Assignment 1A

|  |  |
| --- | --- |
| **Name** | Abhishek Dilip Agashe |
| **Roll No** | 43103 |
| **Batch** | P9 |
| **Problem Statement** | To develop any distributed application through implementing client-server  communication programs based on Java Sockets and RMI techniques. |

**Code**

**1. TCP**

**ServerTCP.java**

*// A Java program for a Server*

**import** java.net.\*;

**import** java.io.\*;

**import** java.lang.Math;

**public** **class** ServerTCP

{

*//initialize socket and input stream*

**private** Socket socket = **null**;

**private** ServerSocket server = **null**;

**private** DataInputStream in = **null**;

**private** DataOutputStream out = **null**;

*// constructor with port*

**public** ServerTCP(**int** port)

{

*// starts server and waits for a connection*

**try**

{

server = **new** ServerSocket(port);

System.out.println("Server started");

**int** i =1;

**while** (**true**) {

**try** {

System.out.println("Waiting for a client ...");

socket = server.accept();

System.out.println("Client "+i+" accepted");

ServerThread st = **new** ServerThread(socket,"Client "+String.valueOf(i));

i++;

st.start();

} **catch** (Exception e) {

System.out.println("connetion error");

}

}

}

**catch**(IOException i)

{

System.out.println(i);

}

}

**public** **static** **void** main(String args[])

{

ServerTCP server = **new** ServerTCP(5000);

}

}

**class** ServerThread **extends** Thread{

**int** a =0, b=0;

DataInputStream in = **null**;

DataOutputStream out = **null**;

Socket socket = **null**;

String clientnum = "";

**public** ServerThread(Socket s, String clientnum) {

socket = s;

**this**.clientnum= clientnum;

}

**public** **void** run() {

**try** {

in = **new** DataInputStream(socket.getInputStream());

out = **new** DataOutputStream(socket.getOutputStream());

a = in.readInt();

b = in.readInt();

System.out.println("Recieved Base and Power. Answer is: "+ Math.pow(a,b));

} **catch** (IOException ie) {

System.out.println("socket close error");

}

}

}

**ClientTCP.java**

*// A Java program for a Client*

**import** java.net.\*;

**import** java.io.\*;

**import** java.math.\*;

**import** java.util.\*;

**public** **class** ClientTCP1

{

*// initialize socket and input output streams*

**private** Socket socket = **null**;

**private** DataInputStream input = **null**;

**private** DataInputStream in = **null**;

**private** DataOutputStream out = **null**;

*// constructor to put ip address and port*

**public** ClientTCP1(String address, **int** port)

{

*// establish a connection*

**try**

{

socket = **new** Socket(address, port);

System.out.println("Connected");

*// takes input from terminal*

input = **new** DataInputStream(System.in);

in = **new** DataInputStream(socket.getInputStream());

*// sends output to the socket*

out = **new** DataOutputStream(socket.getOutputStream());

}

**catch**(UnknownHostException u)

{

System.out.println(u);

}

**catch**(IOException i)

{

System.out.println(i);

}

Scanner sc= **new** Scanner(System.in);

**int** a=0,b=0;

**while** (a != -1)

{

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

a= sc.nextInt();

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

b= sc.nextInt();

System.out.println("Sending Base and Power to Server");

**try**

{

out.writeInt(a);

out.writeInt(b);

}

**catch**(IOException i)

{

System.out.println(i);

}

}

*// close the connection*

**try**

{

System.out.println("Closing connection");

input.close();

out.close();

socket.close();

}

**catch**(IOException i)

{

System.out.println(i);

}

}

**public** **static** **void** main(String args[])

{

ClientTCP1 client = **new** ClientTCP1("127.0.0.1", 5000);

}

}

**2. UDP**

**ServerUDP.java**

*//Java program to illustrate Server side*

*//Implementation using DatagramSocket*

**import** java.io.IOException;

**import** java.net.DatagramPacket;

**import** java.net.DatagramSocket;

**import** java.net.InetAddress;

**import** java.net.SocketException;

**public** **class** ServerUDP

{

**public** **static** **void** main(String[] args) **throws** IOException

{

*// Step 1 : Create a socket to listen at port 1234*

DatagramSocket ds = **new** DatagramSocket(1234);

