TITLE: CLIENT-SERVER APPLICATION (SOCKET)

OBJECTIVES:

→ To understand the concept of Socket Programming in Java.

→ To demonstrate a basic client-server architecture where two programs communicate over the network.

→ To learn to use Socket and ServerSocket.

THEORY:

Socket (Client Side):

→ A socket is one endpoint of a two-way communication link between two programs running on the network.

→ The java.net.Socket class represents the client side of the connection.

→ The client must know the IP Address and port number of the server.

Server Socket (Server Side)

→ The java.net.ServerSocket class is used on the server side.

→ It listens for incoming connection requests from clients on a specific port.

-> The accept() method is a blocking call that waits until a client connects ; once connected it returns socket object.

SOURCE CODE:

Server.java

```
package labs;

import java.*;

public class Server {

    public static void main (String[] args){

        try{

            ServerSocket ss = new ServerSocket (6666);

            Socket s = ss.accept();

            System.out.println ("Client connected");

            System.out.println ("-----------");

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

            String message = dis.readUTF();

            System.out.println("Message received from Client:"
                               + message);

            System.out.println ("----------");
```

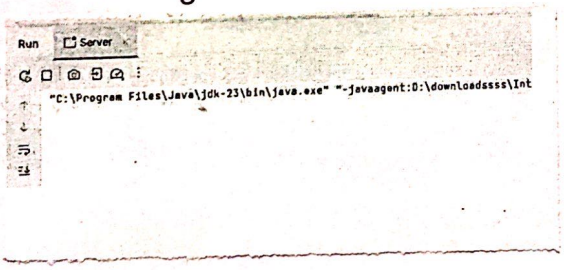
```
            System.out.println("Sending ack to client");
            DataOutputStream dos = new DataOutputStream(s.getOut
                                                putStream());
            dos.writeUTF("Hey Client! Your message has been
                                    received!");
            dos.flush();
            dos.close();
        } catch (Exception e){
            System.out.println(e);
        }
    }
}
```

Client.java

```
package labs;
import java.*;
public class Client{
    public static void main(String[] args){
        try{
            Socket s = new Socket("localhost", 6666);
            DataOutputStream dos = new DataOutputStream(
                                    s.getOutputStream());
            DataInputStream dis = new DataInputStream(
                                    s.getInputStream());
```

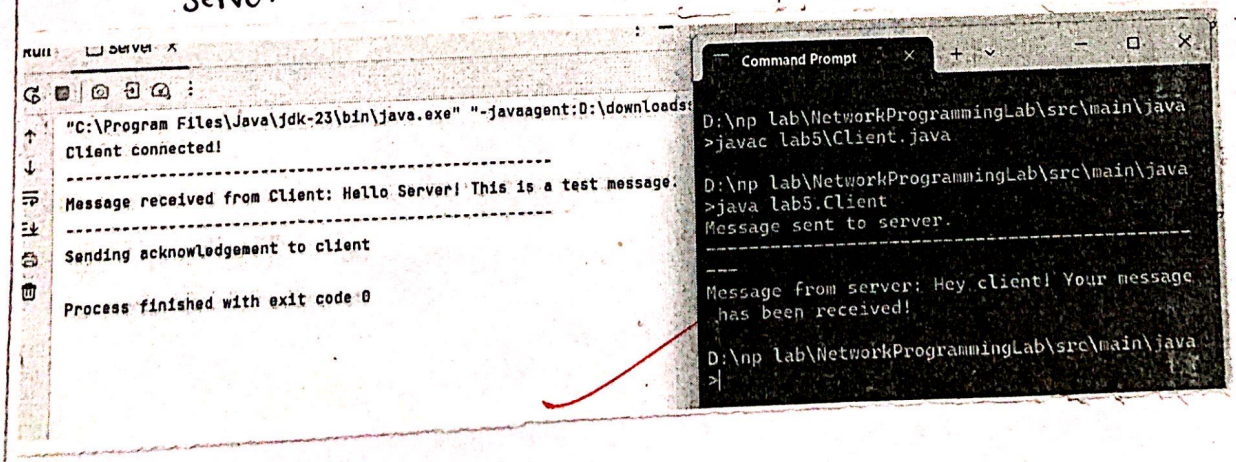
```
dos.writeUTF("Hello Server! This is test message");
System.out.println("Message sent to server");
System.out.println(" ---------- ----");
String message = dis.readUTF();
System.out.println("Message from server :" + message);
dos.flush();
dos.close();
s.close();
} catch(Exception e){
    system.out.println(e);
}
}
}
```

Running Server:



Server

Running client from terminal



CONCLUSION:

In this lab, we successfully implemented a Client-Server application using Java Sockets. We utilized the ServerSocket class to establish a listening port on the server and the Socket class on the client to initiate the connection. We demonstrated how data streams (DataInputStream and DataOutputStream) are used.