Check und Prepare

Vorbereitende Übungen

TECHNISCHE UNIVERSITÄT DARMSTADT

Autoren: Nhan Huynh und Darya Nikitina
Fachbereich: Informatik
Übungsblatt: 10
Version: 20. November 2021
Semesterübergreifend

V1 Theoriefragen

★★★

Welche der folgenden Aussagen ist wahr?

(A) Streams können aus Listen und Arrays erzeugt werden.

(B) Man kann nur mit Iteratoren über die einzelnen Elemente in einem Stream gehen.

(C) Ein Objekt der Klasse `Path` verwaltet den Pfadnamen einer Datei oder eines Verzeichnisses oder eines anderen Objektes, das im jeweiligen Dateisystem über einen Pfadnamen ansprechbar ist.

(D) Streams können in sich Daten speichern.

(E) Byteweiser Zugriff ist nur dann sinnvoll, wenn eine Datei nur Text und keine Bilder, Audio- oder Videodateien enthält, da bei einem Text die Bytes leichter gelesen werden können.

(F) `Runnable` ist ein Functional Interface. Die funktionale Methode heißt `run`, hat keine Parameter und ist `void`.

(G) Einzelne Threads können nicht terminiert werden, sie enden alle mit dem Ende des Gesamtprogramms.

(H) Eine Parallelisierung mit Threads verkürzt immer die Laufzeit eines Programms.

(I) Wann immer auf einen Button geklickt wird, wird Methode `actionPerformed` jedes bei diesem Button registrierten Listeners aufgerufen.

(J) Für jede Art von Listener gibt es eine eigene Registrierungsmethode.

(K) Die Klasse `Canvas` bietet eine unbegrenzte Zeichenfläche in einem Fenster.

Übungsblatt 10: Vorbereitende Übungen von Nhan Huynh und Darya Nikitina

V2 Vereinfachung mittels Lambda-Ausdrücken

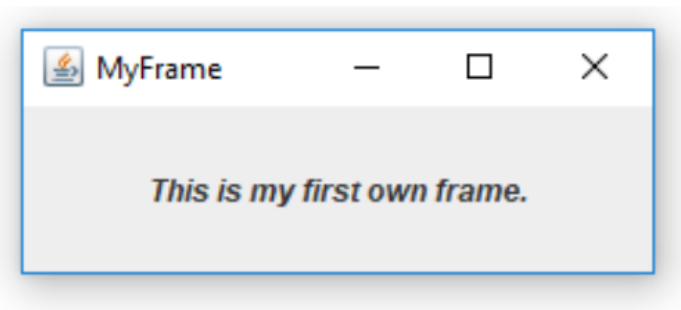
Gegeben sei nachstehender Java-Code. In diesem wird aus einer Liste von Zahlen der Durchschnittswert aller *positiven geraden Zahlen* berechnet, in einer Variablen gespeichert und zurückgegeben.

```java
/**
 * Computes the average of even numbers.
 *
 * @param numbers the array to compute the avearage of even numbers
 * @return the average of even numbers
 */
double averageOfEvenNumbers(int[] numbers) {
  int sum = 0, count = 0;
  for (int i = 0; i < numbers.length; i++) {
    // Only count even numbers and positive
    if (numbers[i] > 0 && numbers[i] % 2 == 0) {
      sum += numbers[i];
      count++;
    }
  }
  // int/int = int, one of the operand must be double to get a double as result
  return ((double) sum) / count;
}
```

1. Nutzen Sie Streams und Lambda-Ausdrücke um den Code zu verkürzen.
2. Beide Implementierungen – die „normale“ und die `Stream`-Variante – weisen eine konzeptionelle Unsauberkeit auf. Bei beiden kann es zum Auftreten einer `Exception` kommen.
   (a) An welcher Stelle kann es bei der „normale“ Implementierung zu einer `Exception` kommen? Um welche handelt es sich?
   (b) An welcher Stelle kann es bei der Implementierung mittels Streams zu einer `Exception` kommen? Um welche handelt es sich?
3. Modifizieren Sie den Code entsprechend, dass die Exceptions aus Teilaufgabe 2 gar nicht mehr auftreten können, sondern die Methode für alle übergebenen Arrays fehlerfrei durchläuft.

