Humboldt-Universität zu Berlin
Institut für Mathematik
Dr. L. Lehmann; Dr. Antoine Laurain, Dipl.Math Torsten Bosse

Übungsaufgaben zur Vorlesung Mathematikorientierte Computernutzung (SS 11) Das Tic Tac Toe Spiel

Keine Abgabe

Aufgabe 3.1:

Implementieren Sie das Spiel *'Tic Tac Toe'* für zwei menschliche Spieler. Die Eingabe sollte hierbei abwechselnd über die Shell geschehen. Benutzen Sie das Turtle Paket um das Spiel zu visualisieren, d.h. die 'Turtle' sollte zuerst ein 3*3 Spielfeld und danach abwechselnd Kreuze bzw. Kreise malen, welche den Positionen der Spielereingaben entsprechen. Strukturieren Sie ihr Programm mittels Funktionen und Klassen. Wir empfehlen Ihnen den folgenden Aufbau:

```
/* tictactoe.java - Tic Tac Toe mit 2 menschlichen Spielern */
/* an den Stellen '...' muss noch eigener Code eingefuegt werden*/
import ch.aplu.turtle.*;
import java.awt.Color;
import java.util.*;

class playground{
  Turtle om = new Turtle();
  int Position1=0;
  int Position2=0;
  ...
  int Position9=0;

  /*Zeichne das 3*3 Gitter mit Turtle om*/
  public void ZeichneSpielfeld(){...}
  /*Zeichne ein X in das Feld p in 1,2,3,4,5,6,7,8,9*/
  public void ZeichneX(int p){...}
  /*Zeichne einen Kreis in das Feld p in 1,2,3,4,5,6,7,8,9*/
  public void ZeichneO(int p){...}
  /*Pruefe, ob das Feld p schon belegt ist*/
  public int  PruefeFeld(int p){...}
  /*Pruefe, ob ein Spieler gewonnen hat*/
  public int  PruefeGewonnen(){...}
  }

class tictactoe{
  public static void main(String[] args){
    playground pl= new playground();
    ...
    }
  }
```

Dabei enthält das obige Programm *tictactoe.java* die zwei Klassen *playground* und *tictactoe*. Innerhalb der *'main'*-Funktion der Klasse *tictactoe* können Sie nach der Zeile *'playground pl = ...'*

auf die Funktionen und Variablen des Objekts *pl* vom Typ playground durch 'pl.Funktion/Variable' zugreifen. Zum Beispiel wird die Funktion *ZeichneX* in *main* durch *'pl.ZeichneO(9)'* aufgerufen und zeichnet einen Kreis auf Feld 9.

Im folgenden finden Sie weitere Erklärungen über die vorgegebene Struktur:

1. Die Funktion *'main'* Funktion sollte den Hauptcode enthalten, welcher den Programmablauf steuert. Benutzen Sie dafür eine while Schleife in welcher die Eingabe beider Spieler mittels einer geeigneten Mitteilung ('Kreuz spielt:' oder 'Spieler A:') abwechselnd abfragt werden - solange noch nicht alle Felder belegt sind und keiner der Spieler gewonnen hat. Rufen Sie entsprechend der Eingabe und der aktuellen Situation die passende Funktion der Klasse *playground* auf.

2. Die Funktion *ZeichneSpielfeld* sollte das Spielfeld zeichnen: zwei horizontale Linien und zwei vertikale Linien. Wählen Sie hierbei eine vernünftige Größe für die Spielfelder. Das Zentrum des Spielfelds sollte die Koordinaten $(0, 0)$ haben. Sie können die Funktionen *om.setPos(x,y)* und *om.setH(winkel)* (siehe Turtle-Dokumentation, Google) verwenden, um ein besondere Anfangsstelle und Anfangswinkel für die Schildkröter zu wählen.

3. Die Unterfunktionen *ZeichneX* und *Zeichne0* erhalten einen integer Parameter p und malen einen Kreuz bzw. ein Kreis an die entsprechende Position. Dabei sollten die neun Felder von dem Spiel mit Zahlen von 1 bis 9 codiert sein. (es gibt aber auch andere Möglichkeiten).

4. Speichern Sie den Status, ob ein Feld i besetzt ist durch das Zuweisen eines Wertes an die entsprechende *Positions*-Variable: Z.B. *Position3=1;* fuer Position 3 ist Kreuz oder *Position1=2* fuer Position 1 ist Kreis

5. Die *PruefeFeld* Unterfunktion prüft, ob ein Stelle schon besetzt ist und sollte vor ZeichneX oder ZeichneY aufgerufen werden. Benutzen Sie hierfür wieder die *Positions*-Variablen und einen geeigneten Rückgabewert. Im Falle einer ungültigen Eingabe sollte eine Mitteilung 'Schon besetzt!' auf dem Bildschirm erscheinen und erneut eine Abfrage stattfinden.

6. Die Funktion *PruefeGewonnen* testet, ob ein Spieler gewonnen hat. Es gibt acht mögliche Gewinnstellungen bei tictactoe, diese können mittels einer if-Abfrage und der *Positions*-Variablen geprüft werden. Falls alle Felder schon besetzt sind und es noch kein Sieger gibt, so sollte die Funktion einen geeigneten Wert zurückliefern und der Algorithmus mit einer entsprechenden Mitteilung stoppen .

7. Das Programm sollte sich zu jedem Zeitpunkt und bei jeder Eingabe 'sinnvoll' verhalten. Die Geschwindigkeit der Schildkröte kann mittels *om.speed(1000)* erhöht werden.

Die vorgegebene Struktur ist nur ein Vorschlag, d.h. andere Lösungswege werden auch akzeptiert (solange diese sich mit dem Thema 'Funktionen' in Java beschäftigen). Fortgeschrittene Teilnehmer dürfen gerne auch eine Einspieler–Version mit dem Computer als Gegner einreichen.

Wir wünschen Ihnen viel Spaß!