Multi-User Program:

Server Side

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.nio.charset.StandardCharsets;



public class Server {

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

    DatagramSocket s=new DatagramSocket(1161);

    System.out.println("Server is listening..........");

    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

    String str="";


    while(!str.equals("bye"))

    {

      byte[] buffer1=new byte[20];

      DatagramPacket p2=new DatagramPacket(buffer1, buffer1.length);

      s.receive(p2);


      buffer1=p2.getData();

      String str1=new String(buffer1,StandardCharsets.UTF_8);

      System.out.println("Client"+str1);

    }

  }


}
```

Client1 Side

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;



public class Client1 {

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

  {

    InetAddress addr=InetAddress.getLocalHost();

    DatagramSocket s=new DatagramSocket(1088);

    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

    byte[] buffer=new byte[20];

    String str="";

    while(!str.equals("bye"))

    {

      System.out.println("Client_1:");

      str=br.readLine();

      str = "1:" + str ;

      buffer=str.getBytes();

      DatagramPacket p1=new DatagramPacket(buffer,buffer.length,addr,1161);

      s.send(p1);

    }

  }

}
```

Client2 Side

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;



public class Client2 {

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

  {

    InetAddress addr=InetAddress.getLocalHost();

        DatagramSocket s=new DatagramSocket(1089);

    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

    byte[] buffer=new byte[20];

    String str="";

    while(!str.equals("bye"))

    {

      System.out.println("Client_2:");

      str=br.readLine();

      str = "2:" + str ;

      buffer=str.getBytes();

      DatagramPacket p1=new DatagramPacket(buffer,buffer.length,addr,1161);

      s.send(p1);

    }

  }


}
```

****

Output:

Server Output

```
Server is listening...........

Client1:hi

Client2:hi
```

Client1 Output

```
Client_1:

hi

Client_1:
```

Client2 Output

```
Client_2:

hi

Client_2:
```

****

Peer to Peer Program:

Server Side

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.nio.charset.StandardCharsets;



public class Server {

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


    InetAddress addr=InetAddress.getLocalHost();

    DatagramSocket s=new DatagramSocket(1055);

    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

    byte[] buffer=new byte[20];

    String str="";

    while(!str.equals("bye"))

    {

      byte[] buffer1=new byte[20];

      DatagramPacket p2=new DatagramPacket(buffer1, buffer1.length);

      s.receive(p2);

      buffer1=p2.getData();

      String str1=new String(buffer1,StandardCharsets.UTF_8);

      System.out.println("Client:"+str1);

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

      str=br.readLine();

      buffer=str.getBytes();

      DatagramPacket p1=new DatagramPacket(buffer,buffer.length,addr,1080);
```

```
            s.send(p1);
        }

    }

}
```

Client Side

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.nio.charset.StandardCharsets;


public class Client {
    public static void main(String[] args) throws IOException
    {
        InetAddress addr=InetAddress.getLocalHost();
        DatagramSocket s=new DatagramSocket(1080);
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        byte[] buffer= new byte[20];
        String str="";
        while(!str.equals("bye"))
        {
            System.out.println("Client:");
            str=br.readLine();
            buffer=str.getBytes();
            DatagramPacket p1=new DatagramPacket(buffer,buffer.length,addr,1055);
            s.send(p1);
            byte[] buffer1=new byte[20];
            DatagramPacket p2=new DatagramPacket(buffer1, buffer1.length);
            s.receive(p2);
            buffer1=p2.getData();
            String str1=new String(buffer1,StandardCharsets.UTF_8);
            System.out.println("Server:"+str1);
        }
    }
}
```

****

Output:

Server Output

```
Client:hi
Server:
hi
```

Client Output

```
Client:
hi
Server:hi
Client:
```

****

Dhruvil Shah 047

F18111051 TE Comp 1

Assignment 2 (B)

Q1 Explain details about protocol required for chat.

Ans. There are two types of chat protocols :

1] Inter Relay Chat (IRC) :

It is text based protocol. TCP sockets are used for connecting. IRC operators are used to manage servers. Used for group chatting on channels also known as chat rooms. C2 server are the IRC server that contain channels.

2] XMPP (Extensible Messaging and Presence Protocol) :

Real time messaging system. TCP/IP are used for connecting. Some applications are whats app. Features such as publish / subscribe, authentication and its security uses to implement IOT.

Q2 Explain different functions in UDP

Ans. The different functions of UDP are :

1] recfrom () :

This function is similar to the read () function, but three additional arguements are required.

2] sendto () :

This function is similar to the send () function, but three additional arguements are required

3] fork ( ):
The fork () function is the only way to unix to create a new process.

Q3 Explain UDP header

Ans

8 bytes

UDP header	UDP data

Source port length	Destination port length

→ all files are of 16 bits.

1] Source port : Source port is 2 bytes. Identifies the source port number

2] Destination Port : It's 2 bytes long and identified destination port number

3] Length : It's UDP length including header and data

4] Checksum : It's a 2 byte field containing 16 bits 1's complement of the 1's complement ~~chec~~ checksum of UDP. header : pseudo header of information from the IP header and the data padded with 0 octates at the end if necessary to make a multiple of 2 octets.