SoSe 2014
Prof. Dr. Margarita Esponda
ALP2
7. Übungsblatt (Abgabe am 10. Juni)

Ziel: Auseinandersetzung mit Objektorientierten Konzepten wie Klassendefinitionen, Konstruktoren, Objekterzeugung, Instanz-Variablen und -Methoden.

1. Aufgabe (8 Punkte)

Definieren Sie eine **Intervall**-Klasse in Java, die die Menge aller Elemente **x**, die sich zwischen zwei Grenzwerten **a** und **b** befinden, darstellt.

Ihre Intervall-Klasse soll ein geschlossenes Intervall darstellen. D.h. $a \leq x \leq b$. Die Menge ist leer, wenn b<a ist.

a) Definieren Sie folgenden Konstruktor in Ihrer Klasse.

```
public Intervall (double a, double b)
```

b) Definieren Sie folgende Methoden:

`public boolean enthaelt(double x)`	überprüft, ob x im Intervall ist
`public boolean schneidet(Intervall cd)`	ein Intervall-Objekt überprüft, ob es sich mit einem zweiten Intervall-Object **cd** schneidet.
`public boolean beinhaltet(Intervall cd)`	ein Intervall-Objekt testet, ob es das eingegebene Intervall cd vollständig beinhaltet.
`public double laenge()`	berechnet die Länge des Intervalls.
`public String toString()`	das Intervall soll als Text dargestellt werden.

c) Programmieren Sie eine **TestIntervall**-Klasse, in der Sie Intervall-Objekte erzeugen und alle Methoden der **Intervall**-Klasse austesten.

2. Aufgabe (10 Punkte)

Definieren Sie eine Klasse **Gambler** mit der Sie Glücksspieler simulieren können.

a) Definieren Sie sinnvolle Instanzvariablen für die **Gambler** Klasse, wie z.B. Name, Startgeld usw. und geeignete Konstruktoren, die alle Instanzvariablen initialisieren.

b) Eine Methode **bet**, die eine Wette simuliert, bei der ein Glücksspieler mit 0,5 Wahrscheinlichkeit ein Euro gewinnt oder verliert, soll programmiert werden.

c) Eine zweite **play-**Methode soll einen Gambler spielen lassen, bis er sein ganzes Bargeld verliert.

d) Definieren Sie eine **Casino**-Klasse, die ein Spielhaus simuliert, das **n** Spieler initialisiert. Die **Casino** Klasse soll eine **getMoney**-Methode haben, die so lange die Gambler gegen das Casino wetten lässt, bis alle Ihr gesamtes Startgeld verlieren.

e) Stellen Sie mit Hilfe geeigneter Ausgabe den Verlauf des Spiels dar, bis alle Spieler pleite sind.

3. Aufgabe (14 Punkte)

In dieser Aufgabe möchten wir das Spiel aus der 3. Aufgabe des 3. Übungsblatts in Java simulieren.

Starten Sie mit der Definition einer **PlayField** Klasse, die als Instanz-Variable eine **n**x**m** Matrix von **Position**-Objekte beinhaltet.

Die **Position** Klasse soll jede Position des Spielfeld darstellen. Eine **Position** ist aufgedeckt oder verdeckt, hat eine bestimmte Anzahl von benachbarten Löchern und enthält ein Loch oder kein Loch.

Folgende Instanzvariablen soll die Position-Klasse mindestens haben:

```
public class Position {
        int holeNeighbours = 0;
        int hole = 0;
        boolean open;
        ...
}
```

a) Definieren Sie zuerst einen Konstruktor, der als Argument eine Wahrscheinlichkeit **p** und die (**n**, **m**) Dimensionen der Matrix bekommt. Auch hier soll **p** die Wahrscheinlichkeit, dass ein Loch an der (x,y)-Position vorkommt, darstellen. Ein zweiter Konstruktor ohne Argumente soll das Spielfeld mit Defaultwerten initialisieren.

Die Positionen mit Löchern haben in der **hole** Instanzvariable eine 0 oder eine 1, um die benachbarten Löcher einer bestimmten Position einfacher berechnen zu können.

b) Definieren Sie folgende Methoden für die **PlayField**-Klasse:

`public void clearField()`	Alle Positionen bekommen 0 benachbarten Löcher, werden auf nicht open gesetzt und alle Löcher werden gelöscht.
`public void initField()`	Löcher werden mit Wahrscheinlichkeit **p** produziert.
`private int calculateNeighbours(int i, int j)`	berechnet die Anzahl benachbarten Löcher an der Position **i**, **j** der Matrix.
`public void generateSolution()`	berechnet alle benachbarten Löcher der Matrix.
`public String toScring()`	Das Spielfeld wird in ein String verwandelt, damit dieser mit der print-Methode ausgegeben werden kann.

c) Programmieren Sie eine geeignete **TestPlayField**-Klasse.

4. Aufgabe (8 Bonuspunkte)
Ergänzen Sie die 3. Aufgabe um eine **Player** Klasse und eine **HolePlay** Klasse.

Die **Player**-Klasse soll Spieler darstellen, die Positionen aufdecken können und dabei Punkte gewinnen. Die Player gewinnen Punkte, indem sie Zahlen von benachbarten Löchern der aufgedeckten Positionen zusammenaddiert bekommen.

Die **HolePlay** Klasse startet ein Array mit mehreren Player und simuliert ein Spiel, bis alle Spieler verlieren oder am Ende alle Positionen aufgedeckt sind und der Spieler mit der größten Punktzahl gewinnt.

Wichtige Hinweise:

1) **Verwenden Sie selbsterklärende Namen von Variablen und Methoden.**
2) **Für die Name aller Bezeichner müssen Sie die Java-Konventionen verwenden.**
3) **Verwenden Sie vorgegebene Klassen und Methodennamen.**
4) **Methoden sollten klein gehalten werden, sodass auf den ersten Blick ersichtlich ist, was diese Methode leistet.**
5) **Methoden sollten möglichst wenige Argumente haben.**
6) **Methoden sollten entweder den Zustand der Eingabeargumente ändern oder einen Rückgabewert liefern.**
7) **Verwenden Sie geeignete Hilfsvariablen und definieren Sie sinnvolle Hilfsmethoden in Ihren Klassendefinitionen.**
8) **Zahlen sollten durch Konstanten ersetzt werden.**
9) **Löschen Sie alle Programmzeilen und Variablen, die nicht verwendet werden.**