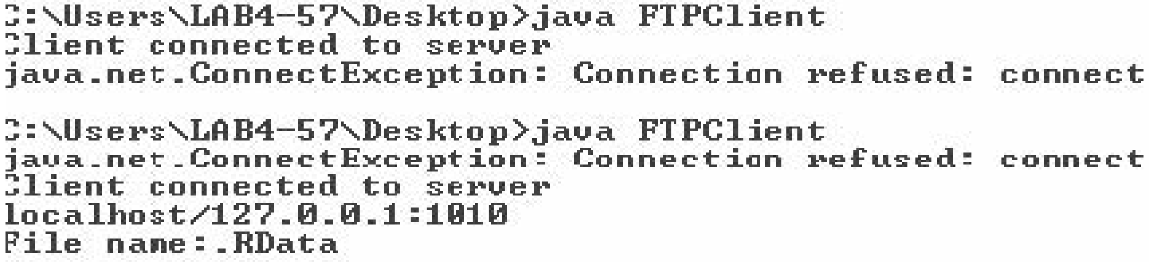
Implementation FTP Client

FTP Client:

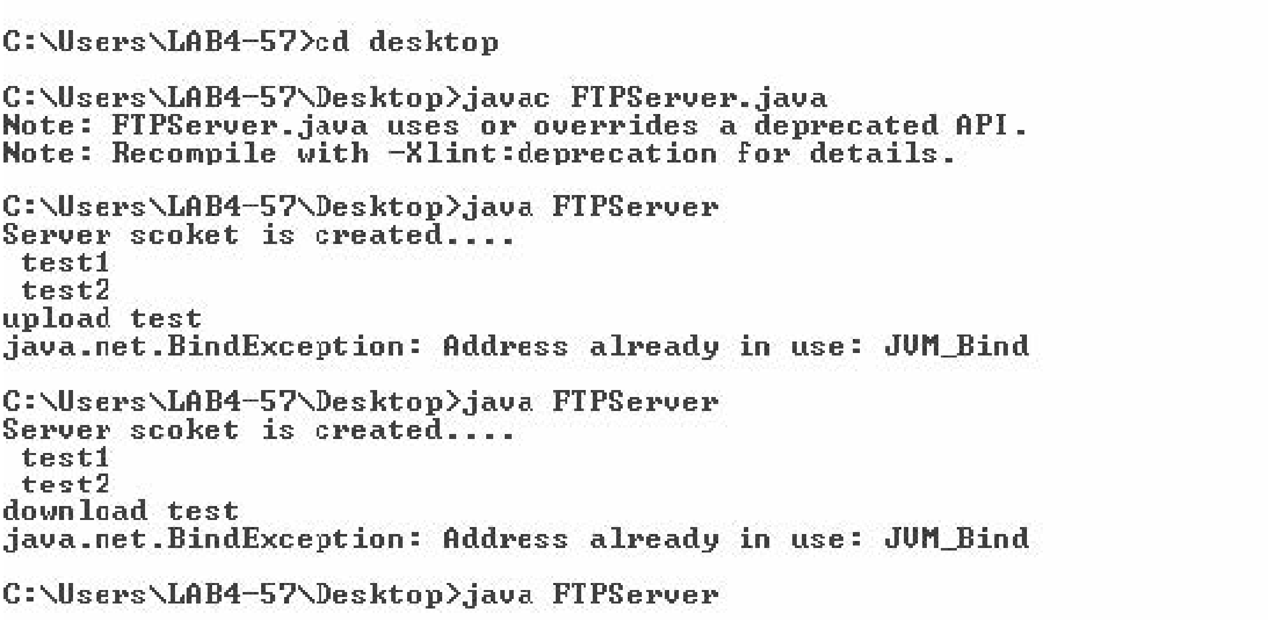
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
class One extends JFrame implements ActionListener
{
/* ctrl space */
public JButton b,b1;
public JLabel l;
public JLabel l1,lmsg1,lmsg2;
One()
{
b=new JButton("Upload");
l=new JLabel("Uplaod a file : ");
lmsg1=new JLabel("");
b1=new JButton("Download");
l1=new JLabel("Downlaod a file");
lmsg2=new JLabel("");
setLayout(new GridLayout(2,3,10,10));
add(l);add(b);add(lmsg1);add(l1);add(b1);add(lmsg2);
b.addActionListener(this);
b1.addActionListener(this);
setVisible(true);
setSize(600,500);
}
public void actionPerformed(ActionEvent e)
{
try {
if (b.getModel().isArmed())
{
Socket s=new Socket("localhost",1010);
System.out.println("Client connected to server");
JFileChooser j=new JFileChooser();
int val;
val=j.showOpenDialog(One.this);
String filename=j.getSelectedFile().getName();
String path=j.getSelectedFile().getPath();
PrintStream out=new PrintStream(s.getOutputStream());
out.println("Upload");
out.println(filename);
FileInputStream fis=new FileInputStream(path);
int n=fis.read();
while (n!=-1)
{
out.print((char)n);n=fis.read();
}
fis.close(); out.close();lmsg1.setText(filename+"is uploaded");
//s.close();
repaint();
