

Don Bosco Institute of Technology, Kurla
Academic Year 2023-24

EXPERIMENT NO. 7

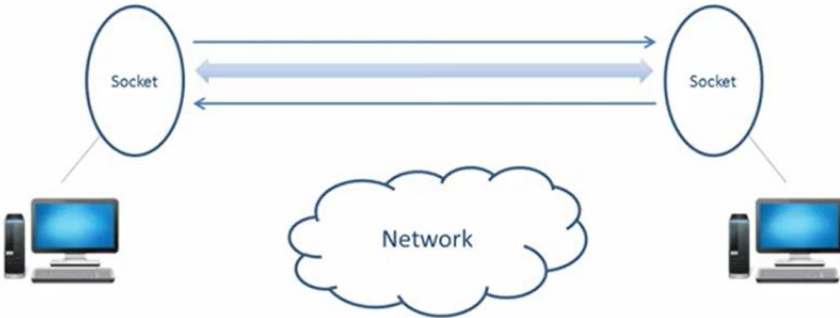
SEMESTER: V

DATE OF PERFORMANCE: 25th September 2024

SUBJECT: CN Lab

DATE OF SUBMISSION: 05th October 2024

NAME OF THE STUDENT: David James Elvuathingal ROLL NO.: 22

AIM	Socket Programming using TCP
LEARNING OBJECTIVE	Students will be able to develop a chat application using TCP protocol..
LEARNING OUTCOME	The students will be able to write client server chat application programs using TCP.
COURSE OUTCOME	CSL502.4: Illustrate socket programming for TCP connections for demonstrating networking concepts
PROGRAM OUTCOME	PO1,PO2,PO3,PO4,PO5,PO9,PO10,PSO1,PSO2,PSO3
BLOOM'S TAXONOMY LEVEL	Analyze
THEORY	<p>Java Socket Programming</p> <ul style="list-style-type: none">o Java Socket programming is used for communication between the applications running on different JRE.o Java Socket programming can be connection-oriented or connection-less.o Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming. <div style="text-align: center;"></div> <p>The client in socket programming must know two information:</p> <ol style="list-style-type: none">a. IP Address of Server, andb. Port number. <p>Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket.</p>

Don Bosco Institute of Technology, Kurla
Academic Year 2023-24

The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

#Socket class

A socket is simply an endpoint for communications between the machines.

The Socket class can be used to create a socket.

#ServerSocket class

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

Creating Server:

To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

```
ServerSocket ss=new ServerSocket(6666);
```

```
Socket s=ss.accept();//establishes connection and waits for the client
```

Creating Client:

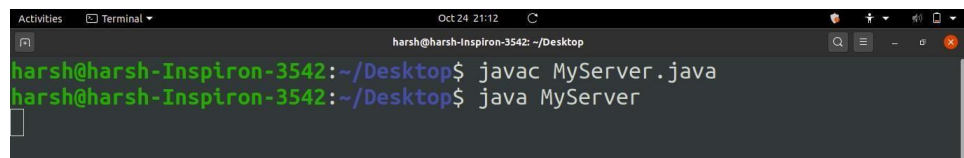
To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

```
Socket s=new Socket("localhost",6666);
```

Output:

To execute this program open two command prompts and execute each program at each command prompt as displayed in the below figures.

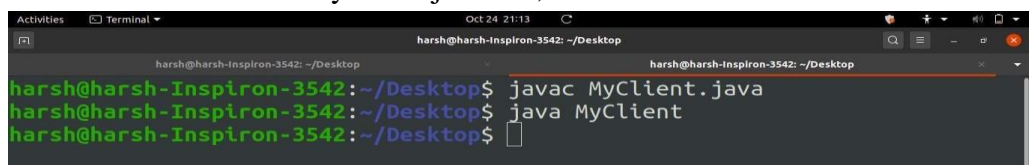
First run Myserver.java file in terminal/cmd,

