Matrices – Laboratoire 5

Durée du travail :	**Du 01.11.2023 au 15.11.2023**
Auteurs :	**Calvin Graf** **Julien Holzer**
Langage de programmation :	**Java**
Enseignant :	**Pier Donini**
Cours :	**POO**
Lieu de travail :	**HEIG-VD, Yverdon-les-Bains**

Table des matières

1. Introduction

Ce laboratoire a pour objectif de concevoir une classe « Matrix » capable de représenter des matrices de taille variable contenant des entiers entre 0 et n-1. Avec n étant un entier spécifié par l'utilisateur et donc cela implique que les valeurs des matrices sont modulo n.

Il doit être possible de générer une matrice soit aléatoirement en spécifiant la taille et le modulo, soit en passant ses valeurs en paramètre. De plus, il faut pouvoir afficher le contenu d'une matrice.

La classe doit permettre d'effectuer des opérations, composante par composante entre deux matrices telles que l'addition, la soustraction et la multiplication. Toutes ces opérations devront être réalisées modulo n. Il est également demandé de définir des objets représentant l'opération à effectuer sur les éléments des matrices opérandes, afin de pouvoir éventuellement ajouter de nouvelles opérations ultérieurement.

Enfin, nous devrons implémenter la gestion d'exceptions en cas d'erreur, par exemple si l'on souhaite créer une matrice avec un nombre de lignes négatif ou encore si l'on essaie de créer une matrice avec des valeurs invalides.

2. Documentation

2.1 Description des fonctionnalités

Création de matrice :

- Possibilité de créer une matrice avec un contenu généré aléatoirement, en connaissant sa taille et son modulo.
- Possibilité de créer une matrice en passant des valeurs en paramètre. Dans ce cas, le modulo de la matrice est déduit en prenant la valeur la plus haute de la matrice + 1.

Affichage du contenu :

- La classe permet d'afficher les éléments d'une matrice.

Opération entre matrices :

- Addition, soustraction et multiplication composante par composante entre deux matrices (à condition que le modulo des deux matrices soit égal).
- Les valeurs de la matrice résultat sont modulo n.

Gestion des dimensions :

- Si les tailles des matrices ne correspondent pas lors d'une opération, elle s'effectue quand même en comblant les valeurs manquantes avec des 0. Le résultat sera une matrice avec les dimensions maximales des deux matrices.

Gestion des exceptions :

- Vérification de l'égalité entre les modulos de deux matrices dont on effectue une opération.
- Les éléments d'une matrice ne peuvent pas être négatifs. Il doit y avoir uniquement des entiers positifs.
- La taille d'une matrice est au minimum 1x1 (nombre de colonnes et nombre de lignes plus grandes ou égales à 1)
- Une matrice ne peut pas être vide (sans élément).
- Le modulo doit être un entier égal à 1 ou plus grand
- Les arguments du Main lors de l'exécution du programme doivent tous être des entiers positifs compris entre 1 et la valeur maximale d'un int.
- Vérification des overflow lors des opérations d'addition et de multiplication.

2.2 Phase de test

Test avec les valeurs des matrices fournies dans la donnée :

The modulo is : 5

Matrix 1 :

1 3 1 1
3 2 4 2
1 0 1 0

Matrix 2 :

1 4 2 3 2
0 1 0 4 2
0 0 2 0 2

Addition :

2 2 3 4 2
3 3 4 1 2
1 0 3 0 2

Substraction :

0 4 4 3 3
3 1 4 3 3
1 0 4 0 3

Multiplication :

1 2 2 3 0
0 2 0 3 0
0 0 2 0 0

Les résultats des matrices obtenues sont équivalents à ceux fournis dans la donnée. Les trois opérations se font correctement, tout en ayant le modulo appliqué aux valeurs de la matrice résultat.

Les matrices n'étant pas de tailles égales dans cet exemple, nous pouvons constater que les opérations prennent bien en compte la gestion des valeurs manquantes en utilisant la valeur 0. Nous remarquons également que la taille finale de la matrice résultat prend le nombre de colonnes et de lignes maximal.

Lors d'une soustraction, la gestion du cas d'un calcul de modulo négatif est bien appliquée. Par exemple m1[0][1] (= 3) moins m2[0][1] (= 4) qui donne le résultat 4 – 3 = -1, nous appliquons ensuite le modulo 5 ce qui donne 4 (et non -1).

