EX. NO. 5a **APPLICATIONS USING TCP SOCKETS (FILE TRANSFER)**

Client :

```
import java.io.*;
import java.net.Socket;
public class Client {
private static DataOutputStream dataOutputStream = null;
private static DataInputStream dataInputStream = null;

public static void main(String[] args)
{
// Create Client Socket connect to port 900
try (Socket socket = new Socket("localhost", 900)) {

dataInputStream = new DataInputStream(
socket.getInputStream());
dataOutputStream = new DataOutputStream(
socket.getOutputStream());
System.out.println(
"Sending the File to the Server");
// Call SendFile Method
sendFile(
"C:/Users/MOHAN K/Documents/2022-2023 Even Semester - Academic calendar - II Years.pdf");

dataInputStream.close();
dataInputStream.close();
}
catch (Exception e) {
e.printStackTrace();
}
}

// sendFile function define here
private static void sendFile(String path)
throws Exception
{
int bytes = 0;
// Open the File where he located in your pc
File file = new File(path);
FileInputStream fileInputStream
= new FileInputStream(file);

// Here we send the File to Server
dataOutputStream.writeLong(file.length());
// Here we break file into chunks
```

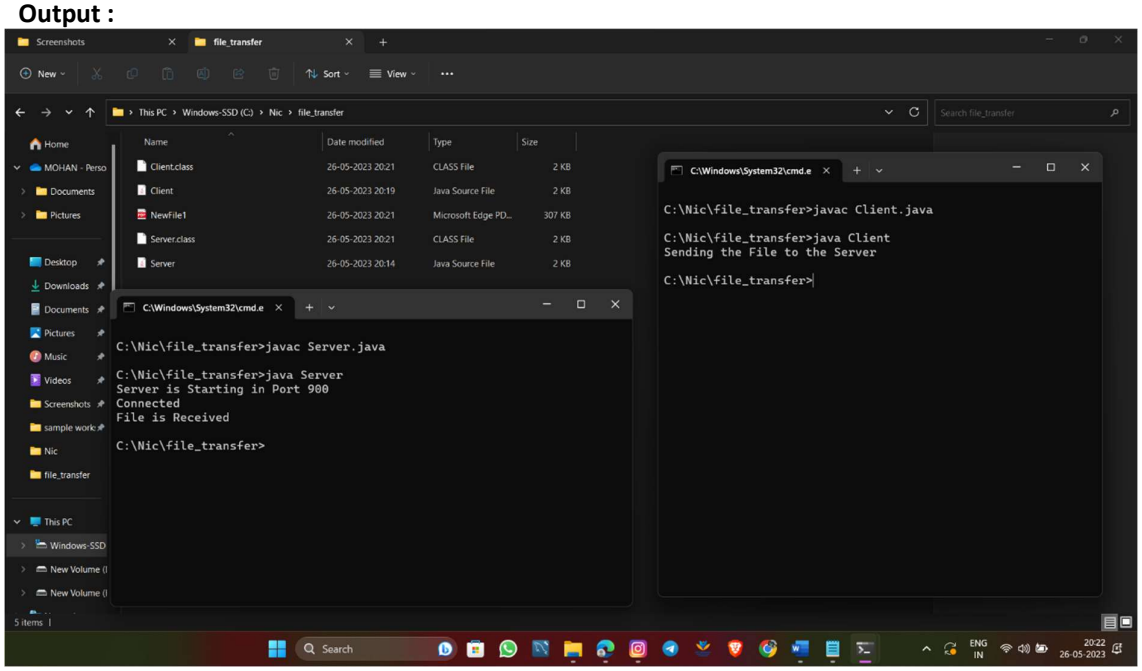
```
byte[] buffer = new byte[4 * 1024];
while ((bytes = fileInputStream.read(buffer))
!= -1) {
// Send the file to Server Socket
dataOutputStream.write(buffer, 0, bytes);
dataOutputStream.flush();
}
// close the file here
fileInputStream.close();
}
}
```

Server :

