Bereich: Arrays (eindimensional)

Mittelwert und Standardabweichung

Package: `de.dhbwka.java.exercise.arrays`	**Klasse:** `StandardDeviation`

Aufgabenstellung:

Schreiben Sie ein Programm `StandardDeviation`, das von den zufällig erzeugten Werten eines Arrays beliebiger Größe den Mittelwert und die Standardabweichung berechnet.
Die Formel für den Mittelwert ist:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Die Formel für die Standardabweichung lautet (ohne die Wurzel haben wir die *Varianz*):

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Das Array soll eine Größe von *n* = 100 haben, die zufälligen Werte können mit Hilfe der Methode `Math.random()` oder der Klasse `java.util.Random` erzeugt werden und sollen ganzzahlige Werte zwischen 0 und 10 haben.

Hinweis*:*
Potenzen können mit der Methode `Math.pow(double basis, double exponent)`, Quadratwurzeln mit der Methode `Math.sqrt(double zahl)` berechnet werden.

Beispielausgabe:

```
Mittelwert: 5.35
Standardabweichung: 3.2887165529514033
```

Bereich: Arrays (eindimensional)

Fibonacci-Folge

Package: `de.dhbwka.java.exercise.arrays`	**Klasse:** `Fibonacci`

Aufgabenstellung:

Mit den Startwerten $f_0 = 1$ und $f_1 = 1$ wird die Folge der Fibonacci-Zahlen erklärt durch

$$f_n = f_{n-1} + f_{n-2}$$

Schreiben Sie ein Programm `Fibonacci`, das die ersten 20 (50) Fibonacci-Zahlen berechnet, gemäß ihrem Index in einem Array speichert und die Zahlen anschließend ausgibt!

Bereich: Arrays (eindimensional)

Sieb des Eratostenes*

Package: `de.dhbwka.java.exercise.arrays`	**Klasse:** `Eratostenes`

Aufgabenstellung:

*Die *-Aufgaben sind für alle, die schon Erfahrung im Programmieren haben und/oder schon früher fertig geworden sind; manchmal anspruchsvoller, manchmal für Fleißige.*

Der folgende Algorithmus („Sieb des Eratostenes“) ermittelt alle Primzahlen zwischen 2 und einer vorgegebenen Grenze *n*:

(1) Menge *sieve* = alle natürlichen Zahlen von 2 bis *n*
(2) Menge *primes* = leere Menge
(3) wiederhole die Schritte (4) bis (6) solange, bis *sieve* leer wird
(4) bestimme die kleinste Zahl *min* in *sieve*
(5) füge *min* zu *primes* hinzu
(6) entferne *min* und alle seine ganzzahligen Vielfachen aus *sieve*

Schreiben Sie ein Programm, das diesen Algorithmus implementiert, z.B. für *n = 100*.

Tipp:
Sie können ein Array von *n* (oder auch *n+1*) `boolean`-Werten zur Implementierung des Siebs verwenden.

Bereich: Arrays (eindimensional)

Betrag eines Vektors

Package: `de.dhbwka.java.exercise.arrays`	**Klasse:** `Norm`

Aufgabenstellung:

Der Betrag eines Vektors $x = (x_1, x_2, ..., x_n)$ ist definiert als

$$\sqrt{x_1 \cdot x_1 + x_2 \cdot x_2 + \cdots + x_n \cdot x_n}$$

Erstellen Sie ein Programm `Norm`, das einen Vektor mit *n* Komponenten in ein Array einliest, den Betrag des Vektors berechnet und das Ergebnis ausgibt!
Lesen Sie zunächst *n* ein!

Beispielausgabe:

```
Bitte Anzahl der Elemente n eingeben: 3
Bitte x_0 eingeben: 1
Bitte x_1 eingeben: 2
Bitte x_2 eingeben: 4
Der Betrag von x ist 4.58257569495584
```

Bereich: Arrays (eindimensional)

Skalarprodukt zweier Vektoren

Package: `de.dhbwka.java.exercise.arrays`	**Klasse:** `DotProduct`

Aufgabenstellung:

Das Skalarprodukt zweier Vektoren $x = (x_1, x_2, ..., x_n)$ und $y = (y_1, y_2, ..., y_n)$ ist definiert als

$$x_1 \cdot y_1 + x_2 \cdot y_2 + \cdots + x_n \cdot y_n$$

Erstellen Sie ein Programm `DotProduct`, das zwei Vektoren mit jeweils *n* Komponenten jeweils in ein Array einliest, dann das Skalarprodukt der beiden Vektoren berechnet und schließlich das Ergebnis ausgibt!
Lesen Sie zunächst *n* ein!

Beispielausgabe:

```
Bitte Anzahl der Elemente n eingeben: 3
Bitte x_0 eingeben: 5
Bitte x_1 eingeben: 3
Bitte x_2 eingeben: 1
Bitte y_0 eingeben: -2
Bitte y_1 eingeben: 4
Bitte y_2 eingeben: 9
Das Skalarprodukt von x und y ist 11
```

Bereich: Arrays (eindimensional)

Sortieren mit Bubblesort

Package: `de.dhbwka.java.exercise.arrays`	**Klasse:** `BubbleSort`

Aufgabenstellung:

Der Bubblesort-Sortieralgorithmus arbeitet nach der folgenden Idee:
Je zwei nebeneinanderliegende Elemente eines Feldes werden verglichen; sie werden vertauscht, wenn sie in falscher Reihenfolge stehen. Das wird solange wiederholt, bis alle benachbarten Elemente richtig sortiert sind.

Beispiel:
`(5,3,1,2)` → `(3,5,1,2)` → `(3,1,5,2)` → `(3,1,2,5)` → `(1,3,2,5)` → `(1,2,3,5)`

Schreiben Sie ein Programm `BubbleSort` zum aufsteigenden Sortieren von ganzen Zahlen!

Lesen Sie die Anzahl *n* der zu sortierenden Zahlen sowie diese *n* Zahlen von der Konsole ein und speichern Sie die Zahlen in einem Array.
Sortieren Sie dann das Array mit Hilfe von Bubblesort in aufsteigender Reihenfolge und geben Sie am Ende die sortierten Zahlen aus.

Beispielausgabe:

```
Bitte Anzahl der Elemente n eingeben: 4
Zahl 0 eingeben: 5
Zahl 1 eingeben: 3
Zahl 2 eingeben: 1
Zahl 3 eingeben: 2
Sortiert: 1 2 3 5
```