; i < dice.size(); i++) {
    for (int j = i + 1; j < dice.size(); j++) {
     if (wins[i] == wins[j] && wins[i] != 0)
      sameWins.put(i, j);
    }
   }
   if (sameWins.isEmpty())
    inProgress = false;
  }
 }
 private static void playRound(int id1, int id2, int[] wins, int rounds) {
```

```
    for (int k = 0; k < rounds; k++) {

      Player p1 = new Player(dice.get(id1), -20);
      Player p2 = new Player(dice.get(id2), 20);
      if (p1.faultyDice())
        break;
      if (p2.faultyDice())
        break;
      p1.setOpponent(p2);
      p2.setOpponent(p1);
      while (true) {
        if (p1.takeTurn()) {
          wins[id1]++;
          break;
        }
        if (p2.takeTurn()) {
          wins[id2]++;
          break;
        }
        if (!p1.possibleWin() && !p2.possibleWin())
          break;
      }
      p1 = new Player(dice.get(id1), -20);
      p2 = new Player(dice.get(id2), 20);
      p1.setOpponent(p2);
      p2.setOpponent(p1);
      while (true) {
        if (p2.takeTurn()) {
          wins[id2]++;
          break;
        }
        if (p1.takeTurn()) {
          wins[id1]++;
          break;
        }
        if (!p1.possibleWin() && !p2.possibleWin())
          break;
      }
    }
  }
}
```