AIM: TCP client program to implement echo using well known port (Port 7).

PROGRAM:

Client Code:

```
import java.io.*;
import java.net.*;
private void CSUBMITActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
      String cipaddr = CIPADDR.getText();
      Integer cportno = Integer.parseInt(CPORTNO.getText());
      String cmsg = CMSG.getText();
       Socket s = new Socket(cipaddr, cportno);
      DataInputStream dis = new DataInputStream(s.getInputStream());
      DataOutputStream dos = new DataOutputStream(s.getOutputStream());
      dos.writeUTF(cmsg);
      String newStr = dis.readUTF();
      CRESPONSE.append("/n" + newStr);
    } catch (Exception e) {
      e.printStackTrace();
    }
```

OUTPUT:

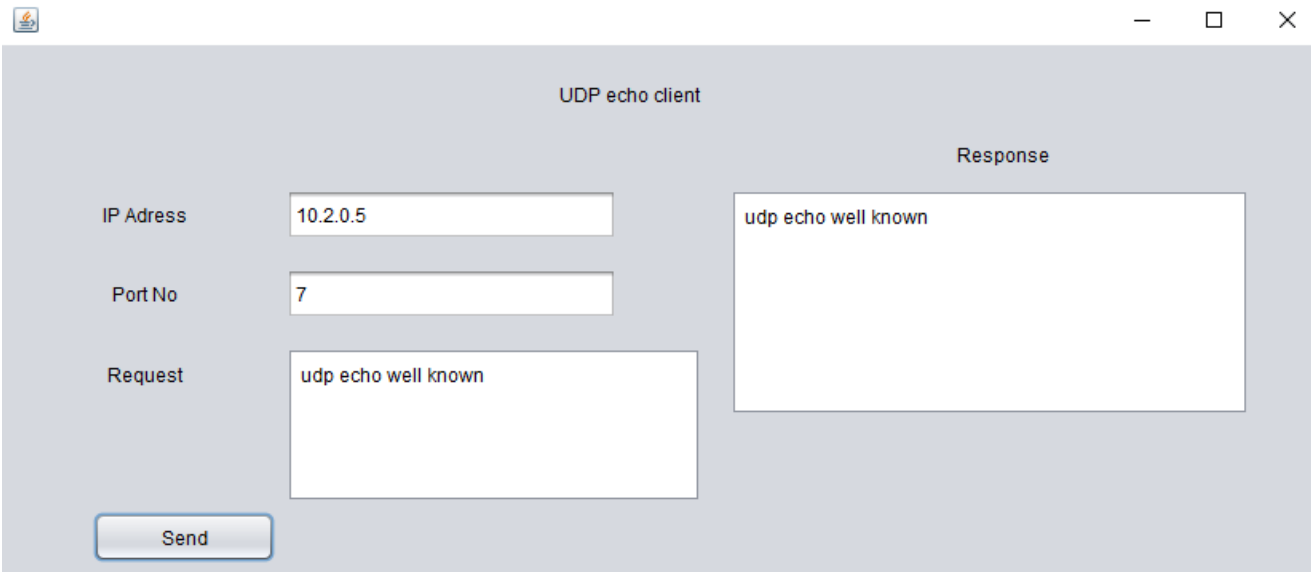
AIM: UDP client program to implement echo using well known port (Port 7).

PROGRAM:

Client Code:

```
import java.io.*;
import java.net.*;
private void sendActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String ip = ipadd.getText();
    int port = Integer.parseInt(portno.getText());
    String msg = jTextArea2.getText();

    try{
      DatagramSocket s = new DatagramSocket();
      byte msg1[] = msg.getBytes();
      InetAddress ia = InetAddress.getByName(ip);
      byte resp[] = new byte[255];
      DatagramPacket dps = new DatagramPacket(msg1, msg1.length, ia, port);
      DatagramPacket dpr = new DatagramPacket(resp, resp.length, ia, port);

      s.send(dps);
      s.receive(dpr);

      byte res[] = dpr.getData();
      String response = new String(res);
      msg2.append(response+"\n");
      jTextArea2.setText(null);
      s.close();

    }catch(Exception e){
      e.printStackTrace();
    }
  }
```

OUTPUT:


UDP echo client
Response
IP Adress
10.2.0.5
Port No
7
Request
udp echo well known
udp echo well known
Send

AIM: TCP client server program to implement echo server.

PROGRAM:

Client Code:

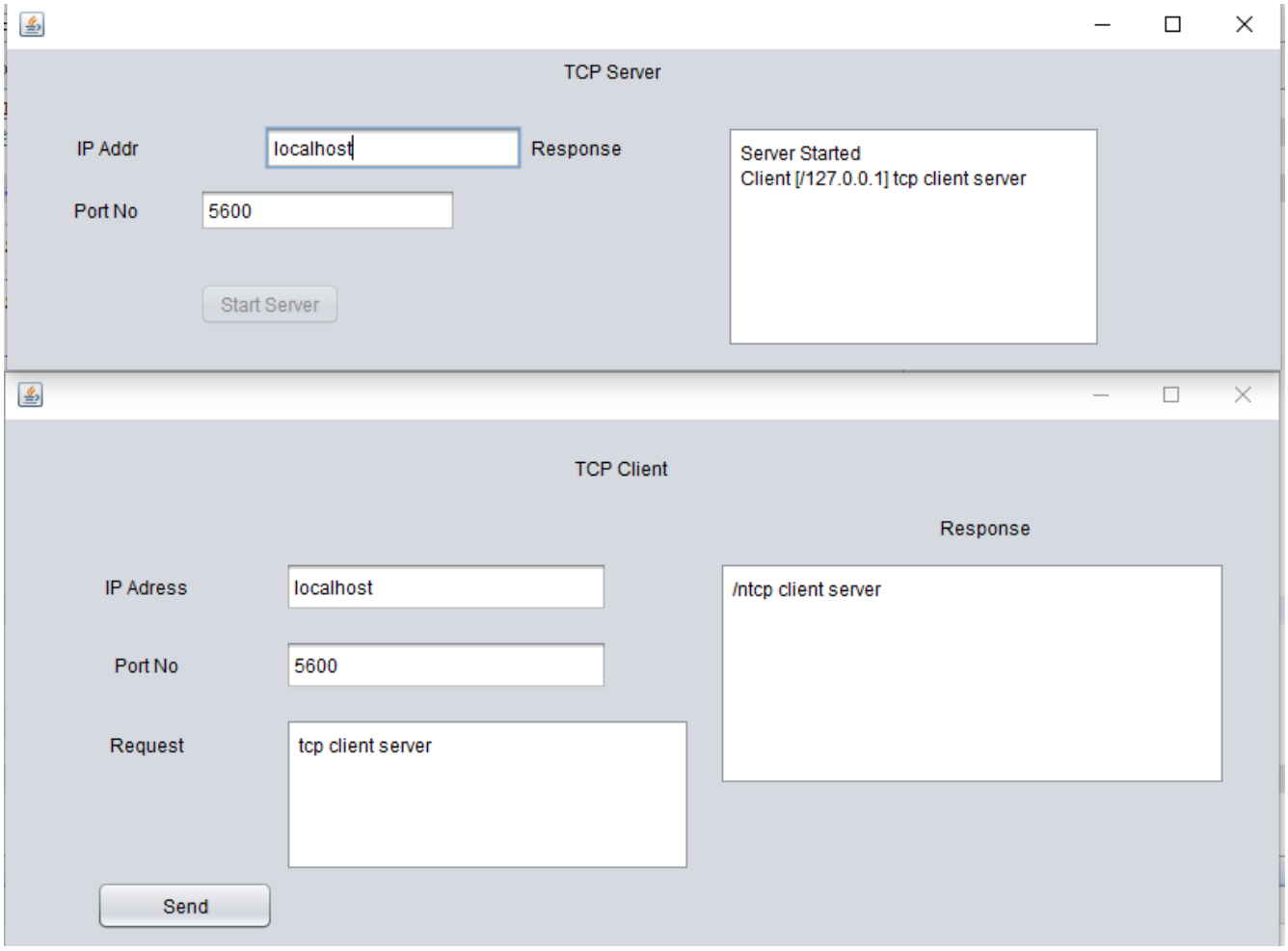
```
import java.io.*;
import java.net.*;
private void CSUBMITActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
      String cipaddr = CIPADDR.getText();
      Integer cportno = Integer.parseInt(CPORTNO.getText());
      String cmsg = CMSG.getText();
       Socket s = new Socket(cipaddr, cportno);
      DataInputStream dis = new DataInputStream(s.getInputStream());
      DataOutputStream dos = new DataOutputStream(s.getOutputStream());
      dos.writeUTF(cmsg);
      String newStr = dis.readUTF();
      CRESPONSE.append("/n" + newStr);
    } catch (Exception e) {
      e.printStackTrace();
    }
```

SERVER CODE :

```
public class Server extends javax.swing.JFrame implements Runnable {
   public void run(){
    String  sipaddr = SIPADDR.getText();
    int sportno = Integer.parseInt(SPORTNO.getText());
    try{
      ServerSocket ss = new ServerSocket(sportno, 5,
InetAddress.getByName(sipaddr));
      Socket s = ss.accept();
      DataInputStream dis = new DataInputStream(s.getInputStream());
      DataOutputStream dos = new DataOutputStream(s.getOutputStream());
       String req = dis.readUTF();
      SMSG.append("Client [" + s.getInetAddress() + "] " + req);
      dos.writeUTF(req);
      s.close();
    }
     catch(Exception e){
      e.printStackTrace();
```

```
        }
    }
    private void SBUTTONActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Thread t = new Thread(this, "server");
        t.start();
        SBUTTON.setEnabled(false);
        SMSG.append("Server is Listening \n");
    }
```

OUTPUT:



AIM: UDP client server program to implement echo server.

