**IT5512- WEB TECHNOLOGY LAB-SESSION-5**

**DATE: 04/10/2021**

**NAME: A.S. PRUTHIEV**

**REG NO.2019506067**

**SOCKET PROGRAMMING**

**1)AIM:**

To write a Java Program for implementing echo client and echo server interaction (Two -way communication) via TCP Protocol

**ALGORITHM:**

**Client Side:**

* Create an object of the socket class and an ArrayList of integers
* Read the n elements to the ArrayList
* Create an object of the OutputStreamAbstractClass and ObjectOutputStream class to send the array List to the output port of the Server.
* When the control reaches back to the client again display the result (i.e., the sum)

**Server Side :**

* Create a ServerSocket class and mention the port number (same as the one used in client file)
* Use the InputStreamAbstractClass and InputStream classes to receive and type cast to the appropriate data type and calculate the sum and send it back to the client

**PROGRAM CODE:**

**Client Code:**

package Java.SocketProgramming;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.net.Socket;

import java.util.ArrayList;

import java.util.Scanner;

public class TwoWayCommunicationClient {

private static Scanner input = new Scanner(System.in);

public static void main(String[] args) throws Exception {

String ipAddress = "192.168.1.18";

int portNumber = 9999;

Socket clientSocket = new Socket(ipAddress,portNumber);

int n;

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

n = input.nextInt();

ArrayList<Integer>values = new ArrayList<>(n);

System.out.print("Enter values :");

for(int i = 0,j ; i < n ; i++){

j = input.nextInt();

values.add(j);

}

OutputStream outputStream = clientSocket.getOutputStream();

ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream);

System.out.println("The client writes the data");

objectOutputStream.writeObject(values);

InputStream inputStream = clientSocket.getInputStream();

ObjectInputStream objectInputStream = new ObjectInputStream(inputStream);

System.out.println("The client reads the sum from server ");

Integer sum = (Integer) objectInputStream.readObject();

System.out.println("The sum of the values are :" + sum);

}

}

**Server Code:**

package Java.SocketProgramming;

import java.io.\*;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.stream.Stream;

public class TwoWayCommunicationServer {

public static void main(String[] args) throws Exception{

System.out.println("The server is started ");

ServerSocket serverSocket = new ServerSocket(9999);

System.out.println("Server is waiting to receive the data");

Socket socket = serverSocket.accept();

InputStream inputStream = socket.getInputStream();

ObjectInputStream objectInputStream = new ObjectInputStream(inputStream);

System.out.println("Server reads the data");

List<Integer> values = (List<Integer>)objectInputStream.readObject();

System.out.print("The data read is :");

values.forEach(x -> System.out.print(x + " "));

System.out.println();

Integer sumValues = values.stream().mapToInt(x -> x).sum();

System.out.println("The sum is calculted ");

OutputStream outputStream = socket.getOutputStream();

ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream);

System.out.println("The server writes into the client ");

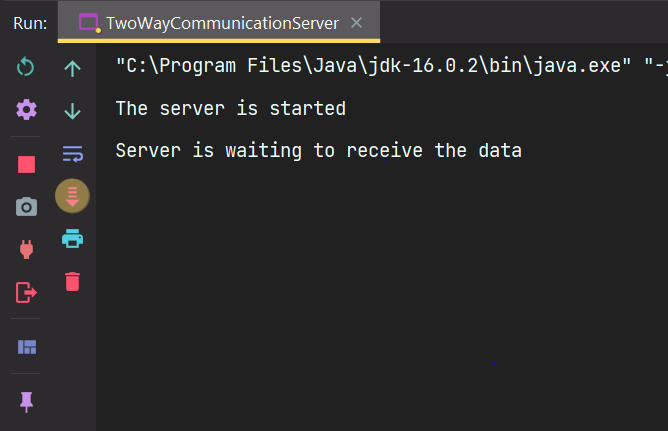
objectOutputStream.writeObject(sumValues);

