United States International University

MIS 6060: DISTRIBUTED COMPUTING & INTERNET TECHNOLOGY

Lab Exercise 3 – Implementing a Client Server Communication Model

Objective

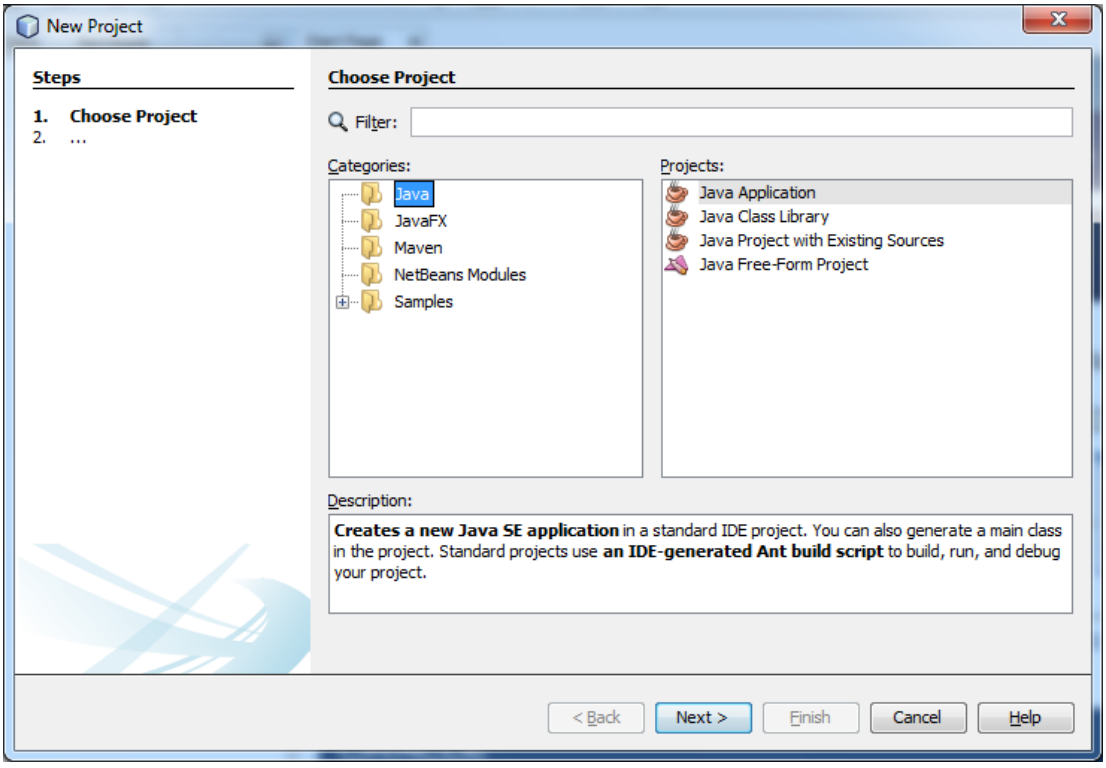
Write a program for implementing Client Server communication model.
A client server based program using TCP to find if the number entered is prime.

Requirements

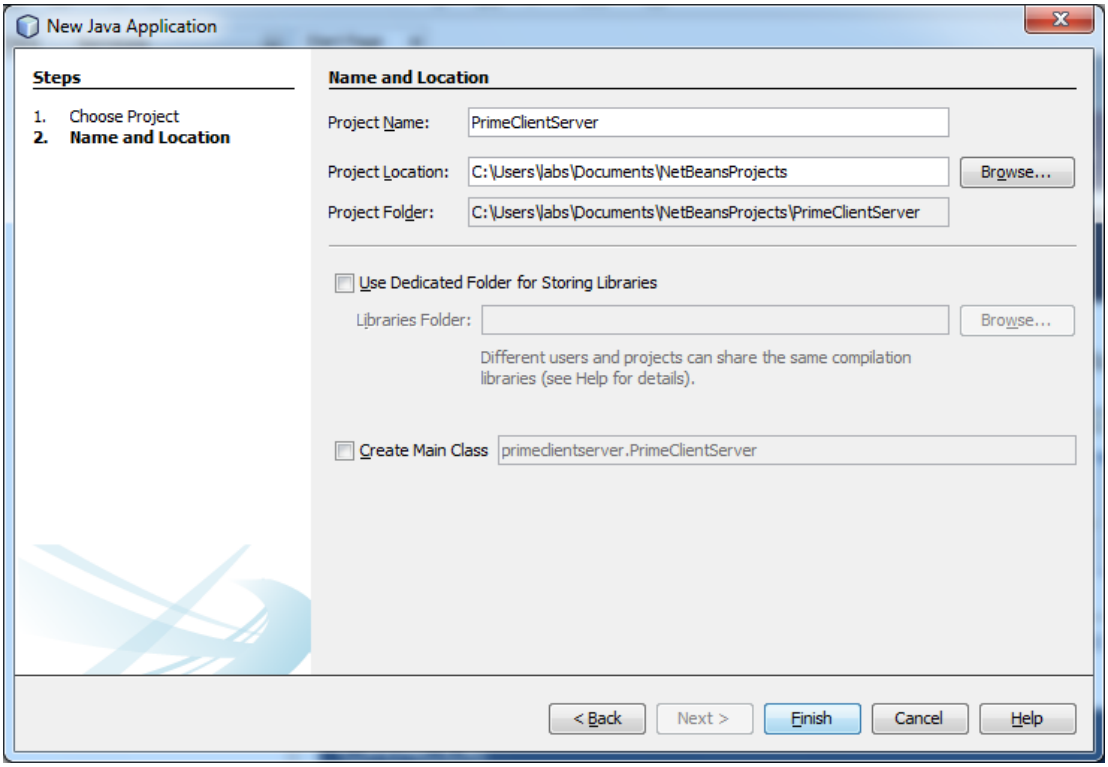
NetBeans will be used to demonstrate this exercise. For the client server communication to happen sockets are required. Sockets are an interprocess communication mechanism originally defined in BSD 4.x UNIX that support communication between processes on different computers. A socket is one endpoint of a two-way communication link between two processes running on a network. Sockets are used in pairs, one for each of the communicating processes. A socket address consists of an IP address and a port number so that the TCP layer can identify which process should receive the data.

