Aufgabenblatt 4

Kompetenzstufe 1 & Kompetenzstufe 2

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihr IntelliJ-Projekt bis spätestens **Mittwoch, 26.11.2025 23:55 Uhr** in TUWEL hoch.
- Zusätzlich müssen Sie in TUWEL ankreuzen, welche Aufgaben Sie gelöst haben.
- Ihre Programme müssen kompilierbar und korrekt ausführbar sein.
- Ändern Sie bitte **nicht** die **Dateinamen** und die **vorhandene Ordnerstruktur**.
- Bei manchen Aufgaben finden Sie Zusatzfragen. Diese Zusatzfragen beziehen sich thematisch auf das erstellte Programm. Sie müssen diese Zusatzfragen für gekreuzte Aufgaben in der Übung beantworten können. Sie können die Antworten dazu als Java-Kommentare in die Dateien schreiben.
- Verwenden Sie, falls nicht anders angegeben, für alle Ausgaben `System.out.println()` bzw. `System.out.print()`.
- Verwenden Sie für die Lösung der Aufgaben keine Aufrufe (Klassen) aus der Java-API, außer diese sind ausdrücklich erlaubt.
- Erlaubt sind die Klassen `String`, `Math`, und `Integer`, es sei denn in den Hinweisen zu den einzelnen Aufgaben ist etwas anderes angegeben.
- Bitte beachten Sie die Vorbedingungen! Sie dürfen sich darauf verlassen, dass alle Aufrufe die genannten Vorbedingungen erfüllen. Sie müssen diese nicht in den Methoden überprüfen.

In diesem Aufgabenblatt werden folgende Themen behandelt:

- Code-Verstehen mit Arrays
- Verwendung von ein- und zweidimensionalen Arrays

Aufgabe 1 (1 Punkt)

Beantworten Sie die folgenden Fragen bitte in dem dafür vorgesehenen Bereich (ganz unten) im Code von *Aufgabe1.java*. Sie können die Antworten zu den Zusatzfragen direkt darunter angeben. Änderungen im Code sind nicht durchzuführen, außer für die erste Frage, wo es darum geht, die Exception zu vermeiden:

a) Warum kommt es in der Methode `printArray` (Zeile 6) zu einem Fehler (Exception)? Korrigieren Sie den Fehler, sodass die Methode keine Exception wirft und die Ausgabe des Arrays durchführt. In welcher Reihenfolge die Elemente ausgegeben werden, ist an dieser Stelle nicht relevant und muss daher auch nicht geändert werden.

b) Wieso hat die Methode `fillArray` keinen Rückgabewert, obwohl ein Array befüllt werden soll?

c) Der Aufruf `printContentFilteredArray(filledArray)` in Zeile 47 soll im Array `filledArray` alle durch 14 teilbaren Zahlen auf -1 setzen und das Array ausgeben. Warum aber ergibt der Aufruf `printArray(filledArray)` in Zeile 48 dann ebenfalls dieses gefilterte Array, obwohl innerhalb der Methode `printContentFilteredArray` anscheinend auf einer Kopie gearbeitet wurde?

d) In Zeile 50 wird in `filledArray` an der Stelle 0 der Wert 2412 gespeichert. Danach wird in Zeile 53 die Methode `fillArrayWithNewContent` aufgerufen, welche ein neues Array mit neuem Inhalt erzeugen soll. Wie in Zeile 39 gezeigt, befindet sich ein neuer Arrayinhalt in `workArray`, aber wieso ergibt der Aufruf in Zeile 54 wiederum den alten Arrayinhalt?

Zusatzfrage(n): Gehen Sie hier von eindimensionalen Arrays aus!

1. Welchen Datentyp muss der Indexausdruck haben, mit dem die Position in einem Array bestimmt wird?
2. Wie kann die Länge eines Arrays verändert werden?
3. Wie gehen Sie vor, wenn Sie ein `int`-Array kopieren müssen?
4. Ist es sinnvoll, zwei Arrays mit `"=="` zu vergleichen? Was passiert im Detail bei einem Vergleich mit `"=="`?

Aufgabe 2 (1 Punkt)

Bearbeiten Sie folgende Aufgabenstellung:

- Es ist eine Methode `replaceValues` gegeben:

  ```
  void replaceValues(int[] workArray, int threshold)
  ```

  Diese Methode ermittelt den Durchschnittswert aller Einträge des Arrays `workArray`. Danach werden alle Werte, die kleiner gleich dem `int`-Parameterwert `threshold` sind, auf 0 gesetzt und alle anderen Werte werden durch den Durchschnittswert ersetzt.

  Vorbedingungen: `workArray != null` und `workArray.length > 0`.

  Beispiele:
  `int[] array1 = new int[]{0, 9, 9, 9, 9};`
  `replaceValues(array1, 10)` führt zu `[0, 0, 0, 0, 0]`

  `int[] array2 = new int[]{12, 12, 12};`
  `replaceValues(array2, 10)` führt zu `[12, 12, 12]`

  `int[] array3 = new int[]{0, 20, 100, 100};`
  `replaceValues(array3, 50)` führt zu `[0, 0, 55, 55]`

- Diese Implementierung wurde mit Hilfe eines *Large Language Models* (LLM) erstellt. Laut den Testfällen oben arbeitet die Methode korrekt. Es ist jetzt Ihre Aufgabe zu analysieren, ob die Methode korrekt arbeitet, oder sich doch Fehler in der Implementierung befinden. Zeigen Sie mit konkreten Testfällen, an welchen Stellen die Methode nicht korrekt arbeitet. Geben Sie als Kommentar im Code an, bei welchen Stellen es zu Problemen kommt und wie man diese beseitigen kann.

