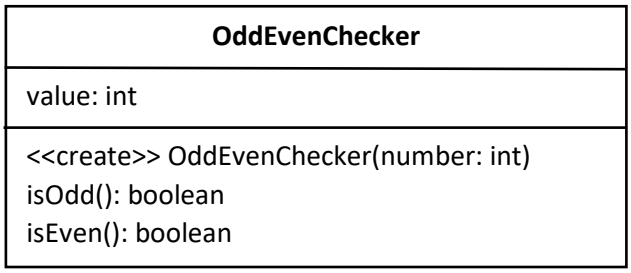
Objektorientierte Programmierung I
freiwillige Übungsaufgaben
03.05.2022

Aufgabe 1

Erstelle eine Java-Klasse anhand des folgenden UML-Klassendiagramms:



Die Methoden `isOdd()` und `isEven()` sollen `true` bzw. `false` zurückgeben, falls der Wert `value` ungerade bzw. gerade ist. Überlege, wie du nach dem DRY-Prinzip („Don't Repeat Yourself") Code-Wiederholungen vermeiden kannst.

Fertige zusätzlich ein Hauptprogramm an, um die Klasse zu testen.

Aufgabe 2

Welche der folgenden Bezeichner entsprechen den Java-Namenskonventionen, wenn eine **Klasse** benannt werden soll? Sind Bezeichner dabei, die den Konventionen entsprechen, aber dennoch eher „unpassend" sind? Wenn ja, warum?

- `NetworkManager`
- `databaseAccessor`
- `network_connection`
- `ConnectToHost`
- `HUMAN_PLAYER`

Welche der folgenden Bezeichner entsprechen den Java-Namenskonventionen, wenn eine **Methode** benannt werden soll? Sind Bezeichner dabei, die den Konventionen entsprechen, aber dennoch eher „unpassend" sind? Wenn ja, warum?

- `move_to_target`
- `CalculateAverage`
- `isEmpty`
- `GROW`
- `databaseAccessor`

Sind die folgenden Variablen passend benannt? Werden die Java-Namenskonventionen verletzt? Sind andere Probleme vorhanden?

- `int Number = 42;`
- `double text = 1.23;`
- `String[] lines = new String[10];`
- `int[] values = new byte[5];`
- `char firstCharacter = 'a';`
- `float[] value = new float[100];`

Aufgabe 3

Gegeben ist der folgende Code:

```
class Test {
    public double value = 0.0;

    public Test() {

    }
}
```

Handelt es sich bei `Test()` um einen Default-Konstruktor?

Aufgabe 4

Schreibe ein Java-Programm, dass die Fakultät (engl. *factorial*) $n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1 = \prod_{k=1}^{n} k$ einer Zahl n berechnet und am Bildschirm ausgibt. Das Programm soll auch den Sonderfall für $n = 0$ ($0! = 1$) korrekt behandeln können.

Das Programm soll die Fakultät von 20 korrekt berechnen können (Kontrollergebnis: $20! = 2\,432\,902\,008\,176\,640\,000$).

Aufgabe 5

Schreibe ein Programm, dass die ersten 20 Elemente der Fibonacci-Folge berechnet und ausgibt. Die ersten beiden Folgenglieder haben den Wert 1. Jedes nachfolgende Element ergibt sich jeweils aus der Summe der beiden Vorgängerelemente. Daraus ergibt sich die Folge $1, 1, 2, 3, 5, 8, 13, \ldots$

Aufgabe 6

Schreibe ein Programm, dass die ganzen Zahlen von 1 bis 100 ausgibt, jedoch mit folgenden Ausnahmen:

- Wenn die Zahl **durch drei teilbar** ist, soll statt der Zahl das Wort „Fizz“ ausgegeben werden.
- Wenn die Zahl **durch fünf teilbar** ist, soll statt der Zahl das Wort „Buzz“ ausgegeben werden.
- Wenn die Zahl **sowohl durch drei als auch durch fünf teilbar** ist, soll statt der Zahl das Wort „Fizzbuzz“ ausgegeben werden.

Die Ausgabe des Programms soll also wie folgt beginnen:

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
```

```
Fizzbuzz
16
```
usw…

Aufgabe 7

Aufbauend auf unserem „Filmsammlung“-Beispiel sei der folgende Code gegeben:

```
Movie[] movies = new Movie[4];
movies[0] = new Movie("Back to the Future", 1985);
movies[1] = new Movie("Interstellar", 2014);
movies[2] = new Movie("Vier Fäuste gegen Rio", 1984);
movies[3] = new Movie("Forrest Gump", 1994);
```

Füge diesen Code in ein entsprechendes Hauptprogramm ein und erweitere das Programm so, dass alle Filme ausgegeben werden, die nach 1990 erschienen sind (ältere Film sollen nicht ausgegeben werden).