}

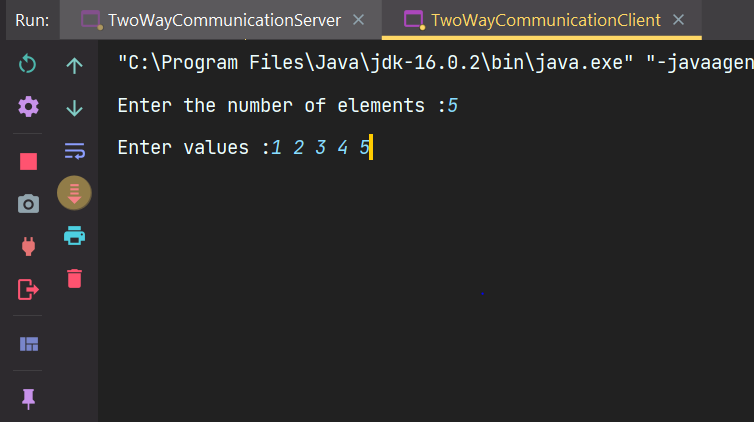
}

**OUTPUT:**

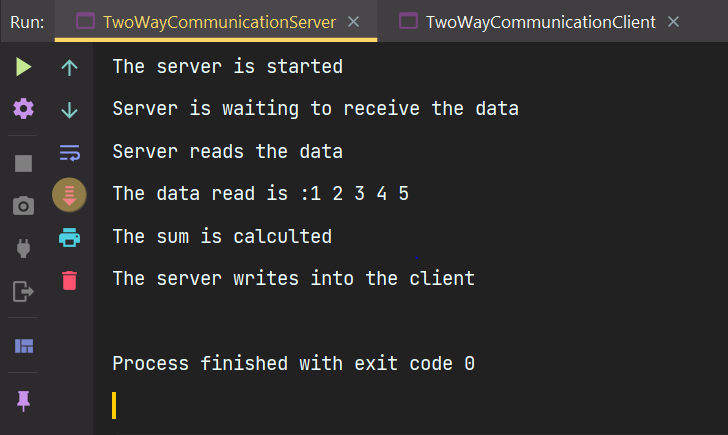
Initial run of the server:



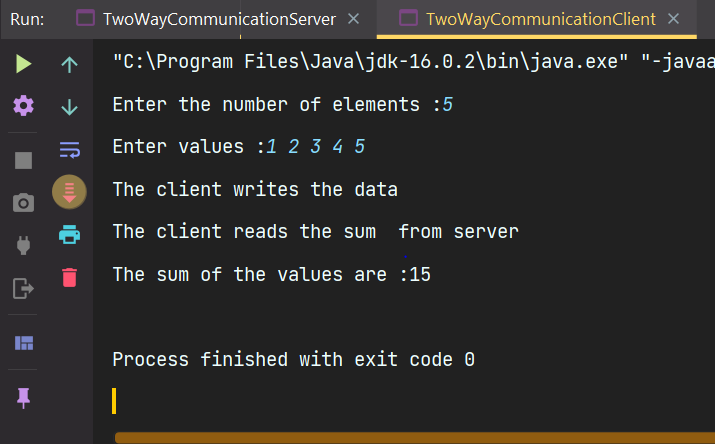
Entering the data in the client:



Server processing the received data :



Final result shown in client :



**RESULT:** Thus, the program has been executed successfully

**2)AIM:**

To write a java program in which we create a chat application between client and server via TCP Protocol

**ALGORITHM:**

**Client side:**

* Read the message from the user through a scanner
* Send it the output port of the server through OutputStreamClass
* Receive the message from inputStreamClass and print this process goes on until we send bye;

**Server side:**

* Read the message from the client through InputStream class and print it
* Read the message from the server and send it to the client via OutputStreamClass
* This process goes on until we send bye;

**PROGRAM CODE:**

**Client Code:**

package Java.SocketProgramming;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.net.Socket;

import java.util.Scanner;

public class ChatAppClient1 {

private static Scanner input = new Scanner(System.in);

public static void main(String[] args) throws Exception {

String ipAddress = "192.168.1.18";

int portNumber = 9999;

Socket socket = new Socket(ipAddress , portNumber);

OutputStream outputStream = socket.getOutputStream();

ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream);

InputStream inputStream = socket.getInputStream();

