Bereich: Klassen (3)

Nimmspiel

Package: `de.dhbwka.java.exercise.classes`	**Klasse:** `Nimmspiel`

Aufgabenstellung:

Ein Spiel, nennen wir es Nimm-Spiel, hat folgende Regeln: Auf dem Tisch liegen zwei Haufen mit je einer beliebigen Anzahl an Kugeln. Zwei Spieler ziehen abwechselnd. Sie müssen dabei jeweils von einem (nicht-leeren) Haufen eine beliebige Anzahl (> 0 und <= Anzahl der Kugeln auf dem Haufen) Kugeln entfernen. Es gewinnt derjenige Spieler, der die letzten Kugeln vom Tisch räumt.

Beispiel:

1. Haufen: 5 Kugeln; 2. Haufen: 3 Kugeln

Spieler A: nimmt vom 1. Haufen 2 Kugeln

Spieler B: nimmt vom 2. Haufen 2 Kugeln

Spieler A: nimmt vom 2. Haufen 1 Kugel

Spieler B: nimmt vom 1. Haufen 3 Kugeln und hat gewonnen!

Implementieren Sie eine Klasse, die dieses Nimm-Spiel realisiert. Überlegen Sie, welche Attribute die Klasse auf jeden Fall haben muss (Spieler, Anzahl der Kugel auf jedem Haufen, welcher Spieler als nächster an der Reihe ist).

Schreiben Sie dazu einen Konstruktor, der als Parameter die Namen der beiden Spieler erhält. Im Konstruktor soll die Anzahl der Kugel eines jeden Haufens beliebig erzeugt werden. (Benutzen Sie dazu die Methode `Math.random()`).

Implementieren Sie geeignete Methoden (für die Ausgabe (`toString`), für einen Spielzug, zum Überprüfen, ob alle Kugeln entfernt sind etc.), so dass zwei menschliche Spieler gegeneinander spielen können. Geben Sie die Zwischenstände (Anzahl verbliebener Kugeln pro Haufen) nach jedem Spielzug in geeigneter Form auf den Bildschirm aus, sowie am Schluss eine Nachricht welcher Spieler gewonnen hat. Sehen Sie in Ihrem Programm vor, dass fehlerhafte Benutzereingaben (wie das Entfernen von Kugeln von einem leeren Haufen) wiederholt werden müssen!

Beispielausgabe:

```
Spieler: Alf und Ben, Haufen 1: 5 Kugel(n), Haufen 2: 7 Kugel(n)
Spieler Alf: Von welchem Haufen ziehen Sie Kugeln? 1
Spieler Alf: Wieviele Kugeln ziehen Sie? 4
Spieler: Alf und Ben, Haufen 1: 1 Kugel(n), Haufen 2: 7 Kugel(n)
Spieler Ben: Von welchem Haufen ziehen Sie Kugeln? 2
Spieler Ben: Wieviele Kugeln ziehen Sie? 6
Spieler: Alf und Ben, Haufen 1: 1 Kugel(n), Haufen 2: 1 Kugel(n)
Spieler Alf: Von welchem Haufen ziehen Sie Kugeln? 1
Spieler Alf: Wieviele Kugeln ziehen Sie? 1
```

```
Spieler: Alf und Ben, Haufen 1: 0 Kugel(n), Haufen 2: 1 Kugel(n)
Spieler Ben: Von welchem Haufen ziehen Sie Kugeln? 2
Spieler Ben: Wieviele Kugeln ziehen Sie? 1
Spieler: Alf und Ben, Haufen 1: 0 Kugel(n), Haufen 2: 0 Kugel(n)
Spiel beendet.
Gewonnen hat Spieler Ben
```

Bereich: Klassen (3)

Lotto

Package: `de.dhbwka.java.exercise.classes`	**Klasse:** `Lotto`

Aufgabenstellung:

Bei einem (vereinfachten) deutschen Lottospiel werden 6 Kugeln aus einer Anzahl von 49 Kugeln gezogen. Implementieren Sie eine Klasse, die ein beliebiges Lottospiel simuliert.

a) Implementieren Sie dazu einen geeigneten Konstruktor der Art:

```
Lotto(int m, int n)
```

Damit soll ein Lottospiel der Art m Kugeln aus n erzeugt werden. Das deutsche Lotto müsste damit wie folgt erzeugt werden:

```
Lotto deutschesLotto = new Lotto(6,49);
```

b) Die Klasse Lotto soll folgende Attribute besitzen:

- ein Attribut, das die Anzahl der zu ziehenden Kugel beschreibt,
- ein Attribut, das die Gesamtzahl der Kugel beschreibt,
- ein Array, das Ihren Tipp enthält,
- ein Array, das die gezogenen Zahlen enthält.