Step 1

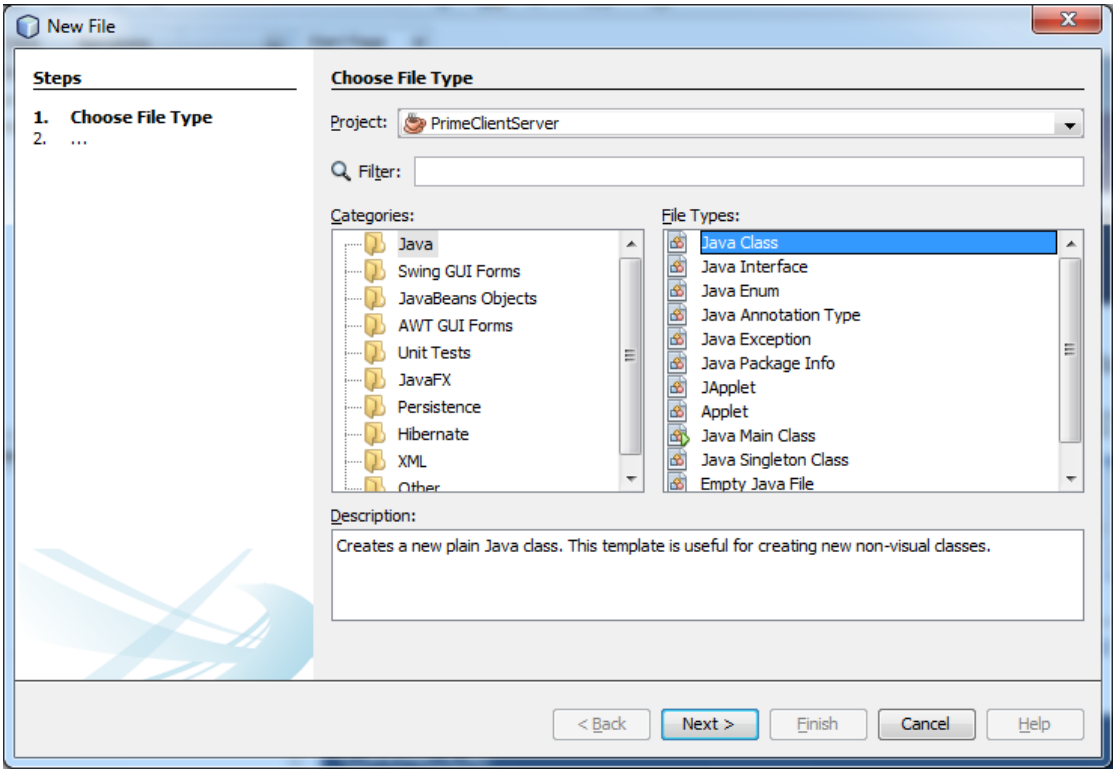
Run the NetBeans program and create a new project.

