Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Task 0

1. **Project2Task0Client:**

```
import java.net.*; //network package
import java.io.*;  //I/O package
import java.util.Scanner;

public class EchoClientUDP{
    public static void main(String args[]){
        System.out.println("The UDP client is running.");
        // Initialize a DatagramSocket for sending/receiving packets
        DatagramSocket aSocket = null;
        Scanner scanner = new Scanner(System.in);
        try {
            // Set the server address and port
            //Set 6789 first
            System.out.println("Please Enter server port number: ");
            int serverPort = scanner.nextInt();

            //hard coded the localhost for host
            InetAddress aHost = InetAddress.getByName("localhost");
            // Initialize a socket for sending UDP packets
            aSocket = new DatagramSocket();
            String nextLine;
            // Set BufferedReader to read user's input
            BufferedReader typed = new BufferedReader(new
InputStreamReader(System.in));

            System.out.println("The client is listening on port:
"+serverPort);
            // Loop until user inputs is end (ctrl+z)
            while ((nextLine = typed.readLine()) != null) {


                // Set user input to bytes
                byte [] m = nextLine.getBytes();
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
                // Set a UDP packet to contain the message, sending to
the server
                DatagramPacket request = new DatagramPacket(m,
m.length, aHost, serverPort);

                // Send the packet to the server
                aSocket.send(request);
                // Initialize a buffer
                byte[] buffer = new byte[1000];

                //Receive the reply from the server
                DatagramPacket reply = new DatagramPacket(buffer,
buffer.length);

                // Wait until receive the response from the server
                aSocket.receive(reply);

                // Get the exact number of bytes received
                byte[] replyData = new byte[reply.getLength()];
                System.arraycopy(reply.getData(), 0, replyData, 0,
reply.getLength());

                String replyString = new String(replyData);
                // Print out the reply
                System.out.println("Reply from server: " +
replyString);
                //Check if server reply halt! to shut down
                if (replyString.equals("halt!")) {
                    System.out.println("Server received halt!");
                    System.out.println("UDP Client side quitting.");
                    System.exit(0);
                }
            }

        }catch (SocketException e) {System.out.println("Socket
Exception: " + e.getMessage());
        }catch (IOException e){System.out.println("IO Exception: " +
e.getMessage());
        }finally {if(aSocket != null) aSocket.close();}
    }
}
```

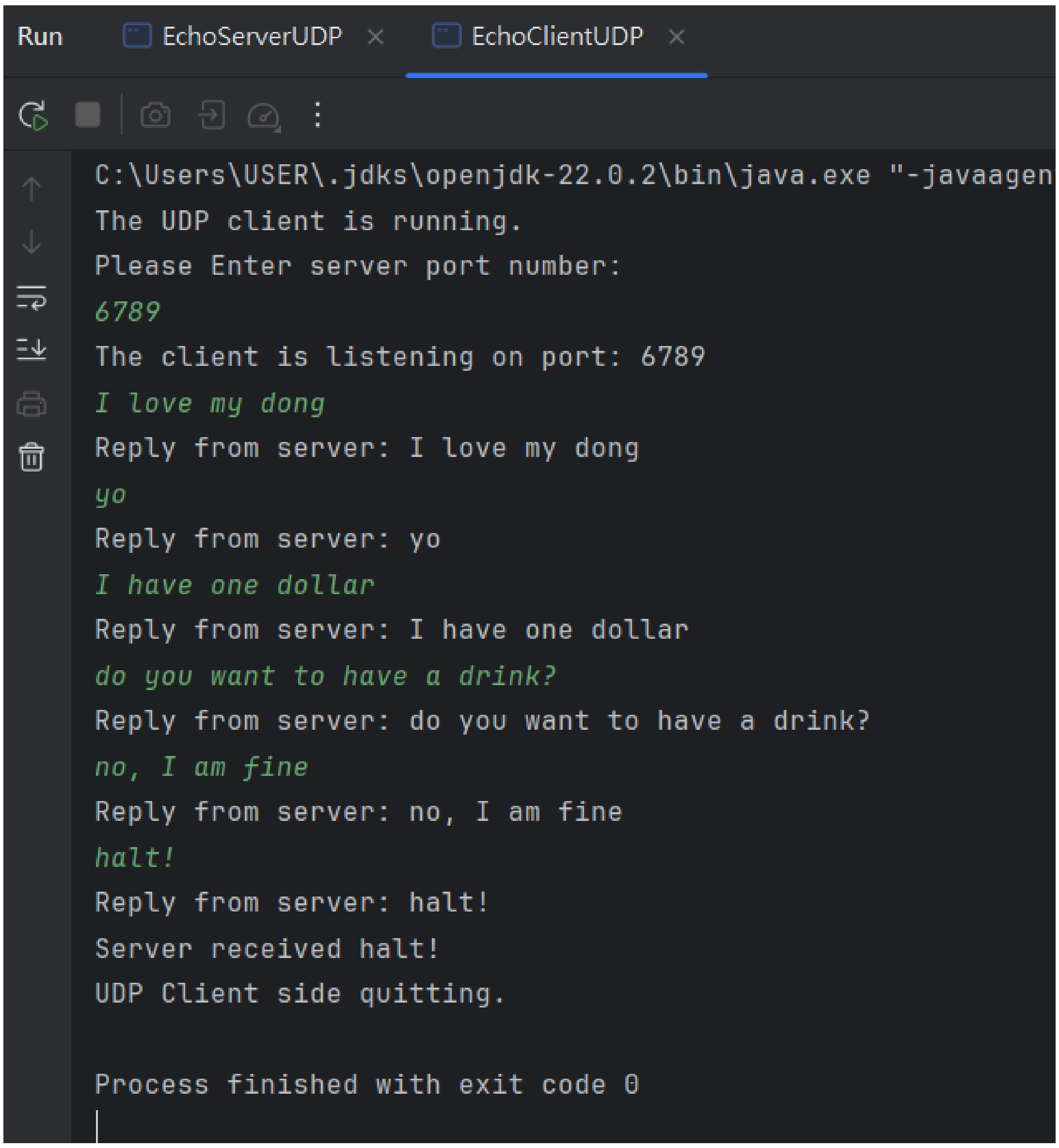
2. **Project2Task0Server:**

```
import java.net.*; //network package
import java.io.*;  //I/O package
import java.util.Scanner;

public class EchoServerUDP{
    public static void main(String args[]){
        System.out.println("The UDP server is running.");
        // Initialize a DatagramSocket for UDP communication
        DatagramSocket aSocket = null;
        // Initialize a buffer to store incoming data
        byte[] buffer = new byte[1000];

        Scanner scanner = new Scanner(System.in);
        try{
            //Prompt the listening port
            System.out.println("Enter the port number to listen on:
");
            int serverPort = scanner.nextInt();
            // Set a socket to port 6789 to listen for packets from
client
            aSocket = new DatagramSocket(serverPort);
            System.out.println("The server is listening on port:
"+serverPort);
            // Set a DatagramPacket to receive data from client
            DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
            //Let server keep running
            while(true){
                //wait until client send package to server
                aSocket.receive(request);

                //Get the correct number of bytes received
                byte[] requestData = new byte[request.getLength()];
                System.arraycopy(request.getData(), 0, requestData,
0, request.getLength());
                // Convert the received data to a string
                String requestString = new String(requestData);


                //Check if client sent halt! to shut down
                if (requestString.equals("halt!")) {
                    System.out.println("Server received halt!
command.");
                    System.out.println("UDP Server side quitting");
                    //send the reply of halt! to client
                    DatagramPacket reply = new DatagramPacket(
                            "halt!".getBytes(), "halt!".length(),
request.getAddress(), request.getPort()
                    );
                    aSocket.send(reply);
                    System.exit(0);
                }

                // Create a reply packet using the received client's
data, address, and port
                DatagramPacket reply = new DatagramPacket(
                        requestData, requestData.length,
request.getAddress(), request.getPort()
                );

                // Print out the data
                System.out.println("Echoing: "+ requestString);

                // Send the reply packet back to the client
                aSocket.send(reply);
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
            }
        }catch (SocketException e){System.out.println("Socket: " +
e.getMessage());
        }catch (IOException e) {System.out.println("IO: " +
e.getMessage());
        }finally {if(aSocket != null) aSocket.close();}
    }
}
```

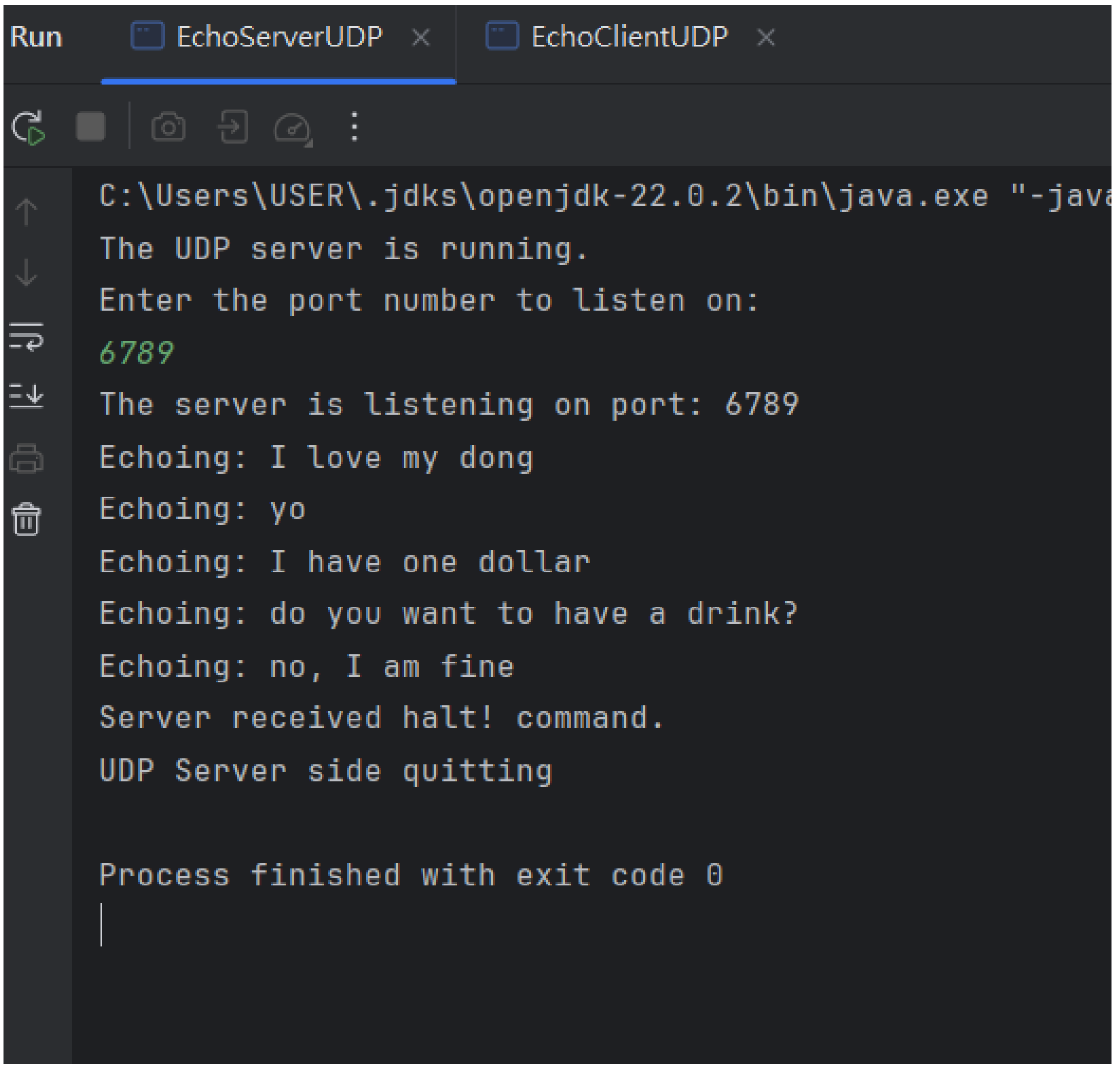
3. **"Project2Task0ClientConsole".**



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagen
The UDP client is running.
Please Enter server port number:
6789
The client is listening on port: 6789
I love my dong
Reply from server: I love my dong
yo
Reply from server: yo
I have one dollar
Reply from server: I have one dollar
do you want to have a drink?
Reply from server: do you want to have a drink?
no, I am fine
Reply from server: no, I am fine
halt!
Reply from server: halt!
Server received halt!
UDP Client side quitting.

Process finished with exit code 0
```

4. **"Project2Task0ServerConsole".**

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-java
The UDP server is running.
Enter the port number to listen on:
6789
The server is listening on port: 6789
Echoing: I love my dong
Echoing: yo
Echoing: I have one dollar
Echoing: do you want to have a drink?
Echoing: no, I am fine
Server received halt! command.
UDP Server side quitting

Process finished with exit code 0
```

Task 1:

1. EavesdropperUDP.java program

```java
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.Scanner;

public class EavesdropperUDP {
    public static void main(String[] args) {
        System.out.println("The Eavesdropper is running.");
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
        // initialize eavesdropperSocket
        DatagramSocket eavesdropperSocket = null;
        Scanner scanner = new Scanner(System.in);

        try {
            // Type the port number for Eavesdropper to listen
            System.out.print("Enter the port number for Eavesdropper to
listen on: ");
            int listenPort = scanner.nextInt();
            System.out.print("Enter the masquerading server port: ");
            int serverPort = scanner.nextInt();

            // Set the socket with listenPort
            eavesdropperSocket = new DatagramSocket(listenPort);
            System.out.println("Eavesdropper is listening on port: " +
listenPort + " and masquerading as the server on port: " +
serverPort);

            byte[] buffer = new byte[1000];

            while (true) {
                DatagramPacket packetFromClient = new
DatagramPacket(buffer, buffer.length);
                //Received client's message if client set the same port
number
                eavesdropperSocket.receive(packetFromClient);

                //get client's Message
                String receivedMessage = new
String(packetFromClient.getData(), 0, packetFromClient.getLength());
                System.out.println("Received message from client: " +
receivedMessage);

                // If the message contains "like", replace the first
"like" to "dislike"
                if (receivedMessage.contains("like")) {
                    receivedMessage =
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
receivedMessage.replaceFirst("like", "dislike");
                    System.out.println("Modified message: " +
receivedMessage);
                }

                // Send the message to real server
                byte[] modifiedData = receivedMessage.getBytes();
                InetAddress serverAddress =
InetAddress.getByName("localhost");
                //Send the modified data to the actual server
                DatagramPacket packetToServer = new
DatagramPacket(modifiedData, modifiedData.length, serverAddress,
serverPort);
                eavesdropperSocket.send(packetToServer);

                // Receive response from server
                DatagramPacket packetFromServer = new
DatagramPacket(buffer, buffer.length);
                eavesdropperSocket.receive(packetFromServer);

                //get Response Message
                String serverResponse = new
String(packetFromServer.getData(), 0, packetFromServer.getLength());
                System.out.println("Server Response: " +
serverResponse);

                // Sent response back to client
                DatagramPacket packetToClient = new
DatagramPacket(serverResponse.getBytes(), serverResponse.length(),
packetFromClient.getAddress(), packetFromClient.getPort());
                eavesdropperSocket.send(packetToClient);
            }
        } catch (SocketException e) {
            System.out.println("Socket Exception: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
```

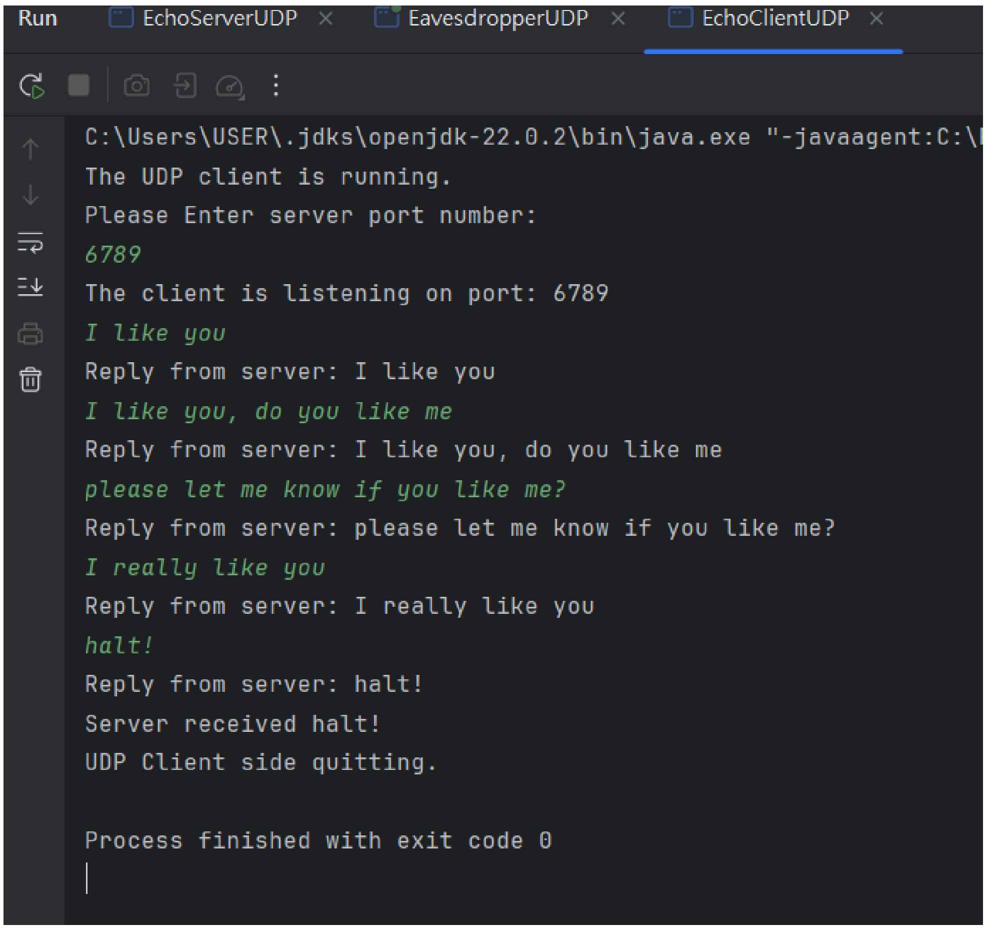
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
            if (eavesdropperSocket != null) eavesdropperSocket.close();
        }
    }
}
```

2. Project2Task1ThreeConsoles

When client is set to 6789:

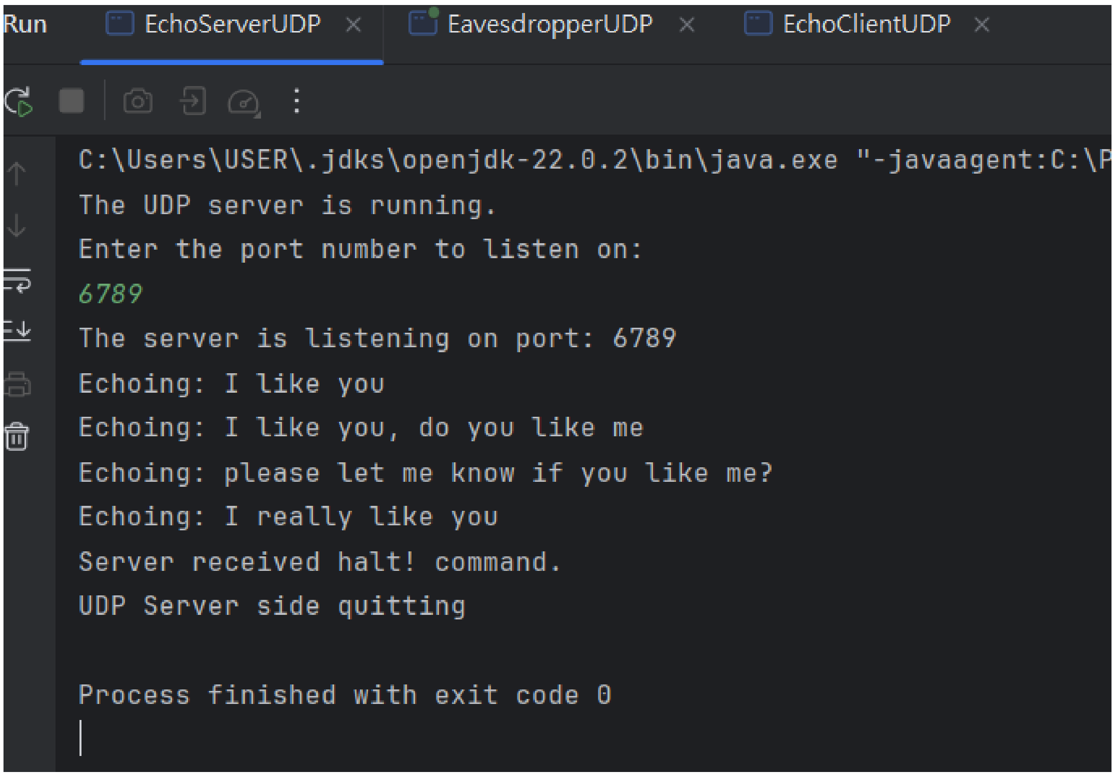
Client:



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\
The UDP client is running.
Please Enter server port number:
6789
The client is listening on port: 6789
I like you
Reply from server: I like you
I like you, do you like me
Reply from server: I like you, do you like me
please let me know if you like me?
Reply from server: please let me know if you like me?
I really like you
Reply from server: I really like you
halt!
Reply from server: halt!
Server received halt!
UDP Client side quitting.

Process finished with exit code 0
```

