## **LAB #1**

## **OBJECTIVE:- To learn creation of datagram sockets**

**TASK1:- Modify the sample code so that the sender uses the same socket to send the same**

**message to two different receivers. Start the two receivers first, then the sender.**

**Does each receiver receive the message? Capture the code and output. Describe**

**he outcome.**

import java.net.\*;

import java.io.\*;

public class MyServer{

public static void main(String[] args)

{

if (args.length!=1)

System.out.println("This program requires a command line argument.");

else

{

int port =Integer.parseInt(args[0]);

final int MAX\_LEN=20;

try

{

DatagramSocket mySocket= new DatagramSocket(port);

byte[] buffer= new byte[MAX\_LEN];

DatagramPacket datagram= new DatagramPacket(buffer, MAX\_LEN);

mySocket.receive(datagram);

String message= new String(buffer);

System.out.println(message);

Thread.sleep(10000);

System.out.print("Exiting");

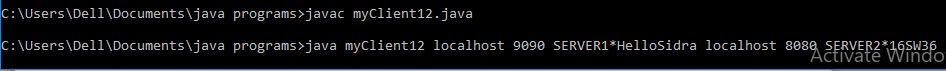
mySocket.close();

}catch(Exception ex)

{ ex.printStackTrace();}

} } }

**CLIENT OUTPUT:-**



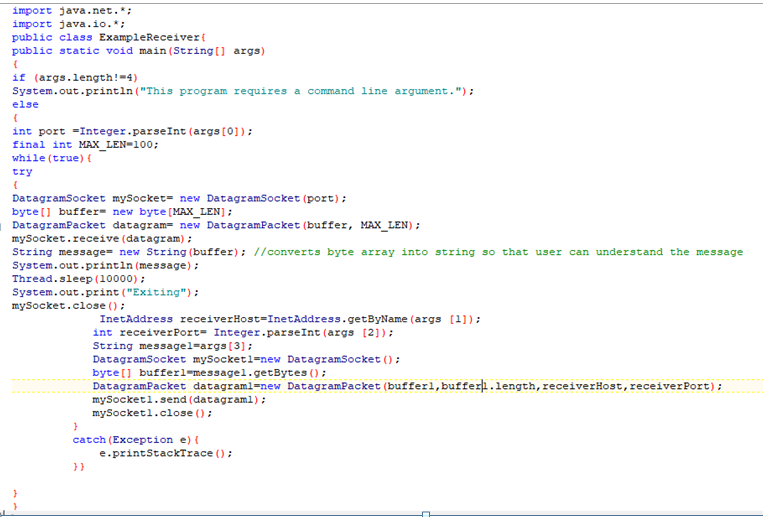
**SERVERS OUTPUT:-**



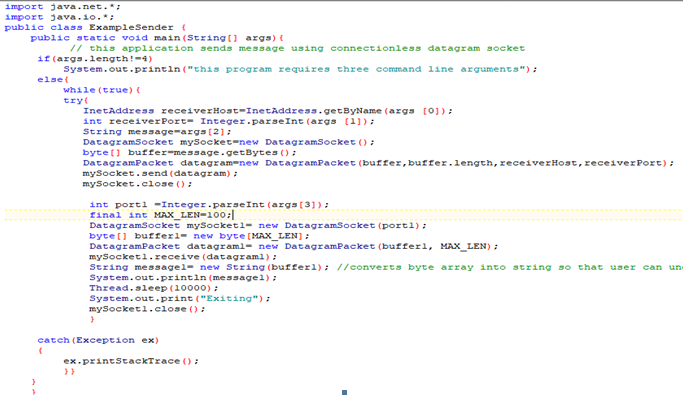
**Task#02:- Modify the sample code so that the receiver loops five times to repeatedly receive**

**then display your bio data (name, roll num etc.) received. Recompile. Then  
i. start the receiver  
ii. Execute the sender, sending your bio data, and  
iii. In another window, start another instance of the sender, sending your friend’s bio data. Does the receiver receive both the messages? Capture the code and output.**

