Lab No: 01

Objective: To become familiar with datagramSocket programming.

Task 1: 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 the outcome.

Sender

```
import java.net.*;
import java.io.*;
public class Sender {
    public static void main(String[] args){
    if(args.length!=4)
                    System.out.println("this program requires four command li
        else{
            try{
                InetAddress receiverHost=InetAddress.getByName(args [0]);
                int receiverPort= Integer.parseInt(args [1]);
                int receiverPort2= Integer.parseInt(args [2]);
                String message=args[3];
                DatagramSocket mySocket=new DatagramSocket();
                byte[] buffer=message.getBytes();
                DatagramPacket datagram1=new DatagramPacket(buffer,buffer.len
            DatagramPacket datagram2=new DatagramPacket(buffer,buffer.length,
        mySocket.send(datagram1);
        mySocket.send(datagram2);
mySocket.close();
```

Receiver:

```
public class Receiver{
    public static void main(String[] args){
        if (args.length!=1)
            System.out.println("This program requires a command line argumen
        else{
            int port =Integer.parseInt(args[0]);
            final int MAX_LEN=10;
            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();} } } }
```

```
C:\WINDOWS\system32\cmd.exe
F:\Semesters\8th Semester\DC\labs\lab1>javac *.java
F:\Semesters\8th Semester\DC\labs\lab1>start java Receiver 102
F:\Semesters\8th Semester\DC\labs\lab1>start java Receiver 103
F:\Semesters\8th Semester\DC\labs\lab1>java Sender localhost 102 103 hello
F:\Semesters\8th Semester\DC\labs\lab1>
```

Task 2: Modify the sample code so that the receiver loops five times to repeatedly receive then display the data received. Recompile. Then
i. start the receiver
ii. Execute the sender, sending a message "message1", and
iii. In another window, start another instance of the sender, sending a message "message2". Does the receiver receive both the messages? Capture the code and output.

<u>SenderExample:</u>

```java
import java.net.*;
import java.io.*;
public class ExampleSender {
public static void main(String[] args){
  // this application sends message using connectionless datagram socket
if(args.length!=3)
 System.out.println("this program requires three command line arguments");
 else{
 try{
InetAddress receiverHost=InetAddress.getByName(args [0]);
int receiverPort= Integer.parseInt(args [1]);
String message=args[2];
DatagramSocket mySocket=new DatagramSocket();
byte[] buffer=message.getBytes();
DatagramPacket datagram1=new DatagramPacket(buffer,buffer.length,receiverHost
mySocket.send(datagram1);
mySocket.close();
 } catch(Exception e){
 e.printStackTrace(); } }  } }
```

<u>ReceiverExample:</u>

```
public class ExampleReceiver{
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=10;
try{
DatagramSocket mySocket= new DatagramSocket(port);
for(int i=1;i<=5;i++){
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(100000);
System.out.print("Exiting");
mySocket.close(); }
catch(Exception ex)
```

```
C:\WINDOWS\system32\cmd.exe

F:\Semesters\8th Semester\DC\labs\lab1\task3>javac *.java

F:\Semesters\8th Semester\DC\labs\lab1\task3>start java ExampleReceiver 1033

F:\Semesters\8th Semester\DC\labs\lab1\task3>java ExampleSender localhost 1033 H
ello

F:\Semesters\8th Semester\DC\labs\lab1\task3>java ExampleSender localhost 1033 h
ow

F:\Semesters\8th Semester\DC\labs\lab1\task3>java ExampleSender localhost 1033 a
re

F:\Semesters\8th Semester\DC\labs\lab1\task3>java ExampleSender localhost 1033 y
ou

F:\Semesters\8th Semester\DC\labs\lab1\task3>java ExampleSender localhost 1033 t
hankyou

F:\Semesters\8th Semester\DC\labs\lab1\task3>java ExampleSender localhost 1033 e
nd

F:\Semesters\8th Semester\DC\labs\lab1\task3>
```

Recompiling:

```
C:\WINDOWS\system32\cmd.exe

F:\Semesters\8th Semester\DC\labs\lab1\task3>javac *.java

F:\Semesters\8th Semester\DC\labs\lab1\task3>start java ExampleReceiver 24

F:\Semesters\8th Semester\DC\labs\lab1\task3>
F:\Semesters\8th Semester\DC\labs\lab1\task3>start java ExampleSender localhost
24 message1

F:\Semesters\8th Semester\DC\labs\lab1\task3>start java ExampleSender localhost
24 message2

F:\Semesters\8th Semester\DC\labs\lab1\task3>
```

```
C:\ProgramData\Oracle\Ja

message1
message2
```

Task3: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.