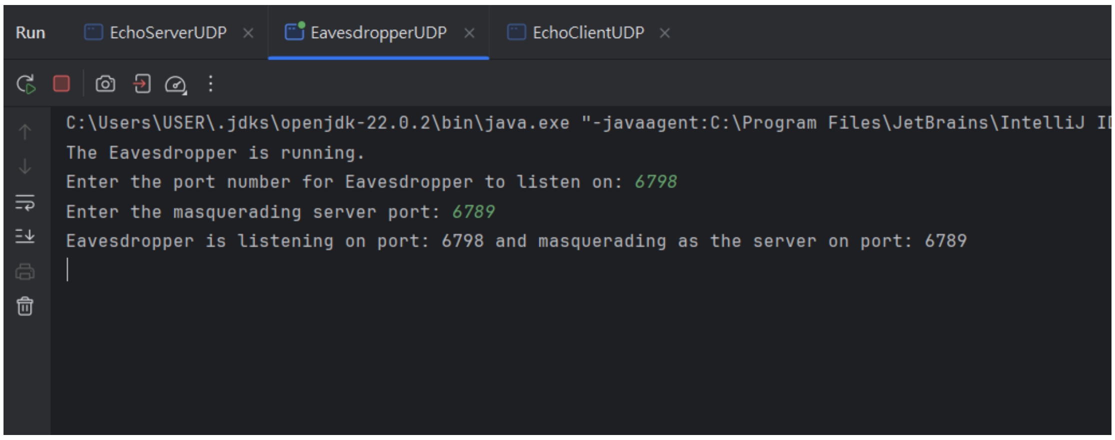
Server:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\P
The UDP server is running.
Enter the port number to listen on:
6789
The server is listening on port: 6789
Echoing: I like you
Echoing: I like you, do you like me
Echoing: please let me know if you like me?
Echoing: I really like you
Server received halt! command.
UDP Server side quitting

Process finished with exit code 0
```

Eavesdropper:

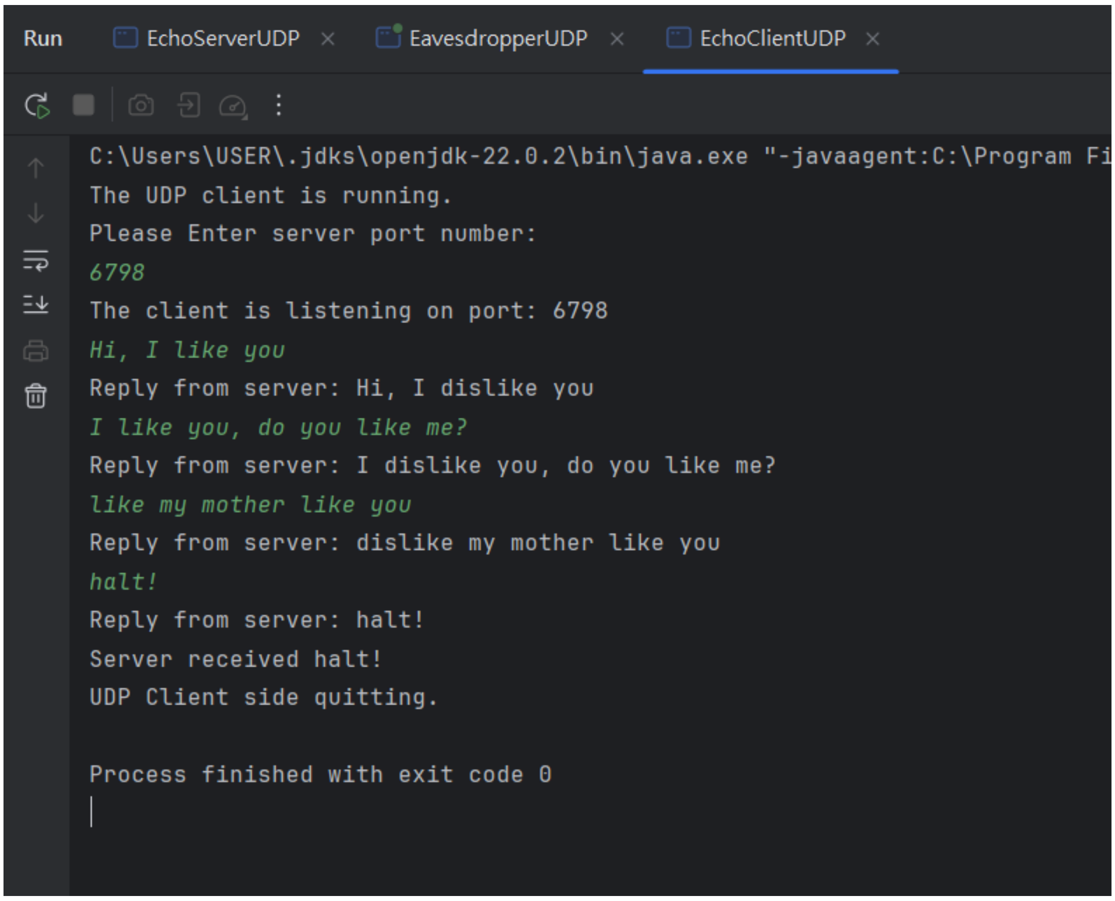


```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ I
The Eavesdropper is running.
Enter the port number for Eavesdropper to listen on: 6798
Enter the masquerading server port: 6789
Eavesdropper is listening on port: 6798 and masquerading as the server on port: 6789
```

When client is set to 6798:

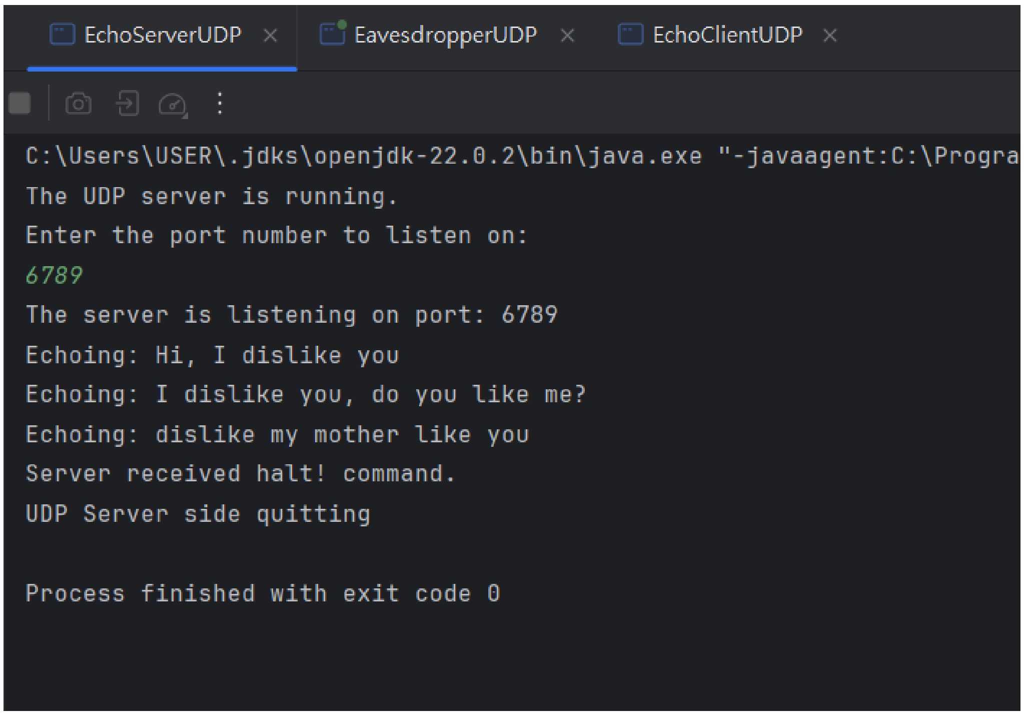
Client

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Fi
The UDP client is running.
Please Enter server port number:
6798
The client is listening on port: 6798
Hi, I like you
Reply from server: Hi, I dislike you
I like you, do you like me?
Reply from server: I dislike you, do you like me?
like my mother like you
Reply from server: dislike my mother like you
halt!
Reply from server: halt!
Server received halt!
UDP Client side quitting.

Process finished with exit code 0
```

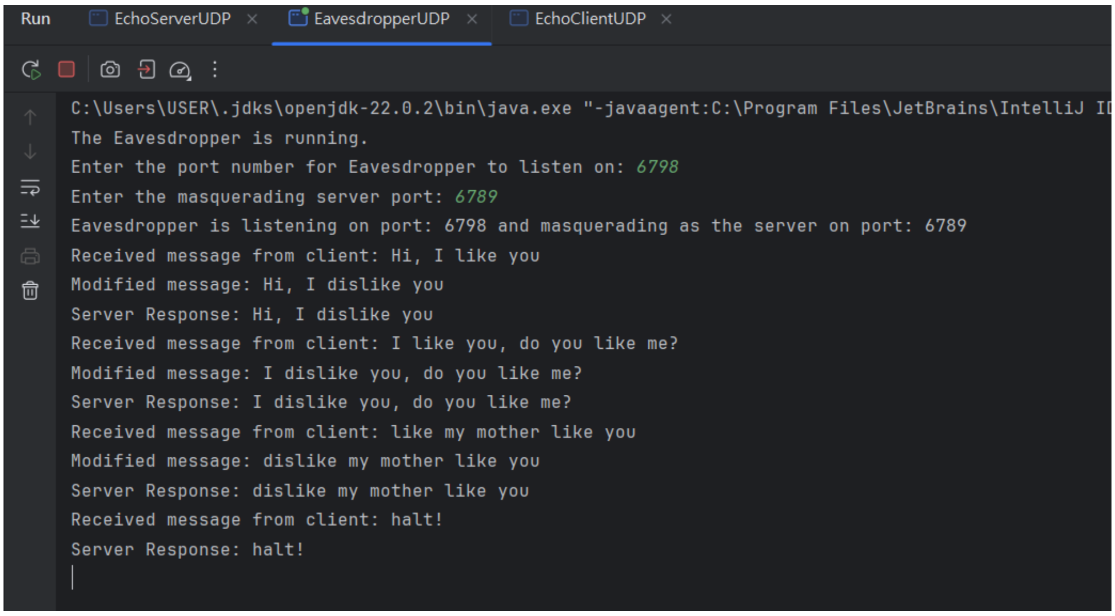
Server:



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Progra
The UDP server is running.
Enter the port number to listen on:
6789
The server is listening on port: 6789
Echoing: Hi, I dislike you
Echoing: I dislike you, do you like me?
Echoing: dislike my mother like you
Server received halt! command.
UDP Server side quitting

Process finished with exit code 0
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Eavesdropper:



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ I
The Eavesdropper is running.
Enter the port number for Eavesdropper to listen on: 6798
Enter the masquerading server port: 6789
Eavesdropper is listening on port: 6798 and masquerading as the server on port: 6789
Received message from client: Hi, I like you
Modified message: Hi, I dislike you
Server Response: Hi, I dislike you
Received message from client: I like you, do you like me?
Modified message: I dislike you, do you like me?
Server Response: I dislike you, do you like me?
Received message from client: like my mother like you
Modified message: dislike my mother like you
Server Response: dislike my mother like you
Received message from client: halt!
Server Response: halt!
```

Task 2:

1. Project2Task2Client

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.Scanner;

public class AddingClientUDP {
    private static int serverPort;
    private static InetAddress aHost;
    private static DatagramSocket aSocket;

    public static void main(String args[]){
        System.out.println("The client is running.");
        // Initialize a DatagramSocket for sending/receiving packets
        aSocket = null;
        Scanner scanner = new Scanner(System.in);
        try {
            // Set the server address and port
            //Set 6789 first
            System.out.println("Please enter server port: ");
            serverPort = scanner.nextInt();

            //hard coded the localhost for host
            aHost = InetAddress.getByName("localhost");
            // Initialize a socket for sending UDP packets
            aSocket = new DatagramSocket();
            String nextLine;
            // Set BufferedReader to read user's input
            BufferedReader typed = new BufferedReader(new
InputStreamReader(System.in));

            // Loop until user inputs is end (ctrl+z)
            while ((nextLine = typed.readLine()) != null) {

                String requestString = nextLine;
                if (requestString.equals("halt!")) {
                    System.out.println("Client side quitting.");
                    System.exit(0);
                }

                try{
                    //get the value from user's prompt
                    int clientValue = Integer.parseInt(nextLine);
                    //call add to request server to sum the value
                    clientValue = add(clientValue);
                    System.out.println("The server returned " +
clientValue);

                }catch (NumberFormatException e){
                    System.out.println("Input is not the number. Please
enter a valid number");
                }
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
          }




       }catch (SocketException e) {System.out.println("Socket
Exception: " + e.getMessage());
       }catch (IOException e){System.out.println("IO Exception: " +
e.getMessage());
       }finally {if(aSocket != null) aSocket.close();}
    }
    public static int add(int i) {
       try{
          //Another approach would be to only transmit byte arrays
containing String data.
          //get the integer byte that provided by client
          byte [] m = String.valueOf(i).getBytes();
          //get the request packet and send
          DatagramPacket request = new DatagramPacket(m, m.length,
aHost, serverPort);
          aSocket.send(request);

          byte[] buffer = new byte[1000];
          //create a reply packet to receive server's message.
          DatagramPacket reply = new DatagramPacket(buffer,
buffer.length);
          aSocket.receive(reply);
          return Integer.parseInt(new String(reply.getData(), 0 ,
reply.getLength()));
       }catch (IOException e){
          System.out.println(e.getMessage());
          return -1;
       }
    }

}
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

2. Project2Task2Server

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.util.Scanner;

public class AddingServerUDP {

    private static int sumValue = 0;

    public static void main(String args[]){
        System.out.println("Server started");
        // Initialize a DatagramSocket for UDP communication
        DatagramSocket aSocket = null;
        // Initialize a buffer to store incoming data
        byte[] buffer = new byte[1000];


        Scanner scanner = new Scanner(System.in);
        try{
            //Prompt the listening port
            System.out.println("Enter the port number to listen on: ");
            int serverPort = scanner.nextInt();
            // Set a socket to port 6789 to listen for packets from
client
            aSocket = new DatagramSocket(serverPort);

            // Set a DatagramPacket to receive data from client
            DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
            //Let server keep running
            while(true){

                //wait until client send package to server
                aSocket.receive(request);
```

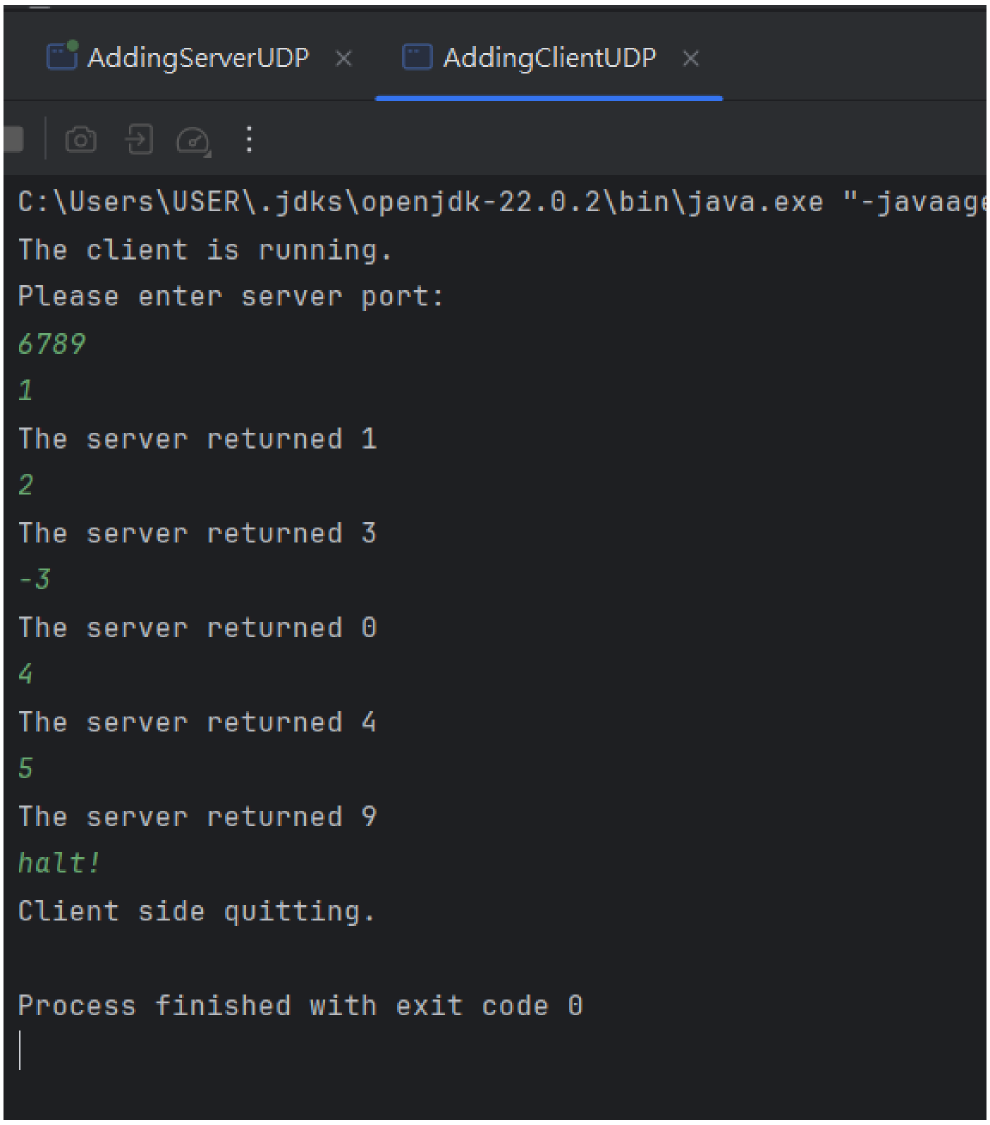
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
                //Get the correct number of bytes requested by client
                byte[] requestData = new byte[request.getLength()];
                System.arraycopy(request.getData(), 0, requestData, 0,
request.getLength());

