Übung04: Arrays, Zeichen, Strings

Abgabetermin: 05.04.19, 15:30 Uhr

Name: _____ Matrikelnummer: _____

Aufgabe	**schriftlich abzugeben**	**elektronisch abzugeben**	**gelöst**
Aufgabe11	Prosabeschreibung, Java Programm, Inhalt der Ausgabedatei	Java Programm	☐
Aufgabe12	Prosabeschreibung, Java Programm, Testfälle	Java Programm	☐

Achtung!

Bitte auf diesem Deckblatt:

- **Name** und **Matrikelnummer** ausfüllen,
- gelöste Aufgaben **ankreuzen**

und dann der schriftlichen Abgabe als erste Seite anheften.

Aufgabe11: Measurements (12 Punkte)

Gegeben sei eine Datei *measurements.txt* bestehend aus mehreren Zeilen, wobei jede Zeile einen Temperaturmesswert zwischen 0.0 und 50.0 °C beinhaltet. Entwicklen Sie ein Java Programm, das die Messwerte einliest und wie folgt verarbeitet:

- Ein Schiebefenster (Sliding Window) der Länge zehn soll durch die Messwerte geschoben werden, und für die aktuellen Werte im Fenster soll der Durchschnitt berechnet und ausgegeben werden.
- Ein Histogramm soll für die Anzahl der Messwerte in den jeweiligen Kategorien erstellt und ausgegeben werden. Die Kategorien sind wie folgt aufgeteilt:
  - < 10.0 °C
  - >= 10 und < 20 °C
  - >= 20 und < 30 °C
  - >= 30 °C

Die Ergebnisse, d.h. die Mittelwerte und das Histogramm, sollen in eine Datei *results.txt* ausgegeben werden. Sowohl das Schiebefenster, als auch das Histogramm, soll unter Verwendung von Arrays implementiert werden.

Hinweise:

- Verwenden Sie für das Schiebefenster ein double-Array der Länge zehn. Wenn Sie einen neuen Wert aus der Datei einlesen, verschieben Sie alle aktuellen Werte im Fenster um eine Position nach hinten, sodass der erste Wert vorne im Fenster rausfällt und die letzte Stelle hinten im Fenster frei wird für den neuen Wert. Berechnen Sie den Mittelwert nur, wenn das Fenster voll ist, also wenn Sie mind. zehn Messwerte eingelesen haben.
- Verwenden Sie für das Zählen der Werte in den vier Histogrammklassen ein int-Array der Länge vier. Die Stellen null, eins, zwei und drei im Array repräsentieren die Anzahl der Werte in der jeweiligen Klasse.
- Sie können `Out.println(String.format(java.util.Locale.US, formatString, values))` verwenden, um über einen Format-String die Messwerte mit Punkten statt Beistrichen auszugeben.

Beispiel Ausgabedatei:

```
Window Means:
20.61 19.91 19.43 19.16 19.18 19.01 18.62 18.25 17.49 16.98 16.51 16.01 15.47 15.06 14.38 13.77 …
Histogram:
[ 0.0, 10.0):  181
[10.0, 20.0):  136
[20.0, 30.0):   48
[30.0, 50.0]:    0
```

Abzugeben sind:

- Prosabeschreibung
- Java Programm
- Inhalt der Ausgabedatei für die gegebene Eingabedatei

Aufgabe12: CeasarCipher (12 Punkte)

Schon Cäsar wusste um die Bedeutung der Kryptographie und so geht die Geschichte, dass er einen Angriffsbefehl verschlüsselt an seine Befehlshaber sandte. Er verwendete dabei ein einfaches Verfahren, welches heute unter dem Namen Cäsar-Chiffre bekannt ist. Dabei wird das Alphabet um eine bestimmte Anzahl von Buchstaben verschoben, z.B. um zwei Zeichen. Für einen gegebenen Text bekommt dann das Zeichen 'a' die Bedeutung des Zeichens 'c' und ein 'b' die des 'd' usw. Dabei dient als Schlüssel die Anzahl der Zeichen, um die verschoben wird.