PROGRAM:

Client Code:

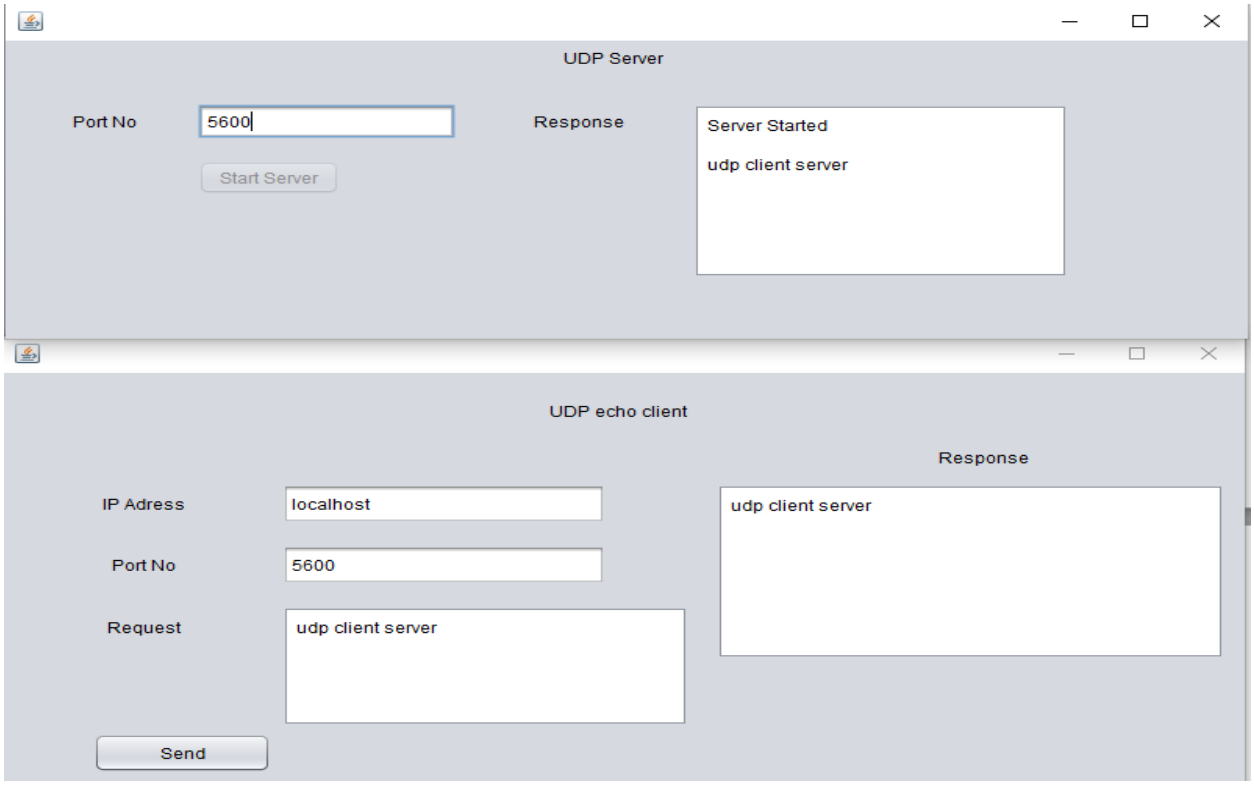
```
import java.io.*;
import java.net.*;
private void sendActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String ip = ipadd.getText();
    int port = Integer.parseInt(portno.getText());
    String msg = jTextArea2.getText();

    try{
      DatagramSocket s = new DatagramSocket();
      byte msg1[] = msg.getBytes();
      InetAddress ia = InetAddress.getByName(ip);
      byte resp[] = new byte[255];
      DatagramPacket dps = new DatagramPacket(msg1, msg1.length, ia, port);
      DatagramPacket dpr = new DatagramPacket(resp, resp.length, ia, port);

      s.send(dps);
      s.receive(dpr);

      byte res[] = dpr.getData();
      String response = new String(res);
      msg2.append(response+"\n");
      jTextArea2.setText(null);
      s.close();

    }catch(Exception e){
      e.printStackTrace();
    }
  }
```

Server Code:

```
import java.net.*;
import java.util.*;
public void run(){
    int port = Integer.parseInt(server_port.getText());
    try{
```

```
            DatagramSocket ss = new DatagramSocket(port);
            while(true){
            byte[] msg = new byte[255];
            DatagramPacket dps = new DatagramPacket(msg,msg.length);
            ss.receive(dps);
            String a = new String(msg);
            ss.send(dps);
            response_area.append("\n"+a);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
private void start_serverActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Thread t = new Thread(this,"t1");
        t.start();
        response_area.append("Server Started\n");
        start_server.setEnabled(false);
    }
```

OUTPUT:



AIM: To perform Chat Server Bulletin Board application.

PROGRAM:

Client Code:

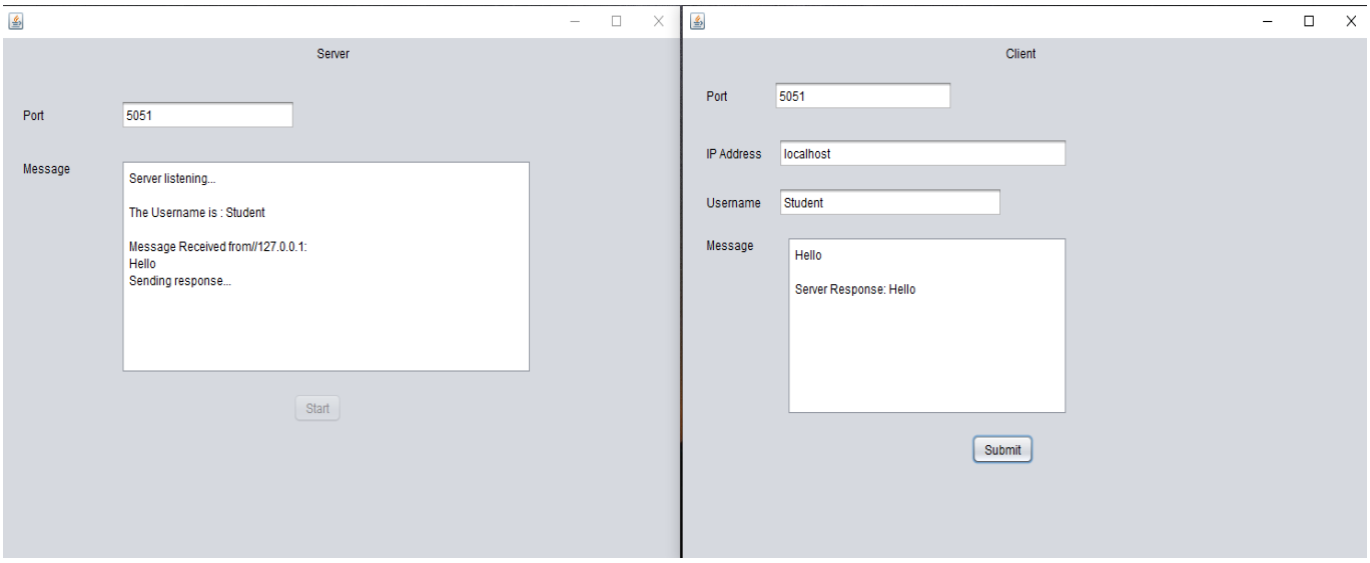
```
private void SubmitActionPerformed(java.awt.event.ActionEvent evt) {try{
            String ip = IP.getText();
            Integer port = Integer.parseInt(CPort.getText());Socket s = new Socket(ip,port);
            DataInputStream dis = new DataInputStream(s.getInputStream()); DataOutputStream
            dos = new DataOutputStream(s.getOutputStream());String st = Username.getText();
            dos.writeUTF(st);
            String str = CMessage.getText();
            dos.writeUTF(str);
            String newStr = dis.readUTF(); CMessage.append("\n"+newStr);s.close();
        }
        catch(Exception e){ e.printStackTrace();
        }       // TODO add your handling code here:
    }
```

Server Code:

```
public class Server extends javax.swing.JFrame implements Runnable {public void run(){
    try{
        Integer port = Integer.parseInt(SPort.getText());ServerSocket ss = new ServerSocket(port);
        while(true){
            Socket s = ss.accept();
            DataInputStream dis = new DataInputStream(s.getInputStream()); DataOutputStream
            dos = new DataOutputStream(s.getOutputStream());String st = dis.readUTF();
            String str = dis.readUTF(); SMessage.append("\nThe Username is : "+st+"\n");
            SMessage.append("\nMessage Received
from/"+s.getInetAddress().toString()+":\n"+str+"\nSending response...\n");
            dos.writeUTF("\nServer Response: "+str);s.close();
        }
    }
    catch(Exception e){ e.printStackTrace();
```

```
    }
  }
private void StartActionPerformed(java.awt.event.ActionEvent evt) {Thread t = new
        Thread(this,"t1");
        t.start();
        SMessage.append("Server listening...\n");Start.setEnabled(false);
    // TODO add your handling code here:
  }
```

OUTPUT:



AIM: Program to demonstrate Domain Name Server.

