Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences
Fachbereich Informatik
Prof. Dr. Rudolf Berrendorf

Einführung in die Programmierung
WS 19/20
Ausgabe : 11.11.2019
Bearbeiten bis : 17.11.2019

Übung 7

Aufgabe 1 :
Schreiben Sie ein Java-Programm, das den Inhalt des Übergabeparameters `args` (ein Feld von Strings) aus `main` zeilenweise auf dem Bildschirm ausgibt (pro Feldelement eine Zeile).

Aufgabe 2 :

(a) Schreiben Sie eine Methode, die ein int-Feld und einen int-Wert annimmt und zurückgibt, wieviele Einträge des Feldes gleich dem Wert sind.
Beispiel: Das Ergebnis zum Feld `[1,2,2,4,3,2]` und Wert 2 ist 3.

(b) Schreiben Sie eine Methode, die ein int-Feld annimmt und ein neues int-Feld zurückgibt, das die Einträge des übergebenen Feldes in umgekehrter Reihenfolge enthält.
Beispiel: Das Ergebnis zum Feld `[1,2,2,4,3,2]` wäre das Feld `[2,3,4,2,2,1]`.

(c) Schreiben Sie eine Methode, die ein int-Feld annimmt und es wieder zurückgibt, nachdem die Reihenfolge der Einträge innerhalb des Feldes umgekehrt wurde.
Beispiel: Das Ergebnis zum Feld `[1,2,2,4,3,2]` wäre das *gleiche* Ursprungsfeld, allerdings mit dem Inhalt `[2,3,4,2,2,1]`.

Aufgabe 3 :
Geben Sie eine Methode an, die für zwei int-Felder als Methodenparameter ein Ergebnisfeld als Methodenresultat liefert. Der Inhalt des Ergebnisfeldes entspricht dem Hintereinanderfügen der Inhalte der beiden Argumentfelder. Im Resultatfeld steht also zu Beginn der Inhalt des ersten Argumentfeldes und danach folgend der Inhalt des zweiten Argumentfeldes.

Beispiel: Für die beiden Felder `[1,2,3]` und `[4,5,6]` würde die Methode als Ergebnis das Feld `[1,2,3,4,5,6]` liefern.

Aufgabe 4 :
Schreiben Sie eine Methode, die zwei gleichgroße eindimensionale Felder $a, b \in \mathbb{R}^n$ vom Java Basistyp `double` auf gleichen Inhalt untersucht und `true` liefert, wenn beide Felder inhaltsgleich sind bzgl. dieses Kriteriums. Die beiden Felder gelten als gleich, wenn für alle möglichen Indexwerte i für die Elemente a_i und b_i gilt: $|a_i - b_i| < \varepsilon$. Der Wert für ε wird der Methode neben den beiden Feldern mit übergeben.

Schreiben Sie eine zweite Methode, die dies analog für zweidimensionale Felder untersucht.

Aufgabe 5 :
Schreiben Sie eine Methode, die eine zweidimensionale quadratische Matrix transponiert, d.h. der Wert a[i][j] steht anschließend in a[j][i]. Diese Methode hat den Resultattyp `void`, da die existierende Matrix, die an die Methode übergeben wird, direkt verändert wird.

Sie dürfen in der Methode keinen nennenswerten weiteren Speicherplatz anlegen, also zum Beispiel keine weitere Matrix anlegen.

Beispiel: transponieren $\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}\right) = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$

Hinweise:

- Wie immer, überlegen Sie zuerst, bevor Sie programmieren. Führen Sie den Ansatz, den Sie sich überlegt haben, zuerst an einer kleinen Beispielmatrix (s.o.) mit Bleistift und Papier durch.
- Wie müsste die Aufgabenstellung (und die Lösung) verändert werden, wenn man auch nicht-quadratische Matrizen zulässt?

Aufgabe 6 :
Eine Permutation der Zahlen $0,...,N-1$ ist eine beliebige Anordung dieser Zahlen, so dass jede Zahl genau ein mal vorkommt. Beispiele für $N=3$: $0,1,2$ und $2,1,0$ und $1,0,2$ (es gibt weitere). Eine solche Permutation lässt sich in Java in einem int-Feld darstellen, etwa `int[] a = {0,1,2};`.

Zu einer solchen Permutation a ist die inverse Permutation so definiert, dass gilt: `a[b[i]] = b[a[i]] = i` für alle $i = 0,...,N-1$.

Schreiben Sie eine Methode `public static int[] inversePermutation(int[] a)`, die zu einem Feld mit einer Permutation der Zahlen $0,...,N-1$ ein neues Feld `b` gleicher Länge erzeugt und als Resultat der Methode liefert. Der Inhalt des Feldes `b` soll die inverse Permutation von `a` haben.

