## LAB NO:01

## 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]);

finalint MAX\_LEN=20;

try

{

DatagramSocketmySocket= 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");

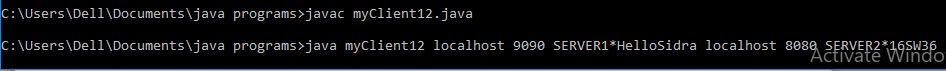
mySocket.close();

}catch(Exception ex)

{ ex.printStackTrace();}

} } }

**CLIENT OUTPUT:-**



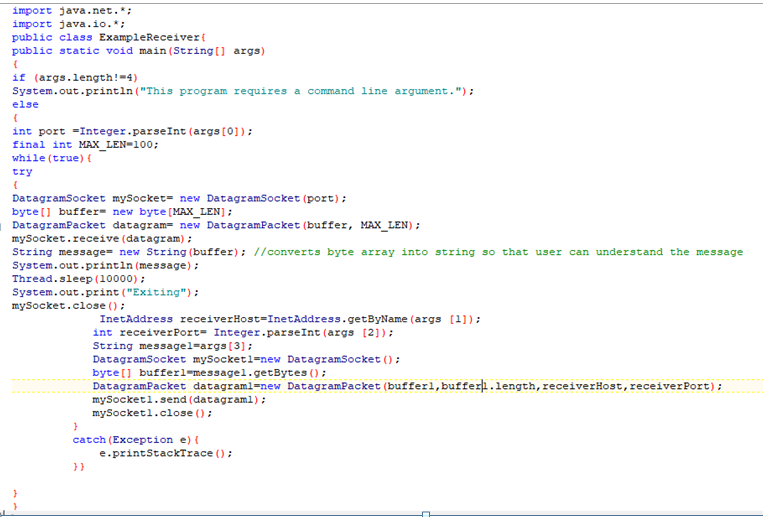
**SERVERS OUTPUT:-**



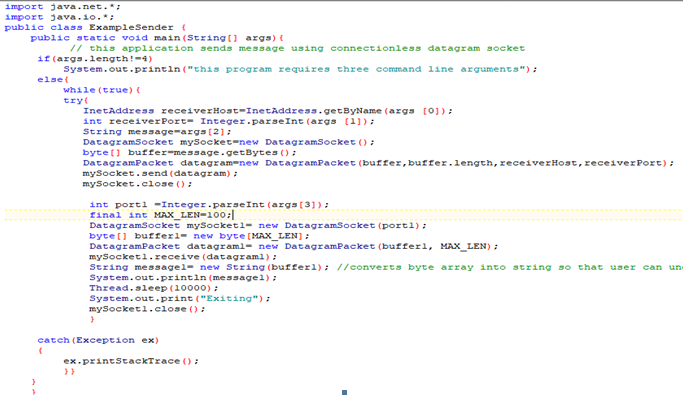
**Task#02:- Modify the sample code so that the receiver loops five times to repeatedly receive**