PROGRAM:

i. Client Code:

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

private void SubmitActionPerformed(java.awt.event.ActionEvent evt) {try{

        String ip =IP.getText();

        int port =Integer.parseInt(CPort.getText());Socket s=new Socket(ip,port);

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

        DataOutputStream dos=new DataOutputStream(s.getOutputStream());String

        domain=DName.getText();

        dos.writeUTF(domain);

        String response=dis.readUTF(); Message.append(" "+response+"\n");s.close();

    }

    catch(Exception e)
                                        {

        e.printStackTrace();
                                        }
```

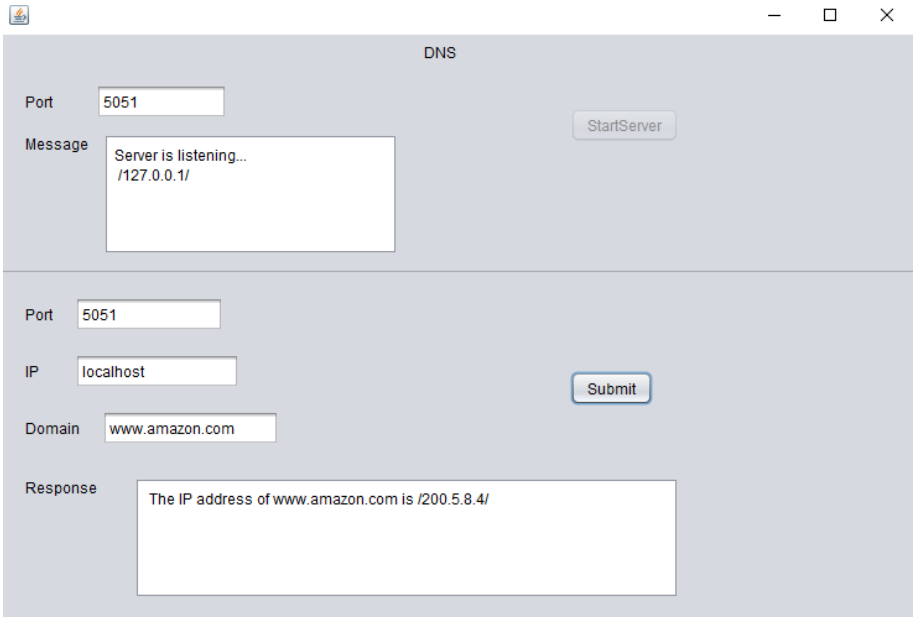
ii. Server Code:

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

public class dnsserver extends javax.swing.JFrame implements Runnable {public void run()

  {

    try{

    int sPort=Integer.parseInt(SPort.getText());ServerSocket ss=new ServerSocket(sPort);

    //binded server socket - listens for connectionswhile(true)

     {

        Socket s=ss.accept();

    //client's request has come; connection is established

                /* Getting I/O Streams */

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

        DataOutputStream dos=new DataOutputStream(s.getOutputStream());
```

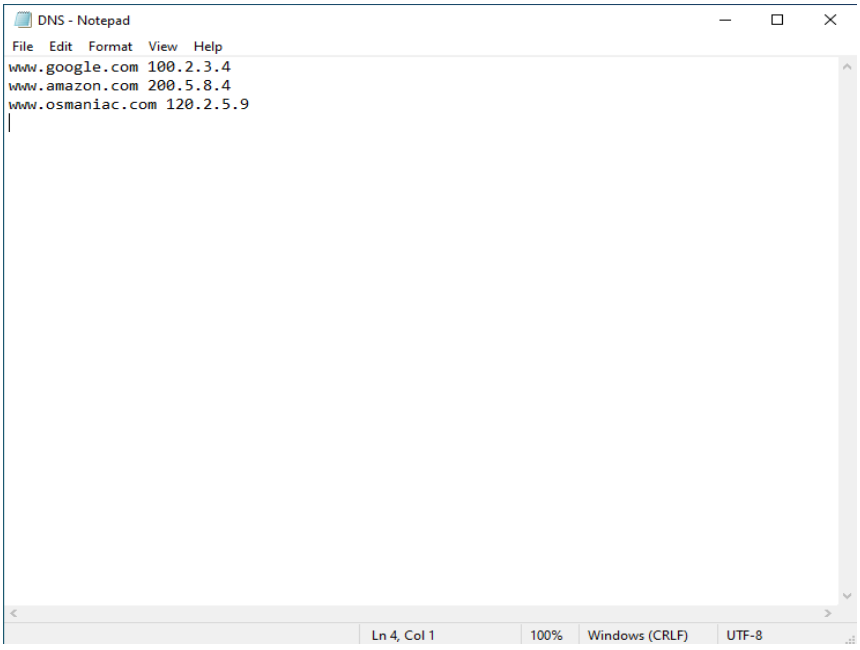
```
            //Get the request
            String req=dis.readUTF();
              Smessage.append(" " +s.getInetAddress().toString()+"/");
 //displaying from which client what domain request is coming
            /* READING FROM FILE */
            try{
                BufferedReader br=new BufferedReader(new InputStreamReader(new
FileInputStream("DNS.txt")));
                String fInput=br.readLine();int flag=0; while(fInput!=null)
                {
                   StringTokenizer stk=new StringTokenizer(fInput); //tokensString
                   dname=stk.nextToken();
                   String dIP=stk.nextToken();if(req.equals(dname))
                   {
                      dos.writeUTF(dname+" " +"IP is "+"/"+dIP+"/"+"\n");flag=1;
                   }
                   fInput=br.readLine();
                }
                if(flag==0)
                   dos.writeUTF(req+"/NOT FOUND");
             }
             catch(Exception e)
             {
                e.printStackTrace();
             }
          }
       }
       catch(Exception e)
       {
          e.printStackTrace();
       }
    }
private void StartActionPerformed(java.awt.event.ActionEvent evt) {
       // TODO add your handling code here:
```

```
        Thread t=new Thread(this,"ns");t.start();

        Start.setEnabled(false); Smessage.append("Server is listening...\n");

    }
```

OUTPUT:



Input File:



AIM: Program to design Chat Application using Client and Server

PROGRAM:

i. Client Code:

```
import java.io.*; import java.net.*;try{
            String ip = IP.getText();
            Integer port = Integer.parseInt(Port.getText());Socket s = new Socket(ip,port);
            DataInputStream dis = new DataInputStream(s.getInputStream()); DataOutputStream dos
            = new DataOutputStream(s.getOutputStream());String str = Cmessage.getText();
            dos.writeUTF(str);
            String newStr = dis.readUTF(); Cmessage.append("\n"+newStr);s.close();
        }
        catch(Exception e){ e.printStackTrace();
        }
```

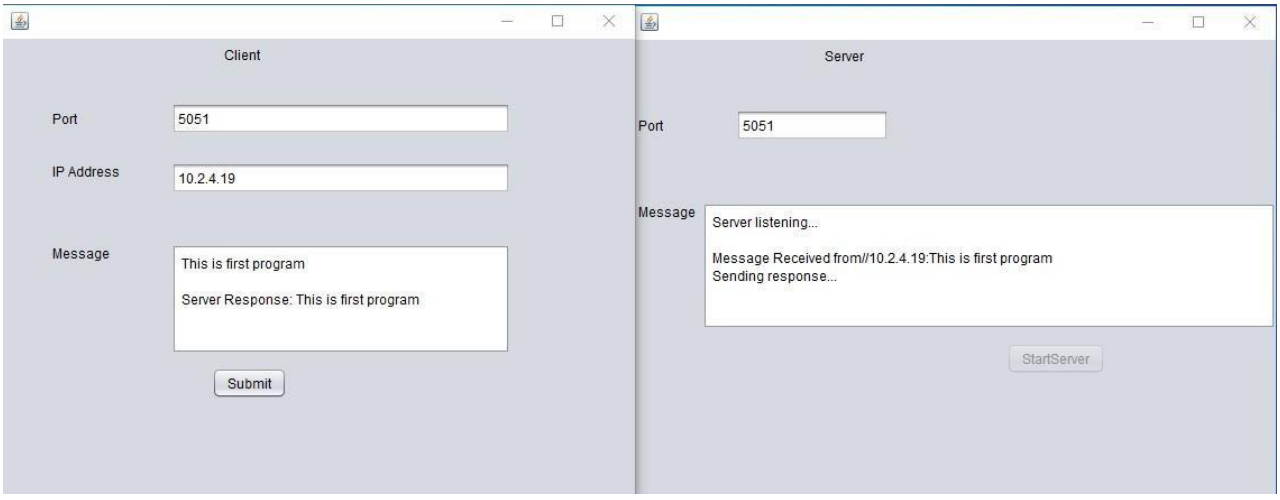
ii. Server Code:

```
import java.io.*; import java.net.*;
public class Server extends javax.swing.JFrame implements Runnable {public void run(){
    try{
        Integer port = Integer.parseInt(SPort.getText());ServerSocket ss = new ServerSocket(port);
        while(true){
            Socket s = ss.accept();
            DataInputStream dis = new DataInputStream(s.getInputStream()); DataOutputStream dos
            = new DataOutputStream(s.getOutputStream());String str = dis.readUTF();
            Smessage.append("\nMessage Received
         from/"+s.getInetAddress().toString()+":"+str+"\nSending response...\n");
            dos.writeUTF("\nServer Response: "+str);s.close();
        }
    }
    catch(Exception e){ e.printStackTrace();
```

