```java
/*-----------------------------------------------------------------------------------
File name       : Program.java
Author(s)       : Kevin Farine, Timothée Van Hove
Created         : 3 nov. 2022
Description     : Program that execute +, - and * operations on matrices created
                  with random values depending on the given program input arguments.
                  The arguments must be: <R1> <C1> <R2> <C2> <Modulus> as integers
Remark(s)       : This program automatically closes after displaying the result.
JDK             : OpenJDK Runtime Environment Temurin-17.0.5+8 (build 17.0.5+8)
----------------------------------------------------------------------------------*/
package ch.heigvd.poo.labo5;

import ch.heigvd.poo.labo5.Matrix.Matrix;
import ch.heigvd.poo.labo5.operations.*;
import static ch.heigvd.poo.labo5.util.Util.StringArrayToIntArray;

public class Program {
    final static int ARG_NUMBER = 5;
    final static String M1 = "one", M2 = "two";

    public static void main(String[] args) {
        try {
            if (args.length != ARG_NUMBER) {
                throw new RuntimeException("Error : Expected 5 arguments, " +
                        args.length + " given.\n" +
                        "The arguments must be: <R1> <C1> <R2> <C2> <Modulus> \n" +
                        "<R1> : number of rows in the matrix 1\n" +
                        "<C1> : number of columns in the matrix 1\n");
            }
            int[] arguments = StringArrayToIntArray(args);
            Matrix m1 = new Matrix(arguments[0], arguments[1], arguments[4]);
            Matrix m2 = new Matrix(arguments[2], arguments[3], arguments[4]);

            //Display the matrices
            System.out.println(M1 + ": \n" + m1 + "\n" + M2 + ": \n" + m2);

            // Create an Operation array to iterate on it
            Operation[] operations = new Operation[]{
                    new Addition(), new Subtraction(), new Multiplication()};

            //Display each operation result
            for (Operation op : operations) {
                System.out.println(M1 + " " + op + " " + M2 + ":");
                System.out.println(m1.executeOperation(m2, op));
            }
        } catch (RuntimeException e) {
            e.printStackTrace();
            System.err.println("The program will exit");
        }
    }
}
```

```
/*-----------------------------------------------------------------------------
File name       : Matrix.java
Author(s)       : Kevin Farine, Timothée Van Hove
Created         : 3 nov. 2022
Description     : Class who will create matrix with an array of array of int
Remark(s)       :
JDK             : OpenJDK Runtime Environment Temurin-17.0.5+8 (build 17.0.5+8)
-----------------------------------------------------------------------------*/
package ch.heigvd.poo.labo5.Matrix;

import ch.heigvd.poo.labo5.operations.Operation;
import java.util.Random;
import ch.heigvd.poo.labo5.util.Util;

public class Matrix {

    private final int[][] internalValue;

    private int modulus;

    private static final Random random = new Random();

    /**
     * Constructor who take a 2D array of int and a modulus
     * @param values 2D array of int who is the base of the matrix
     * @param modulus The modulus who will be used for the matrix
     * @throws RuntimeException If modulus, nbRows, nbColumns <= 0, or if a matrix
     * value is >= modulus
     * @throws  NullPointerException if the given int[][], or one of its inner array
     * is null*/
    public Matrix(int[][] values, int modulus) throws RuntimeException {
        checkAndSetModulus(modulus);
        internalValue = new int[values.length][values[0].length];
        for (int i = 0; i < values.length; ++i) {
            if(values[i].length != values[0].length){
                throw new RuntimeException("The given 2d array must have the same " +
                        "number of elements j for each rows i");
            }
            for (int j = 0; j < values[0].length; ++j) {
                if (values[i][j] < 0 || values[i][j] >= this.modulus) {
                    throw new RuntimeException(
                            "The given values must be > 0 and < " + (modulus - 1));
                }
                internalValue[i][j] = values[i][j];
            }
        }
    }

    /**
     * Constructor who take the number of rows, columns and a modulus.
