**Programmation Orientée Objet**

Rapport – Laboratoire 5 : Matrices

Gonçalo Carvalheiro Heleno

Sven Ferreira Silva

# **Table of Contents**

[**Table of Contents** 1](#_Toc149681797)

[**Modélisation** 2](#_Toc149681798)

[**Choix de conception** 2](#_Toc149681799)

[**Classes du programme** 3](#_Toc149681800)

[**Classe Main** 3](#_Toc149681801)

[**Classe Matrix** 3](#_Toc149681802)

[**Classes de test** 4](#_Toc149681803)

[**Classe MatrixTest** 4](#_Toc149681804)

[**Classe BinaryOperationTest** 4](#_Toc149681805)

# **Modélisation**

Celui-ci est dans le même dossier que ce rapport, en format PDF. Le fichier SLY est dans le dossier UML du dossier parent à celui-ci.

# **Choix de conception**

Le programme est divisé en plusieurs packages.

1. Un premier package *Matrix* contenant la classe Matrix,
2. un second package binaryOperation contenant les classes de calcul.

Celles-ci sont décrites plus en détail dans la section suivante.

# **Considérations techniques**

La valeur du module peut être positive comme négative. Cependant, les valeurs contenues dans les matrices ne peuvent être comprise que dans l’intervalle [0, module – 1], ainsi seule la valeur absolue du module fait foi.

Notre implémentation utilise la fonction *floorMod* dû au besoin d’un résultat positif ou nul pour les valeurs des matrices. La soustraction de matrices pouvant engendrer des résultats partiels négatifs (avant l’application du module), nous nous sommes inspirés d’exemples dans la documentation de la fonction pour obtenir un module positif en toute circonstance. Ces exemples sont disponibles à cette adresse : https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#floorMod-int-int

Les constructeurs des classes de classes sont privés, ceci afin de limiter l’instanciation de celles-ci en-dehors des classes dédiées à l’implémentation des diverses opérations. Ces classes ne sont instanciées que lors d’appels à des fonctions statiques.

# **Classes du programme**

## **Classe Main**

Contient le programme principal et les fonctions accessoires. Cette classe fait appel à la classe *Input* implémentant une fonction d’analyse et de traitement des arguments d’entrée. Une fois les données d’entrée vérifiées et converties en entier, le programme génère deux matrices grâce à ces valeurs numériques (ces valeurs étant les dimensions de la première matrice, celles de la deuxième, ainsi que leur module commun).

Enfin, après avoir affiché ces matrices, il effectue les opérations disponibles ainsi que leur résultat.

# ADD IMAGE

## **Classe Matrix**

La classe *Matrix* implémente les attributs privés suivants :

* le module,
* nombre de lignes,
* nombre de colonnes,
* les données numériques stockées en tant que tableau de tableaux d’entiers.

Les constructeurs implémentés sont :

* le constructeur par défaut, créant une matrice de côté 1 et de module unitaire,
* un constructeur prenant les dimensions attendues et le module, remplissant les données de valeurs aléatoires,
* un constructeur prenant un tableau de tableaux d’entiers ainsi que le module attendu,
* un constructeur de copie,
* un constructeur prenant une matrice et des dimensions différentes de celle-ci.

Les getters usuels sont naturellement implémentés, ainsi que les fonctions membres suivantes :

* *printMatrix()*, afin d’afficher le contenu d’une matrice en respectant ses dimensions à l’affichage,
* *addTo*, *subtractWith*, *multiplyBy*, qui effectuent les opérations arithmétiques homonymes.

## **Classe abstraite binaryOperation**

Cette classe abstraite implémente la fonction de calcul principal, *loopAndPerformOperation*, qui itère sur les valeurs de même coordonnées dans les deux matrices reçues en paramètre afin d’effectuer l’opération implémentée par l’une de ses sous-classes.

Egalement, cette classe implémente la fonction membre abstraite *operation*, prenant en paramètre les deux valeurs courantes des matrices, ainsi que leur module. Cette fonction membre, qui doit être implémentée dans les sous-classes de la classe binaryOperation, doit effectuer l’opération arithmétique attendue et retourner la valeur numérique résultante de cette opération.

A l’heure actuelle, trois sous-classes de binaryOperation sont disponibles: Addition, Subtraction, Multiplication.

# **Classes de test**

## **Classe MatrixTest**

Cette classe implémente les tests suivants :

* test du constructeur par défaut,
* test du constructeur aléatoire (construction par dimensions et module),
* test des exceptions du constructeur aléatoire,
* test du constructeur par tableau de tableaux d’entiers,
* test des exceptions du constructeur susmentionné,
* test du constructeur de copie,
* test des exceptions du constructeur de copie,
* test du constructeur de redimensionnement,
* test des exceptions du constructeur de redimensionnement

## **Classe BinaryOperationTest**

Cette classe implémente les tests suivants :

* test d’addition,
* test d’addition, cas limite : l’une des matrices est nulle,
* test des exceptions du mécanisme d’addition,
* test de soustraction,
* test de soustraction, cas limite: l’une des matrices est nulle,
* test des exceptions du mécanisme de soustraction,
* test de multiplication,
* test de multiplication, cas limite : l’une des matrices est nulle,
* test des exceptions du mécanisme de multiplication.