A terminal window titled "Terminal" with a dark background. The prompt is "harsh@harsh-Inspiron-3542: ~/Desktop". The user enters "javac MyServer.java" and "java MyServer".

```
harsh@harsh-Inspiron-3542: ~/Desktop$ javac MyServer.java
harsh@harsh-Inspiron-3542: ~/Desktop$ java MyServer
```

Running MyServer.java

Then in new terminal/cmd run MyClient.java file,

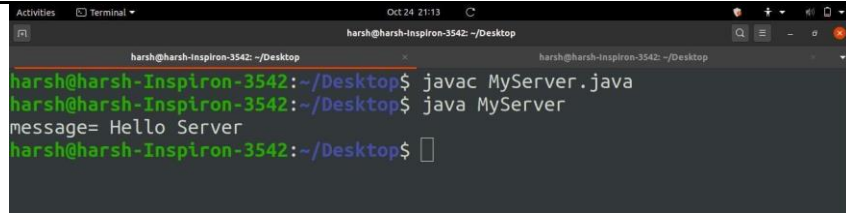
A terminal window titled "Terminal" with a dark background. The prompt is "harsh@harsh-Inspiron-3542: ~/Desktop". The user enters "javac MyClient.java" and "java MyClient".

```
harsh@harsh-Inspiron-3542: ~/Desktop$ javac MyClient.java
harsh@harsh-Inspiron-3542: ~/Desktop$ java MyClient
```

Running MyClient.java

As soon as you run MyClient program a message is sent to server and displayed in MyServer Terminal/CMD as shown below,

Don Bosco Institute of Technology, Kurla
Academic Year 2023-24



```
harsh@harsh-Inspiron-3542: ~/Desktop
harsh@harsh-Inspiron-3542:~/Desktop$ javac MyServer.java
harsh@harsh-Inspiron-3542:~/Desktop$ java MyServer
message= Hello Server
harsh@harsh-Inspiron-3542:~/Desktop$
```

Message displayed in MyServer after running MyClient

LAB EXERCISE

Write a program to build chat application using SOCKET programming (TCP).

Code:

MyServer.java

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

public class MyServer {
    public static void main(String[] args) {
        try (ServerSocket ss = new ServerSocket(6666)) {
            System.out.println("Server is listening on port 6666");
            Socket socket = ss.accept(); // Establish connection
            System.out.println("Client connected");

            // Create input and output streams
            DataInputStream dis = new DataInputStream(socket.getInputStream());
            DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

            // Create threads for reading and writing messages
            Thread readThread = new Thread() -> {
                try {
                    String msg;
                    while (true) {
                        msg = dis.readUTF();
                        System.out.println("Client: " + msg);
                        if (msg.equalsIgnoreCase("bye")) {
                            break;
                        }
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            };

            Thread writeThread = new Thread() -> {
                try {
                    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
                    String msg;
                    while (true) {
                        msg = reader.readLine();
```

Don Bosco Institute of Technology, Kurla
Academic Year 2023-24

```
        dos.writeUTF(msg);
        dos.flush();
        if (msg.equalsIgnoreCase("bye")) {
            break;
        }
    }
} catch (IOException e) {
    e.printStackTrace();
}
});

// Start both threads
readThread.start();
writeThread.start();

// Wait for both threads to finish
readThread.join();
writeThread.join();

// Close resources
dis.close();
dos.close();
socket.close();
System.out.println("Chat ended.");
} catch (IOException | InterruptedException e) {
    e.printStackTrace();
}
}
}
```

MyClient.java

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

public class MyClient {
    public static void main(String[] args) {
        try (Socket socket = new Socket("localhost", 6666)) {
            System.out.println("Connected to the server");

            // Create input and output streams
            DataInputStream dis = new DataInputStream(socket.getInputStream());
            DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

            // Create threads for reading and writing messages
            Thread readThread = new Thread(() -> {
                try {
                    String msg;
                    while (true) {
                        msg = dis.readUTF();
```