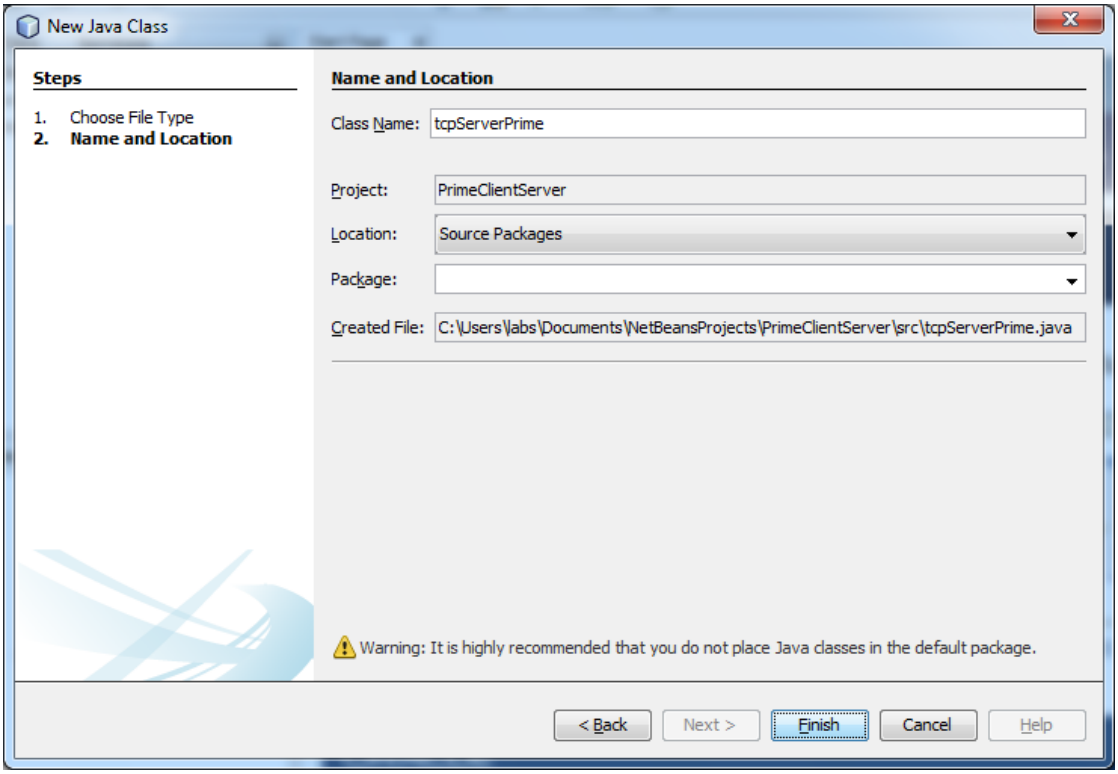


Name the project PrimeClientServer

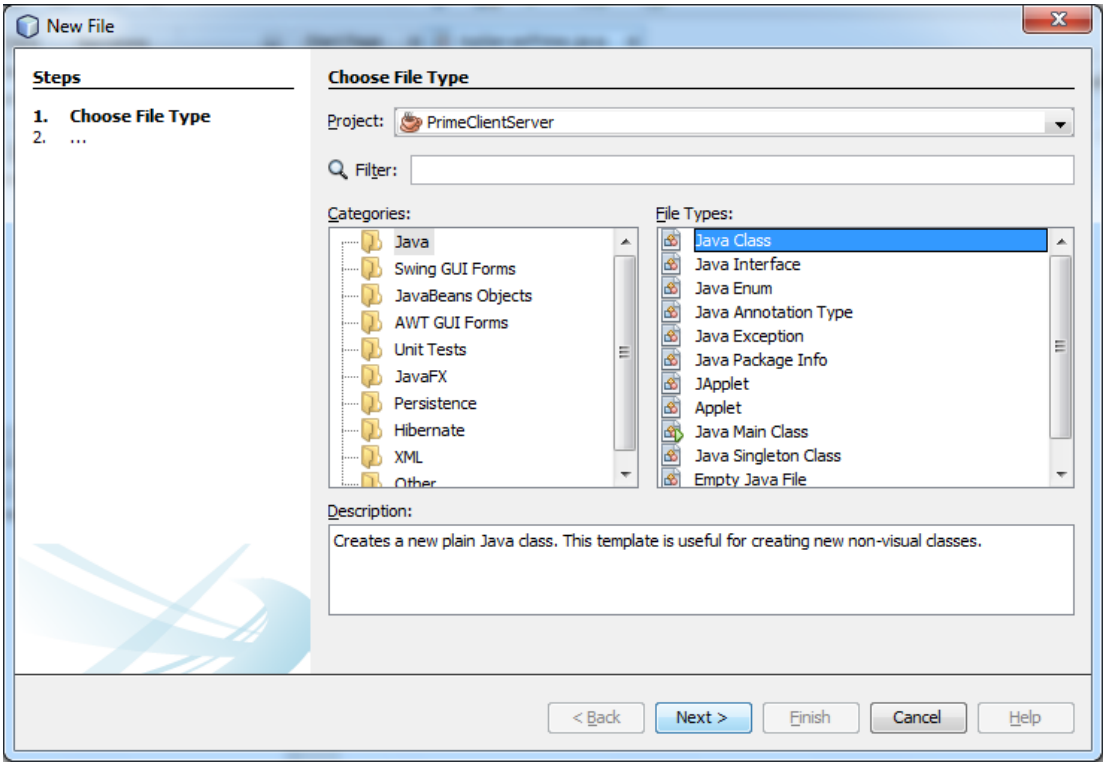


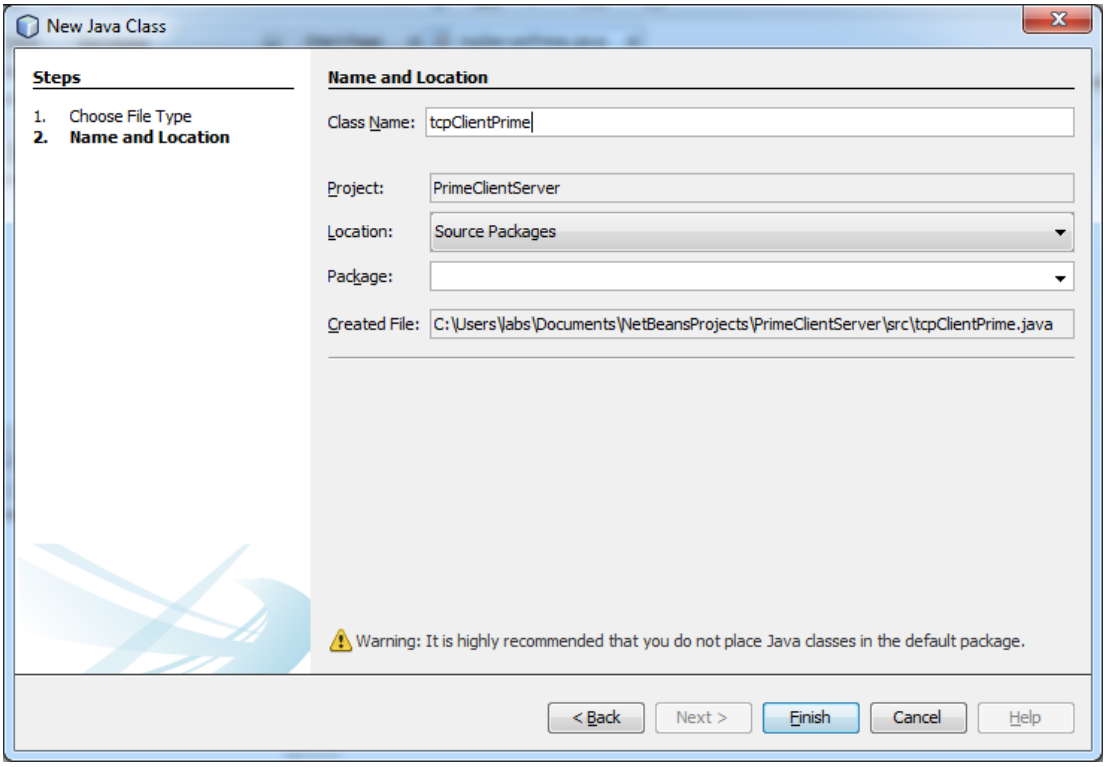
Create a new class and name it **tcpPrimeServer**

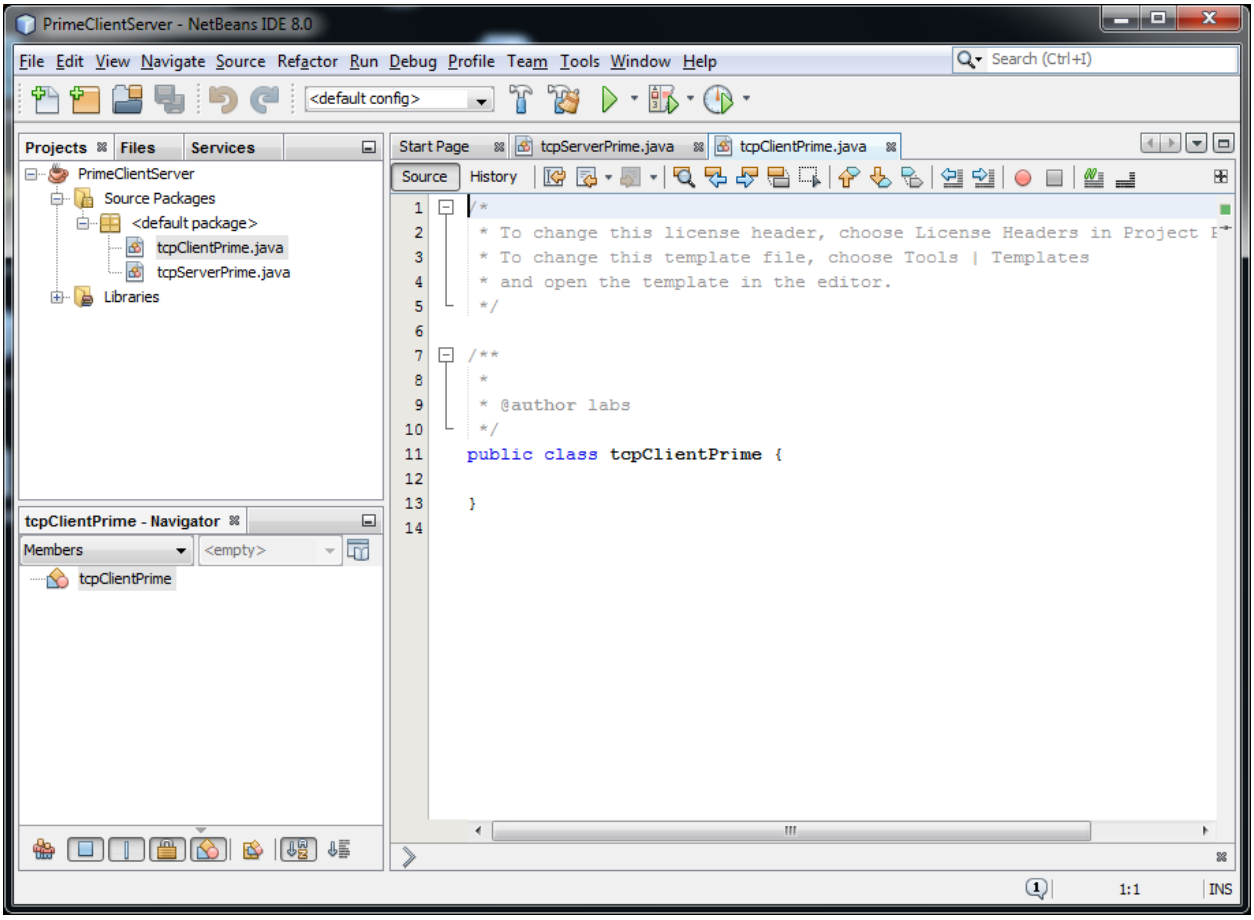




Create another java class and name it **tcpClientPrime**







Step 2

Double click on the tcpServerPrime.java file. We need to add the code for the server. A socket needs to be created to allow communication between the client and server.

