**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();

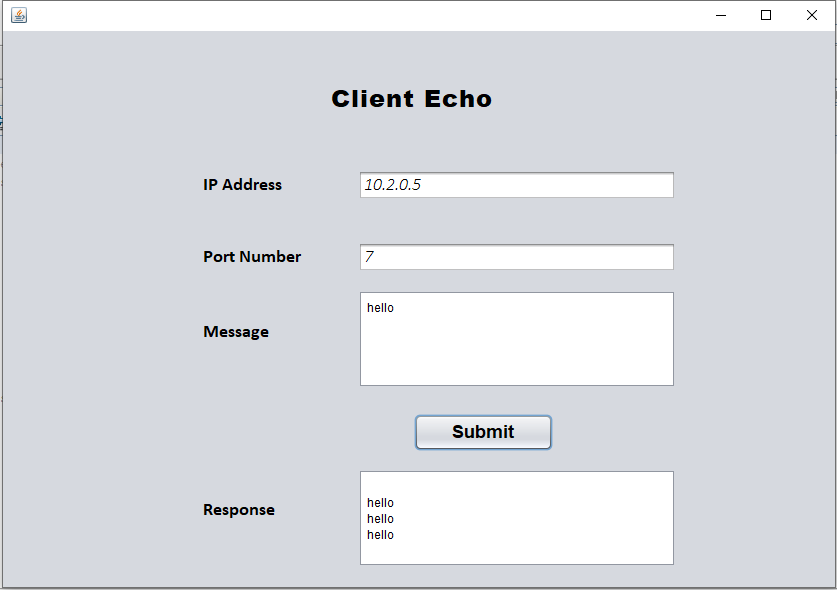
CRESPONSE.append("/n" + newStr);

} 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);

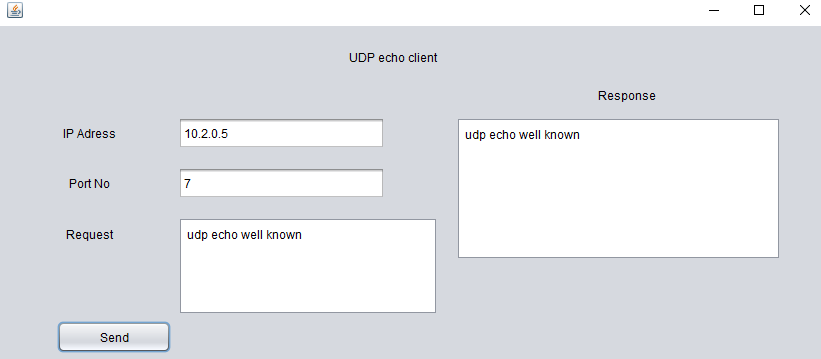
s.close();

}catch(Exception e){

e.printStackTrace();

}

}

**OUTPUT:**

**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");

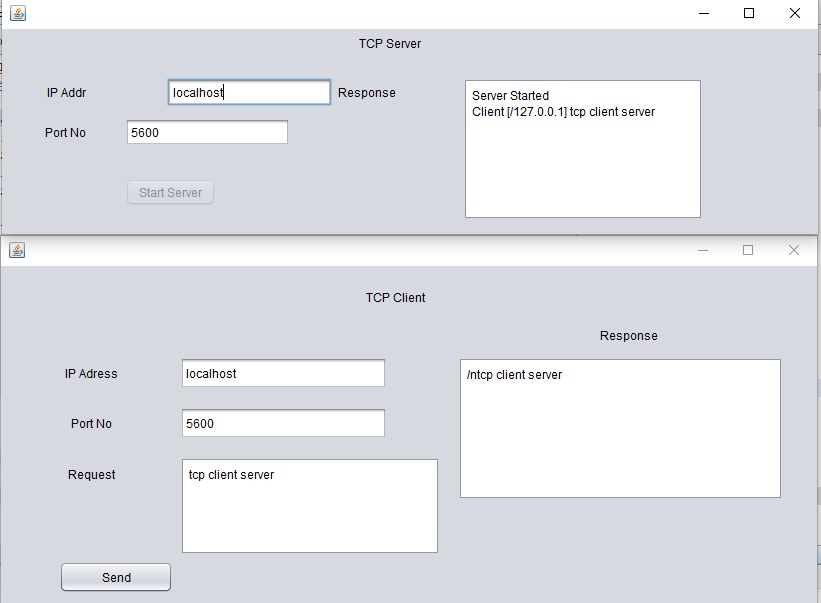
t.start();

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");