                // Convert the client request data to a string
                String requestString = new String(requestData);

                int clientNumber;
                try{
                        clientNumber = Integer.parseInt(requestString);
                } catch (NumberFormatException e) {
                    System.out.println("The input should be Integer.
Error message " + requestString);
                    continue;
                }
                System.out.println("Adding: " + clientNumber+ " to "+
sumValue);
                int result = add(clientNumber);
                System.out.println("Returning sum of " + result +" to
client");



                String response = String.valueOf(sumValue);
                //Another approach would be to only transmit byte
arrays containing String data.
                byte[] responseData = response.getBytes();
                //packet the reply and send back to client
                DatagramPacket reply = new DatagramPacket(
                        responseData, responseData.length,
request.getAddress(), request.getPort()
                );
                aSocket.send(reply);

            }
        }catch (SocketException e){System.out.println("Socket: " +
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

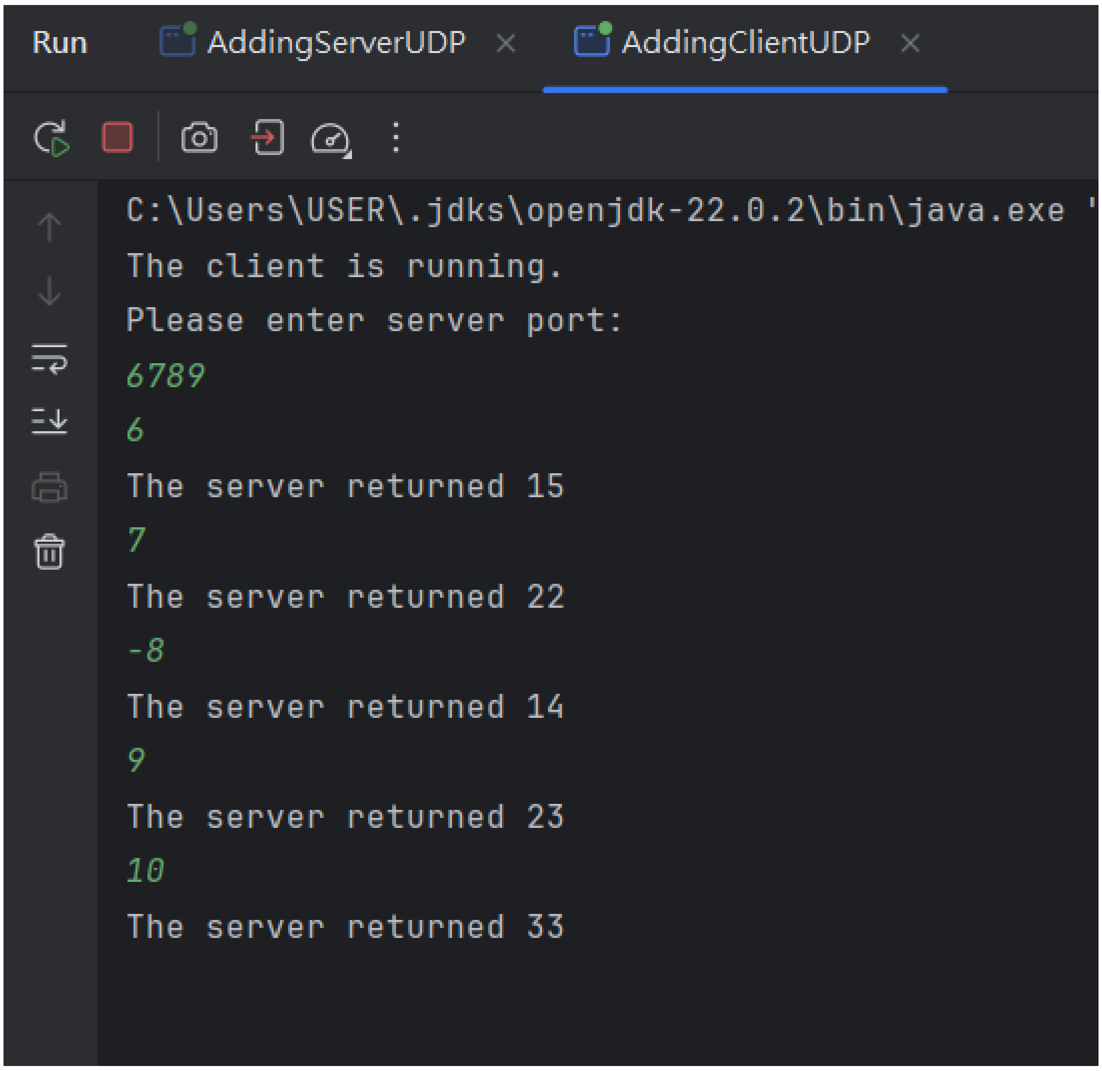
```
e.getMessage());
        }catch (IOException e) {System.out.println("IO: " +
e.getMessage());
        }finally {if(aSocket != null) aSocket.close();}
    }
    private static int add(int i) {
        sumValue += i;
        return sumValue;
    }
}
```

3. Project2Task2ClientConsole



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaage
The client is running.
Please enter server port:
6789
1
The server returned 1
2
The server returned 3
-3
The server returned 0
4
The server returned 4
5
The server returned 9
halt!
Client side quitting.

Process finished with exit code 0
```

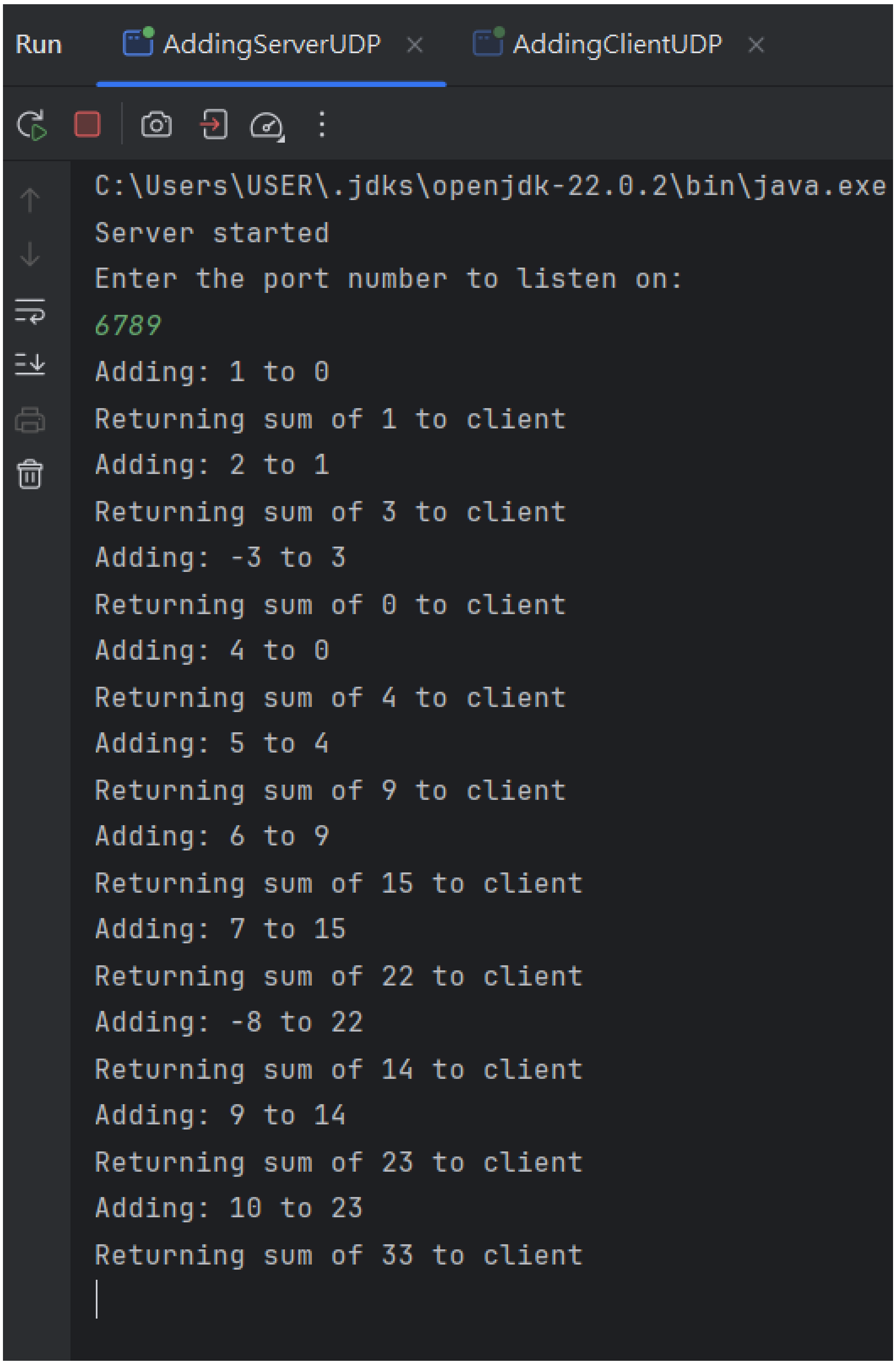
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe '
The client is running.
Please enter server port:
6789
6
The server returned 15
7
The server returned 22
-8
The server returned 14
9
The server returned 23
10
The server returned 33
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

4. Project2Task2ServerConsole



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe
Server started
Enter the port number to listen on:
6789
Adding: 1 to 0
Returning sum of 1 to client
Adding: 2 to 1
Returning sum of 3 to client
Adding: -3 to 3
Returning sum of 0 to client
Adding: 4 to 0
Returning sum of 4 to client
Adding: 5 to 4
Returning sum of 9 to client
Adding: 6 to 9
Returning sum of 15 to client
Adding: 7 to 15
Returning sum of 22 to client
Adding: -8 to 22
Returning sum of 14 to client
Adding: 9 to 14
Returning sum of 23 to client
Adding: 10 to 23
Returning sum of 33 to client
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

<u>Task 3:</u>

1. Project2Task3Client:

```
import java.io.IOException;
import java.net.*;
import java.util.Scanner;

public class RemoteVariableClientUDP {
    private static RemoteVariableProxy proxy;

    public static void main(String args[]){
        //client start
        System.out.println("The client is running.");
        Scanner scanner = new Scanner(System.in);

        try {

            // Set the server address and port
            System.out.println("Please enter server port: ");
            int serverPort = scanner.nextInt();
            //use proxy to set up the serverPort and localhost
            proxy = new RemoteVariableProxy(serverPort, "localhost");

            // Loop until user inputs is end (ctrl+z)
            while (true) {
                //selection for user's choice
                //number for user's input value
                //id for user ID
                //clientRequest is decided by user's choice
                int selection;
                int number = 0;
                int id;
                String clientRequest = "";
                //Options for user
                System.out.println("1. Add a value to your sum.");
                System.out.println("2. Subtract a value from your
sum.");
                System.out.println("3. Get your sum.");
                System.out.println("4. Exit client");

                selection = scanner.nextInt();
                //user prompt 4, quit the client
                if (selection == 4) {
                    System.out.println("Client side quitting. The remote
variable server is still running. ");
                    System.exit(0);
                }
                //1. set the id, request, input number
                if (selection == 1) {
                    System.out.println("Enter value to add: ");
                    number = scanner.nextInt();
                    System.out.println("Enter your ID: ");
                    id = scanner.nextInt();
                    while(id > 999 || id < 0) {
                        System.out.println("Invalid ID. Please try
again.");
                        id = scanner.nextInt();
                    }
                    clientRequest = "add";
                } else if (selection == 2) {
                    System.out.println("Enter value to subtract: ");
                    number = scanner.nextInt();
                    System.out.println("Enter your ID: ");
                    id = scanner.nextInt();
                    while(id > 999 || id < 0) {
                        System.out.println("Invalid ID. Please try
again.");
                        id = scanner.nextInt();
                    }
                    clientRequest = "subtract";
                } else if (selection == 3) {
                    System.out.println("Enter your ID: ");
                    id = scanner.nextInt();
```

```
                    while(id > 999 || id < 0) {
                        System.out.println("Invalid ID. Please try
again.");
                        id = scanner.nextInt();
                    }
                    clientRequest = "get";
                }else{
                    //if the selection is not found, re-type the
selection
                    System.out.println("Please enter a valid option.");
                    continue;
                }
                //send out the request with id, clientRequest, number
set
                int result = proxy.sendRequestToServer(id,
clientRequest, number);
                System.out.println("The result is " + result);
            }

        }catch (SocketException e) {System.out.println("Socket
Exception: " + e.getMessage());
        }catch (IOException e){System.out.println("IO Exception: " +
e.getMessage());
        }finally {if(proxy.getSocket()!= null)
proxy.getSocket().close();}
    }



}
//use a proxy design to encapsulate the communication code
class RemoteVariableProxy{
    private int serverPort;
    private InetAddress aHost;
    private DatagramSocket aSocket;

    //constructor to set the serverPort, address of host, and
initialize the socket
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
    public RemoteVariableProxy(int serverPort, String aHost) throws UnknownHostException, SocketException {
        this.serverPort = serverPort;
        this.aHost = InetAddress.getByName(aHost);
        this.aSocket = new DatagramSocket();
    }
    public DatagramSocket getSocket(){
        return aSocket;
    }

    //By using ID, clientRequest(add, subtract, get), and number input to request server
    public int sendRequestToServer(int id, String clientRequest, int number) {
        try{
            //build up message
            String clientRequestValue = id+","+clientRequest+","+number;

            //get the integer byte that provided by client
            byte [] m = clientRequestValue.getBytes();
            //get the request packet and send
            DatagramPacket request = new DatagramPacket(m, m.length, aHost, serverPort);
            aSocket.send(request);

            byte[] buffer = new byte[1000];
            //create a reply packet to receive server's message.
            DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
            aSocket.receive(reply);
            //return to the integer result of the reply packet
            return Integer.parseInt(new String(reply.getData(), 0 , reply.getLength()));
        }catch (IOException e){
            //Error
            System.out.println(e.getMessage());
            return -1;
        }
    }
}
```

2. Project2Task3Server

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.util.Map;
import java.util.Scanner;
import java.util.TreeMap;

public class RemoteVariableServerUDP {
    //create the request handler
    private static RemoteVariableRequestHandler
remoteVariableRequestHandler = new RemoteVariableRequestHandler();

    public static void main(String args[]){
        System.out.println("Server started");
        // Initialize a DatagramSocket for UDP communication
        DatagramSocket aSocket = null;
        // Initialize a buffer to store incoming data
        byte[] buffer = new byte[1000];
        Scanner scanner = new Scanner(System.in);

        try{
            //id for client's user ID
            //clientNumber for client's input value
            //clientRequest for client's command
            int id;
            String clientRequest;
            int clientNumber;
            //Prompt the listening port
            System.out.println("Enter the port number to listen on: ");
            int serverPort = scanner.nextInt();
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
          // Set a socket to port 6789 to listen for packets from
client
          aSocket = new DatagramSocket(serverPort);

          // Set a DatagramPacket to receive data from client
          DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
          //Let server keep running
          while(true){
              //wait until client send package to server
              aSocket.receive(request);

              //extract the message from client
              String clientMessage = new String(request.getData(), 0,
request.getLength());
              String [] userInputs = clientMessage.split(",");

              //get the info from client
              id = Integer.parseInt(userInputs[0]);
              clientRequest = userInputs[1];
              clientNumber = Integer.parseInt(userInputs[2]);

              //do the required action that send by client
              int result =
remoteVariableRequestHandler.serverActionOnRequest(id, clientRequest,
clientNumber);
              //client info details
              System.out.println("Server received client's id: "+
id);
              System.out.println("Server received client's request:
"+ clientRequest);
              System.out.println("Server received client's number: "+
clientNumber);
              System.out.println("Server response: " + result);

              //set result to be the response
              String response = String.valueOf(result);
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

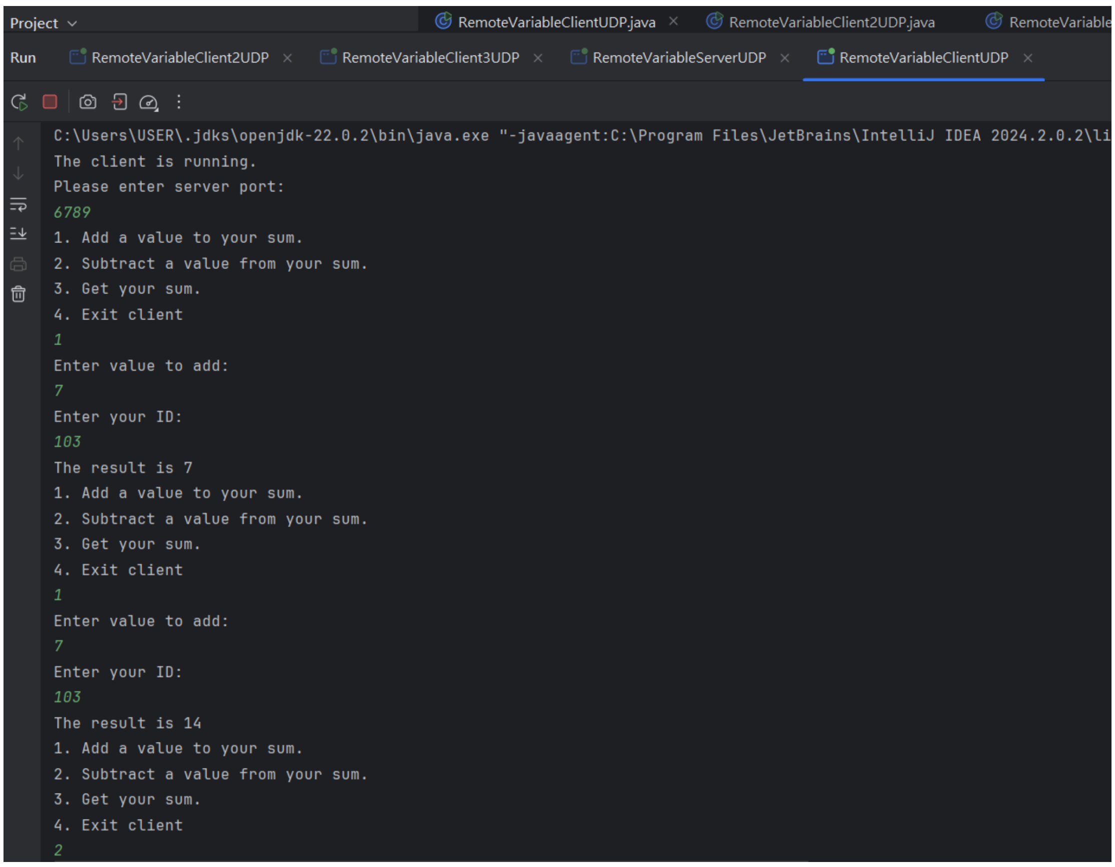
```
                //Another approach would be to only transmit byte
arrays containing String data.
                byte[] responseData = response.getBytes();
                //packet the reply and send back to client
                DatagramPacket reply = new DatagramPacket(
                        responseData, responseData.length,
request.getAddress(), request.getPort()
                );
                aSocket.send(reply);

            }
        }catch (SocketException e){System.out.println("Socket: " +
e.getMessage());
        }catch (IOException e) {System.out.println("IO: " +
e.getMessage());
        }finally {if(aSocket != null) aSocket.close();}
    }
}
class RemoteVariableRequestHandler{
    //TreeMap for each user's input
    private Map<Integer, Integer> userNumbersMap = new
TreeMap<Integer, Integer>();

