Qatar University

Operating System CMPS 405

PROJECT #2

Eng. Ahmed Hussain

Amira Abdalla 201802220

Nirvana Aladal 201802829

Maryam AlHumaidi 201701341

Tasks Distribution

Nirvana Aladal	- Perform the Convolution operation on a matrix (Convolution.java) - Get the average of two matrices and find the maximum average (Average.java) - Find a number in the matrix and replace it with 0 or 1 (Replace.java) - Word File - (Client.java) added methods
Maryam AlHumaidi	- Matrix multiplication - Find the maximum and minimum elements in a matrix (question2.java) (main.java)
Amira Abdalla	- Sum numbers below the diagonal line and above the diagonal line, then compare the results (SumAboveBelowDiagonal.java) - Sort the elements of a matrix (SortMatrixServer.java) - (Client.java) - (Server.java)

Table of Contents

Average.java

```
package proj_2_2;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Formatter;
import java.util.Scanner;

public class Average extends Thread {
	private Socket client;
	private Scanner fromNet = null;
	private Formatter toNet = null;

	public Average(Socket client)
	{
		this.client = client;
		//this.start();
	}




	   public double cellNeighborsAverage(int[][] matrix,int row, int col)
	   {
```

```
            // Ignore center cell
            int sum = matrix[row - 1][col - 1] + matrix[row - 1][col]
                  + matrix[row - 1][col + 1] + matrix[row][col - 1]
                  + matrix[row][col + 1] + matrix[row + 1][col - 1]
                  + matrix[row + 1][col] + matrix[row + 1][col + 1];
            return sum / 8;
          }

      public void run() {
            try {
                  int row1,col1,row2, col2,  Matrix1[][],Matrix2[][];

                  fromNet = new Scanner(client.getInputStream());
                  toNet = new Formatter(client.getOutputStream());
                  DataInputStream dis = new
DataInputStream(client.getInputStream());
                  DataOutputStream dos = new
DataOutputStream(client.getOutputStream());


                  //System.out.println(" Enter a number of rows for kernel matrix [2 or
3]\n:");

                  dos.writeUTF(" Enter a number of rows for the first matrix \n:");
                  int row = dis.readInt();
```

```
                dos.writeUTF(" Enter same number for columns for the first  matrix
\n:");
                int col = dis.readInt();


                int inmat[][] = new int[row][col];
                for (int i = 0; i < row; i++) {
                        for (int j = 0; j < col; j++) {
                                inmat[i][j] = dis.readInt();
                        }
                }


                dos.writeUTF("  Enter a number of rows for second  matrix  \n:");
                int krow = dis.readInt();


                dos.writeUTF("  Enter same number for columns for second matrix
\n:");
                int kcol = dis.readInt();



                int kmat[][] = new int[krow][kcol];
                for (int i = 0; i < krow; i++) {
                        for (int j = 0; j < kcol; j++) {
                                kmat[i][j] = dis.readInt();
```

```
                }
            }



        double[][] computedMatrix = new double[row][col];

        for (int i = 1; i < row - 1; i++)
           {
              for (int j = 1; j < col - 1; j++)
              {
                 computedMatrix[i][j] = cellNeighborsAverage(inmat,i, j);
              }
           }



double[][] computedMatrix2 = new double[krow][kcol];

              for (int i = 1; i < row - 1; i++)
              {
                 for (int j = 1; j < col - 1; j++)
                 {
                    computedMatrix2[i][j] = cellNeighborsAverage(kmat,i, j);
                 //
                 }
              }
```

```
            dos.writeInt(computedMatrix.length);
                dos.writeInt(computedMatrix[0].length);
                    for (int i = 0; i < computedMatrix.length; i++) {
                    for (int j = 0; j < computedMatrix[0].length; j++) {
dos.writeUTF(computedMatrix[i][j] + " ");
                        }
                        dos.writeUTF("\n");
                        }


                    dos.writeInt(computedMatrix2.length);
                dos.writeInt(computedMatrix2[0].length);
                    for (int i = 0; i < computedMatrix2.length; i++) {
                    for (int j = 0; j < computedMatrix2[0].length; j++) {
dos.writeUTF(computedMatrix2[i][j] + " ");
                        }
                        dos.writeUTF("\n");
                        }



                    double maxValue = 0;
                for (int j = 0; j < computedMatrix.length; j++) {
                   for (int i = 0; i < computedMatrix[j].length; i++) {
                      if (computedMatrix[j][i] > maxValue) {
                         maxValue = computedMatrix[j][i];
```

```
                }
            }
        }

        for (int j = 0; j < computedMatrix2.length; j++) {
            for (int i = 0; i < computedMatrix2[j].length; i++) {
                if (computedMatrix2[j][i] > maxValue) {
                    maxValue = computedMatrix2[j][i];
                }
            }
        }


                dos.writeUTF("Max Average is "+maxValue);




} catch (IOException ioe) {
        System.err.println(ioe);
```

```
            } finally {
                try {
                    if (client != null)
                        client.close();
                    if (fromNet != null)
                        fromNet.close();
                    if (toNet != null)
                        toNet.close();
                } catch (Exception e) {
                    System.err.println(e);
                }
            }
        }


}
```

