Aufgabenblatt 4

Kompetenzstufe 1 & Kompetenzstufe 2

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihr IntelliJ-Projekt bis spätestens **Freitag, 14.12.2018 13:00 Uhr** in TUWEL hoch.
- Zusätzlich müssen Sie in TUWEL ankreuzen, welche Aufgaben Sie gelöst haben und während der Übung präsentieren können.
- Ihre Programme müssen kompilierbar und 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, wie z.B. die Klassen `StdDraw`, `Scanner` und `Math` oder Klassen, die in den Hinweisen zu den einzelnen Aufgaben aufscheinen.

In diesem Aufgabenblatt werden folgende Themen behandelt:

- Code-Verstehen mit Arrays
- Verwendung von eindimensionalen Arrays
- Verwendung von zweidimensionalen Arrays

Aufgabe 1

Die folgenden Fragen beantworten Sie bitte in dem dafür vorgesehenen Bereich (ganz unten) im Code. 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:

1. Warum kommt es in der Methode `printArray` zu einem Fehler (Exception) und wie kann diese Exception vermieden werden?

2. Wieso hat die Methode `genArray` keinen Rückgabewert, obwohl ein Array befüllt wird?

3. Der Aufruf der Methode `printFilteredArrayContent(filledArray)` in Zeile 47 soll alle durch 4 teilbaren Zahlen auf 0 setzen und das Array ausgeben. Warum aber ergibt der Aufruf `printArray(filledArray)` in Zeile 48 dann ebenfalls dieses gefilterte Array, obwohl innerhalb der Methode `printFilteredArrayContent` anscheinend auf einer Kopie gearbeitet wurde?

4. In Zeile 50 wird in `filledArray` an der letzten Position der Wert 333 eingefügt. Danach wird in Zeile 53 die Methode `genNewArrayContent` 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 berechnet 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?

Aufgabe 2

Implementieren Sie folgende Aufgabenstellung:

- Implementieren Sie eine Methode `genRandomArray`:

  ```
  int[] genRandomArray(int numValues, int maxValue)
  ```

  Die Methode erzeugt ein Array mit ganzzahligen Zufallswerten zwischen 0 (inklusive) und `maxValue` (exklusive). Die Länge des Arrays wird durch den Parameter `numValues` angegeben.

- Implementieren Sie eine Methode `getMaxValueFromArray`:

  ```
  int getMaxValueFromArray(int[] array)
  ```

  Die Methode nimmt ein ganzzahliges Array als Parameter entgegen und gibt den größten Wert innerhalb des Arrays zurück.

- Implementieren Sie eine Methode `calcStatistics`:

  ```
  int[] calcStatistics(int[] array)
  ```

  Die Methode nimmt ein ganzzahliges Array als Parameter entgegen und zählt, wie oft jeder Wert im Array vorkommt. Es wird dazu ein neues Array angelegt. Die Länge des neuen Arrays ergibt sich aus dem größten vorhandenen Wert des zu analysierenden Arrays `array`. Jede Stelle des neuen Arrays fungiert als Zähler für die vorkommenden Zahlen im zu analysierenden Array `array`. Ein Beispiel: Hat das übergebene Array `array` die Länge 10 und ist mit den folgenden Werten befüllt:

  $$[2|5|0|4|8|5|3|7|2|5],$$

  dann hat das Ergebnis-Array die Länge 9, da der höchste Wert im übergebenen Array 8 ist (Werte im Intervall $[0, 8] \rightarrow 9$ verschiedene Einträge möglich) . Wird z.B. die Zahl 5 im übergebenen Array gefunden, dann wird der Wert beim Index 5 im Ergebnis-Array um 1 erhöht. Dies wird für jede Zahl gemacht, bis am Ende für das obige Beispiel-Array folgendes Ergebnis-Array entsteht:

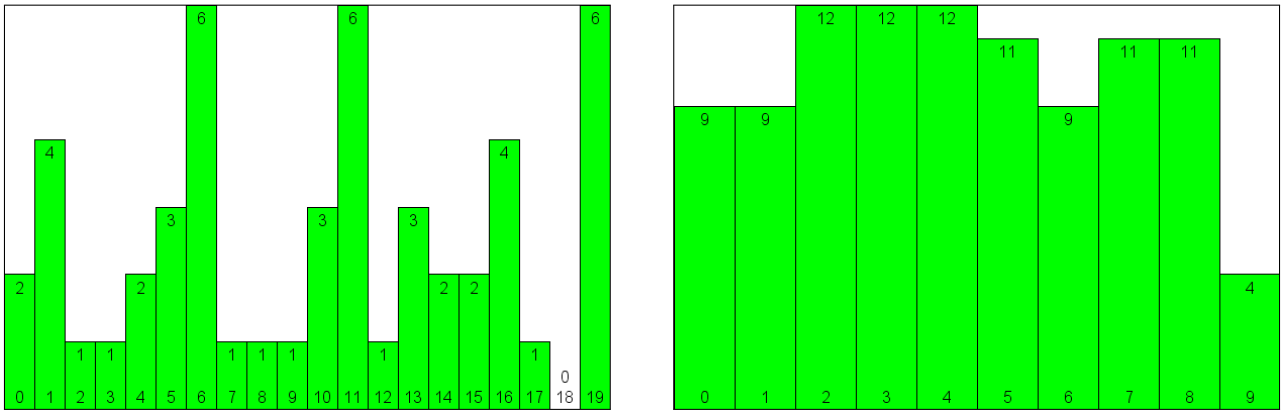
  $$[1|0|2|1|1|3|0|1|1].$$

- Implementieren Sie eine Methode `drawBarChart`:

  ```
  void drawBarChart(int[] array)
  ```

  Die Methode nimmt ein ganzzahliges Array als Parameter entgegen und gibt dieses in Balkenform aus (Abbildung 1). In Abbildung 1a und Abbildung 1b steht jeder Balken für einen Eintrag im Array. Die größte vorkommende Zahl innerhalb des Arrays erzeugt den höchsten Balken, der die ganze Bildhöhe (Fenstergröße ist 600×400 Pixel) ausfüllt. Alle anderen Werte werden entsprechend skaliert. Am unteren Ende des Balkens werden die Indexpositionen angegeben und am oberen Ende der Inhalt des Arrays an dieser Position. Steht eine 0 an einer Indexposition im Array, dann muss das beim Zeichnen berücksichtigt werden, da in diesem Fall kein Balken vorhanden ist (für ein Beispiel siehe 1a). Verwenden Sie diese Methode, um Ihre zuvor mit `calcStatistics` erstellte Statistik zu visualisieren.

! Für die Umwandlung eines Integer-Wertes in einen String können Sie die Methode `Integer.toString(...)` verwenden.



(a) (b)

Abbildung 1: Ausgabe eines Arrays in Balkenform. (a) 50 Werte im Intervall $[0, 20[$ und (b) 100 Werte im Intervall $[0, 10[$.

Aufgabe 3

Implementieren Sie folgende Aufgabenstellung:

- Implementieren Sie eine Methode `extractSentences`:

  ```
  String[] extractSentences(String longString)
  ```

  Diese Methode nimmt einen String entgegen und zerlegt diesen in seine einzelnen Sätze, welche durch einen Punkt getrennt sind. Alle gefundenen Sätze werden in einem String-Array abgelegt und am Ende zurückgegeben.

  Annahmen:

  - Wird ein String ohne Punkt übergeben, so wird dieser String ignoriert (kein Satz vorhanden).
  - Hört der String nicht mit einem Punkt auf, wird alles vom letzten Punkt (Satzende) bis zum Ende des Strings ignoriert.
  - Es können beliebig viele Sätze im String vorkommen.

  Beispiel:

  `"Ich gehe jetzt einkaufen.Das Auto ist rot.Morgen gehen wir ins Kino."`

  wird zerlegt in ein String-Array mit dem Inhalt:

  ```
  {"Ich gehe jetzt einkaufen.",
  "Das Auto ist rot.",
  "Morgen gehen wir ins Kino."}.
  ```

(!) Implementieren Sie diese Methode selbständig. Sie dürfen für die Aufgabe **nur** die Methoden `charAt` und `length` aus der String-Klasse verwenden.

- Implementieren Sie eine Methode `printArray`:

  ```
  void printArray(String[] workArray)
  ```

  Diese Methode nimmt ein String-Array entgegen und gibt jeden einzelnen Eintrag in einer eigenen Zeile aus.

- Implementieren Sie eine Methode `countSentenceLength`:

  ```
  String[][] countSentenceLength(String[] workArray)
  ```

  Diese Methode nimmt ein String-Array entgegen (z.B. das Array, welches von der Methode `extractSentences` retourniert wird) und retourniert ein zweidimensionales String-Array. In jedem Unterarray dieses Arrays steht jeweils an der ersten Stelle ein Satz aus `workArray` und an der zweiten Stelle die Länge dieses Satzes als String codiert.

(!) Für die Umwandlung eines Integer-Wertes in einen String können Sie die Methode `Integer.toString(...)` verwenden.