**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.**

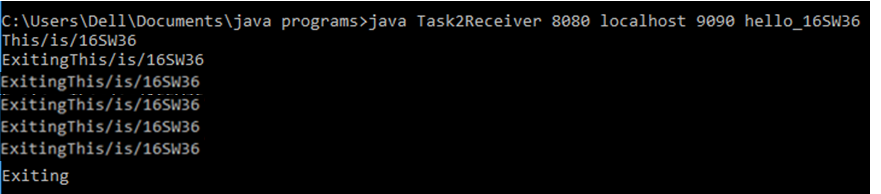
**CODE OF RECEIVER:-**



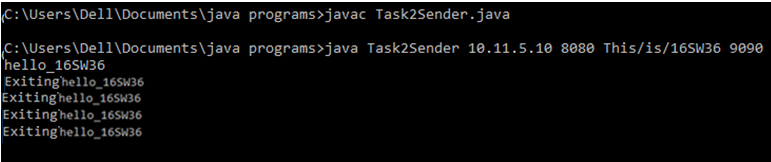
**CODE OF SENDER:-**

****

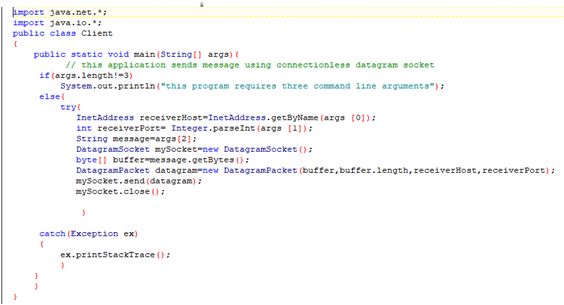
**OUTPUT OFRECEIVER:-**

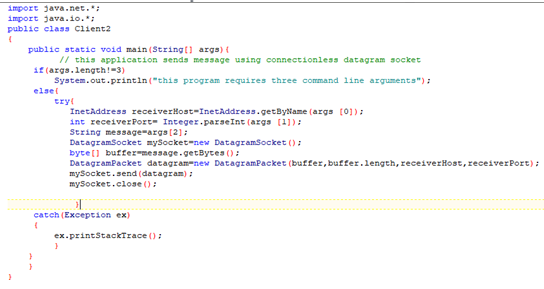


**OUTPUT OF SENDER:-**

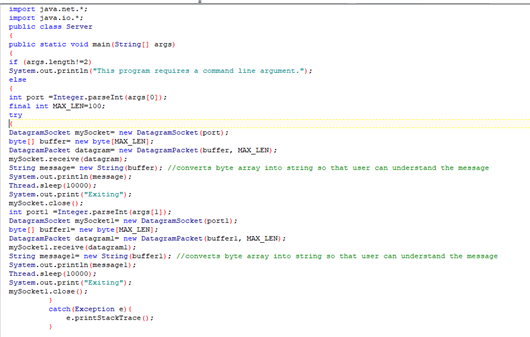


**CODEClient1**

**Client2:**



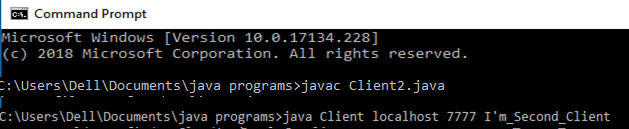
**Server:**



**OUTPUT:(Client1)**

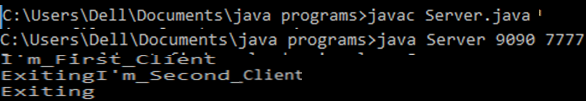


**(Client2)**



**Server:**

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



**TASK:03**

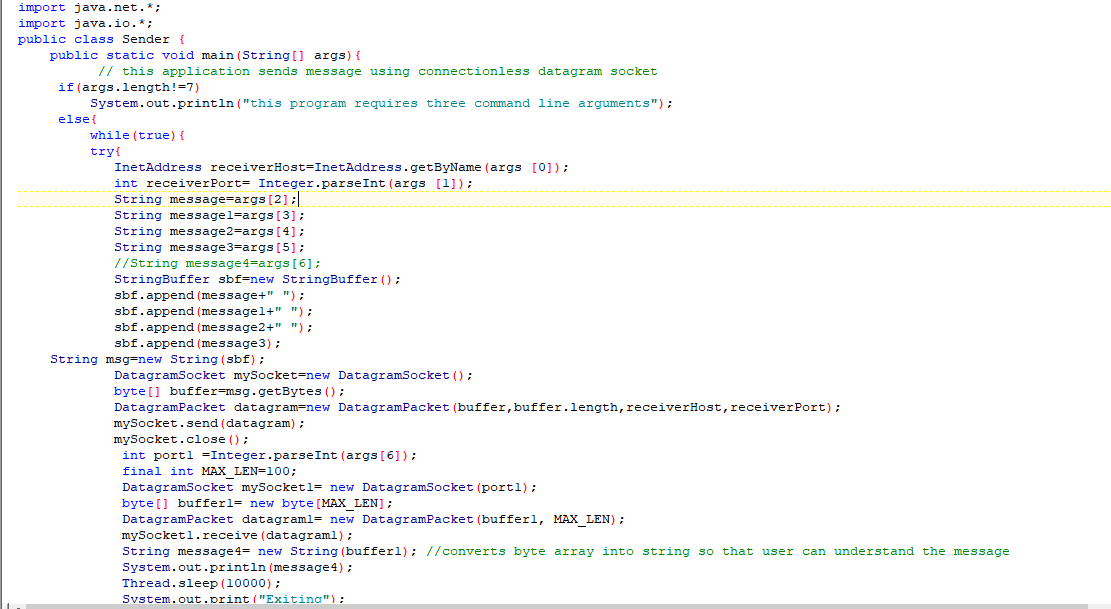
**Implement an encoding client-server application. The client sends a sentence to**