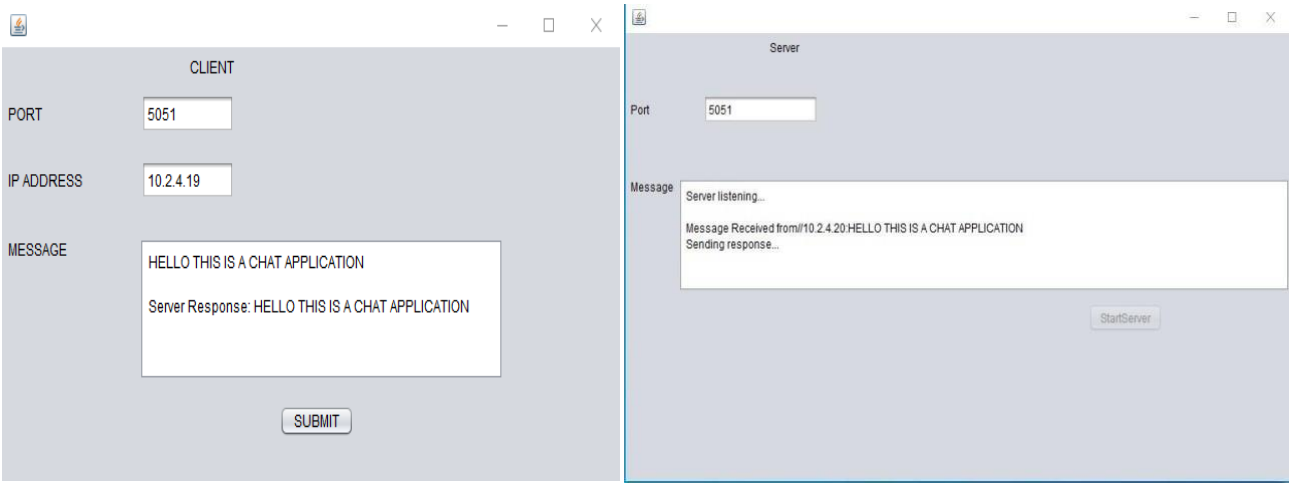
```
    }
  }
private void ReceiveActionPerformed(java.awt.event.ActionEvent evt) {Thread t = new
        Thread(this,"t1");
        t.start();
        Smessage.append("Server listening...\n"); Receive.setEnabled(false);// TODO add your
        handling code here:
  }
```

OUTPUT: Execution Steps

a. Executing from same system:



b. Executing from remote system:



AIM : Program to implement RPC application for Echo Message

PROGRAM:

Echos.x

```
program ECHOSERVER_PROGRAM
{
        version ECHOSERVER_VERSION
        {
                string ECHO(string) = 1;
    }=1;
}=0x21234589;
```

Execution command

Command : $ rpcgen -a echos.x

$ ls

i. Echo_client.c

```
#include "echos.h"

Void echoserver_program_1(char *host)
{
    CLIENT *clnt; char * *result_1;
    char * echo_1_arg;#ifndef DEBUG
 clnt = clnt_create (host, ECHOSERVER_PROGRAM, ECHOSERVER_VERSION, "udp");if
    (clnt == NULL) {
        clnt_pcreateerror (host);exit (1);
      }
    #endif /* DEBUG */ echo_1_arg=(char *)malloc(20);printf("\n Enter a message:");
    scanf("%s",echo_1_arg);
    result_1 = echo_1(&echo_1_arg, clnt);if (result_1 == (char **) NULL) {
        clnt_perror (clnt, "call failed");}
    else
        printf("\n The message returned is %s",*result_1);#ifndef DEBUG
    clnt_destroy (clnt);#endif  /* DEBUG */
}
Int  main (int argc, char *argv[])
    {   char *host; if (argc < 2) {
            printf ("usage: %s server_host\n", argv[0]);exit (1);
          }
          host = argv[1]; echoserver_program_1 (host);
    exit (0);}
```

iii.Echos_server.c

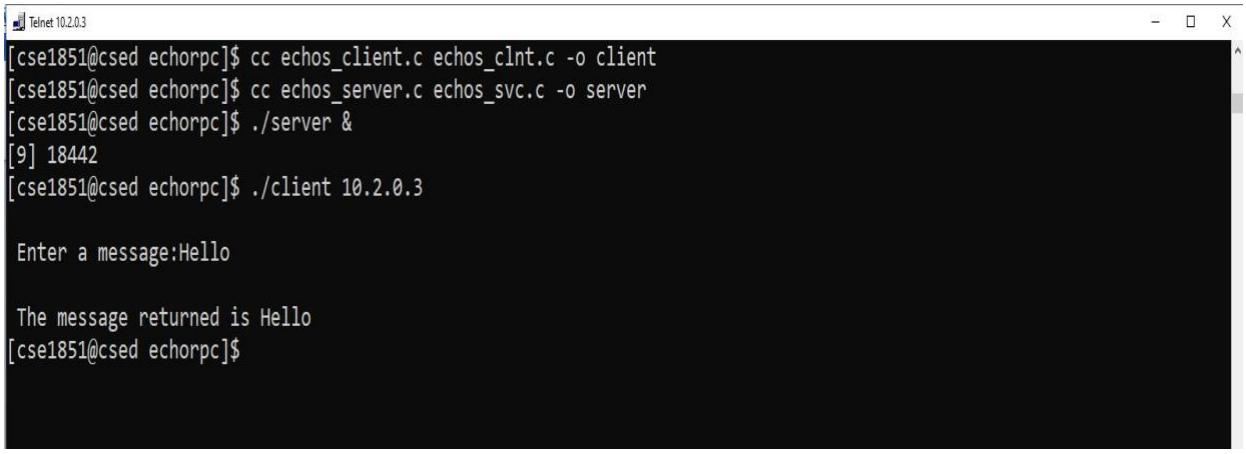
```
#include "echos.h"char **
```

```
echo_1_svc(char **argp, struct svc_req *rqstp)
{
    static char * result;
    / * insert server code here
     */ result=*argp; return &result;
}
```

Execution Steps:

```
$ cc echos_client.c echos_clnt.c -o client

$ cc echos_server.c echos_svc.c -o server
```

Output:



AIM : RPC program to add two numbers

PROGRAM:

Vi add.x

```
struct num{
        int a;
        int b;
};
program add_prog{
version add_ver{
        int addition(num)=1;
}=1;
}=0x20000002;
```

Next compile the program

$ rpcgen –a add.x

Vi add_client.c

```
#include "add.h"
Void add_prog_1(char *host)
{         CLIENT *clnt;
        int  *result_1;
        num  addition_1_arg;
#ifndef         DEBUG
        clnt = clnt_create (host, add_prog, add_ver, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);    }
#endif          /* DEBUG */
        printf("\n enter the two number to add...\n"); // reading 2 numbers for addition
        scanf("%d%d",&addition_1_arg.a,&addition_1_arg.b); // assigned readed number
        result_1=(int *) malloc(sizeof(int)); // allocate memroy
        result_1 = addition_1(&addition_1_arg, clnt);
        if (result_1 == (int *) NULL) {
                clnt_perror (clnt, "call failed");      }
#ifndef         DEBUG
        printf("\n the of %d\t%d is ..... %d\n",addition_1_arg.a,addition_1_arg.b,*result_1);
        clnt_destroy (clnt);
#endif           /* DEBUG */     }
int
main (int argc, char *argv[])
{
        char *host;
        if (argc < 2) {
                printf ("usage: %s server_host\n", argv[0]);
                exit (1); }
        host = argv[1];
        add_prog_1 (host);
exit (0);
}
```

Vi add_server.c

```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */
#include "add.h"
int *
addition_1_svc(num *argp, struct svc_req *rqstp)
{
        static int  result;
        /*
         * insert server code here
         */
        result = argp->a + argp->b; // adding numbers
        return &result;
}
```

Next compile the code using the command

$cc –o add_client.c add_clnt.c –lnsl addclient

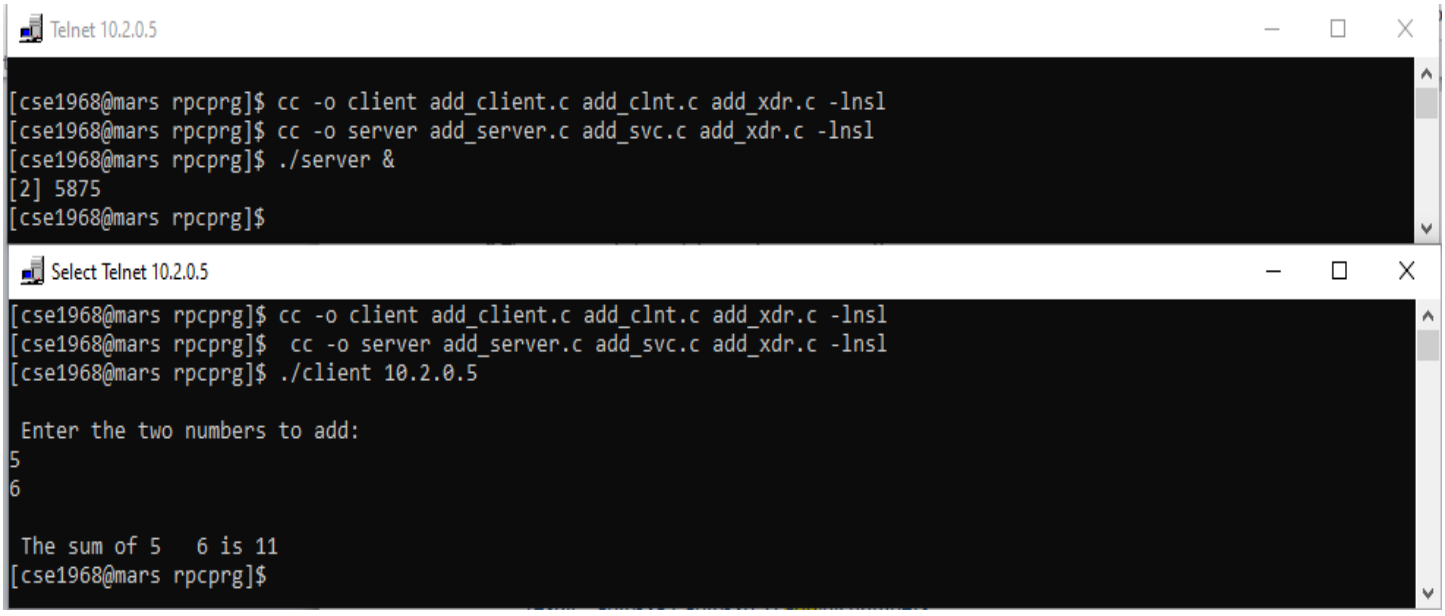
$cc –o add_server.c add_svc.c –lnsl addserver