     * @param nbRows The number of rows of the matrix
     * @param nbColumns The number of columns of the matrix
     * @param modulus The modulus who will be used for the matrix
     * @throws RuntimeException if modulus, nbRows, nbColumns <= 0
     */
    public Matrix(int nbRows, int nbColumns, int modulus) throws RuntimeException {
        checkAndSetModulus(modulus);
        if(nbRows <= 0 || nbColumns <= 0){
            throw new RuntimeException("The number of rows / columns must be > 0");
        }
        internalValue = new int[nbRows][nbColumns];
        for (int i = 0; i < nbRows; ++i) {
            for (int j = 0; j < nbColumns; ++j) {
                internalValue[i][j] = random.nextInt(modulus);
            }
        }
    }
```

```java
    /**
     * Returns the formatted representation of the matrix with auto-align
     * @return  The formatted representation of the matrix as String
     */
    @Override
    public String toString() {
        StringBuilder result = new StringBuilder();
        int[] maxDigits = new int[internalValue[0].length];
        for (int i = 0; i < internalValue[0].length; ++i) {
            //Find the max value of each column
            int maxValue = 0;
            for (int[] row : internalValue) {
                maxValue = Math.max(maxValue, row[i]);
            }
            //Get the number of digits of the max number of each column
            maxDigits[i] = Util.nbDigits(maxValue);
        }
        for (int[] row : internalValue) {
            for (int j = 0; j < internalValue[0].length; ++j) {
                // Get the number of space characters to add after each value
                int nbSpace = maxDigits[j] + 1 - Util.nbDigits(row[j]);
                result.append(row[j]).append(" ".repeat(nbSpace));
            }
            result.append("\n");
        }
        return result.toString();
    }

    /**
     * Executes the given operation and return a matrix as result
     * @param rhs The other matrix used as right operand
     * @param op The operation used to calculate the new matrix
     * @return The operation result as a new matrix object
     * @throws RuntimeException if the modulus of the 2 matrices are different
     * @throws NullPointerException if the given Matrix or Operation is null
     */
    public Matrix executeOperation(Matrix rhs, Operation op) throws RuntimeException
    {
        if (this.modulus != rhs.modulus){
            throw new RuntimeException("The modulus of the 2 matrices must be " +
                    "identical");
        }
        int maxRows = Math.max(internalValue.length, rhs.internalValue.length);
        int maxColumns = Math.max(internalValue[0].length, rhs.internalValue[0].
        length);
        int[][] result = new int[maxRows][maxColumns];

        for (int i = 0; i < maxRows; ++i) {
            for (int j = 0; j < maxColumns; ++j) {
                int valM1 = checkBounds(i, j), valM2 = rhs.checkBounds(i, j);
                result[i][j] = Math.floorMod(op.execute(valM1, valM2), modulus);
            }
        }
        return new Matrix(result, modulus);
    }

    /**
     * Checks the value of the modulus and set it to the member attribute
     * @param modulus The modulus to check and set
     */
    private void checkAndSetModulus(int modulus){
        if (modulus <= 0)
            throw new RuntimeException("The modulus must be > 0");
        this.modulus = modulus;
    }
    /**
     * Controls if the indexes are within the bounds of the matrix
     * @param rowIndex Index of the row which is checked
     * @param columnIndex Index of the column which is checked
     * @return internalValue[rowIndex][columnIndex] if the indexes are in the bounds
     * of the matrix, else 0
     */
    private int checkBounds(int rowIndex, int columnIndex) {
        if(rowIndex <= internalValue.length - 1
                && columnIndex <= internalValue[0].length - 1){
            return internalValue[rowIndex][columnIndex];
        }
        return 0;
    }
}
```

```
/*-----------------------------------------------------------------------------------
File name       : Operation.java
Author(s)       : Kevin Farine, Timothée Van Hove
Created         : 3 nov. 2022
Description     : Abstract class who serve as base for the subclasses
JDK             : OpenJDK Runtime Environment Temurin-17.0.5+8 (build 17.0.5+8)
-----------------------------------------------------------------------------------*/
package ch.heigvd.poo.labo5.operations;

public abstract class Operation {
    protected String symbol = "";

    /**
     * Executes a subtraction of operand1 - operand2
     * @param operand1 The first operand
     * @param operand2 The second operand
     * @return The result of the operation
     */
    public abstract int execute(int operand1, int operand2);

    /**
     * Returns the symbol corresponding to the operation
     * @return the symbol of the operation
     */
    @Override
    public String toString(){
        return symbol;
    }
}
```

```
/*-----------------------------------------------------------------------------------
File name       : Addition.java
Author(s)       : Kevin Farine, Timothée Van Hove
Created         : 3 nov. 2022
Description     : Subclass of Operation who do the addition
Remark(s)       :
JDK             : OpenJDK Runtime Environment Temurin-17.0.5+8 (build 17.0.5+8)
-----------------------------------------------------------------------------------*/
package ch.heigvd.poo.labo5.operations;