Client.java

```
package proj_2_2;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Formatter;
```

```
import java.util.Scanner;

public class Client {
	private Socket server;
	private Scanner fromUser =new Scanner(System.in);
	private Scanner fromNet = null;
	private Formatter toNet = null;

	public Client() {

		while(true) {
			System.out.println();
			menu();
			System.out.println();
			int select = fromUser.nextInt();

			if (select ==0){
				return;
			}

			else if (select == 1){
				Multiplication();
			}

			else if (select == 2){
				Convolution();
			}
```

```
                else if(select == 3) {
                        SumAboveBelow();
                }

                else if(select == 4) {
                        MaxMin();
                }

                else if(select == 5) {
                        Sort();
                }

                else if(select == 6) {
                        Average();
                }

                else if(select == 7) {
                        Replace();
                }
                System.out.println();

        }
}

private void MaxMin() {
        // TODO Auto-generated method stub
```

```
        try {
                int row,col;
                server = new Socket("localhost",2000);
                fromNet = new Scanner(server.getInputStream());
                toNet = new Formatter(server.getOutputStream());
                DataOutputStream dos = new
DataOutputStream(server.getOutputStream());
                DataInputStream dis = new
DataInputStream(server.getInputStream());

                System.out.println(dis.readUTF());
                row = fromUser.nextInt();
                dos.writeInt(row);

                System.out.println(dis.readUTF());
                col = fromUser.nextInt();
                dos.writeInt(col);

                System.out.println("enter the matrix elements: ");

                 int[][] amat = new int[row][col];


                 for (int i = 0; i < row; i++) {
                                for (int j = 0; j < col; j++) {
                                        amat[i][j] = fromUser.nextInt();
                                        dos.writeInt(amat[i][j]);
```

```
                    }
                }

                System.out.println(dis.readUTF());
                System.out.println(dis.readUTF());




        }catch(Exception e) {
                e.printStackTrace();
        }finally {
                server = null;
        }

}

private void Multiplication() {
        // TODO Auto-generated method stub
        try {
        int brow,bcol,row,col;
        server = new Socket("localhost",3000);
        fromNet = new Scanner(server.getInputStream());
        toNet = new Formatter(server.getOutputStream());
```

```
            DataOutputStream dos = new
DataOutputStream(server.getOutputStream());
            DataInputStream dis = new DataInputStream(server.getInputStream());



            System.out.println(dis.readUTF());
            row = fromUser.nextInt();
            dos.writeInt(row);

            System.out.println(dis.readUTF());
            col = fromUser.nextInt();
            dos.writeInt(col);

            System.out.println("enter the matrix elements: ");

             int[][] amat = new int[row][col];



             for (int i = 0; i < row; i++) {
                        for (int j = 0; j < col; j++) {
                                amat[i][j] = fromUser.nextInt();
                                dos.writeInt(amat[i][j]);
                        }
                }


             System.out.println(dis.readUTF());
```

```
brow = fromUser.nextInt();
dos.writeInt(brow);

System.out.println(dis.readUTF());

bcol = fromUser.nextInt();
dos.writeInt(bcol);


System.out.println("enter the matrix elements: ");

int[][] bmat = new int[brow][bcol];


 for (int i = 0; i < brow; i++) {
                for (int j = 0; j < bcol; j++) {
                        bmat[i][j] = fromUser.nextInt();
                        dos.writeInt(bmat[i][j]);
                }
        }


 System.out.println("Multiplication of 2 Matrix: ");

        int rrow = dis.readInt();
        int rcol = dis.readInt();
```

```
                    for (int i = 0; i < rrow; i++) {
                        for (int j = 0; j < rcol; j++) {
                            System.out.print(dis.readUTF());
                       }
                        System.out.print(dis.readUTF());
                    }
                }
            }catch(Exception e) {
                e.printStackTrace();
            }finally {
                server = null;
            }

}
```

```
	public void menu() {
		System.out.println("Choose from the menu:\n"
				+ "0. for exit\n"
				+ "1. matrix multiplication \n"
				+ "2. matrix convolution \n"
				+ "3. Sum numbers below the diagonal line and above the
diagonal line\n"
				+ "4. Find the maximum and minimum elements in a
matrix\n"
				+ "5. Sort the elements of a matrix\n"
				+ "6. Get the average of two matrices and find the maximum
average\n"
				+ "7. Find a number in the matrix and replace it with 0 or 1");
	}



	public int EnterMatrixAndSendToServer() {
		System.out.println("enter the row and colum: ");
		int numRow = fromUser.nextInt();
		int numCol = fromUser.nextInt();
		while(numRow != numCol) {
			System.out.println("enter the row and colum again to be smilar n x
n: ");
			numRow = fromUser.nextInt();
			numCol = fromUser.nextInt();
```

```
        }

        System.out.println("enter the matrix: ");
        int mat[][] = new int[numRow][numCol];

        for (int i = 0; i < numRow; i++) {
                for (int j = 0; j < numCol; j++) {
                        mat[i][j] = fromUser.nextInt();
                }
        }

        toNet.format("%d\n", numRow);
        toNet.format("%d\n", numCol);
        for (int i = 0; i <numRow; i++)
                for (int j = 0; j < numCol; j++)
                        toNet.format("%d\n",mat[i][j]).flush();
        return numRow;
}


public void Sort() {
        try {
                server = new Socket("localhost",4000);
                fromNet = new Scanner(server.getInputStream());
                toNet = new Formatter(server.getOutputStream());

                int numMat = EnterMatrixAndSendToServer();
```

```
                System.out.println("The sort is ascending or descending? 1 for
ascending, 2 for descending");
                int choosen = fromUser.nextInt();
                while(choosen!=1 && choosen !=2) {
                        System.out.println("\nplease enter 1 for ascending, 2 for
descending ");
                        choosen = fromUser.nextInt();
                }
                toNet.format("%d\n", choosen).flush();

                int [][]matResult = new int[numMat][numMat];
                for (int i = 0; i < numMat; i++)
                {
                        for (int j = 0; j < numMat; j++)
                        {
                                matResult[i][j] = fromNet.nextInt();
                                System.out.print(matResult[i][j] + " ");
                        }
                        System.out.println();
                }

        }catch(Exception e) {
                e.printStackTrace();
        }finally {
                server = null;
        }
```

```
	}

	public void SumAboveBelow() {
		try {
			server = new Socket("localhost",5000);
			fromNet = new Scanner(server.getInputStream());
			toNet = new Formatter(server.getOutputStream());

			EnterMatrixAndSendToServer();
			String sumAbove = fromNet.next();
			String sumBelow = fromNet.next();
			if(Integer.parseInt(sumAbove)> Integer.parseInt(sumBelow)) {
				System.out.println("Sum Above is grater than Below:
"+sumAbove+" > "+sumBelow+"\n");
			}else
				System.out.println("Sum Below is grater than Above:
"+sumBelow+" > "+sumAbove+"\n");
			System.out.printf("The sum Above: %s\nThe sum Below: %s",
sumAbove,sumBelow);

		}catch(Exception e) {
			e.printStackTrace();
		}finally {
			server = null;
		}
	}
```

```
	public void Average() {
		try {
			//server = new Socket("localhost",8000);


			int krow,kcol,row,col;
			server = new Socket("localhost",8000);
			fromNet = new Scanner(server.getInputStream());
			toNet = new Formatter(server.getOutputStream());
			DataOutputStream dos = new
DataOutputStream(server.getOutputStream());
			DataInputStream dis = new
DataInputStream(server.getInputStream());


			System.out.println(dis.readUTF());
			row = fromUser.nextInt();
			dos.writeInt(row);

			System.out.println(dis.readUTF());

			col = fromUser.nextInt();
			dos.writeInt(col);
```

```
System.out.println("enter the matrix elements: ");

 int[][] kernel = new int[row][col];


 for (int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                        kernel[i][j] = fromUser.nextInt();
                        dos.writeInt(kernel[i][j]);
                }
        }



   System.out.println(dis.readUTF());
        krow = fromUser.nextInt();
        dos.writeInt(krow);

        System.out.println(dis.readUTF());

        kcol = fromUser.nextInt();
        dos.writeInt(kcol);

        System.out.println("enter the matrix elements: ");
```

```
int[][] input = new int[krow][kcol];


 for (int i = 0; i < krow; i++) {
                for (int j = 0; j < kcol; j++) {
                        input[i][j] = fromUser.nextInt();
                        dos.writeInt(input[i][j]);
                }
        }


 System.out.println("Average First Matrix : ");

        int rrow = dis.readInt();
        int rcol = dis.readInt();



         for (int i = 0; i < rrow; i++) {
                 for (int j = 0; j < rcol; j++) {
                         System.out.print(dis.readUTF());
            }
                 System.out.println(dis.readUTF());
         }
```

```
                    System.out.println("Average scond Matrix : ");

                        rrow = dis.readInt();
                        rcol = dis.readInt();



                        for (int i = 0; i < rrow; i++) {
                                for (int j = 0; j < rcol; j++) {
                                        System.out.print(dis.readUTF());
                           }
                                System.out.println(dis.readUTF());
                        }



                        System.out.println(dis.readUTF());



                // System.out.printf("Maximum average of the 2 matrices: ", avg1 ,
avg2);

        }catch(Exception e) {
                e.printStackTrace();
        }finally {
```

```
                    server = null;
                }
        }

        public void Replace() {
                try {

                    int row,col;
                    server = new Socket("localhost",7000);
                    fromNet = new Scanner(server.getInputStream());
                    toNet = new Formatter(server.getOutputStream());
                    DataOutputStream dos = new
DataOutputStream(server.getOutputStream());
                    DataInputStream dis = new
DataInputStream(server.getInputStream());

                    System.out.println(dis.readUTF());
                    row = fromUser.nextInt();
                    dos.writeInt(row);

                    System.out.println(dis.readUTF());
                    col = fromUser.nextInt();
                    dos.writeInt(col);

                    System.out.println("enter the matrix elements: ");

                     int[][] amat = new int[row][col];
```

```
        System.out.println(dis.readUTF());

for (int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                        amat[i][j] = fromUser.nextInt();
                        dos.writeInt(amat[i][j]);
                }
        }

System.out.println(dis.readUTF());
        int one = fromUser.nextInt();
        dos.writeInt(one);

        System.out.println(dis.readUTF());
        int tow = fromUser.nextInt();
        dos.writeInt(tow);




         for (int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                        System.out.print(dis.readUTF());
            }
                System.out.println(dis.readUTF());
        }
```

```
            }catch(Exception e) {
                    e.printStackTrace();
            }finally {
                    server = null;
            }
        }

        public void Convolution() {
            try {
                    int krow,kcol,row,col;
                    server = new Socket("localhost",6000);
                    fromNet = new Scanner(server.getInputStream());
                    toNet = new Formatter(server.getOutputStream());
                    DataOutputStream dos = new
DataOutputStream(server.getOutputStream());
```

```
                DataInputStream dis = new
DataInputStream(server.getInputStream());


                System.out.println(dis.readUTF());
                row = fromUser.nextInt();
                dos.writeInt(row);

                System.out.println(dis.readUTF());

                col = fromUser.nextInt();
                dos.writeInt(col);


                System.out.println("enter the matrix elements: ");


                 int[][] kernel = new int[row][col];


                 for (int i = 0; i < row; i++) {
                                for (int j = 0; j < col; j++) {
                                        kernel[i][j] = fromUser.nextInt();
                                        dos.writeInt(kernel[i][j]);
                                }
                        }
```

```
System.out.println(dis.readUTF());
        krow = fromUser.nextInt();
        dos.writeInt(krow);

        System.out.println(dis.readUTF());

        kcol = fromUser.nextInt();
        dos.writeInt(kcol);

        System.out.println("enter the matrix elements: ");

        int[][] input = new int[krow][kcol];


         for (int i = 0; i < krow; i++) {
                        for (int j = 0; j < kcol; j++) {
                                input[i][j] = fromUser.nextInt();
                                dos.writeInt(input[i][j]);
                        }
                }


                System.out.println("Convulation Matrix: ");
```

```
                int rrow = dis.readInt();
                int rcol = dis.readInt();



                for (int i = 0; i < rrow; i++) {
                        for (int j = 0; j < rcol; j++) {
                                System.out.print(dis.readUTF());
                   }
                        System.out.println(dis.readUTF());
                }
```

```
}catch(Exception e) {
        e.printStackTrace();
}finally {
        server = null;
```

```
                }
        }


/*      public void Multiplication(){

        }

        public void MaxMin(){

        }
*/

        public static void main(String[] args) throws IOException {
                new Client();
        }
}
```

