Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Fall, Year:2024), B.Sc. in CSE (Day)

Lab Report NO # 02
Course Title: Computer Networking
Course Code: CSE 312 Section: 222_D11

Lab Experiment Name: **Client-Server Socket .**

Student Details

	Name	ID
1.	Sadik Saroar	212002136

Lab Date : 3 Oct 2024
Submission Date : 21 Oct 2024
Course Teacher's Name : **Md. Zahidul Hasan**
Lecturer
Green University of Bangladesh

Lab Report Status
Marks: ………………………………… **Signature:.....................**
Comments:... **Date:..............................**

1. TITLE OF THE LAB REPORT EXPERIMENT

Client-Server Socket Programming.

2. OBJECTIVES/AIM [2 marks]

The primary objective of this lab is to understand the fundamental concepts of socket programming in Java and establish communication between a client and a server. This includes:

- Creating a client that can connect to a server.
- Establishing a handshake between client and server.
- Sending and receiving messages between the client and server.

3. PROCEDURE / ANALYSIS / DESIGN [3 marks]

Server Setup:

- Initialize a `ServerSocket` on a specific port (e.g., 5000).
- Wait for a client to connect using `accept()` .
- Once connected, set up input and output streams for communication with the client.

Client Setup:

- Initialize a `Socket` to connect to the server on the specified port.
- Set up input and output streams to send and receive messages from the server.

Communication Loop:

- For both client and server, implement a loop to continuously read and write messages until a termination condition (e.g., receiving the message "Stop").

4. IMPLEMENTATION [3 marks]

Server:

```
package labscoket;
```

```
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

/**
 *
 * @author Fahad
 */
public class server {
    public static void main(String[] args)throws IOException {
        ServerSocket ss = new ServerSocket (5000);
        System.out.println("Server connection :"+ss.getLocalPort());
        System.out.println("server runing");
        System.out.println("server wait for client");

        Socket s =ss.accept();
        System.out.println("client conncetion :"+s.getPort());
        System.out.println("Client communcation :"+s.getLocalPort());

       DataInputStream input = new DataInputStream(s.getInputStream());
       DataOutputStream output = new DataOutputStream(s.getOutputStream());
       BufferedReader read = new BufferedReader(new InputStreamReader(System.in));

       String str = "";
       String serverMsg="";
       while(!str.equals("Stop")){
       str = input.readUTF();
           System.out.println("Client Say"+str);


           serverMsg= read.readLine();
           output.writeUTF(serverMsg);
           output.flush();
       }
```

```
        s.close();
        output.close();
        input.close();


    }

}
```

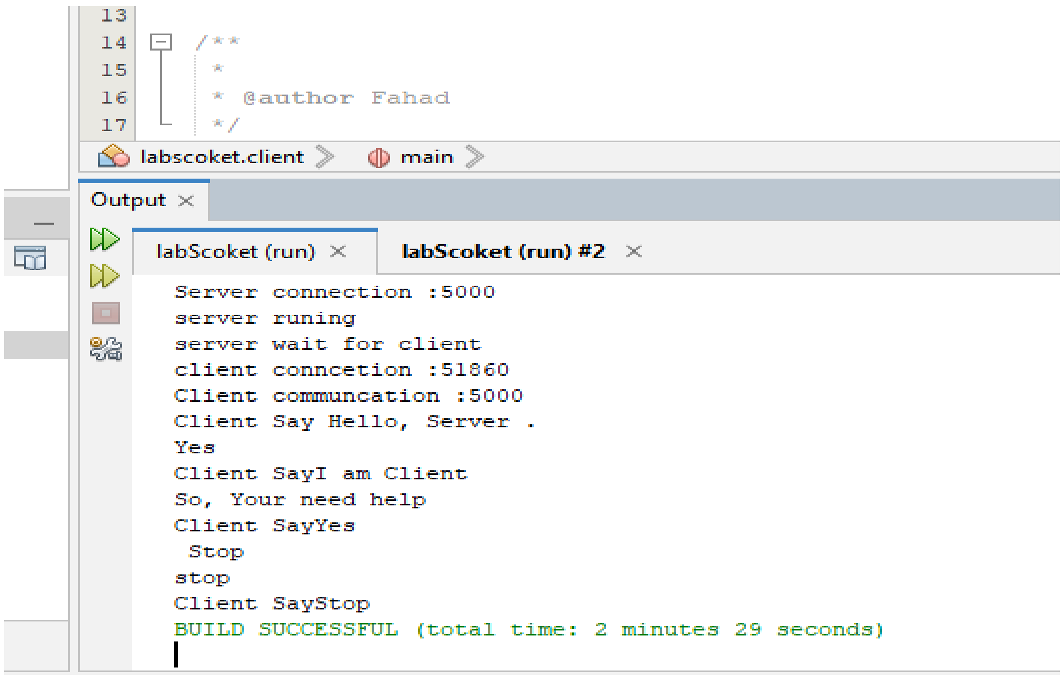
Client:

```
package labscoket;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

/**
 *
 * @author Fahad
 */
public class server {
    public static void main(String[] args)throws IOException {
        ServerSocket ss = new ServerSocket (5000);
        System.out.println("Server connection :"+ss.getLocalPort());
        System.out.println("server runing");
        System.out.println("server wait for client");

        Socket s =ss.accept();
        System.out.println("client conncetion :"+s.getPort());
        System.out.println("Client communcation :"+s.getLocalPort());

       DataInputStream input = new DataInputStream(s.getInputStream());
       DataOutputStream output = new DataOutputStream(s.getOutputStream());
       BufferedReader read = new BufferedReader(new InputStreamReader(System.in));

       String str = "";
```

```
        String serverMsg="";
        while(!str.equals("Stop")){
        str = input.readUTF();
           System.out.println("Client Say"+str);


           serverMsg= read.readLine();
           output.writeUTF(serverMsg);
           output.flush();
        }

         s.close();
         output.close();
         input.close();


    }

}
```