Server Code

```
//Add the necessary packages

import java.net.*;

import java.io.*;


public class tcpServerPrime {


   public static void main(String args[])

        {                    try

                    {

//create a new socket
```

```
                        ServerSocket ss = new ServerSocket(8001);
                        System.out.println("Server Started.....");
                        Socket s = ss.accept();
// writing a component that needs to read input from a stream
                        DataInputStream in = new DataInputStream(s.getInputStream());
        int x= in.readInt();
//writing a component that needs to write output to a stream
  DataOutputStream otc = new DataOutputStream(s.getOutputStream());
                        int y = x/2;
                        if(x ==1 || x ==2 || x ==3)
                        {
                                otc.writeUTF(x + "is Prime");
                                System.exit(0);
                        }
                        for(int i=2; i<=y; i++)
                        {
                                if(x%i != 0)
                                {
                                        otc.writeUTF(x + " is Prime");
                                }
                                else
                                {
                                        otc.writeUTF(x + " is not Prime");
                                }
                        }
                }
                catch(Exception e)
                {
                        System.out.println(e.toString());
```

```
                }
        }

}
```

Client code

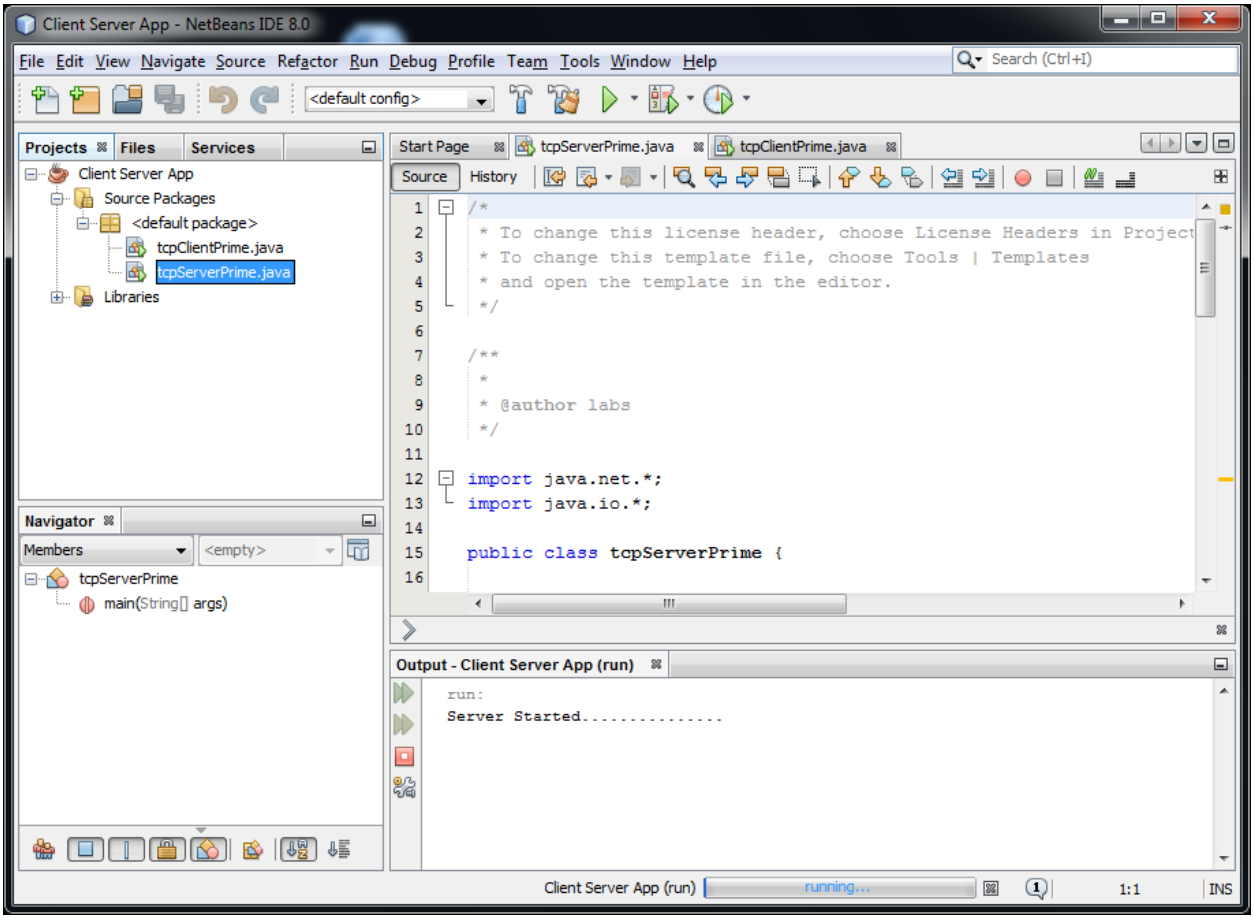
Double click on the tcpClientPrime.java file. We need to add the code for the client. A socket needs to be created to allow communication to the server.

