# Les exceptions dans le langage Java – TP 2

*Voir en annexe des rappels sur les tableaux à 2 dimensions.*

Le but du TP est de coder différentes méthodes qui s'appliqueront à des tableaux à 2 dimensions, ou autrement dit des matrices. Ces méthodes seront codées dans une classe outil nommée ***OutilTableau2D.***

Pour les différentes méthodes de la classe, le principe sera le suivant : si le traitement demandé n'est pas réalisable, la méthode propagera vers son appelant l'exception prédéfinie ***IllegalArgumentException*** accompagnée d'un message explicitant la cause de l'erreur. Ces messages sont définis en tant que constantes dans la classe ***OutilTableau2D.***

Certaines chaînes de caractères correspondant à un message d'erreur sont paramétrées par un format ***%d***. Ce format devra être remplacé par un nombre entier, numéro de ligne ou de colonne par exemple. La méthode qui permet d'effectuer cette substitution est la méthode ***format*** de la classe *String*.

Exemple `String.format("Ce TP porte le numéro %d et traite des exceptions.", 2)`

renvoie comme résultat la chaîne `"Ce TP porte le numéro 2 et traite des exceptions."`

Au fur et à mesure de l'écriture des méthodes, il vous sera demandé d'écrire un programme de test. Le but est de tester le bon fonctionnement dans le cas nominal (celui qui ne génère pas la levée de l'exception ***IllegalArgumentException***) et aussi dans le cas d'erreur (donc lorsque l'exception est levée).

## Préalable

Récupérer les fichiers ***OutilSasie.java***, ***OutilTableau2DACompleter.java*** et ***TestOutilTableau2DACompleter.java***. Modifier les noms des fichiers et les paquetages pour les adapter à votre organisation.

- La classe ***OutilSaisie*** ne doit pas être modifiée, en principe. Lisez attentivement son contenu. Les méthodes qu'elle contient vous aideront à coder d'autres méthodes.
- Vous remarquerez dans la classe de test, la définition de 4 matrices constantes qui serviront de jeu de test.

## Exercice 1 – *Calculer la somme des valeurs d'une ligne*

**1)** Dans la classe ***OutilTableau2D,*** coder la méthode nommée ***sommeLigne*** qui a en paramètre une matrice et un numéro de ligne. La méthode doit renvoyer la somme des valeurs de la matrice situées sur la ligne dont le numéro est donné argument. Si la configuration de la matrice ne permet pas de calculer le résultat, l'exception ***IllegalArgumentException*** sera levée accompagnée du message d'erreur adéquat (voir les messages définis en tant que constante).

**2)** Dans la classe ***TestOutilTableau2D***, compléter la méthode ***afficherSommeLigne*** de manière à ce qu'elle affiche la somme de la ligne dont le numéro est passé en argument, à partir de la matrice elle aussi passée en argument. Conformément au commentaire, si le numéro de la ligne est invalide, un message d'erreur sera affiché (voir aussi le résultat attendu pour la question 3).

**3)** Compléter la méthode ***testSommeLigne***. Cette méthode demandera 3 fois à l'utilisateur de saisir un numéro de ligne. Pour chaque numéro, elle affiche la somme des lignes ayant ce numéro, à partir des 4 matrices jeux de tests.

Exemple d'exécution

```
TEST DE LA METHODE sommeLigne :
---------------------------------

******** TEST 1/3 ********
Entrez un numéro de ligne : 1

  1)  Résultat pour la matrice carrée :
Somme de la ligne 1 = 31

  2)  Résultat pour la matrice rectangulaire :
Somme de la ligne 1 = 48

  3)  Résultat pour la matrice irrégulière :
Somme de la ligne 1 = 48

  4)  Résultat pour la matrice nulle :
Somme de la ligne 1 = Calcul impossible.
  Erreur. La matrice argument n'a pas été créée.

******** TEST 2/3 ********
Entrez un numéro de ligne : 3

  1)  Résultat pour la matrice carrée :
Somme de la ligne 3 = 26

  2)  Résultat pour la matrice rectangulaire :
Somme de la ligne 3 = 31

  3)  Résultat pour la matrice irrégulière :
Somme de la ligne 3 = 34

  4)  Résultat pour la matrice nulle :
Somme de la ligne 3 = Calcul impossible.
  Erreur. La matrice argument n'a pas été créée.

******** TEST 3/3 ********
Entrez un numéro de ligne : 8

  1)  Résultat pour la matrice carrée :
Somme de la ligne 8 = Calcul impossible.
  Erreur. La ligne numéro 8 n'existe pas.

  2)  Résultat pour la matrice rectangulaire :
Somme de la ligne 8 = Calcul impossible.
  Erreur. La ligne numéro 8 n'existe pas.

  3)  Résultat pour la matrice irrégulière :
Somme de la ligne 8 = Calcul impossible.
  Erreur. La ligne numéro 8 n'existe pas.

  4)  Résultat pour la matrice nulle :
Somme de la ligne 8 = Calcul impossible.
  Erreur. La matrice argument n'a pas été créée.
```

## Exercice 2 – *Afficher une matrice*

**1)** Dans la classe ***OutilTableau2D,*** coder la méthode nommée ***afficher*** qui a en paramètre une matrice. La méthode doit afficher sur la console les valeurs présentes dans la matrice. Pour aligner les valeurs en colonne, on pensera à utiliser des tabulations (caractère \t) ou éventuellement un affichage formaté avec *printf*.