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
private static DataOutputStream dataOutputStream = null;
private static DataInputStream dataInputStream = null;
public static void main(String[] args)
{
// Here we define Server Socket running on port 900
try (ServerSocket serverSocket
= new ServerSocket(900)) {
System.out.println(
"Server is Starting in Port 900");
// Accept the Client request using accept method
Socket clientSocket = serverSocket.accept();
System.out.println("Connected");
dataInputStream = new DataInputStream(
clientSocket.getInputStream());
dataOutputStream = new DataOutputStream(
clientSocket.getOutputStream());
// Here we call receiveFile define new for that
// file
receiveFile("NewFile1.pdf");

dataInputStream.close();
dataOutputStream.close();
clientSocket.close();
}
catch (Exception e) {
e.printStackTrace();
```

```
}
}
// receive file function is start here
private static void receiveFile(String fileName)
throws Exception
{
int bytes = 0;
FileOutputStream fileOutputStream
= new FileOutputStream(fileName);

long size
= dataInputStream.readLong(); // read file size
byte[] buffer = new byte[4 * 1024];
while (size > 0
&& (bytes = dataInputStream.read(
buffer, 0,
(int)Math.min(buffer.length, size)))
!= -1) {
// Here we write the file using write method
fileOutputStream.write(buffer, 0, bytes);
size -= bytes; // read upto file size
}
// Here we received file
System.out.println("File is Received");
fileOutputStream.close();
}
}
```

Output :



EX. NO.5b **APPLICATIONS USING TCP SOCKETS (CHAT)**

Client:

```
import java.io.*;
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;
public class ChatSocketClient {
private Socket socket = null;
private InputStream inStream = null;
private OutputStream outStream = null;
public ChatSocketClient() {
}
public void createSocket() {
try {
socket = new Socket("localHost", 3339);
System.out.println("Connected");
inStream = socket.getInputStream();
outStream = socket.getOutputStream();
createReadThread();
createWriteThread();
} catch (UnknownHostException u) {
u.printStackTrace();
} catch (IOException io) {
io.printStackTrace();
}
}
public void createReadThread() {
Thread readThread = new Thread() {
public void run() {
while (socket.isConnected()) {
try {
byte[] readBuffer = new byte[200];
int num = inStream.read(readBuffer);
if (num > 0) {
byte[] arrayBytes = new byte[num];
System.arraycopy(readBuffer, 0, arrayBytes, 0, num);
String recvedMessage = new String(arrayBytes, "UTF-8");
System.out.println("Received message :" + recvedMessage);
}/* else {
// notify();
}*/
;
//System.arraycopy();
}catch (SocketException se){
System.exit(0);
} catch (IOException i) {
i.printStackTrace();
}
}
}
};
```