}
if (b1.getModel().isArmed())
{
Socket s=new Socket("localhost",1010);
System.out.println("Client connected to server");
String remoteadd=s.getRemoteSocketAddress().toString();
System.out.println(remoteadd);
JFileChooser j1=new JFileChooser(remoteadd);
int val;
val=j1.showOpenDialog(One.this);
String filename=j1.getSelectedFile().getName();
String filepath=j1.getSelectedFile().getPath();
System.out.println("File name:"+filename);
PrintStream out=new PrintStream(s.getOutputStream());
out.println("Download");
out.println(filepath);
FileOutputStream fout=new FileOutputStream(filename);
DataInputStream fromserver=new
DataInputStream(s.getInputStream());
int ch;
while ((ch=fromserver.read())!=-1)
{
fout.write((char) ch);
}
fout.close();//s.close();
lmsg2.setText(filename+"is downlaoded");
repaint();
}
}
catch (Exception ee)
{
System.out.println(ee);
}
}
}
public class FTPClient
{
public static void main(String[] args)
{
new One();
}
}
```



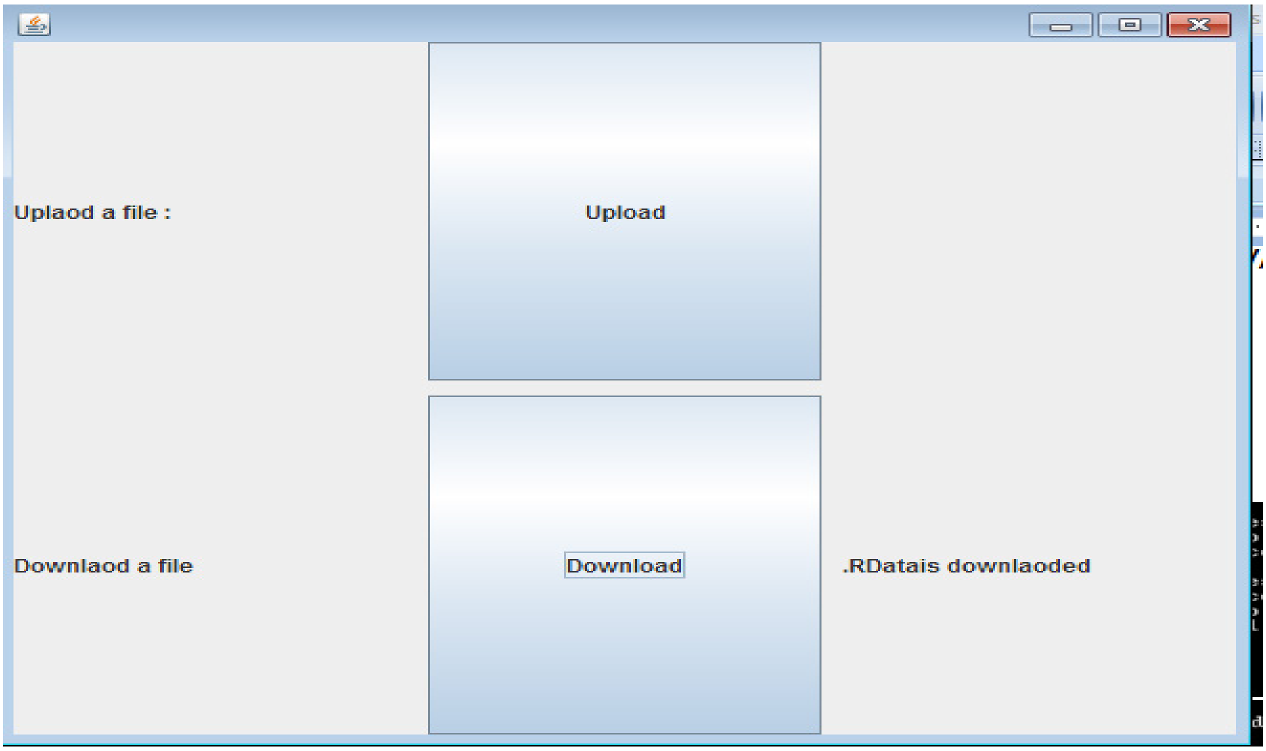
```
C:\Users\LAB4-57\Desktop>java FTPClient
Client connected to server
java.net.ConnectException: Connection refused: connect

C:\Users\LAB4-57\Desktop>java FTPClient
java.net.ConnectException: Connection refused: connect
Client connected to server
localhost/127.0.0.1:1010
File name:.RData
```

FTP Server:

```
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
public class FTPServer {
public static void main(String[] args)
{
try {
while (true)
{
ServerSocket ss=new ServerSocket(1010);
Socket sl=ss.accept();
System.out.println("Server scoket is created....");
System.out.println(" test1");
DataInputStream fromserver=new DataInputStream(sl.getInputStream());
System.out.println(" test2");
String option=fromserver.readLine();
if (option.equalsIgnoreCase("upload"))
{
System.out.println("upload test");
String filefromclient=fromserver.readLine();
File clientfile=new File(filefromclient);
FileOutputStream fout=new FileOutputStream(clientfile);
int ch;
while ((ch=fromserver.read())!=-1)
{
fout.write((char)ch);
}
fout.close();
}
if (option.equalsIgnoreCase("download"))
{
System.out.println("download test");
String filefromclient=fromserver.readLine();
File clientfile=new File(filefromclient);
FileInputStream fis=new FileInputStream(clientfile);
PrintStream out=new PrintStream(sl.getOutputStream());
int n=fis.read();
while (n!=-1)
{
out.print((char)n);
n=fis.read();
}
fis.close();
out.close();
} //while
}
}
catch (Exception e)
{
System.out.println(e);
// TODO: handle exception
}
}
}
```



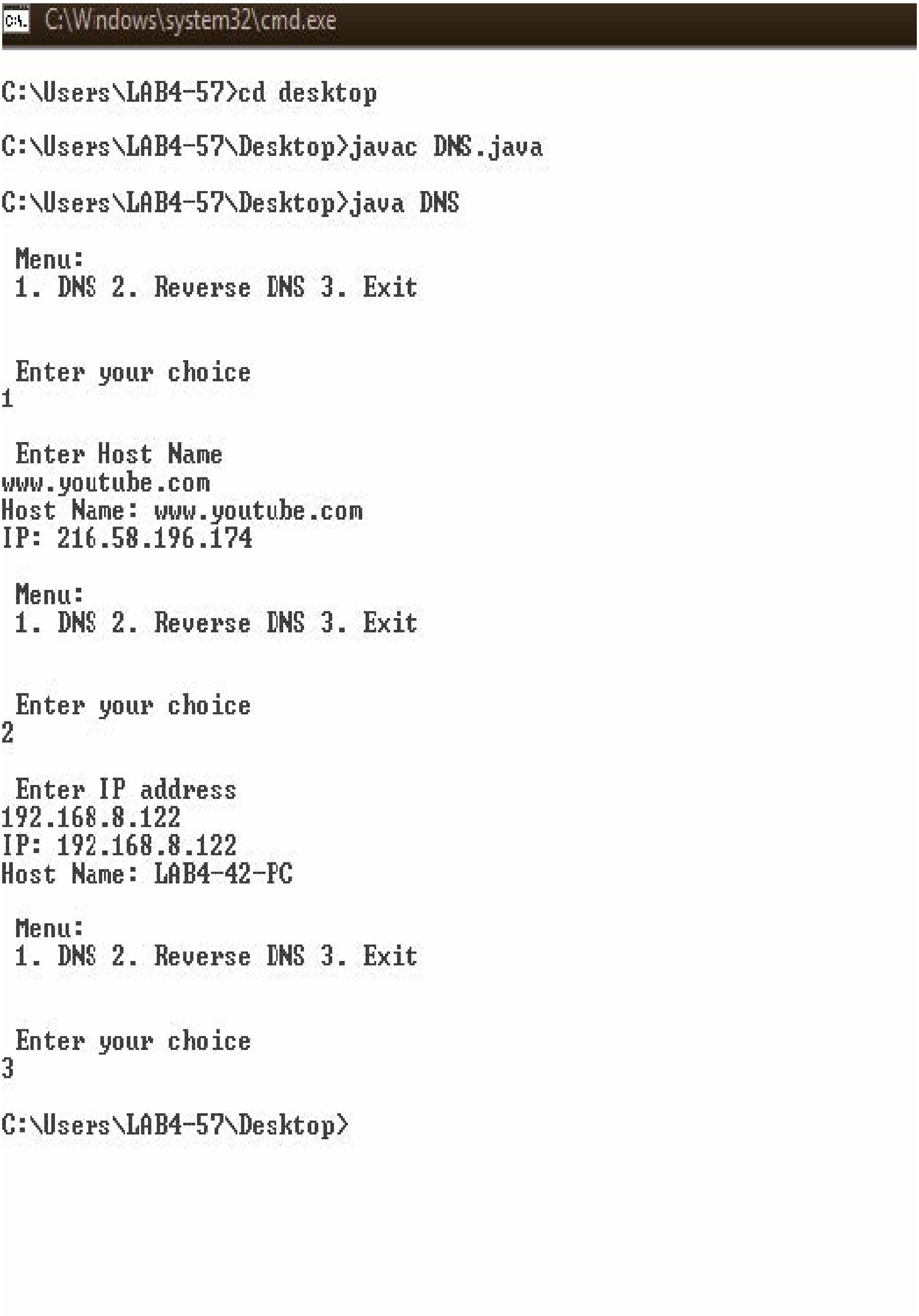
```
C:\Users\LAB4-57>cd desktop