    //do the action that request by client
    public int serverActionOnRequest(int id, String clientRequest,
int number){
        //check if the user is absent or not. set new user if the
users has no record
        userNumbersMap.putIfAbsent(id, 0);
        if(clientRequest.equals("add")){
            //add the value
            userNumbersMap.put(id, userNumbersMap.get(id) + number);
        } else if (clientRequest.equals("subtract")) {
            //subtract the value
            userNumbersMap.put(id, userNumbersMap.get(id) - number);
        } else if (clientRequest.equals("get")) {
            //get the value
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
            userNumbersMap.get(id);
        }else{
            //Error if the ambiguous variables
            System.out.println("Unknown request");
            return -1;
        }
        return userNumbersMap.get(id);
    }
}
```

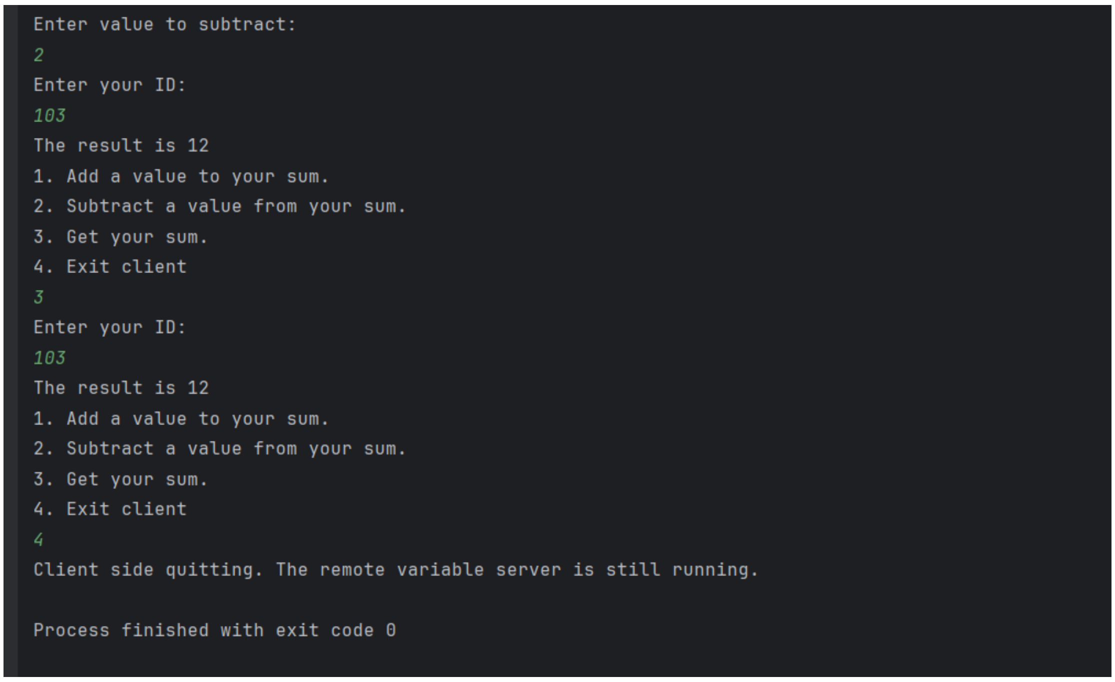
3. Project2Task3ClientConsole

Client 1:



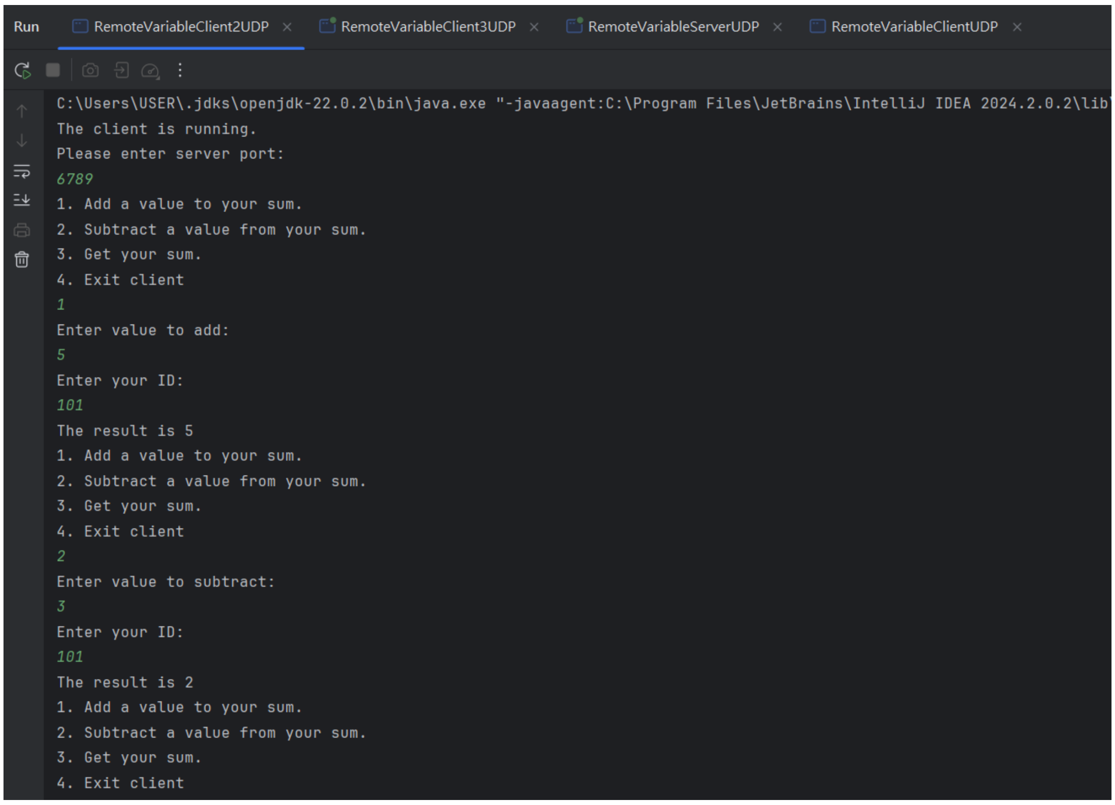
```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2\li
The client is running.
Please enter server port:
6789
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
7
Enter your ID:
103
The result is 7
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
7
Enter your ID:
103
The result is 14
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
2
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



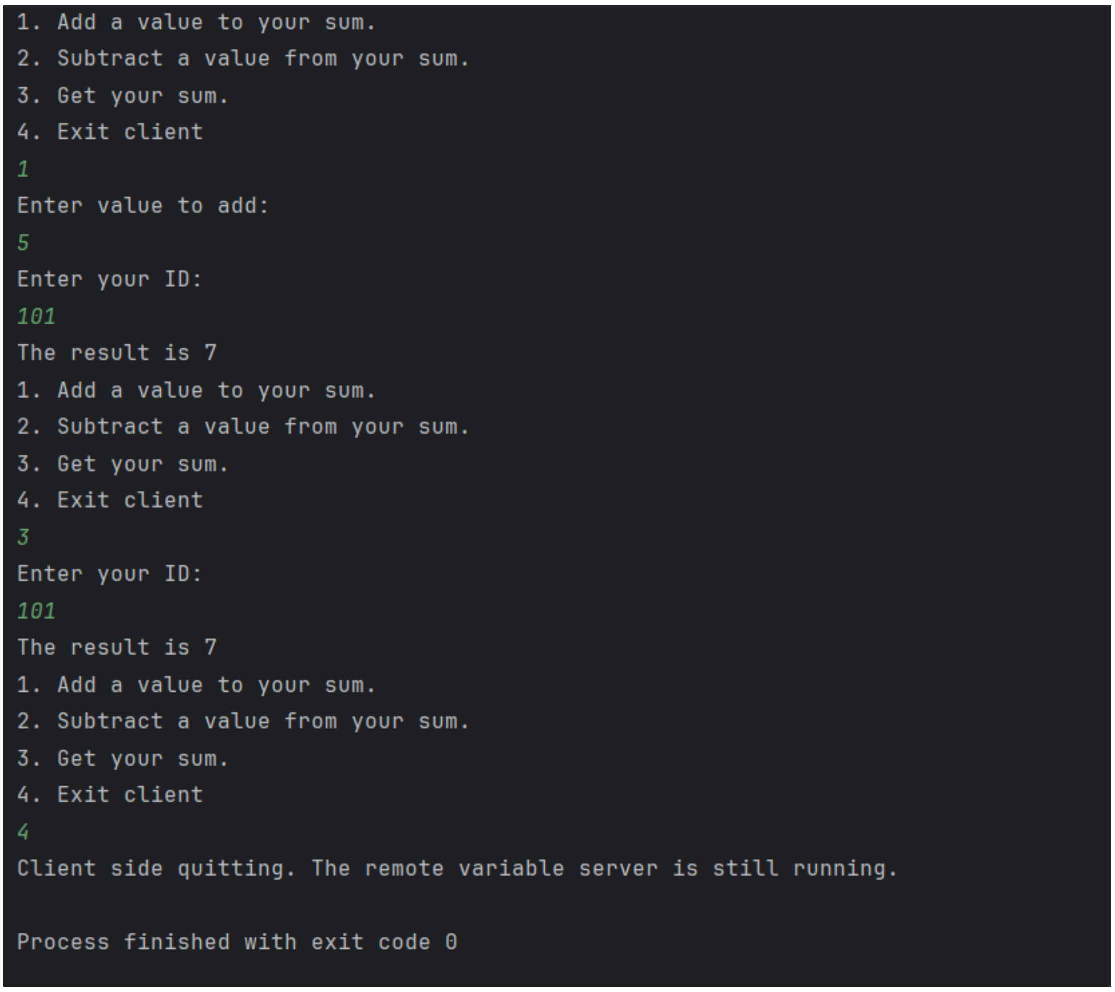
```
Enter value to subtract:
2
Enter your ID:
103
The result is 12
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
103
The result is 12
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Client 2:



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2\lib
The client is running.
Please enter server port:
6789
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
5
Enter your ID:
101
The result is 5
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
2
Enter value to subtract:
3
Enter your ID:
101
The result is 2
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
```

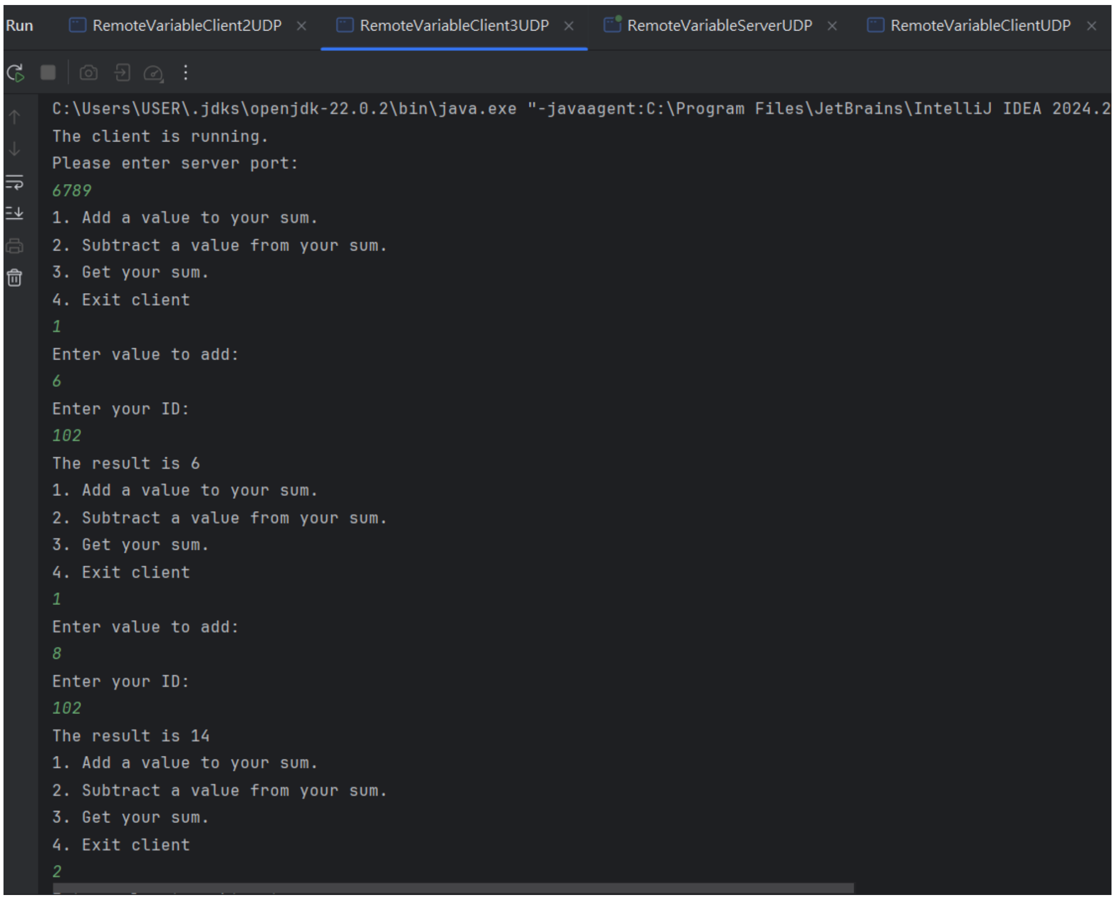
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



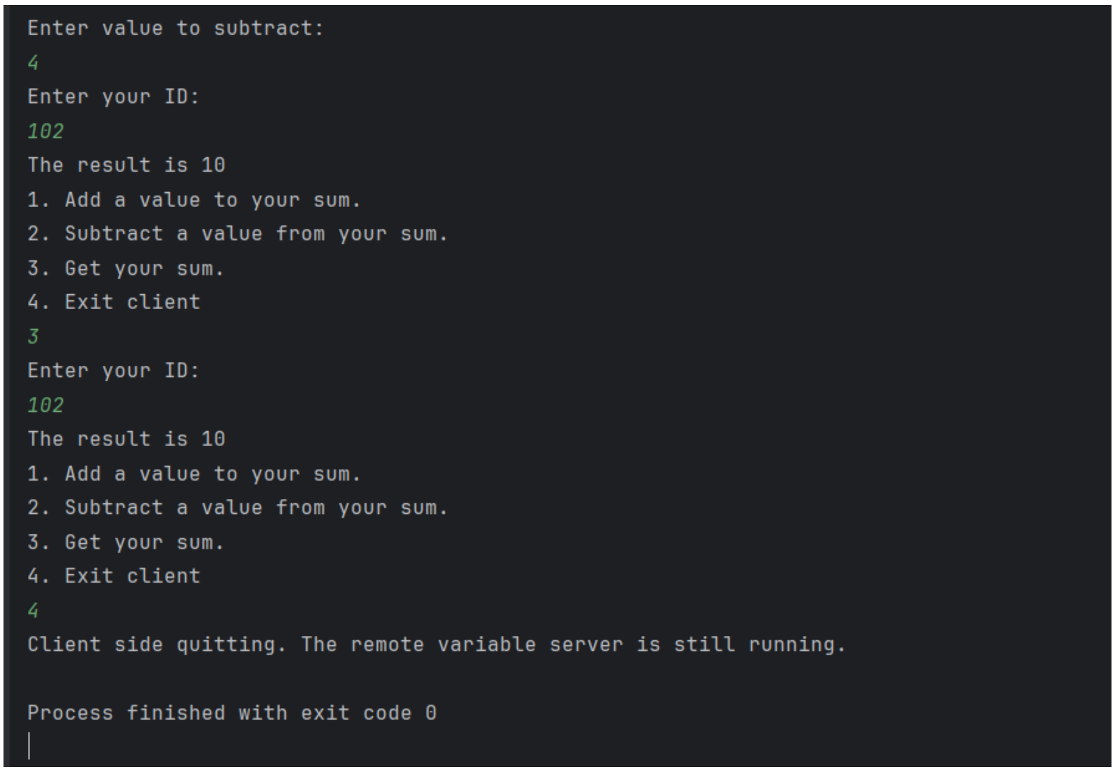
```
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
5
Enter your ID:
101
The result is 7
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
101
The result is 7
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Client 3:



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2
The client is running.
Please enter server port:
6789
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
6
Enter your ID:
102
The result is 6
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
8
Enter your ID:
102
The result is 14
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
2
```

