Programmierung A/B (in Java) – Übung 12

Thema: *Kreuz und quer durch den Vorlesungsinhalt*

Lernziele: *In dieser Übung sollen Sie Ihr über das Semester erworbene Wissen anwenden.*

Hinweis: *Die mit * markierten Aufgaben sind nur für Studierende relevant, welche an Teil B der Veranstaltung teilnehmen. Das heißt natürlich nicht, dass man sie nicht trotzdem zur Übung bearbeiten kann ;-)*

Aufgabe 1

Gegeben sei folgende main-Methode:

```
public static void main(String[] args) {
  int x = Integer.parseInt(args[0]);

  boolean check = false;
  for(int i=1, j=0; j<x; System.out.println(i++ + ": " + check + " " +
                                          (check ? "(" + ++j + ")" : ""))) {
    check = true;
    for(int k=i-1; k>1; k--) {
      if(i%k == 0) {
        check = false;
        break;
      }
    }
  }
}
```

Was genau macht dieser Code?
Schreiben Sie die beiden verschachtelten for-Schleifen in äquivalente verschachtelte while-Schleifen um.

Aufgabe 2

Schreiben Sie eine Klassenmethode ersetzen, welche als Parameter sowohl ein zweidimensionales Array vom Typ int, als auch eine weitere Zahl z vom Typ int annimmt.
Das Array soll so modifiziert werden, dass alle Zahlen des Arrays, von denen z ein Teiler ist, durch den Wert der Division ersetzt werden. Am Ende soll die Methode die Anzahl der Werte zurückgeben, die ersetzt wurden.

Aufgabe 3

Schreiben Sie eine Klassenmethode *berechneReihe*, welche den Wert der Reihe

$$\sum_{n=0}^{k} \frac{1}{2^n} = \frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} + \ \dots \ + \frac{1}{2^k}$$

rekursiv bis zum k-ten Schritt (→ Parameter) berechnet und als double zurückgibt.
Modifizieren Sie Ihre Methode anschließend so, dass sie Endrekursiv wird, falls sie diese Eigenschaft nicht bereits besitzt.

Tipp: Sie können optional eine zusätzliche Methode verwenden, um diese Eigenschaft zu erfüllen.

Aufgabe 4

Gegeben sei folgende Klasse zur Implementierung eines Stacks für Integer, die mit Hilfe einer ArrayList realisiert werden soll:

```
public class SimpleStack {
  private ArrayList<Integer> stackList = new ArrayList<>();

  public void push(Integer value) {
    //Legt den Parameter value auf die Spitze des Stacks
  }

  public Integer pop() {
    //Entfernt das oberste Element des Stacks und gibt es zurück
  }

  public Integer addFirstTwo() {
    //Entfernt und addiert die oberen beiden Elemente, legt die Summe auf den
    //Stack und gibt sie zudem zurück. Bei weniger Elementen soll nur deren Wert
    //zurückgegeben werden
  }
}
```

Implementieren Sie die drei fehlenden Methoden, wobei die Spitze des Stacks immer das erste Element der Liste und null nicht als Element erlaubt sein soll.

***** Erweitern/Verändern Sie diese Klasse so, dass sie beliebige Zahlentypen speichern und verwalten kann. Nehmen Sie dazu zudem die Existenz einer Hilfsmethode add(x,y) an, welche zwei Zahlen gleichen Typs addiert und das Resultat zurückgibt.

Aufgabe 5*

Schreiben Sie ein funktionales Interface *IOperation* mit einer Methode, welche drei Parameter (x,y,z) vom Typ double akzeptiert und selbigen Datentyp zurückgibt. Nun implementieren Sie mit Hilfe von Lambda-Ausdrücken anonyme Klassen vom Typ *IOperation* mit folgenden Funktionen:

- Der Mittelwert der drei Zahlen soll berechnet werden
- Angenommen (x,y,z) bilden einen Vektor – die Länge dieses Vektors soll berechnet werden (*Tipp:* die Methode *Math.sqrt(double)* berechnet die Wurzel der übergebenen Zahl)
- Alle Zahlen zwischen x und y mit Schrittweite z sollen beginnend mit x in die Konsole ausgegeben werden. Die übergebenen Parameter sollen zudem auf Korrektheit überprüft werden und 0 (alles okay) oder 1 (ungültige Parameter) zurückgegeben werden.

Testen Sie Ihre Implementierungen durch den Aufruf mit Testparametern.