Aufgabenzettel 10
Funktionale Programmierung

Dieser Aufgabenzettel muss spätestens in Ihrer Praktikumsgruppe in **KW 28** vorgestellt werden.

Alle für diesen Aufgabenzettel entwickelten Programme sollen sich in einem Paket **de.hsruhrwest.oop.ss2025.assignment10** oder entsprechenden **Unterpaketen** befinden.

Hinweis: Alle Klassen müssen vollständig mit JavaDoc kommentiert sein.

Aufgabe 1: Funktionale Interfaces bestimmen

Geben Sie für die folgenden Ausdrücke einen passenden Datentyp für die linke Seite der Zuweisung an (siehe Vorlesung 37 Folie 13 „Generische funktionale Interfaces in Java“):

a) _____ `f1 = x -> "x";`

b) _____ `f2 = (x, y) -> x*x*x - 3*y*y + 5;`

c) _____
```
f3 = str ->
       System.out.println("*** " + str + "***");
```

d) _____
```
f4 = (str, count) -> {
  for (int i = 0; i < count; i++) {
    System.out.println(str);
  }
};
```

e) _____
```
f5 = str ->
  str.matches(".*[aeiouAEIOU].*");
```

f) _____
```
f6 = (str, number) ->
  str.contains(Integer.toString(number));
```

g) `_____ f7 = () -> "Hello world!";`

h) `_____ f8 = x -> x > 0 ? x : 0;`

i) `_____ f9 = (n, e) -> Math.pow(n, e);`

Aufgabe 2: Funktionale Interfaces verwenden

Schreiben Sie die folgenden statischen Hilfsmethoden:

a) Die Methode **printMatching** erhält eine Liste von Spielern (siehe beigefügte Klasse **Player**) und ein **Predicate** für Spieler. Es gibt alle Spieler aus, die dem Predicate genügen. Beispiel: Der folgende Aufruf gibt alle Spieler aus, deren Punktzahl über 20 liegt.

```
printMatching(playerList, p -> p.getScore() > 20);
```

b) Die Methode **sumAfterMap** erhält eine Liste von Integer-Werten und eine Abbildung für Integer in Form eines **UnaryOperators**. Die Methode wendet den UnaryOperator auf alle Zahlen an und summiert das Ergebnis auf. Beispiel: Der folgende Aufruf gibt den Wert 1674 aus.

```
var numbers = List.of(12, 3, 39);
System.out.println( sumAfterMap(numbers, n -> n * n) );
```

c) Die Methode **getMaximumMappedString** erhält eine Liste von Strings, bildet jeden String mit einer **Function** auf einen Integer-Wert ab und gibt das Maximum zurück. Beispiel: Der folgende Aufruf gibt den Wert 13 aus.

```
var strings = List.of("Alice", "Bob", "Schweinshaxen");
System.out.println(getMaximumMappedString(strings,
  string -> string.length())
);
```

Aufgabe 3: Optionals

Schreiben Sie eine Hilfsmethode **getNextPrimeInRange**. Sie erhält eine Untergrenze und eine Obergrenze und gibt die kleinste Primzahl im Bereich [Untergrenze, Obergrenze] als **Optional** zurück. Gibt es keine Primzahl in dem Bereich, wird ein leeres Optional zurückgegeben.

Aufgabe 4: Streams erstellen

Schreiben Sie Code, um die folgenden Streams zu erstellen.

a) Ein endlicher Stream, der alle Long-Werte im Bereich [100, 200] generiert.

b) Ein unendlicher Stream, der alle durch 3 teilbaren positiven Zahlen zurückgibt.

c) Ein unendlicher Stream, der abwechselnd den String ∗ (Stern), + (Plus) und − (Minus) generiert.

d) Ein unendlicher Stream, der einen zufälligen Namen aus einer Liste von sechs Namen zurückgibt.

Freiwillige Aufgabe 5: Streams verwenden

Diese Aufgabe ist freiwillig. Sie stammt aus der Klausur des Wintersemesters 2024/25. Sie wurde mit 15% der Klausurpunkte bewertet und ist auf 18 Minuten Bearbeitungszeit angelegt.

Betrachten Sie die beigefügte Klasse **Exam**, die eine Prüfung mit Namen und Notenliste repräsentiert. Die Notenliste besteht aus Objekten des Typs **ExamGrade** (siehe Anhang). Implementieren Sie die folgenden Hilfsmethoden:

a) Die Methode **getTotalStudents** erhält eine **Liste** von Prüfungen und gibt die Gesamtzahl aller Prüfungsnoten zurück.

b) Die Methode **getNumberOfPassedGrades** erhält ein **Array** von Prüfungsnoten. Die Methode gibt zurück, wie viele Studierende bestanden haben (d.h. 4,0 oder besser).

c) Die Methode **containsGradeBetterThan** erhält eine Prüfung **exam** und eine Note **grade**. Sie gibt zurück, ob es in der gegebenen Prüfung eine Note gibt, die besser ist als **grade**.

d) Die Methode **getAllPassedStudentNumbers** erhält eine Liste von Prüfungen. Sie gibt die Matrikelnummern aller Studierenden zurück, die bestanden haben. Eine Matrikelnummer soll nicht doppelt zurückgegeben werden. Tipp: **flatMap** verwenden.