Convolution.java

```
package proj_2_2;



import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Formatter;
import java.util.Scanner;


public class Convolution extends Thread {
	private Socket client;
	private Scanner fromNet = null;
	private Formatter toNet = null;
	//int krow = fromNet.nextInt();
//	int kcol = fromNet.nextInt();
//	int row = fromNet.nextInt();
	//int col = fromNet.nextInt();


	public Convolution(Socket client) {
		this.client = client;
	}
```

```
public void run() {
        try {

                fromNet = new Scanner(client.getInputStream());
                toNet = new Formatter(client.getOutputStream());
                DataInputStream dis = new DataInputStream(client.getInputStream());
                DataOutputStream dos = new DataOutputStream(client.getOutputStream());



                //System.out.println(" Enter a number of rows for kernel matrix [2 or 3]\n:");

                dos.writeUTF(" Enter a number of rows for kernel matrix [2 or 3]\n:");
                int row = dis.readInt();



                dos.writeUTF(" Enter same number for columns for kernel matrix [2 or 3]\n:");
                int col = dis.readInt();



                int inmat[][] = new int[row][col];
                for (int i = 0; i < row; i++) {
                        for (int j = 0; j < col; j++) {
```

```
                    inmat[i][j] = dis.readInt();
            }
}


dos.writeUTF("  Enter a number of rows for matrix size [8 or 16]\\n:");
int krow = dis.readInt();


dos.writeUTF("  Enter same number for columns for matrix size [8 or 16]\\n:");
int kcol = dis.readInt();


int kmat[][] = new int[krow][kcol];
for (int i = 0; i < krow; i++) {
            for (int j = 0; j < kcol; j++) {
                    kmat[i][j] = dis.readInt();
            }
}


int convulationMat[][] = convolution2D2(kmat, inmat);
```

```
                //int[][] convulationMat = convolution2D(kmat, kcol, krow, inmat,

                    //          col, row);

                    dos.writeInt(convulationMat.length);

                    dos.writeInt(convulationMat[0].length);

                                for (int i = 0; i < convulationMat.length; i++) {

                                for (int j = 0; j < convulationMat[0].length; j++) {

dos.writeUTF(convulationMat[i][j] + " ");

                                }

                                dos.writeUTF("\n");

                                }
```

```
        } catch (IOException ioe) {
                System.err.println(ioe);
        } finally {
                try {
                        if (client != null)
                                client.close();
                        if (fromNet != null)
                                fromNet.close();
                        if (toNet != null)
                                toNet.close();
                } catch (Exception e) {
                        System.err.println(e);
                }
        }



}


public static int[][] convolution2D2(int[][] input,int[][] kernel) {
```

```
            int rows = input[0].length;
            int cols= input.length;
            int kCols= kernel[0].length;
int kRows= kernel.length;
int kCenterX = kCols / 2;
int kCenterY = kRows / 2;
int [][] out= new int [rows][cols];

int mm;
int ii;
int jj;
int nn;

for(int i=0; i < rows; ++i)            // rows
{
    for(int j=0; j < cols; ++j)         // columns
    {
            for(int m=0; m < kRows; ++m)     // kernel rows
            {
                    mm = kRows - 1 - m;      // row index of flipped kernel

                    for(int n=0; n < kCols; ++n) // kernel columns
                    {
                            nn = kCols - 1 - n;  // column index of flipped kernel
```

```
                        // index of input signal, used for checking boundary
                        ii = i + (kCenterY - mm);
                        jj = j + (kCenterX - nn);

                        // ignore input samples which are out of bound
                        if( ii >= 0 && ii < rows && jj >= 0 && jj < cols )
                                out[i][j] += input[ii][jj] * kernel[mm][nn];
                    }
                }
    }
}return out;
    }

    public static int sConvolution(int[][] input, int x, int y, int[][] k, int kernelWidth,

                    int kernelHeight) {

                    int output = 0;

                    for (int i = 0; i < kernelWidth; ++i) {

                    for (int j = 0; j < kernelHeight; ++j) {
```

```
                        output = output + (input[x + i][y + j] * k[i][j]);

                    }

                }

                return output;

            }

            public static int[][] convolution2D(int[][] input, int width, int height, int[][]
kernel, int kernelWidth,

                    int kernelHeight) {

                int smallWidth = width - kernelWidth + 1;

                int smallHeight = height - kernelHeight + 1;

                int[][] output = new int[smallWidth][smallHeight];

                for (int i = 0; i < smallWidth; ++i) {

                    for (int j = 0; j < smallHeight; ++j) {
```

```
                output[i][j] = 0;

                }

                }

                for (int i = 0; i < smallWidth; ++i) {

                for (int j = 0; j < smallHeight; ++j) {

                output[i][j] = sConvolution(input, i, j, kernel, kernelWidth, kernelHeight);

                }

                }

                return output;

                }
```