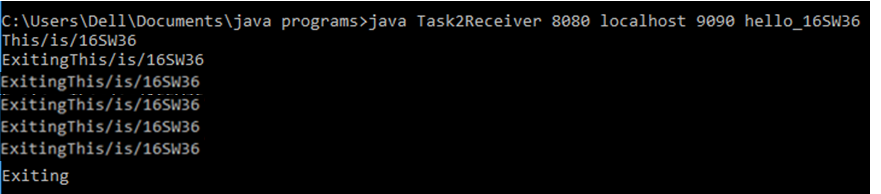
**CODE OF RECEIVER:-**



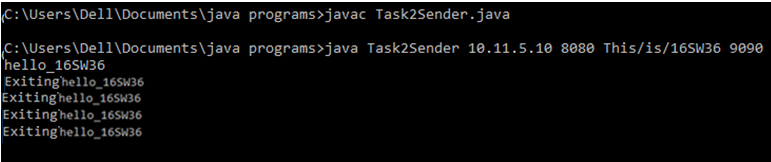
**CODE OF SENDER:-**



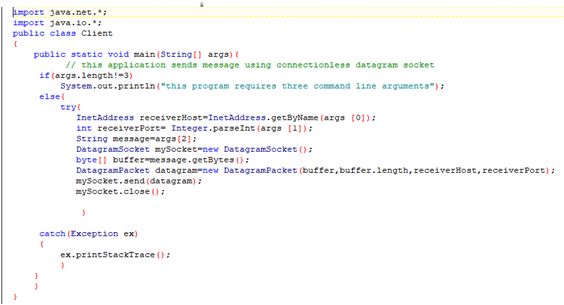
**OUTPUT OF RECEIVER:-**

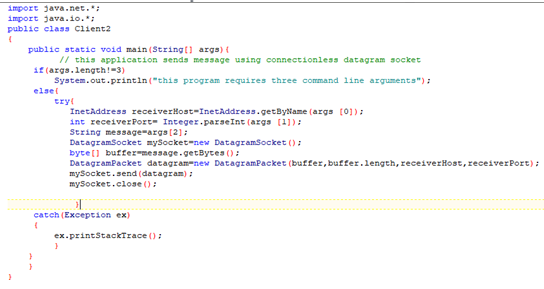


**OUTPUT OF SENDER:-**

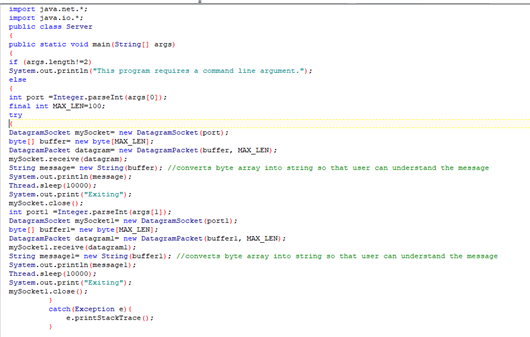