After successfully compilation execut the program by using command

$./addserver &

$./addclient 10.2.0.3

OUTPUT:



AIM : RPC program to find GCD of two numbers

PROGRAM:

Vi gcd.x

```
struct num
{
long a;
long b;
};
program gcd_prog{
 version gcd_vers{
long gcd_fn(num)=1;
}=1;
}=0x30000001;
```

Execution: $ rpcgen gcd.x

```
$ ls
client        Echos.h       Echos.x     gcd_svc.c  Makefile.Echos
Echos_client.c  Echos_server.c  gcd_clnt.c  gcd.x     server
Echos_clnt.c    Echos_svc.c     gcd.h       gcd_xdr.c
```

GCD CLIENT CODE

Vi gcd_client.c

```
#include "gcd.h"

void
gcd_prog_1(char *host,num number)
{
	CLIENT *clnt;
	long  *result_1;
	num  gcd_fn_1_arg;

	gcd_fn_1_arg.a=number.a;
	gcd_fn_1_arg.b=number.b;
#ifndef DEBUG
	clnt = clnt_create (host, gcd_prog, gcd_vers, "udp");
	if (clnt == NULL) {
		clnt_pcreateerror (host);
		exit (1);
	}

#endif  /* DEBUG */

	result_1 = gcd_fn_1(&gcd_fn_1_arg, clnt);
	if (result_1 == (long *) NULL) {
		clnt_perror (clnt, "call failed");
	}

printf("gcd is %d",*result_1);
#ifndef DEBUG
	clnt_destroy (clnt);
```

```
#endif   /* DEBUG */
}


int
main (int argc, char *argv[])
{
	char *host;

num n;
	if (argc < 2) {
		printf ("usage: %s server_host\n", argv[0]);
		exit (1);

	}
	host = argv[1];
n.a=atol(argv[2]);
	n.b=atol(argv[3]);

	gcd_prog_1 (host,n);
exit (0);
}
```

Server code

Vi gcd_server.c

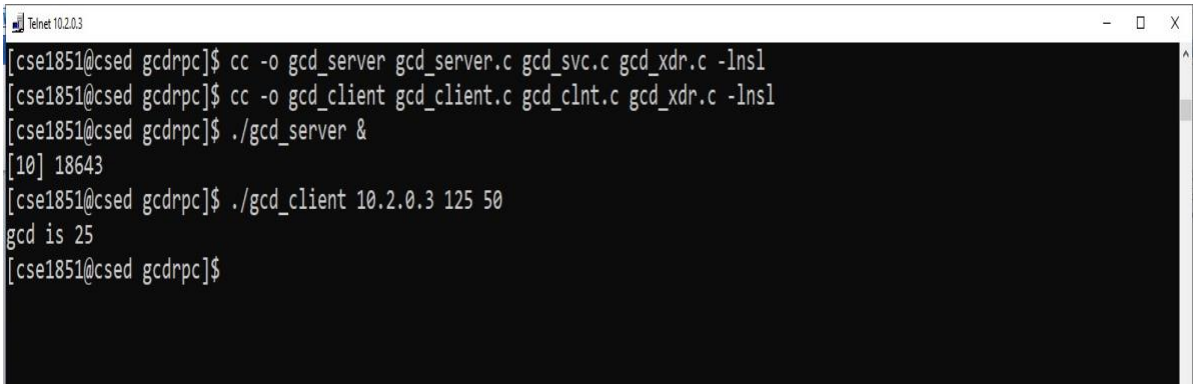
```
int gcd(int a ,int b){
	if (b==0)
	return a;
	return gcd(b,a%b);}

#include "gcd.h"

long *
gcd_fn_1_svc(num *argp, struct svc_req *rqstp)
{
	static long  result;

	/*
	 * insert server code here
	 */

result=gcd((*argp).a,(*argp).b);

	return &result;
}
```

Execution :-

```
$cc -o gcd_server gcd_server.c gcd_svc.c gcd_xdr.c –lnsl
$cc -o gcd_client gcd_client.c gcd_clnt.c gcd_xdr.c –lnsl
```

$./gcd_server &
[4] 5086
$ ./gcd_client 10.2.0.5 10 12
gcd is 2

OUTPUT:



AIM: Program to demonstrate echo message using RMI

PROGRAM:

i. HelloInterface

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface HelloInterface extends Remote{
		String helloMsg(String s) throws RemoteException;
}
```

ii. HelloImpl

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class HelloImpl extends UnicastRemoteObject implements HelloInterface{
	public HelloImpl() throws RemoteException{
	}
	public String helloMsg(String s1)
	{
		System.out.println("REMOTE SERVICE: Remote Client REquest Message is :
"+s1);
		StringBuilder sb=new StringBuilder(s1);
		String response=sb.reverse().toString();
		return response;
	}
}
```

iii. HelloServer

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.Naming;
import java.net.MalformedURLException;
import java.rmi.registry.LocateRegistry;

public class HelloServer{
	public HelloServer() throws RemoteException{
	}
	public static void main(String args[]) throws RemoteException
	{
		HelloImpl hiObj=new HelloImpl();
		int port=Integer.parseInt(args[1]);
		try{
			LocateRegistry.createRegistry(port);
			System.out.println("\n RMI registry created \n");
			String host=args[0];
```

```
                        String bindLocation="//"+host+":"+port+"/"+args[2];
                        Naming.bind(bindLocation,hiObj);
                        System.out.println("\nRMI server ready at "+bindLocation);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

iv. HelloClient

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

public class HelloClient {
        public static void main(String args[])
        {
                String
connectLocation="//"+args[0]+":"+Integer.parseInt(args[1])+"/"+args[2];
                HelloInterface hintf=null;
                try{
                        System.out.println("\n Connecting the client at:
"+connectLocation);
                        hintf=(HelloInterface)Naming.lookup(connectLocation);
                        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
                        System.out.println("\nCLI: Enter the request message to send to
remote service:");
                        String s=br.readLine();
                        String response=hintf.helloMsg(s);
                        System.out.println("\n CLI: response from remote method
is:"+response);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

Execution Steps
C: javac *.java
C: rmic AddServerImpl
C: start rmiregistry
C: java AddServer
C: java AddClient localhost 30 23

OUTPUT:

Command Prompt

```
C:\Users\Student>cd Desktop

C:\Users\Student\Desktop>cd Raj116-20230110T085513Z-001

C:\Users\Student\Desktop\Raj116-20230110T085513Z-001>cd Raj116

C:\Users\Student\Desktop\Raj116-20230110T085513Z-001\Raj116>cd src

C:\Users\Student\Desktop\Raj116-20230110T085513Z-001\Raj116\src>cd rmi

C:\Users\Student\Desktop\Raj116-20230110T085513Z-001\Raj116\src\RMI>javac Hello
Client.java

C:\Users\Student\Desktop\Raj116-20230110T085513Z-001\Raj116\src\RMI>java HelloC
lient 10.2.4.17 5085 helloservice

 Connecting the client at: //10.2.4.17:5085/helloservice

CLI: Enter the request message to send to remote service:
hello

 CLI: response from remote method is:hello

C:\Users\Student\Desktop\Raj116-20230110T085513Z-001\Raj116\src\RMI>
```

Command Prompt - java HelloServer 10.2.4.17 5085 helloservice

```
        at java.base/java.net.DualStackPlainSocketImpl.socketConnect(DualStackPlainSocke
tImpl.java:79)
        at java.base/java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.
java:400)
        at java.base/java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSock
etImpl.java:243)
        at java.base/java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.ja
va:225)
        at java.base/java.net.PlainSocketImpl.connect(PlainSocketImpl.java:148)
        at java.base/java.net.SocksSocketImpl.connect(SocksSocketImpl.java:402)
        at java.base/java.net.Socket.connect(Socket.java:591)
        at java.base/java.net.Socket.connect(Socket.java:540)
        at java.base/java.net.Socket.<init>(Socket.java:436)
        at java.base/java.net.Socket.<init>(Socket.java:213)
        at java.rmi/sun.rmi.transport.tcp.TCPDirectSocketFactory.createSocket(TCPDirectS
ocketFactory.java:40)
        at java.rmi/sun.rmi.transport.tcp.TCPEndpoint.newSocket(TCPEndpoint.java:613)
        ... 6 more

C:\Users\Student\Desktop\Raj116-20230110T085513Z-001\Raj116\src\RMI>

C:\Users\Student\Desktop\Raj116-20230110T085513Z-001\Raj116\src\RMI>java HelloServer 10.
2.4.17 5085 helloservice

 RMI registry created


RMI server ready at //10.2.4.17:5085/helloservice
REMOTE SERVICE: Remote Client REquest Message is : hello
```

AIM: Program to reverse a string using RMI

PROGRAM:

i. hellointf

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface hellointf extends Remote {

	String hellomsg(String s) throws RemoteException;

}
```

ii. helloimpl

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

import java.lang.String;

class helloimpl extends UnicastRemoteObject implements hellointf {

	public helloimpl() throws RemoteException {};

	public String hellomsg(String s1){
			System.out.println("REMOTE SERVICE : Remote Client Request Message is "
+ s1);

			StringBuilder sb = new StringBuilder(s1);
			String response = sb.reverse().toString();
			return response;
	}

}
```

iii. helloserver

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.Naming;
import java.net.MalformedURLException;
import java.rmi.registry.LocateRegistry;

class helloserver extends helloimpl {

	helloserver() throws RemoteException {};