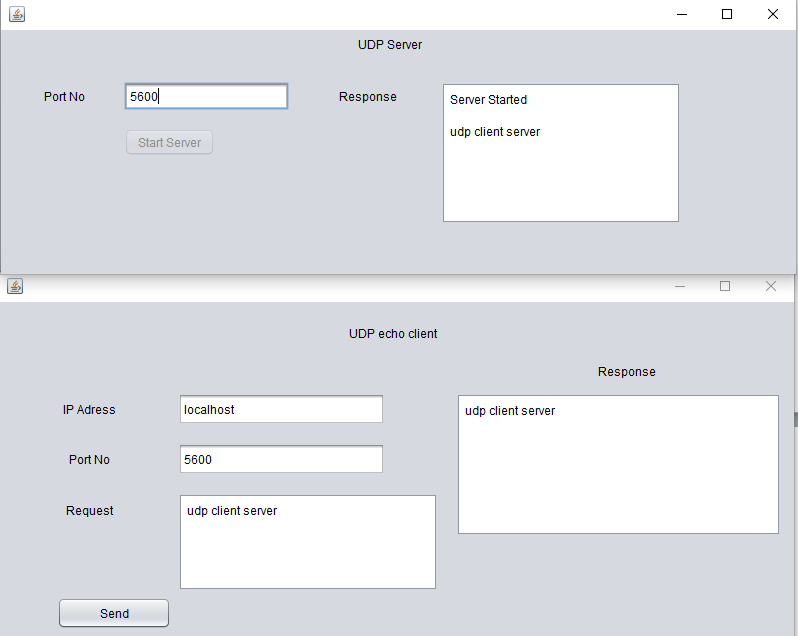
t.start();

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);

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

String st = Username.getText();

dos.writeUTF(st);

CMessage.append("\n"+newStr);

s.close();

}

catch(Exception e){

e.printStackTrace();

}

}

# 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 str = dis.readUTF();

SMessage.append("\"\n"+s.getInetAddress()+" : "+str);

dos.writeUTF("\nServer Response: "+str);

s.close();

}

}

catch(Exception e){

e.printStackTrace();

}

}

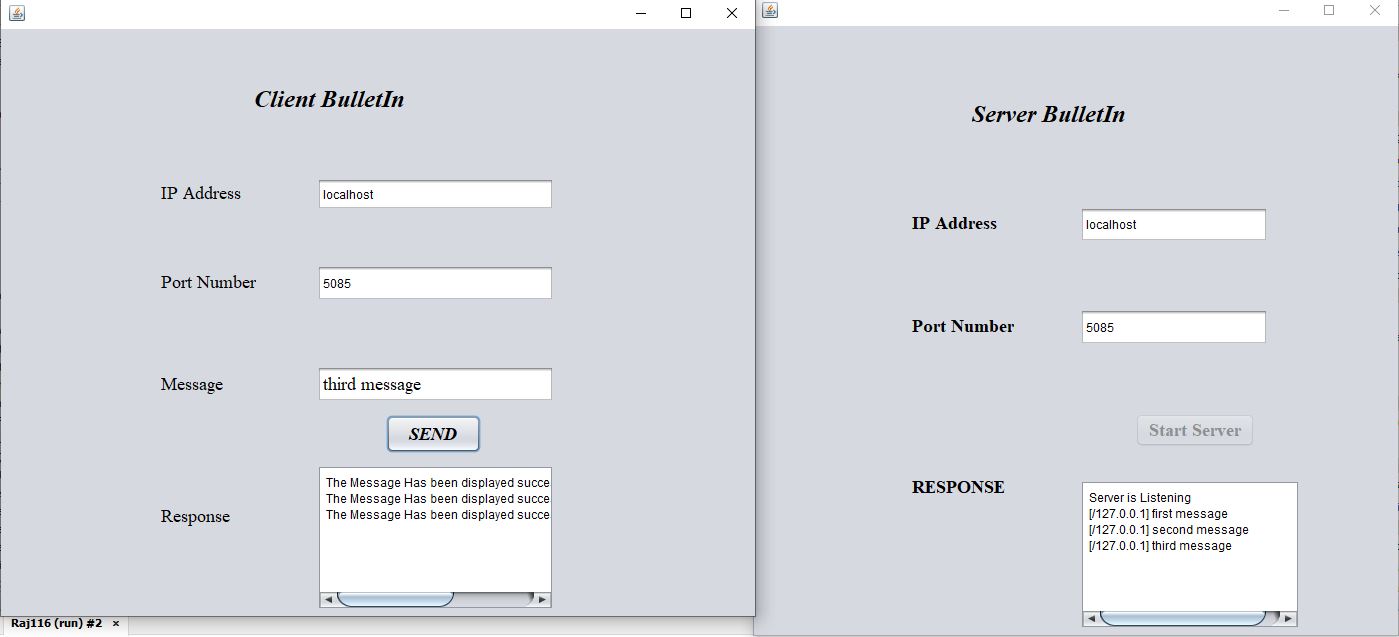
private void StartActionPerformed(java.awt.event.ActionEvent evt) { Thread t = new Thread(this,"t1");

t.start();

