Präsenzaufgaben 11 13./14.12.2023

Die Lösung der Aufgaben wird am Ende der Übung von Ihnen vorgestellt.

Aufgabe 1: Schreiben Sie eine Funktion, die einen `IntStream` mit den Potenzen von 2 zurückgibt, solange sie in eine int-Variable passen ($2^0 \ldots 2^{30}$).

```
public static IntStream powersOfTwo()
```

Folgende `Stream`-Methoden könnten nützlich sein: *iterate, limit, range, map.*

Aufgabe 2: Schreiben Sie eine Funktion

```
public static void testStream(IntStream stream, int num)
```

die die ersten `num` Werte des übergebenen Streams auf dem Bildschirm ausgibt.

Folgende `Stream`-Methoden könnten nützlich sein: *forEach.*

Aufgabe 3: Schreiben Sie eine Funktion

```
public static int[] toArray(ArrayList<Integer> list)
```

die einen Stream verwendet, um eine `ArrayList<Integer>` in ein `int`-Array umzuwandeln.

Folgende Stream-Methoden könnten nützlich sein: *list.stream, toArray, mapToInt.*

Aufgabe 4: Schreiben Sie eine Funktion

```
public static double[] getRandNumbers(int cnt)
```

die einen Stream verwendet, um ein Feld mit *cnt* Zufallszahlen zwischen 0 (inkl.) and 1 (exkl.) zu erzeugen und zurückzugeben.

Folgende Stream-Methoden könnten nützlich sein: *generate, limit.*

Aufgabe 5: Schreiben Sie eine Funktion

```
public static Stream<Character> getCharStream(String s)
```

die einen Stream mit den Zeichen von `s` zurückgibt.
Folgende Stream-Methoden könnten nützlich sein: *range, mapToObj.*

Aufgabe 6: Schreiben Sie eine Funktion

```
public static int getMax(int[] arr)
```

die den größten Wert des Felds *arr* zurückgibt. Benutzen Sie dazu Java-Streams. Parallelisieren Sie Ihren Algorithmus.
Folgende Stream-Methoden könnten nützlich sein: *Arrays.stream, getAsInt, max, parallel*.

Aufgabe 7: Schreiben Sie eine Funktion

```
public static int[] getPos(String x, char c)
```

die ein Feld mit allen Positionen zurückgibt, an denen das Zeichen `c` im String `x` vorhanden ist. Parallelisieren Sie die Funktion und nutzen Sie dazu Java-Streams.

Beispiel: Ein Funktionsaufruf mit `x="banana"` und `c='a'` als Parametern sollte das Feld `[1,3,5]` zurückgeben.
Folgende Stream-Methoden könnten nützlich sein: *range, filter.*

Aufgabe 8: Schreiben Sie eine Funktion

```
public static boolean isPrime(long z)
```

die Streams benutzt, um zu ermitteln, ob `z` eine Primzahl ist. Parallelisieren Sie Ihren Algorithmus. Überprüfen Sie Ihr Programm mit der Zahl 10000000000000181 (eine Primzahl). Vergleichen Sie die Laufzeit der seriellen und der parallelen Version. Sie können *System.nanoTime()* für Ihre Zeitmessungen benutzen.

Folgende Stream-Methoden könnten nützlich sein: *noneMatch, range.*

Aufgabe 9: Schreiben Sie eine Funktion

```
public static IntStream getFactors(int n)
```

die einen `IntStream` mit den Teilern von `n` zurückgibt. Die Teiler sollen in aufsteigender Reihenfolge sortiert sein. Beispiel: `getFactors(12)` gibt als Ergebnis den Stream `(1,2,3,4,6,12)` zurück.

Folgende Stream-Methoden könnten nützlich sein: *range, filter.*

Aufgabe 10: Schreiben Sie eine Funktion

```
public static int[] uniq(int[] x)
```

die alle Elemente aus dem Feld `x` löscht, die denselben Wert haben wie das vorhergehende Element. Benutzen Sie Java-Streams und parallelisieren Sie die Funktion.

Beispiel: Die Funktion sollte das Feld `[5,2,7,7,2,9,3,3,3,2]` in das Feld `[5,2,7,2,9,3,2]` umwandeln.

Folgende Stream-Methoden könnten nützlich sein: *range, filter, map.*

Aufgabe 11: Schreiben Sie eine Funktion

```
public static Stream<String> binaryNumbers()
```

zur Rückgabe eines unendlichen `Stream<String>` mit den Binärzahlen (0, 1, 10, 11, 100, 101, 110, …).
Hinweis: Die Funktion `Integer.toBinaryString(int i)` wandelt *i* in die String-Repräsentation von *i* im Binärsystem um.

Folgende Stream-Methoden könnten nützlich sein: *iterate, mapToObj.*