```
readThread.setPriority(Thread.MAX_PRIORITY);
readThread.start();
}
public void createWriteThread() {
Thread writeThread = new Thread() {
public void run() {
while (socket.isConnected()) {
try {
BufferedReader inputReader = new BufferedReader(new InputStreamReader(System.in));
sleep(100);
String typedMessage = inputReader.readLine();
if (typedMessage != null && typedMessage.length() > 0) {
synchronized (socket) {
outStream.write(typedMessage.getBytes("UTF-8"));
sleep(100);
}};
//System.arraycopy();
} catch (IOException i) {
i.printStackTrace();
} catch (InterruptedException ie) {
ie.printStackTrace();
}}}};
writeThread.setPriority(Thread.MAX_PRIORITY);
writeThread.start();
}
public static void main(String[] args) throws Exception {
ChatSocketClient myChatClient = new ChatSocketClient();
myChatClient.createSocket();
/*myChatClient.createReadThread();
myChatClient.createWriteThread();*/
}}
```

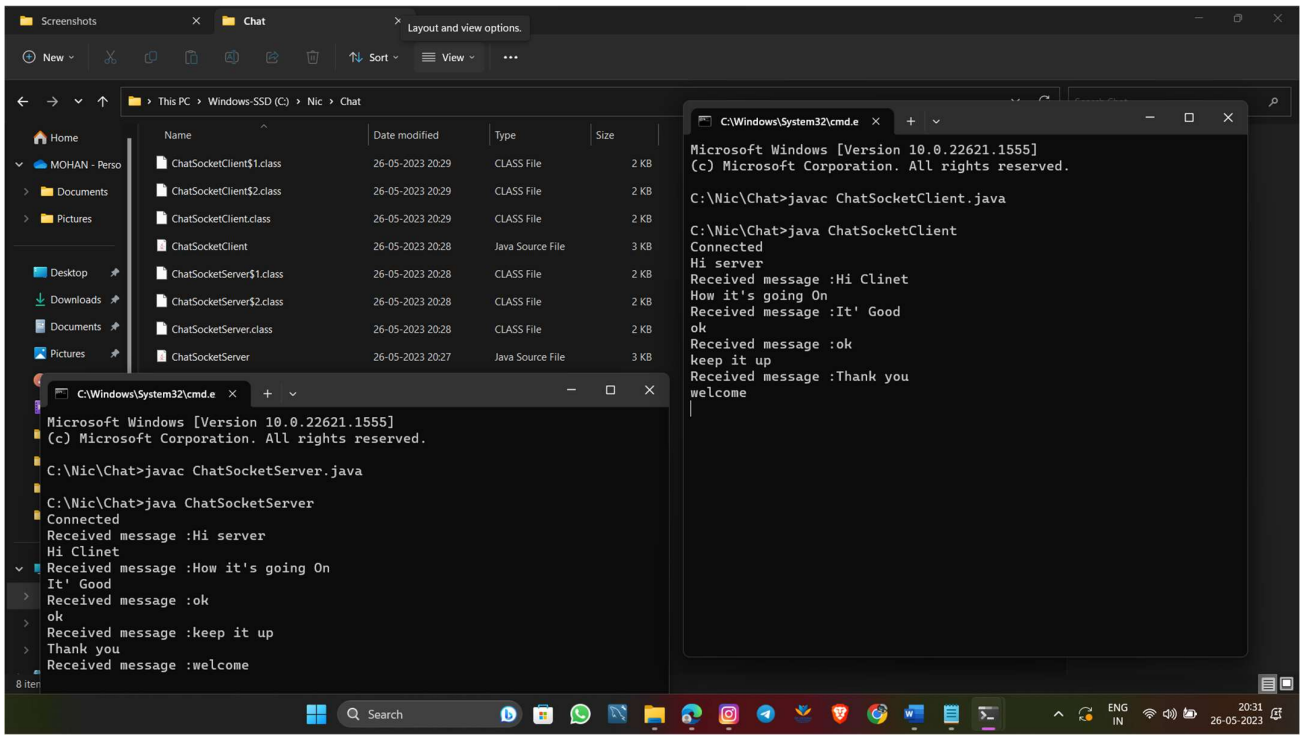
Server :

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
public class ChatSocketServer {
private ServerSocket severSocket = null;
private Socket socket = null;
private InputStream inStream = null;
private OutputStream outStream = null;
public ChatSocketServer() {}
public void createSocket() {
try {
ServerSocket serverSocket = new ServerSocket(3339);
while (true) {
socket = serverSocket.accept();
inStream = socket.getInputStream();
outStream = socket.getOutputStream();
System.out.println("Connected");
createReadThread();
```

```
createWriteThread();}
} catch (IOException io) {
io.printStackTrace();}
}
public void createReadThread() {
Thread readThread = new Thread() {
public void run() {
while (socket.isConnected()) {
try {
byte[] readBuffer = new byte[200];
int num = inStream.read(readBuffer);
if (num > 0) {
byte[] arrayBytes = new byte[num];
System.arraycopy(readBuffer, 0, arrayBytes, 0, num);
String recvedMessage = new String(arrayBytes, "UTF-8");
System.out.println("Received message :" + recvedMessage);
} else {
notify();};
//System.arraycopy();
} catch (SocketException se) {
System.exit(0);
} catch (IOException i) {
i.printStackTrace();
}}}};
readThread.setPriority(Thread.MAX_PRIORITY);
readThread.start();
}
public void createWriteThread() {
Thread writeThread = new Thread() {
public void run() {
while (socket.isConnected()) {
try {
BufferedReader inputReader = new BufferedReader(new InputStreamReader(System.in));
sleep(100);
String typedMessage = inputReader.readLine();
if (typedMessage != null && typedMessage.length() > 0) {
synchronized (socket) {
outStream.write(typedMessage.getBytes("UTF-8"));
sleep(100);
}
}/* else {
notify();
}*/
;
//System.arraycopy();

} catch (IOException i) {
i.printStackTrace();
} catch (InterruptedException ie) {
ie.printStackTrace();
}
```

```
}
}
};
writeThread.setPriority(Thread.MAX_PRIORITY);
writeThread.start();

}

public static void main(String[] args) {
ChatSocketServer chatServer = new ChatSocketServer();
chatServer.createSocket();

}
}
```