ObjectInputStream objectInputStream = new ObjectInputStream(inputStream);

while(true){

System.out.print("Client :");

String msg,msgReceive;

msg = input.nextLine();

objectOutputStream.writeObject(msg);

msgReceive = (String) objectInputStream.readObject();

System.out.println("Server : " + msgReceive);

if(msg.toLowerCase().contains("bye") && msgReceive.toLowerCase().contains("bye")){

System.out.println("Client chat ended ");

break;

}

}

}

}

**Server Code:**

package Java.SocketProgramming;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Scanner;

public class ChatAppServer1 {

private static Scanner input = new Scanner(System.in);

public static void main(String[] args) throws Exception {

System.out.println("Server has started , server waiting for client calls");

ServerSocket serverSocket = new ServerSocket(9999);

Socket socket = serverSocket.accept();

OutputStream outputStream = socket.getOutputStream();

ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream);

InputStream inputStream = socket.getInputStream();

ObjectInputStream objectInputStream = new ObjectInputStream(inputStream);

while(true){

String msgReceived,msg;

msgReceived = (String)objectInputStream.readObject();

System.out.println("Client :" + msgReceived);

System.out.print("Server :");

msg = input.nextLine();

if(msgReceived.toLowerCase().contains("bye")){

objectOutputStream.writeObject(msg);

break;

}

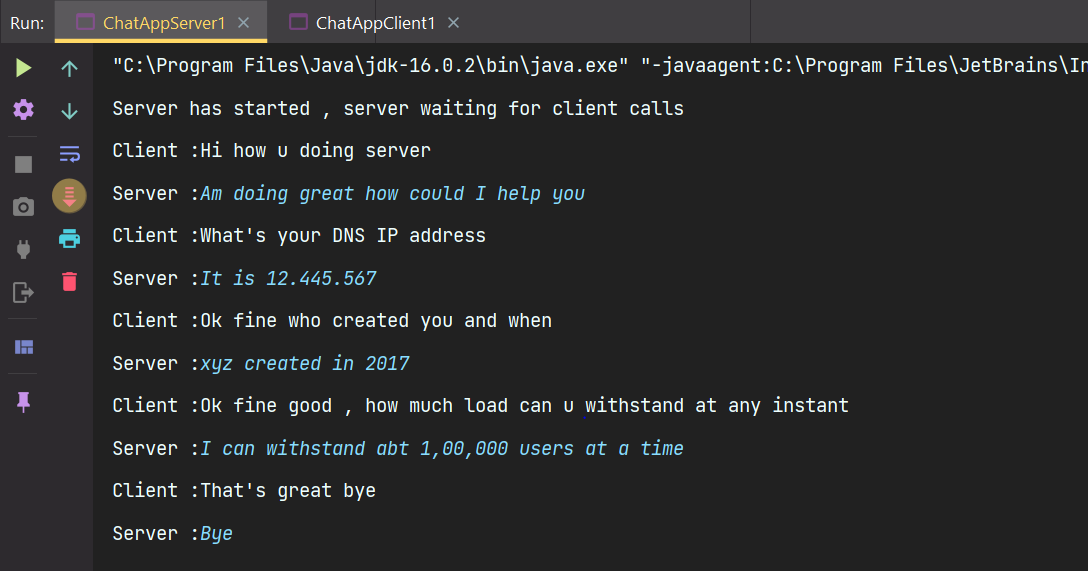
objectOutputStream.writeObject(msg);