Test de la création de matrice avec des valeurs aléatoires :

Commande sur le terminal qui représente la taille des lignes et colonnes de la matrice 1 (2x3) et 2 (3x2), puis du modulo (5) :

java Main.java 2 3 3 2 5

Modulo is : 5

Matrix 1 :

4 2 4
1 3 1

Matrix 2 :

0 3
2 0
0 3

Test de l'exception en cas de valeur en paramètre invalide :

Java Main.java 2 3 3 f 5

Exception in thread "main" java.lang.RuntimeException: Please enter valid numbers in the parameters.

java.lang.NumberFormatException: For input string: "f"

at Main.main(Main.java:20)

Test en cas d'opération avec deux matrices ayant un modulo différent :

Matrix 1 :

1 3 1 1
3 2 0 2
1 0 1 0

Matrix 2 :

1 4 2 3 2
0 1 0 4 2
0 0 2 0 2

Exception in thread "main" java.lang.RuntimeException: The two modulos must be equal.

at Matrix.calculateMatrix(Matrix.java:43)

at Matrix.add(Matrix.java:59)

at Main.main(Main.java:59)

Dans ce cas, les valeurs des matrices sont passés en paramètres directement depuis le code et non le terminal. Le modulo de la matrice 1 est égal à 4 (déduit à partir de la valeur la plus haute) tandis que celui de la matrice 2 est égal à 5, d'où le message d'erreur pour l'exception en cas de modulo non égal, qui se produit non pas à la création des matrices mais lorsqu'on réalise une opération entre elles.

Test d'une matrice avec une valeur négative (en voulant créer une matrice en donnant des valeurs en paramètre mais avec une valeur négative présente) :

Matrix 1 :

1 4 2 3 -2
0 1 0 -4 2
0 0 -3 0 2

Exception in thread "main" java.lang.RuntimeException: Matrix elements must be equal or greater than 0.

at Matrix.<init>(Matrix.java:18)

at Main.main(Main.java:48)

Test d'une matrice avec un modulo égal à 0 :

java Main.java 2 3 3 2 0

Exception in thread "main" java.lang.RuntimeException: The modulo must be equal or greater than 1.

at Matrix.<init>(Matrix.java:27)

at Main.main(Main.java:23)

Si le modulo est égal à 1, cela donne des matrices nulles (et donc les résultats des opérations le sont également). Mais cela est possible.

Test d'une matrice dont les colonnes ou lignes sont plus petites ou égales à 0 :

java Main.java 2 3 3 0 5

Exception in thread "main" java.lang.RuntimeException: The matrix must be at least 1x1 in size.

at Matrix.<init>(Matrix.java:26)

at Main.main(Main.java:24)

Test avec une valeur trop grande pour être contenu dans un int (2147483647 [valeur max d'un int] + 1 = 2147483648) :

java Main.java 2 3 3 2 2147483648

Exception in thread "main" java.lang.RuntimeException: Please enter valid numbers in the parameters.

java.lang.NumberFormatException: For input string: "2147483648"

at Main.main(Main.java:20)

Nous avons décidé de ne pas entrer un message spécifiques pour préciser que la valeur est trop grande car cela à peu d'intérêt. Mais c'est possible de le réaliser.

Test d'une opération qui donnerait comme résultat un chiffre plus grand que la valeur max d'un int :

Par exemple : 2147483647 + 1 => produit une erreur :

Exception in thread "main" java.lang.RuntimeException: The numbers in your matrices are too large to be added together and cause capacity overflow.

java.lang.ArithmeticException: integer overflow

Par exemple : 2147483647 * 2 => produit une erreur :

Exception in thread "main" java.lang.RuntimeException: The numbers in your matrices are too large to be multiplied and cause your capacity to be exceeded.

java.lang.ArithmeticException: integer overflow

Remarque : il n'y a pas de cas de underflow possible car il n'y a pas d'opération avec des nombres négatifs. Le chiffre le plus bas que nous pouvons obtenir dans notre cas est 0 – 2147483647 (valeur max d'un élément) ce qui donnerait -2147483647 et donc qui se trouve dans les champs limites d'un int (de -2147483648 à 2147483647). Bien sûr la valeur finale du résultat de cette opération ne sera pas négative et sera définie en fonction du modulo fourni.