}Main.java

```
package proj_2_2;

// Java program to multiply two square matrices.

import java.io.*;
```

```
import java.util.Scanner;

public class Main {

   // Function to print Matrix
   static void printMatrix(int M[][],
                           int rowSize,
                           int colSize)
   {
      for (int i = 0; i < rowSize; i++) {
         for (int j = 0; j < colSize; j++)
            System.out.print(M[i][j] + " ");

         System.out.println();
      }
   }

   // Function to multiply
   // two matrices A[][] and B[][]
   static void multiplyMatrix(
         int row1, int col1, int A[][],
         int row2, int col2, int B[][])
   {
      int i, j, k;

      // Print the matrices A and B
      System.out.println("\nMatrix A:");
      printMatrix(A, row1, col1);
      System.out.println("\nMatrix B:");
```

```
    printMatrix(B, row2, col2);

    // Check if multiplication is Possible
    if (row2 != col1) {

        System.out.println(
                "\nMultiplication Not Possible");
        return;
    }

    // Matrix to store the result
    // The product matrix will
    // be of size row1 x col2
    int C[][] = new int[row1][col2];

    // Multiply the two matrices
    for (i = 0; i < row1; i++) {
        for (j = 0; j < col2; j++) {
            for (k = 0; k < row2; k++)
                C[i][j] += A[i][k] * B[k][j];
        }
    }

    // Print the result
    System.out.println("\nResultant Matrix:");
    printMatrix(C, row1, col2);
}

public static void fillingMatrix(Scanner scan, int num[][], int rows, int columns)
```

```
    {
        System.out.println("Please enter elements in the 2 matrices you defined above : ");
        for(int a = 0; a < rows; a++)
        {
            for(int b = 0; b < columns; b++)
            {
                num[a][b] = scan.nextInt();
            }
        }
    }

    // Driver code
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Please enter number of matrix rows for matrix 1: ");
        int row = sc.nextInt();
        System.out.println("Please enter number of matrix columns for matrix 1 : ");
        int col = sc.nextInt();
        int[][] numbers = new int[row][col];

       // Scanner sc = new Scanner(System.in);
        System.out.println("Please enter number of matrix rows for matrix 2: ");
        int row2matrix = sc.nextInt();
        System.out.println("Please enter number of matrix columns for matrix 2 : ");
        int col2matrix = sc.nextInt();
        int[][] number2 = new int[row2matrix][col2matrix];
//assigning matrix values
```

```
        fillingMatrix(sc, numbers, row, col);
        fillingMatrix(sc, number2, row2matrix, col2matrix);
        //end assigning values




        int row1 = row2matrix, col1 = col2matrix, row2 = row, col2 = col;

//      int A[][] = { { 1, 1, 1 },
//              { 2, 2, 2 },
//              { 3, 3, 3 },
//              { 4, 4, 4 } };

//      int B[][] = { { 1, 1, 1, 1 },
//              { 2, 2, 2, 2 },
//              { 3, 3, 3, 3 } };

        int B[][] = numbers;
        int A[][] = number2;

        multiplyMatrix(row1, col1, A,
              row2, col2, B);
    }
```

```
}
```