Output :

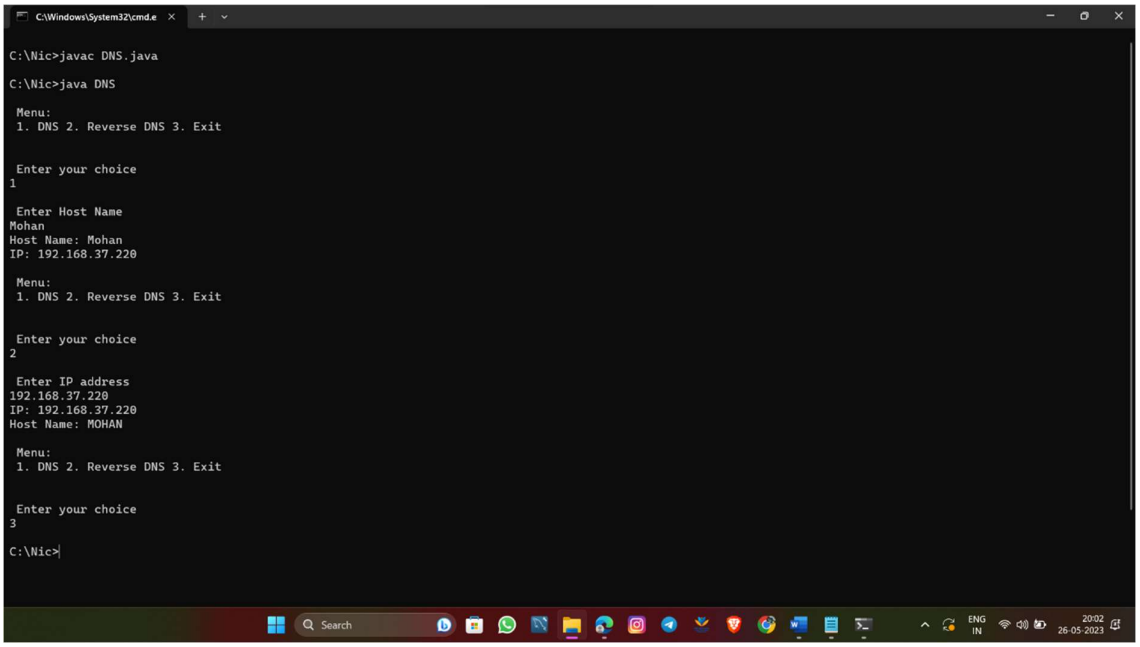


EX. NO.5d **APPLICATIONS (DNS)**

```
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));
 }
}
```

Output :



EX. NO.5c APPLICATIONS USING TCP SOCKETS (CONCURRENT SERVER)