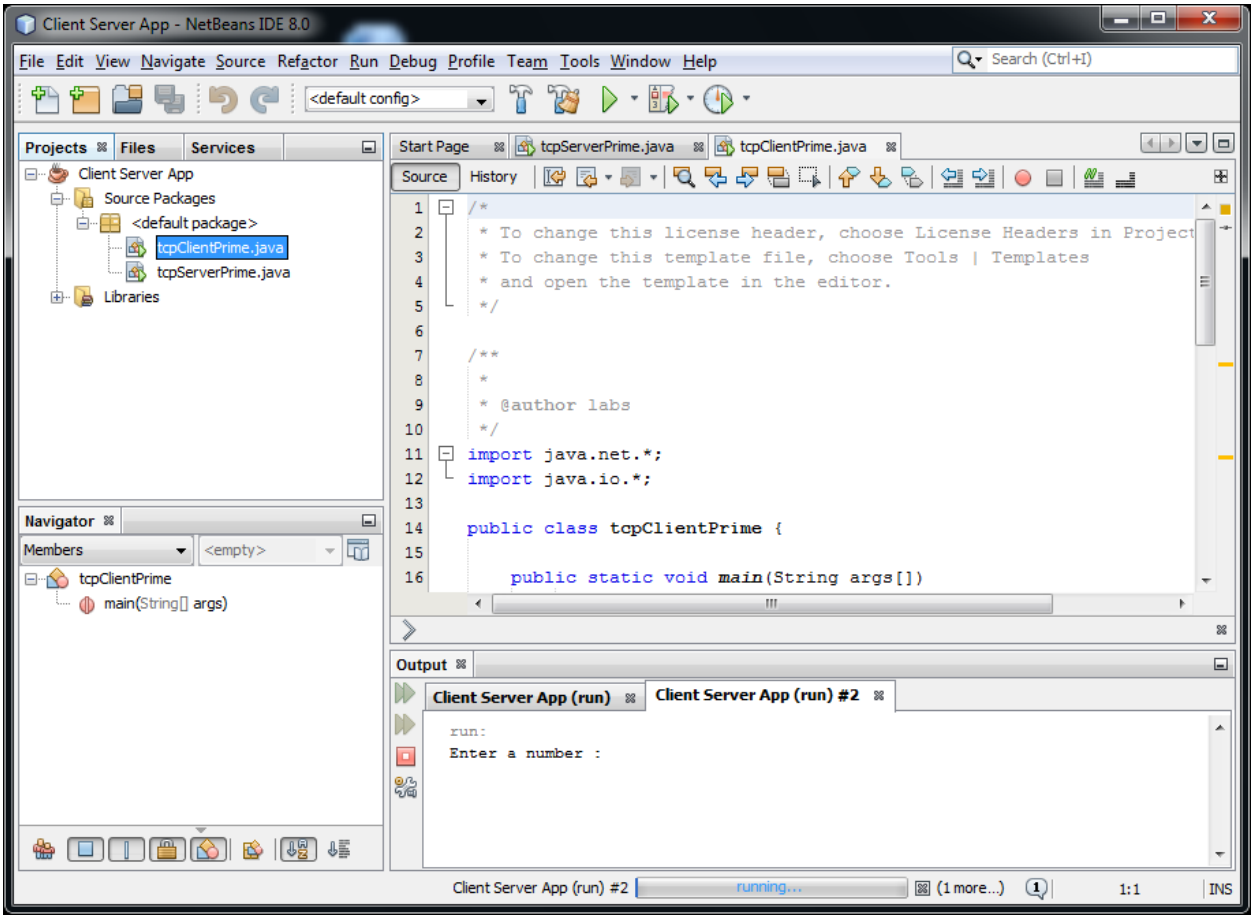
```
*/
//add the necessary packages
import java.net.*;
import java.io.*;


public class tcpClientPrime {


   public static void main(String args[])
        {                   try
                {
//create a socket
                        Socket cs = new Socket("LocalHost",8001);
// writing a component that needs to read input from a stream
                        BufferedReader infu = new BufferedReader(new
InputStreamReader(System.in));
                        System.out.println("Enter a number : ");
                        int a = Integer.parseInt(infu.readLine());
//writing a component that needs to write output to a stream
                        DataOutputStream out = new
DataOutputStream(cs.getOutputStream());
                        out.writeInt(a);
```

```
   DataInputStream in = new DataInputStream(cs.getInputStream());
//Print to the console
                        System.out.println(in.readUTF());
                        cs.close();
                }
                catch(Exception e)
                {
                        System.out.println(e.toString());
                }
        }

}
```

Step 3

Run the server file first. Once successful, run the client file. Enter a number and press the return key. View your results.


Client Server App - NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
Client Server App
Source Packages
<default package>
tcpClientPrime.java
tcpServerPrime.java
Libraries
Navigator
Members
<empty>
tcpServerPrime
main(String[] args)
Start Page
tcpServerPrime.java
tcpClientPrime.java
Source
History
1 /*
2 * To change this license header, choose License Headers in Project
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6
7 /**
8 *
9 * @author labs
10 */
11
12 import java.net.*;
13 import java.io.*;
14
15 public class tcpServerPrime {
16
Output - Client Server App (run)
run:
Server Started.....
Client Server App (run)
running...
1:1
INS



Lab Exercise 3 B – implementing Client Server chat.