MaxMin.java

```
package proj_2_2;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Formatter;
import java.util.Scanner;

public class MaxMin {
	private Socket client;
	private Scanner fromNet = null;
	private Formatter toNet = null;

	public MaxMin(Socket client) {
		this.client = client;
	}
	public void run() {
		try {
			fromNet = new Scanner(client.getInputStream());
			toNet = new Formatter(client.getOutputStream());
			DataInputStream dis = new
DataInputStream(client.getInputStream());
```

```
                DataOutputStream dos = new
DataOutputStream(client.getOutputStream());

                dos.writeUTF(" Enter a number of rows for the matrix \n:");
                int arow = dis.readInt();


                dos.writeUTF(" Enter same number for columns for the matrix \n:");
                int acol = dis.readInt();


                int amat[][] = new int[arow][acol];
                for (int i = 0; i < arow; i++) {
                        for (int j = 0; j < acol; j++) {
                                amat[i][j] = dis.readInt();
                        }
                }



                dos.writeUTF("  The Maximum is  : "+getMaxValue(amat));

                dos.writeUTF("  The Minimum is  : "+getMinValue(amat));
```

```
        } catch (IOException ioe) {
                System.err.println(ioe);
        } finally {
                try {
                        if (client != null)
                                client.close();
                        if (fromNet != null)
                                fromNet.close();
                        if (toNet != null)
                                toNet.close();
                } catch (Exception e) {
                        System.err.println(e);
                }
        }

}

 public static int getMaxValue(int[][] numbers) {
        int maxValue = numbers[0][0];
        for (int j = 0; j < numbers.length; j++) {
            for (int i = 0; i < numbers[j].length; i++) {
                if (numbers[j][i] > maxValue) {
                    maxValue = numbers[j][i];
                }
```

```
            }
        }
        return maxValue;
    }

    public static int getMinValue(int[][] numbers) {
        int minValue = numbers[0][0];
        for (int j = 0; j < numbers.length; j++) {
            for (int i = 0; i < numbers[j].length; i++) {
                if (numbers[j][i] < minValue ) {
                    minValue = numbers[j][i];
                }
            }
        }
        return minValue ;
    }
}
```

Multiplication.java

```
package proj_2_2;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Formatter;
import java.util.Scanner;
```

```
public class Multiplication {
	private Socket client;
	private Scanner fromNet = null;
	private Formatter toNet = null;

	public Multiplication(Socket client) {
		this.client = client;
	}


	public void run() {
		try {

			fromNet = new Scanner(client.getInputStream());
			toNet = new Formatter(client.getOutputStream());
			DataInputStream dis = new
DataInputStream(client.getInputStream());
			DataOutputStream dos = new
DataOutputStream(client.getOutputStream());

			dos.writeUTF(" Enter a number of rows for A matrix\n:");
			int arow = dis.readInt();


			dos.writeUTF(" Enter same number for columns for A matrix \n:");
			int acol = dis.readInt();
```

```
int amat[][] = new int[arow][acol];
for (int i = 0; i < arow; i++) {
        for (int j = 0; j < acol; j++) {
                amat[i][j] = dis.readInt();
        }
}



dos.writeUTF("  Enter a number of rows for B matrix size \\n:");
int brow = dis.readInt();


dos.writeUTF("  Enter same number for columns B for matrix size
\\n:");
int bcol = dis.readInt();



int bmat[][] = new int[brow][bcol];
for (int i = 0; i < brow; i++) {
        for (int j = 0; j < bcol; j++) {
                bmat[i][j] = dis.readInt();
        }
}
```

```
int multiply[][] = new int[arow][bcol];

int sum = 0;
for (int c = 0; c < arow; c++)
{
  for (int d = 0; d < bcol; d++)
  {
    for (int k = 0; k < brow; k++)
    {
      sum = sum + amat[c][k]*bmat[k][d];
    }

    multiply[c][d] = sum;
    sum = 0;
  }
}



//return the multiplication to the client
dos.writeInt(multiply.length);
        dos.writeInt(multiply[0].length);

for (int i = 0; i < multiply.length; i++) {
```

```
                        for (int j = 0; j < multiply[0].length; j++) {

dos.writeUTF(multiply[i][j] + " ");

                        }

                        dos.writeUTF("\n");

                        }



    } catch (IOException ioe) {
            System.err.println(ioe);
    } finally {
            try {
                    if (client != null)
                            client.close();
                    if (fromNet != null)
                            fromNet.close();
                    if (toNet != null)
                            toNet.close();
            } catch (Exception e) {
                    System.err.println(e);
            }
    }
```

```
        }
}
```

question2.java

```
package proj_2_2;

// Java program for finding maximum
// and minimum in a matrix.

import java.util.Scanner;

class t
{
    static final int MAX = 100;

    // Finds maximum and minimum
    // in arr[0..n-1][0..n-1]
    // using pair wise comparisons
    static void maxMin(int arr[][], int n)
    {
        int min = +2147483647;
        int max = -2147483648;
```

```
    // Traverses rows one by one
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j <= n/2; j++)
        {
            // Compare elements from beginning
            // and end of current row
            if (arr[i][j] > arr[i][n - j - 1])
            {
                if (min > arr[i][n - j - 1])
                    min = arr[i][n - j - 1];
                if (max< arr[i][j])
                    max = arr[i][j];
            }
            else
            {
                if (min > arr[i][j])
                    min = arr[i][j];
                if (max< arr[i][n - j - 1])
                    max = arr[i][n - j - 1];
            }
        }
    }
    System.out.print("Maximum = "+max+
            ", Minimum = "+min);
}
```

```
public static void fillingMatrix(Scanner scan, int num[][], int rows, int columns)
{
    System.out.println("Please enter elements in the 2 matrices you defined above : ");
    for(int a = 0; a < rows; a++)
    {
        for(int b = 0; b < columns; b++)
        {
            num[a][b] = scan.nextInt();
        }
    }
}

// Driver program
public static void main (String[] args)
{
    //Scanner sc = new Scanner(System.in);
    Scanner sc = new Scanner(System.in);
    System.out.println("Please enter number of matrix rows for your matrix: ");
    int row = sc.nextInt();
    System.out.println("Please enter number of matrix columns for your matrix  : ");
    int col = sc.nextInt();
    int[][] numbers = new int[row][col];

    fillingMatrix(sc, numbers, row, col);
    int arr[][] = numbers;
    maxMin(arr, 3);
}
```

```
}
```