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

}

# OUTPUT:



**AIM: Program to demonstrate Domain Name Server.**

**PROGRAM:**

# 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());

String domain=DName.getText();

Socket s=new Socket(ip,port);

DataInputStream dis=new DataInputStream(s.getInputStream()); DataOutputStream dos=new DataOutputStream(s.getOutputStream());

dos.writeUTF(domain);

String response=dis.readUTF();

Message.append(" "+response+"\n");

s.close();

}

catch(Exception e){

e.printStackTrace();

}

# 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 connections

while(true){

Socket s=ss.accept(); //client's request has come; connection is established

DataInputStream dis=new DataInputStream(s.getInputStream()); DataOutputStream dos=new DataOutputStream(s.getOutputStream());

String req=dis.readUTF();

Smessage.append(" " +s.getInetAddress().toString()+"/");

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); //tokens String 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) {

Thread t=new Thread(this,"ns");

t.start();

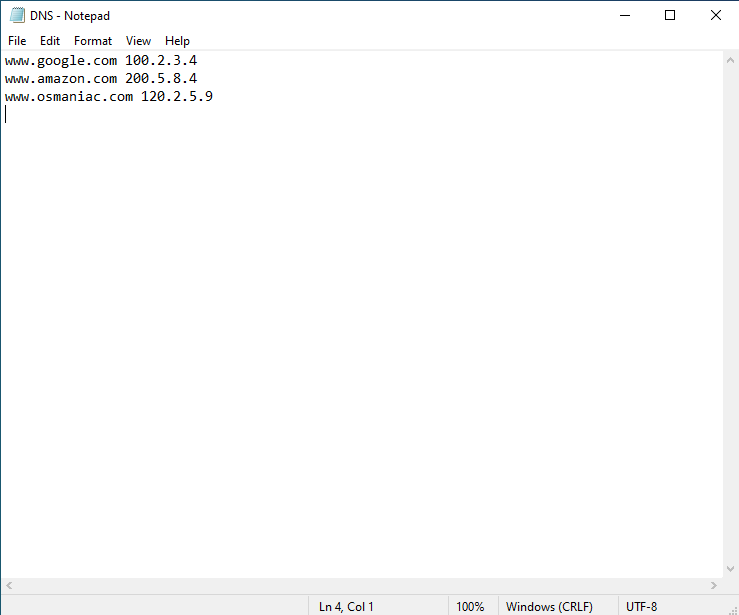
Start.setEnabled(false);

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

}

**OUTPUT:**

**Input File:**



# AIM: Program to design Chat Application using Client and Server

# PROGRAM:

public void run()

{

try{

Integer port=Integer.parseInt(SPORT.getText());

ServerSocket ss=new ServerSocket(port);

while(true)

{

Socket s=ss.accept(); //s is a socket which has got connection i.e., full association tuple

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

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

String str=dis.readUTF();

SRESP.append("\n"+s.getInetAddress()+" : "+str);

dos.writeUTF(str);

}

}

catch(Exception e)

{

e.printStackTrace();

}

private void SENDBtnActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

try{

String ip=CIP.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 str=CMSG.getText();

dos.writeUTF(str);

String newStr=dis.readUTF();

CMSG.append("\n"+newStr);

s.close();

}

catch(Exception e)

{

e.printStackTrace();

}

}

private void STARTSERVERBtnActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