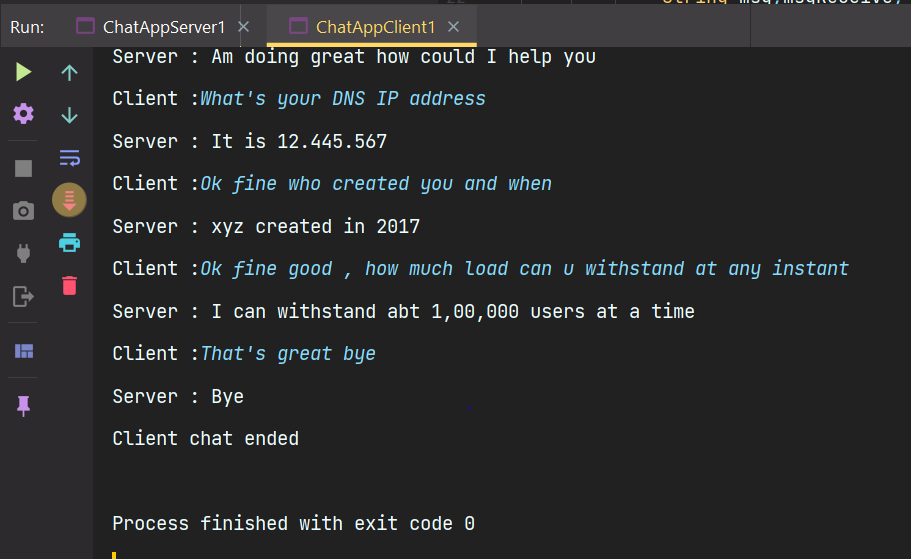
}

}

}

**OUTPUT:**





**RESULT:**

Thus, the program has been executed successfully.

**3)AIM:**

To write a Java Program for implementing echo client and echo server interaction via UDP Protocol

**ALGORITHM:**

**Client Side:**

* Write a message that the client can interact with the server from now
* Read the message from the client side
* Convert the string literal to byte array
* Get the localhost from InetAddress class
* Create a datagram socket object handle it since it throws exception
* Create a datagram packet object and pass the byte array ,it’s length,Inetaddress and port number as args to the constructor
* Pass the datagramPacket object as the parameter to the send method of the datagram socket class
* Create a new data packet for receiving and use the receive method of the datagram socket class
* Convert it to a string and display the message from the server

**Server Side:**

* Write a message that the server is ready to receive the data packets from the client.
* Get the localhost from InetAddress class
* Create a datagram socket object handle it since it throws exception
* Create a datagram packet object and receive the message as bytes from the client using the receive method of the datagram socket class.
* Convert the received message to strings and print it
* Read the message from the server side and convert it into the byte array
* Create a new datagram packet for sending the data to the client and send the converted byte array,it’s length,inetAddress and port number

**PROGRAM CODE:**

**Client Code:**

package Java.SocketProgramming;

import javax.xml.crypto.Data;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

public class UDPEchoClient {

private static final Scanner input = new Scanner(System.in);

private static final int PORT\_NUMBER = 9999;

public static final String ANSI\_YELLOW = "\u001b[33m";

public static final String ANSI\_PURPLE = "\u001B[35m";

public static final String ANSI\_GREEN = "\u001B[32m";

public static void main(String[] args) throws Exception{

System.out.println(ANSI\_YELLOW + "The client can start a conversation with the server");

String msg;

System.out.print(ANSI\_PURPLE + "Enter a string message :");

msg = input.nextLine();

DatagramSocket datagramSocket = new DatagramSocket();

byte []msgArray = msg.getBytes();

InetAddress inetAddress = InetAddress.getLocalHost();

DatagramPacket datagramPacket = new DatagramPacket(msgArray, msgArray.length,inetAddress,PORT\_NUMBER);

datagramSocket.send(datagramPacket);

byte [] msgReceivedArray = new byte[1024];

DatagramPacket datagramPacket1 = new DatagramPacket(msgReceivedArray, msgReceivedArray.length);

datagramSocket.receive(datagramPacket1);

String msgReceived = new String(datagramPacket1.getData(),0,datagramPacket1.getLength());

System.out.println(ANSI\_YELLOW + "The message received from server :" + ANSI\_GREEN + msgReceived);

}

}

**Server Code:**

package Java.SocketProgramming;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

public class UDPEchoServer {

private static final Scanner input = new Scanner(System.in);

private static final int PORT\_NUMBER = 9999;

public static final String ANSI\_YELLOW = "\u001b[33m";

public static final String ANSI\_PURPLE = "\u001B[35m";

public static final String ANSI\_GREEN = "\u001B[32m";

public static void main(String[] args) throws Exception{

System.out.println(ANSI\_YELLOW + "The server is started and ready to receive the packets from the client side");

DatagramSocket datagramSocket = new DatagramSocket(PORT\_NUMBER);

byte []msgReceivedArray = new byte[1024];