c) Schreiben Sie zu der Klasse Lotto die folgenden Methoden:

```
<Typ> tippen()
```

Diese Methode frägt Sie nach Ihrem Tipp. Die Methode muss sicherstellen, dass die getippten Zahlen im erlaubten Zahlenbereich sind.

```
<Typ> tippen(int[] tipp)
```

Diese Methode fragt Sie nicht nach Ihrem Tipp sondern erhält ein Array von Zahlen mit Ihrem Tipp. (Gut geeignet zum Testen der Klasse, damit man nicht immer alle Zahlen eingeben muss.)

```
<Typ> ziehen()
```

Diese Methode führt eine Ziehung der Zahlen durch, d.h. erstellt eine entsprechende Anzahl an zufälligen Zahlen.

```
int richtige()
```

Diese Methode ermittelt, wie viele richtige Zahlen Sie bei Ihrem Tipp hatten.

```
public String toString()
```

Diese Methode gibt den Zustand Ihres Lottospiels aus. Falls sie schon getippt haben, soll Ihr Tipp ausgegeben werden. Falls die Ziehung schon erfolgte, sollen auch die gezogenen Zahlen ausgegeben werden. Die Zahlen sollen dabei in aufsteigender Reihenfolge ausgegeben werden. Am besten sortieren Sie dazu die beiden Arrays, die die Klasse als Attribute besitzt.

Beispielausgabe:

```
Geben Sie bitte Ihren Tipp für die 1. Zahl ein: 22
Geben Sie bitte Ihren Tipp für die 2. Zahl ein: 11
Geben Sie bitte Ihren Tipp für die 3. Zahl ein: 33
Geben Sie bitte Ihren Tipp für die 4. Zahl ein: 1
Geben Sie bitte Ihren Tipp für die 5. Zahl ein: 49
```

```
Geben Sie bitte Ihren Tipp für die 6. Zahl ein: 17
Tipp: 1 11 17 22 33 49

Tipp: 1 11 17 22 33 49
Gezogene Zahlen: 6 11 32 35 47 49

Richtige: 2
```

Bereich: Klassen (3)

MasterMind

Package: `de.dhbwka.java.exercise.classes`	**Klasse:** `MasterMind`

Aufgabenstellung:

Beim bekannten Spiel MasterMind ist das Ziel, eine vom Gegenspieler gewählte Farbkombination herauszufinden. Implementieren Sie eine Version von MasterMind mit textueller Ein-/Ausgabe.

Anstelle von Farben verwenden Sie die Buchstaben A bis H. Gegenspieler ist der Computer. Der Computer soll eine zufällige Kombination von 5 Buchstaben (aus A bis H) auswählen, wobei ein Buchstabe auch mehrfach vorkommen kann.

Sie geben dann dem Computer Ihren Tipp ein, den der Computer bewertet, indem er Ihnen die Anzahl der richtigen Buchstaben auf den richtigen Stellen und die Anzahl der richtigen Buchstaben auf den falschen Stellen angibt.

Dabei soll Ihr Programm nur eine Höchstanzahl an Versuchen zulassen (z.B. 20), in denen der Spieler die richtige Kombination erraten haben muss. Eine mögliche Ausgabe sieht wie folgt aus:

```
Geben Sie ihren Tipp ab: abbcc
1 bisherige Versuche:
ABBCC 0  0

Geben Sie ihren Tipp ab: deeff
2 bisherige Versuche:
ABBCC 0  0
DEEFF 1  1

Geben Sie ihren Tipp ab: dghdd
3 bisherige Versuche:
ABBCC 0  0
DEEFF 1  1
DGHDD 1  3

Geben Sie ihren Tipp ab: dedgh
4 bisherige Versuche:
ABBCC 0  0
DEEFF 1  1
DGHDD 1  3
DEDGH 0  4

Geben Sie ihren Tipp ab: fdgdh
5 bisherige Versuche:
ABBCC 0  0
DEEFF 1  1
DGHDD 1  3
DEDGH 0  4
FDGDH 3  2

Geben Sie ihren Tipp ab: hdgdf
Mit 6 Versuchen gewonnen!
```

Die erste Zahl gibt dabei die Anzahl der richtigen Buchstaben an den richtigen Stellen und die zweite Zahl die Anzahl der richtigen Buchstaben auf falschen Stellen an.

Für die Bestimmung der einzelnen Buchstaben in dem eingelesenen String können Sie die Instanzmethode
`char charAt(int index)` (0 `<=` `index` `<` String-Länge)
der Klasse String verwenden.