Clinet :

```
import java.io.*;
import java.net.*;
import java.util.*;
// Client class
class Client {
// driver code
public static void main(String[] args)
{
// establish a connection by providing host and port
// number
try (Socket socket = new Socket("localhost", 1234)) {
// writing to server
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
// reading from server
BufferedReader in= new BufferedReader(new InputStreamReader(socket.getInputStream()));
// object of scanner class
Scanner sc = new Scanner(System.in);
String line = null;
while (!"exit".equalsIgnoreCase(line)) {
// reading from user
line = sc.nextLine();
// sending the user input to server
out.println(line);
out.flush();
// displaying server reply
System.out.println("Server replied "+ in.readLine());
}
// closing the scanner object
sc.close();
}
catch (IOException e) {
e.printStackTrace();
}
}
}
```

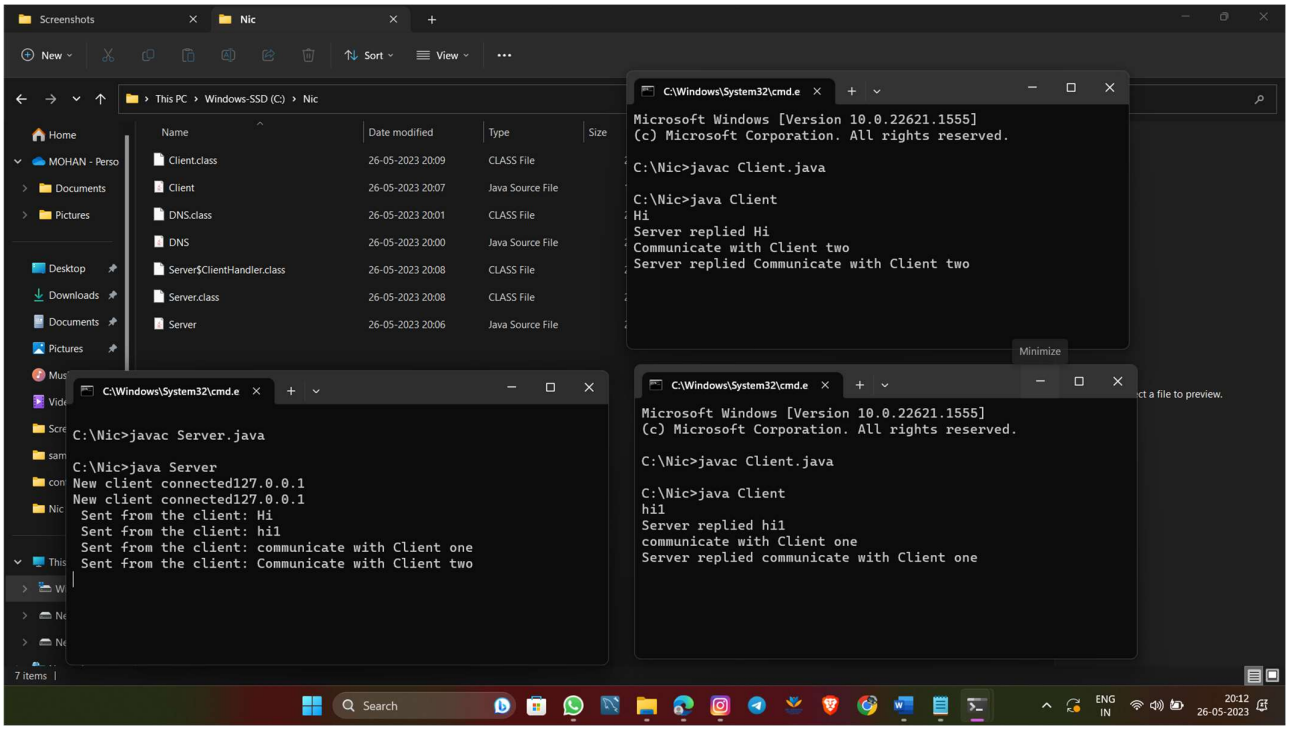
Server :

```
import java.io.*;
import java.net.*;
// Server class
class Server {
public static void main(String[] args)
{
ServerSocket server = null;
```

```
try {
// server is listening on port 1234
server = new ServerSocket(1234);
server.setReuseAddress(true);
// running infinite loop for getting
// client request
while (true) {
// socket object to receive incoming client
// requests
Socket client = server.accept();
// Displaying that new client is connected
// to server
System.out.println("New client connected"+ client.getInetAddress().getHostAddress());
// create a new thread object
ClientHandler clientSock= new ClientHandler(client);
// This thread will handle the client
// separately
new Thread(clientSock).start();
}
}
catch (IOException e) {
e.printStackTrace();
}
finally {
if (server != null) {
try {
server.close();
}
catch (IOException e) {
e.printStackTrace();
}}}}
// ClientHandler class
private static class ClientHandler implements Runnable {
private final Socket clientSocket;
// Constructor
public ClientHandler(Socket socket)
{
this.clientSocket = socket;
}
public void run(){
PrintWriter out = null;
BufferedReader in = null;
try {
// get the outputstream of client
out = new PrintWriter(clientSocket.getOutputStream(), true);
// get the inputstream of client
in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
String line;
```

```
while ((line = in.readLine()) != null) {
// writing the received message from
// client
System.out.printf(" Sent from the client: %s\n",line);
out.println(line);
}
}
catch (IOException e) {
e.printStackTrace();}
finally {
try {
if (out != null) {
out.close();}
if (in != null) {
in.close();
clientSocket.close();}}
catch (IOException e) {
e.printStackTrace();
}
}
}
}
}
```

