//SERVER

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class AddServer {

    public static void main(String args[]) {

        try {

            LocateRegistry.createRegistry(12345); // Starts RMI registry on port 12345

            AddServerImpl addServerImpl = new AddServerImpl();

            Naming.rebind("AddServer", addServerImpl);

            System.out.println("Server is ready.");

        } catch (Exception e) {

            System.out.println("Exception: " + e);

        }

    }

}

//CLIENT

import java.rmi.Naming;

import java.util.Scanner;

public class AddClient {

    public static void main(String args[]) {

        try {

            if (args.length < 1) {

                System.out.println("Usage: java AddClient <server-ip>");

                return;

            }

            String addServerURL = "rmi://" + args[0] + "/AddServer";

            AddServerIntf addServerIntf = (AddServerIntf) Naming.lookup(addServerURL);

            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter number of rows: ");

            int rows = scanner.nextInt();

            System.out.print("Enter number of columns: ");

            int cols = scanner.nextInt();

            int[][] matrix = new int[rows][cols];

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

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

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

                    matrix[i][j] = scanner.nextInt();

                }

            }

            System.out.println("\nOriginal Matrix:");

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

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

                    System.out.print(matrix[i][j] + " ");

                }

                System.out.println();

            }

            int[][] transposed = addServerIntf.transposeMatrix(matrix);

            System.out.println("\nTransposed Matrix:");

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

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

                    System.out.print(transposed[i][j] + " ");

                }

                System.out.println();

            }

            scanner.close();

        } catch (Exception e) {

            System.out.println("Exception: " + e);

        }

    }

}

//ServerImp

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class AddServerImpl extends UnicastRemoteObject implements AddServerIntf {

    public AddServerImpl() throws RemoteException {

        super();

    }

    public double add(double d1, double d2) throws RemoteException {

        return d1 + d2;

    }

    public String getSubstring(String mainString, int beginIndex, int endIndex) throws RemoteException {

        try {

            return mainString.substring(beginIndex, endIndex);

        } catch (Exception e) {

            return "Error in substring: " + e.getMessage();

        }

    }

    public int[][] transposeMatrix(int[][] matrix) throws RemoteException {

        int rows = matrix.length;

        int cols = matrix[0].length;

        int[][] transposed = new int[cols][rows];

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

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

                transposed[j][i] = matrix[i][j];

            }

        }

        return transposed;

    }

}

//AddServerIntf

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface AddServerIntf extends Remote {

    double add(double d1, double d2) throws RemoteException;

    String getSubstring(String mainString, int beginIndex, int endIndex) throws RemoteException;

    int[][] transposeMatrix(int[][] matrix) throws RemoteException; // Method for matrix transposition

}