**byte**[] receive = **new** **byte**[65535];

DatagramPacket DpReceive = **null**;

**while** (**true**)

{

**if** (data(receive).equals("bye"))

{

System.out.println("Client sent bye.....EXITING");

**break**;

}

*// Step 2 : create a DatgramPacket to receive the data.*

DpReceive = **new** DatagramPacket(receive, receive.length);

*// Step 3 : receive the data in byte buffer.*

ds.receive(DpReceive);

String[] s = data(receive).split("\\s+");

System.out.println("Base:-" + Integer.valueOf(s[0]));

System.out.println("Power:-" + Integer.valueOf(s[1]));

**double** c = Math.pow(Integer.valueOf(s[0]),Integer.valueOf(s[1]));

System.out.println("Answer:-" + c);

*// Exit the server if the client sends "bye"*

*// Clear the buffer after every message.*

receive = **new** **byte**[65535];

}

}

*// A utility method to convert the byte array*

*// data into a string representation.*

**public** **static** String data(**byte**[] a)

{

**if** (a == **null**)

**return** **null**;

StringBuilder ret = **new** StringBuilder();

**int** i = 0;

**while** (a[i] != 0)

{

ret.append((**char**) a[i]);

i++;

}

**return** ret.toString();

}

}

**ClientUDP.java**

*//Java program to illustrate Client side*

*//Implementation using DatagramSocket*

**import** java.io.IOException;

**import** java.net.DatagramPacket;

**import** java.net.DatagramSocket;

**import** java.net.InetAddress;

**import** java.util.Scanner;

**public** **class** ClientUDP1

{

**public** **static** **void** main(String args[]) **throws** IOException

{

Scanner sc = **new** Scanner(System.in);

*// Step 1:Create the socket object for*

*// carrying the data.*

DatagramSocket ds = **new** DatagramSocket();

InetAddress ip = InetAddress.getLocalHost();

**byte** buf[] = **null**;

*// loop while user not enters "bye"*

System.out.println("Enter Base and Power: <base power>");

**while** (**true**)

{

String eqn = sc.nextLine();

**if** (eqn.equals("bye"))

**break**;

buf = eqn.getBytes();

DatagramPacket DpSend = **new** DatagramPacket(buf, buf.length, ip, 1234);

ds.send(DpSend);

