```java
import VectorialObject.Matrice;

public class Main {
    public static void main(String[] args) {
        final int NBR_ARG = 5;

        if (args.length != NBR_ARG) {
            throw new RuntimeException(
                    "nombre d'argument de args " +
                            "non valide : " + args.length +
                            ", attendu : " + NBR_ARG);
        }

        final int MODULO = Integer.parseInt(args[4]);

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

        System.out.println("one");
        Matrice one = new Matrice(MODULO, Integer.parseInt(args[0]),
                Integer.parseInt(args[1]));
        System.out.println(one);

        System.out.println("two");
        Matrice two = new Matrice(MODULO, Integer.parseInt(args[2]),
                Integer.parseInt(args[3]));
        System.out.println(two);

        System.out.println("one + two");
        System.out.println(one.plus(two));

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

        System.out.println("one x two");
        System.out.println(one.multiply(two));

    }
}
```

```java
package Operator;

abstract public class Operator {
    abstract public int op(int i, int j);
}

```

```java
package Operator;

public class Plus extends Operator {
    @Override
    public String toString() {
        return "+";
    }

    @Override
    public int op(int i, int j) {
        return i + j;
    }
}

```

```java
package Operator;

public class Minus extends Operator {
    @Override
    public String toString() {
        return "-";
    }

    @Override
    public int op(int i, int j) {
        return i - j;
    }
}

```

```java
package Operator;

public class Mult extends Operator {
    @Override
    public String toString() {
        return "x";
    }

    @Override
    public int op(int i, int j) {
        return i * j;
    }
}

```

```java
package VectorialObject;

import Operator.Minus;
import Operator.Mult;
import Operator.Operator;
import Operator.Plus;

public class Matrice {
    private int[][] values;
    private final int m, n, modulo;

    public Matrice(int modulo, int m, int n) {
        if (m <= 0 || n <= 0) {
            throw new RuntimeException("in Matrice(int m, int n, " +
                    "int modulo) : m = " + m + ", n = " + n);
        }
        if (modulo < 0) {
            throw new RuntimeException(
                    "negative modulo : not supported");
        }
        this.m = m;
        this.n = n;
        this.modulo = modulo;
        values = new int[m][n];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                values[i][j] = (int) Math.round(Math.random() *
                        (Math.abs(modulo) - 1));
            }
        }
    }

    public Matrice(int modulo, int[][] values) {
        this.values = values;
        this.modulo = modulo;
        m = values.length;
        if (m == 0) {
            throw new RuntimeException("in Matrice(int modulo, " +
                    "int[][] values) : m = " + m);
        }
        if (modulo < 0) {
            throw new RuntimeException(
                    "negative modulo : not supported");
        }
        int maxN = 0;
        //Gets the largest n
        for (int[] value : values) {
            if (value.length > maxN) {
                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 modulos 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
```