Aufgabenblatt 4

Kompetenzstufe 1 & Kompetenzstufe 2

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihr IntelliJ-Projekt bis spätestens **Donnerstag, 08.12.2022 20:00 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
- Rekursion mit eindimensionalen Arrays

Aufgabe 1 (1 Punkt)

Die folgenden Fragen beantworten Sie bitte in dem dafür vorgesehenen Bereich (ganz unten) im Code von *Aufgabe1.java*. Die Zusatzfragen können dann anschließend daran beantwortet werden. 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 35) zu einem Fehler (Exception)? Korrigieren Sie den Fehler, sodass die Methode keine Exception wirft und auch die Funktionalität von `printArray` (Ausgabe des gesamten Arrays in einer Zeile auf der Konsole) korrekt implementiert ist.

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

c) Der Aufruf der Methode `printContentFilteredArray(filledArray)` in Zeile 47 soll alle durch 4 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` auf einer Kopie gearbeitet wurde?

d) In Zeile 50 wird in `filledArray` an der Stelle 0 der Wert 777 eingefügt. Danach wird in Zeile 53 die Methode `fillArrayWithNewContent` aufgerufen, welche ein neues Array mit neuem Inhalt erzeugen soll. Wie in Zeile 32 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. Müssen Sie ein Array initialisieren?
3. Wie kann die Länge eines Arrays verändert werden?
4. Wie gehen Sie vor, wenn Sie ein int-Array kopieren müssen?
5. Beginnt die Indexzählung eines Arrays immer bei 0?
6. Ist es sinnvoll, zwei Arrays mit `"=="` zu vergleichen? Was passiert im Detail, bei einem Vergleich mit `"=="`?

Aufgabe 2 (1 Punkt)

Implementieren Sie folgende Aufgabenstellung:

(!) Die folgenden Aufgabenstellungen implementieren Sie direkt in `main`. Sie können sich aber für die Ausgabe der Arrays Hilfsmethoden schreiben.

a) Erstellen Sie ein eindimensionales ganzzahliges Array und initialisieren Sie es mit den Werten `{6, 1, 8, 2, 5, 3, 4, 7, 9, 0}`. Geben Sie anschließend jeden Arrayeintrag getrennt durch Hashzeichen auf der Konsole aus.
Erwartete Ausgabe:
`6#1#8#2#5#3#4#7#9#0`

b) Erstellen Sie ein eindimensionales ganzzahliges Array der Länge 12 und initialisieren Sie es mit Werten der Sechserreihe beginnend bei 6. Setzen Sie alle Elemente des Arrays auf 0, die durch 9 teilbar sind. Geben Sie anschließend alle Elemente des Arrays nebeneinander getrennt durch Leerzeichen auf der Konsole aus.
Erwartete Ausgabe:
`6 12 0 24 30 0 42 48 0 60 66 0`

c) Erstellen Sie ein eindimensionales ganzzahliges Array und initialisieren Sie es mit den Werten `{7, 3, 2, 7, 6, 7, 7, 8, 9, 5}`. Nach der Erstellung und Initialisierung des Arrays soll ein neues Array erstellt werden, das die Inhalte des zuvor erstellten Arrays enthält und zusätzlich noch den Wert -1 nach jedem Vorkommen der Zahl 7 im Array. Geben Sie anschließend alle Elemente des neuen Arrays nebeneinander getrennt durch Leerzeichen auf der Konsole aus.
Erwartete Ausgabe:
`7 -1 3 2 7 -1 6 7 -1 7 -1 8 9 5`

d) Erstellen Sie ein eindimensionales ganzzahliges Array der Länge 11 und initialisieren Sie es mittels Schleife mit den Werten von 1 bis 11. Geben Sie anschließend die Elemente des Arrays in umgekehrter Reihenfolge getrennt durch Beistriche aus. Geben Sie das Array einmal mit Hilfe einer while-Schleife und einmal mit Hilfe einer for-Schleife auf der Konsole aus. Zusätzlich soll zur Unterscheidung der Ausgabe der String `while-Schleife:` bzw. `for-Schleife:` bei der jeweiligen Schleife ausgegeben werden.
Erwartete Ausgabe:
for-Schleife: `11,10,9,8,7,6,5,4,3,2,1`
while-Schleife: `11,10,9,8,7,6,5,4,3,2,1`

Aufgabe 3 (1 Punkt)

Implementieren Sie folgende Aufgabenstellung:

(!) Für die folgenden Methoden dürfen keine globalen Variablen oder zusätzliche eigene Hilfsmethoden verwendet werden. Die vorgegebenen Methodenköpfe dürfen nicht erweitert oder geändert werden.

a) Implementieren Sie eine Methode `genRandomArray`:

```
int[] genRandomArray(int length, int maxNumber)
```

Diese Methode generiert ein eindimensionales ganzzahliges Array der Länge `length`, befüllt dieses mit Zufallszahlen aus dem Intervall von `[0, maxNumber[` und gibt das Array anschließend zurück.
Vorbedingungen: `length` > 0 und `maxNumber` > 0.

b) Implementieren Sie eine Methode `replaceValues`:

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

Diese Methode berechnet den Mittelwert aller Arrayeinträge als ganzzahligen (Nachkommaanteil abschneiden) Wert. Danach werden alle Werte innerhalb des Arrays vom Index `idx` beginnend bis zum Ende des Arrays, die größer als der Mittelwert sind, durch den Mittelwert ersetzt.
Vorbedingungen: `workArray != null`, `workArray.length > 0` und `idx` ≥ 0.

Beispiele:
`int[] array2 = new int[]{12, 3, 15, 18, 22, 9, 5, 8, 16, 21};`
`replaceValues(array2, 4)` führt zu
`[12, 3, 15, 18, 12, 9, 5, 8, 12, 12]`
`int[] array3 = new int[]{21, 14, 17, 12};`
`replaceValues(array3, 0)` führt zu `[16, 14, 16, 12]`
`int[] array4 = new int[]{3, 4, 6, 7};`
`replaceValues(array4, 2)` führt zu `[3, 4, 5, 5]`
`int[] array5 = new int[]{7, 7};`
`replaceValues(array5, 1)` führt zu `[7, 7]`
`int[] array6 = new int[]{3, 2, 1};`
`replaceValues(array6, 2)` führt zu `[3, 2, 1]`

Aufgabe 4 (1 Punkt)

Implementieren Sie folgende Aufgabenstellung:

(!) Folgende Methoden dürfen keine globalen Variablen oder zusätzliche eigene Hilfsmethoden verwenden. Die vorgegebenen Methodenköpfe dürfen nicht erweitert oder geändert werden. Für die Implementierung der Aufgabenstellung dürfen keine Schleifen verwendet werden.

a) Implementieren Sie eine **rekursive** Methode `shiftHighestValue`:

```
void shiftHighestValue(int[] workArray, int index)
```

Diese Methode soll den größten Wert innerhalb des Array beginnend vom Index `index` bis zum Ende des Arrays ermitteln und diesen an die letzte Stelle im Array schreiben. Immer wenn ein größerer Wert gefunden wird, dann tauscht dieser den Platz mit dem letzten Arrayeintrag. Die ursprüngliche Reihenfolge der Arrayeinträge darf sich verändern.
Vorbedingungen: `workArray != null`, `workArray.length > 1`, `index` ≥ 0 und `index` $<$ `workArray.length`.

Beispiele:
`int[] array1 = {32, 46, 22, 38, 41, 24, 33, 28, 12};`
`shiftHighestValue(array1, 0)` führt zu `[32, 41, 22, 38, 33, 24, 28, 12, 46]`
`int[] array2 = {5, 4, 3, 2, 1};`
`shiftHighestValue(array2, 0)` führt zu `[4, 3, 2, 1, 5]`
`int[] array3 = {10, 2, 3, 5, 4};`
`shiftHighestValue(array3, 1)` führt zu `[10, 2, 3, 4, 5]`
`int[] array4 = {1, 5, 3, 6, 9};`
`shiftHighestValue(array4, 4)` führt zu `[1, 5, 3, 6, 9]`

b) Implementieren Sie eine **rekursive** Methode `containsValue`:

```
boolean containsValue(int[] workArray, int value)
```

Diese Methode prüft, ob eines der Elemente innerhalb des Arrays `workArray` dem Wert von `value` entspricht. Die Methode soll in ihrem Methodenrumpf zwei rekursive Aufrufe enthalten, wobei ein rekursiver Aufruf die erste Hälfte des Arrays durchsucht und der zweite Aufruf die zweite Hälfte. Sie können die Methode `Arrays.copyOfRange(...)` für die Implementierung der Methode benutzen.
Vorbedingungen: `workArray != null` und `workArray.length > 0`.