**CODEClient1**

**Client2:**



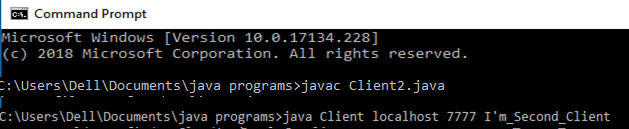
**Server:**



**OUTPUT: (Client1)**

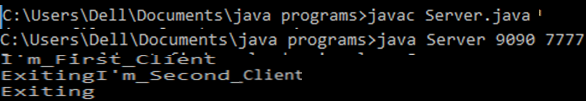


**(Client2)**



**Server:**

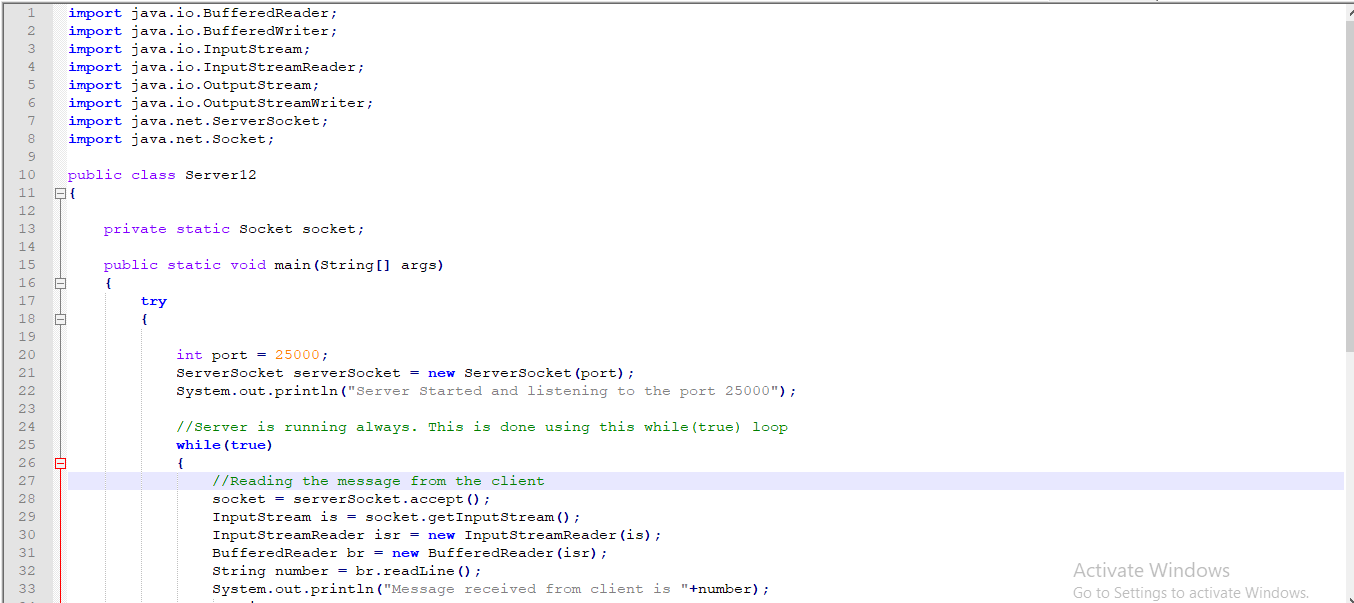
**Server is receiving both the messages(from client1 as well as from client2)**

