Folder src

7 printable files

(file list disabled)

src\Main.java

```java
package src;

import src.matrix.Matrix;
import src.matrix.operations.*;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public class Main {
    public static void main(String[] args) {
        // Arguments in args {N1,M1,N2,M2,mod}
        // NX: rows in matrix x
        // MX: columns in matrix x
        // mod: modulus for all numbers contained in the matrices
        if (args.length != 5) {
            throw new RuntimeException("Invalid number of arguments passed");
        }

        Matrix m1, m2;
        try {
            m1 = new Matrix(Integer.parseInt(args[0]), Integer.parseInt(args[1]), Integer.parseInt(args[4]));
            m2 = new Matrix(Integer.parseInt(args[2]), Integer.parseInt(args[3]), Integer.parseInt(args[4]));
        } catch (NumberFormatException e) {
            throw new RuntimeException("Arguments must be numbers");
        }

        System.out.println("The modulus is " + args[4]);
        System.out.println("one:");
        System.out.println(m1);

        System.out.println("two:");
        System.out.println(m2);

        Operation[] operations = new Operation[]{
                new Addition(),
                new Subtraction(),
                new Multiplication()
        };

        for (Operation operation : operations) {
            System.out.println("one " + operation + " two:");
            System.out.println(m1.calculate(m2, operation));
        }
    }
}
```

src\Test.java

```java
package src;

import src.matrix.Matrix;
import src.matrix.operations.*;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
```

```
 */
public class Test {
    public static void main(String[] args) {
        int modulus = 5;
        Matrix m1 = new Matrix(3, 4, modulus, new int[][]{
                {1, 3, 1, 1},
                {3, 2, 4, 2},
                {1, 0, 1, 0}
        });
        Matrix m2 = new Matrix(3, 5, modulus, new int[][]{
                {1, 4, 2, 3, 2},
                {0, 1, 0, 4, 2},
                {0, 0, 2, 0, 2},
        });

        System.out.println("The modulus is " + modulus);
        System.out.println("one:");
        System.out.println(m1);

        System.out.println("two:");
        System.out.println(m2);

        Operation[] operations = new Operation[]{
                new Addition(),
                new Subtraction(),
                new Multiplication()
        };

        for (Operation operation : operations) {
            System.out.println("one " + operation + " two:");
            System.out.println(m1.calculate(m2, operation));
        }
    }
}
```

src\matrix\Matrix.java

```
package src.matrix;

import src.matrix.operations.Operation;

import java.util.Random;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public class Matrix {
    private final int height;
    private final int width;
    private final int modulus;
    private final int[][] values;

    Random random = new Random();

    // Default constructor
    private Matrix() {
        height = 1;
        width = 1;
        modulus = 1;
        values = new int[1][1];
    }

    // Constructor with random numbers
    public Matrix(int height, int width, int modulus) {
        checkConstructorParams(height, width, modulus, null);

        this.height = height;
```

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

        for (int i = 0; i < height; ++i) {
            for (int j = 0; j < width; ++j) {
                values[i][j] = random.nextInt(modulus);
            }
        }
    }

    // Constructor with chosen numbers
    public Matrix(int height, int width, int modulus, int[][] values) {
        checkConstructorParams(height, width, modulus, values);

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

        // We must copy the values array in order to not copy the reference
        for (int i = 0; i < height; ++i) {
            System.arraycopy(values[i], 0, this.values[i], 0, width);
        }
    }

    // Checks that we can create a matrix with the given parameters
    private void checkConstructorParams(int height, int width, int modulus, int[][] values) {
        // Matrix dimensions and modulus must be greater than 0
        if (height < 1 || width < 1 || modulus < 1) {
            throw new RuntimeException("Invalid parameters");
        }

        // Passed values array must be the same size as the matrix and all values must be less than the modulus
        if (values != null) {
            if (values.length != height || values[0].length != width) {
                throw new RuntimeException("Invalid matrix dimensions");
            } else {
                for (int i = 0; i < height; ++i) {
                    for (int j = 0; j < width; ++j) {
                        if (values[i][j] >= modulus) {
                            throw new RuntimeException("Invalid matrix values");
                        }
                    }
                }
            }
        }
    }

    // So that we can simply "print" the matrix
    public String toString() {
        StringBuilder result = new StringBuilder();

        for (int i = 0; i < this.height; ++i) {
            for (int j = 0; j < this.width; ++j) {
                result.append(values[i][j]);

                if (j != this.width - 1) {
                    result.append(" ");
                }
            }

            result.append("\n");
        }

        return result.toString();
    }

    // Calculates the result of the operation between this matrix and another matrix
    public Matrix calculate(Matrix other, Operation operation) {
        if (this.modulus != other.modulus) {
```

```
            throw new RuntimeException("Matrices must have the same modulus");
        }

        int maxHeight = Math.max(this.height, other.height);
        int maxWidth = Math.max(this.width, other.width);
        int[][] newValues = new int[maxHeight][maxWidth];

        for (int i = 0; i < maxHeight; ++i) {
            for (int j = 0; j < maxWidth; ++j) {
                int op1 = 0, op2 = 0;

                // If the index is out of bounds, we use 0 as the operand
                if (i < this.height && j < this.width) {
                    op1 = this.values[i][j];
                }

                if (i < other.height && j < other.width) {
                    op2 = other.values[i][j];
                }

                int result = operation.calculate(op1, op2);
                newValues[i][j] = Math.floorMod(result, this.modulus);
            }
        }

        // Returns the new matrix with the same modulus
        return new Matrix(maxHeight, maxWidth, this.modulus, newValues);
    }
}
```

src\matrix\operations\Addition.java

```
package src.matrix.operations;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public class Addition extends Operation {
    public int calculate(int op1, int op2) {
        return op1 + op2;
    }

    public String toString() {
        return "+";
    }
}
```

src\matrix\operations\Multiplication.java

```
package src.matrix.operations;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public class Multiplication extends Operation {
    public int calculate(int op1, int op2) {
        return op1 * op2;
    }

    public String toString() {
        return "x";
    }
}
```

src\matrix\operations\Operation.java

```java
package src.matrix.operations;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public abstract class Operation {
    public abstract int calculate(int op1, int op2);

    public abstract String toString();
}
```

src\matrix\operations\Subtraction.java

```java
package src.matrix.operations;

/**
 * @author Calum Quinn
 * @author Dylan Ramos
 */
public class Subtraction extends Operation {
    public int calculate(int op1, int op2) {
        return op1 - op2;
    }

    public String toString() {
        return "-";
    }
}
```