C:\Users\LAB4-57\Desktop>javac FTPServer.java
Note: FTPServer.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Users\LAB4-57\Desktop>java FTPServer
Server scoket is created....
 test1
 test2
upload test
java.net.BindException: Address already in use: JVM_Bind

C:\Users\LAB4-57\Desktop>java FTPServer
Server scoket is created....
 test1
 test2
download test
java.net.BindException: Address already in use: JVM_Bind

C:\Users\LAB4-57\Desktop>java FTPServer
```



.RDatais downlaoded

Implementation of Name Server

```
import java.net.*;
import java.io.*;
import java.util.*;
public class DNS
{
public static void main(String[] args)
{
int n;
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
do
{
System.out.println("\n Menu: \n 1. DNS 2. Reverse DNS 3. Exit \n");
System.out.println("\n Enter your choice");
n = Integer.parseInt(System.console().readLine());
if(n==1)
{
try
{
System.out.println("\n Enter Host Name ");
String hname=in.readLine();
InetAddress address;
address = InetAddress.getByName(hname);
System.out.println("Host Name: " + address.getHostName());
System.out.println("IP: " + address.getHostAddress());
}
catch(IOException ioe)
{
ioe.printStackTrace();
}
}
if(n==2)
{
try
{
System.out.println("\n Enter IP address");
String ipstr = in.readLine();
InetAddress ia = InetAddress.getByName(ipstr);
System.out.println("IP: "+ipstr);
System.out.println("Host Name: " +ia.getHostName());
}
catch(IOException ioe)
{
ioe.printStackTrace();
}
}
}while(!(n==3));
}
}
```

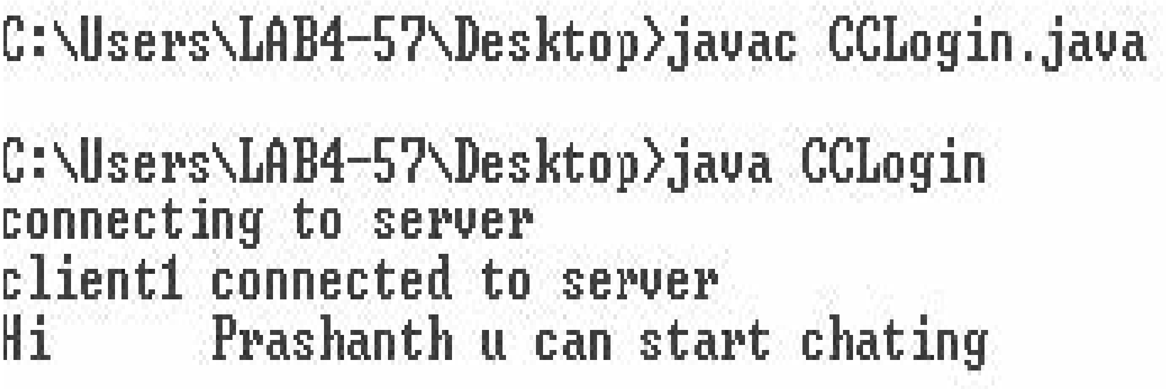