Don Bosco Institute of Technology, Kurla
Academic Year 2023-24

```
        System.out.println("Server: " + msg);
        if (msg.equalsIgnoreCase("bye")) {
            break;
        }
    }
} catch (IOException e) {
    e.printStackTrace();
}
});

Thread writeThread = new Thread() -> {
    try {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        String msg;
        while (true) {
            msg = reader.readLine();
            dos.writeUTF(msg);
            dos.flush();
            if (msg.equalsIgnoreCase("bye")) {
                break;
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
});

// Start both threads
readThread.start();
writeThread.start();

// Wait for both threads to finish
readThread.join();
writeThread.join();

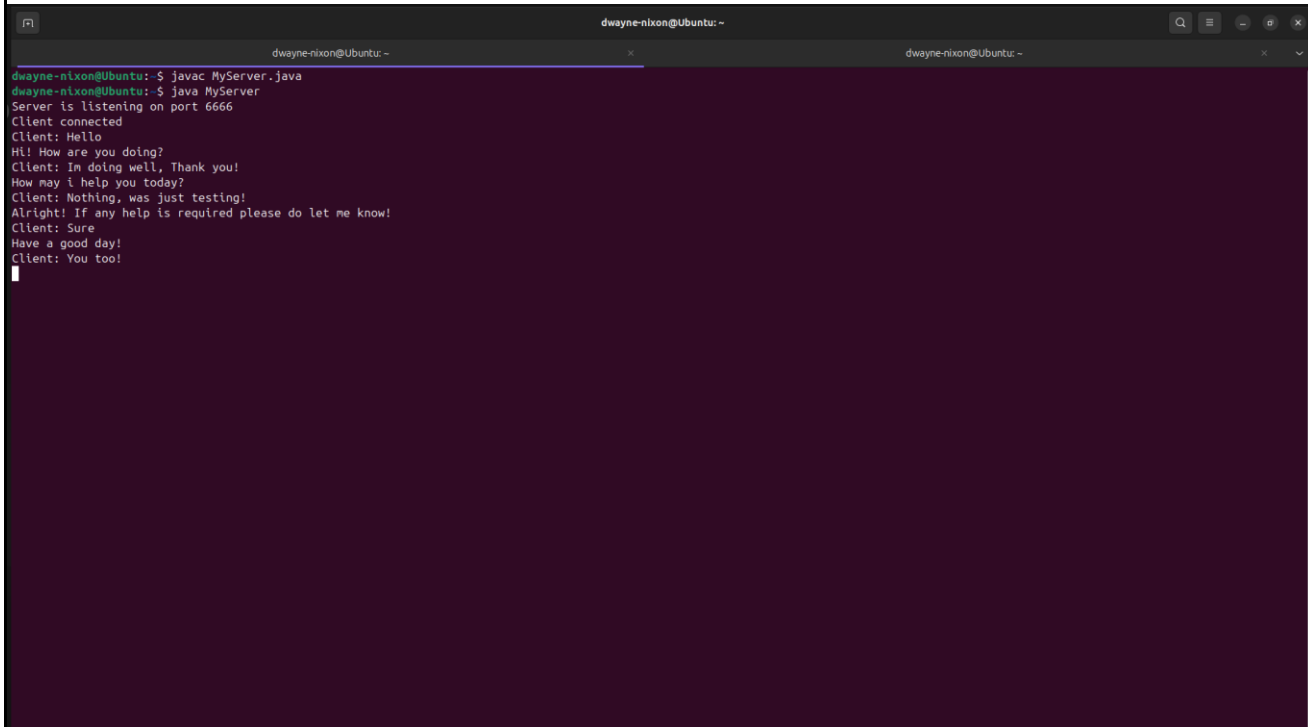
// Close resources
dis.close();
dos.close();
socket.close();
System.out.println("Chat ended.");
} catch (IOException | InterruptedException e) {
    e.printStackTrace();
}
}
```