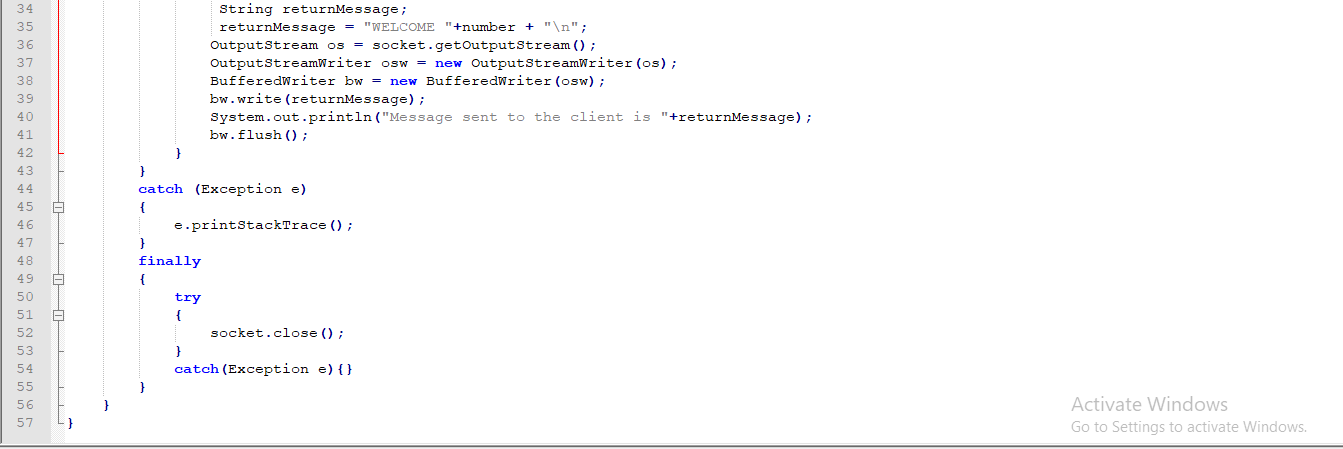


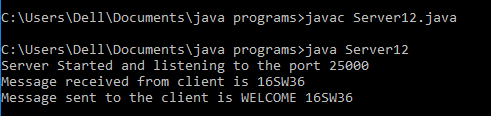
**TASK:03**

Modify the sample code to cater to a two way communication i.e. Sender sends a message to the Receiver, the Receiver receives the message and sends a reply to the Sender in return.