Server:

UdpServer - Notepad

File Edit Format View Help

```
// 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 UdpServer
{
        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)
                {

                        // Step 2 : create a DatgramPacket to receive the data.
                        DpReceive = new DatagramPacket(receive, receive.length);

                        // Step 3 : revieve the data in byte buffer.
                        ds.receive(DpReceive);

                        System.out.println("Client:-" + data(receive));

                        // Exit the server if the client sends "bye"
                        if (data(receive).toString().equals("bye"))
                        {
                                System.out.println("Client sent bye.....EXITING");
                                break;
                        }

                        // 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 StringBuilder 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;
        }}
```

```
C:\Users\zarmeena>cd desktop

C:\Users\zarmeena\Desktop>java UdpServer
Client:-Hello
Client:-Iam
Client:-Clent!
Client:-bye
Client sent bye.....EXITING

C:\Users\zarmeena\Desktop>
```

Client:

UdpClient - Notepad

File Edit Format View Help

```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 UdpClient
{
        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;
                DatagramSocket ds1 = new DatagramSocket(1235);

                // loop while user not enters "bye"
                while (true)
                {
                        String inp = sc.nextLine();

                        // convert the String input into the byte array.
                        buf = inp.getBytes();

                        // Step 2 : Create the datagramPacket for sending
                        // the data.
                        DatagramPacket DpSend =
                                new DatagramPacket(buf, buf.length, ip, 1234);

                        // Step 3 : invoke the send call to actually send
                        // the data.
                        ds.send(DpSend);

                        // break the loop if user enters "bye"
                        if (inp.equals("bye"))
                                break;
                }
        }
}
```

```
C:\Users\zarmeena\Desktop>java UdpClient
Hello
I am
Client!
bye

C:\Users\zarmeena\Desktop>_
```

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

File Edit Format View Help

```java
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.net.MulticastSocket;
/*  ww  w .j av  a 2 s .c  o  m*/
public class Multicast1 {
  public static void main(String[] args) throws Exception {
    int mcPort = 12345;
    String mcIPStr = "230.1.1.1";
    MulticastSocket mcSocket = null;
    InetAddress mcIPAddress = null;
    mcIPAddress = InetAddress.getByName(mcIPStr);
    mcSocket = new MulticastSocket(mcPort);
    System.out.println("Multicast Receiver running at:"
        + mcSocket.getLocalSocketAddress());
    mcSocket.joinGroup(mcIPAddress);

    DatagramPacket packet = new DatagramPacket(new byte[1024], 1024);

    System.out.println("Waiting for a  multicast message...");
    mcSocket.receive(packet);
    String msg = new String(packet.getData(), packet.getOffset(),
        packet.getLength());
    System.out.println("[Multicast  Receiver] Received:" + msg);

    mcSocket.leaveGroup(mcIPAddress);
    mcSocket.close();
  }
```

```
Multicast Receiver running at:0.0.0.0/0.0.0.0:12345
Waiting for a  multicast message...
```

MultiMsg - Notepad

File Edit Format View Help

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
/*from  www.j av a2 s. c o m*/
public class MultiMsg {
  public static void main(String[] args) throws Exception {
    int mcPort = 12345;
    String mcIPStr = "230.1.1.1";
    DatagramSocket udpSocket = new DatagramSocket();

    InetAddress mcIPAddress = InetAddress.getByName(mcIPStr);
    byte[] msg = "Hello".getBytes();
    DatagramPacket packet = new DatagramPacket(msg, msg.length);
    packet.setAddress(mcIPAddress);
    packet.setPort(mcPort);
    udpSocket.send(packet);

    System.out.println("Sent a  multicast message.");
    System.out.println("Exiting application");
    udpSocket.close();
  }
}
```