DatagramPacket datagramPacket = new DatagramPacket(msgReceivedArray, msgReceivedArray.length);

datagramSocket.receive(datagramPacket);

String msgReceived = new String(datagramPacket.getData(),0,datagramPacket.getLength());

System.out.println(ANSI\_PURPLE + "The message received from client is :" +ANSI\_GREEN + msgReceived);

System.out.print("Enter your message :");

String msg = input.nextLine();

byte[] msgArray = msg.getBytes();

InetAddress inetAddress = InetAddress.getLocalHost();

DatagramPacket datagramPacket1 = new DatagramPacket(msgArray, msgArray.length,inetAddress,datagramPacket.getPort());

datagramSocket.send(datagramPacket1);

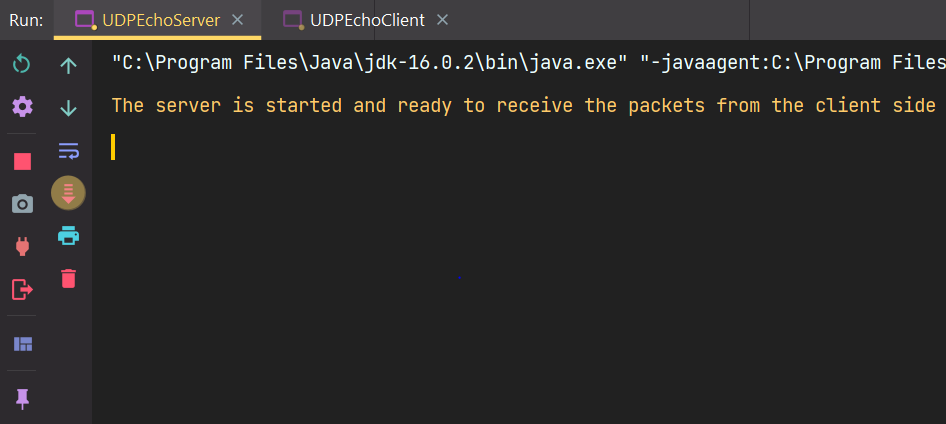
}

}

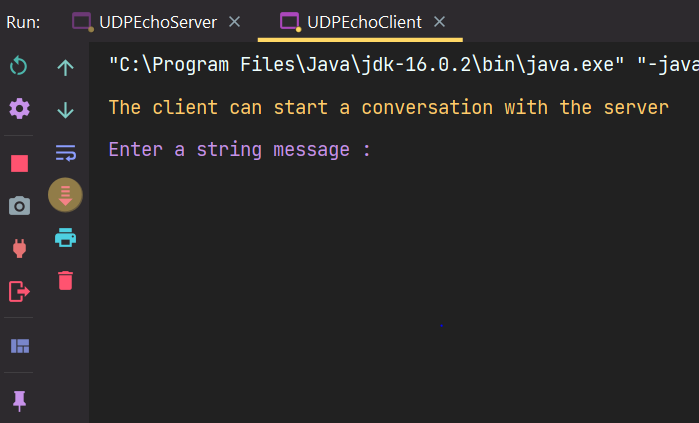
**OUTPUT:**

Run the Server file and then client file

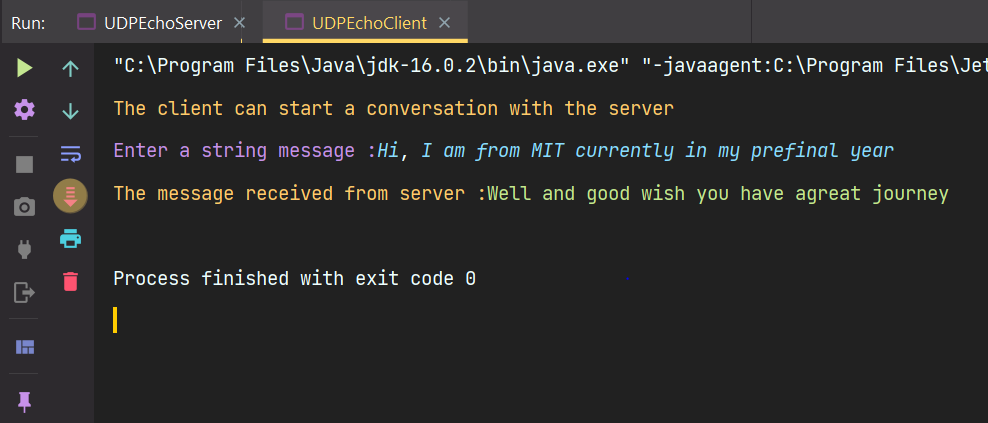
Initial run of the server file:

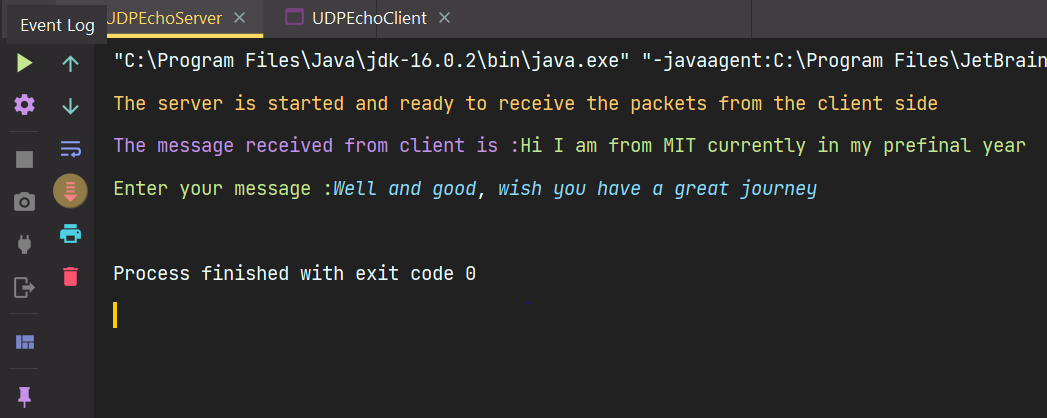


Intial Run of the client file :



Final Result after displaying the message from client in the server file





**RESULT:**  
Thus the program has been executed successfully.

**4)AIM:**

To write a Java Program for implementing echo client and echo server interaction via UDP Protocol

**ALGORITHM:**

**Client Side:**

* Write a message that the client can interact with the server from now
* Read the number of elements to be read from the user and create a byte array of that size
* Read the values from the user.
* Get the localhost from InetAddress class
* Create a datagram socket object handle it since it throws exception
* Create a datagram packet object and pass the byte array ,it’s length,Inetaddress and port number as args to the constructor
* Pass the datagramPacket object as the parameter to the send method of the datagram socket class
* Create a new data packet for receiving and use the receive method of the datagram socket class
* Convert it to a string and display the message from the server.

**Server Side:**

* Write a message that the server is ready to receive the data packets from the client.
* Get the localhost from InetAddress class
* Create a datagram socket object handle it since it throws exception
* Create a datagram packet object and receive the message as bytes from the client using the receive method of the datagram socket class.
* Fint the sum of the byte array that we received from the client side.
* Convert it to a string and then to a byte array which holds the sum of the entire byte array
* Create a new datagram packet for sending the data to the client and send the converted byte array,it’s length,inetAddress and port number.

**PROGRAM CODE:**

Client Code:

package Java.SocketProgramming;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

public class UDPClient1 {

private static final Scanner input = new Scanner(System.in);

public static final String ANSI\_YELLOW = "\u001b[33m";

public static final String ANSI\_PURPLE = "\u001B[35m";

public static final String ANSI\_GREEN = "\u001B[32m";

public static void main(String[] args) throws Exception {

System.out.println(ANSI\_YELLOW + "The server is started and ready to receive the packets from the client side");

DatagramSocket datagramSocket = new DatagramSocket();

int n;

System.out.print(ANSI\_PURPLE + "Enter the number of elements :");

n = input.nextInt();

byte []arr = new byte[n];

System.out.print(ANSI\_PURPLE + "Enter elements :");

for(int i = 0 ; i < n ; i++)arr[i] = input.nextByte();

InetAddress inetAddress = InetAddress.getLocalHost();