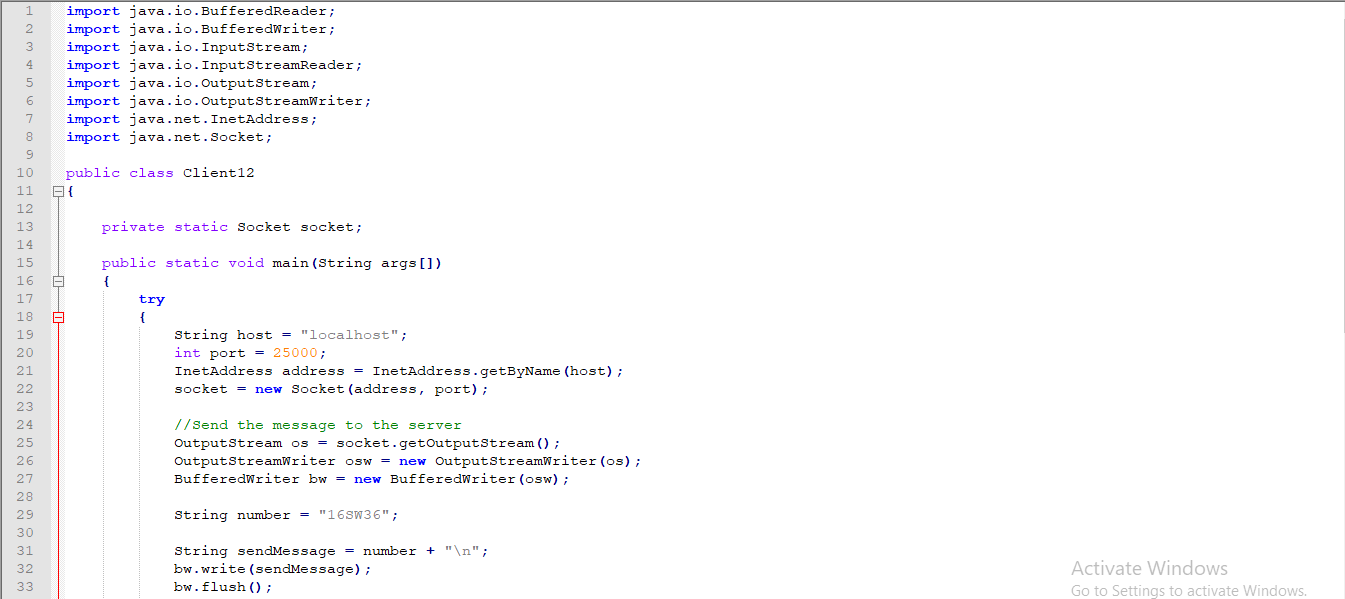
**Server12.java:-**

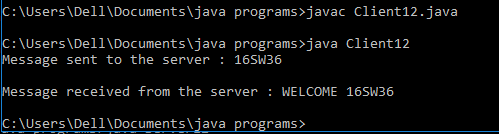
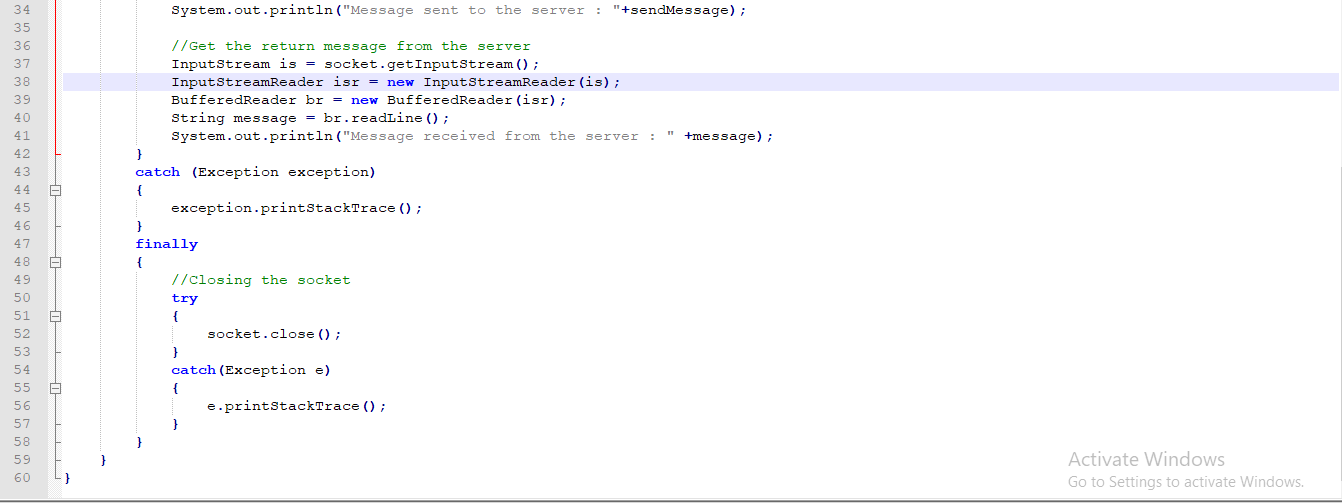