- Implementieren Sie eine Methode `printArray`:

  ```
  void printArray(String[][] workArray)
  ```

  Diese Methode nimmt ein zweidimensionales String-Array entgegen und gibt den Inhalt des Arrays formatiert aus. Die Zeilen des Arrays werden untereinander ausgegeben und jedes Element innerhalb einer Zeile wird durch ein `"-->"` getrennt. Für das obige Beispiel mit den Sätzen und Längen wird folgende Ausgabe erzeugt:

  ```
  Ich gehe jetzt einkaufen. --> 25
  Das Auto ist rot. --> 17
  Morgen gehen wir ins Kino. --> 26
  ```

Zusatzfrage(n):

1. Wie wird ein String-Array initialisiert?
2. Was ist der Unterschied zwischen `String myString = "";` und `String myString = null;`?

Aufgabe 4

Implementieren Sie folgende Aufgabenstellung:

- Implementieren Sie eine Methode `genFilledArray`:

  ```
  int[][] genFilledArray(int centerNumber)
  ```

  Die Methode erzeugt ein zweidimensionales Array der Größe `centerNumber` × `centerNumber`. Überprüfen Sie, ob der Wert für `centerNumber` ungerade ist, ansonsten geben Sie den Wert `null` zurück. Die Methode soll mit einer beliebigen ungeraden Zahl für `centerNumber` funktionieren. Kann ein Array erzeugt werden, dann werden die einzelnen Stellen mit Zahlen belegt, die nach einem bestimmten Muster generiert werden. Die mittlere Position des Arrays erhält dabei immer den Wert von `centerNumber`. Zwei Beispiele finden Sie in den Abbildungen 2a für `centerNumber = 5` und 2b für `centerNumber = 9`. Sie dürfen sich selbst eine Vorgehensweise für die Berechnung der Einträge an den verschiedenen Stellen überlegen, aber am Ende muss Ihre Implementierung Belegungen produzieren, die den unten abgebildeten Beispielen entsprechen (und auch für andere Werte von `centerNumber` funktionieren).

1	2	3	2	1
2	3	4	3	2
3	4	5	4	3
2	3	4	3	2
1	2	3	2	1

(a)

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

(b)

Abbildung 2: Befüllte Arrays mit (a) dem Wert 5 in der Mitte und (b) mit dem Wert 9 in der Mitte.

- Implementieren Sie eine Methode `calcSumInArray`:

  ```
  int[][] calcSumInArray(int[][] workArray)
  ```

  Die Methode nimmt ein ganzzahliges zweidimensionales Array als Parameter entgegen und berechnet für jede Position eine Summe, die in ein neues Array (Ergebnis-Array, das am Ende retourniert wird) an dieselbe Position geschrieben wird. Für Positionen, die nicht am Rand liegen, werden die Werte aus einem 3×3-Fenster, d.h. alle 8 Nachbarn plus der Wert in der Mitte (das ist die aktuelle Position), addiert und die Summe danach an die Stelle dieses Mittelpunktes des 3×3-Fensters geschrieben. Nimmt man aus Abbildung 2a das 3×3 Fenster von Position `[1][1]`, so bekommt man:
  ```
  1 2 3
  2 3 4
  3 4 5
  ```
  Berechnet man die Summe für dieses 3×3-Fenster, ergibt sich der Wert 27, welcher dann im

Ergebnis-Array an die Position `[1][1]` geschrieben wird (Abbildung 3a). Für die Elemente an den Rändern des Arrays wird die Summe mit den vorhandenen Nachbarn berechnet. So ergibt sich z. B. für einen Eckpunkt die Summe 8, da nur der Eckpunkt und drei Nachbarn zur Summe beitragen. Abbildung 3a und 3b zeigen die berechneten Summen für Abbildung 2a und 2b.

8	15	17	15	8
15	27	30	27	15
17	30	33	30	17
15	27	30	27	15
8	15	17	15	8

(a)

8	15	21	27	29	27	21	15	8
15	27	36	45	48	45	36	27	15
21	36	45	54	57	54	45	36	21
27	45	54	63	66	63	54	45	27
29	48	57	66	69	66	57	48	29
27	45	54	63	66	63	54	45	27
21	36	45	54	57	54	45	36	21
15	27	36	45	48	45	36	27	15
8	15	21	27	29	27	21	15	8

(b)

Abbildung 3: Ergebnis-Arrays mit den gebildeten Summen aus Abbildung 2a in (a) und aus Abbildung 2b in (b).

- Testen Sie Ihre Methoden mit den in `main` zur Verfügung gestellten Aufrufen.