Siehe folgende Darstellung:

```
Schlüssel:      2
Klartext:       a b c d e f g h i j k l m n o p q r s t u v w x y z
Chiffre:        c d e f g h i j k l m n o p q r s t u v w x y z a b
```

Schreiben Sie ein Java Programm, mit dem man Texte verschlüsseln und wieder entschlüsseln kann. Dabei sollen Buchstaben nach dem Cäsar-Chiffre transformiert werden. Alle anderen Zeichen werden unverändert übernommen.

Das Programm soll über einen Dialog wie im folgenden Beispiel gezeigt bedienbar sein (blau und fett sind Eingaben des Benutzers, grauer Text sind zusätzliche Infos, die eigentlich nicht in der Ausgabe erscheinen). Man wählt zuerst die Operation aus: 'e' für encrypt (verschlüsseln), 'd' für decrypt (entschlüsseln) und 'q' für quit (beenden). Dann wird ein Dateiname und ein Schlüssel als Zahl eingegeben. Beim Verschlüsseln kann man eine Zeile Text eingeben, der chiffriert in die gegebene Datei geschrieben wird. Beim Entschlüsseln wird der chiffrierte Text aus der gegebenen Datei gelesen und in dechiffrierter Form auf der Konsole ausgegeben.

```
# java CaesarCipher
'e' for encrypt, 'd' for decrypt, 'q' to quit: e
Filename: text.txt
Key: 12
Text: Dies ist ein Text, der verschluesselt werden soll.
// nach dieser Operation beinhaltet die Datei text.txt folgenden Text:
// Puqe uef quz Fqjf, pqd hqdeotxgqeeqxf iqdpqz eaxx.
'e' for encrypt, 'd' for decrypt, 'q' to quit: d
Filename: text.txt
Key: 6 // falscher Schlüssel
Text: Joky oyz kot Zkdz, jkx bkxyinrakyykrz ckxjkt yurr.
'e' for encrypt, 'd' for decrypt, 'q' to quit: d
Filename: text.txt
Key: 12 // richtiger Schlüssel
Text: Dies ist ein Text, der verschluesselt werden soll.
'e' for encrypt, 'd' for decrypt, 'q' to quit: q
```

Hinweise:

Verwenden Sie folgende Methoden zum Prüfen von Zeichen:

- `boolean Character.isLetter(char ch)`
- `boolean Character.isUpperCase(char ch)`
- `boolean Character.isLowerCase(char ch)`

Verwenden Sie die Klasse StringBuilder, um den chiffrierten bzw. dechiffrierten Text zu produzieren. Erzeugen Sie dabei ein StringBuilder Objekt mit dem eingelesenen Text als Parameter und rufen Sie Methoden des StringBuilders auf, um den Text zu manipulieren.

Beispiel:

```
StringBuilder cipher = new StringBuilder(text);
for (int i = i; i < cipher.length(); i++) {
  char oldChar = cipher.charAt(i);
  char newChar = /* transform oldChar */ …;
  cipher.setCharAt(i, newChar);
}
Out.print(cipher);
```

Vermeiden Sie unnötige Code-Duplizierung und versuchen Sie, die beiden Fälle für ver- und entschlüsseln gleich zu behandeln. Beides kann als die selbe Transformation gesehen werden, wenn man beim Entschlüsseln den eingelesenen Schlüssel negiert.

Achten Sie besonders darauf, dass sich die produzierten Zeichen in gültigen Bereichen befinden ('a' – 'z' bzw. 'A' – 'Z'). Rechnen Sie weiters, um den Offset für die Transformation zu erhalten, den eingegebenen Schlüssel modulo 26.

Abzugeben sind:

- Prosabeschreibung
- Java Programm
- mind. zwei Testfälle:
  - Dialogausgaben und verschlüsselte Inhalte der Dateien