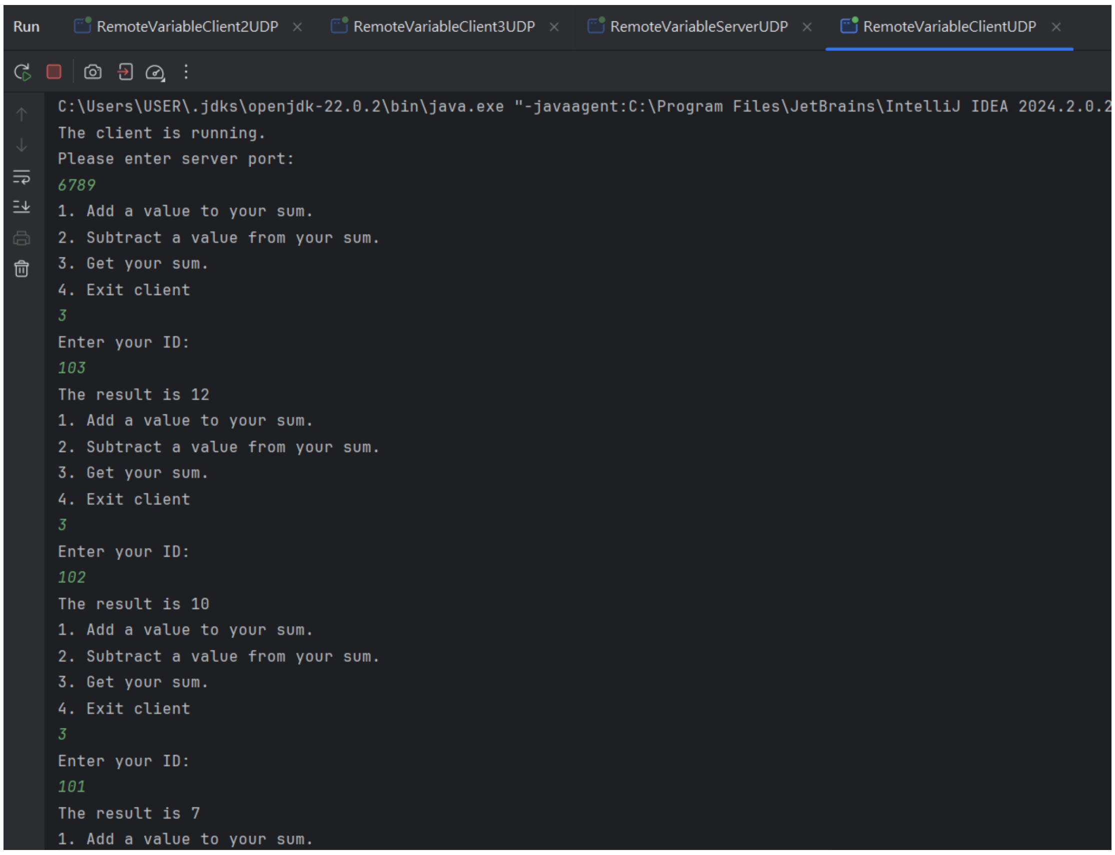


```
Enter value to subtract:
4
Enter your ID:
102
The result is 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
102
The result is 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

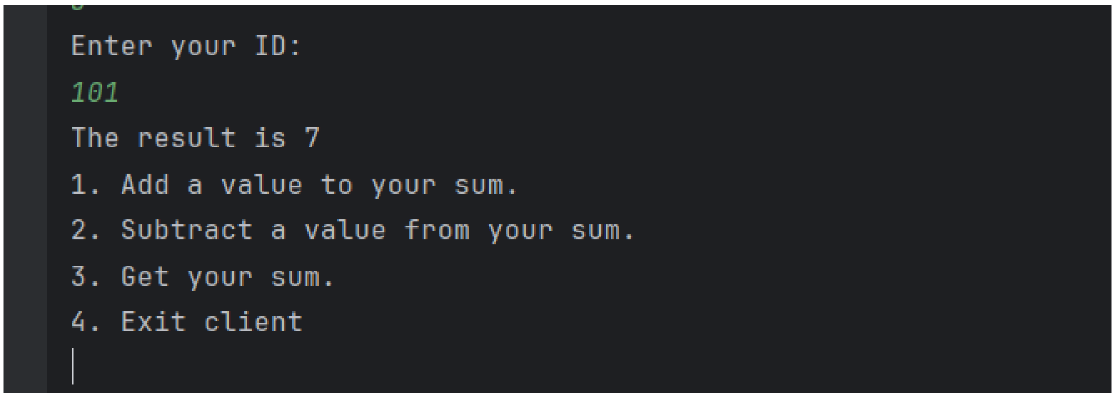
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Re-run the client:

Client 1:



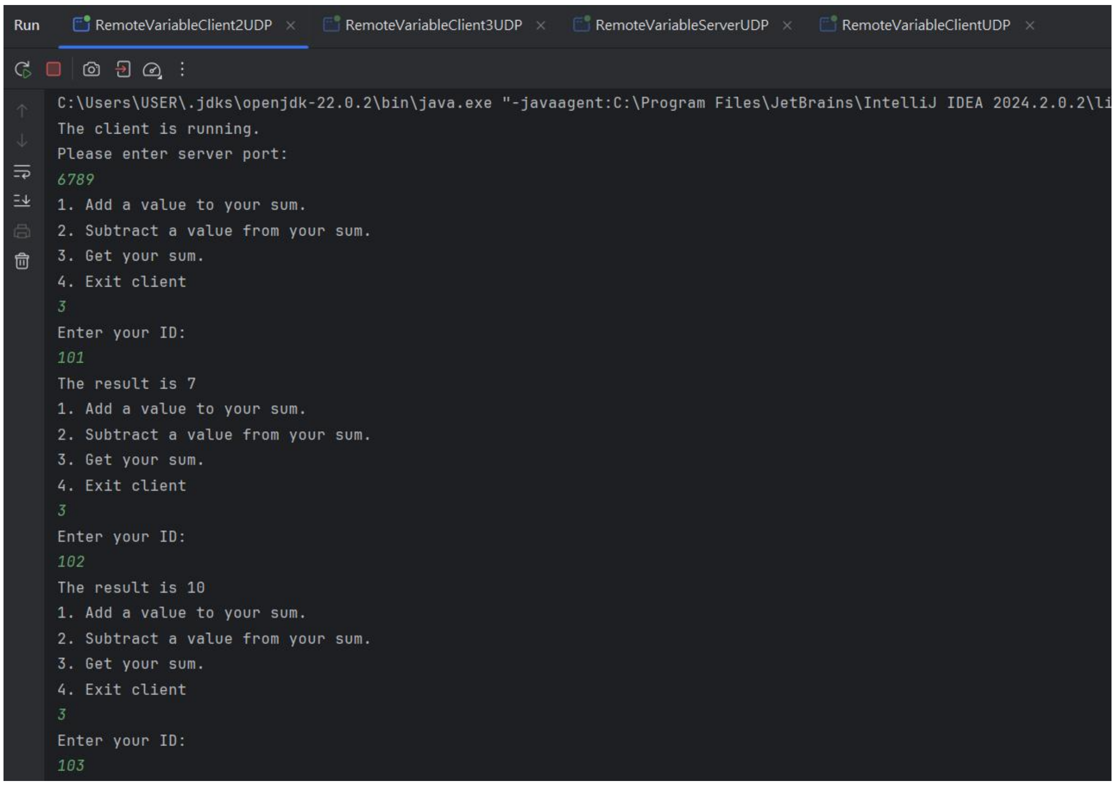
```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2
The client is running.
Please enter server port:
6789
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
103
The result is 12
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
102
The result is 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
101
The result is 7
1. Add a value to your sum.
```



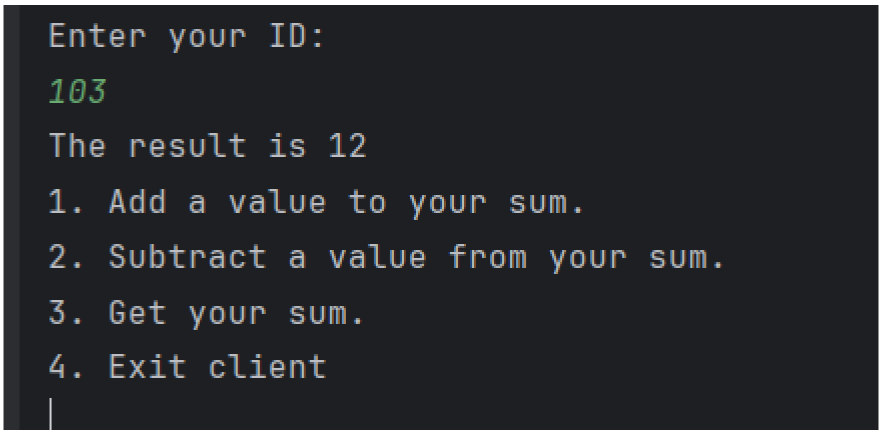
```
Enter your ID:
101
The result is 7
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Client 2:



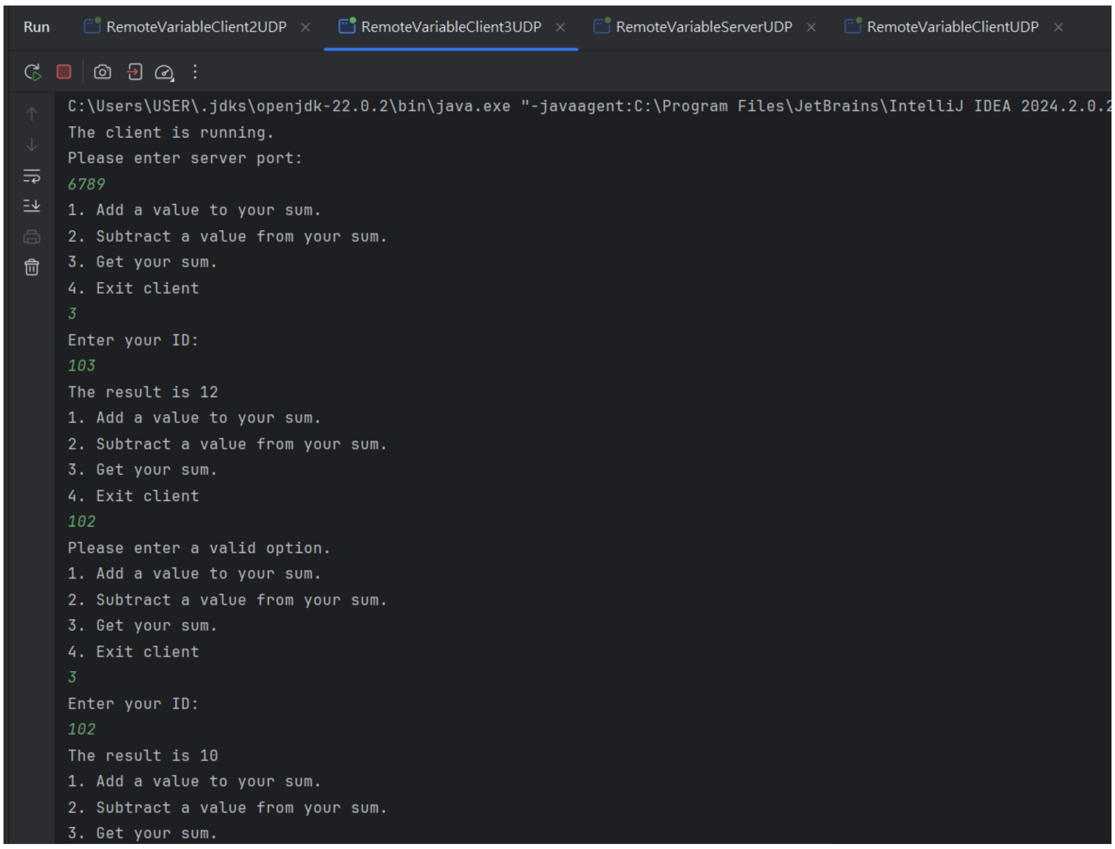
```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2\li
The client is running.
Please enter server port:
6789
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
101
The result is 7
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
102
The result is 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
103
```



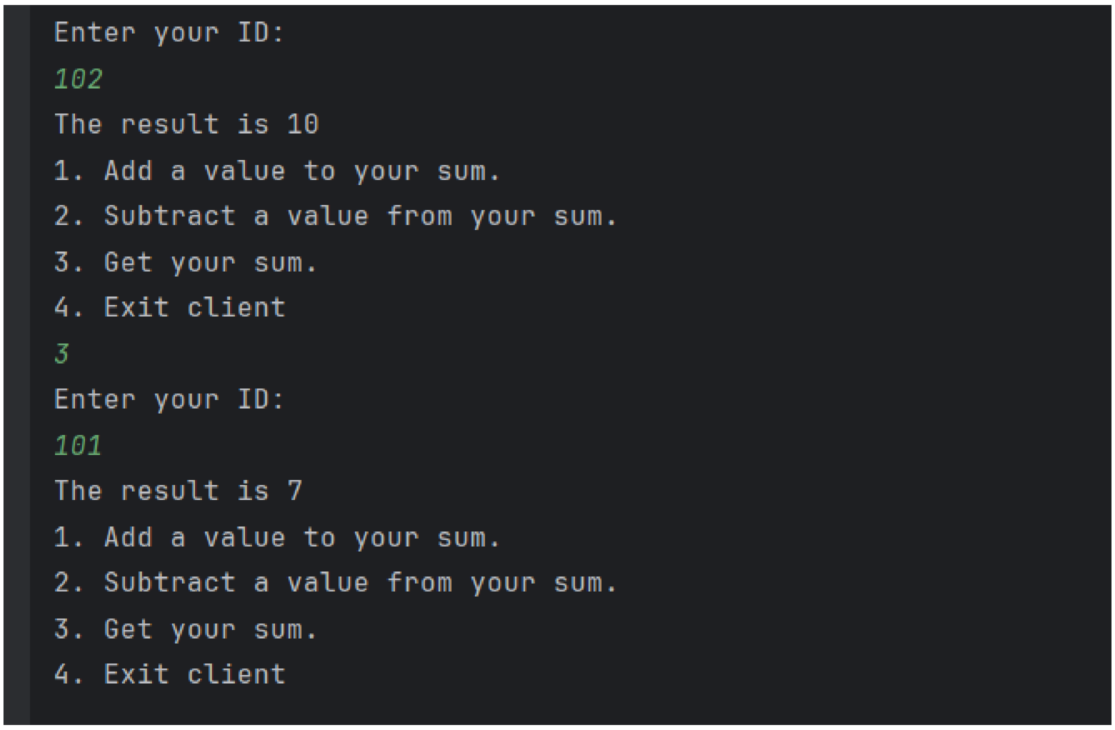
```
Enter your ID:
103
The result is 12
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Client 3:



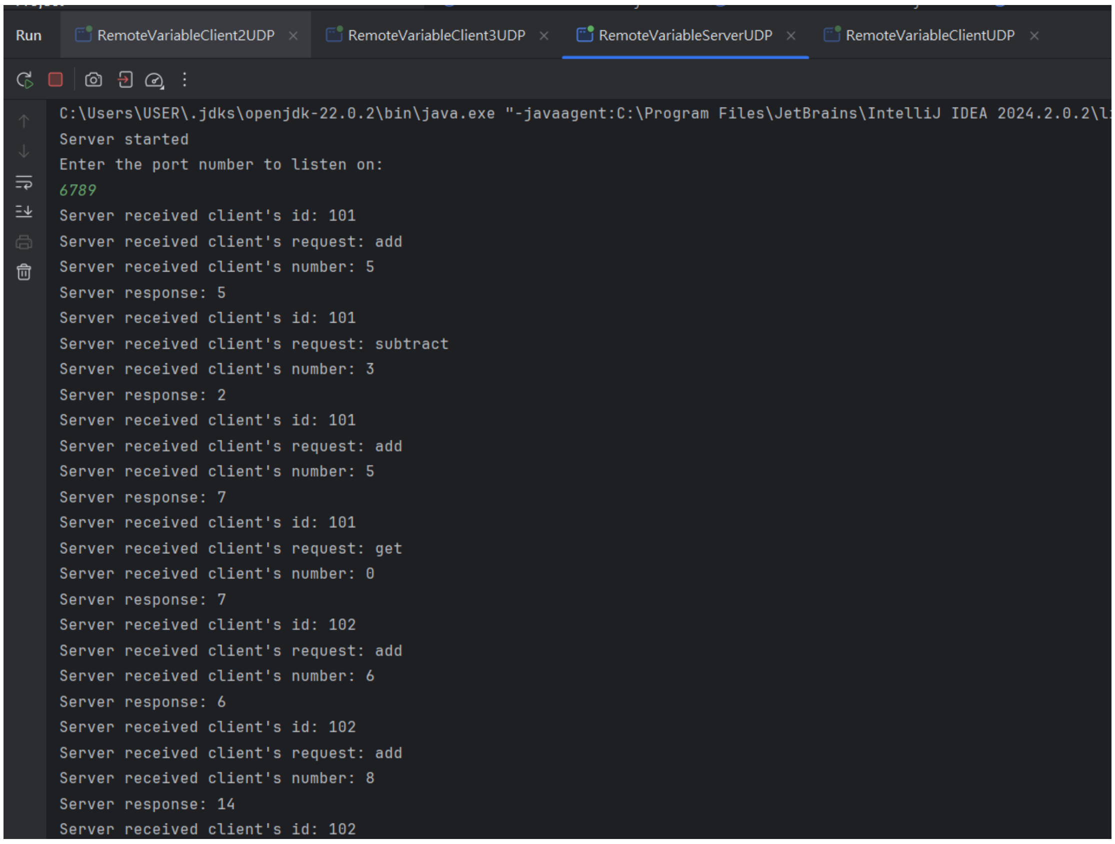
```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2
The client is running.
Please enter server port:
6789
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
103
The result is 12
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
102
Please enter a valid option.
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
102
The result is 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
```



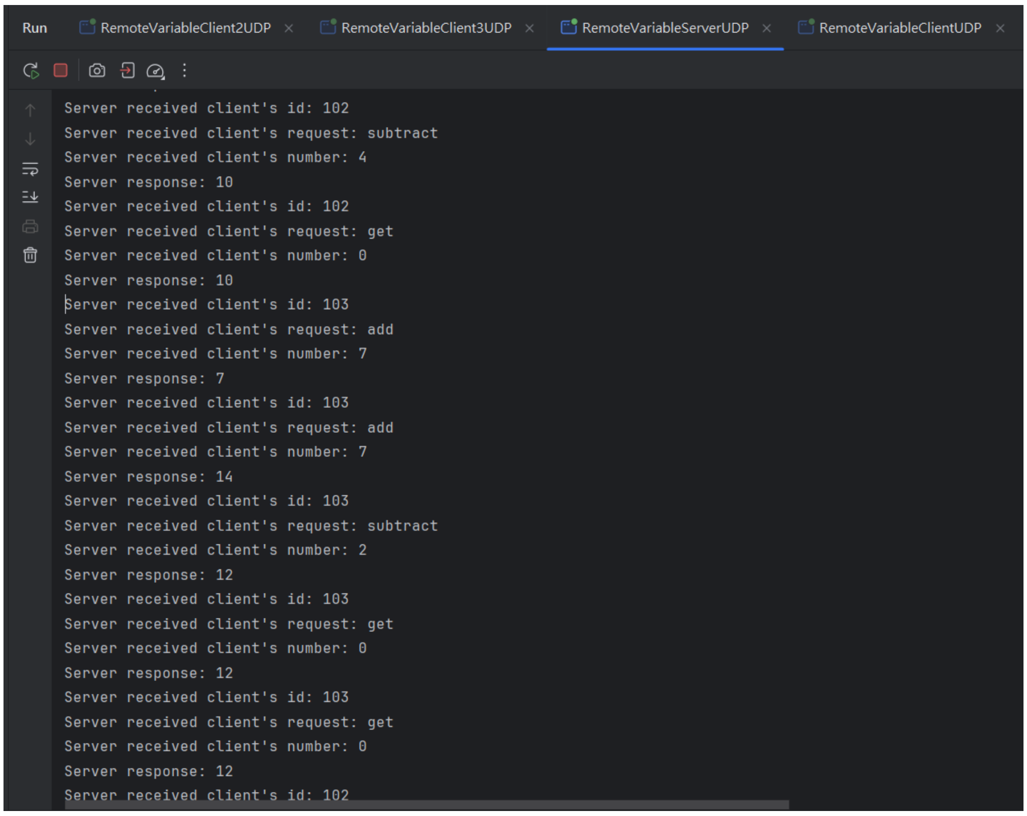
```
Enter your ID:
102
The result is 10
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
101
The result is 7
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

4. Project2Task3ServerConsole

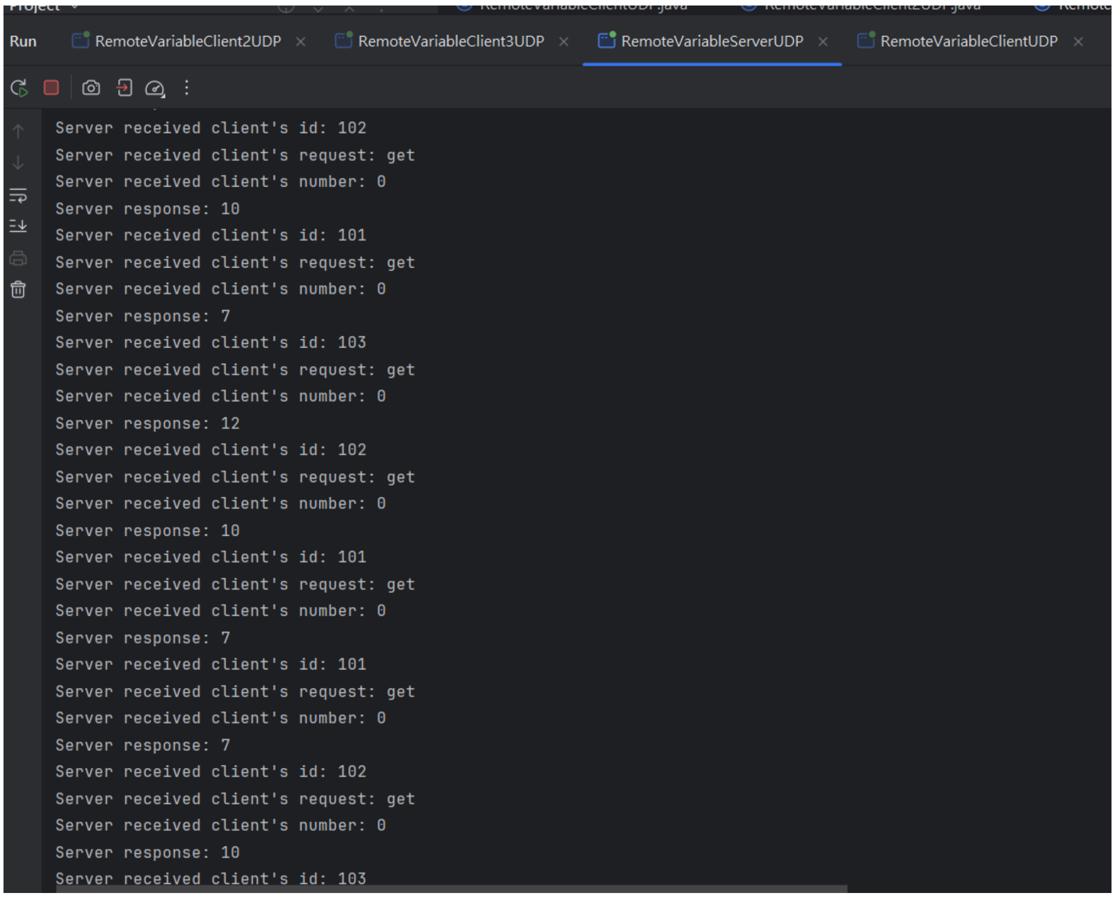


```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2\l
Server started
Enter the port number to listen on:
6789
Server received client's id: 101
Server received client's request: add
Server received client's number: 5
Server response: 5
Server received client's id: 101
Server received client's request: subtract
Server received client's number: 3
Server response: 2
Server received client's id: 101
Server received client's request: add
Server received client's number: 5
Server response: 7
Server received client's id: 101
Server received client's request: get
Server received client's number: 0
Server response: 7
Server received client's id: 102
Server received client's request: add
Server received client's number: 6
Server response: 6
Server received client's id: 102
Server received client's request: add
Server received client's number: 8
Server response: 14
Server received client's id: 102
```