Beispiele:

1. Zu `[2, 0, 1]` ist das Resultat das Feld `[1, 2, 0]`
2. Zu `[3, 0, 4, 2, 1]` ist das Resultat das Feld `[1, 4, 3, 0, 2]`

Aufgabe 7 :
Entwickeln Sie eine Methode, die für eine Eingabezahl zwischen 20 und 99 den Zahltext zu dieser Zahl als Ergebnis liefert, also einen String. Beispiel: zu 45 würde der Text fuenfundvierzig geliefert.

Zusatzaufgabe: Überlegen Sie, wie man die Zahlen kleiner als 20 und größer als 99 behandeln könnte.

Aufgabe 8 :
Geben Sie ein Programm an, das zu einem nichtleeren Feld mit int-Werten ein Histogramm in einer vorgegebenen Form auf dem Bildschirm ausgibt. Die Werte des Feldes sind alles int-Werte zwischen 0 und 5, die das Programm in der Kommandozeile übergeben bekommt (siehe Beispielaufruf unten). Aus der Anzahl der Werte ergibt sich auch die Feldlänge.

Ein Histogramm gibt an, wie oft Werte in dem Datenfeld vorkommen. Die Ausgabe soll so sein, dass es zu jedem möglichen Wert eine Ausgabezeile gibt, von der Zeile für den Wert 0 bis zur Zeile für den Wert 5 (siehe Beispiele unten). Zuerst wird der Wert selber ausgegeben, direkt gefolgt von einem Doppelpunkt. Danach kommt ein Leerzeichen, dann eine bestimmte Anzahl an Sternen, dann ein Leerzeichen und dann in Klammern durch Komma getrennt die Anzahl der Vorkommen dieses Wertes und die Anzahl der Sternchen.

Die Anzahl an Sternchen n_{stern} zu einem Wert ergibt sich wie folgt. Zur höchsten ermittelten Wertanzahl n_{max} werden immer 10 Sternchen ausgegeben. Die Anzahl an Sternchen für alle anderen Werte ergibt sich proportional zu dem Referenzverhältnis n_{max} zu 10. Wenn also zum Beispiel $n_{max} = 4$ ist, so wird für eine Anzahl von 2 dann 5 Sternchen ausgegeben. Ermittelt man auf diese Weise eine nichtganzzahlige Sternchenanzahl, so wird diese Anzahl gerundet (siehe dazu die Klasse `Math`). Für nichtvorkommende Werte (die Anzahl ist also 0) wird kein Sternchen angezeigt.

Überlegen Sie sich zuerst anhand eines überschaubaren Beispiels, wie Sie die Gesamtaufgabe in Teilaufgaben zerlegen können und anschließend, wie Sie die Teilaufgaben in Java lösen können. Setzen Sie sinnvoll Methoden für Teillösungen ein.

Beispiele:

- Für das Feld `1 1 1 2` würden folgende Sternchen ausgegeben (in Zahlen dahinter die Anzahl der aufgetretenen Werte und die Anzahl an Sternchen):

```
> java Histogramm 1 1 1 2
0:  (0,0)
1: ********** (3,10)
2: *** (1,3)
3:  (0,0)
4:  (0,0)
5:  (0,0)
```

- Für das Feld **`0 1 0 0 3 1`** würden folgende Sternchen ausgegeben:

```
> java Histogramm 0 1 0 0 3 1
0: ********** (3,10)
1: ******* (2,7)
2:  (0,0)
3: *** (1,3)
4:  (0,0)
5:  (0,0)
```

Aufgabe 9 (1 Punkt) :

Dies ist eine Praktomataufgabe!

Gegeben sind n Punkte im zweidimensionalen reellen Raum mit ihren Koordinaten $(x_i, y_i), i = 0, ..., n-1$, worüber ein geschlossenes Polygon definiert wird. Dies bedeutet, dass vom 0. Punkt eine Verbindung/Linie zum 1. Punkt existiert, vom 1. Punkt zum 2. Punkt usw. und vom letzten Punkt zurück zum 0. Punkt.

Erfüllt ein solches Polygon bestimmte zusätzliche Eigenschaften (auf die hier nicht weiter eingegangen wird), so spricht man von einem einfachen Polygon. Zu einem einfachen Polygon kann man die durch das Polygon eingeschlossene Fläche A wie folgt berechnen:

$$A = \left| \frac{\sum_{i=0}^{n-1}((x_i + x_{i+1}) \cdot (y_{i+1} - y_i))}{2} \right|$$

Hierbei ist der Indexausdruck $i+1$ jeweils modulo n zu sehen. Für $n = 4$ ist also der letzte Summand $((x_3 + x_0) \cdot (y_0 - y_3))$ weil 3+1 modulo 4 = 0 ist.