public class Addition extends Operation {
    public Addition(){
        symbol = "+";
    }

    /**
     * Executes an addition of operand1 + operand2
     * @param operand1 The first operand
     * @param operand2 The second operand
     * @return The sum
     */
    @Override
    public int execute(int operand1, int operand2) {
        return operand1 + operand2;
    }
}
```

```
/*-----------------------------------------------------------------------------------
File name       : Subtraction.java
Author(s)       : Kevin Farine, Timothée Van Hove
Created         : 3 nov. 2022
Description     : Subclass of Operation who do the subtraction
Remark(s)       :
JDK             : OpenJDK Runtime Environment Temurin-17.0.5+8 (build 17.0.5+8)
-----------------------------------------------------------------------------------*/
package ch.heigvd.poo.labo5.operations;

public class Subtraction extends Operation {
    public Subtraction(){
        symbol = "-";
    }

    /**
     * Executes a subtraction of operand1 - operand2
     * @param operand1 The first operand
     * @param operand2 The second operand
     * @return The difference
     */
    @Override
    public int execute(int operand1, int operand2) {
        return operand1 - operand2;
    }
}
```

```
/*-----------------------------------------------------------------------------------
File name       : Multiplication.java
Author(s)       : Kevin Farine, Timothée Van Hove
Created         : 3 nov. 2022
Description     : Subclass of Operation who do the multiplication
Remark(s)       :
JDK             : OpenJDK Runtime Environment Temurin-17.0.5+8 (build 17.0.5+8)
-----------------------------------------------------------------------------------*/
package ch.heigvd.poo.labo5.operations;

public class Multiplication extends Operation {
    public Multiplication(){
        symbol = "*";
    }

    /**
     * Executes a multiplication of operand1 * operand2
     * @param operand1 The first operand
     * @param operand2 The second operand
     * @return The result
     */
    @Override
    public int execute(int operand1, int operand2) {
        return operand1 * operand2;
    }
}
```

```
/*-----------------------------------------------------------------------------------
File name       : Util.java
Author(s)       : Kevin Farine, Timothée Van Hove
Created         : 8 nov. 2022
Description     : Collection of utility static methods
Remark(s)       : This class can't be extended
JDK             : OpenJDK Runtime Environment Temurin-17.0.5+8 (build 17.0.5+8)
-----------------------------------------------------------------------------------*/
package ch.heigvd.poo.labo5.util;

public final class Util {
    /**
     * Finds the number of digits in a number
     *
     * @param number the number to evaluate
     * @return the number of digits of the given number
     */
    public static int nbDigits(int number) {
        int absNumber = Math.abs(number);
        if (absNumber == 0)
            return 1;
        return (int) Math.log10(absNumber) + 1;
    }

    /**
     * Converts a String[] into an int[]
     *
     * @param stringArray The String array to be converted
     * @return The converted int array
     * @throws NumberFormatException if the character cannot be parsed to int
     * @throws NullPointerException  if the given array is null
     */
    static public int[] StringArrayToIntArray(String[] stringArray)
            throws NumberFormatException, NullPointerException {
        int[] intArray = new int[stringArray.length];
        for (int i = 0; i < stringArray.length; ++i) {
            try {
                intArray[i] = Integer.parseInt(stringArray[i]);
            } catch (NumberFormatException e) {
                //Throw a more comprehensible exception for the user
                throw new NumberFormatException(
                        "Error cannot parse \"" + stringArray[i] + "\" to int");
            }
        }
        return intArray;
    }
}
```

```java
/*-----------------------------------------------------------------------------
File name       : MatrixTest.java
Author(s)       : Kevin Farine, Timothée Van Hove
Created         : 3 nov. 2022
Description     : Test program for the Matrix class
Remark(s)       : Use "mvn clean test" command to launch the test
JDK             : OpenJDK Runtime Environment Temurin-17.0.5+8 (build 17.0.5+8)
-----------------------------------------------------------------------------*/
package ch.heigvd.poo.labo5;

import ch.heigvd.poo.labo5.Matrix.Matrix;
import ch.heigvd.poo.labo5.operations.*;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;

public class MatrixTest {

    @Test
    public void testNominalCase() {
        Matrix m1 = new Matrix(new int[][]{
                                {1, 3, 1, 1},
                                {3, 2, 4, 2},
                                {1, 0, 1, 0}}, 5);
        Matrix m2 = new Matrix(new int[][]{
                                {1, 4, 2, 3, 2},
                                {0, 1, 0, 4, 2},
                                {0, 0, 2, 0, 2}}, 5);

        Matrix resultAdd = new Matrix(new int[][]{
                                        {2, 2, 3, 4, 2},
                                        {3, 3, 4, 1, 2},
                                        {1, 0, 3, 0, 2}}, 5);

        Matrix resultSub = new Matrix(new int[][]{
                                        {0, 4, 4, 3, 3},
                                        {3, 1, 4, 3, 3},
                                        {1, 0, 4, 0, 3}}, 5);