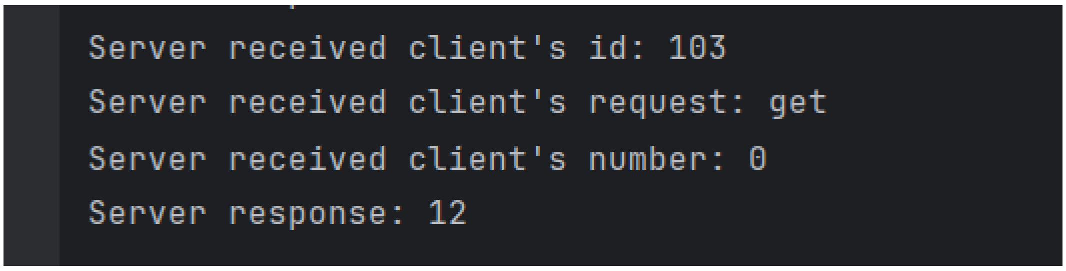


```
Server received client's id: 102
Server received client's request: subtract
Server received client's number: 4
Server response: 10
Server received client's id: 102
Server received client's request: get
Server received client's number: 0
Server response: 10
Server received client's id: 103
Server received client's request: add
Server received client's number: 7
Server response: 7
Server received client's id: 103
Server received client's request: add
Server received client's number: 7
Server response: 14
Server received client's id: 103
Server received client's request: subtract
Server received client's number: 2
Server response: 12
Server received client's id: 103
Server received client's request: get
Server received client's number: 0
Server response: 12
Server received client's id: 103
Server received client's request: get
Server received client's number: 0
Server response: 12
Server received client's id: 102
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



```
Server received client's id: 102
Server received client's request: get
Server received client's number: 0
Server response: 10
Server received client's id: 101
Server received client's request: get
Server received client's number: 0
Server response: 7
Server received client's id: 103
Server received client's request: get
Server received client's number: 0
Server response: 12
Server received client's id: 102
Server received client's request: get
Server received client's number: 0
Server response: 10
Server received client's id: 101
Server received client's request: get
Server received client's number: 0
Server response: 7
Server received client's id: 101
Server received client's request: get
Server received client's number: 0
Server response: 7
Server received client's id: 102
Server received client's request: get
Server received client's number: 0
Server response: 10
Server received client's id: 103
```



```
Server received client's id: 103
Server received client's request: get
Server received client's number: 0
Server response: 12
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Task 4:

1. Project2Task4Client

```
package ds;
import java.io.IOException;
import java.net.*;
import java.util.*;
import com.google.gson.*;


public class NeuralNetworkClient {
    private static NeuralNetworkProxy proxy;
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String args[]){
        //client start
        System.out.println("The client is running.");
        try {
            // Set the server address and port
            System.out.println("Please enter server port: ");
            int serverPort = scanner.nextInt();
            //use proxy to set up the serverPort and localhost
            proxy = new NeuralNetworkProxy(serverPort, "localhost");
            while (true) {
                // let user prompt the choice (get it from
NeuralNetwork.java)
                int userSelection = menu();
                //check if user prompt is out of the menu selection
                if(userSelection < 0 || userSelection > 5){
                    System.out.println("Please enter a number between 0
and 5");
                    continue;
                }
                //Create Json request
                JsonObject request = new JsonObject();
                // Options:
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
            // 0. Display the current truth table.
            // 1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
            // 2. Perform a single training step.
            // 3. Perform n training steps. 10000 is a typical value for n.
            // 4. Test with a pair of inputs.
            // 5. Exit program.
            if(userSelection == 0){
                //set the request command into request
                request.addProperty("request", "getCurrentRange");
            } else if (userSelection == 1) {
                //input the values
                System.out.println("Enter the four results of a 4 by 2 truth table. Each value should be 0 or 1.");
                Double val1 = scanner.nextDouble();
                Double val2 = scanner.nextDouble();
                Double val3 = scanner.nextDouble();
                Double val4 = scanner.nextDouble();
                //set the request command into request
                request.addProperty("request", "setCurrentRange");
                request.addProperty("val1", val1);
                request.addProperty("val2", val2);
                request.addProperty("val3", val3);
                request.addProperty("val4", val4);
            }else if(userSelection == 2){
                //set the request command into request
                request.addProperty("request", "train");
                //One-step training
                request.addProperty("iterations", 1);
            }else if(userSelection == 3){
                //prompt the number of steps for training
                System.out.println("Enter the number of training sets.");
                int steps = scanner.nextInt();
                while(steps < 0 || steps > 10000){
                    System.out.println("Please enter a number between
0 and 10000");
                    steps = scanner.nextInt();
                }
                //set the request command into request
                request.addProperty("request", "train");
                request.addProperty("iterations", steps);
            }else if(userSelection == 4){
                //prompt the double pairs
                System.out.println("Enter a pair of doubles from a
row of the truth table. These are domain values.");
                double testVal1 = scanner.nextDouble();
                double testVal2 = scanner.nextDouble();
                //set the request command into request
                request.addProperty("request", "test");
                request.addProperty("val1", testVal1);
                request.addProperty("val2", testVal2);
            }else if(userSelection == 5){
                //Quit the client
                System.out.println("Client quit...");
                System.exit(0);
            }
            //get the response from server
            String response = proxy.sendJsonRequest(request);
            JsonObject jsonResponse =
JsonParser.parseString(response).getAsJsonObject();

            //get the response status by JsonObject
            String status =
jsonResponse.get("status").getAsString();
            System.out.println("Server response status: "+ status);
            //Use userSelection to determine which result should
client get
            //Once the status is "Ok" reply, get the response
            if(userSelection == 0){
                if(status.equals("OK")){
                    //get the values from server
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
                    double val1Result =
jsonResponse.get("val1").getAsDouble();
                    double val2Result =
jsonResponse.get("val2").getAsDouble();
                    double val3Result =
jsonResponse.get("val3").getAsDouble();
                    double val4Result =
jsonResponse.get("val4").getAsDouble();
                    //get the response value from Server
                    //Based on the fromula on Neural Network, print
out in this way
                    System.out.println("Working with the following
truth table");
                    System.out.println("0.0  0.0  " + val1Result);
                    System.out.println("0.0  1.0  " + val2Result);
                    System.out.println("1.0  0.0  " + val3Result);
                    System.out.println("1.0  1.0  " + val4Result);
                 }
              }else if(userSelection == 1){
                 if(status.equals("OK")){
                    //print out if Server successfully complete the
process
                    System.out.println("Range is successfully set");
                 }
              }else if(userSelection == 2){
                 if(status.equals("OK")){
                    //get the response value from Server
                    double totalError =
jsonResponse.get("val1").getAsDouble();
                    System.out.println("After this step the error
is :"+totalError);
                 }
              }else if(userSelection == 3){
                 if(status.equals("OK")){
                    //get the user prompt for training
                    String steps =
request.get("iterations").getAsString();
```

```
                        //get the response value from Server
                        double totalError =
jsonResponse.get("val1").getAsDouble();
                        System.out.println("After "+ steps+" step the
error is :"+totalError);
                    }
                }else if(userSelection == 4){
                    if(status.equals("OK")){
                        //get the response value from Server
                        double rangeValue =
jsonResponse.get("val1").getAsDouble();
                        System.out.println("The range value is
approximately "+ rangeValue);
                    }
                }
            }




        }catch (SocketException e) {System.out.println("Socket
Exception: " + e.getMessage());
        }catch (IOException e){System.out.println("IO Exception: " +
e.getMessage());
        }finally {if(proxy.getSocket()!= null)
proxy.getSocket().close();}
    }

    public static int menu() {
        System.out.println("Using a neural network to learn a truth
table.\nMain Menu");
        System.out.println("0. Display the current truth table.");
        System.out.println("1. Provide four inputs for the range of
the two input truth table and build a new neural network. To test
XOR, enter 0  1  1  0.");
        System.out.println("2. Perform a single training step.");
        System.out.println("3. Perform n training steps. 10000 is a
typical value for n.");
```

```java
        System.out.println("4. Test with a pair of inputs.");
        System.out.println("5. Exit program.");
        int selection = scanner.nextInt();
        return selection;
    }

}
//use a proxy design to encapsulate the communication code
class NeuralNetworkProxy{
    private int serverPort;
    private InetAddress aHost;
    private DatagramSocket aSocket;

    //constructor to set the serverPort, address of host, and
initialize the socket
    public NeuralNetworkProxy(int serverPort, String aHost) throws
UnknownHostException, SocketException {
        this.serverPort = serverPort;
        this.aHost = InetAddress.getByName(aHost);
        this.aSocket = new DatagramSocket();
    }
    public DatagramSocket getSocket(){
        return aSocket;
    }

    public String sendJsonRequest(JsonObject request) throws
IOException {
        String clientRequest = request.toString();
        byte[] requestJsonData = clientRequest.getBytes();
        DatagramPacket reqestJsonPacket = new
DatagramPacket(requestJsonData, requestJsonData.length, aHost,
serverPort);
        aSocket.send(reqestJsonPacket);

        byte[] responseData = new byte[1000];
        DatagramPacket responsePacket = new
DatagramPacket(responseData, responseData.length);
```

```
        aSocket.receive(responsePacket);

        return new String(responsePacket.getData(), 0,
responsePacket.getLength());

    }

}
```

2. Project2Task4Server

```
package ds;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.util.*;

public class NeuralNetworkServer {
    private static double val1, val2, val3, val4;
    private static double totalError;
    private static NeuralNetwork neuralNetwork;
    private static ArrayList<Double[][]> userTrainingSets;
    private static Random rand = new Random();

    public static void main(String args[]){
        System.out.println("Server started");
        // Initialize a DatagramSocket for UDP communication
        DatagramSocket aSocket = null;
        // Initialize a buffer to store incoming data
        byte[] buffer = new byte[1000];
        Scanner scanner = new Scanner(System.in);

        // Create an initial truth table with all 0's in the range.
        ArrayList<Double[][]> userTrainingSets = new
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
ArrayList<Double[][]>(Arrays.asList(
              new Double[][]{{0.0, 0.0}, {0.0}},
              new Double[][]{{0.0, 1.0}, {0.0}},
              new Double[][]{{1.0, 0.0}, {0.0}},
              new Double[][]{{1.0, 1.0}, {0.0}}
       ));
       //    Create a neural network suitable for working with truth
tables.
       //    There will be two inputs, 5 hidden neurons, and 1
output. All weights and biases will be random.
       //    This is the initial neural network on start up.
       NeuralNetwork neuralNetwork = new NeuralNetwork(2, 5, 1, null,
null, null, null);

       try{

           System.out.println("Enter the port number to listen on: ");
           int serverPort = scanner.nextInt();
           // Set a socket to port 6789 to listen for packets from
client
           aSocket = new DatagramSocket(serverPort);

           // Set a DatagramPacket to receive data from client
           DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
           //Let server keep running
           while(true){
               //Wait until Server receives the request from client
               aSocket.receive(request);
               //Get the client data
               String requestData = new String(request.getData(), 0,
request.getLength());
               //Create get the json data from client, processing the
request
               JsonObject jsonRequest =
JsonParser.parseString(requestData).getAsJsonObject();
               JsonObject jsonResponse =
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
serverActionOnJsonRequest(jsonRequest);
                //get the bytes of response
                byte[] responseData =
jsonResponse.toString().getBytes();
                //Packeted, send to the client
                DatagramPacket responsePacket= new
DatagramPacket(responseData, responseData.length,
request.getAddress(), request.getPort());
                aSocket.send(responsePacket);
            }
        }catch (SocketException e){System.out.println("Socket: " +
e.getMessage());
        }catch (IOException e) {System.out.println("IO: " +
e.getMessage());}
        finally {if(aSocket != null) aSocket.close();}
    }
    public static JsonObject serverActionOnJsonRequest(JsonObject
request){
        //Create response
        JsonObject response = new JsonObject();
        //get the request command
        String requestAction = request.get("request").getAsString();

        if(requestAction.equals("getCurrentRange")){
            //send back to client
            response.addProperty("request", "getCurrentRange");
            response.addProperty("status", "OK");
            response.addProperty("val1", val1);
            response.addProperty("val2", val2);
            response.addProperty("val3", val3);
            response.addProperty("val4", val4);
            //Call when server receive client's request
            System.out.println("Server Response: ");
            System.out.println(response.toString());
        } else if (requestAction.equals("setCurrentRange")) {
            //Get the values provided by client
            val1 = request.get("val1").getAsDouble();
```

```
          val2 = request.get("val2").getAsDouble();
          val3 = request.get("val3").getAsDouble();
          val4 = request.get("val4").getAsDouble();
          //Set the userTrainingsSets, provided by NeuralNetwork
          userTrainingSets = new ArrayList<Double[][]>(Arrays.asList(
            new Double[][] {{0.0, 0.0}, {val1}},
            new Double[][] {{0.0, 1.0}, {val2}},
            new Double[][] {{1.0, 0.0}, {val3}},
            new Double[][] {{1.0, 1.0}, {val4}}
          ));
          //Send back to client
          response.addProperty("request", "setCurrentRange");
          response.addProperty("status", "OK");
          response.addProperty("val1", val1);
          response.addProperty("val2", val2);
          response.addProperty("val3", val3);
          response.addProperty("val4", val4);
          System.out.println("Server Response: ");
          System.out.println(response.toString());
          // Build a new neural network with new random weights.
          neuralNetwork = new NeuralNetwork(2, 5, 1, null, null,
null, null);

       } else if (requestAction.equals("train")) {
          //code is provided by NeuralNetwork
          //get the number of training sets
          int iterations = request.get("iterations").getAsInt();
          totalError = 0.0;
          for(int i = 0; i < iterations; i++){
             // perform trainng step and display total error.
             int random_choice = rand.nextInt(4);
             // Get the two inputs
             List<Double> userTrainingInputs =
Arrays.asList(userTrainingSets.get(random_choice)[0]);
             // Get the one output (in the case of truth tables).
             List<Double> userTrainingOutputs =
Arrays.asList(userTrainingSets.get(random_choice)[1]);
```

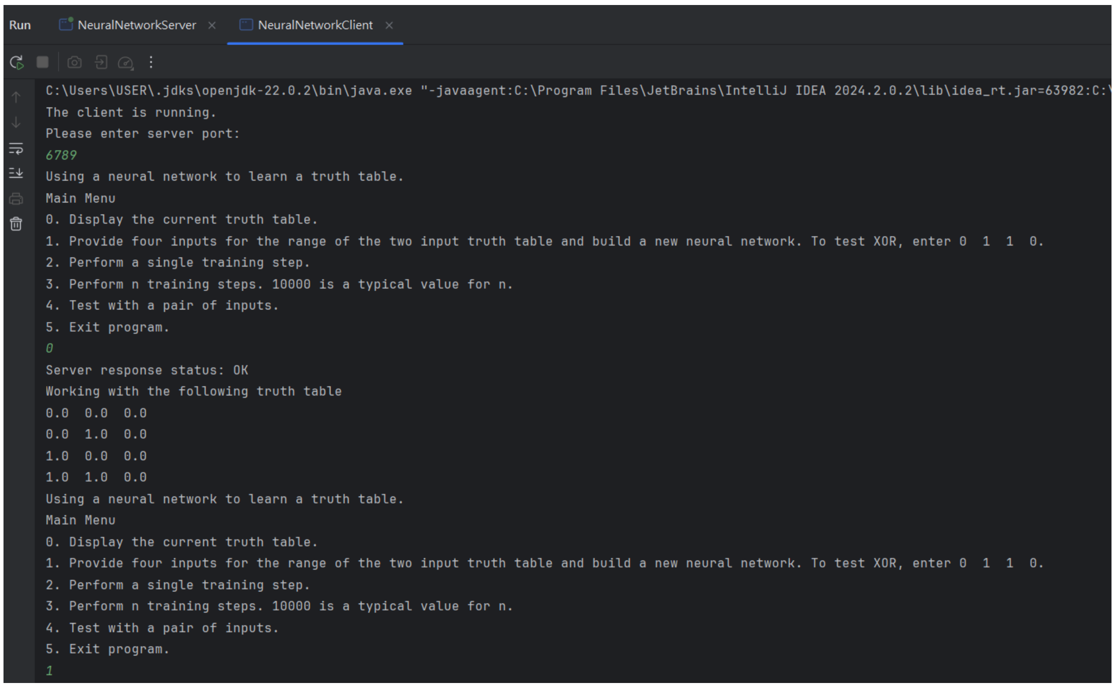
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
                // Show that row to the neural network
                neuralNetwork.train(userTrainingInputs,
userTrainingOutputs);
                // Calculate the error
                totalError =
neuralNetwork.calculateTotalError(userTrainingSets);
            }
            //Send back to client
            response.addProperty("request", "train");
            response.addProperty("status", "OK");
            response.addProperty("val1", totalError);
            System.out.println("Received Client's request: ");
            System.out.println(response.toString());

        } else if (requestAction.equals("test")) {
            //code is provied by NeuralNetwork
            // test with a pair of inputs.
            double testVal1 = request.get("val1").getAsDouble();
            double testVal2 = request.get("val2").getAsDouble();
            List<Double> testInputs = Arrays.asList(testVal1,
testVal2);
            List<Double> testResult =
neuralNetwork.feedForward(testInputs);
            response.addProperty("request", "test");
            response.addProperty("status", "OK");
            response.addProperty("val1", testResult.get(0));
            //send response to client
            System.out.println("Server Response: ");
            System.out.println(response.toString());