- Schreiben Sie danach zusätzlich eine korrekte Version der Methode, die alle Fehler ausgebessert hat. Rufen Sie diese Methode mit den Testfällen auf und überprüfen Sie, ob die Methode korrekt arbeitet.

Aufgabe 3 (1 Punkt)

Implementieren Sie folgende Aufgabenstellung:

- Implementieren Sie eine Methode `generateFilledArray`:

  ```
  int[][] generateFilledArray(int n)
  ```

  Die Methode erzeugt ein zweidimensionales Array der Größe `n`×`n` und befüllt dieses mit Zahlen wie in den nachfolgenden Beispielen gezeigt. Das Array wird zeilenweise befüllt. Dabei werden die Zeilen, beginnend mit der ersten Zeile, abwechselnd von links nach rechts und von rechts nach links mit aufeinanderfolgenden Zahlen befüllt. In das erste Feld der ersten Zeile wird eine 1 eingetragen, alle nachfolgenden Zahlen sind jeweils um 1 größer als der zuvor eingetragene Wert.

  Vorbedingung: `n > 0`.

  Beispiele:

  `generateFilledArray(1)` erzeugt →

  ```
  1
  ```

  `generateFilledArray(2)` erzeugt →

  ```
  1 2
  4 3
  ```

  `generateFilledArray(3)` erzeugt →

  ```
  1 2 3
  6 5 4
  7 8 9
  ```

  `generateFilledArray(5)` erzeugt →

  ```
   1  2  3  4  5
  10  9  8  7  6
  11 12 13 14 15
  20 19 18 17 16
  21 22 23 24 25
  ```

`generateFilledArray(7)` erzeugt →

```
 1  2  3  4  5  6  7
14 13 12 11 10  9  8
15 16 17 18 19 20 21
28 27 26 25 24 23 22
29 30 31 32 33 34 35
42 41 40 39 38 37 36
43 44 45 46 47 48 49
```

Aufgabe 4 (1 Punkt)

Implementieren Sie folgende Aufgabenstellung:

- Implementieren Sie eine Methode `generateExtendedArray`:

  `int[][] generateExtendedArray(int[] inputArray)`

  Diese Methode erstellt ein ganzzahliges zweidimensionales Array, bei dem die Größe aus dem Array `inputArray` abgeleitet wird. Das `inputArray` hat genau die Länge drei. Der erste Wert gibt die Anzahl der Zeilen des neuen Arrays an. Der zweite und dritte Wert geben die Längen der Zeilen an. Die erste Zeile des neuen Arrays hat die Länge `inputArray[1]`, die zweite Zeile die Länge `inputArray[2]`, die dritte Zeile wieder `inputArray[1]`, usw., bis alle Zeilen erstellt wurden. Der Inhalt jeder Zeile dieses neuen Arrays wird spaltenweise beginnend von 1 durchnummeriert bzw. befüllt (siehe Beispiele). Anschließend wird das neu erstellte Array zurückgegeben.

  Vorbedingungen: `inputArray != null`, `inputArray.length = 3`, `inputArray[0] > 1`, `inputArray[1] >= 0` und `inputArray[2] >= 0`.

  Beispiele:

  `generateExtendedArray(new int[]{2, 1, 1})` erzeugt →

  ```
  1
  2
  ```

  `generateExtendedArray(new int[]{4, 3, 0})` erzeugt →

  ```
  1 3 5

  2 4 6
  ```

  `generateExtendedArray(new int[]{3, 4, 7})` erzeugt →

  ```
  1 4 7 10
  2 5 8 11 13 14 15
  3 6 9 12
  ```

  `generateExtendedArray(new int[]{7, 4, 2})` erzeugt →

  ```
  1 8 15 19
  2 9
  3 10 16 20
  4 11
  5 12 17 21
  6 13
  7 14 18 22
  ```

Aufgabe 5 (1 Punkt)

Implementieren Sie folgende Aufgabenstellung:

- Implementieren Sie eine Methode `generateReformattedArray`:

  ```
  int[][] generateReformattedArray(int[][] inputArray)
  ```

  Diese Methode erstellt ein ganzzahliges zweidimensionales Array mit so vielen Spalten wie das `inputArray` Zeilen aufweist. Die Werte werden zeilenweise mit den Werten aus dem Array `inputArray` befüllt. Die Anzahl der Zeilen ergibt sich aus der Gesamtanzahl an Einträgen im `inputArray`. Wenn das neue Array nicht vollständig befüllt werden kann (d.h. wenn die Anzahl der Elemente in `inputArray` nicht durch die neue Anzahl an Zeilen teilbar ist), bleiben die übrigen Array-Einträge auf 0.

  Vorbedingungen: `inputArray != null`, `inputArray.length > 0` und es gilt für alle gültigen `i`, dass `inputArray[i] != null` und `inputArray[i].length > 0` ist.

  Beispiele:

  `generateReformattedArray(new int[][]{{1}})` erzeugt →

  ```
  1
  ```

  ```
  generateReformattedArray(new int[][]{
  {1}, {2, 3, 4, 5, 6, 7}, {8, 9}})
  ```
  erzeugt →

  ```
  1 2 3
  4 5 6
  7 8 9
  ```

  ```
  generateReformattedArray(new int[][]{
  {1}, {3}, {8, 5}, {6, 5, 9}, {10, 4, 7, 11}})
  ```
  erzeugt →

  ```
  1 3 8 5 6
  5 9 10 4 7
  11 0 0 0 0
  ```

  ```
  generateReformattedArray(new int[][]{
  {1, 2, 3, 4, 5, 6, 7, 8}, {9, 10, 11}})
  ```
  erzeugt →

  ```
  1 2
  3 4
  5 6
  7 8
  9 10
  11 0
  ```