Aufgabenblatt 4

Kompetenzstufe 1 & Kompetenzstufe 2

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihr IntelliJ-Projekt bis spätestens **Freitag, 11.12.2020 15:00 Uhr** in TUWEL hoch.
- Zusätzlich müssen Sie in TUWEL ankreuzen, welche Aufgaben Sie gelöst haben.
- 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.
- Erlaubt sind die Klassen `String`, `Math`, `Integer` und `StdDraw` oder Klassen, die in den Hinweisen zu den einzelnen Aufgaben aufscheinen.
- 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 eindimensionalen Arrays
- Rekursion mit eindimensionalen Arrays und Strings
- Massendatenverarbeitung und eindimensionale 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:

1. Warum kommt es in der Methode `printArray` (Zeile 35) zu einem Fehler (Exception)? Korrigieren Sie die 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.

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

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` auf einer Kopie gearbeitet wurde?

4. In Zeile 50 wird in `filledArray` an der Stelle 0 der Wert 2020 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 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:

(!) Sie können sich für die Ausgabe der Arrays Hilfsmethoden schreiben.

a) Erstellen Sie ein eindimensionales ganzzahliges Array und initialisieren Sie es mit den Werten `{1, 4, 7, 0, 3, 6, 2, 8}`. Führen Sie die Initialisierung des Arrays ohne Schleife durch! Geben Sie anschließend das Array auf der Konsole getrennt durch Beistriche aus. Achtung: Vermeiden Sie, dass nach dem letzten Element ein Beistrich ausgegeben wird.
Erwartete Ausgabe:
`1,4,7,0,3,6,2,8`

b) Erstellen Sie ein eindimensionales ganzzahliges Array der Länge 20 und initialisieren Sie es mit Werten der Viererreihe beginnend bei 12. 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:
`12 16 20 24 28 32 0 40 44 48 52 56 60 64 68 0 76 80 84 88`

c) Erstellen Sie ein eindimensionales ganzzahliges Array und initialisieren Sie es mit den Werten `{4, 8, 1, 5, 2}`. Nach der Erstellung und Initialisierung des Arrays soll ein neues Array erstellt werden, das die Inhalte des zuvor erstellten Array enthält und zusätzlich noch den Wert 100 an erste Stelle und den Wert 200 an letzter Stelle dazubekommt. Die Elemente zwischen ersten und letztem Element werden entsprechend dem zuvor erstellten Array befüllt. Geben Sie anschließend alle Elemente des neuen Arrays nebeneinander getrennt durch Leerzeichen auf der Konsole aus.
Erwartete Ausgabe:
`100 4 8 1 5 2 200`

d) Erstellen Sie ein eindimensionales ganzzahliges Array der Länge 15 und initialisieren Sie es mit den Werten von 15 bis 1. Geben Sie anschließend die Elemente des Arrays in umgekehrter Reihenfolge aus. Führen Sie die Ausgabe mit einer while-Schleife und mit einer for-Schleife durch und geben Sie diese 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:
```
while-Schleife: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
for-Schleife: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

e) Erstellen Sie ein eindimensionales ganzzahliges Array der Länge 15 und initialisieren Sie es mit den Werten `{61, 13, 19, 10, 2, 33, 41, 73, 0, 56, 94, 6, 45, 84, 23}`. Suchen Sie den größten und kleinsten Wert innerhalb des Arrays. Berechnen Sie zusätzlich den Durchschnittswert aller Arrayeinträge. Erstellen Sie anschließend ein weiteres eindimensionales ganzzahliges Array der Länge drei, wo Sie das ermittelte Minimum, den Durchschnittswert und das Maximum hintereinander abspeichern. Geben Sie beide Arrays auf der Konsole aus.
Erwartete Ausgabe:
```
61 13 19 10 2 33 41 73 0 56 94 6 45 84 23
0 37 94
```

Aufgabe 3 (1 Punkt)

Implementieren Sie folgende Aufgabenstellung:

- 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 im Intervall von [0, `maxNumber`[und gibt das Array anschließend zurück.
  Vorbedingungen: `length` > 0 und `maxNumber` > 0.

- Implementieren Sie eine Methode `filterMaxMinValue`:

  ```
  void filterMaxMinValue(int[] workArray)
  ```

  Diese Methode sucht den maximalen und minimalen ganzzahligen Wert innerhalb des Arrays `workArray` und setzt beide Werte auf -1. Wenn das Array nur gleiche Werte als Einträge aufweist, dann wird das erste Element als Maximum und Minimum betrachtet. Kommt das Minimum und Maximum mehrfach vor, dann wird immer das erste Auftreten (beginnend beim Index 0) innerhalb des Arrays als Minimum bzw. Maximum angesehen.
  Vorbedingungen: `workArray != null` und `workArray.length > 0`.

  Beispiele:
  `int[] array1 = new int[]{12, 13, 5, 23, 7, 14, 9, 2, 15, 19};`
  `filterMaxMinValue(array1)` führt zu `[12, 13, 5, -1, 7, 14, 9, -1, 15, 19]`
  `int[] array2 = new int[]{34, 14, 26, 18, 45, 21};`
  `filterMaxMinValue(array2)` führt zu `[34, -1, 26, 18, -1, 21]`
  `int[] array3 = new int[]{5, 5};`
  `filterMaxMinValue(array3)` führt zu `[-1, 5]`
  `int[] array4 = new int[]{3, 2, 1};`
  `filterMaxMinValue(array4)` führt zu `[-1, 2, -1]`
  `int[] array6 = new int[]{5, 5, 7, 7};`
  `filterMaxMinValue(array3)` führt zu `[-1, 5, -1, 7]`

- Implementieren Sie eine Methode `combineArrays`:

  ```
  int[] combineArrays(int[] workArray1, int[] workArray2)
  ```

  Diese Methode erstellt ein neues eindimensionales Array der Länge `workArray1.length + workArray2.length` und befüllt dieses mit den Elementen der beiden Arrays. Die Reihenfolge der Elemente bleibt erhalten, zuerst alle Elemente von `workArray1` und danach alle Elemente von `workArray2`.
  Vorbedingungen: `workArray1 != null`, `workArray1.length > 0`, `workArray2 != null` und `workArray2.length > 0`.

  Beispiele:
  `int[] array10 = new int[]{2, 6, 4, 5, 1};`
  `int[] array20 = new int[]{28, 34, 56};`
  `combineArrays(array10, array20)` liefert `[2, 6, 4, 5, 1, 28, 34, 56]` zurück

Aufgabe 4 (1 Punkt)

Implementieren Sie folgende Aufgabenstellung:

(!) Gilt für alle zu implementierenden Methoden: Sie 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.

- Implementieren Sie eine rekursive Methode `getHighestAverage`:

  ```
  int getHighestAverage(int[] workArray, int start, int end)
  ```

  Diese Methode soll den größten Durchschnittswert vierer benachbarter Arrayeinträge innerhalb des Arrays `workArray` im Indexintervall von `[start, end]` suchen und retournieren.
  Vorbedingungen: `workArray != null`, `start` < `end` und `start` und `end` sind gültige Indizes von `workArray`.

  Beispiele:
  `int[] array = {2, 13, 3, 16, 12, 4, 9, 14};`
  `getHighestAverage(array, 0, 7))` liefert `11` zurück
  `getHighestAverage(array, 4, 7))` liefert `9` zurück
  `getHighestAverage(array, 1, 4))` liefert `11` zurück
  `getHighestAverage(array, 0, 1))` liefert `0` zurück

- Implementieren Sie eine rekursive Methode `getHighestDifference`:

  ```
  int getHighestDifference(int[] workArray, int index)
  ```

  Diese Methode soll die größte absolute Differenz zweier benachbarter Arrayeinträge beginnend beim Index `index` des Arrays `workArray` finden und zurückgeben.
  Vorbedingungen: `workArray != null`, `workArray.length > 1` und `index` < `workArray.length-1`.

  Beispiele:
  `int[] array = {33, 23, 53, 29, 12, 34, 41, 44, 28, 13};`
  `getHighestDifference(array, 1))` liefert `30` zurück
  `getHighestDifference(array, 4))` liefert `22` zurück
  `getHighestDifference(array, 6))` liefert `16` zurück
  `getHighestDifference(array, 8))` liefert `15` zurück

- Implementieren Sie eine rekursive Methode `genArrayWithEvenNumbers`:

  ```
  int[] genArrayWithEvenNumbers(int[] workArray, int index)
  ```

  Diese Methode erstellt eine Kopie von `workArray`, filtert alle Zahlen im Indexintervall von `[index, workArray.length[` die nicht gerade sind und ersetzt diese durch 0. Für das Erzeugen der Kopie dürfen Sie die Methode `clone` verwenden. Die Werte werden nur in der Kopie verändert, sodass das Array `workArray` unverändert bleibt. Die gefilterte Kopie wird am Ende zurückgegeben.
  Vorbedingungen: `workArray != null`, `workArray.length > 0`, `index` ist ein gültiger Index von `workArray` und alle Arrayeinträge von `workArray` sind positive Zahlen.

  Beispiele:
  `int[] array = {35, 12, 7, 15, 20, 5, 50, 15, 26, 8};`
  `genArrayWithEvenNumbers(array, 0)` → `[0, 12, 0, 0, 20, 0, 50, 0, 26, 8]`
  `genArrayWithEvenNumbers(array, 9)` → `[35, 12, 7, 15, 20, 5, 50, 15, 26, 8]`
  `genArrayWithEvenNumbers(array, 8)` → `[35, 12, 7, 15, 20, 5, 50, 15, 26, 8]`
  `genArrayWithEvenNumbers(array, 4)` → `[35, 12, 7, 15, 20, 0, 50, 0, 26, 8]`

- 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[] array = {2, 4, 7, 10, -10, 4, 0, 0, 27, 11, 4, 6};`
  `containsValue(array,11)` liefert `true` zurück
  `containsValue(array,2)` liefert `true` zurück
  `containsValue(array,25)` liefert `false` zurück
  `containsValue(array,0)` liefert `true` zurück
  `containsValue(array,14)` liefert `false` zurück
  `containsValue(array,6)` liefert `true` zurück

Aufgabe 5 (2 Punkte)

Implementieren Sie folgende Aufgabenstellung:

- In diesem Beispiel sollen Sie aus einer csv-Datei Wetterdaten einlesen, auswerten und visualisieren. Dazu ist die Datei `weather_data.csv` in Ihrem Projekt bereitgestellt. Sie benötigen für die Aufgabe keinen Scanner und können die Datei mit der Klasse *In*[1] einlesen und verarbeiten. Diese Klasse ist in der Verwendung ähnlich zum Scanner und hat den Vorteil, dass Sie sich um kein Exception-Handling kümmern müssen.

- In dieser csv-Datei befinden sich verschiedenste Wetterdaten der Stadt Wien seit Jänner 1955. Jede Zeile entspricht dem Zeitintervall von einem Monat. Wir wollen aus den Daten die Gesamtsonnenstunden (in h) jedes Jahres von 1955 bis 2015 auslesen und visuell darstellen. Dafür muss die folgende Spalte (Spalte 16) in der csv-Datei betrachtet werden:

  `SUN_H:` Gibt die Anzahl der Sonnenstunden (in h) in einem Monat an.

  Beispiele:

  - Jänner 1955: Sonnenstunden = 45 Stunden.
  - Juni 1990: Sonnenstunden = 197 Stunden.
  - August 2005: Sonnenstunden = 208 Stunden.
  - Juli 2019: Sonnenstunden = 276 Stunden.

  Wir wollen diese Kategorie vom Jänner 1955 (Datensatz für Jänner 1955 ist in Zeile 2 zu finden) bis Dezember 2015 (Datensatz für Dezember 2015 ist in Zeile 733 zu finden) betrachten. Dazu müssen Sie in der Methode `main` die folgenden drei von Ihnen implementierten Methoden für das Auslesen, Verarbeiten und Darstellen aufrufen:

- Implementieren Sie eine Methode `readFileData`:

  ```
  String[] readFileData(String fileName, int lineStart, int lineEnd)
  ```

  Der Parameter `fileName` gibt die einzulesende Datei an. Mit den Parametern `lineStart` und `lineEnd` wird der Bereich angegeben, der eingelesen werden soll. Dazu wird ein String-Array angelegt, in das jede eingelesene Zeile als Arrayeintrag abgelegt wird. Das Array wird am Ende der Methode zurückgegeben.

- Implementieren Sie eine Methode `extractData`:

  ```
  void extractData(String[] dataArray, int[] resultArray,
          int numColumn, int entriesPerYear)
  ```

  Diese Methode extrahiert aus `dataArray` (beinhaltet alle eingelesenen Daten) die Anzahl der Sonnenstunden der gesuchten Jahre und schreibt diese in `resultArray`. Der Parameter `numColumn` bestimmt, an welcher Stelle in der betrachteten Zeile von `dataArray` der gewünschte Wert (z.B. Sonnenstunden Spalte 16) steht. Der Parameter `entriesPerYear`

[1] `https://introcs.cs.princeton.edu/java/stdlib/javadoc/In.html`

gibt an, in welcher Auflösung die Daten in `dataArray` gespeichert sind. In unserem Fall wird dieser Parameter 12 sein, da es immer 12 Einträge pro Jahr gibt. Es soll die Anzahl an Sonnenstunden immer als Summe für ein ganzes Jahr ermittelt werden. Die so errechnete Summe wird in das Array `resultArray` geschrieben, das die Länge 61 hat. Zum Beispiel wird die Anzahl der Sonnenstunden des Jahres 1955 in `resultArray[0]` gespeichert und die Anzahl aller Sonnenstunden des Jahres 2015 in `resultArray[60]`.

(!) Für diese Methode können die Methoden `split` der Klasse `String` und `parseInt` der Klasse `Integer` hilfreich sein.

- Implementieren Sie eine Methode `drawChart`:

```
void drawChart(int[] sunHours)
```

Im ganzzahligen Array `sunHours` (Sonnenstunden) sind alle Daten für die Darstellung in Abbildung 1 enthalten. Auf der x-Achse finden Sie die Jahre (19)55 bis (20)15. Auf der y-Achse wird für jedes Jahr die Anzahl der Sonnenstunden dargestellt. Zusätzlich wird die y-Achse links und rechts mit den Maximal- und Minimalwerten der Sonnenstunden beschriftet. Bei diesen Werten wird zusätzlich eine horizontale schwarze Linie eingezeichnet. Für die Kennzeichnung der einzelnen Balken soll innerhalb der orangen Balken die Jahreszahl geschrieben werden. Zusätzlich werden die Balken an der oberen Seite mittels einer schwarzen Linie verbunden.

Für die gezeigte Grafik wurden folgende Zeichenparameter verwendet:

  - Vier Sonnenstunden entsprechen einem Pixel auf der y-Achse.
  - Abstand des ersten und letzten Balkens zum Rand entspricht 30 Pixel.
  - Abstand zwischen den Balken entspricht 5 Pixel.
  - Die Balkenbreite entspricht 15 Pixel.
  - Die Liniendicke der schwarzen Balkenverbindungslinien entspricht 0.005.
  - Die Liniendicke der horizontalen schwarzen Linien entspricht 0.002.
  - Die Schriftformatierung für Sonnenstunden (y-Achse) und den entsprechenden Jahren (x-Achse) entspricht `Font("Times", Font.PLAIN, 10)`.
  - Der Text für die Beschriftung rechts und links wurde um jeweils 15 Pixel eingerückt.

(!) Wir haben bei dieser Aufgabe auf Vorbedingungen verzichtet. Achten Sie darauf, dass die Methoden von Ihrer Implementierung in der main-Methode mit sinnvollen Werten (z.B. keine null-Referenzen, Dateiname ist korrekt und Datei existiert, Arrays haben korrekte Längen etc.) aufgerufen werden.

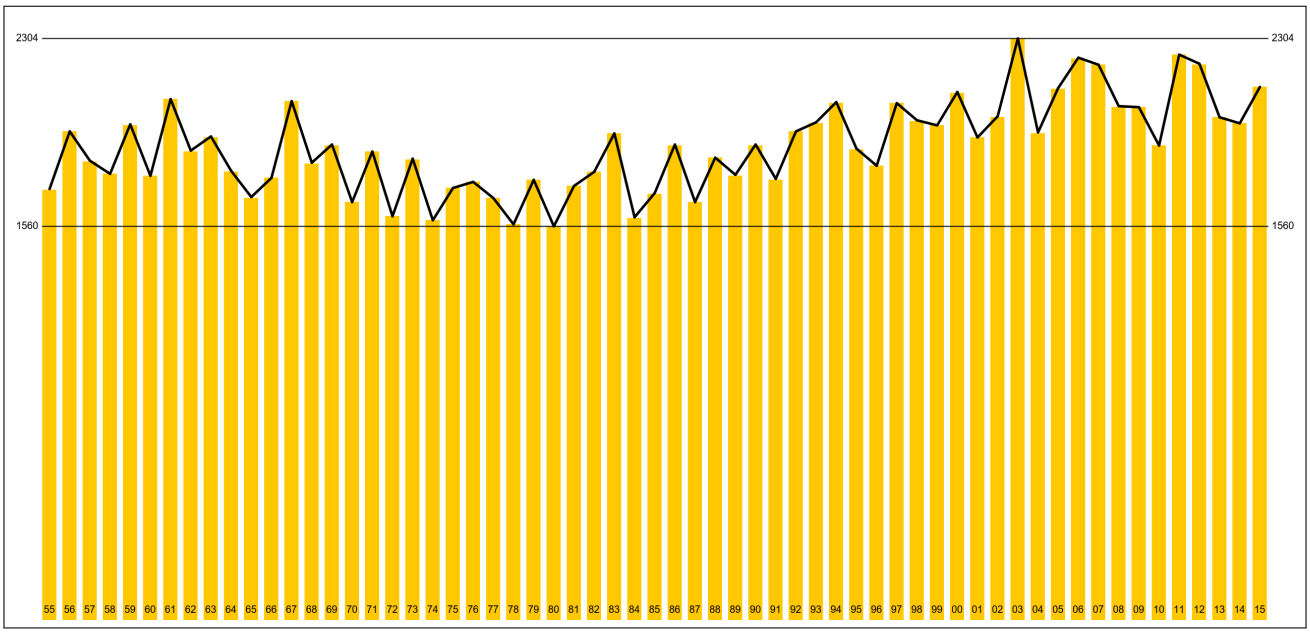

Abbildung 1: Darstellung der Sonnenstunden (orange) der Jahre 1955 bis 2015.