```
Multicast Receiver running at:0.0.0.0/0.0.0.0:12345
Waiting for a  multicast message...
[Multicast  Receiver] Received:Hello
```

Broadcast:

DataSocket - Notepad

File Edit Format View Help

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

public class DataSocket
{
        public static void main(String[] args) throws IOException
        {
                // Constructor to create a datagram socket
                DatagramSocket socket = new DatagramSocket();
                InetAddress address = InetAddress.getByName("localhost");
                int port = 5252;
                byte buf[] = { 12, 13 };
                byte buf1[] = new byte[2];
                DatagramPacket dp = new DatagramPacket(buf, 2, address, port);
                DatagramPacket dptorec = new DatagramPacket(buf1, 2);

                // connect() method
                socket.connect(address, port);

                // isBound() method
                System.out.println("IsBound : " + socket.isBound());

                // isConnected() method
                System.out.println("isConnected : " + socket.isConnected());

                // getInetAddress() method
                System.out.println("InetAddress : " + socket.getInetAddress());

                // getPort() method
                System.out.println("Port : " + socket.getPort());

                // getRemoteSocketAddress() method
                System.out.println("Remote socket address : " + socket.getRemoteSocketAddress());
```

```
            // getLocalSocketAddress() method
            System.out.println("Local socket address : " +
                                    socket.getLocalSocketAddress());

            // send() method
            socket.send(dp);
            System.out.println("...packet sent successfully....");

            // receive() method
            socket.receive(dptorec);
            System.out.println("Received packet data : " +
                                    Arrays.toString(dptorec.getData()));

            // getLocalPort() method
            System.out.println("Local Port : " + socket.getLocalPort());

            // getLocalAddress() method
            System.out.println("Local Address : " + socket.getLocalAddress());

            // setSOTimeout() method
            socket.setSoTimeout(50);

            // getSOTimeout() method
            System.out.println("SO Timeout : " + socket.getSoTimeout());
        }

}
```

SmallServer - Notepad

File Edit Format View Help

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
public class SmallServer {

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

        DatagramSocket ds = new DatagramSocket(5252);
        byte buf[] = new byte[2];
        byte send[] = { 13, 18 };
        DatagramPacket dp = new DatagramPacket(buf, 2);

        ds.receive(dp);

        DatagramPacket senddp = new DatagramPacket(send, 2,
                                dp.getAddress(), dp.getPort());
        ds.send(senddp);
    }

}
```

Client:

```
sBound : true
sConnected : true
netAddress : localhost/127.0.0.1
ort : 5252
emote socket address : localhost/127.0.0.1:5252
ocal socket address : /127.0.0.1:50531
..packet sent successfully....
xception in thread "main" java.net.PortUnreachableException: ICMP Port Unreachable
        at java.net.DualStackPlainDatagramSocketImpl.socketReceiveOrPeekData(Native Method)
        at java.net.DualStackPlainDatagramSocketImpl.receive0(DualStackPlainDatagramSocketImpl.java:124)
        at java.net.AbstractPlainDatagramSocketImpl.receive(AbstractPlainDatagramSocketImpl.java:143)
        at java.net.DatagramSocket.receive(DatagramSocket.java:812)
        at DataSocket.main(DataSocket.java:48)
```

BroadcastReceiver:

DataSock2 - Notepad

File Edit Format View Help

```
import java.io.IOException;
import java.net.DatagramSocket;

public class DataSock2 {

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

                // Constructor
                DatagramSocket socket = new DatagramSocket(1235);

                // setSendBufferSize() method
                socket.setSendBufferSize(20);

                // getSendBufferSize() method
                System.out.println("Send buffer size : " +
                                socket.getSendBufferSize());

                // setReceiveBufferSize() method
                socket.setReceiveBufferSize(20);

                // getReceiveBufferSize() method
                System.out.println("Receive buffer size : " +
                                socket.getReceiveBufferSize());

                // setReuseAddress() method
                socket.setReuseAddress(true);

                // getReuseAddress() method
                System.out.println("SetReuse address : " +
                                socket.getReuseAddress());

                // setBroadcast() method
                socket.setBroadcast(false);

                // getBroadcast() method
                System.out.println("setBroadcast : " +
                                socket.getBroadcast());
```

```
                // setTrafficClass() method
                socket.setTrafficClass(45);

                // getTrafficClass() method
                System.out.println("Traffic class : " +
                                socket.getTrafficClass());

                // getChannel() method
                System.out.println("Channel : " +
        ((socket.getChannel()!=null)?socket.getChannel():"null"));

                // setSocketImplFactory() method
                socket.setDatagramSocketImplFactory(null);

                // close() method
                socket.close();

                // isClosed() method
                System.out.println("Is Closed : " + socket.isClosed());

        }
}
```

```
Send buffer size : 20
Receive buffer size : 20
SetReuse address : true
setBroadcast : false
Traffic class : 44
Channel : null
Is Closed : true
```