Replace.java

```
package proj_2_2;


import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Formatter;
import java.util.Scanner;

public class Replace extends Thread {

	private Socket client;
	private Scanner fromNet = null;
	private Formatter toNet = null;

	public Replace(Socket client)
	{
		this.client = client;
		//this.start();
	}
```

```
        public void run() {
                try {

                        fromNet = new Scanner(client.getInputStream());
                        toNet = new Formatter(client.getOutputStream());
                        DataInputStream dis = new
DataInputStream(client.getInputStream());
                        DataOutputStream dos = new
DataOutputStream(client.getOutputStream());

                        dos.writeUTF(" Enter a number of rows for the matrix \n:");
                        int arow = dis.readInt();


                        dos.writeUTF(" Enter same number for columns for the matrix
\n:");
                        int acol = dis.readInt();

                        dos.writeUTF(" Enter Element of the matrix \n:");

                        int amat[][] = new int[arow][acol];
                        for (int i = 0; i < arow; i++) {
                                for (int j = 0; j < acol; j++) {
                                        amat[i][j] = dis.readInt();
```

```
                }
            }

            dos.writeUTF(" Enter the number that the you  wants to replace
\n:");
            int one = dis.readInt();

            dos.writeUTF(" Enter the number to be replaced with. \n:");
            int tow = dis.readInt();

        for (int row = 0 ; row < arow ; row++) {
           for (int col = 0 ; col < acol ; col++) {
              int d = col-row;
              if (amat[row][col] == one) {
                 amat[row][col] = tow;
              }
           }
        }

            for (int i = 0; i < amat.length; i++) {
```

```
                for (int j = 0; j < amat[0].length; j++) {

dos.writeUTF(amat[i][j] + " ");

                }

                dos.writeUTF("\n");

                }




        } catch (IOException ioe) {
                System.err.println(ioe);
        } finally {
                try {
                        if (client != null)
```

```
                client.close();
            if (fromNet != null)
                fromNet.close();
            if (toNet != null)
                toNet.close();
        } catch (Exception e) {
            System.err.println(e);
        }
    }
}
```

Server.java

```
package proj_2_2;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Server extends Thread{
    ServerSocket server;
    Socket client;
    int port;

    public Server(int port) {
        this.port = port;
```

```
}

public void run() {
        try {
                server = new ServerSocket(port);

                while (true) {
                        client = server.accept();
                        System.out.println("A new client is connected : " + client);
                        switch (port) {
                        case 2000:
                                new MaxMin(client).run();
                                break;
                        case 3000:
                                new Multiplication(client).run();
                                break;
                        case 4000:
                                new SortMatrixService(client).run();
                                break;
                        case 5000:
                                new SumAboveBelowDiagonal(client).run();
                                break;
                        case 6000:
                                new Convolution(client).run();
                                break;
                        case 7000:
                                new Replace(client).run();
```

```
                        break;
                    case 8000:
                        new Average(client).run();
                        break;
                    default:
                        System.exit(0);
                    }
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        public static void main(String[] args) {
            new Server(2000).start();
            new Server(3000).start();
            new Server(4000).start();
            new Server(5000).start();
            new Server(6000).start();
            new Server(7000).start();
            new Server(8000).start();
        }
}
```

SortMatrixServer.java

```
package proj_2_2;
```

```
import java.io.IOException;
import java.net.Socket;
import java.util.Arrays;
import java.util.Formatter;
import java.util.Scanner;

public class SortMatrixService extends Thread{
	private Socket client;

	private Formatter toNet  = null ;
	private Scanner fromNet= null;

	public SortMatrixService(Socket client) {
		this.client=client;

	}

	public void run() {

		try {

			fromNet = new Scanner(client.getInputStream());
			toNet = new Formatter(client.getOutputStream());

			int numRow = fromNet.nextInt();
			int numCol = fromNet.nextInt();
			int mat[][] = new int[numRow][numCol];
			int array[] = new int[numRow * numCol];
```

```
for (int i = 0; i < numRow; i++) {
        for (int j = 0; j < numCol; j++) {
                mat[i][j] = fromNet.nextInt();
        }
}

int k = 0;
for (int i = 0; i < numRow; i++)
        for (int j = 0; j < numCol; j++)
                array[k++] = mat[i][j];
Arrays.sort(array);
int choosen = fromNet.nextInt();


if(choosen == 1) {
        k = 0;
        for (int i = 0; i < numRow; i++)
                for (int j = 0; j < numCol; j++)
                        mat[i][j] = array[k++];
        for (int i = 0; i <numRow; i++)
                for (int j = 0; j < numCol; j++)
                        toNet.format("%d\n",mat[i][j]).flush();
}
else if (choosen == 2) {
        k = array.length-1;
        for (int i = 0; i <numRow; i++)
                for (int j = 0; j < numCol; j++)
                        mat[i][j] = array[k--];
        for (int i = 0; i <numRow; i++)
```

```
                                        for (int j = 0; j < numCol; j++)
                                                toNet.format("%d\n",mat[i][j]).flush();
                        }


                } catch (IOException ioe) {
                        System.err.println(ioe);
                } finally {
                        try {
                                if (client != null)
                                        client.close();
                                if (fromNet != null)
                                        fromNet.close();
                                if (toNet != null)
                                        toNet.close();
                        } catch (Exception e) {
                                System.err.println(e);
                        }
                }
        }

}
```

SumAboveBelowDiagonal.java

```
package proj_2_2;




import java.io.IOException;
```

```
import java.net.Socket;
import java.util.Formatter;
import java.util.Scanner;

public class SumAboveBelowDiagonal extends Thread {

	private Socket client;

	private Formatter toNet  = null ;
	private Scanner fromNet= null;

	public SumAboveBelowDiagonal(Socket client) {
		this.client=client;
	}

	public void run() {

		try {
			fromNet = new Scanner(client.getInputStream());
			toNet = new Formatter(client.getOutputStream());

			int numRow = fromNet.nextInt();
			int numCol = fromNet.nextInt();
			int mat[][] = new int[numRow][numCol];
```

```
for (int i = 0; i < numRow; i++) {
        for (int j = 0; j < numCol; j++) {
                mat[i][j] = fromNet.nextInt();
        }
}

//to calculate sum of elements above diagonal.
int sumAbove=0;
for (int j = 1; j < numCol; j++) {
        for (int i=j-1 ; i>=0 ; i--) {
                sumAbove= sumAbove + mat[i][j];
        }

}

//to calculate sum of elements below diagonal.
int sumbelow=0;
for (int i = 1; i < numRow; i++) {
        for (int j=i-1 ; j>=0 ; j--) {
                sumbelow= sumbelow + mat[i][j];
        }
}

toNet.format("%d\n",sumAbove).flush();
```

```
                toNet.format("%d\n",sumbelow).flush();


            } catch (IOException ioe) {
                System.err.println(ioe);
            } finally {
                try {
                    if (client != null)
                        client.close();
                    if (fromNet != null)
                        fromNet.close();
                    if (toNet != null)
                        toNet.close();
                } catch (Exception e) {
                    System.err.println(e);
                }
            }
        }
}
```