System.out.println(ANSI\_YELLOW + "The data packet is sending ....");

DatagramPacket datagramPacket = new DatagramPacket(arr,arr.length,inetAddress,9999);

datagramSocket.send(datagramPacket);

byte [] receivedData = new byte[1024];

DatagramPacket datagramPacket1 = new DatagramPacket(receivedData,receivedData.length);

datagramSocket.receive(datagramPacket1);

System.out.println(ANSI\_YELLOW + "The data processed is being received ");

String res = new String(datagramPacket1.getData(),0,datagramPacket1.getLength());

System.out.println(ANSI\_PURPLE + "The result is :" + res);

}

}

**Server Code:**

package Java.SocketProgramming;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class UDPServer1 {

public static final String ANSI\_YELLOW = "\u001b[33m";

public static final String ANSI\_PURPLE = "\u001B[35m";

public static final String ANSI\_GREEN = "\u001B[32m";

public static void main(String[] args)throws Exception {

System.out.println(ANSI\_YELLOW + "The server is started and ready to receive the packets from the client side");

DatagramSocket datagramSocket = new DatagramSocket(9999);

byte [] receivedArray = new byte[1024];

DatagramPacket datagramPacket = new DatagramPacket(receivedArray,receivedArray.length);

datagramSocket.receive(datagramPacket);

System.out.println("The data packet is received...");

receivedArray = datagramPacket.getData();

int n = datagramPacket.getLength();

int sum = 0;

System.out.println(ANSI\_PURPLE + "The data received is :");

for(int i = 0 ; i < n ; i++)System.out.print(receivedArray[i] + " ");

System.out.println();

for(int i = 0 ; i < n ; i++)sum = sum + receivedArray[i];

byte [] calArray = String.valueOf(sum).getBytes();

InetAddress inetAddress = InetAddress.getLocalHost();

DatagramPacket datagramPacket1 = new DatagramPacket(calArray,calArray.length,inetAddress,datagramPacket.getPort());

System.out.println(ANSI\_YELLOW + "The data is processed and it is sending back to the client...");

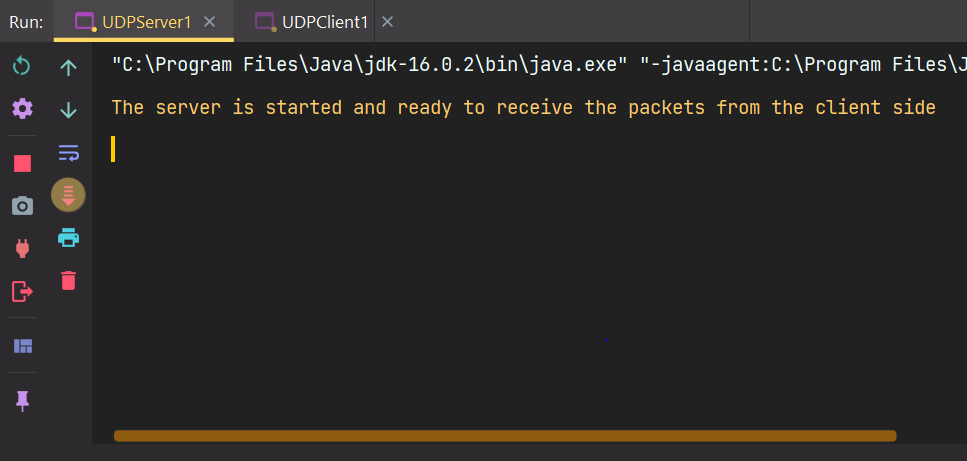
datagramSocket.send(datagramPacket1);