Geben Sie eine Java-Methode **`public static float flaecheBerechnen(float[][] coord)`** (die Methode muss genau so definiert sein!) in einer Klasse **`FlaecheBerechnen`** an, die für ein zweidimensionales Feld **`coord`** mit Punktangaben die entsprechende Fläche berechnet und als Ergebnis (Rückgabewert) der Methode liefert (*nicht* mit **`System.out.println`** auf dem Bildschirm ausgeben!). Das Feld **`coord`** hat für n Punkte n Zeilen und zwei Spalten. Zu jedem Punkt i existieren damit zwei Angaben: **`coord[i][0]`** gibt die x-Koordinate des i-ten Punktes an und **`coord[i][1]`** die y-Koordinate des i-ten Punktes.

Beispiel: das Einheitsquadrat von $(0,0)$ bis $(1,1)$ besitzt 4 Eckpunkte $(0,0), (1,0), (1,1), (0,1)$ und hat einen Flächeninhalt von

$$\begin{aligned} A &= \frac{((0+1)\cdot(0-0)) + ((1+1)\cdot(1-0)) + ((1+0)\cdot(1-1)) + ((0+0)\cdot(0-1))}{2} \\ &= \frac{(1\cdot 0) + (2\cdot 1) + (1\cdot 0) + (0\cdot -1)}{2} \\ &= 1 \end{aligned}$$

Aufgabe 10 (1 Punkt) :

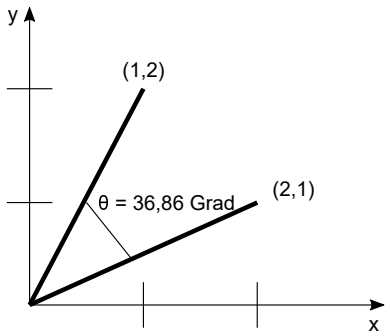
Dies ist eine Praktomataufgabe!

Für zwei reelle Vektoren $a = (a_1, ..., a_n)$ und $b = (b_1, ..., b_n) \in \mathbb{R}^n$ lässt sich der Winkel θ zwischen diesen Vektoren bestimmen über die Formel: $cos(\theta) = \frac{<a,b>}{||a|| \cdot ||b||}$ mit $< a,b >= \sum_{i=1}^{n} a_i b_i$ das Skalarprodukt ist und $||a|| = \sqrt{< a,a >}$ die Norm eines Vektors.

Hinweise:

- Ein Skalarprodukt ist also ein Wert, der die Summe über mehrere Produkte $a_i * b_i$ ist.
- $< a,a >$ bezeichnet also das Skalarprodukt eines Vektors a mit sich selbst.

Siehe nachfolgende Skizze für ein Beispiel zur Aufgabenstellung mit zwei Vektoren $(1,2)$ und $(2,1)$ im zweidimensionalen Raum:



Geben Sie in einer Java-Klasse `Vektorwinkel` eine Java-Methode `public static double winkel(double[] a, double[] b)` an, die für zwei beliebig aber gleich große Felder a, b, die zwei Vektoren darstellen, den Winkel in Grad berechnet, der durch die beiden Vektoren eingeschlossen wird.

Entwickeln Sie eigene Methoden zur Berechnung des Skalarprodukts und der Norm für beliebig lange Vektoren, die Sie sinnvoll zur Winkelberechnung einsetzen.

Da das Ergebnis der trigonometrischen Funktion in Java in Radiant ist, nutzen Sie bei sich eine weitere Methode, die eine Umrechnung von Radiant nach Grad bewirkt. Die Umrechnungsformel für einen Winkel ϕ gegeben in Radiant zu einer Winkelangabe in Grad ist: $grad = \phi * 180/\pi$.

Hinweise:

- Die Arcuscosinusfunktion, die Sie benötigen (oben in der Formel steht ja $cos(\theta) = ...$ und Sie wollen θ), lässt sich in Java über `Math.acos(x)` aufrufen, die Wurzelfunktion über `Math.sqrt(x)`, π über `Math.PI`.
- Die Wert in Java für das erste Beispiel unten ist bei einer Berechnung 45.00000000000001 (statt 45). Der Praktomat erlaubt solche Toleranzen.

Beispiele:

1. Für die beiden Vektoren $(1,0)$ und $(1,1)$ ist der Winkel 45 Grad.
2. Für die beiden Vektoren $(1,2)$ und $(2,1)$ ist der Winkel 36.86989764584404 Grad.
3. Für die beiden Vektoren $(1,0,0)$ und $(0,1,0)$ ist der Winkel 90 Grad.
4. Für die beiden Vektoren $(1,1,1,1)$ und $(1,2,3,4)$ ist der Winkel 24.094842552110695 Grad.