**the server using datagram packet, the server accepts it and then encodes the**

**sentence using any simple encoding algorithm (for e.g. letters reversal etc.) and**

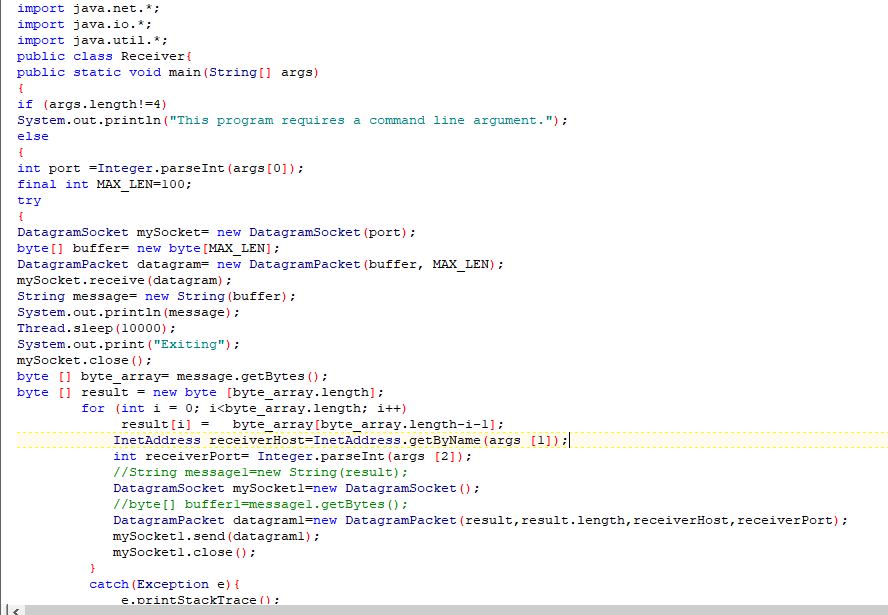
**sends the encoded sentence back to the client using a datagram packet.**

**Client:**

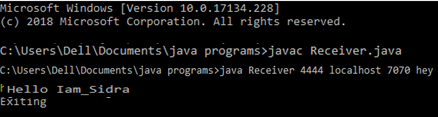




**Server**



**OUTPUT OF RECEIVER:-**



**OUTPUT OF SENDER:-**



Task:MULTICASTING

CLIENT:

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.net.MulticastSocket;

import java.lang.Thread;

public class UDPMulticastClient implements Runnable {

public static void main(String[] args) {

Thread t=new Thread(new UDPMulticastClient());

t.start();

}

public void receiveUDPMessage(String ip, int port) throws

IOException {

byte[] buffer=new byte[1024];

MulticastSocket socket=new MulticastSocket(4321);

InetAddress group=InetAddress.getByName("230.0.0.0");

socket.joinGroup(group);

while(true){

System.out.println("Waiting for multicast message...");

DatagramPacket packet=new DatagramPacket(buffer,

buffer.length);

socket.receive(packet);

String msg=new String(packet.getData(),

packet.getOffset(),packet.getLength());

System.out.println("Multicast UDP message received"+msg);

if("OK".equals(msg)) {

System.out.println("No more message. Exiting"+msg);

break;