Output :



EX. NO.5e CREATE A SOCKET FOR HTTP FOR WEBPAGE UPLOAD AND DOWNLOAD

Client :

```
import java.io.*;
import java.net.Socket;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;

public class Client {
   public static void main(String[] args) {
      Socket soc = null;
      try {
         soc = new Socket("localhost", 4000);
         System.out.println("Client is running.");

         // Read image from disk
         BufferedImage img = ImageIO.read(new File("C:/Users/MOHAN K/Desktop/lion.jpg"));
         ByteArrayOutputStream baos = new ByteArrayOutputStream();
         ImageIO.write(img, "jpg", baos);
         baos.flush();
         byte[] bytes = baos.toByteArray();
         baos.close();

         // Send image to server
         OutputStream out = soc.getOutputStream();
         DataOutputStream dos = new DataOutputStream(out);
         dos.writeInt(bytes.length);
         dos.write(bytes, 0, bytes.length);
         System.out.println("Image sent to server.");
      } catch (IOException e) {
         e.printStackTrace();
      } finally {
         try {
            if (soc != null)
               soc.close();
         } catch (IOException e) {
            e.printStackTrace();
         }
      }
   }
}
```

Server :

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import javax.swing.*;

public class Server {
   public static void main(String[] args) {
      ServerSocket server = null;
      Socket socket = null;
      try {
         // Create Server Socket
         server = new ServerSocket(4000);
         System.out.println("Server Waiting for image");

         // Accept client connection
         socket = server.accept();
         System.out.println("Client connected.");

         // Read image data from client
         InputStream in = socket.getInputStream();
         DataInputStream dis = new DataInputStream(in);
         int len = dis.readInt();
         System.out.println("Image Size: " + len/1024 + "KB");
         byte[] data = new byte[len];
         dis.readFully(data);
         dis.close();
         in.close();

         // Convert byte array to BufferedImage
         ByteArrayInputStream ian = new ByteArrayInputStream(data);
         BufferedImage bImage = ImageIO.read(ian);

         // Create a frame window entitled "Server" and display the image
         JFrame f = new JFrame("Server");
         ImageIcon icon = new ImageIcon(bImage);
         JLabel label = new JLabel(icon);
         f.getContentPane().add(label);
         f.pack();
         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         f.setVisible(true);
      } catch (IOException e) {
         e.printStackTrace();
      } finally {
         try {
            if (socket != null)
```

```
                socket.close();
            if (server != null)
                server.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
  }
}
```

Output :