Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2021
DEV2 – JAVL

DEV2 – JAVL – Laboratoire Java

TD 5 – Les tableaux à 2 dimensions

Résumé

Ce TD a pour objectif de mettre en œuvre les tableaux à deux dimensions.

Avant de commencer, vérifiez que votre projet est bien synchronisé avec la version sur le dépôt. N'oubliez pas de faire régulièrement des `commits`.

1 Exercices de base

Dans un package `g12345.dev2.td05`[1], créez une classe `Tableau2D`. Créez également une classe `MainExoBase` avec une méthode principale pour tester les méthodes relatives aux exercices de base.

1.1 Affichage, exercice 6 du syllabus d'algorithmique

Écrivez dans la classe `Tableau2D` une méthode de classe `afficher` qui reçoit un tableau à deux dimensions et l'affiche, ligne par ligne. N'oubliez pas d'en écrire la documentation. Testez la méthode écrite dans la méthode principale de la classe `MainExoBase`.

1.2 Les nuls, exercice 8 du syllabus d'algorithmique

Dans la classe `Tableau2D`, ajoutez une méthode de classe `proportionNuls` qui reçoit un tableau $(n \times m)$ d'entiers et qui retourne la proportion d'éléments nuls dans ce tableau.

Testez la méthode écrite dans la méthode principale de la classe `MainExoBase`.

Écrivez également des tests unitaires pour valider votre travail.

1.3 Tous positifs, exercice 11 du syllabus d'algorithmique

Écrivez une méthode qui reçoit un tableau $(n \times m)$ d'entiers et qui vérifie si tous les nombres qu'il contient sont strictement positifs. Bien sûr, vous veillerez à éviter tout travail inutile ; la rencontre d'un nombre négatif ou nul doit arrêter l'algorithme.

Testez la méthode écrite dans la méthode principale de la classe `MainExoBase`.

Écrivez également des tests unitaires pour valider votre travail.

1. Remplacez bien sûr 12345 par votre identifiant.

Version du document
1er mars 2021

2 Classe MagicSquare

Un carré magique d'ordre n est un tableau 2D de n lignes et de n colonnes. Il est construit de telle manière que le somme des éléments de chaque ligne, chaque colonne et chaque diagonale est une valeur constante (version orientée objet de l'exercice 13 du syllabus d'algorithmique).

Dans le package `g12345.dev2.td05`, créez une classe *MagicSquare*. Créez également une classe *MainMagicSquare* avec une méthode principale pour tester l'utilisation de la classe *MagicSquare*. Veillez régulièrement à tester valablement ce que vous codez.

2.1 Attributs et constructeurs

1. La classe `MagicSquare` possède les attributs (privés) suivants :
   - ▷ `int n` qui représente l'ordre du carré magique (nombre de lignes et de colonnes du tableau) ;
   - ▷ `int[][] values` qui est le tableau à deux dimensions qui contiendra les coefficients du carré magique ;
2. Ajoutez à cette classe un constructeur sans attribut qui initialise le carré magique {{4,9,2},{3,5,7},{8,1,6}}
3. Ajoutez à cette classe un constructeur qui reçoit comme attribut l'ordre du carré. Celui doit être `impair` et supérieur ou égal à 3. Vérifiez la validité du paramètre. Ce constructeur initialise le carré de la manière suivante
   - ▷ le nombre 1 est mis dans la case $[n-1][n/2]$ du tableau (division entière) ou $[n-1][(n+1)/2-1]$;
   - ▷ ayant placé un nombre k dans la case $[i][j]$, on place $k+1$ dans la case $[i+1][j+1]$;
   - ▷ si cette case est déjà occupée, on place le nombre $k+1$ dans la case $[i-1][j]$;
   - ▷ lorsqu'un indice calculé pour accéder à une case est *out of range*, on réinitilise cet indice à 0 ;
   - ▷ l'opération est répètée jusqu'à ce que le nombre $n*n$ soit placé ;

   Écrivez les tests unitaires adéquats.
4. Dans la méthode `main` de la classe `MainApp`, créez et affichez tous les carrés magiques d'ordre impair de 3 à 15.

> N'oubliez pas de faire vos commits !

2.2 Accesseurs

Écrivez les accesseurs de la classe `MagicSquare`

2.3 Autres méthodes de la classe MagicSquare

Écrivez les méthodes suivantes (n'oubliez de les documenter et d'écrire les tests unitaires utiles). Testez les différentes méthodes écrites dans la méthode main de la classe `MainApp`.

1. `sumLine(int num)` qui calcule la somme des éléments de la ligne num du tableau. Testez la validité de l'attribut ;

2. `sumColumn(int num)` qui calcule la somme des éléments de la colonne num du tableau.Testez la validité de l'attribut ;
3. `sumDiag(int num)` qui calcule la somme des éléments de la diagonnale descendante (si num vaut 1) ou ascendante (si num vaut 2). Testez la validité de l'attribut ;
4. `isMagicSquare()` qui teste et renvoie vrai ou faux que le carré soit magique ;
5. `displayMagicSquare()` qui affiche le carré magique (méthode à utiliser pour des carrés d'ordre < 20). Astuce : utiliser `System.out.printf()` ou `String.format()` pour obtenir un alignement des différentes valeurs du carré ;

```
Square order:5

 11   18   25    2    9  ->65
 10   12   19   21    3  ->65
  4    6   13   20   22  ->65
 23    5    7   14   16  ->65
 17   24    1    8   15  ->65
---- ---- ---- ---- ----
 65   65   65   65   65
```

3 Exercices plus complexes

Les exercices qui suivent demandent de bien comprendre la façon dont les tableaux multi-dimensionnels sont représentés en Java.

Pour ces méthodes, on ne vous demande pas d'écrire des tests unitaires. Il faut écrire ces méthodes dans la classe `Tableau2D`. Créez une classe `MainExoSup` avec une méthode principale pour tester les méthodes relatives aux exercices plus complexes.

3.1 Échange de deux lignes

Écrivez une méthode qui reçoit un tableau à deux dimensions ainsi que deux numéros de lignes et qui échange (efficacement) ces deux lignes.

Testez la méthode écrite dans la méthode principale de la classe `MainApp`.

3.2 Triangle de Pascal, exercice 10 du syllabus d'algorithmique

Il s'agit d'écrire une méthode qui crée et retourne un triangle de Pascal à n lignes, où n est le paramètre de la méthode.

Attention, le résultat ne sera pas un tableau rectangulaire mais un tableau triangulaire où chaque ligne est de la taille strictement adéquate.

Testez la méthode écrite dans la méthode principale de la classe `MainApp`.