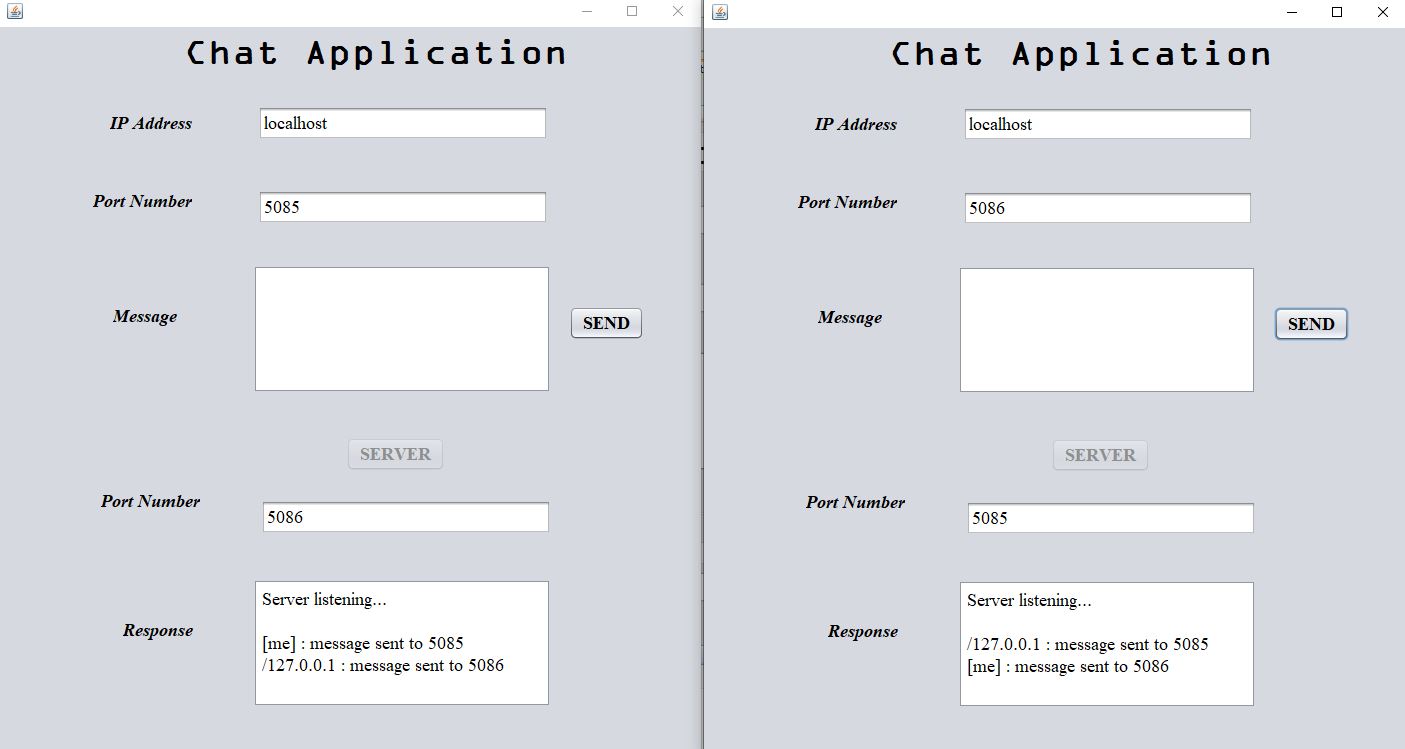
Thread t=new Thread(this,"t1");

t.start();

SRESP.append("Server listening...\n");

STARTSERVERBtn.setEnabled(false);

}

**OUTPUT:**

**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



1. **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;

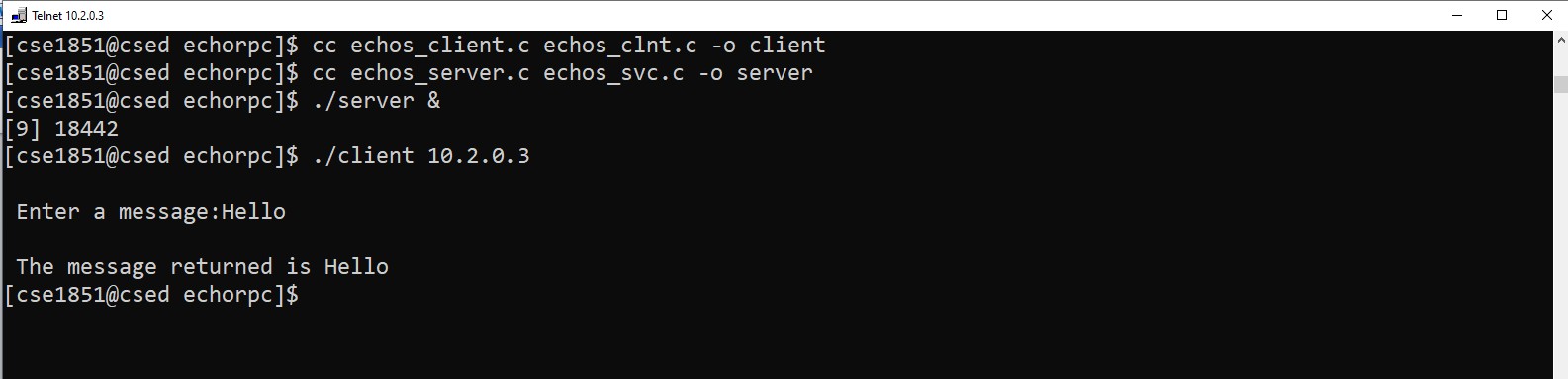
}

# 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

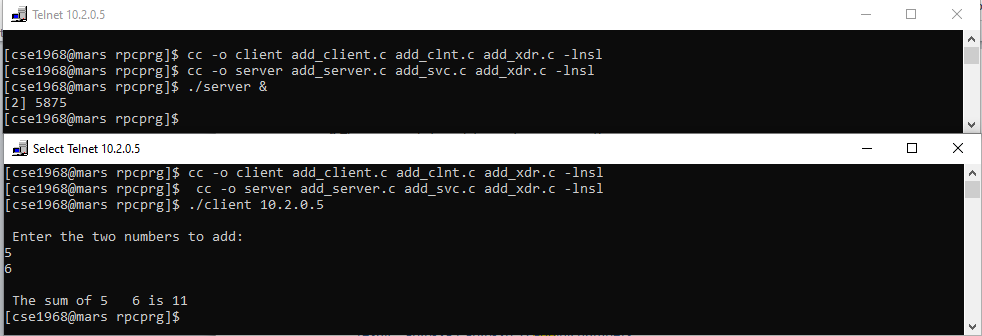
$cc –o add\_server.c add\_svc.c –lnsl addserver