        }else {
            //if there is an ambiguous request, set the status to ERROR
            response.addProperty("status", "ERROR");
            System.out.println("Server Response: ");
            System.out.println(response.toString());
        }
        return response;
```

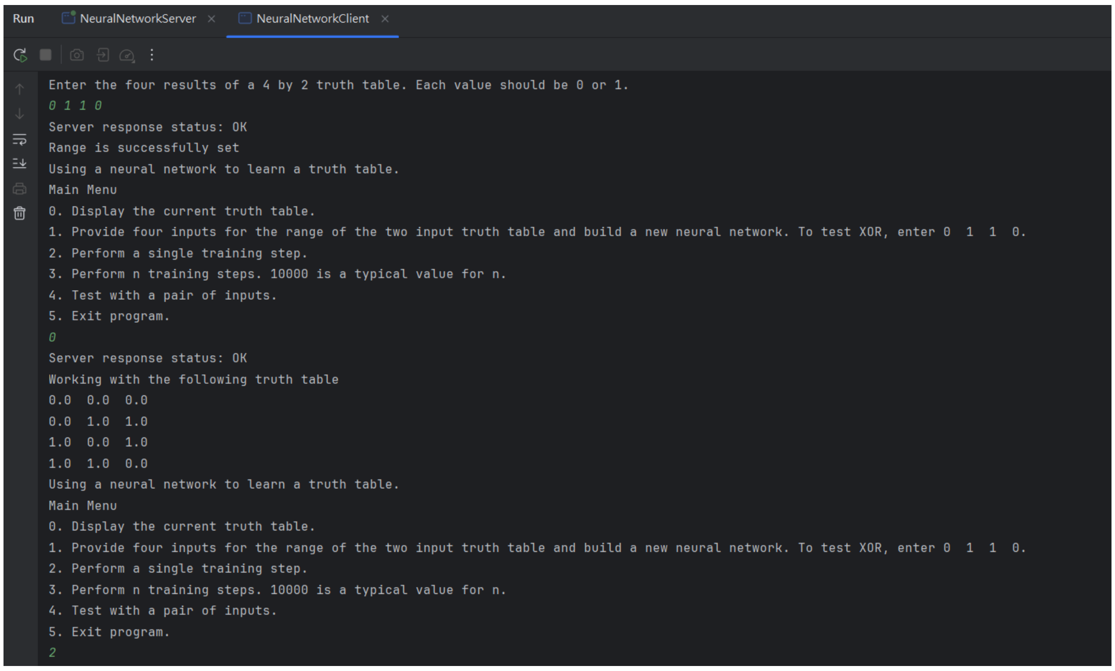
```
    }
}
```

3. Project2Task4ClientConsole

XOR 0 1 1 0:

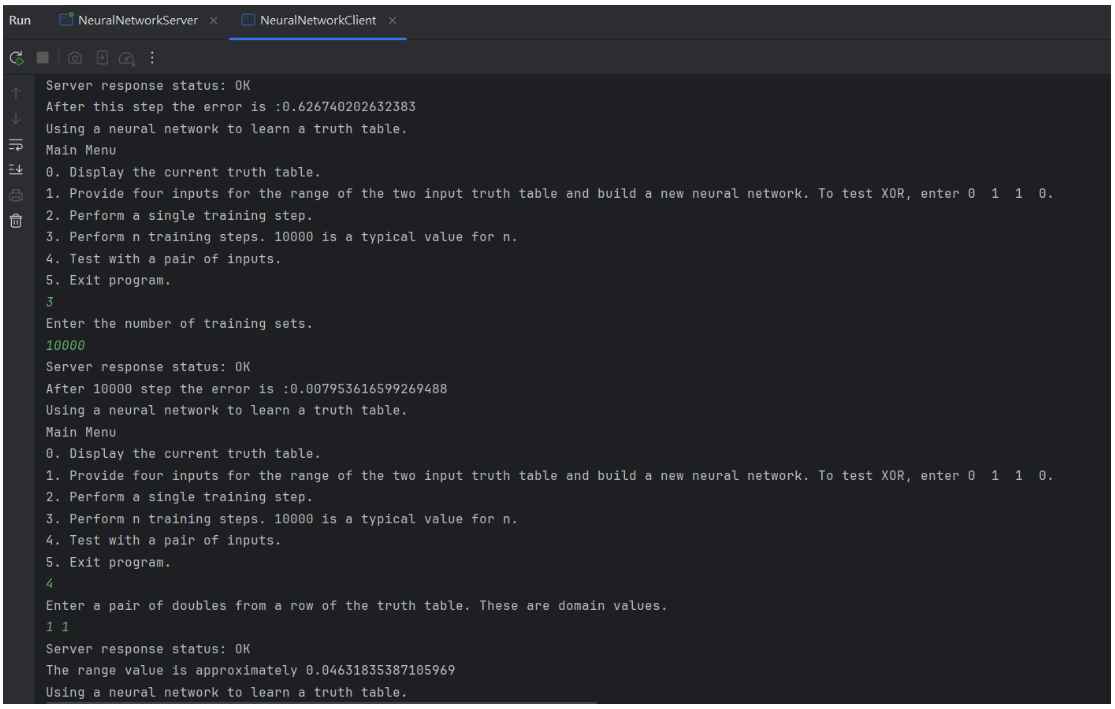

Run NeuralNetworkServer NeuralNetworkClient

```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2\lib\idea_rt.jar=63982:C:\
The client is running.
Please enter server port:
6789
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
0
Server response status: OK
Working with the following truth table
0.0  0.0  0.0
0.0  1.0  0.0
1.0  0.0  0.0
1.0  1.0  0.0
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
1
```

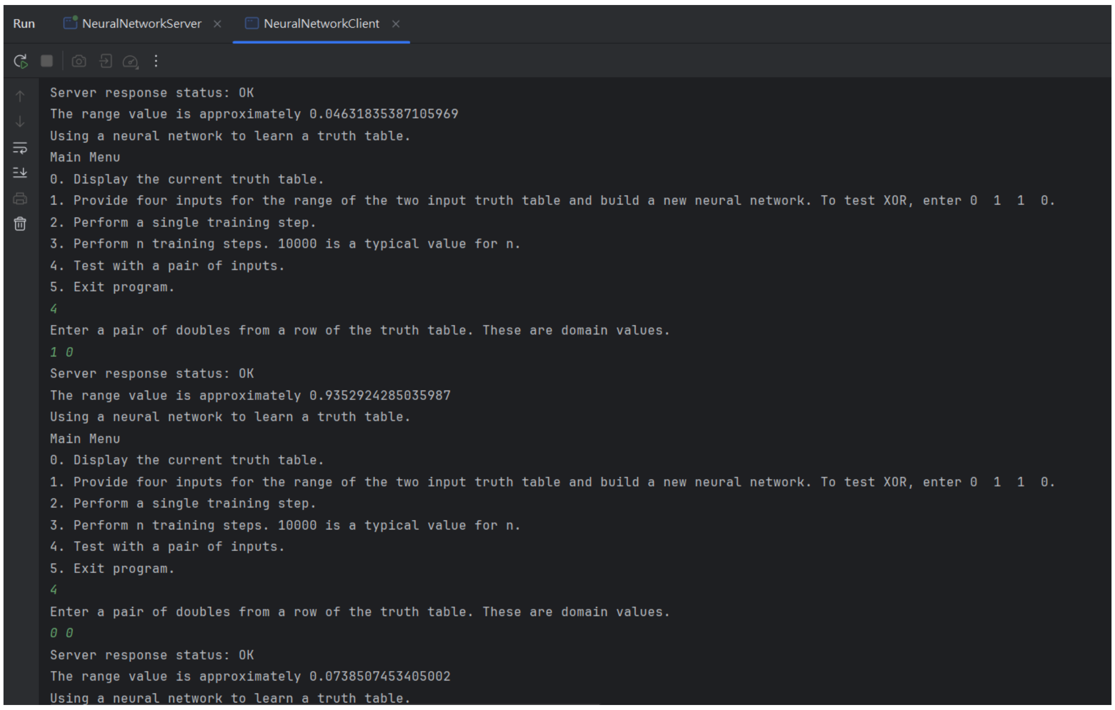

Run NeuralNetworkServer NeuralNetworkClient

```
Enter the four results of a 4 by 2 truth table. Each value should be 0 or 1.
0 1 1 0
Server response status: OK
Range is successfully set
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
0
Server response status: OK
Working with the following truth table
0.0  0.0  0.0
0.0  1.0  1.0
1.0  0.0  1.0
1.0  1.0  0.0
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
2
```

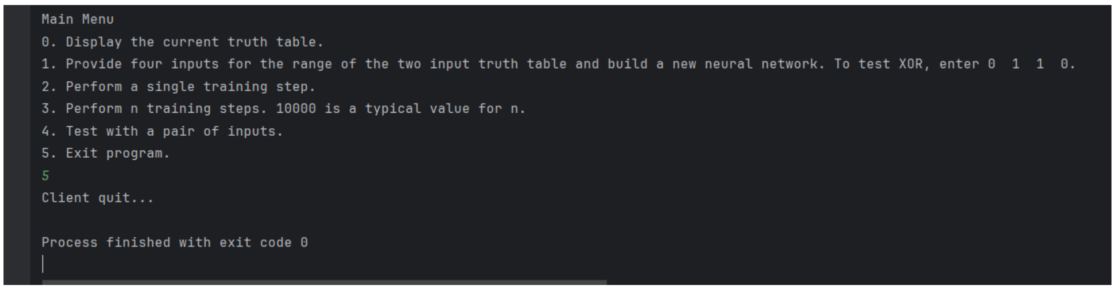
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



```
Server response status: OK
After this step the error is :0.626740202632383
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
3
Enter the number of training sets.
10000
Server response status: OK
After 10000 step the error is :0.007953616599269488
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
4
Enter a pair of doubles from a row of the truth table. These are domain values.
1 1
Server response status: OK
The range value is approximately 0.04631835387105969
Using a neural network to learn a truth table.
```



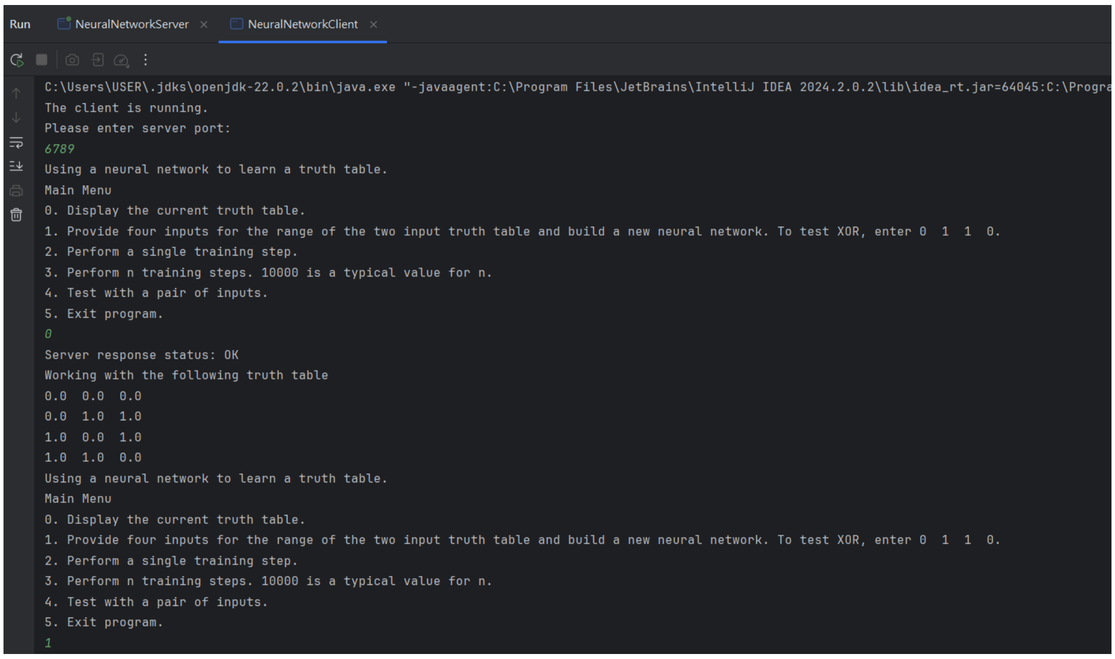
```
Server response status: OK
The range value is approximately 0.04631835387105969
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
4
Enter a pair of doubles from a row of the truth table. These are domain values.
1 0
Server response status: OK
The range value is approximately 0.9352924285035987
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
4
Enter a pair of doubles from a row of the truth table. These are domain values.
0 0
Server response status: OK
The range value is approximately 0.0738507453405002
Using a neural network to learn a truth table.
```



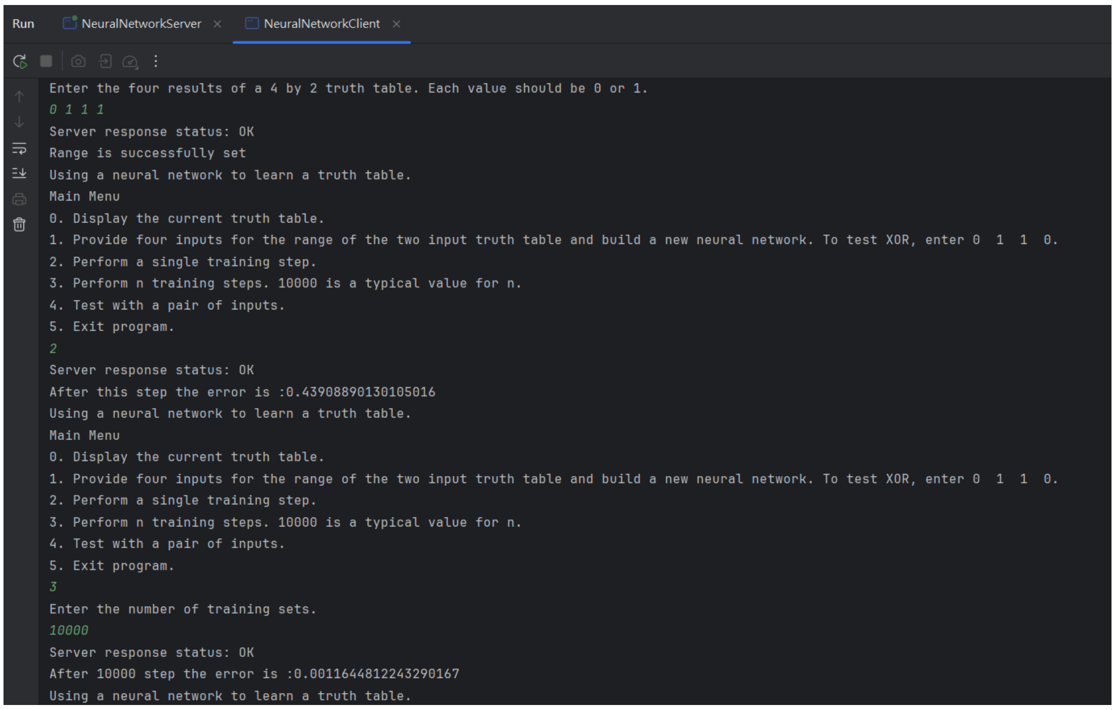
```
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
5
Client quit...

Process finished with exit code 0
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

OR: 0 1 1 1

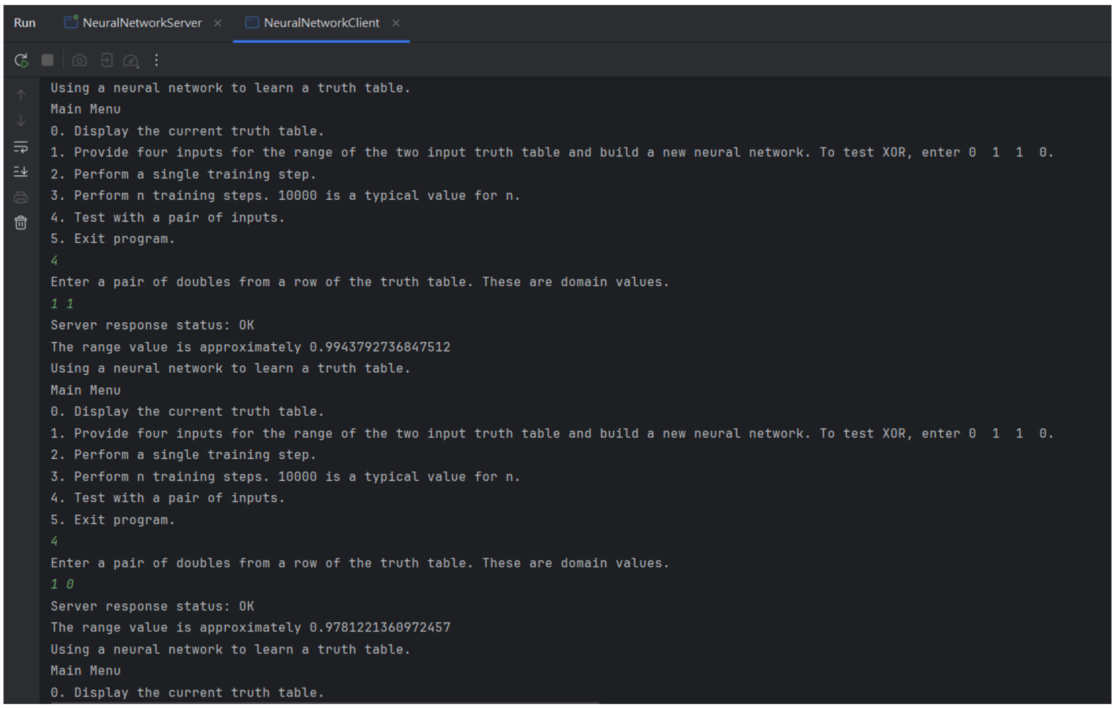


```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2\lib\idea_rt.jar=64045:C:\Progra
The client is running.
Please enter server port:
6789
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
0
Server response status: OK
Working with the following truth table
0.0  0.0  0.0
0.0  1.0  1.0
1.0  0.0  1.0
1.0  1.0  0.0
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
1
```