        Matrix resultMult = new Matrix(new int[][]{
                                        {1, 2, 2, 3, 0},
                                        {0, 2, 0, 3, 0},
                                        {0, 0, 2, 0, 0}}, 5);


        assertEquals(resultAdd.toString(),
                m1.executeOperation(m2, new Addition()).toString());

        assertEquals(resultSub.toString(),
                m1.executeOperation(m2, new Subtraction()).toString());

        assertEquals(resultMult.toString(),
                m1.executeOperation(m2, new Multiplication()).toString());
    }

    @Test
    public void testChainOperations(){
        Matrix m1 = new Matrix(new int[][]{
                                {1, 1, 1},
                                {2, 2, 2},
                                {3, 3, 3}}, 10);

        Matrix m2 = new Matrix(new int[][]{
                                {1, 1, 1, 1, 1},
                                {1, 1, 1, 1, 1},
                                {1, 1, 1, 1, 1}}, 10);
        Matrix result =
                new Matrix(new int[][]{
                                {3, 3, 3, 2, 2},
                                {4, 4, 4, 2, 2},
                                {5, 5, 5, 2, 2}}, 10);

        m1 = m1.executeOperation(m2, new Addition())
                .executeOperation(m2, new Addition());

        assertEquals(result.toString(),m1.toString());
    }

    @Test
    public void testAdditionWithDifferentModulus() {
        assertThrows(RuntimeException.class,
                () -> new Matrix(3, 4, 5).executeOperation(
                        new Matrix(3, 4, 6), new Addition()));
    }

    @Test
    public void testSubtractionWithDifferentModulus() {
        assertThrows(RuntimeException.class,
                () -> new Matrix(1, 4, 1).executeOperation(
                        new Matrix(10, 11, 6), new Subtraction()));
    }

    @Test
    public void testMultiplicationWithDifferentModulus() {
        assertThrows(RuntimeException.class,
                () -> new Matrix(100, 200, 1000).executeOperation(
                        new Matrix(66, 1, 1001), new Multiplication()));
    }

    @Test
    public void testConstructionWithValuesHigherThanModulus() {
        assertThrows(RuntimeException.class, () ->
            new Matrix(new int[][]{{6}, {0}, {3}}, 5));
    }

    @Test
    public void testConstructionWithIrregularMatrix(){
        assertThrows(RuntimeException.class, () ->
                new Matrix(new int[][]{{6, 3, 1}, {0}, {3, 1}}, 5));
        assertThrows(RuntimeException.class, () ->
                new Matrix(new int[][]{{6}, {0}, {3, 1}}, 5));
    }

    @Test
    public void testConstructionWithNullArray(){
        assertThrows(RuntimeException.class, () -> new Matrix(null, 5));
    }

    @Test
    public void testConstructionWithArrayContainingNullRow(){
        assertThrows(RuntimeException.class, () -> new Matrix(
                new int[][]{{0, 0, 2, 0, 0}, {0, 0, 2, 0, 0}, null}, 5));
        assertThrows(RuntimeException.class, () -> new Matrix(
                new int[][]{null, {0, 0, 2, 0, 0}, {0, 0, 2, 0, 0}}, 5));
    }

    @Test
    public void testConstructionWith0Row() {
        assertThrows(RuntimeException.class, () -> new Matrix(0, 2, 2));
    }

    @Test
    public void testConstructionWith0Column() {
        assertThrows(RuntimeException.class, () -> new Matrix(2, -1, 2));
    }

    @Test
    public void testConstructionWithNegativeAnd0Modulus() {
        assertThrows(RuntimeException.class, () -> new Matrix(3, 3, -1));
    }

    @Test
    public void testConstrictionWith0Modulus(){
        assertThrows(RuntimeException.class, () ->
                new Matrix(new int[][]{{6}, {0}, {3}}, 0));
    }
}
```

```
/*-----------------------------------------------------------------------------------
File name       : ProgramTest.java
Author(s)       : Kevin Farine, Timothée Van Hove
Created         : 3 nov. 2022
Description     : Tests for the main program
Remark(s)       : Use "mvn clean test" command to launch the test
JDK             : OpenJDK Runtime Environment Temurin-17.0.5+8 (build 17.0.5+8)
----------------------------------------------------------------------------------*/
package ch.heigvd.poo.labo5;

import org.junit.jupiter.api.Test;
import java.util.Arrays;

public class ProgramTest {
    @Test
    //Note: We cannot assert anything in those tests because the main function
    // does not return anything and all the exceptions are catch inside it;
    //We can only verify that the program runs as expected by looking at the console
    public void testTheMainProgram() {

        String[][] argList = {
                //The program should show correct matrices (nominal case)
                {"4", "5", "2", "7", "100"},

                // The program should display an exception about the illegal
                // character in the argument array
                {"4", "5", "2", "7", "test"},

                // The program should display an error exception the non-integer
                // value in the argument array
                {"4", "5", "2", "7", "0.42"},

                //The program should display an exception about the wrong number of
                // arguments given
                {"4", "5", "2", "7"}};

        for(String[] args : argList){
            System.out.println("Running the program with " + Arrays.toString(args) +
                    "as arguments");
            Program.main(args);
        }
    }
}
```