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)

{

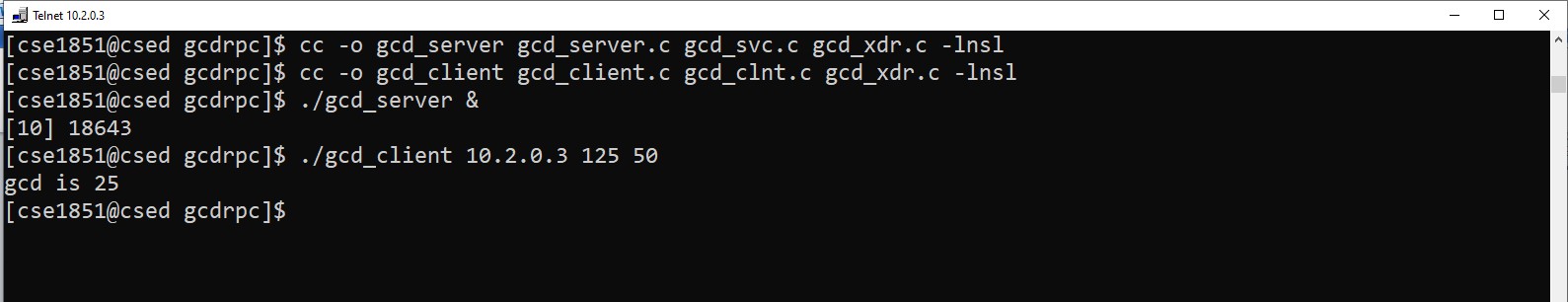
static long result;

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

return &result;

}

**OUTPUT:**



**AIM: Program to demonstrate echo message using RMI**

**PROGRAM:**

1. **HelloInterface**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface HelloInterface extends Remote{

String helloMsg(String s) throws RemoteException;

}

1. **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;

}

}

1. **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();

}

}

}

1. **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**

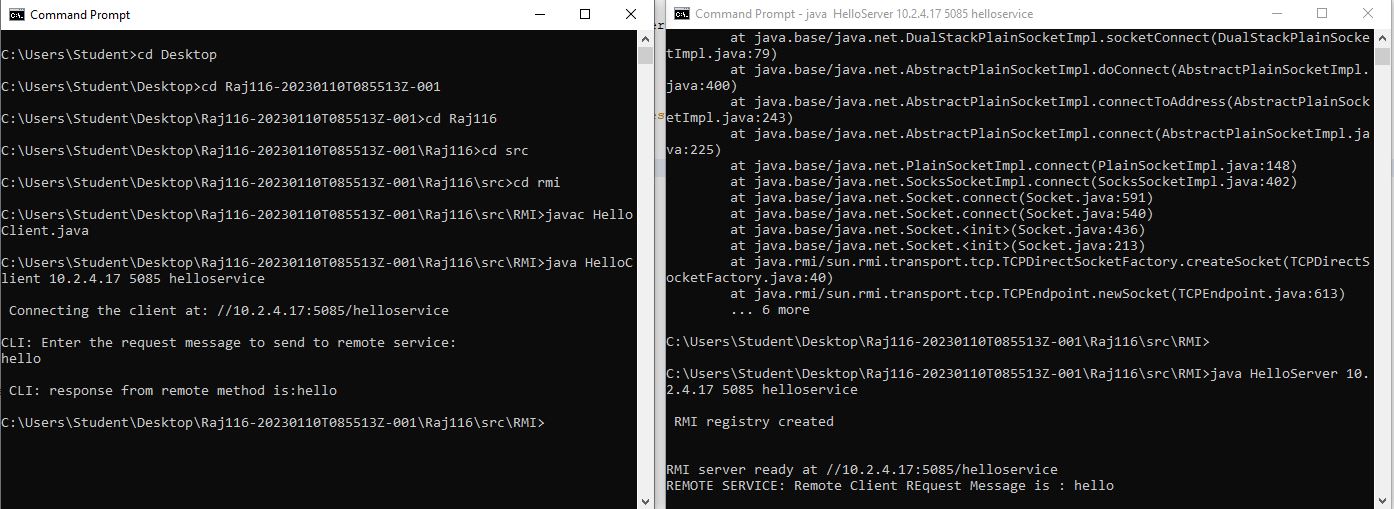
**C: javac \*.java**

**C: rmic AddServerImpl**

**C: start rmiregistry**

**C: java AddServer**

**C: java AddClient localhost 30 23**

**OUTPUT:**

**AIM: Program to reverse a string using RMI**

**PROGRAM:**

1. **hellointf**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface hellointf extends Remote {