V3 Mein eigenes kleines Fenster

Implementieren Sie ein kleines Programm mit einer GUI, welche genauso aussieht wie in der nachfolgenden Abbildung:



Übungsblatt 10: Vorbereitende Übungen von Nhan Huynh und Darya Nikitina

V4 Innere Klassen und Scope

Betrachten Sie den untenstehenden Java-Code, welcher eine innere Klasse `InnerClass` enthält. In dieser wiederum ist eine Methode definiert, die einen Lambda-Ausdruck enthält, dessen Operation mit der Methode `accept` des Interface `Consumer` aufgerufen wird.

```
1  public class LambdaScope {
2
3    public int x = 0;
4
5    class InnerClass {
6
7      public int x = 11;
8
9      void methodInInnerClass(int x) {
10       int z = 55;
11
12       Consumer<Integer> myConsumer = (y) -> {
13         System.out.println("x = " + x);
14         System.out.println("y = " + y);
15         System.out.println("this.x = " + this.x);
16         System.out.println("LambdaScope.this.x = " + LambdaScope.this.x);
17         System.out.println("z = " + z);
18       };
19       myConsumer.accept(x);
20     }
21   }
22
23   public static void main(String[] args) {
24     LambdaScope scope = new LambdaScope();
25     LambdaScope.InnerClass fl = scope.new InnerClass();
26     fl.methodInInnerClass(123);
27   }
28 }
```

Welche Ausgabe wird bei der Ausführung des Codes auf der Konsole ausgegeben? Überlegen Sie sich die Antworten ohne die Nutzung eines Compilers!

V5 Zeilen nummerieren

Implementieren Sie die Methode `void insertRowNumbers(String path)`, welche den Pfad einer Text-Datei als `String` übergeben bekommt. Die Methode soll den Text der Datei Zeile für Zeile einlesen. Beginnend ab der ersten Zeile soll dann in jeder zweiten Zeile nun die Zeilennummer eingefügt werden. War die alte Text-Datei folgendermaßen aufgebaut: `“Row1 \n Row2“` so soll die neue Text-Datei so aussehen: `“1 Row1 \n 2 Row2“`. Dabei steht `“\n“` für einen Zeilenumbruch.

V6 Bäckerei gesucht ★★★

Sie planen eine Party und benötigen eine Menge Brötchen dafür. Sie suchen nun den Bäcker in Ihrer Umgebung, der Ihnen die günstigsten Brötchen verkaufen kann. Dafür gibt es Objekte des Typs `Bakery`, welche zum einen zwei `public double`-Attribute `distance` (gibt die Distanz zu Ihnen in km an) und `price` (gibt den Preis pro Brötchen an) besitzt. Außerdem gibt es Objekte des Typs `BakeryOffer`, welche ebenfalls zwei `public`-Attribute `bakery` vom Typ `Bakery` und `totalPrice` vom Typ `double` besitzt. Der Konstruktor der Klasse sieht folgendermaßen aus:

```
1 public BakeryOffer(Bakery b, double tp) {
2   this.bakery = b;
3   this.totalPrice = tp;
4 }
```

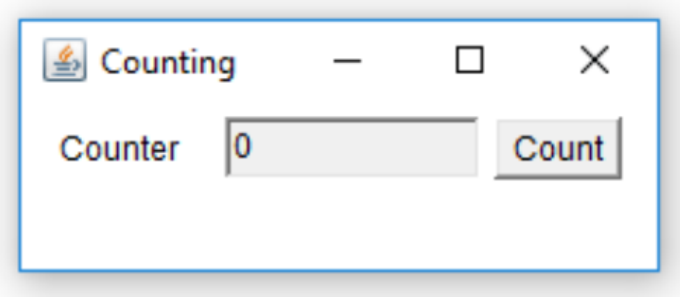
Implementieren Sie nun eine `public`-Methode vom Rückgabetyp `List<BakeryOffer>` mit dem Methodenkopf:

```
sortedBakeryOffers(Collection<Bakery> bakeries, double maxDistance)
```

Die Methode erhält eine Sammlung von Bäckereien und die maximale Distanz zu einer solchen. Zurückgeliefert werden soll eine nach Gesamtpreis aufsteigend sortierte Liste mit Angeboten des Typs `BakeryOffer`. Bäckereien, die weiter als die übergebene maximale Distanz entfernt sind, sollen von der Betrachtung ausgeschlossen werden.

V7 Zähler ★★★

Implementieren Sie ein kleines Programm mit einer GUI, welche genauso aussieht wie in der nachfolgenden Abbildung:



Jedes Mal wenn der *Count*-Button gedrückt wird, soll sich der Wert im Feld um 1 erhöhen.

Hinweis: Ihr Programm besteht aus drei Komponenten:

1. `java.awt.Label` `"Counter"`
2. [a]non-editable `java.awt.TextField`
3. `java.awt.Button` `"Count"`

[a] http://programmedlessons.org/java5/Notes/chap60/ch60_13.html