Output

Multiplication

```
9
  Enter a number of rows for B matrix size \n:
3
  Enter same number for columns B for matrix size \n:
3
1
2
3
4
5
6
7
8
9
Multiplication of 2 Matrix:
30 36 42
66 81 96
102 126 150


Choose from the menu:
0. for exit
1. matrix multiplication
2. matrix convolution
3. Sum numbers below the diagonal line and above the diagonal line
4. Find the maximum and minimum elements in a matrix
5. Sort the elements of a matrix
6. Get the average of two matrices and find the maximum average
 7. Find a number in the matrix and replace it with 0 or 1
```

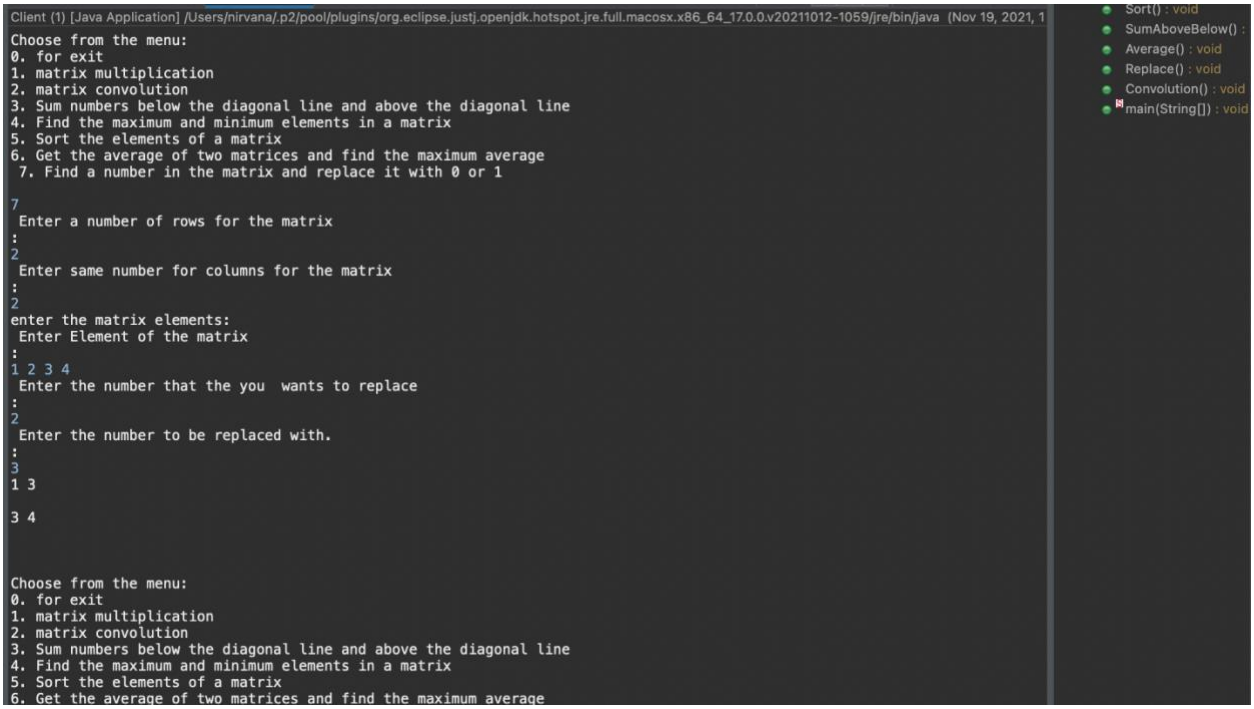
Maxmin

```
Choose from the menu:
0. for exit
1. matrix multiplication
2. matrix convolution
3. Sum numbers below the diagonal line and above the diagonal line
4. Find the maximum and minimum elements in a matrix
5. Sort the elements of a matrix
6. Get the average of two matrices and find the maximum average
 7. Find a number in the matrix and replace it with 0 or 1

4
 Enter a number of rows for the matrix
:
2
 Enter same number for columns for the matrix
:
2
enter the matrix elements:
1 2 3 4
 The Maximum is  : 4
  The Minimum is  : 1


Choose from the menu:
0. for exit
1. matrix multiplication
2. matrix convolution
3. Sum numbers below the diagonal line and above the diagonal line
4. Find the maximum and minimum elements in a matrix
5. Sort the elements of a matrix
6. Get the average of two matrices and find the maximum average
 7. Find a number in the matrix and replace it with 0 or 1
```

