Programmieren 1
Aufgabenblatt 2

Wintersemester 2024/25
Prof. Dr. J. Rexilius

Abgabetermin: 31.10.2024, 08:00

Aufgabe 2 (9 Punkte)

Beachten Sie die aktuelle Version zu den allgemeinen Abgabehinweisen (Praktikumsordner im ILIAS).

(a) Berge (**3 Punkte**)

Implementieren Sie eine Methode `zeichneBerge(int w1, int w2, int w3)`, mit der drei Berge nebeneinander auf der Konsole ausgegeben werden (siehe Beispiel). Dabei wird jeder Berg mit einem anderen Zeichen dargestellt. Die Größe soll über die Parameter `w1,w2,w3` festgelegt werden. Diese legen jeweils die Breite der letzten Zeile eines Berges in Anzahl der Zeichen fest, wobei sich in jeder Zeile darüber die Anzahl der Zeichen pro Berg um zwei reduziert und diese entsprechend eingerückt werden. Die Methode soll für alle Eingabeparameter funktionieren für die gilt: $0 < w1, w2, w3 \leq 21; w1, w2, w3\ ungerade.$

Testen Sie Ihr Programm für (`w1=5,w2=3,w3=7`) und (`w1=3,w2=5,w3=9`).

Beispiel: (für w1=3, w2=5, w3=5)

```
      #      o
 *    ###    ooo
*** ##### ooooo
```

(b) Quiz (**3 Punkte**)

In dieser Aufgabe soll ein Programm entwickelt werden, das zufällige Rechenaufgaben mit zwei Operanden erzeugt. Die Operanden liegen im Bereich [1,20]. Als Operatoren soll für jede Aufgabe eine zufällig Auswahl der drei Grundrechenarten `+, -, *` getroffen werden.

Entwerfen Sie zunächst ein Klassendiagramm für eine Klasse `RechenQuiz`, deren Objekte zufällige Rechenaufgaben erzeugen und korrekte Ergebnis dazu bereitstellen. Nutzen Sie dazu die Notation aus der Vorlesung. Implementieren Sie danach Ihre Klasse und testen Sie diese. Nutzen Sie dazu eine separate Testklasse.

Nutzen Sie zur Zufallszahlenberechnung den Befehl `Math.random`, der double-Zufallszahlen zwischen 0.0 (einschließlich) und 1.0 (ausschließlich) generiert.

Die Klasse `RechenQuiz` soll wie folgt verwendet werden können:

```
RechenQuiz quiz = new RechenQuiz();
quiz.getExercise();
int result = quiz.getResult();
System.out.println("Result: "+result);
```

Hinweise:

- Das Programm erzeugt bei jedem Aufruf der Funktion `getExercise()` eine neue Aufgabe.
- Die Funktion `getExercise()` gibt die erzeugte Aufgabe wie folgt auf der Konsole aus:
  `<Operand1> <Operator> <Operand2> = ?`
  Beispiele: 13 * 8 = ? 7 - 5 = ? 20 + 1 = ?
- Die beiden Operanden sind vom Typ `int`.

- Eine Aufgabe ist gültig, wenn das Ergebnis größer oder gleich als 0 ist. Ungültige Aufgaben werden in der Funktion `getExercise()` abgefangen. Es wird solange eine neue, zufällige Aufgabe erstellt, bis diese gültig ist.

(c) Tiere und Pflanzen (**3 Punkte**)

Der Biologe Benno Blütenstaub beobachtet eine Reihe von Pflanzen und Tieren. Seine Erkenntnisse möchte er gerne digital ablegen.

- Zu jeder Pflanze und zu jedem Tier möchte er die Bezeichnung speichern können (z.B. „Gras“, „Zebra“, „Löwe“).
- Zu jeder Pflanze möchte er eine kurze textuelle Beschreibung speichern können (z.B. „grün“, „rot“).
- Zu jedem Tier möchte er gerne speichern können, was es frisst.
- Abhängig davon, was ein Tier frisst, soll es automatisch als Pflanzenfresser, Fleischfresser oder Allesfresser identifiziert werden.

Helfen Sie Benno Blütenstaub, sein Programm zu entwerfen. Skizzieren Sie zunächst das zugehörige Klassendiagramm mit allen Methoden, etc. Nutzen Sie dazu die Notation aus der Vorlesung. Implementieren Sie dann die benötigten Klassen mit ihren Attributen und Methoden.

- Speichern Sie die Nahrung eines Tieres in zwei Parametern vom Typ `String` ab. Einen String für die Tiere, die gefressen werden, und einen String für die Pflanzen. Beide Parameter werden bei der Deklaration direkt mit „-“ belegt.
- Definieren Sie zwei Methoden zum Hinzufügen von Nahrung eines Tieres. Die eine Methode übergibt als Parameter ein Pflanzenobjekt, die andere Methode übergibt als Parameter ein Tierobjekt.
- Definieren Sie eine Methode, die ausgibt ob ein Tier ausschließlich Pflanzen frisst (Pflanzenfresser), ausschließlich Tiere frisst (Fleischfresser), oder beides frisst (Allesfresser) – abhängig von der gespeicherten Nahrung des Tieres.
- Alle Instanzvariablen sind `private`.

Schreiben Sie eine Testklasse `BioTest` mit einer main-Methode, in der Sie folgendes Szenario nachbilden:

- Gras ist grün
- Beeren sind rot
- Algen sind grün
- Fische fressen Algen
- Zebras fressen Gras
- Löwen fressen Zebras
- Bären fressen Beeren
- Bären fressen Fische

Die main-Methode soll zusätzlich folgende Ausgaben erzeugen

```
Zebra ist ein Pflanzenfresser.
Löwe ist ein Fleischfresser.
Bär ist ein Allesfresser.
```