Par exemple, pour la matrice carrée, on souhaite obtenir l'affichage suivant :

```
4     9     17    26    13
-3    2     8     -6    30
12    9     1     3     11
7     -2    0     6     15
11    10    -5    3     -1
```

**2)** Lancer la méthode de test ***testAfficher*** telle qu'elle est codée. Que constatez-vous ?

**3)** Modifier la méthode de test pour qu'elle affiche un message cohérent dans le cas d'erreur que vous avez identifié à la question précédente.

## Exercice 3 – *Déterminer si une matrice est carrée*

**1)** Dans la classe ***OutilTableau2D,*** coder la méthode nommée ***matriceCarree*** qui a en paramètre une matrice. La méthode doit déterminer si cette matrice est carrée ou pas, dit autrement déterminer si le nombre de lignes est égal au nombre de colonnes Si la configuration de la matrice ne permet pas de calculer le résultat, l'exception ***IllegalArgumentException*** sera levée accompagnée du message d'erreur adéquat.

**2)** Lancer la méthode de test ***testMatriceCarree*** telle qu'elle est codée. Que constatez-vous ?

**3)** Modifier la méthode de test pour qu'elle affiche un message cohérent dans le cas d'erreur que vous avez identifié à la question précédente. L'objectif est d'obtenir l'affichage suivant :

```
TEST DE LA METHODE matriceCarree :
----------------------------------

  1)  Résultat pour la matrice carrée : carrée

  2)  Résultat pour la matrice rectangulaire : pas carrée

  3)  Résultat pour la matrice irregulière : pas carrée

  4)  Résultat pour la matrice nulle :
Impossible de tester la matrice.
Erreur. La matrice argument n'a pas été créée.
```

## Exercice 4 – *Somme de la première diagonale*

**1)** Dans la classe ***OutilTableau2D,*** coder la méthode nommée ***sommePremiereDiagonale*** qui a en paramètre une matrice. La méthode doit calculer la somme des valeurs situées sur la première diagonale de la matrice, celle qui part de la première ligne et première colonne et se termine sur la dernière ligne et dernière colonne. Si la configuration de la matrice ne permet pas de calculer le résultat, l'exception ***IllegalArgumentException*** sera levée accompagnée du message d'erreur adéquat.

**2)** Dans la classe ***TestOutilTableau2D***, compléter la méthode ***afficherSommePremiereDiagonale*** de manière à ce qu'elle affiche la somme de diagonale de la matrice passée en argument. Conformément au commentaire, si l'affichage n'est pas possible, un message d'erreur sera affiché (voir aussi le résultat attendu à la question 3).

**3)** Lancer la méthode ***testSommePremiereDiag*** pour vérifier le bon fonctionnement de la méthode précédente et de la méthode ***sommePremiereDiag***. On doit obtenir le résultat suivant :

```
TEST DE LA METHODE sommePremiereDiag :
--------------------------------------

  1)  Résultat pour la matrice carrée :
      Somme de la diagonale = 12

  2)  Résultat pour la matrice rectangulaire :
      Somme de la diagonale = Calcul impossible.
      Erreur. La matrice argument n'est pas carrée.

  3)  Résultat pour la matrice irregulière :
      Somme de la diagonale = Calcul impossible.
      Erreur. La matrice argument n'est pas carrée.

  4)  Résultat pour la matrice nulle :
      Somme de la diagonale = Calcul impossible.
      Erreur. La matrice argument n'a pas été créée.
```

## Exercice 5 – *Somme d'une colonne*

**1)** Dans la classe ***OutilTableau2D,*** coder la méthode nommée ***sommeColonne*** qui a en paramètre une matrice et un numéro de colonne. La méthode doit calculer la somme des valeurs situées sur la colonne indiquée. Si la configuration de la matrice ne permet pas de calculer le résultat, l'exception ***IllegalArgumentException*** sera levée accompagnée du message d'erreur adéquat. Attention, il faudra bien prendre en considération qu'une matrice, dans le langage Java, peut être irrégulière.

**2)** Dans la classe ***TestOutilTableau2D***, compléter la méthode ***afficherSommeColonne*** de manière à ce qu'elle affiche la somme de la colonne dont le numéro est passé en argument, à partir de la matrice elle aussi passée en argument. Conformément au commentaire, si le numéro est invalide, un message d'erreur sera affiché (voir aussi le résultat attendu à la question 3).