Beispiele:
`int[] array2 = {3, 9, 17, 11, -7, 8, 0, 9, 24, -3, 17, 4};`
`containsValue(array2, 11)` liefert `true` zurück
`containsValue(array2, 2)` liefert `false` zurück
`containsValue(array2, -7)` liefert `true` zurück
`containsValue(array2, 0)` liefert `true` zurück
`containsValue(array2, 9)` liefert `true` zurück
`containsValue(array2, 16)` liefert `false` zurück

Aufgabe 5 (2 Punkte)

Implementieren Sie folgende Aufgabenstellung:

a) Implementieren Sie eine Methode `genFilledArray`:

```
int[][] genFilledArray(int n)
```

Die Methode erzeugt ein zweidimensionales Array der Größe n×n und befüllt dieses mit Zahlen, wie in den nachfolgenden Beispielen gezeigt. Es wird links oben (0,0) mit 0 begonnen und in jeder weiteren Gegendiagonalen die Zahl um 1 vergrößert. Anschließend wird das neu erzeugte Array zurückgegeben.

Vorbedingung: `n > 0`.

Beispiele:

`genFilledArray(2)` erzeugt →

```
0 1
1 2
```

`genFilledArray(4)` erzeugt →

```
0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6
```

`genFilledArray(7)` erzeugt →

```
0 1 2 3 4 5 6
1 2 3 4 5 6 7
2 3 4 5 6 7 8
3 4 5 6 7 8 9
4 5 6 7 8 9 10
5 6 7 8 9 10 11
6 7 8 9 10 11 12
```

b) Implementieren Sie eine Methode `extendArray`:

```
int[][] extendArray(int[] inputArray)
```

Diese Methode erstellt ein ganzzahliges zweidimensionales Array, bei dem jede Zeile eine andere Länge aufweisen kann. Die Länge der Zeilen ergibt sich durch die Inhalte von `inputArray`.

Der Arrayinhalt an der Indexstelle 0 wird laut der Formel `inputArray[0]` $\cdot\, 2 + 1$ für die Berechnung der Länge der ersten Zeile verwendet, Index 1 für die Berechnung der zweiten Zeilenlänge, usw. Die Zeilen des neuen Arrays werden nun so befüllt, sodass für Index 0 im ursprünglichen Array `inputArray[0]`-mal -1 eingefügt wird, dann die Zahl selbst und danach nochmal `inputArray[0]`-mal die -1 (siehe Beispiele). Ist ein Wert in `inputArray` gleich 0, dann wird im Ergebnisarray für diese Zeile ein Array der Länge 1 mit dem Wert 0 eingefügt.

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

Beispiele:

`extendArray(new int[]{4, 0, 5, 2})` erzeugt →

```
-1 -1 -1 -1 4 -1 -1 -1 -1
0
-1 -1 -1 -1 -1 5 -1 -1 -1 -1 -1
-1 -1 2 -1 -1
```

`extendArray(new int[]{0})` erzeugt →

```
0
```

`extendArray(new int[]0, 1, 2, 1, 0})` erzeugt →

```
0
-1 1 -1
-1 -1 2 -1 -1
-1 1 -1
0
```

c) Implementieren Sie eine Methode `reformatArray`:

```
int[] reformatArray(int[][] inputArray)
```

Diese Methode zählt in jeder Zeile von `inputArray` die Anzahl der Nullen und erstellt ein neues eindimensionales Array, das diese Anzahl von Nullen als eigenen Eintrag abspeichert. Die Anzahl aller Nullen der ersten Zeile wird im Rückgabearray an der Position mit Index 0 abgespeichert, die Anzahl der zweiten Zeile an der Position mit Index 1, usw.

Vorbedingungen: `inputArray != null`, `inputArray.length > 0`, dann gilt für alle gültigen `i`, dass `inputArray[i].length > 0` ist. Alle Zahlen in `inputArray` sind Nullen oder Einsen.

Beispiele:

```
reformatArray(new int[][]{

{1, 0, 1, 1},
{0, 1, 1},
{0, 1, 0, 1, 1},
{0, 0, 0, 1, 0},
{1, 0},
{1, 1, 1, 1, 1}}) erzeugt →

1 1 2 4 1 0

reformatArray(new int[][]{

{1, 0, 1, 1, 0, 0, 0, 0},
{0, 1, 1, 1, 1, 1, 0, 0},
{0, 0, 0, 0, 0, 0, 1, 1},
{1, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 1, 1, 0, 1},
{0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 1}}) erzeugt →

5 3 6 7 5 8 7
```