}

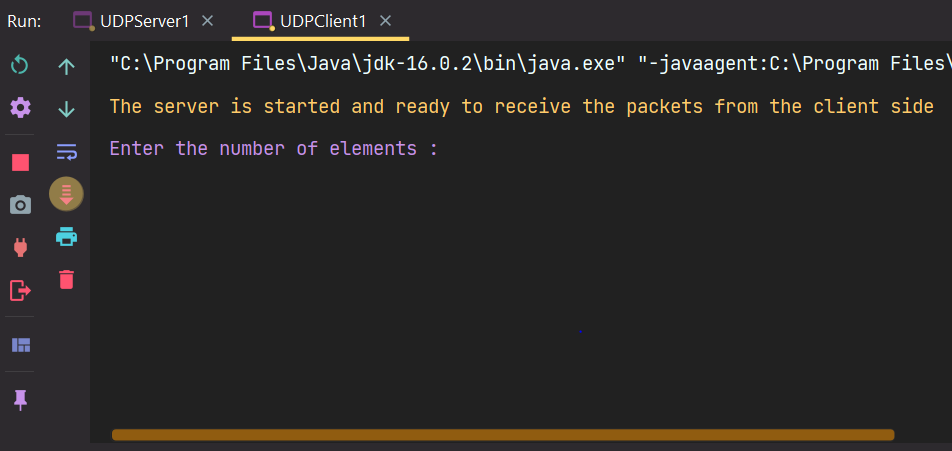
}

**OUTPUT:**

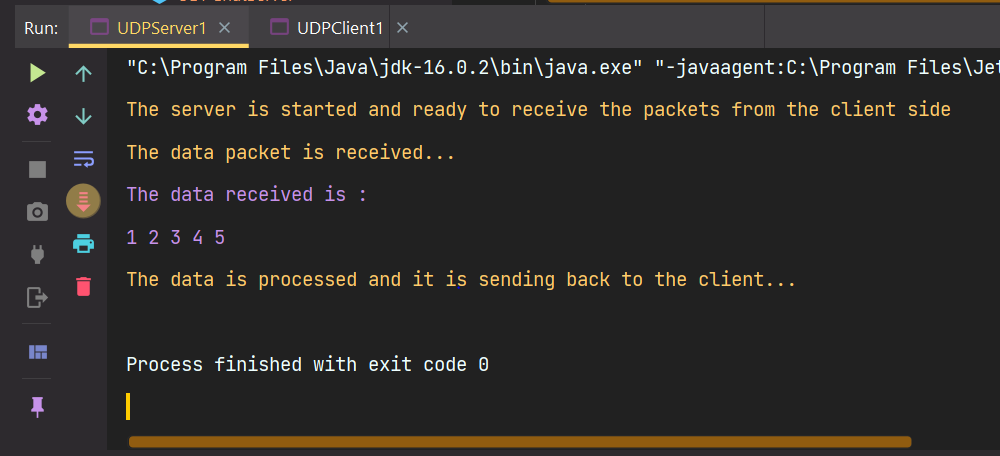
Initial run of the server:



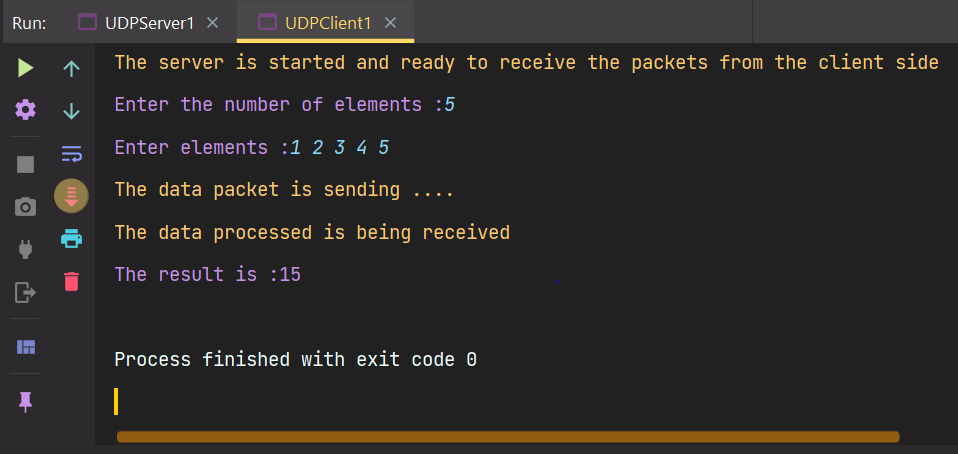
Intial run of the client:



Server processing the received data :



Final result shown in client :



**RESULT:** Thus, the program has been executed successfully