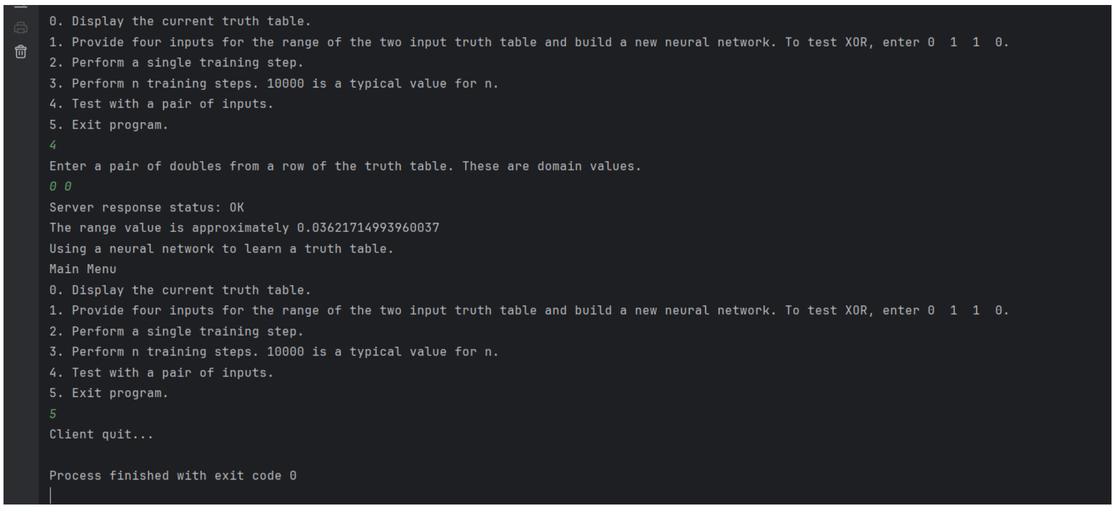


```
Enter the four results of a 4 by 2 truth table. Each value should be 0 or 1.
0 1 1 1
Server response status: OK
Range is successfully set
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
2
Server response status: OK
After this step the error is :0.43908890130105016
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
3
Enter the number of training sets.
10000
Server response status: OK
After 10000 step the error is :0.0011644812243290167
Using a neural network to learn a truth table.
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



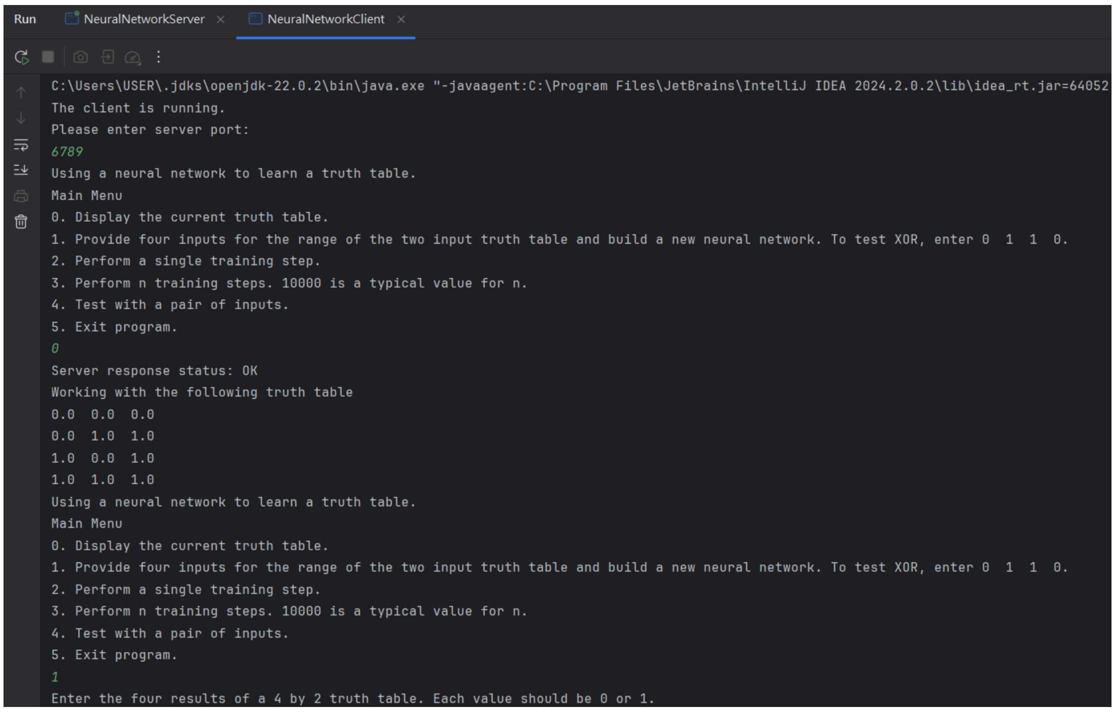
```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
4
Enter a pair of doubles from a row of the truth table. These are domain values.
1 1
Server response status: OK
The range value is approximately 0.9943792736847512
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
4
Enter a pair of doubles from a row of the truth table. These are domain values.
1 0
Server response status: OK
The range value is approximately 0.9781221360972457
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
```



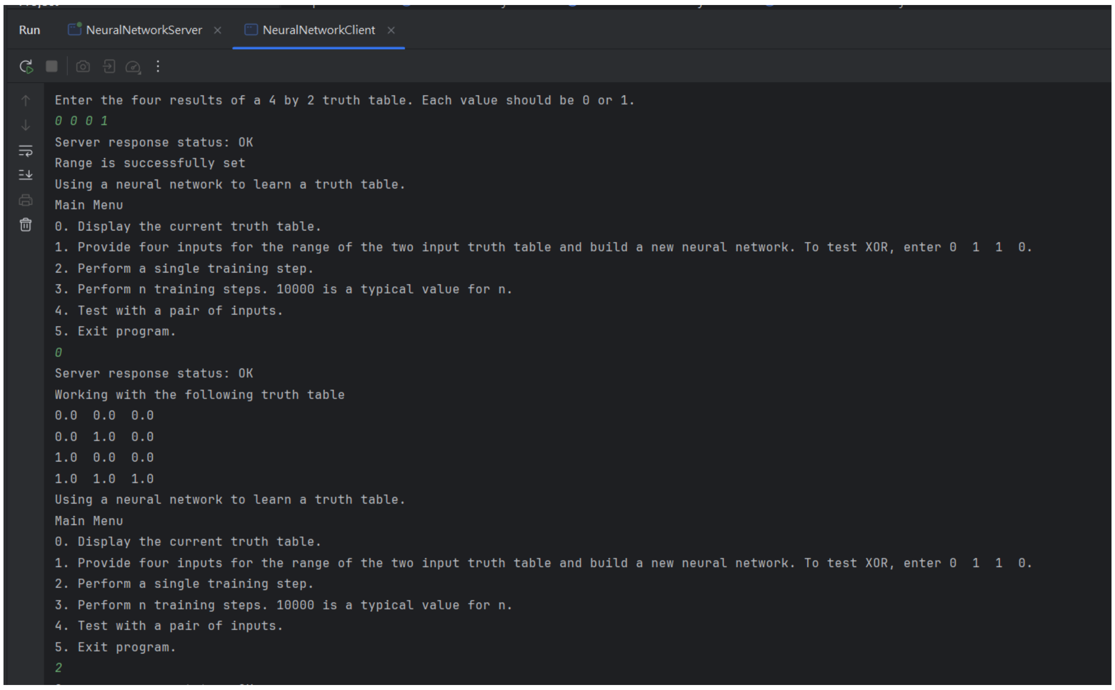
```
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
4
Enter a pair of doubles from a row of the truth table. These are domain values.
0 0
Server response status: OK
The range value is approximately 0.03621714993960037
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
5
Client quit...

Process finished with exit code 0
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

AND: 0 0 0 1

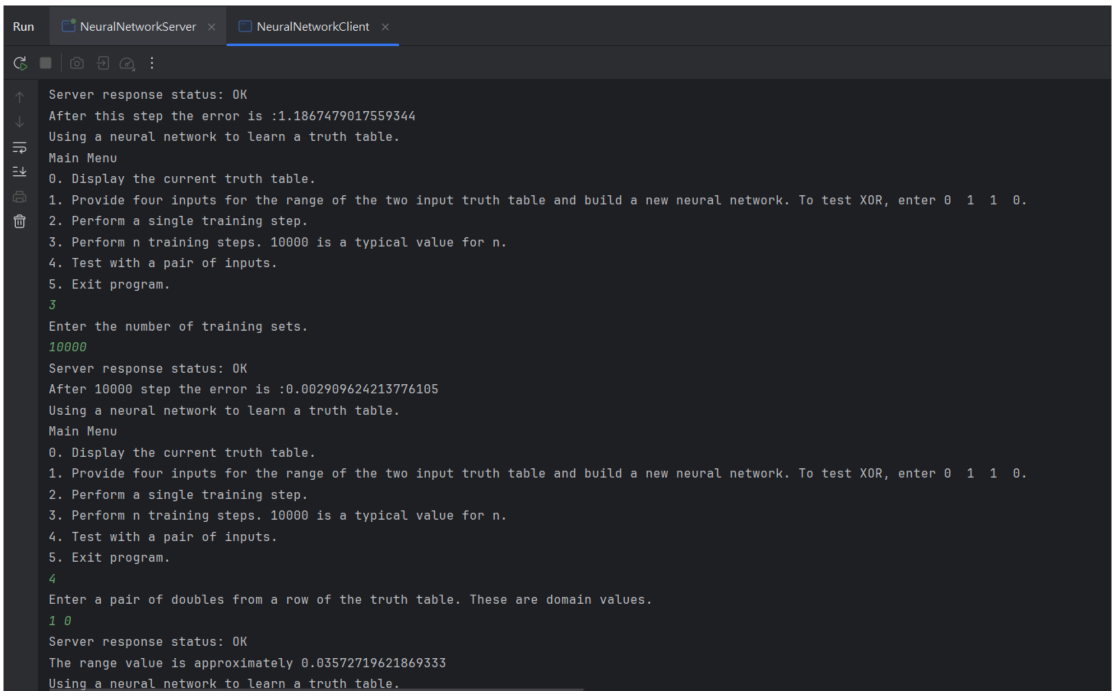


```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2\lib\idea_rt.jar=64052
The client is running.
Please enter server port:
6789
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
0
Server response status: OK
Working with the following truth table
0.0  0.0  0.0
0.0  1.0  1.0
1.0  0.0  1.0
1.0  1.0  1.0
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
1
Enter the four results of a 4 by 2 truth table. Each value should be 0 or 1.
```

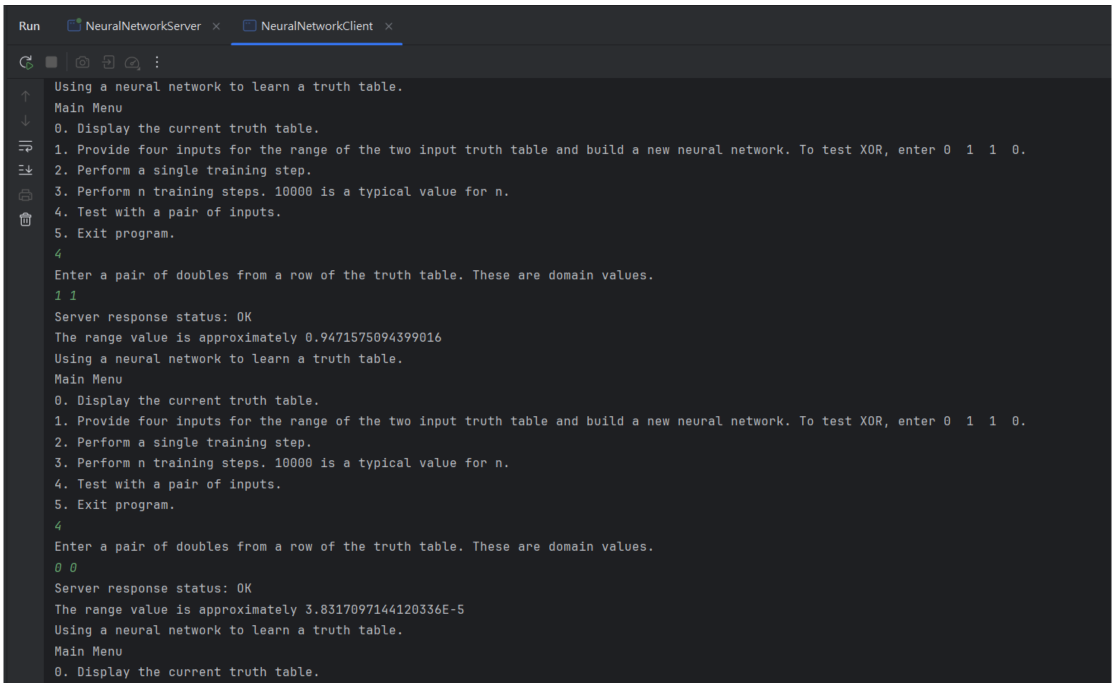


```
Enter the four results of a 4 by 2 truth table. Each value should be 0 or 1.
0 0 0 1
Server response status: OK
Range is successfully set
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
0
Server response status: OK
Working with the following truth table
0.0  0.0  0.0
0.0  1.0  0.0
1.0  0.0  0.0
1.0  1.0  1.0
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
2
```

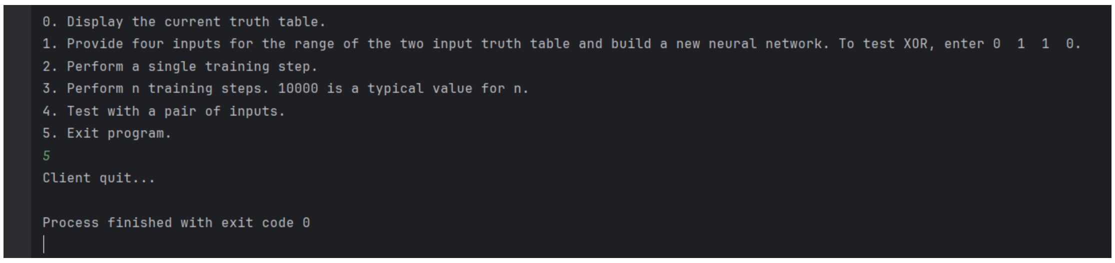
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



```
Server response status: OK
After this step the error is :1.1867479017559344
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
3
Enter the number of training sets.
10000
Server response status: OK
After 10000 step the error is :0.002909624213776105
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
4
Enter a pair of doubles from a row of the truth table. These are domain values.
1 0
Server response status: OK
The range value is approximately 0.03572719621869333
Using a neural network to learn a truth table.
```



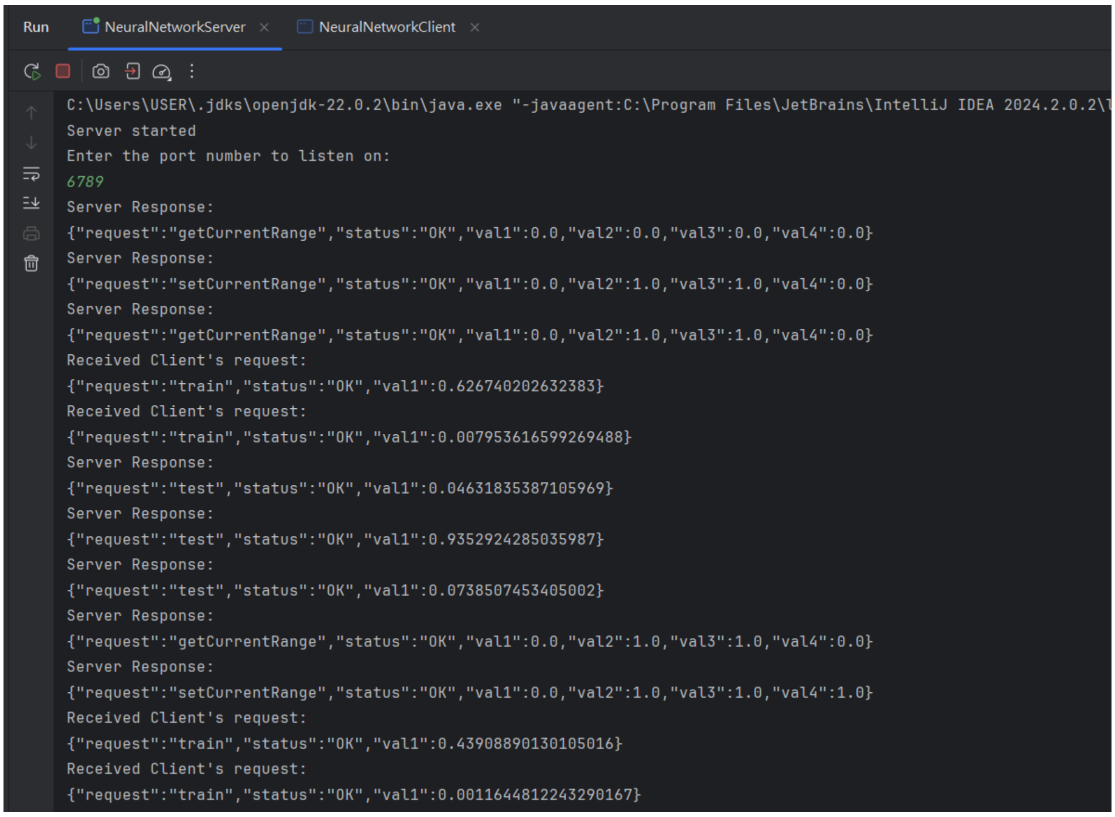
```
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
4
Enter a pair of doubles from a row of the truth table. These are domain values.
1 1
Server response status: OK
The range value is approximately 0.9471575094399016
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
4
Enter a pair of doubles from a row of the truth table. These are domain values.
0 0
Server response status: OK
The range value is approximately 3.8317097144120336E-5
Using a neural network to learn a truth table.
Main Menu
0. Display the current truth table.
```



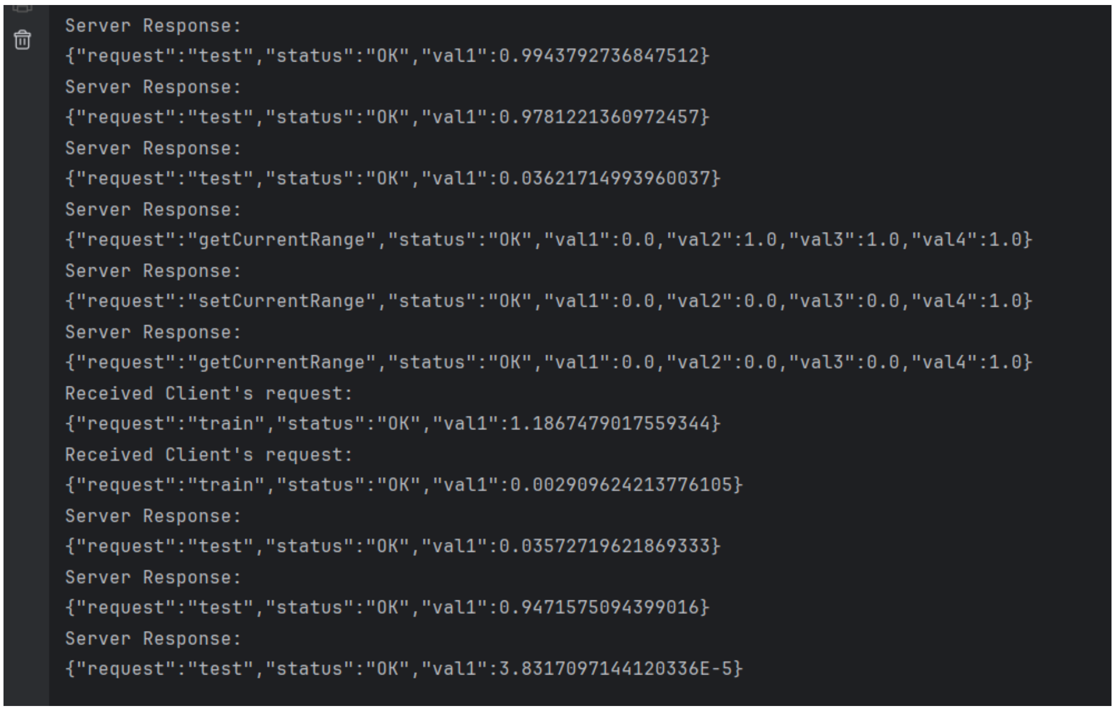
```
0. Display the current truth table.
1. Provide four inputs for the range of the two input truth table and build a new neural network. To test XOR, enter 0  1  1  0.
2. Perform a single training step.
3. Perform n training steps. 10000 is a typical value for n.
4. Test with a pair of inputs.
5. Exit program.
5
Client quit...

Process finished with exit code 0
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

4. Project2Task4ServerConsole



```
C:\Users\USER\.jdks\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2\
Server started
Enter the port number to listen on:
6789
Server Response:
{"request":"getCurrentRange","status":"OK","val1":0.0,"val2":0.0,"val3":0.0,"val4":0.0}
Server Response:
{"request":"setCurrentRange","status":"OK","val1":0.0,"val2":1.0,"val3":1.0,"val4":0.0}
Server Response:
{"request":"getCurrentRange","status":"OK","val1":0.0,"val2":1.0,"val3":1.0,"val4":0.0}
Received Client's request:
{"request":"train","status":"OK","val1":0.626740202632383}
Received Client's request:
{"request":"train","status":"OK","val1":0.007953616599269488}
Server Response:
{"request":"test","status":"OK","val1":0.04631835387105969}
Server Response:
{"request":"test","status":"OK","val1":0.9352924285035987}
Server Response:
{"request":"test","status":"OK","val1":0.0738507453405002}
Server Response:
{"request":"getCurrentRange","status":"OK","val1":0.0,"val2":1.0,"val3":1.0,"val4":0.0}
Server Response:
{"request":"setCurrentRange","status":"OK","val1":0.0,"val2":1.0,"val3":1.0,"val4":1.0}
Received Client's request:
{"request":"train","status":"OK","val1":0.43908890130105016}
Received Client's request:
{"request":"train","status":"OK","val1":0.0011644812243290167}
```



```
Server Response:
{"request":"test","status":"OK","val1":0.9943792736847512}
Server Response:
{"request":"test","status":"OK","val1":0.9781221360972457}
Server Response:
{"request":"test","status":"OK","val1":0.03621714993960037}
Server Response:
{"request":"getCurrentRange","status":"OK","val1":0.0,"val2":1.0,"val3":1.0,"val4":1.0}
Server Response:
{"request":"setCurrentRange","status":"OK","val1":0.0,"val2":0.0,"val3":0.0,"val4":1.0}
Server Response:
{"request":"getCurrentRange","status":"OK","val1":0.0,"val2":0.0,"val3":0.0,"val4":1.0}
Received Client's request:
{"request":"train","status":"OK","val1":1.1867479017559344}
Received Client's request:
{"request":"train","status":"OK","val1":0.002909624213776105}
Server Response:
{"request":"test","status":"OK","val1":0.03572719621869333}
Server Response:
{"request":"test","status":"OK","val1":0.9471575094399016}
Server Response:
{"request":"test","status":"OK","val1":3.8317097144120336E-5}
```