Test en passant une matrice vide en paramètre :

Par exemple sous la forme {} ou encore { {}, {} } qui devrait aussi être considérée comme étant une matrice vide :

Exception in thread "main" java.lang.RuntimeException: The matrix must be at least 1x1 in size.

at Matrix.<init>(Matrix.java:11)

at Main.main(Main.java:55)

Test en passant une matrice plus grande que 500x500 en paramètre :

Exception in thread "main" java.lang.RuntimeException: Please enter a matrix smaller than 500x500.

at POO.Labo5.Main.main(Main.java:23)

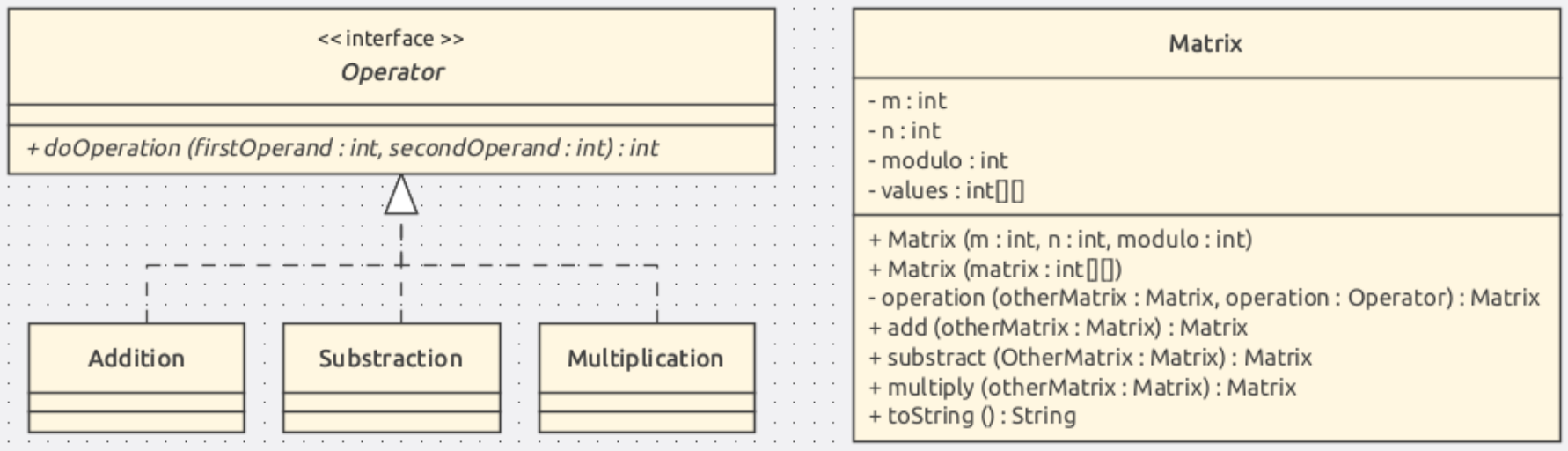
Nous avons décidé de limiter les matrices à 500x500 pour la simple raison qu'au-dessus, le programme commençait à prendre trop de temps à se réaliser et qu'à partir d'une certaine taille, le programme n'est plus capable de suivre et crash. Cependant, l'affichage n'est plus lisible très rapidement et donc l'utilité d'utiliser des matrices trop grandes est moindre. Fixer la limite à une valeur bien inférieure serait une bonne idée selon nous (peut-être aux alentours de 70).

3. Amélioration

Nous avons rempli tous les points qui étaient demandés dans le cahier des charges. Cependant, il y a 2 points qui selon nous, pourraient être améliorés. La gestion des grosses matrices comme expliquée ci-dessus. Mais également l'affichage des matrices.

Dans l'exemple, un simple espace est utilisé pour gérer l'affichage des éléments d'une matrice donc nous avons réutilisé la même méthode. Néanmoins, cela pose problème lorsqu'il faut afficher des nombres conséquents car les éléments ne sont plus alignés correctement. Une solution serait d'utiliser des tabulations.

4. UML


<< interface >>
Operator
+ doOperation (firstOperand : int, secondOperand : int) : int
Addition
Substraction
Multiplication
Matrix
- m : int
- n : int
- modulo : int
- values : int[][]
+ Matrix (m : int, n : int, modulo : int)
+ Matrix (matrix : int[][])
- operation (otherMatrix : Matrix, operation : Operator) : Matrix
+ add (otherMatrix : Matrix) : Matrix
+ substract (OtherMatrix : Matrix) : Matrix
+ multiply (otherMatrix : Matrix) : Matrix
+ toString () : String

5. Conclusion

La classe Matrix permet la génération, la manipulation et l'affichage de matrice de différentes tailles tout en ayant une gestion des erreurs en particulier dans les cas limites.

La conception des classes opérations a été implémenté en factorisant le plus possible afin de faciliter l'implémentation d'éventuelles nouvelles opérations dans le futur. Tous les points du cahier des charges ont été remplis et testés.