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

```
		helloimpl hi = new helloimpl();
		int port = Integer.parseInt(args[1]);

		try {
			LocateRegistry.createRegistry(port);
			System.out.println("\n RMI registry created\n");
			String host = args[0];

			String bindLocation = "//" + host + ":" + port + "/" + args[2];
			Naming.bind(bindLocation, hi);
			System.out.println("\n RMI Server ready at " + bindLocation);
		}catch(Exception e){
			e.printStackTrace();
		}
	}
}
```

iv. helloclient

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

public class helloclient{

	public static void main(String[] args) {

		String connectLocation = "//" + args[0] + ":" + Integer.parseInt(args[1]) + "/"
+ args[2];
		hellointf hintf = null;

		try {
			System.out.println("\n Connecting to Client at : " +
connectLocation);
			hintf = (hellointf) Naming.lookup(connectLocation);

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

			System.out.println("\n CLI : Enter the request message to send to
Remote Service : ");
			String s = br.readLine();

			String response = hintf.hellomsg(s);
			System.out.println("\n CLI : Response from Remote Method is : " +
response);

		} catch(Exception e){
			e.printStackTrace();
		}
	}
```

}

OUTPUT:

```
Command Prompt - java  ReverseServer 10.2.4.17 5085 reverse
C:\Users\Student\Desktop>cd DSLab

C:\Users\Student\Desktop\DSLab>cd src

C:\Users\Student\Desktop\DSLab\src>cd "RMI - reverse"

C:\Users\Student\Desktop\DSLab\src\RMI - reverse>javac ReverseServer.java

C:\Users\Student\Desktop\DSLab\src\RMI - reverse>java ReverseServer 10.2.4.17 5085 reverse

 RMI registry created


RMI server ready at //10.2.4.17:5085/reverse
REMOTE SERVICE: Remote Client REquest Message is : reverse
```

```
C:\Users\Student\Desktop\DSLab\src>cd "RMI - reverse"

C:\Users\Student\Desktop\DSLab\src\RMI - reverse>javac ReverseClient.java

C:\Users\Student\Desktop\DSLab\src\RMI - reverse>java ReverseClient 10.2.4.17 5085 reverse

 Connecting the client at: //10.2.4.17:5085/reverse

CLI: Enter the request message to send to remote service:
reverse

 CLI: response from remote method is:esrever

C:\Users\Student\Desktop\DSLab\src\RMI - reverse>
```

AIM: Program to concatenate 2 strings using RMI

PROGRAM:

i. ConcatInterface

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ConcatInterface extends Remote{
	String reverseCall(String s1,String s2) throws RemoteException;
}
```

ii. ConcatImpl

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class ConcatImpl extends UnicastRemoteObject implements ConcatInterface{
	public ConcatImpl() throws RemoteException{
	}
	public String reverseCall(String s1,String s2)
	{
			System.out.println("REMOTE SERVICE: Remote Client REquest Message is :
"+s1);
			StringBuilder sb=new StringBuilder(s1);
//			String response=sb.reverse().toString();
			return s1+s2;
	}
}
```

iii. ConcatServer

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.Naming;
import java.net.MalformedURLException;
import java.rmi.registry.LocateRegistry;

public class ConcatServer{
	public ConcatServer() throws RemoteException{
	}
	public static void main(String args[]) throws RemoteException
	{
			ConcatImpl hiObj=new ConcatImpl();
			int port=Integer.parseInt(args[1]);
			try{
				LocateRegistry.createRegistry(port);
				System.out.println("\n RMI registry created \n");
				String host=args[0];
```

```
                        String bindLocation="//"+host+":"+port+"/"+args[2];
                        Naming.bind(bindLocation,hiObj);
                        System.out.println("\nRMI server ready at "+bindLocation);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

iv. **ConcatClient**

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

public class ConcatClient {
        public static void main(String args[])
        {
                String
connectLocation="//"+args[0]+":"+Integer.parseInt(args[1])+"/"+args[2];
                ConcatInterface hintf=null;
                try{
                        System.out.println("\n Connecting the client at:
"+connectLocation);
                        hintf=(ConcatInterface)Naming.lookup(connectLocation);
                        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
                        System.out.println("\nCLI: Enter the request message to send to
remote service:");
                        String s1=br.readLine();
                String s2=br.readLine();
                        String response=hintf.reverseCall(s1,s2);
                        System.out.println("\n CLI: response from remote method
is:"+response);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

OUTPUT:

Select Command Prompt - java ConcatServer 10.2.4.17 5085 concat

```
C:\Users\Student\Desktop\DSLab\src\RMI - reverse>javac ConcatServer.java

C:\Users\Student\Desktop\DSLab\src\RMI - reverse>java ConcatServer 10.2.4.17 5085 concat

 RMI registry created


RMI server ready at //10.2.4.17:5085/concat
REMOTE SERVICE: Remote Client REquest Message is : hello server
```

Command Prompt

```
C:\Users\Student\Desktop\DSLab\src\RMI - reverse>javac ConcatClient.java

C:\Users\Student\Desktop\DSLab\src\RMI - reverse>java ConcatClient 10.2.4.17 5085 concat

 Connecting the client at: //10.2.4.17:5085/concat

CLI: Enter the request message to send to remote service:
hello server
new

 CLI: response from remote method is:hello servernew

C:\Users\Student\Desktop\DSLab\src\RMI - reverse>
```

AIM: program to find GCD of 2 numbers using RMI

PROGRAM:

i. GCDInterface

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface GCDInterface extends Remote{
	int gcdCall(int a, int b) throws RemoteException;
}
```

v. GCDImpl

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class GCDImpl extends UnicastRemoteObject implements GCDInterface{
	public GCDImpl() throws RemoteException{
	}
	public int gcdCall(int a, int b)
	{
			System.out.println("REMOTE SERVICE: Remote Client REquest Message is :
");

		int result = Math.min(a, b); // Find Minimum of a nd b
		while (result > 0) {
		   if (a % result == 0 && b % result == 0) {
		      break;
		   }
		   result--;
		}
		return result;

	}
}
```

vi. GCDServer

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.Naming;
import java.net.MalformedURLException;
import java.rmi.registry.LocateRegistry;

public class GCDServer{
	public GCDServer() throws RemoteException{
	}
	public static void main(String args[]) throws RemoteException
```

```
    {
            GCDImpl hiObj=new GCDImpl();
            int port=Integer.parseInt(args[1]);
            try{
                    LocateRegistry.createRegistry(port);
                    System.out.println("\n RMI registry created \n");
                    String host=args[0];
                    String bindLocation="//"+host+":"+port+"/"+args[2];
                    Naming.bind(bindLocation,hiObj);
                    System.out.println("\nRMI server ready at "+bindLocation);
            }
            catch(Exception e)
            {
                    e.printStackTrace();
            }
    }
}
```

vii. **GCDClient**

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

public class GCDClient {
    public static void main(String args[])
    {
            String
connectLocation="//"+args[0]+":"+Integer.parseInt(args[1])+"/"+args[2];
            GCDInterface hintf=null;
            try{
                    System.out.println("\n Connecting the client at:
"+connectLocation);
                    hintf=(GCDInterface)Naming.lookup(connectLocation);
                    BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
                    System.out.println("\nCLI: Enter the request message to send to
remote service:");
                    int a=Integer.parseInt(br.readLine());
          int b=Integer.parseInt(br.readLine());
                    int response=hintf.gcdCall(a,b);
                    System.out.println("\n CLI: response from remote method
is:"+response);
            }
            catch(Exception e)
            {
                    e.printStackTrace();
            }
    }
}
```

OUTPUT:

Command Prompt - java GCDServer 10.2.4.7 5080 gcd

```
D:\JavaApplication32\src>javac GCDServer.java

D:\JavaApplication32\src>java GCDServer 10.2.4.7 5080 gcd

 RMI registry created


RMI server ready at //10.2.4.7:5080/gcd
REMOTE SERVICE: Remote Client REquest Message is :
```

Command Prompt

```
D:\JavaApplication32\src>javac GCDClient.java

D:\JavaApplication32\src>java GCDClient 10.2.4.7 5080 gcd

 Connecting the client at: //10.2.4.7:5080/gcd

CLI: Enter the request message to send to remote service:
24
18

 CLI: response from remote method is:6

D:\JavaApplication32\src>
```

AIM: program to find addition of 2 numbers using RMI

PROGRAM:

ii. ADDInterface

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ADDInterface extends Remote{
	int addCall(int a, int b) throws RemoteException;
}
```

viii. ADDImpl

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class ADDImpl extends UnicastRemoteObject implements ADDInterface{
	public ADDImpl() throws RemoteException{
	}
	public int addCall(int a, int b)
	{
			System.out.println("REMOTE SERVICE: Remote Client REquest Message is :
");

		return a+b;

	}
}
```

ix. ADDServer

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.Naming;
import java.net.MalformedURLException;
import java.rmi.registry.LocateRegistry;

public class ADDServer{
	public ADDServer() throws RemoteException{
	}
	public static void main(String args[]) throws RemoteException
	{
			ADDImpl hiObj=new ADDImpl();
			int port=Integer.parseInt(args[1]);
			try{
				LocateRegistry.createRegistry(port);
				System.out.println("\n RMI registry created \n");
				String host=args[0];
```

```
                        String bindLocation="//"+host+":"+port+"/"+args[2];
                        Naming.bind(bindLocation,hiObj);
                        System.out.println("\nRMI server ready at "+bindLocation);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

x. **ADDClient**

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

public class ADDClient {
        public static void main(String args[])
        {
                String
connectLocation="//"+args[0]+":"+Integer.parseInt(args[1])+"/"+args[2];
                ADDInterface hintf=null;
                try{
                        System.out.println("\n Connecting the client at:
"+connectLocation);
                        hintf=(ADDInterface)Naming.lookup(connectLocation);
                        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
                        System.out.println("\nCLI: Enter the request message to send to
remote service:");
                        int a=Integer.parseInt(br.readLine());
                int b=Integer.parseInt(br.readLine());
                        int response=hintf.addCall(a,b);
                        System.out.println("\n CLI: response from remote method
is:"+response);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

OUTPUT:

Command Prompt - java ADDServer 10.2.4.7 5068 add