```
C:\Users\LAB4-57>cd desktop

C:\Users\LAB4-57\Desktop>javac DNS.java

C:\Users\LAB4-57\Desktop>java DNS

 Menu:
 1. DNS 2. Reverse DNS 3. Exit


 Enter your choice
1

 Enter Host Name
www.youtube.com
Host Name: www.youtube.com
IP: 216.58.196.174

 Menu:
 1. DNS 2. Reverse DNS 3. Exit


 Enter your choice
2

 Enter IP address
192.168.8.122
IP: 192.168.8.122
Host Name: LAB4-42-PC

 Menu:
 1. DNS 2. Reverse DNS 3. Exit


 Enter your choice
3

C:\Users\LAB4-57\Desktop>
```

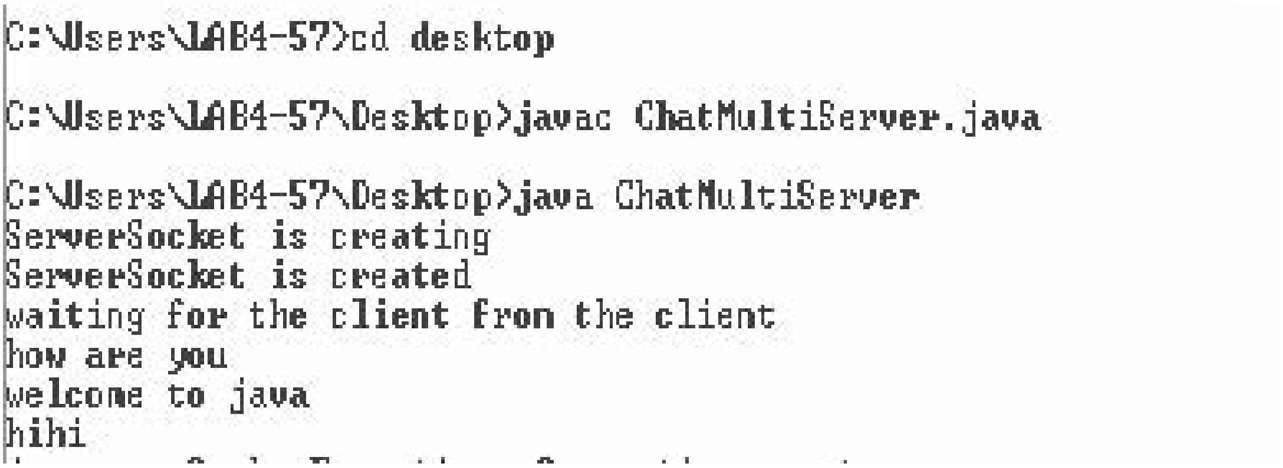
Implementation of Chat Server

CCLogin.java

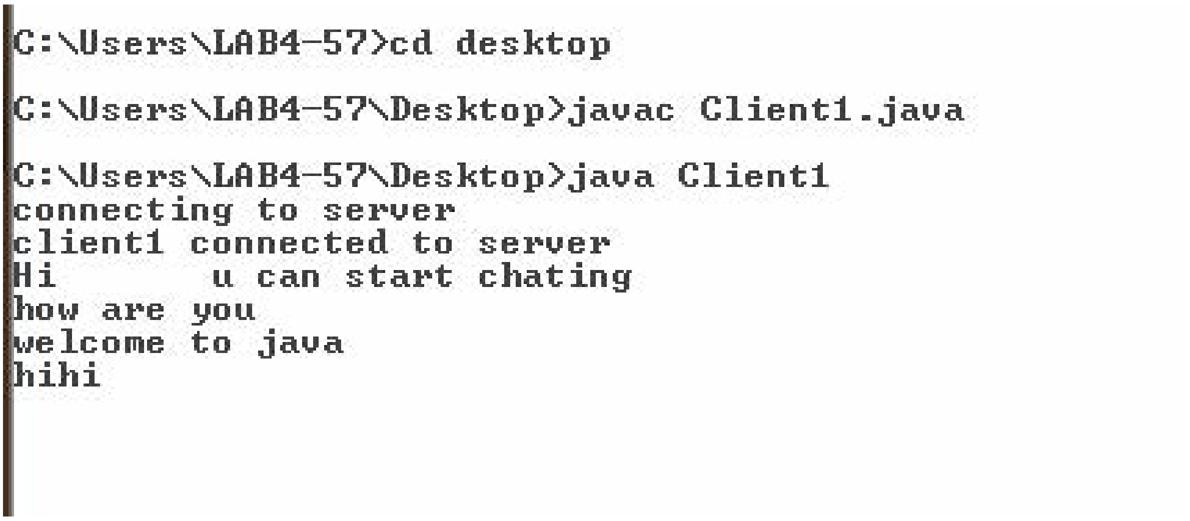
```
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import java.awt.GridLayout;
public class CCLogin implements ActionListener
{
JFrame frame1; JTextField tf,tf1; JButton button;
JLabel heading; JLabel label,label1;
public static void main(String[] paramArrayOfString)
{
new CCLogin();
}
public CCLogin()
{
this.frame1 = new JFrame("Login Page");
this.tf = new JTextField(10);
this.button = new JButton("Login");
this.heading = new JLabel("Chat Server");
this.heading.setFont(new Font("Impact", 1, 40));
this.label = new JLabel("Enter you Login Name");
this.label.setFont(new Font("Serif", 0, 24));
JPanel localJPanel = new JPanel();
this.button.addActionListener(this);
localJPanel.add(this.heading); localJPanel.add(this.label);
localJPanel.add(this.tf);
localJPanel.add(this.button);
this.heading.setBounds(30, 20, 280, 50);
this.label.setBounds(20, 100, 250, 60);
this.tf.setBounds(50, 150, 150, 30);
this.button.setBounds(70, 190, 90, 30);
this.frame1.add(localJPanel);
localJPanel.setLayout(null);
this.frame1.setSize(300,300);
this.frame1.setVisible(true);
this.frame1.setDefaultCloseOperation(3);
}
public void actionPerformed(ActionEvent paramActionEvent)
{
String str = "";
try
{
str = this.tf.getText();
this.frame1.dispose();
Client1 c1= new Client1(str);
c1.main(null);
}
catch(Exception localIOException)
{
}
}
}
```



```
C:\Users\LAB4-57\Desktop>javac CCLogin.java

C:\Users\LAB4-57\Desktop>java CCLogin
connecting to server
client1 connected to server
Hi      Prashanth u can start chating
```



```
C:\Users\LAB4-57>cd desktop

C:\Users\LAB4-57\Desktop>javac ChatMultiServer.java

C:\Users\LAB4-57\Desktop>java ChatMultiServer
ServerSocket is creating
ServerSocket is created
waiting for the client from the client
how are you
welcome to java
hihi
```