}

}

socket.leaveGroup(group);

socket.close();

}

@Override

public void run(){

try {

receiveUDPMessage("230.0.0.0", 4321);

}catch(IOException ex){

ex.printStackTrace();

}

}}

SERVER:

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class UDPMulticastServer {

public static void sendUDPMessage(String message,

String ipAddress, int port) throws IOException {

DatagramSocket socket = new DatagramSocket();

InetAddress group = InetAddress.getByName(ipAddress);

byte[] msg = message.getBytes();

DatagramPacket packet = new DatagramPacket(msg, msg.length,

group, port);

socket.send(packet);

socket.close();

}

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

sendUDPMessage("This is 16SW06", "230.0.0.0",

4321);

sendUDPMessage("messge by 16SW06",

"230.0.0.0", 4321);

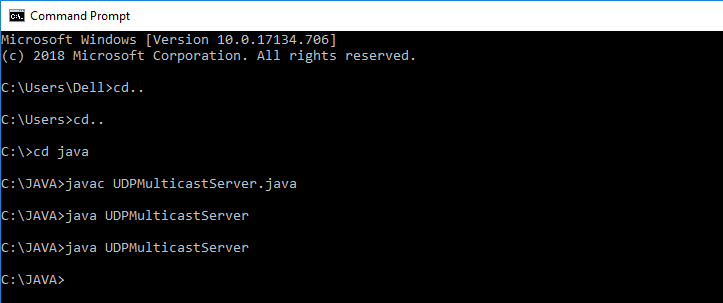
sendUDPMessage("This is a multicast messge",

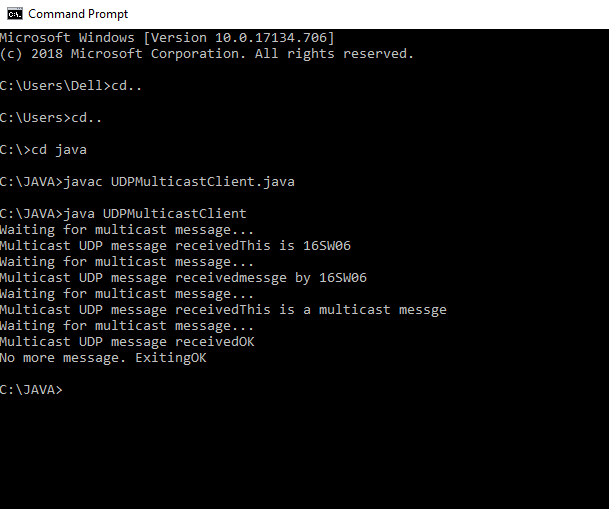
"230.0.0.0", 4321);

sendUDPMessage("OK", "230.0.0.0", 4321);

}

}





TASK :BROADCASTING

import java.net.\*;

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.MulticastSocket;

import java.net.SocketException;

import java.io.\*;

import java.lang.Thread;

import java.List;

public class BroadcastingClient {

private static DatagramSocket socket = null;

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

broadcast("Hello", InetAddress.getByName("255.255.255.255"));

}

public static void broadcast(

String broadcastMessage, InetAddress address) throws IOException {

socket = new DatagramSocket();

socket.setBroadcast(true);

byte[] buffer = broadcastMessage.getBytes();

DatagramPacket packet

= new DatagramPacket(buffer, buffer.length, address, 4445);

socket.send(packet);

socket.close();

listAllBroadcastAddresses();

}

public static void listAllBroadcastAddresses() throws SocketException {

Enumeration<NetworkInterface> interfaces

= NetworkInterface.getNetworkInterfaces();

while (interfaces.hasMoreElements()) {

NetworkInterface networkInterface = interfaces.nextElement();

System.out.println("interface"+interfaces.getNmae()) ;

Enumeration<InetAddress>e2=interfaces.getInetAddresses();

while(e2.hasMoreElements()) {

InetAddress ip=e2.nextElement();

System.out.println("ip:"+ip.toString());

}

System.out.println();

}

}

}