String hellomsg(String s) throws RemoteException;

}

1. **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;

}

}

1. **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();

}

}

}

1. **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){

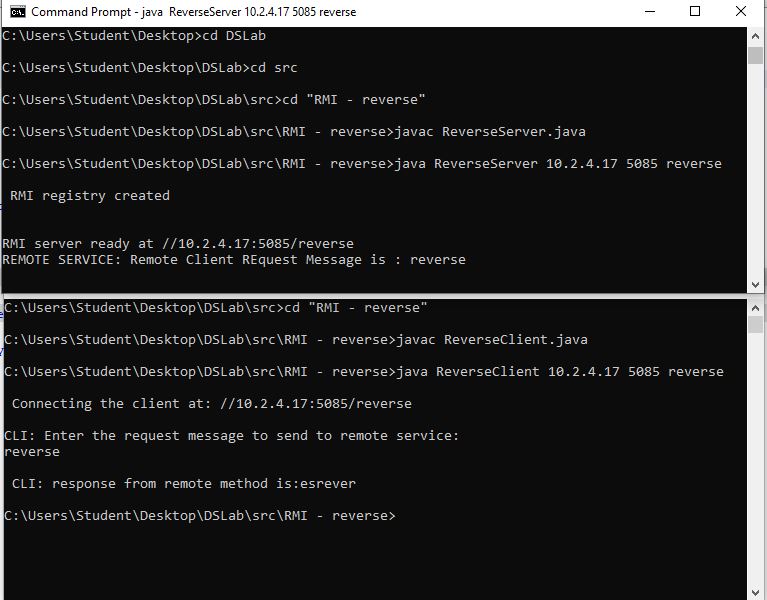
e.printStackTrace();

}

}

}

**OUTPUT:**

****

**AIM: Program to concatenate 2 strings using RMI**

**PROGRAM:**

1. **ConcatInterface**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface ConcatInterface extends Remote{

String reverseCall(String s1,String s2) throws RemoteException;

}

1. **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;

}

}

1. **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();

}

}

}

1. **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)

{

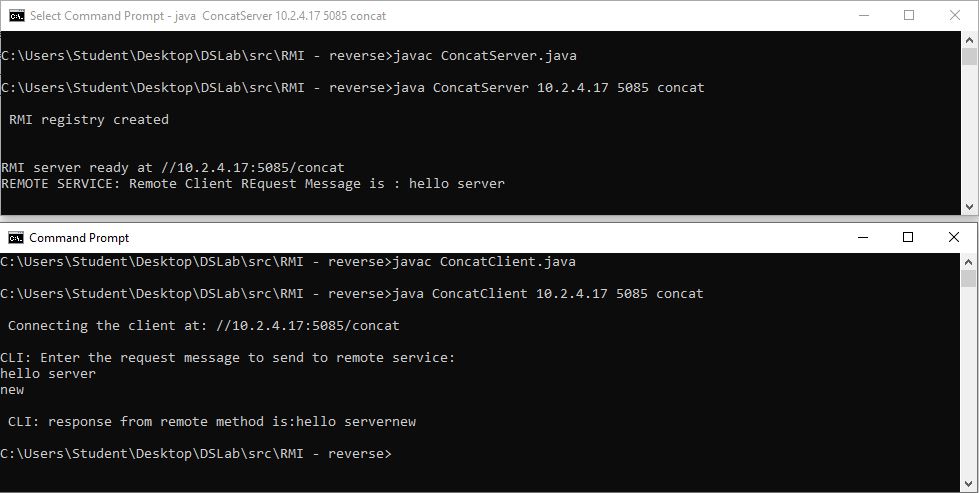
e.printStackTrace();

}

}

}

**OUTPUT:**

****

**AIM: program to find GCD of 2 numbers using RMI**

**PROGRAM:**

1. **GCDInterface**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface GCDInterface extends Remote{

int gcdCall(int a, int b) throws RemoteException;

}

1. **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;

}

}

1. **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();

}

}

}

1. **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)