```
C:\Users\LAB4-57>cd desktop

C:\Users\LAB4-57\Desktop>javac Client1.java

C:\Users\LAB4-57\Desktop>java Client1
connecting to server
client1 connected to server
Hi       u can start chating
how are you
welcome to java
hihi
```

ChatMultiServer:

```
import java.net.*;
import java.io.*;
class A implements Runnable
{
Thread t;
Socket s;
A(Socket x)
{
s=x;
t=new Thread(this);
t.start();
}
public void run()
{
try
{
/* Reading data from client */
InputStream is=s.getInputStream();
byte data[]=new byte[50];
is.read(data);
String mfc=new String(data);
mfc=mfc.trim();
System.out.println(mfc);
/* Sending message to the server */
//System.out.println("Hi"+name+"u can start chating");
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
String n=br.readLine();
OutputStream os=s.getOutputStream();
os.write(n.getBytes());
}
catch(Exception e)
{
e.printStackTrace();
}
}
}
class ChatMultiServer
{
static int c=0;
public static void main(String args[]) throws Exception
{
System.out.println("ServerSocket is creating");
ServerSocket ss=new ServerSocket(1010);
System.out.println("ServerSocket is created");
System.out.println("waiting for the client from the client");
while(true)
{
Socket s=ss.accept();
new A(s);
}
}
}
```

Client1.java

```
import java.net.*;
import java.io.*;
class Client1
{
static String name="";
public Client1(String n)
{
name=n;
}
public static void main(String args[]) throws Exception
{
System.out.println("connecting to server");
System.out.println("client1 connected to server");
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
/* Sending message to the server */
System.out.println("Hi\t"+name+" u can start chating");
while(true)
{
Socket s=new Socket("localhost",1010);
String n=br.readLine();
OutputStream os=s.getOutputStream();
os.write(n.getBytes());
/* Reading data from client */
InputStream is=s.getInputStream();
byte data[]=new byte[50];
is.read(data);
String mfc=new String(data);
mfc=mfc.trim();
System.out.println(mfc);
}
}
}
```

Understanding of Working of NFS (includes exercises Configuration of NFS)

Study of Network File Systems

1. Create a Folder nfs/abc.txt

2. Know the ipaddress

Applications->System Settings->Network—edit (ipaddress, subnetmask)

(or) In terminal type ifconfig

3. Enable the desired services

1. System Services->Server Settings->Services

□ Network (Enable)

□ Nfs (Enable)

□ Iptables (Disable) (we do not firewalls)

2. System Settings ->Security Level (Firewall options-disable, Selinuxdisable)

Creation of Network File System Server

1. System Settings->Server Settings->NFS

+ Add (All are making security levels low)

2. Open Terminal

Type: service nfs restart

Creation of NFS Client

Open terminal

Type: df

Type: mount –t nfs 135.135.5.120:/usr/nfs /root/abc

cd abc

ls : abc.txt

Unmount: umount –t nfs 135.135.5.120:/usr/nfs

Note: service network restart (if n/w is disabled use this)

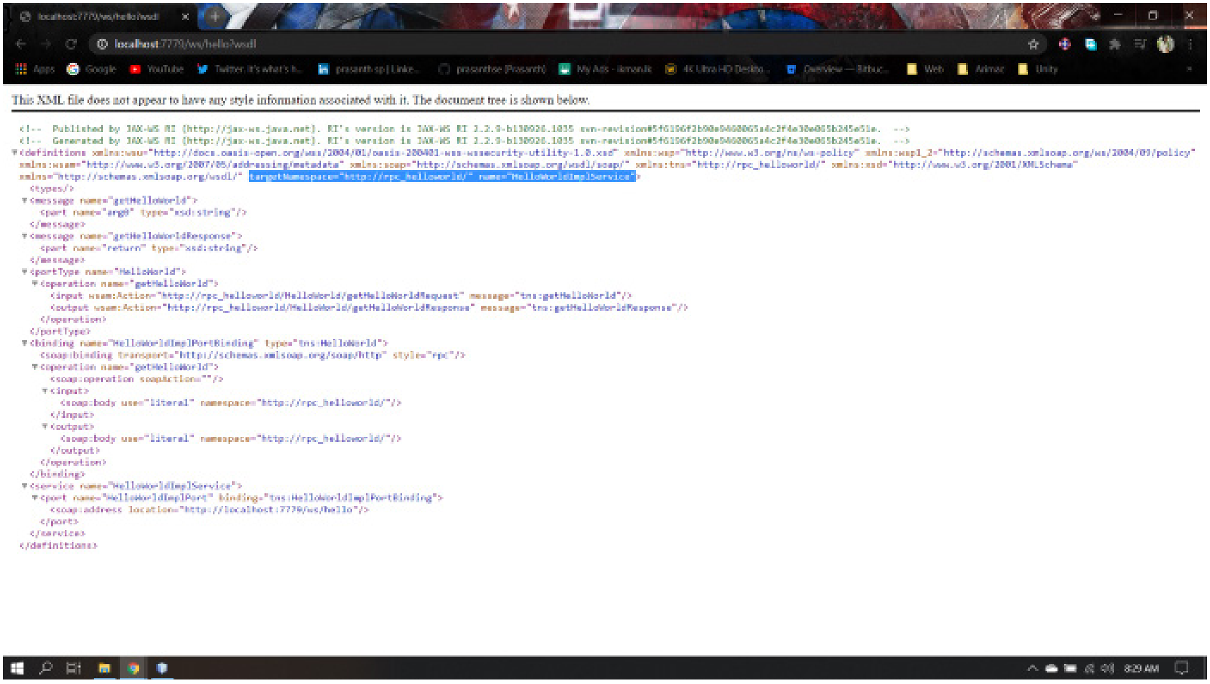
Write a program to implement Hello world service using RPC.

Publisher.java