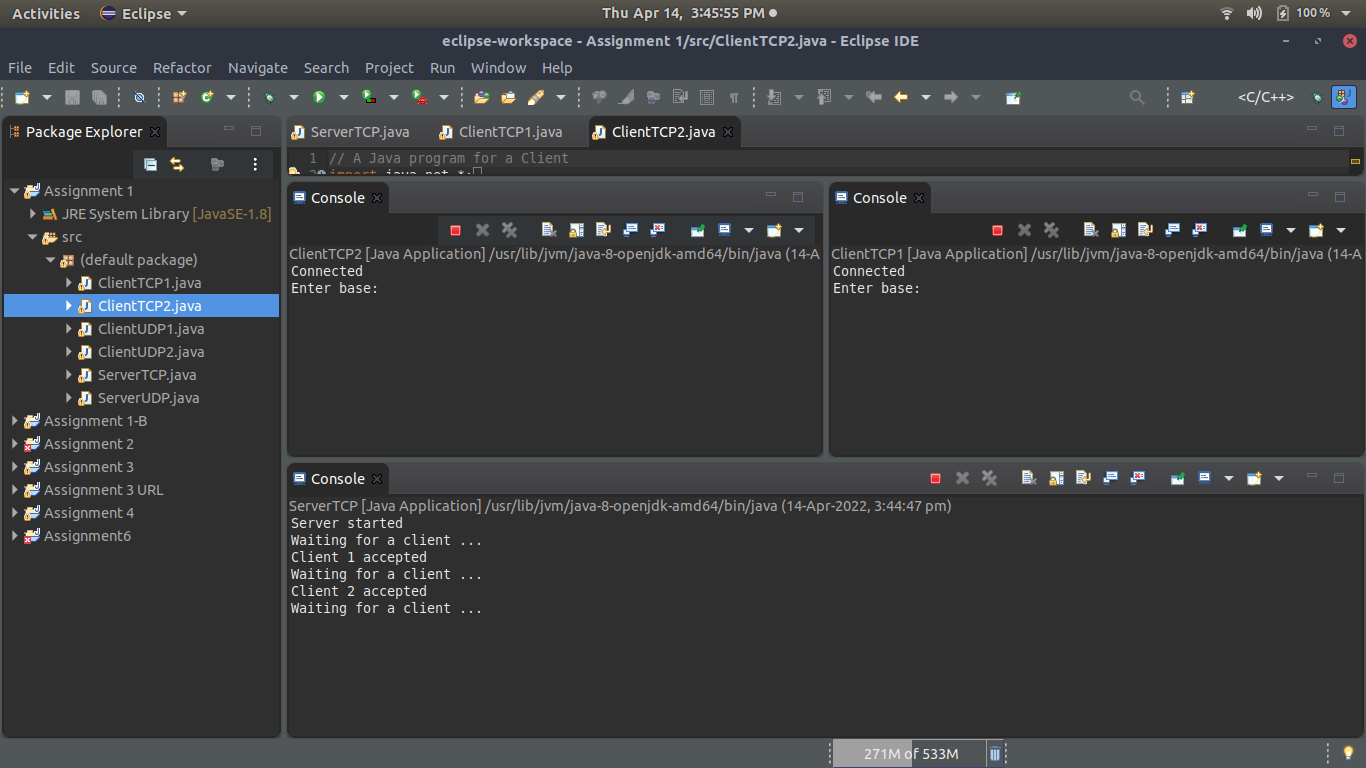
}

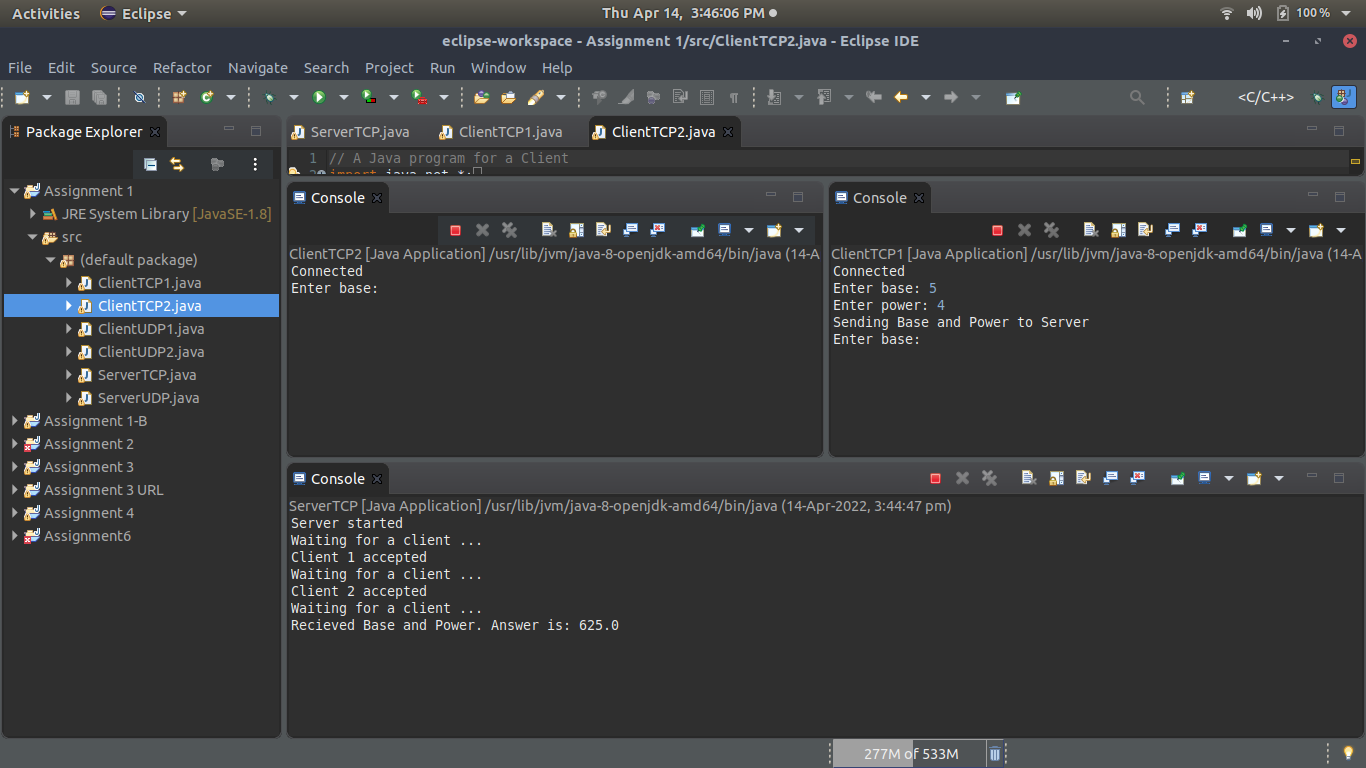