```
C:\Users\Student>d:

D:\>cd JavaApplication32

D:\JavaApplication32>cd src

D:\JavaApplication32\src>javac ADDServer.java

D:\JavaApplication32\src>java ADDServer 10.2.4.7 5068 add

 RMI registry created


RMI server ready at //10.2.4.7:5068/add
REMOTE SERVICE: Remote Client REquest Message is :
```

Command Prompt

```
use --help for a list of possible options

D:\JavaApplication32\src>javac ADDClient.java

D:\JavaApplication32\src>java ADDClient 10.2.4.7 5068 add

 Connecting the client at: //10.2.4.7:5068/add

CLI: Enter the request message to send to remote service:
100
15

 CLI: response from remote method is:115

D:\JavaApplication32\src>
```

AIM: program to find length of the string using RMI

PROGRAM:

i. LENInterface

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface LENInterface extends Remote{
	int lenCall(String s) throws RemoteException;
}
```

ii. LENImpl

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class LENImpl extends UnicastRemoteObject implements LENInterface{
	public LENImpl() throws RemoteException{
	}
	public int lenCall(String s)
	{
			System.out.println("REMOTE SERVICE: Remote Client REquest Message is :
");

		return s.length();

	}
}
```

iii. LENServer

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.Naming;
import java.net.MalformedURLException;
import java.rmi.registry.LocateRegistry;

public class LENServer{
	public LENServer() throws RemoteException{
	}
	public static void main(String args[]) throws RemoteException
	{
			LENImpl hiObj=new LENImpl();
			int port=Integer.parseInt(args[1]);
			try{
				LocateRegistry.createRegistry(port);
				System.out.println("\n RMI registry created \n");
				String host=args[0];
```

```
                        String bindLocation="//"+host+":"+port+"/"+args[2];
                        Naming.bind(bindLocation,hiObj);
                        System.out.println("\nRMI server ready at "+bindLocation);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

iv. LENClient

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

public class LENClient {
        public static void main(String args[])
        {
                String
connectLocation="//"+args[0]+":"+Integer.parseInt(args[1])+"/"+args[2];
                LENInterface hintf=null;
                try{
                        System.out.println("\n Connecting the client at:
"+connectLocation);
                        hintf=(LENInterface)Naming.lookup(connectLocation);
                        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
                        System.out.println("\nCLI: Enter the request message to send to
remote service:");
                        String s=br.readLine();
                        int response=hintf.lenCall(s);
                        System.out.println("\n CLI: response from remote method
is:"+response);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

OUTPUT:

Command Prompt - java LENServer 10.2.4.7 5085 len

```
RMI server ready at //10.2.4.7:5068/len
REMOTE SERVICE: Remote Client REquest Message is :
REMOTE SERVICE: Remote Client REquest Message is :

D:\JavaApplication32\src>java LENServer 10.2.4.7 5085 len

 RMI registry created


RMI server ready at //10.2.4.7:5085/len
REMOTE SERVICE: Remote Client REquest Message is :
```

Command Prompt

```
CLI: Enter the request message to send to remote service:
welcome

 CLI: response from remote method is:7

D:\JavaApplication32\src>java LENClient 10.2.4.7 5085 len

 Connecting the client at: //10.2.4.7:5085/len

CLI: Enter the request message to send to remote service:
welcome

 CLI: response from remote method is:7

D:\JavaApplication32\src>
```

AIM: Program to perform FTP Upload.

PROGRAM:

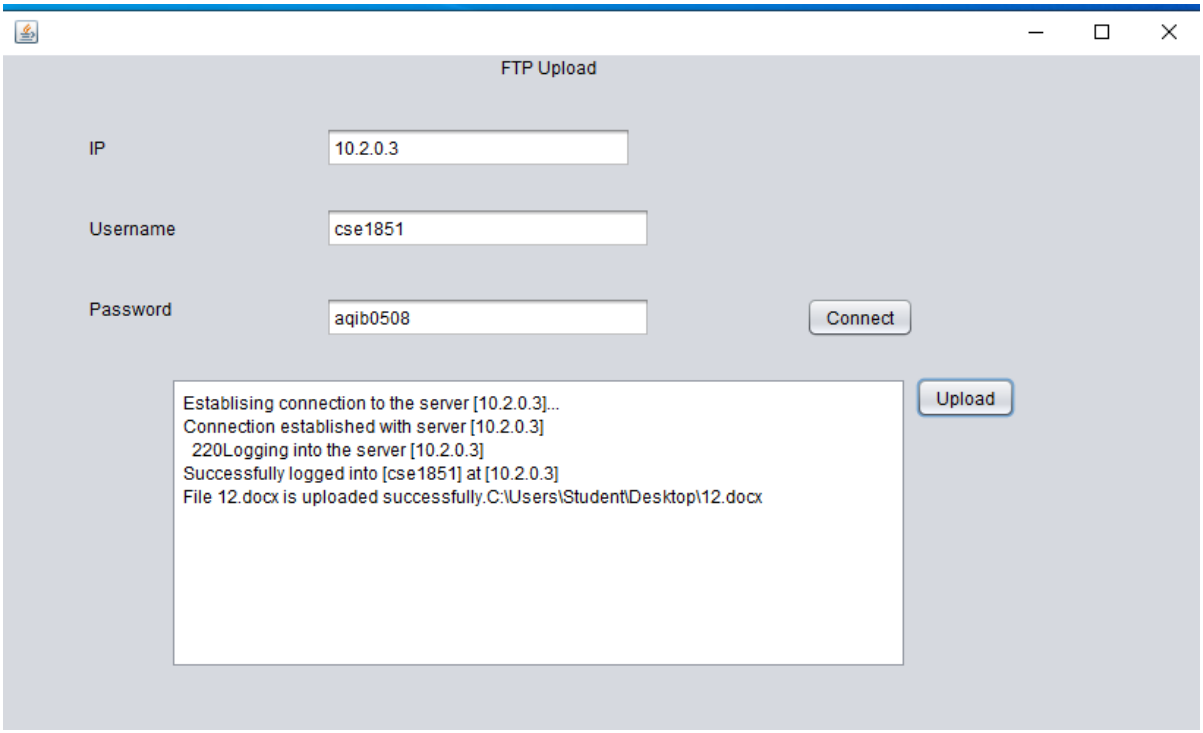
FtpUpload:

```
import java.io.*; import java.util.*;

import org.apache.commons.net.ftp.*;

import org.apache.commons.net.ftp.FTPClient;import org.apache.commons.net.ftp.FTPReply;

import javax.swing.*; //will have to search all classes of swing package. takes timeimport

javax.swing.SwingUtilities; //saves search. immediate reference

import javax.swing.filechooser.*; //for file dialog for upload/download

public class ftpUpload extends javax.swing.JFrame {

        private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {try{

            String IP=txtIP.getText();  //GET IP

            FTPClient ftpc=new FTPClient(); //CREATE CLIENT txtAC.append("Establising connection

            to the server ["+IP+"]...\n");ftpc.connect(IP);  //CONNECT WITH SERVER

            int reply=ftpc.getReplyCode();

    //to check the status of the connection

            //    Positive Completion reply: the requested action has been successfully
                  completed.
A new request may be initiated.

            if(FTPReply.isPositiveCompletion(reply))

              txtAC.append("Connection established with server ["+IP+"]\n"+"  "+reply);//220
Service ready for new user.

            else

              txtAC.append("Connection failed with the server ["+IP+"]\n");String

            uname=txtUN.getText();  //GET USERNAME

            String pwd=txtPW.getText();     //GET PASSWORD txtAC.append("Logging into the server

            ["+IP+"]\n");if(ftpc.login(uname,pwd))

              txtAC.append("Successfully logged into ["+uname+"] at ["+IP+"]\n");else
```

```
                    txtAC.append("Unable to login to ["+uname+"] at ["+IP+"]\n");

                //txtAC.append("Disconnecting from the server...\n");ftpc.disconnect();

//if client is idle for long then server disconnects and other operations fail. That is why
disconnectedhere.

            }
        catch(Exception e)

        {
           e.printStackTrace();
        }

    // TODO add your handling code here:

    }

private void btnuploadActionPerformed(java.awt.event.ActionEvent evt) {String fileName = "";

        String fileAbsName = "";

        JFileChooser fc=new JFileChooser();//JFileChooser is a easy and an effective way to prompt the
user to choose a file or a directory

        int returnVal = fc.showOpenDialog(ftpUpload.this);

        if (returnVal == JFileChooser.APPROVE_OPTION) //Approve option: returns yes or ok

        {

            File file = fc.getSelectedFile(); fileAbsName = file.getAbsolutePath();fileName =

            file.getName();

        }

            try

        {

           String ServerIP = txtIP.getText();FTPClient f = new FTPClient(); f.connect(ServerIP);

           String user = txtUN.getText(); String passwd = txtPW.getText();f.login(user, passwd);

           File firstLocalFile = new File(fileAbsName);

           String firstRemoteFile = fileName;