```
package rpc_helloworld;
import javax.xml.ws.Endpoint;
public class Publisher {
    public static void main(String[] args) {
       Endpoint.publish("http://localhost:7779/ws/hello", new HelloWorldImpl());
    }
}
```

RPC_HelloWorld.java

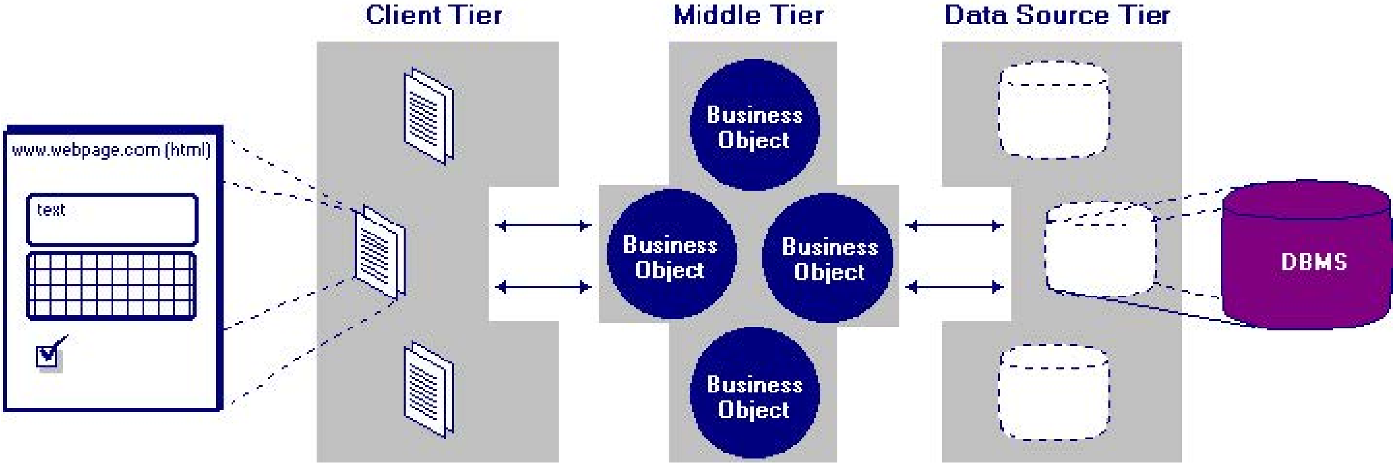
```
package rpc_helloworld;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;

public class RPC_HelloWorld {

    public static void main(String[] args) {

        try {
            URL url = new URL("http://localhost:7779/ws/hello?wsdl");

            QName qname = new QName("http://rpc_helloworld/", "HelloWorldImplService");

            Service service = Service.create(url, qname);
            HelloWorld hello = service.getPort(HelloWorld.class);

            System.out.println(hello.getHelloWorld("Hello World!"));

