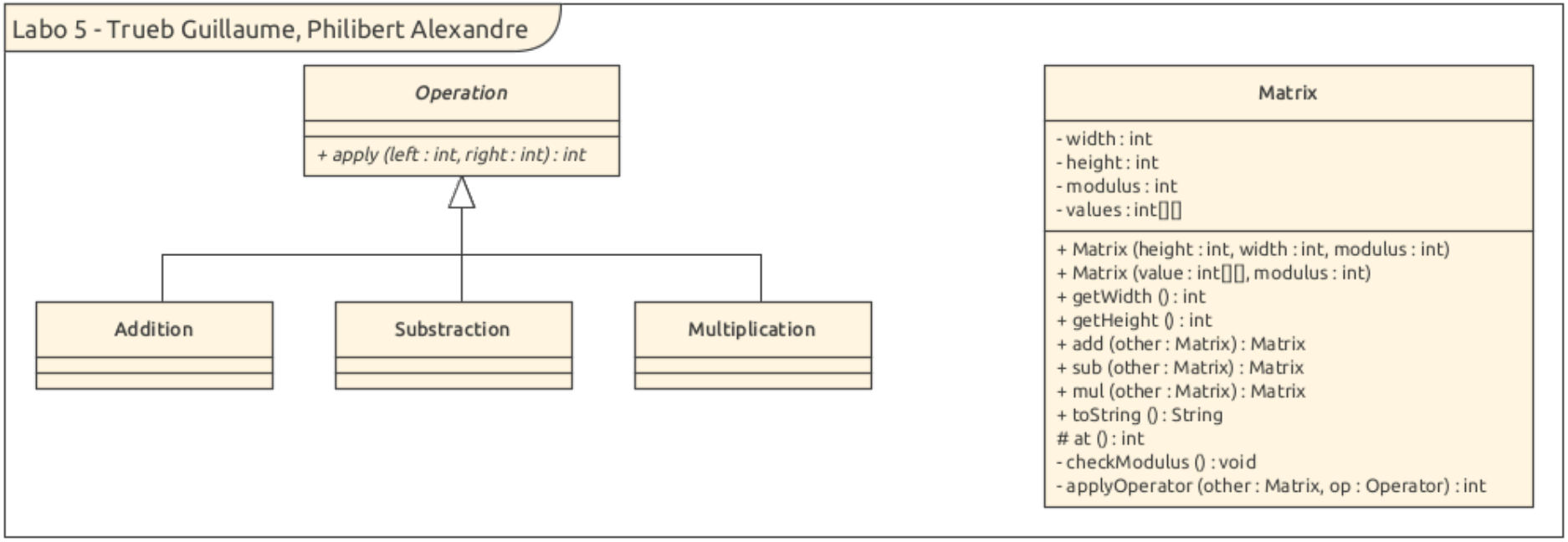
POO - Labo 5 Matrices

Alexandre Philibert **Guillaume Trüeb**

1 Diagramme


Labo 5 - Trueb Guillaume, Philibert Alexandre
Operation
+ apply (left : int, right : int) : int
Addition
Substraction
Multiplication
Matrix
- width : int
- height : int
- modulus : int
- values : int[][]
+ Matrix (height : int, width : int, modulus : int)
+ Matrix (value : int[][], modulus : int)
+ getWidth () : int
+ getHeight () : int
+ add (other : Matrix) : Matrix
+ sub (other : Matrix) : Matrix
+ mul (other : Matrix) : Matrix
+ toString () : String
# at () : int
- checkModulus () : void
- applyOperator (other : Matrix, op : Operator) : int

2 Choix et hypothèses de travail

- Lorsque le constructeur de Matrice par valeur est utilisé, aucun contrôle n'est effectué sur les valeurs. L'utilisateur est résponsable d'insérer des valeurs < modulus et > 0. Ce choix a été fait pour éviter un parcours avec une complexité O(n^2)

3 Rapport de tests

Nom	Description	Résultat
differentModulusAdd()	Essaie d'additionner deux matrices de modulo différent	OK
sameModulusAddDoesNotThrow()	Essaie d'additionner deux matrices de même modulo	OK
differentModulusSub()	Essaie de soustraire deux matrices de modulo différent	OK
sameModulusSubDoesNotThrow()	Essaie de soustraire deux matrices de même modulo	OK
differentModulusMul()	Essaie de multiplier duex matrices de modulo différent	OK
sameModulusMulDoesNotThrow()	Essaie de multiplier deux matrices de même modulo	OK
printMatrix()	Compare le résultat de la méthode toString()	OK
givenValuesCorrectlySetsWidth()	Créer une matrice à partir de valeurs et valide la largeur calculée	OK
givenValuesCorrectlySetsHeight()	Créer une matrice à partir de valeurs et valide la hauteur calculée	OK
valuesMatrixWithModulusZeroThrows()	Créer une matrice à partir de valeurs et lance une exception si modulo = 0	OK
randomMatrixWithModulusZeroThrows()	Créer une matrice aléatoire et lance une exception si modulo = 0	OK
randomMatrixHasCorrectWidth	Valide que la matrice créé à la bonne largeur	OK
randomMatrixHasCorrectHeight	Valide que la matrice créé à la bonne hauteur	OK

