

```
1 import VectorialObject.Matrice;
2
3 public class Main {
4     public static void main(String[] args) {
5         final int NBR_ARG = 5;
6
7         if (args.length != NBR_ARG) {
8             throw new RuntimeException(
9                 "nombre d'argument de args " +
10                 "non valide : " + args.length +
11                 ", attendu : " + NBR_ARG);
12         }
13
14         final int MODULO = Integer.parseInt(args[4]);
15
16         System.out.println("The modulus is " + MODULO + "\n");
17
18         System.out.println("one");
19         Matrice one = new Matrice(MODULO, Integer.parseInt(args[0]),
20             Integer.parseInt(args[1]));
21         System.out.println(one);
22
23         System.out.println("two");
24         Matrice two = new Matrice(MODULO, Integer.parseInt(args[2]),
25             Integer.parseInt(args[3]));
26         System.out.println(two);
27
28         System.out.println("one + two");
29         System.out.println(one.plus(two));
30
31         System.out.println("one - two");
32         System.out.println(one.minus(two));
33
34         System.out.println("one x two");
35         System.out.println(one.multiply(two));
36
37     }
38 }
```

```
1 package MatriceLabo5;
2
3 public class Mult extends Operator {
4     @Override
5     public String toString() {
6         return "x";
7     }
8
9     @Override
10    public int op(int i, int j) {
11        return i * j;
12    }
13 }
14
```

```
1 package MatriceLabo5;
2
3 public class Plus extends Operator {
4     @Override
5     public String toString() {
6         return "+";
7     }
8
9     @Override
10    public int op(int i, int j) {
11        return i + j;
12    }
13 }
14
```

```
1 package MatriceLabo5;
2
3 public class Minus extends Operator {
4     @Override
5     public String toString() {
6         return "-";
7     }
8
9     @Override
10    public int op(int i, int j) {
11        return i - j;
12    }
13 }
14
```

```
1 package MatriceLabo5;  
2  
3 abstract public class Operator {  
4     abstract public int op(int i, int j);  
5 }  
6
```

```

1 package VectorialObject;
2
3 import MatriceLabo5.Minus;
4 import MatriceLabo5.Mult;
5 import MatriceLabo5.Operator;
6 import MatriceLabo5.Plus;
7
8 public class Matrice {
9     private int[][] values;
10    private final int m, n, modulo;
11
12    public Matrice(int modulo, int m, int n) {
13        if (m <= 0 || n <= 0) {
14            throw new RuntimeException("in Matrice(int m, int n, " +
15                "int modulo) : m = " + m + ", n = " + n);
16        }
17        if (modulo < 0) {
18            throw new RuntimeException(
19                "negative modulo : not supported");
20        }
21        this.m = m;
22        this.n = n;
23        this.modulo = modulo;
24        values = new int[m][n];
25        for (int i = 0; i < m; i++) {
26            for (int j = 0; j < n; j++) {
27                values[i][j] = (int) Math.round(Math.random() *
28                    (Math.abs(modulo) - 1));
29            }
30        }
31    }
32
33    public Matrice(int modulo, int[][] values) {
34        this.values = values;
35        this.modulo = modulo;
36        m = values.length;
37        if (m == 0) {
38            throw new RuntimeException("in Matrice(int modulo, " +
39                "int[][] values) : m = " + m);
40        }
41        if (modulo < 0) {
42            throw new RuntimeException(
43                "negative modulo : not supported");
44        }
45        int maxN = 0;
46        //Gets the largest n
47        for (int[] value : values) {
48            if (value.length > maxN) {
49                maxN = value.length;

```

```

50     }
51 }
52 n = maxN;
53 if (n == 0) {
54     throw new RuntimeException("in Matrice(int modulo, " +
55         "int[][] values) : n = " + n);
56 }
57
58 //Setups the array and fills it with values floorModded
59 this.values = new int[m][this.n];
60 for (int i = 0; i < this.m; i++) {
61     for (int j = 0; j < this.n; j++) {
62         this.values[i][j] = j < values[i].length ?
63             Math.floorMod(values[i][j], modulo) : 0;
64     }
65 }
66 }
67
68 public String toString() {
69     StringBuilder s = new StringBuilder();
70     for (var line : values) {
71         for (var val : line) {
72             s.append(val);
73         }
74         s.append("\n");
75     }
76     return s.toString();
77 }
78
79 public int getM() {
80     return m;
81 }
82
83 public int getN() {
84     return n;
85 }
86
87 public int getModulo() {
88     return modulo;
89 }
90
91 public int at(int i, int j) {
92     return values[i][j];
93 }
94
95 private Matrice operate(Matrice a, Matrice b, Operator oper) {
96     if (a.getModulo() != b.getModulo()) {
97         throw new RuntimeException("Les modules ne sont pas " +
98             "identiques dans les deux matrices");

```

```

99     }
100
101     int m = Math.max(a.getM(), b.getM());
102     int n = Math.max(a.getN(), b.getN());
103
104     int[][] matrice = new int[m][n];
105
106     try {
107         for (int i = 0; i < m; ++i) {
108             for (int j = 0; j < n; ++j) {
109                 int tmpA = i < a.getM() && j < a.getN() ?
110                     a.at(i, j) : 0;
111                 int tmpB = i < b.getM() && j < b.getN() ?
112                     b.at(i, j) : 0;
113                 matrice[i][j] = Math.floorMod(
114                     oper.op(tmpA, tmpB), a.getModulo());
115             }
116         }
117     } catch (Exception e) {
118         throw new RuntimeException(e.getCause() +
119             " ::: Exception levée lors de l'utilisation " +
120             "de '" + oper.toString() + "'");
121     }
122
123     return new Matrice(a.getModulo(), matrice);
124 }
125
126 public Matrice plus(Matrice other) {
127     return operate(this, other, new Plus());
128 }
129
130 public Matrice minus(Matrice other) {
131     return operate(this, other, new Minus());
132 }
133
134 public Matrice multiply(Matrice other) {
135     return operate(this, other, new Mult());
136 }
137 }
138

```