**3)** Lancer la méthode ***testSommeColonne.*** Cette méthode demandera 3 fois à l'utilisateur de saisir un numéro de colonne. Pour chaque numéro, elle affiche la somme des colonnes ayant ce numéro, à partir des 4 matrices jeux de tests.

Exemple d'exécution

```
TEST DE LA METHODE sommeColonne :
---------------------------------

******** TEST 1/3 ********
Entrez un numéro de colonne : 1

  1)  Résultat pour la matrice carrée :
Somme de la colonne 1 = 28

  2)  Résultat pour la matrice rectangulaire :
Somme de la colonne 1 = 28

  3)  Résultat pour la matrice irrégulière :
Somme de la colonne 1 = 28

  4)  Résultat pour la matrice nulle :
Somme de la colonne 1 = Calcul impossible.
  Erreur. La matrice argument n'a pas été créée.

******** TEST 2/3 ********
Entrez un numéro de colonne : 5

  1)  Résultat pour la matrice carrée :
Somme de la colonne 5 = Calcul impossible.
  Erreur. La colonne numéro 5 n'existe pas pour la ligne 0.

  2)  Résultat pour la matrice rectangulaire :
Somme de la colonne 5 = 7

  3)  Résultat pour la matrice irrégulière :
Somme de la colonne 5 = Calcul impossible.
  Erreur. La colonne numéro 5 n'existe pas pour la ligne 0.

  4)  Résultat pour la matrice nulle :
Somme de la colonne 5 = Calcul impossible.
  Erreur. La matrice argument n'a pas été créée.

******** TEST 3/3 ********
Entrez un numéro de colonne : 9

  1)  Résultat pour la matrice carrée :
Somme de la colonne 9 = Calcul impossible.
  Erreur. La colonne numéro 9 n'existe pas pour la ligne 0.

  2)  Résultat pour la matrice rectangulaire :
Somme de la colonne 9 = Calcul impossible.
  Erreur. La colonne numéro 9 n'existe pas pour la ligne 0.

  3)  Résultat pour la matrice irrégulière :
Somme de la colonne 9 = Calcul impossible.
  Erreur. La colonne numéro 9 n'existe pas pour la ligne 0.

  4)  Résultat pour la matrice nulle :
Somme de la colonne 9 = Calcul impossible.
  Erreur. La matrice argument n'a pas été créée.
```

# Annexe – Rappels sur les tableaux à 2 dimensions

En Java, pour créer un tableau à 2 dimensions, il faut créer un tableau de tableaux, c'est-à-dire un tableau dont les éléments sont eux-mêmes des tableaux. On pourra donc créer des tableaux irréguliers dont les lignes n'auront pas toutes le même nombre de colonnes.

## Exemples de déclaration et création de tableaux à 2 dimensions

```
int[][] m1 = {new int[3], new int[2]};

int[][] m2;
m2 = new int[2][];              // création d'un tableau de 2 tableaux d'entiers
int[] ligne1 = new int[3];            // ligne1 référence un tableau de 3 entiers
int[] ligne2 = new int[2];            // ligne2 référence un tableau de 2 entiers
m2[0] = ligne1 ;                       // on affecte ces 2 références aux cases de m2
m2[1] = ligne2 ;

int[][] m3 = {{1, 2, 3}, {10, 11}};

int[][] m4 = new int[4][6];
```

Les tableaux *m1*, *m2* et *m3* comportent tous 2 lignes. Leur première ligne contient 3 colonnes et la deuxième 2 colonnes.

Le tableau *m3* est initialisé lors de sa création. Sa taille est déduite du nombre de valeurs initiales.

Le tableau *m4* possède 4 lignes et 6 colonnes. Il s'agit donc d'un tableau régulier.

Comme pour les tableaux à une dimension, les tableaux d'entiers à 2 dimensions sont initialisés, par défaut, à 0.

## Notations

`m3[0]` est une référence sur la première ligne de *m3*. Cette ligne est un tableau de 3 éléments.
`m3[0][1]` désigne l'élément situé sur la ligne 0, et sur la colonne 1 de *m3*. Cet élément a la valeur 1.
`m3[1][1]` désigne l'élément situé sur la ligne 1, et sur la colonne 1 de *m3*. Cet élément a la valeur 11.

`m3.length` vaut 2. C'est le nombre de lignes de *m3*.
`m3[0].length` vaut 3. C'est le nombre de colonnes de la ligne 0 de *m3*.
`m3[1].length` vaut 2. C'est le nombre de colonnes de la ligne 1 de *m3*.

## Utilisation de la boucle *for…each* pour afficher les éléments d'un tableau

```
for (int[] ligne : t) {         // la variable ligne va balayer toutes les lignes de t
    for (int v : ligne) {       // la variable v va balayer tous les éléments de ligne
        System.out.print(v + "  ");
    }
    System.out.println();
}
```

**Attention** : la boucle ***for...each*** ne peut pas contenir d'instructions permettant de modifier les cases du tableau parcouru.