File - Main.java

```java
package org.example;

import org.example.matrix.Matrix;

public class Main {
    public static void main(String[] args) throws RuntimeException {
        if (args.length != 5) {
            throw new RuntimeException("5 arguments are required");
        }

        int n1, m1, n2, m2, modulus;

        try {
            n1 = Integer.parseInt(args[0]);
            m1 = Integer.parseInt(args[1]);
            n2 = Integer.parseInt(args[2]);
            m2 = Integer.parseInt(args[3]);
            modulus = Integer.parseInt(args[4]);
        } catch (NumberFormatException e) {
            throw new RuntimeException("Could not parse arguments as integers");
        }

        Matrix matrix1 = new Matrix(m1, n1, modulus);
        Matrix matrix2 = new Matrix(m2, n2, modulus);

        System.out.println("The modulus is " + modulus + "\n");

        System.out.println("one");
        System.out.println(matrix1);

        System.out.println("two");
        System.out.println(matrix2);

        System.out.println("one + two");
        System.out.println(matrix1.add(matrix2));

        System.out.println("one - two");
```

File - Main.java

```
38          System.out.println(matrix1.sub(matrix2));
39
40          System.out.println("one x two");
41          System.out.println(matrix1.mul(matrix2));
42
43      }
44  }
```

File - Matrix.java

```java
package org.example.matrix;

import org.example.matrix.operators.Addition;
import org.example.matrix.operators.Multiplication;
import org.example.matrix.operators.Operator;
import org.example.matrix.operators.Substraction;

public class Matrix {
    private final int width;

    private final int height;

    private final int[][] values;

    private final int modulus;

    public Matrix(int height, int width, int modulus) {
        checkModulus(modulus);

        this.width = width;
        this.height = height;
        this.modulus = modulus;

        values = new int[this.height][this.width];

        for (int i = 0; i < height; ++i) {
            for (int j = 0; j < width; ++j) {
                // Use the modulus to have the random number range between [0, modulus[
                values[i][j] = (int) (Math.random() * modulus);
            }
        }
    }

    /**
     * Create a new matrix filled with `values`, the user of this API is responsible for checking that the provided
     * values are valid in the context of this new matrix.
     *
```

```
38      * @param values The values that will be inserted in the resulting matrix
39      * @param modulus The modulus of the matrix
40      */
41     public Matrix(int[][] values, int modulus) {
42         checkModulus(modulus);
43 
44         this.height = values.length;
45         this.width = height == 0 ? 0 : values[0].length;
46         this.modulus = modulus;
47         this.values = values;
48     }
49 
50     public int getWidth() {
51         return width;
52     }
53 
54     public int getHeight() {
55         return height;
56     }
57 
58     /**
59      * Either return the value at (x,y) or 0 if out of bounds
60      *
61      * @param y The y position
62      * @param x The x position
63      * @return the value at (x,y) or 0 if out of bounds
64      */
65     protected int at(int y, int x) {
66         if (y > height - 1 || x > width - 1) {
67             return 0;
68         }
69 
70         return values[y][x];
71     }
72 
73     private void checkModulus(int modulus) {
74         if (modulus <= 0) {
```

```
 75              throw new RuntimeException();
 76          }
 77      }
 78
 79      /**
 80       * @throws RuntimeException When modulus are not equal
 81       */
 82      private Matrix applyOperator(Matrix other, Operator op) throws RuntimeException {
 83          if (modulus != other.modulus) {
 84              throw new RuntimeException("Matrices do not have the same modulus");
 85          }
 86
 87          int[][] values = new int[Math.max(height, other.height)][Math.max(width, other.width)];
 88
 89          for (int y = 0; y < Math.max(height, other.height); ++y) {
 90              for (int x = 0; x < Math.max(width, other.width); ++x) {
 91                  values[y][x] = Math.floorMod(op.apply(at(y, x), other.at(y, x)), modulus);
 92              }
 93          }
 94
 95          return new Matrix(values, modulus);
 96      }
 97
 98      public Matrix add(Matrix other) {
 99          return applyOperator(other, new Addition());
100      }
101
102      public Matrix sub(Matrix other) {
103          return applyOperator(other, new Substraction());
104      }
105
106      public Matrix mul(Matrix other) {
107          return applyOperator(other, new Multiplication());
108      }
109
110      @Override
111      public String toString() {
```

```
112         String s = "";
113
114         // We calculate the number of characters a single cell takes in order to keep things aligned
115         int whitespace = (int) Math.ceil(Math.log10(modulus)) + 1;
116
117         for (int[] i : values) {
118             for (int j : i) {
119                 s += String.format("%-" + whitespace + "d", j);
120             }
121
122             s += "\n";
123         }
124
125         return s;
126     }
127 }
128
```