           InputStream inputStream = new FileInputStream(firstLocalFile);boolean done =

           f.storeFile(firstRemoteFile, inputStream); inputStream.close();

           if (done)
```

```
            txtAC.append("File "+ fileName +" is uploaded successfully."+fileAbsName);else
            txtAC.append("File "+ fileName +" cannot be uploaded.");f.disconnect();
        }
        catch (Exception ex)
        {
          ex.printStackTrace();
        }
        // TODO add your handling code here:
    }
public static void main(String args[]) { java.awt.EventQueue.invokeLater(new Runnable() {
          public void run() {
            new ftpc1().setVisible(true);
                                                                                                                    }
                                                                                                                  });
    }
}
```

OUTPUT:



AIM: Program to perform FTP Download

PROGRAM:

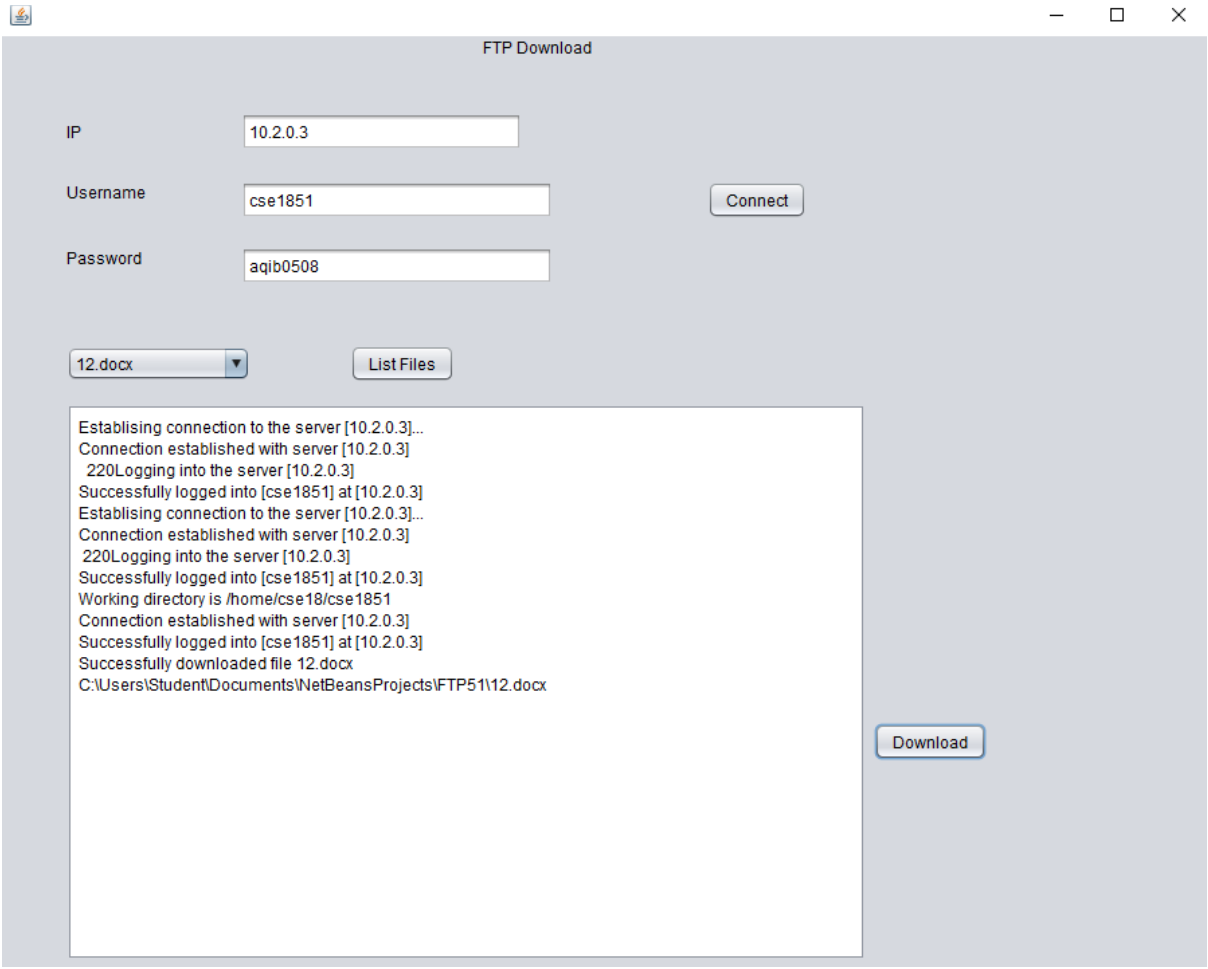
```
import java.io.*; import java.util.*;

import org.apache.commons.net.ftp.*;

import org.apache.commons.net.ftp.FTPClient;import org.apache.commons.net.ftp.FTPReply;

import javax.swing.*; //will have to search all classes of swing package. takes timeimport

javax.swing.SwingUtilities; //saves search. immediate reference

import javax.swing.filechooser.*; //for file dialog for upload/downloadpublic class ftpDownload

extends javax.swing.JFrame {

private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {try{

        String IP=txtIP.getText();  //GET IP

        FTPClient ftpc=new FTPClient(); //CREATE CLIENT txtAC.append("Establising connection to

        the server ["+IP+"]...\n");ftpc.connect(IP);  //CONNECT WITH SERVER

        int reply=ftpc.getReplyCode();

//to check the status of the connection

        //          Positive Completion reply: the requested action has been successfully
                    completed.
A new request may be initiated. if(FTPReply.isPositiveCompletion(reply))

           txtAC.append("Connection established with server ["+IP+"]\n"+"  "+reply);//220Service
ready for new user.

        else

           txtAC.append("Connection failed with the server ["+IP+"]\n");String

        uname=txtUN.getText();  //GET USERNAME

        String pwd=txtPW.getText();     //GET PASSWORD txtAC.append("Logging into the server

        ["+IP+"]\n");if(ftpc.login(uname,pwd))

           txtAC.append("Successfully logged into ["+uname+"] at ["+IP+"]\n");else

           txtAC.append("Unable to login to ["+uname+"] at ["+IP+"]\n");

        txtAC.append("Disconnecting from the server...\n"); ftpc.disconnect();

//if client is idle for long then server disconnects and other operations fail. That is why
```

```
disconnectedhere.
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        // TODO add your handling code here:
    }
private void btndownloadActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
         try{
            String IP=txtIP.getText();
            String uname=txtUN.getText(); String pwd=txtPW.getText(); FTPClient ftpc=new FTPClient();
            ftpc.connect(IP);
            int reply=ftpc.getReplyCode(); if(FTPReply.isPositiveCompletion(reply))


                txtAC.append("Connection established with server ["+IP+"]\n");else
               txtAC.append("Connection failed with server ["+IP+"]\n");if(ftpc.login(uname, pwd))
            {
                txtAC.append("Successfully logged into ["+uname+"] at ["+IP+"]\n");
                txtIP.enableInputMethods(false);
/*Since we have successfully logged in, disable input on these fields*/
                txtUN.enableInputMethods(false); txtPW.enableInputMethods(false);
                String selectedFile=cb1.getSelectedItem().toString();
//get name of the file chosen for download
                File dFileName=new File(selectedFile);
//to get the file object for reading and writing
                OutputStream os=new BufferedOutputStream(new FileOutputStream(dFileName));
```

```
//sincereading is done in parts :.BufferedOutputStream
            boolean success=ftpc.retrieveFile(selectedFile,os);
//storeFile() for upload
            os.close();if(success)
txtAC.append("Successfully downloaded file "+selectedFile+"\n"+dFileName.getAbsolutePath());
            else
                txtAC.append("Could not download file "+selectedFile+"\n");
        }
        else
            txtAC.append("Unable to log into ["+uname+"] at ["+IP+"]\n");ftpc.disconnect();
              }
      catch(Exception e)
      {
        e.printStackTrace();
      }
   }
//listing
 private void btnlistfilesActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
       try{
        String IP=txtIP.getText(); FTPClient ftpc=new FTPClient();
        txtAC.append("Establising connection to the server ["+IP+"]...\n");ftpc.connect(IP);
        int reply=ftpc.getReplyCode();//to check the status of the connection
        if(FTPReply.isPositiveCompletion(reply))
          txtAC.append("Connection established with server ["+IP+"]\n"+" "+reply);else
           txtAC.append("Connection failed with the server ["+IP+"]\n");String
        uname=txtUN.getText();
        String pwd=txtPW.getText(); txtAC.append("Logging into the server ["+IP+"]\n");
```

```
        if(ftpc.login(uname,pwd))
        {
          txtAC.append("Successfully logged into ["+uname+"] at ["+IP+"]\n");String
          pdir=ftpc.printWorkingDirectory();
//changeWorkingDirectory() to change current directory txtAC.append("Working directory is
          "+pdir+"\n");
          FTPFile ftpf[]=ftpc.listFiles();  //takes names of all files in current directory pdir
          //ComboBox.removeAllItems(); //to remove the default 5 itemsfor(int
          i=0;i<ftpf.length;i++)
          {
            cb1.addItem(ftpf[i].getName());
          }
        }
        else
          txtAC.append("Unable to login to ["+uname+"] at ["+IP+"]\n");
        txtAC.append("Disconnecting from the server...\n"); ftpc.disconnect();
      }
      catch(Exception e)
      {
        e.printStackTrace();
      }
      }
public static void main(String args[]) { java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
          new ftpc1().setVisible(true);
        }
      });
    }
}
```

OUTPUT:


FTP Download
IP
10.2.0.3
Username
cse1851
Connect
Password
aqib0508
12.docx
List Files
Establising connection to the server [10.2.0.3]...
Connection established with server [10.2.0.3]
220Logging into the server [10.2.0.3]
Successfully logged into [cse1851] at [10.2.0.3]
Establising connection to the server [10.2.0.3]...
Connection established with server [10.2.0.3]
220Logging into the server [10.2.0.3]
Successfully logged into [cse1851] at [10.2.0.3]
Working directory is /home/cse18/cse1851
Connection established with server [10.2.0.3]
Successfully logged into [cse1851] at [10.2.0.3]
Successfully downloaded file 12.docx
C:\Users\Student\Documents\NetBeansProjects\FTP51\12.docx
Download