}

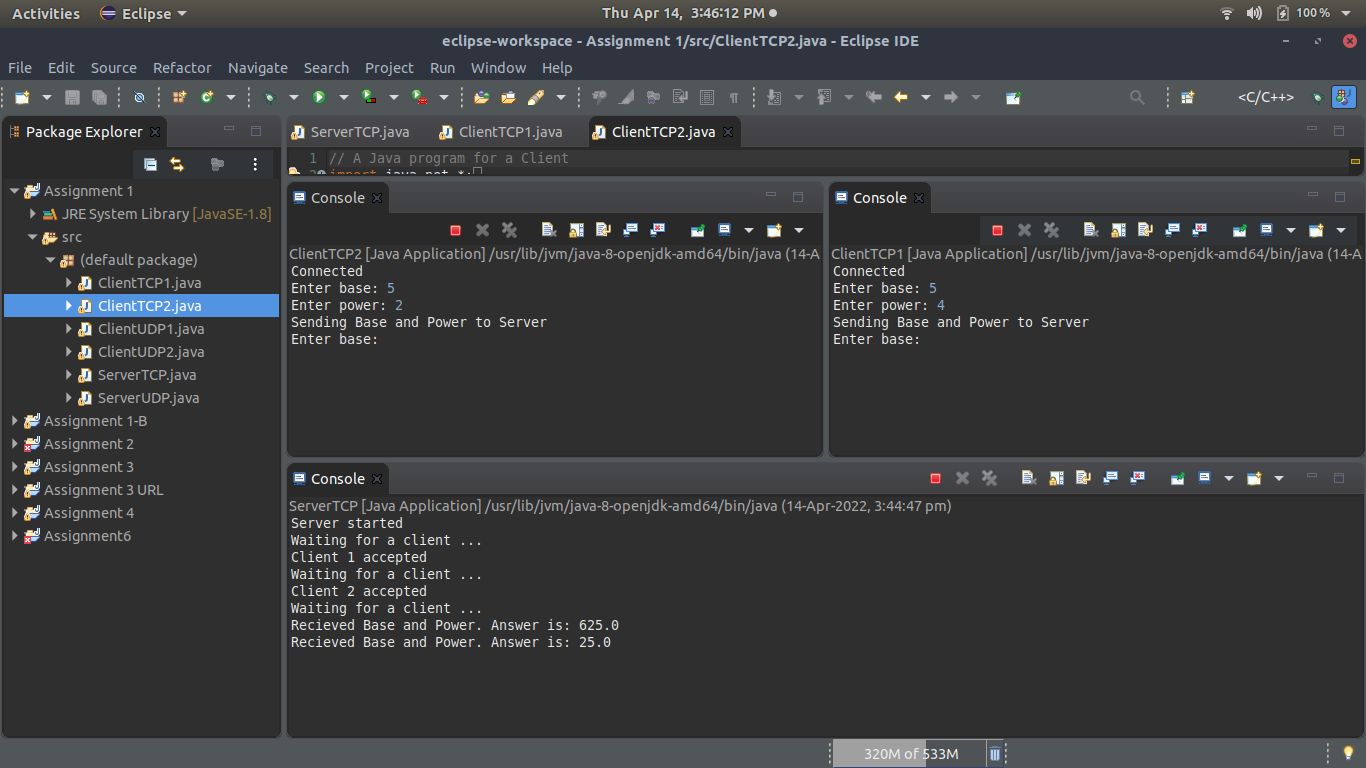
}

**Output**

**TCP**







**UDP**