Objective: A client server TCP based chatting application.

Step1:

Create a new project in netbeans name it **Chat Application.** Create a new class and name it ChatServer. Import the packages. Add the following highlighted code to the file.

Code:

1. **ChatServer.java**

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

class ChatServer
{
        public static void main(String args[])
```

```
        {
try
                {
                        ServerSocket ss = new ServerSocket(8000);
                        System.out.println("Waiting for client to connect..");
                        Socket s = ss.accept();
                        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
  DataOutputStream out = new DataOutputStream(s.getOutputStream());
                        DataInputStream in = new DataInputStream(s.getInputStream());
                        String receive, send;
                        while((receive = in.readLine()) != null)
                        {
                                if(receive.equals("STOP"))
                        break;
                                System.out.println("Client Says : "+receive);
                        System.out.print("Server Says : ");
send = br.readLine();
                                out.writeBytes(send+"\n");
                        }
                        br.close();
in.close();                     out.close();
                        s.close();
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

Create a new class and name it ChatClient. Import the packages. Add the following highlighted code to the file.

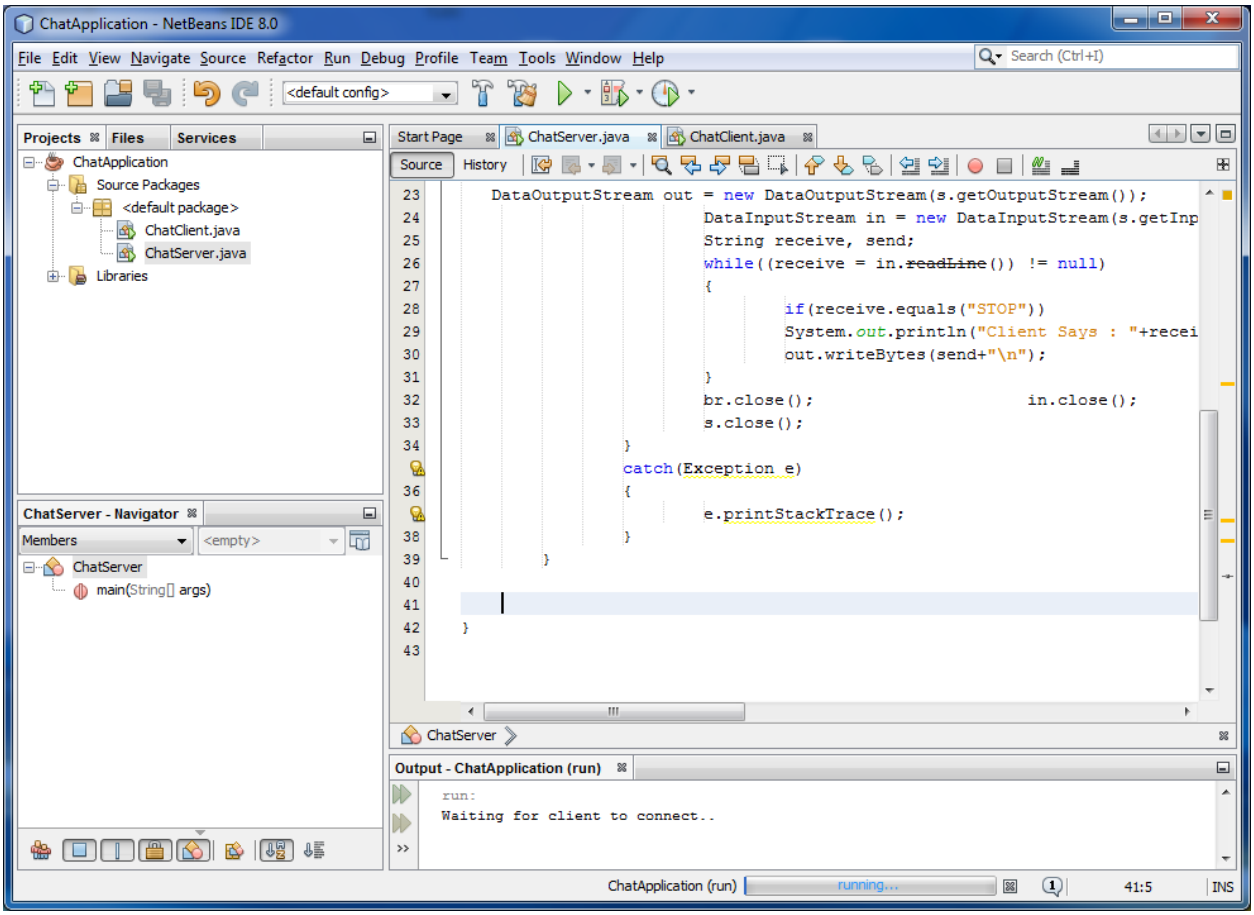
2. <u>ChatClient.java</u>

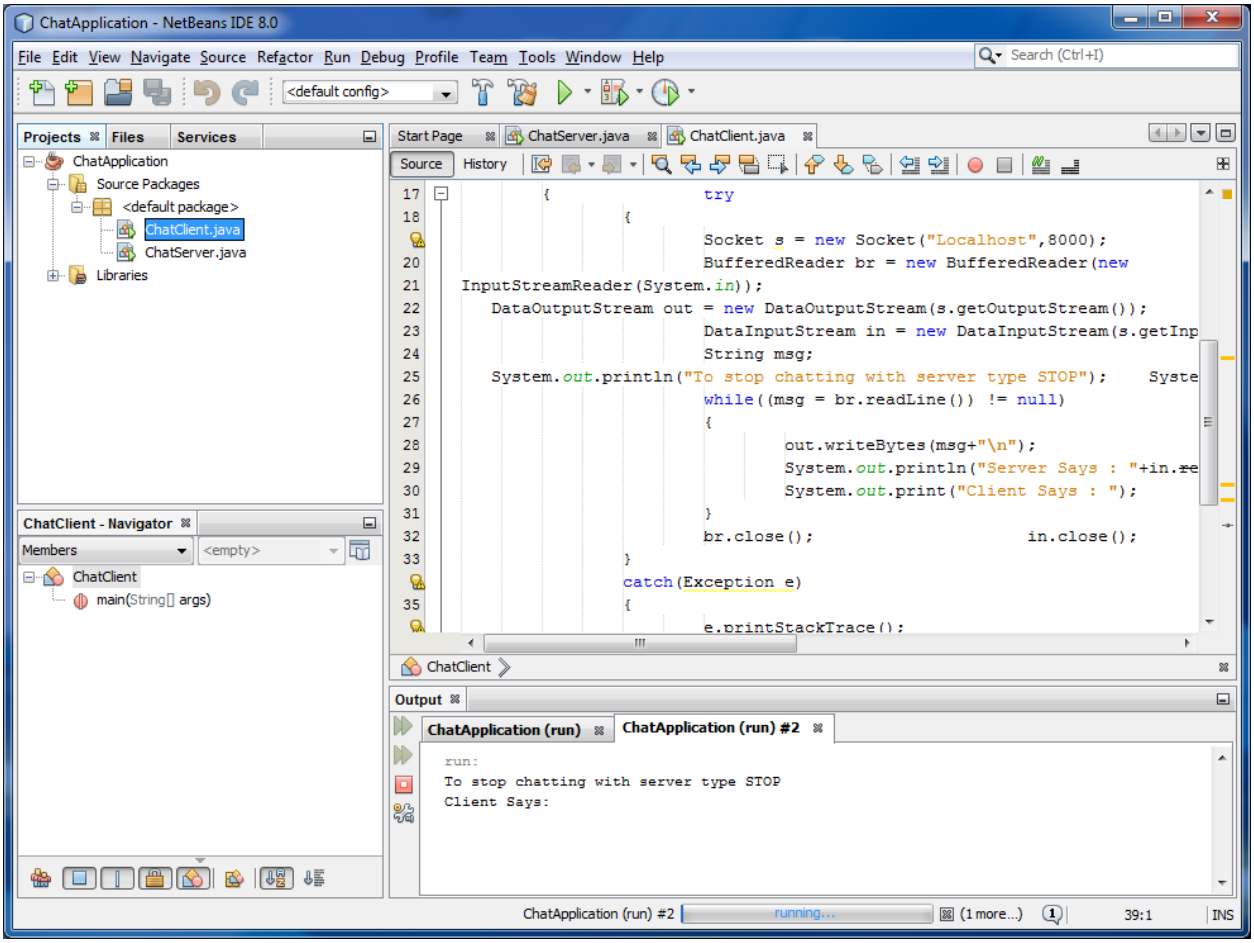
```
import java.net.*;
import java.io.*;

class ChatClient
{
        public static void main(String args[])
        {
try
```

```
                {
                        Socket s = new Socket("Localhost",8000);
