Multi-User Program:

Server Side

```
import java.io.*;

import java.net.*;



public class Server

{

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

   {

      ServerSocket ss2=new ServerSocket(4445);

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


      while(true)

      {

         try

         {

            Socket s= ss2.accept();

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

            ServerThread st=new ServerThread(s);

            st.start();

         }

         catch(Exception e)

         {

            System.out.println("Connection Error");

         }

      }

   }

}


class ServerThread extends Thread
```

```
{
   String line=null;
   BufferedReader br = null;
   DataOutputStream dos=null;
   DataInputStream dis=null;
   Socket s=null;

   public ServerThread(Socket s)
   {
      this.s=s;
   }

   public void run()
   {
      try
      {
         br= new BufferedReader(new InputStreamReader(System.in));
         dos=new DataOutputStream(s.getOutputStream());
         dis=new DataInputStream(s.getInputStream());

      }
      catch(IOException e)
      {
         System.out.println("IO error in server thread");
      }

      try
      {
         line="";
         while(!line.equals("stop"))
         {
            line=dis.readUTF();
            System.out.println("client says: "+line);
            dos.flush();
         }
      }
      catch (IOException e)
      {

         System.out.println("IO Error/ Client ");
      }
      catch(NullPointerException e)
      {
         System.out.println("IO Error/ Client");
      }

      finally{
         try{
            br.close();
            dos.close();
            dis.close();
            s.close();
         }
         catch(IOException ie){
            System.out.println("Socket Close Error");
         }}
   }
}
```

Client Side

```
import java.io.*;
import java.net.*;

public class Client {

  public static void main(String args[]) throws IOException
  {
    InetAddress address=InetAddress.getLocalHost();
    Socket s1=null;
    String line=null;
    BufferedReader br=null;
    BufferedReader br1=null;
    DataOutputStream dos=null;

    try {
      s1=new Socket("localhost", 4445);
      br= new BufferedReader(new InputStreamReader(System.in));
      br1=new BufferedReader(new InputStreamReader(s1.getInputStream()));
      dos= new DataOutputStream(s1.getOutputStream());
    }
    catch (IOException e){
      e.printStackTrace();
      System.err.print("IO Exception");
    }

    System.out.println("Client Address : "+address);
    System.out.println("Enter Data to echo Server ( Enter QUIT to end):");

    String response="";
    try{
      line=br.readLine();
      while(!line.equals("stop"))
      {
         dos.writeUTF(line);
         dos.flush();
         System.out.println("Enter Data to echo Server ( Enter stop to end):");
         line=br.readLine();
      }
    }
    catch(IOException e){
      e.printStackTrace();
      System.out.println("Socket read Error");
    }
    finally{

      br1.close();
      dos.close();
      br.close();
      s1.close();
      System.out.println("Connection Closed");

    }

  }
}
```

****

Output:

Server Output

```
Server Listening......

connection Established

client says: Hi

IO Error/ Client
```

Client Output

```
Client Address : DESKTOP-08BPKQF/192.168.0.106

Enter Data to echo Server ( Enter QUIT to end):

Hi

Enter Data to echo Server ( Enter stop to end):

stop

Connection Closed
```

****

}}

Peer to Peer Program:

Server Side

```
import java.net.*;

import java.io.*;



public class Server {

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

      ServerSocket ss=new ServerSocket(3333);

      System.out.println("Waiting for client to connect....\n");

      Socket s=ss.accept();

      System.out.println("Connection established...\n");

      DataInputStream din=new DataInputStream(s.getInputStream());

      DataOutputStream dout=new DataOutputStream(s.getOutputStream());

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


      String str="",str2="";

      while(!str.equals("stop")){

         str=din.readUTF();

         System.out.println("client says: "+str);

         str2=br.readLine();

         dout.writeUTF(str2);

         dout.flush();

      }

      din.close();

      s.close();

      ss.close();

   }}
```

Client Side

```
import java.net.*;
import java.io.*;

public class Client {

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

    Socket s=new Socket("localhost",3333);
    DataInputStream din=new DataInputStream(s.getInputStream());
    DataOutputStream dout=new DataOutputStream(s.getOutputStream());
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

    String str="",str2="";

    while(!str.equals("stop")){
      str=br.readLine();
      dout.writeUTF(str);
      dout.flush();
      str2=din.readUTF();
      System.out.println("Server says: "+str2);
    }

    dout.close();
    s.close();
  }}
```

****

Output:

Server Output

```
Waiting for client to connect....

Connection established...

client says: hi
```

Client Output

```
hi
```

****

Dhruvil Shah   047
F18111051   TE Comp 1

Assignment 1 (B)

Q1 Explain TCP header

Ans

Source Port			Destination Port
Sequence Number			
Acknowledgement Number			
DO	RSV	Flags	Window
Checksum			Urgent pointer
Options			

Source port : Specifies port number of the sender
Destination port : Specifies port number of the receiver
Sequence Number : Indicates how much data is sent during TCP session.
Acknowledgement Number : This value will be sequence number plus 1
DO : Indicates length of TCP header
RSV : Reserved field.
Flags : Used to ~~ex~~ establish connections.
Window : Specifies how many bytes the receiver is willing to receive.
Checksum : To check if TCP header is ok
Urgent Pointer : Indicates where the urgent data ends.

Q2 What is bind () ?

Ans The bind () assigns a local protocol address to a socket. The bind () returns 0 if succeeds, -1 on error.
Defination of the function :
int bind (Int sockfd, const struct sock addr* servaddr,
socklen_t addrlen);
bind () allows to specify both IP address and port or neither.

Q3 What is listen () ?

Ans The listen () function converts an unconnected socket into a passive socket, indicating the kernel should accept incoming connection requests discreted to this socket.
Defination of the function :
int listen (int sockfd, int backlog);
The function listen returns 0 if it succeeds, -1 on error.