Don Bosco Institute of Technology, Kurla

Academic Year 2023-24

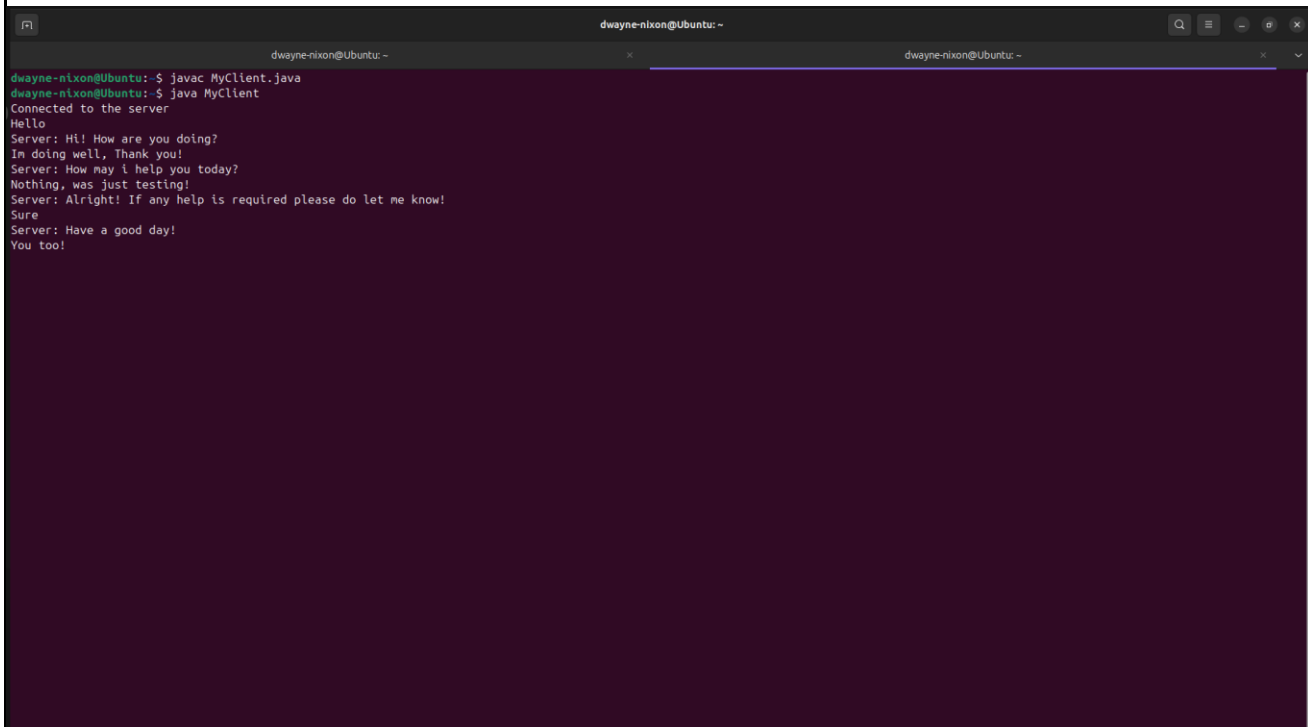
Screenshots:

Server:



```
dwayne-nixon@Ubuntu: ~  
$ javac MyServer.java  
dwayne-nixon@Ubuntu: ~$ java MyServer  
Server is listening on port 6666  
Client connected  
Client: Hello  
Hi! How are you doing?  
Client: Im doing well, Thank you!  
How may i help you today?  
Client: Nothing, was just testing!  
Alright! If any help is required please do let me know!  
Client: Sure  
Have a good day!  
Client: You too!
```

Client:



```
dwayne-nixon@Ubuntu: ~$ javac MyClient.java  
dwayne-nixon@Ubuntu: ~$ java MyClient  
Connected to the server  
Hello  
Server: Hi! How are you doing?  
Im doing well, Thank you!  
Server: How may i help you today?  
Nothing, was just testing!  
Server: Alright! If any help is required please do let me know!  
Sure  
Server: Have a good day!  
You too!
```

Don Bosco Institute of Technology, Kurla
Academic Year 2023-24

REFERENCES	<ul style="list-style-type: none">● B.A. Forouzan, “Data Communications and Networking”, TMH, Fourth Edition.● https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm● https://www.geeksforgeeks.org/socket-programming-in-java/● https://www.youtube.com/watch?v=UaM1JmQliTs
-------------------	---