File - Operator.java

```java
package org.example.matrix.operators;

public abstract class Operator {
    public abstract int apply(int left, int right);
}

```

File - Addition.java

```java
package org.example.matrix.operators;

public class Addition extends Operator {
    @Override
    public int apply(int left, int right) {
        return left + right;
    }
}

```

File - Substraction.java

```java
package org.example.matrix.operators;

public class Substraction extends Operator {
    @Override
    public int apply(int left, int right) {
        return left - right;
    }
}

```

File - Multiplication.java

```java
package org.example.matrix.operators;

public class Multiplication extends Operator {
    @Override
    public int apply(int left, int right) {
        return left * right;
    }
}

```

File - MatrixTest.java

```java
package org.example;

import org.example.matrix.Matrix;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class MatrixTest {
    @Test
    public void differentModulusAdd() {
        Matrix m1 = new Matrix(4, 4, 5);
        Matrix m2 = new Matrix(4, 4, 7);

        assertThrows(RuntimeException.class, () -> m1.add(m2));
    }

    @Test
    public void sameModulusAddDoesNotThrow() {
        Matrix m1 = new Matrix(4, 4, 7);
        Matrix m2 = new Matrix(4, 4, 7);

        assertDoesNotThrow(() -> m1.add(m2));
    }

    @Test
    public void differentModulusSub() {
        Matrix m1 = new Matrix(4, 4, 5);
        Matrix m2 = new Matrix(4, 4, 7);

        assertThrows(RuntimeException.class, () -> m1.sub(m2));
    }

    @Test
    public void sameModulusSubDoesNotThrow() {
        Matrix m1 = new Matrix(4, 4, 7);
        Matrix m2 = new Matrix(4, 4, 7);

        assertDoesNotThrow(() -> m1.sub(m2));
```

```
38      }
39
40      @Test
41      public void differentModulusMul() {
42          Matrix m1 = new Matrix(4, 4, 5);
43          Matrix m2 = new Matrix(4, 4, 7);
44
45          assertThrows(RuntimeException.class, () -> m1.mul(m2));
46      }
47
48      @Test
49      public void sameModulusMulDoesNotThrow() {
50          Matrix m1 = new Matrix(4, 4, 7);
51          Matrix m2 = new Matrix(4, 4, 7);
52
53          assertDoesNotThrow(() -> m1.mul(m2));
54      }
55
56      @Test
57      public void printMatrix() {
58          int[][] values = new int[][] {
59                  new int[] {1, 2, 3},
60                  new int[] {4, 5, 6},
61                  new int[] {7, 8, 9},
62          };
63
64          Matrix m = new Matrix(values, 9);
65
66          assertEquals("""
67          1 2 3\s
68          4 5 6\s
69          7 8 9\s
70          """, m.toString());
71      }
72
73      @Test
74      public void givenValuesCorrectlySetsWidth() {
```

```
 75          int[][] values = new int[][] {
 76                  new int[] {1, 2, 3},
 77                  new int[] {4, 5, 6},
 78                  new int[] {7, 8, 9},
 79                  new int[] {10, 11, 12},
 80          };
 81
 82          Matrix m = new Matrix(values, 8);
 83
 84          assertEquals(3, m.getWidth());
 85      }
 86
 87      @Test
 88      public void givenValuesCorrectlySetsHeight() {
 89          int[][] values = new int[][] {
 90                  new int[] {1, 2, 3},
 91                  new int[] {4, 5, 6},
 92                  new int[] {7, 8, 9},
 93                  new int[] {10, 11, 12},
 94          };
 95
 96          Matrix m = new Matrix(values, 8);
 97
 98          assertEquals(4, m.getHeight());
 99      }
100
101      @Test
102      public void randomMatrixWithModulusZeroThrows() {
103          assertThrows(RuntimeException.class, () -> new Matrix(4, 3, 0));
104      }
105
106      @Test
107      public void randomMatrixHasCorrectWidth() {
108          Matrix m = new Matrix(4, 3, 9);
109
110          assertEquals(3, m.getWidth());
111      }
```

```
112
113     @Test
114     public void randomMatrixHasCorrectHeight() {
115         Matrix m = new Matrix(4, 3, 9);
116
117         assertEquals(4, m.getHeight());
118     }
119
120     @Test
121     public void valuesMatrixWithModulusZeroThrows() {
122         int[][] values = new int[][] {
123                 new int[] {1, 2, 3},
124                 new int[] {4, 5, 6},
125                 new int[] {7, 8, 9},
126         };
127
128         assertThrows(RuntimeException.class, () -> new Matrix(values, 0));
129     }
130 }
131
```