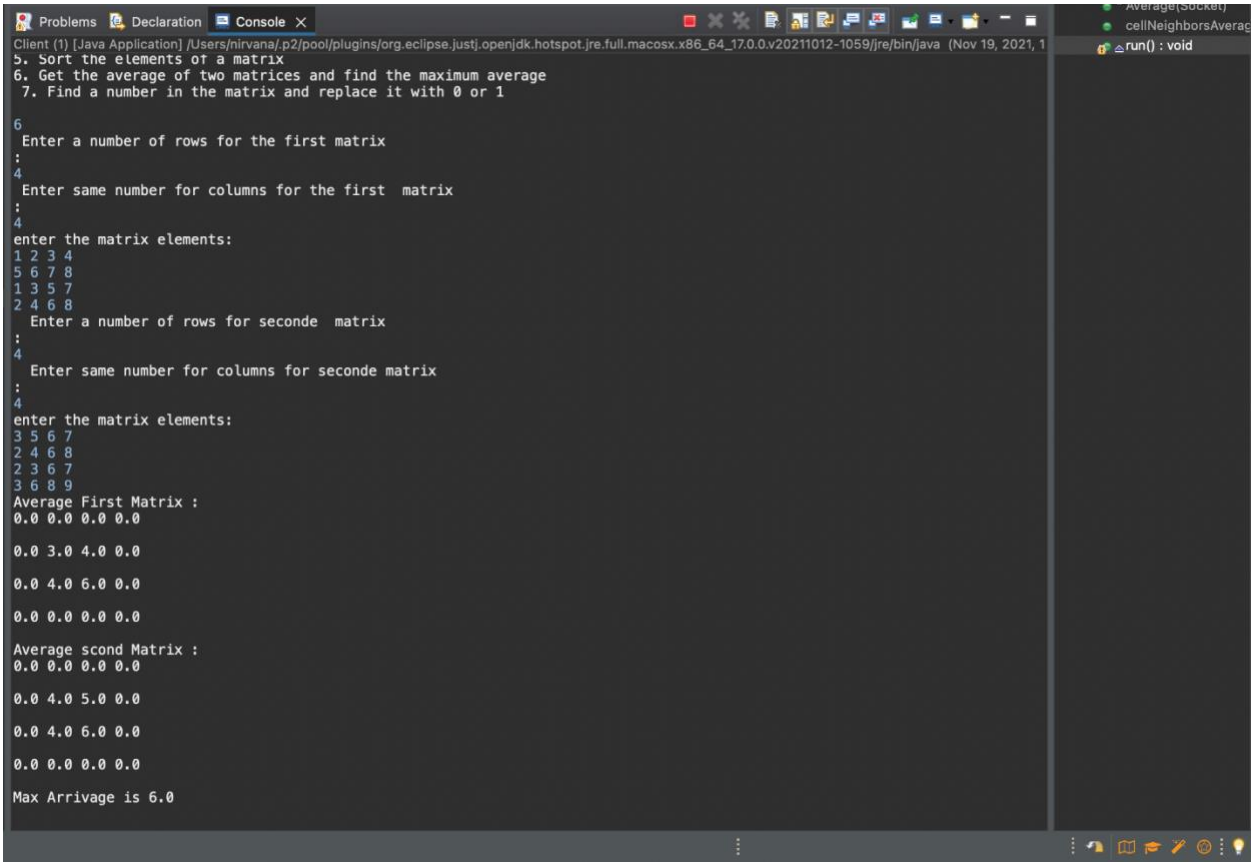
Convolution

```
1. matrix multiplication
2. matrix convolution
3. Sum numbers below the diagonal line and above the diagonal line
4. Find the maximum and minimum elements in a matrix
5. Sort the elements of a matrix
6. Get the average of two matrices and find the maximum average
7. Find a number in the matrix and replace it with 0 or 1

2
 Enter a number of rows for kernel matrix [2 or 3]
:
2
 Enter same number for columns for kernel matrix [2 or 3]
:
2
enter the matrix elements:
1 2 3 4
  Enter a number of rows for matrix size [8 or 16]\n:
8
  Enter same number for columns for matrix size [8 or 16]\n:
8
enter the matrix elements:
9 10 6 2 2 3 9 3
9 3 10 4 2 3 7 5
4 9 6 6 10 3 8 8
4 6 1 1 6 8 10 2
6 5 1 4 2 8 4 6
2 7 3 7 7 8 5 1
3 10 2 9 6 10 1 5
2 10 1 8 9 9 1 4
Convulation Matrix:
87 74 54 24 24 52 64 22

62 66 70 44 40 47 67 36

57 67 45 62 69 62 78 36

51 38 13 32 60 82 60 20

50 40 29 43 54 65 45 26

45 59 46 73 74 68 30 14

56 67 45 79 81 62 25 28

38 43 28 59 63 39 16 16
```

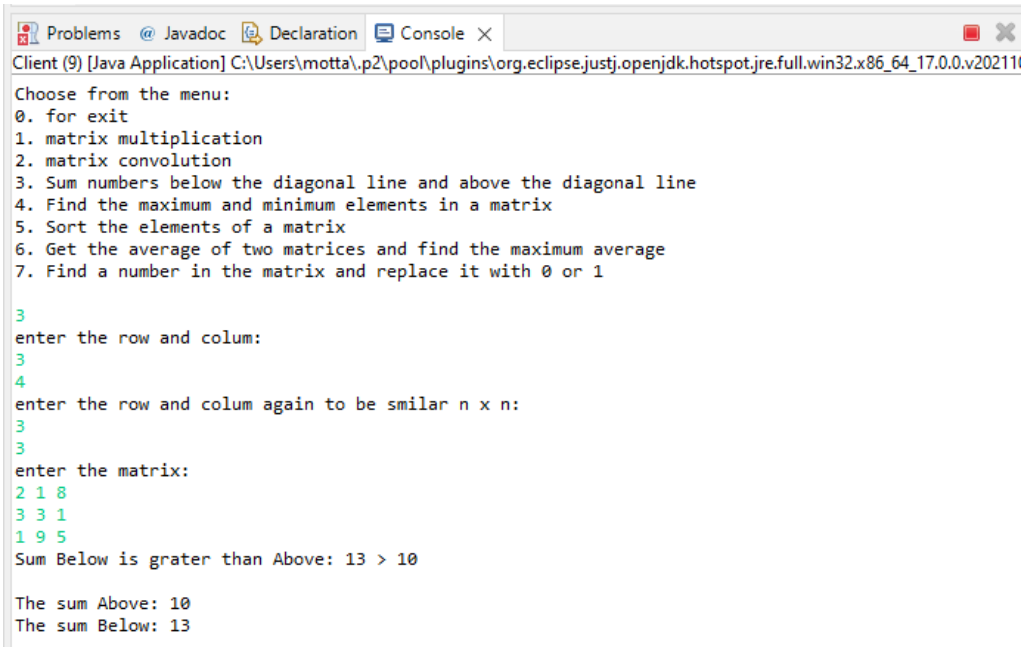
Replace



Average



SumAboveBelow



Sort

```
Choose from the menu:
0. for exit
1. matrix multiplication
2. matrix convolution
3. Sum numbers below the diagonal line and above the diagonal line
4. Find the maximum and minimum elements in a matrix
5. Sort the elements of a matrix
6. Get the average of two matrices and find the maximum average
7. Find a number in the matrix and replace it with 0 or 1

5
enter the row and colum:
3
7
enter the row and colum again to be smilar n x n:
3
3
enter the matrix:
-9 8 0
2 7 8
-3 7 5
The sort is ascending or descending? 1 for ascending, 2 for descending
1
-9 -3 0
2 5 7
7 8 8
```

```
Choose from the menu:
0. for exit
1. matrix multiplication
2. matrix convolution
3. Sum numbers below the diagonal line and above the diagonal line
4. Find the maximum and minimum elements in a matrix
5. Sort the elements of a matrix
6. Get the average of two matrices and find the maximum average
7. Find a number in the matrix and replace it with 0 or 1

5
enter the row and colum:
3
3
enter the matrix:
-9 8 0
2 7 8
-3 7 5
The sort is ascending or descending? 1 for ascending, 2 for descending
2
8 8 7
7 5 2
0 -3 -9
```