        } catch (MalformedURLException ex) {
            System.out.println("WSDL document url error: " + ex);
        }
    }
```

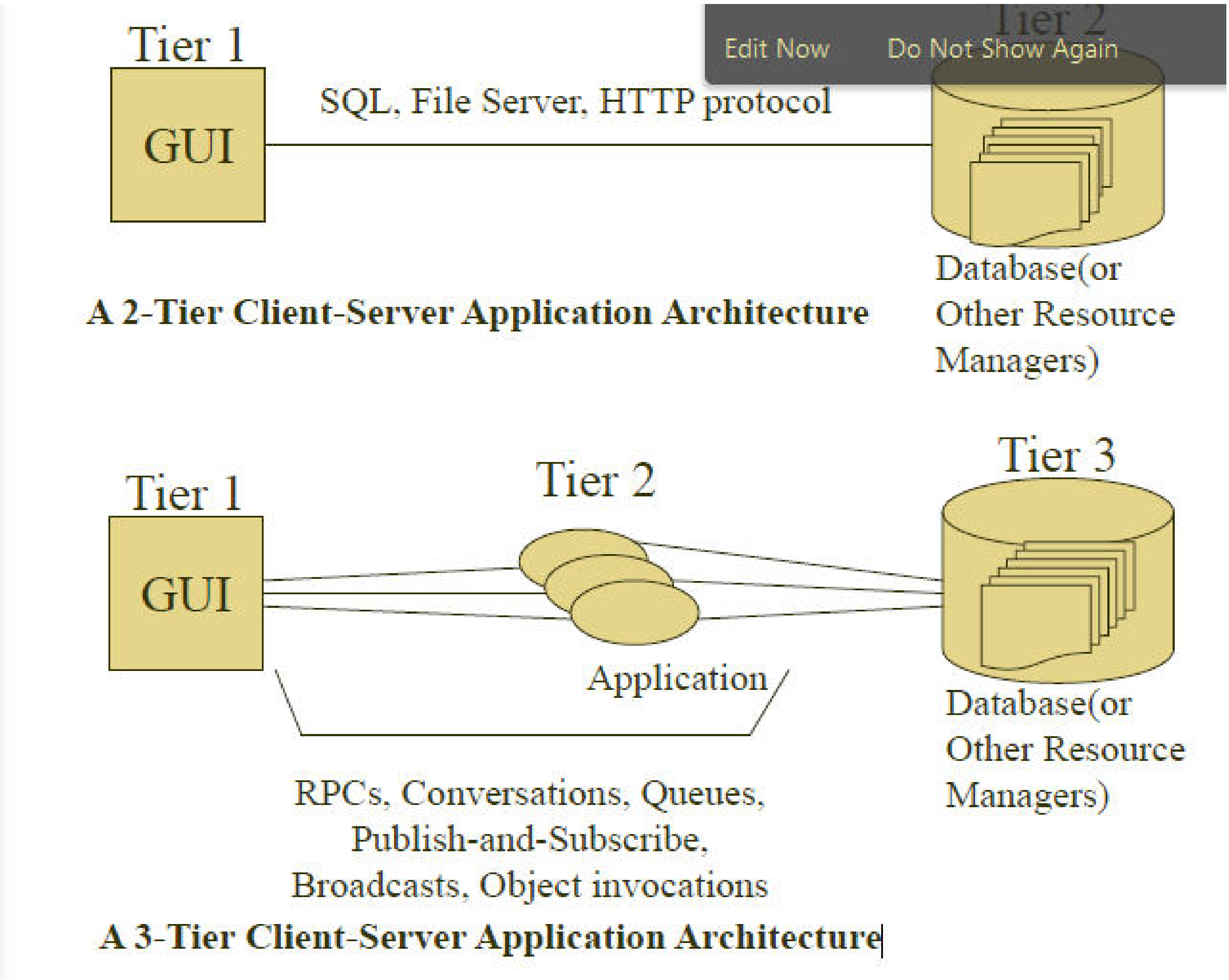


This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!--  Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f2b90e9460065a4c2f4e30e065b245e51e.  -->
<!--  Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f2b90e9460065a4c2f4e30e065b245e51e.  -->
▼<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://rpc_helloworld/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://rpc_helloworld/" name="HelloWorldImplService">
   <types/>
  ▼<message name="getHelloWorld">
     <part name="arg0" type="xsd:string"/>
   </message>
  ▼<message name="getHelloWorldResponse">
     <part name="return" type="xsd:string"/>
   </message>
  ▼<portType name="HelloWorld">
    ▼<operation name="getHelloWorld">
       <input wsam:Action="http://rpc_helloworld/HelloWorld/getHelloWorldRequest" message="tns:getHelloWorld"/>
       <output wsam:Action="http://rpc_helloworld/HelloWorld/getHelloWorldResponse" message="tns:getHelloWorldResponse"/>
     </operation>
   </portType>
  ▼<binding name="HelloWorldImplPortBinding" type="tns:HelloWorld">
     <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    ▼<operation name="getHelloWorld">
       <soap:operation soapAction=""/>
      ▼<input>
         <soap:body use="literal" namespace="http://rpc_helloworld/"/>
       </input>
      ▼<output>
         <soap:body use="literal" namespace="http://rpc_helloworld/"/>
       </output>
     </operation>
   </binding>
  ▼<service name="HelloWorldImplService">
    ▼<port name="HelloWorldImplPort" binding="tns:HelloWorldImplPortBinding">
       <soap:address location="http://localhost:7779/ws/hello"/>
     </port>
   </service>
 </definitions>
```

Develop an application using 3-tier architectures




Tier 1
GUI
SQL, File Server, HTTP protocol
Tier 2
Edit Now
Do Not Show Again
Database(or Other Resource Managers)
Tier 1
GUI
Tier 2
Tier 3
Application
RPCs, Conversations, Queues, Publish-and-Subscribe, Broadcasts, Object invocations
Database(or Other Resource Managers)

A 2-Tier Client-Server Application Architecture

A 3-Tier Client-Server Application Architecture