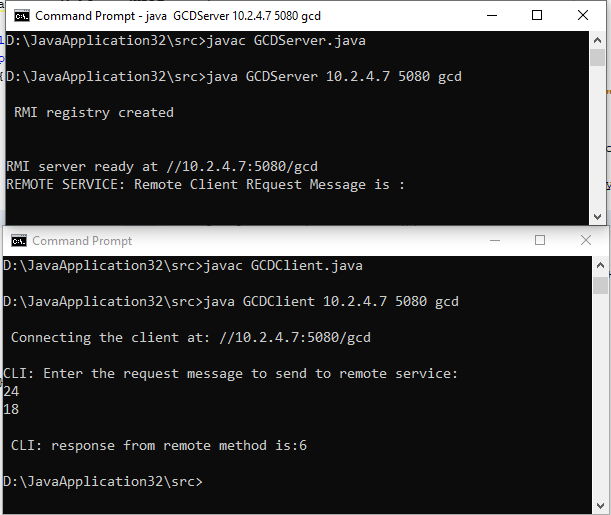
{

e.printStackTrace();

}

}

}

**OUTPUT:**

**AIM: program to find addition of 2 numbers using RMI**

**PROGRAM:**

1. **ADDInterface**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface ADDInterface extends Remote{

int addCall(int a, int b) throws RemoteException;

}

1. **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;

}

}

1. **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();

}

}

}

1. **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)

{

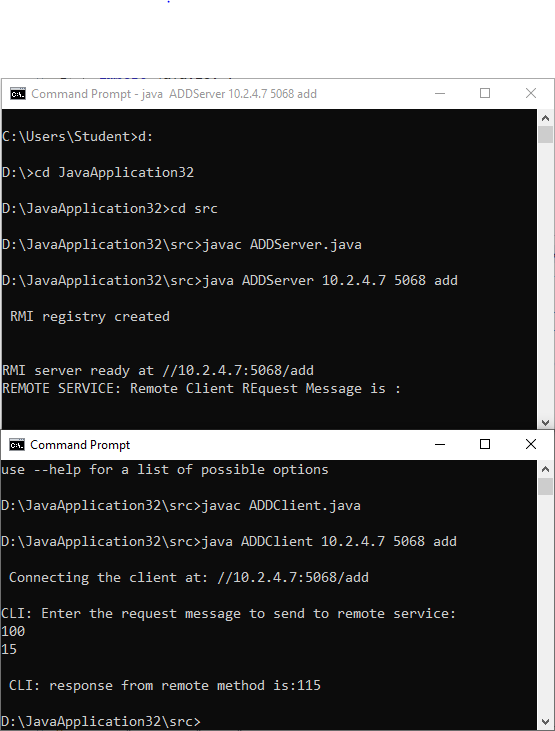
e.printStackTrace();

}

}

}

**OUTPUT:**

****

**AIM: program to find length of the string using RMI**

**PROGRAM:**

1. **LENInterface**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface LENInterface extends Remote{

int lenCall(String s) throws RemoteException;

}

1. **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();

}

}

1. **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();

}

}

}

1. **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)

{

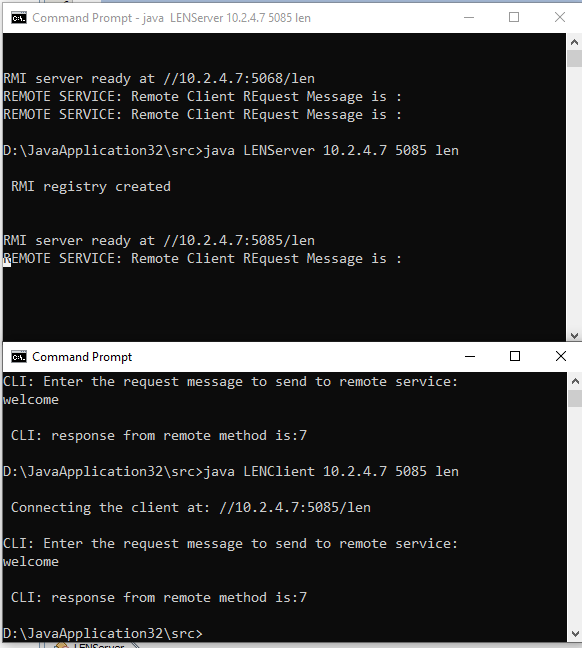
e.printStackTrace();

}

}

}

**OUTPUT:**

****

# AIM: Program to perform FTP Upload, Download and Display

# PROGRAM:

import java.awt.FileDialog;

import java.io.BufferedOutputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.io.OutputStream;

import javax.swing.JFileChooser;

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

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

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

private void connectBtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_connectBtnActionPerformed

// TODO add your handling code here:

String ipAddr=ipAddress.getText();

try{

ftp=new FTPClient();

ftp.connect(ipAddr);

Integer replyCode=ftp.getReplyCode();

if(FTPReply.isPositiveCompletion(replyCode))

{

statusWindow.append("\nconnected to :"+ipAddr+"\n");

}

else

{

statusWindow.append("\nconnection failed for :"+ipAddr+"\n");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}//GEN-LAST:event\_connectBtnActionPerformed

private void disconnectBtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_disconnectBtnActionPerformed

