1 Diagramme de classe

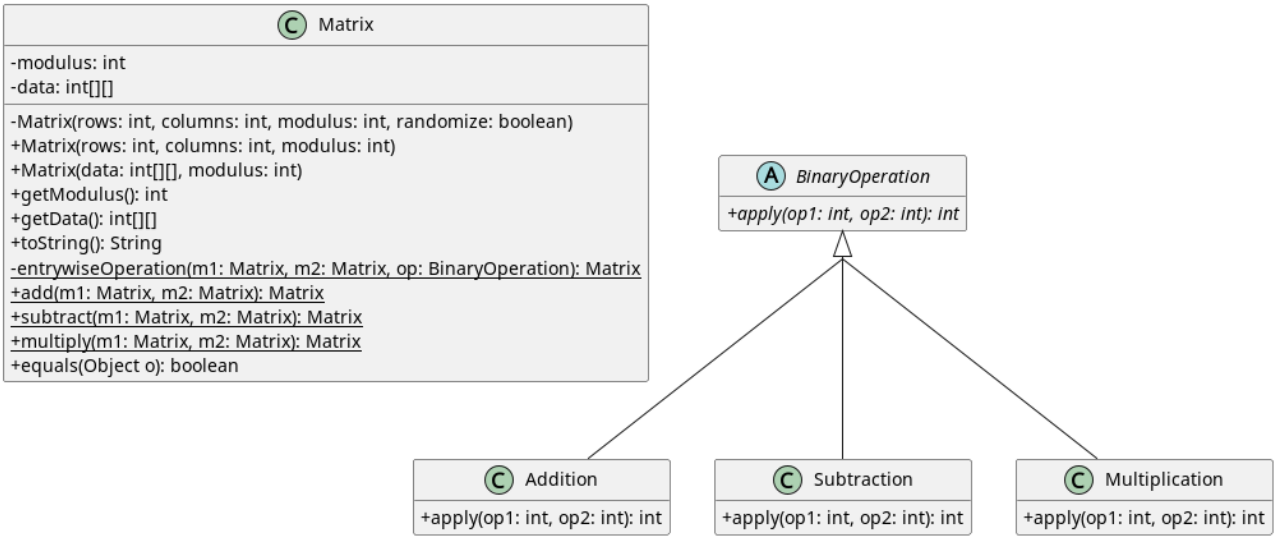

FIGURE 1 – Modélisation d'une matrice et des opérations associées.

2 Choix d'implémentation

Les hypothèses de travail et choix d'implémentation pris lors de ce laboratoire sont les suivantes :

1. Les modulos et données des matrices ne doivent pas être négatifs.
2. La taille d'une matrice doit être supérieure à 0.
3. Les classes ont été mises dans un package à part. Le programme de test est laissé dans le package par défaut.
4. Les opérateurs sont implémentés avec classe abstraite `BinaryOperator` qui a une unique méthode abstraite `apply(int, int)`. Cette méthode est implémentée par chaque sous-classe de `BinaryOperator`. Le choix d'une classe abstraite a été fait car il n'y a qu'une seule hiérarchie d'héritage. **Note** : puisque la superclasse ne fait que déclarer une méthode abstraite, il aurait été possible d'obtenir un comportement similaire en utilisant une interface.
5. La fonctionnalité principale de ce programme étant la manipulation de matrices, il a été décidé que l'implémentation des opérateurs ne doit pas être accessible par les utilisateurs. La visibilité de la classe `BinaryOperator` et ses ses sous-classes est donc `package`.
6. Les constructeurs sont factorisés pour utiliser un constructeur privé.
7. Lors de la création d'une matrice en passant les données et le modulo, tous les éléments doivent être inférieurs au modulo.
8. Les champs non-modifiables sont qualifiés `final`.
9. Des accesseurs pour les attributs `modulo` et `data` ont été ajoutés pour les éventuels utilisateurs de la librairie.
10. Puisque toutes les opérations s'effectuent élément par élément, l'application des opérations est factorisée.
11. La fonction `equals` est surchargée pour comparer les variables membres de la classe lors du test de l'égalité. Cette surcharge est particulièrement utile pour pouvoir utiliser `assertEquals` lors des tests.
12. Le fonctionnement du programme de test est validé avec des tests unitaires afin de ne pas devoir comparer les résultats manuellement. Les données utilisées sont celles de la consigne du laboratoire.
13. Des tests ont été réalisés pour valider que des fonctionnalités sont est implémentées (TDD).

3 Tests

Certains tests ont été réalisés pour valider qu'une fonction est implémentée (Test Driven Development).

Les tests de la classe `Matrix` sont les suivants :
- Tester que la méthode `equals` fonctionne correctement (l'utilisation d'`assertEquals` en dépend).
- Tester qu'une matrice ne puisse pas être créée avec de mauvais paramètres.
- Tester qu'une opération sur deux matrices de tailles différentes met les éléments hors de portée d'une des deux matrices à 0.
- Tester que les opérations d'addition, de soustraction et de multiplication sont implémentées et produisent le résultat escompté.
- Tester que deux matrices doivent avoir le même modulo pour être opérées.
- Tester qu'une matrice ne puisse pas être créée avec un élément égal ou supérieur au modulo.
- Tester qu'une matrice ne puisse pas être créée avec un élément inférieur à 0.
- Tester que la représentation en chaîne de caractères d'une matrice soit correcte.

Les tests des sous-classes de la classe `BinaryOperator` sont les suivants :
- Test de l'addition
- Test de la soustraction
- Test de la multiplication

Les tests de validation du programme de test sont les suivants :
- Tester l'affichage désiré
- Tester la prise en compte des arguments du programme
- Tester les opérations avec des valeurs identiques à l'exemple du laboratoire