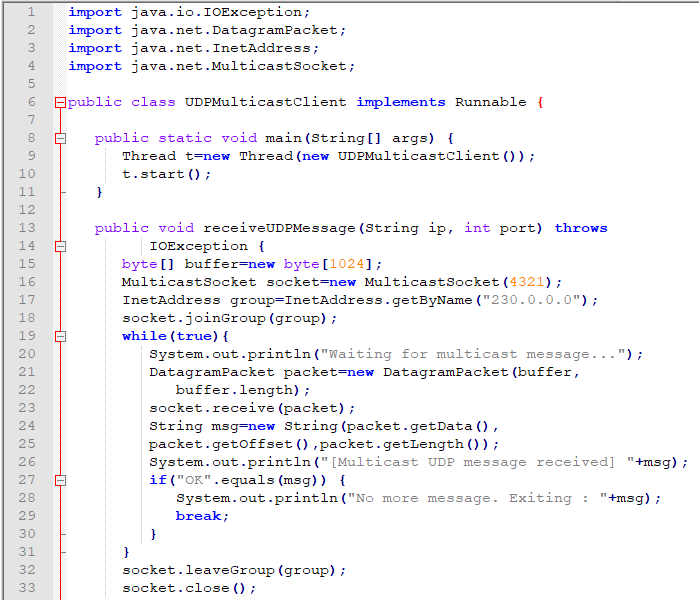
**CLIENT12.java:-**

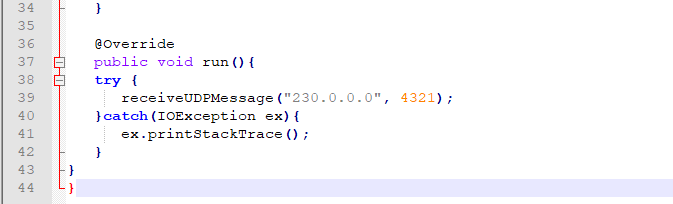




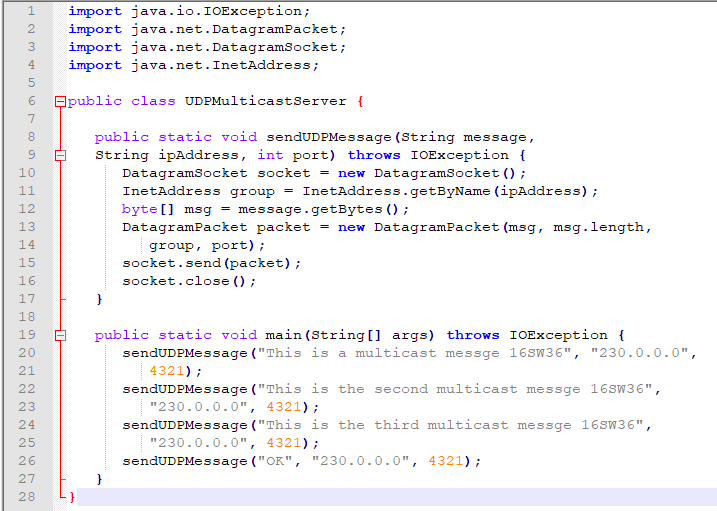
**Task 4:-Implement two simple programs using Java datagram sockets, which broadcasts and multicast your roll number to all or selected network nodes respectively.**

**UDPMulticastClient.java**

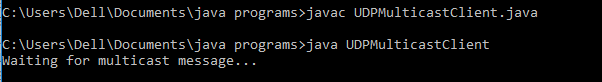


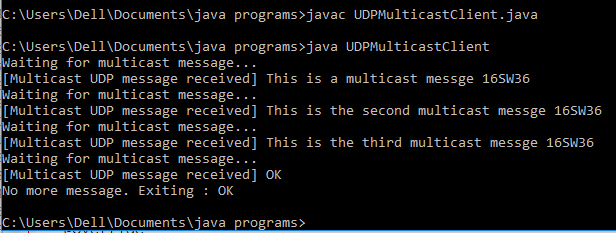


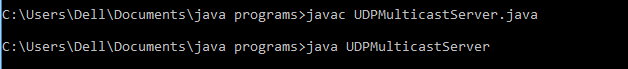
**UDPMulticastServer.java**



**OUTPUT Of Multi-cast:**

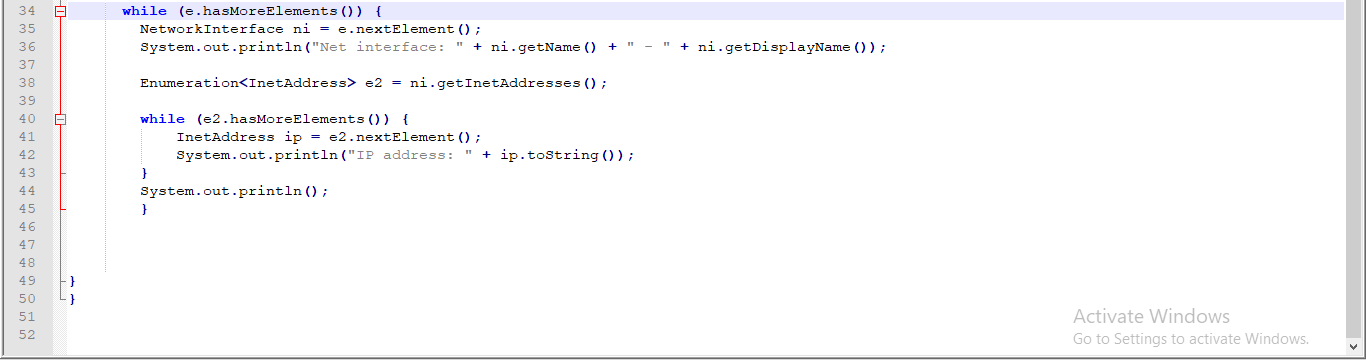






**CODE For Broadcasting Client.java**





**OUTPUT OF BROADCAST:**