// TODO add your handling code here:

try{

ftp.disconnect();

statusWindow.append("\ndisconnected\n");

}

catch(Exception e)

{

e.printStackTrace();

}

}//GEN-LAST:event\_disconnectBtnActionPerformed

private void loginBtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_loginBtnActionPerformed

// TODO add your handling code here:

String user=username.getText();

String pass=new String(password.getPassword());

try{

if(ftp.login(user, pass))

{

statusWindow.append("\nLogin successful to "+username+"\n");

statusWindow.append("\ncurrent working directory:"+ftp.printWorkingDirectory()+"\n");

}

else

{

statusWindow.append("\nLogin unsuccessful to "+username+"\n");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}//GEN-LAST:event\_loginBtnActionPerformed

private void listFilesBtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_listFilesBtnActionPerformed

// TODO add your handling code here:

comboBox.removeAllItems();

try{

for(FTPFile f:ftp.listFiles())

{

comboBox.addItem(f.getName());

}

}

catch(Exception e)

{

e.printStackTrace();

}

}//GEN-LAST:event\_listFilesBtnActionPerformed

private void changeDirectoryBtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_changeDirectoryBtnActionPerformed

// TODO add your handling code here:

try{

ftp.changeWorkingDirectory((String)comboBox.getSelectedItem());

statusWindow.append("\nCurrent dir:"+ftp.printWorkingDirectory()+"\n");

comboBox.removeAllItems();

}

catch(Exception e)

{

e.printStackTrace();

}

}//GEN-LAST:event\_changeDirectoryBtnActionPerformed

private void downloadBtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_downloadBtnActionPerformed

// TODO add your handling code here:

try{

String fileName=(String)comboBox.getSelectedItem();

File f=new File(fileName);

OutputStream os=new BufferedOutputStream(new FileOutputStream(f));

if(ftp.retrieveFile(fileName, os))

{

statusWindow.append("\nFile downloaded successfully\n");

}

else

{

statusWindow.append("\nFile download unsuccessful\n");

}

os.close();

}

catch(Exception e)

{

e.printStackTrace();

}

}//GEN-LAST:event\_downloadBtnActionPerformed

private void GoToParentDirectoryBtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_GoToParentDirectoryBtnActionPerformed

// TODO add your handling code here:

try

{

if(ftp.changeToParentDirectory())

{

statusWindow.append("\n"+ftp.printWorkingDirectory()+"\n");

}

else

{

statusWindow.append("\n"+"change directory unsuccessful"+"\n");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}//GEN-LAST:event\_GoToParentDirectoryBtnActionPerformed

private void uploadBtnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_uploadBtnActionPerformed

// TODO add your handling code here:

try{

JFileChooser fileDialog=new JFileChooser();

int status=fileDialog.showOpenDialog(this);

if(status==JFileChooser.APPROVE\_OPTION)

{

File f=fileDialog.getSelectedFile();

InputStream is=new FileInputStream(f);

String fileName=f.getName();

if(ftp.storeFile(fileName, is))

{

statusWindow.append("\nFile successfully uploaded\n");

}

else

{

statusWindow.append("\nFile upload unsuccessful\n");

}

}

else

{

statusWindow.append("\nFile not chosen\n");

}

}

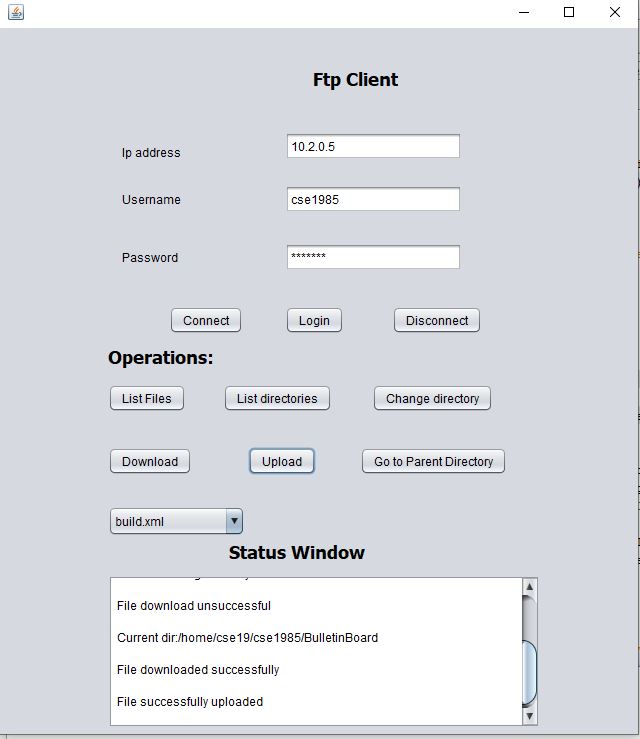
catch(Exception e)

{

e.printStackTrace();

}

**OUTPUT:**

****