Introduction

Bienvenue dans le premier laboratoire!

Pour chaque étape, le fichier dans lequel implémenter l'exercice se trouvera à **gauche** et les instructions se trouveront à **droite**, sur la page de ce guide.

Pour tester vos méthodes, vous pourrez **exécuter** votre programme depuis le guide, ce qui appellera le point d'entrée, c'est-à-dire la méthode `main`. Toute sortie, par exemple avec `System.out.println`, sera affichée directement dans le guide.

Validez vos méthodes à l'aide des **tests d'évaluation** fournis dans le guide. Vous pouvez lancer ces tests unitaires autant de fois que nécessaire. Ce sont ces mêmes tests qui serviront pour l'attribution de la note.

warning

Important

Assurez-vous d'avoir lancé **tous les tests d'évaluation** avant la remise du travail, car ++les résultats des tests sont utilisés pour calculer automatiquement la note++.

Votre travail sera **automatiquement remis à la date limite**. Assurez-vous d'avoir complété l'ensemble des étapes avant cette date. Vous pouvez remettre votre travail en avance en choisissant *Mark as Completed* à la fin du guide ou dans le tableau de bord.

Tableaux

Somme

Implémentez la méthode `sum`, qui reçoit comme paramètres deux tableaux de nombres entiers et fait la somme des nombres formés par les deux tableaux.

Chacun de ces tableaux représente un nombre entier où le premier élément est le chiffre le plus significatif, c'est-à-dire celui qui a la valeur la plus grande selon sa position. Par exemple, le tableau avec les éléments $\begin{bmatrix}5 & 7 & 3\end{bmatrix}$ représente la valeur $5 \times 100 + 7 \times 10 + 3 \times 1 = 573$.

La méthode `sum` doit calculer les valeurs numériques des deux tableaux et doit retourner la somme comme un nombre entier.

info

À noter

- Les tableaux peuvent être de tailles différentes et chacun a au moins un élément
- Chaque élément des tableaux est un nombre entier entre 0 et 9
- On cherche une solution sans passer par une conversion en chaîne de caractères

Programme

Pour tester votre solution, appelez `sum` dans la méthode `main` de la classe avec les exemples suivants :

Premier tableau	Second tableau	Résultat
[3, 2, 8]	[4, 7, 1]	799
[2, 4]	[5, 2, 9]	553
[1, 5, 0]	[3, 4]	184

Matrices

Multiplication

Implémentez la méthode `multiply`, qui calcule la multiplication de deux matrices représentées comme des tableaux bidimensionnels.

La formule pour calculer la multiplication de la matrice A, composée de $n \times m$ éléments, avec la matrice B, composée de $m \times p$ éléments, est :

$$\begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \times \begin{bmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mp} \end{bmatrix} = \begin{bmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{np} \end{bmatrix}$$

où chaque élément c_{ij} est défini par :

$$c_{ij} = a_{i1} \times b_{1j} + \cdots + a_{im} \times b_{mj} = \sum_{k=1}^{m} (a_{ik} \times b_{kj})$$

info

À noter

- Les matrices peuvent être de tailles différentes et chacune contient au moins un élément
- La méthode `multiply` doit vérifier la condition nécessaire, en termes du nombre de colonnes et de lignes, avant de faire la multiplication. Si la condition n'est pas satisfaite, la méthode retourne le tableau `{{ Double.NaN }}` pour indiquer que c'est une valeur incorrecte.
- Pour vérifier vos tests, vous pouvez utiliser le multiplicateur de matrices en ligne de OnlineMSchool

Programme

Dans la méthode `main` de la classe, appelez la méthode `multiply` avec les exemples suivants et affichez les résultats :

Première matrice	Seconde matrice

		Résultat
$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$	$\begin{bmatrix} 6 & 7 \\ 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 36 & 41 \\ 64 & 73 \end{bmatrix}$
$\begin{bmatrix} 1 & 0 \\ 3 & 2 \end{bmatrix}$	$\begin{bmatrix} 8 & 6 & 2 \\ 4 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 8 & 6 & 2 \\ 32 & 20 & 6 \end{bmatrix}$
$\begin{bmatrix} 5 & 6 & 7 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 12 & 13 \end{bmatrix}$

info

À noter

La méthode `printMat` vous est fournie pour pouvoir facilement afficher le contenu d'une matrice.

Statistiques

La classe `Statistics` fournit des méthodes pour obtenir des informations sur des tableaux de nombres réels.

Vous implémenterez ces méthodes, puis écrirez un programme qui prendra en **argument** l'opération à effectuer et le tableau.

Méthodes

Nombre d'éléments

Implémentez la méthode `elemCount`, qui reçoit comme paramètre un **tableau de nombres réels** et qui retourne le **nombre d'éléments** dans le tableau.

Moyenne

Implémentez la méthode `average`, qui reçoit comme paramètre un **tableau de nombres réels** et qui retourne la **moyenne arithmétique** des nombres.

Maximum

Implémentez, `++`à l'aide d'une boucle`++`, la méthode `max`, qui reçoit comme paramètre un **tableau de nombres réels** et qui retourne la **plus grande valeur** du tableau.

Minimum

Implémentez, `++`à l'aide d'une boucle`++`, la méthode `min`, qui reçoit comme paramètre un **tableau de nombres réels** et qui retourne la **plus petite valeur** du tableau.

Tests

Vous pouvez effectuer vos propres tests à l'aide des boutons ci-dessous en ajoutant un **point d'entrée** (méthode `main`).

Programme

La méthode `main` de la classe reçoit comme premier argument un **nombre entier** pour sélectionner la ou les méthodes à appeler avec le reste des valeurs en argument.

Les valeurs possibles pour ce premier argument sont :
- **1** pour appeler la méthode `elemCount`
- **2** pour appeler la méthode `average`
- **3** pour appeler les méthodes `max` et `min`

Pour toutes les autres valeurs, la méthode affiche seulement le message `Invalid operation`, suivi d'un espace et du nombre reçu.

Utilisez une instruction `switch` pour faire l'appel à la méthode adéquate.

info

À noter

- Les arguments d'un programme sont les chaînes de caractères séparées par des espaces passées avec l'appel du programme. On les retrouve dans le paramètre de la méthode `main`.
- On peut obtenir la première valeur avec l'aide de la méthode statique *Integer.parseInt*, qui convertit une chaîne de caractères en un entier
- On peut procéder de façon similaire pour obtenir un nombre réel (*Float.parseFloat*)
- Pour les trois opérations, on doit avoir au moins une valeur d'entrée pour les calculs

Ajoutez et implémentez la méthode `main`, puis faites les exécutions suivantes dans le **terminal** :

Arguments	**Résultat**
`1 3 6.4 8.3 11 23.2 897.44`	`6`
`2 8 7 6 5 4`	`6.0`
`3 2 10 3 6 -10.5 35.9 963.2 123.4`	`963.2, -10.5`
`4 8 7 6 5 4`	`Invalid operation 4`

info

À noter

Pour tester votre programme avec des arguments, entrez la commande suivante :
`./run.sh exercise3/Statistics.java`
suivie d'un **espace et de vos arguments**. Par exemple, pour le deuxième cas :
`./run.sh exercise3/Statistics.java 2 8 7 6 5 4`