                        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
  DataOutputStream out = new DataOutputStream(s.getOutputStream());
                        DataInputStream in = new DataInputStream(s.getInputStream());
                        String msg;
  System.out.println("To stop chatting with server type STOP");    System.out.print("Client Says:
");
                        while((msg = br.readLine()) != null)
                        {
                                out.writeBytes(msg+"\n");
                if(msg.equals("STOP"))
        break;
                                System.out.println("Server Says : "+in.readLine());
                                System.out.print("Client Says : ");
                        }
                        br.close();
in.close();                     out.close();
        s.close();
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

Step2:

Run the ChatServer.java file first. Run the ChatClient.java file next.


ChatApplication - NetBeans IDE 8.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Projects
Files
Services
ChatApplication
Source Packages
<default package>
ChatClient.java
ChatServer.java
Libraries
ChatServer - Navigator
Members
<empty>
ChatServer
main(String[] args)
Start Page
ChatServer.java
ChatClient.java
Source
History
23 DataOutputStream out = new DataOutputStream(s.getOutputStream());
24 DataInputStream in = new DataInputStream(s.getInp
25 String receive, send;
26 while((receive = in.readLine()) != null)
27 {
28 if(receive.equals("STOP"))
29 System.out.println("Client Says : "+recei
30 out.writeBytes(send+"\n");
31 }
32 br.close(); in.close();
33 s.close();
34 }
36 catch(Exception e)
36 {
38 e.printStackTrace();
38 }
39 }
40
41
42 }
43
ChatServer
Output - ChatApplication (run)
run:
Waiting for client to connect..
ChatApplication (run)
running...
41:5
INS



You can now enter the conversations from the client side and it will be sent to the server and vice versa. To end the chat type STOP