5. TEST RESULT / OUTPUT [3 marks]

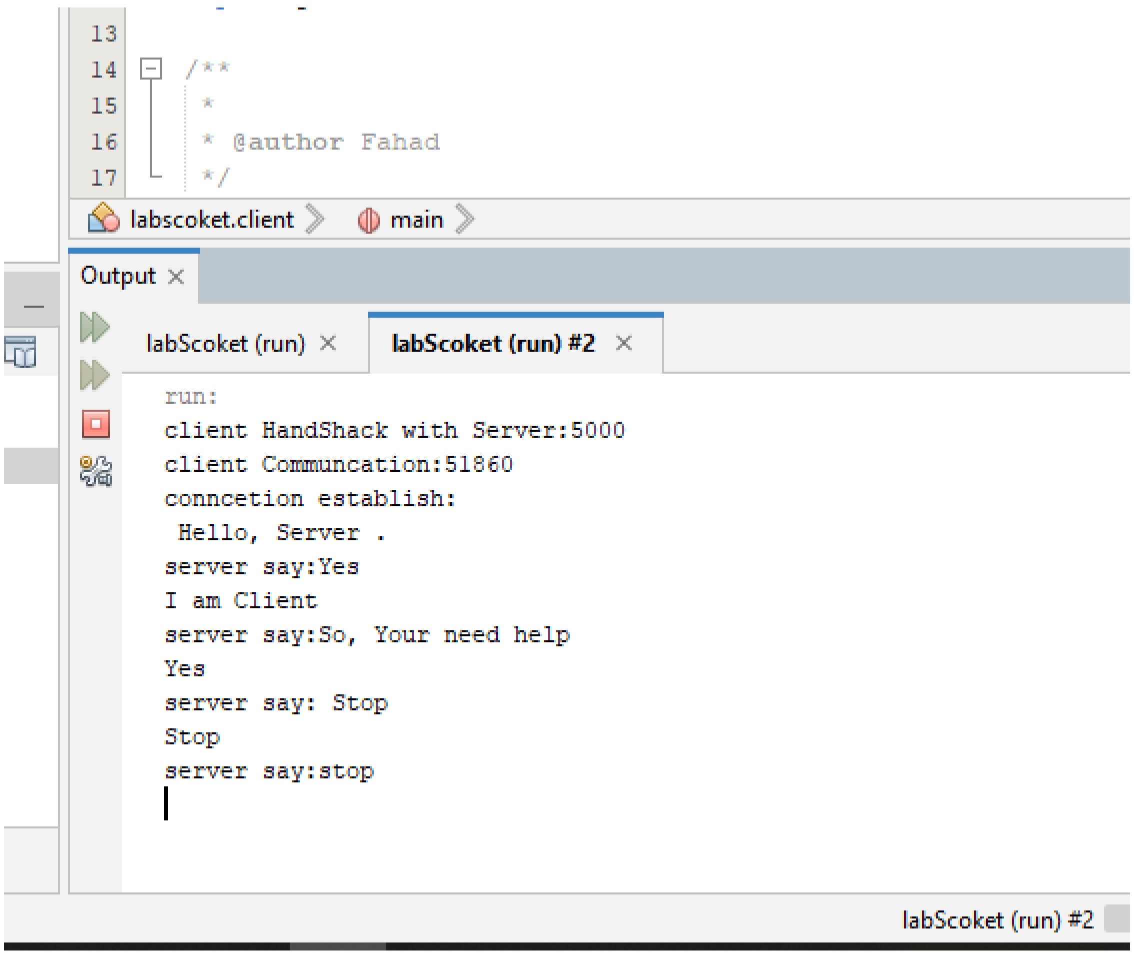
Server:



```
/**
 *
 * @author Fahad
 */
```

```
Server connection :5000
server runing
server wait for client
client conncetion :51860
Client communcation :5000
Client Say Hello, Server .
Yes
Client SayI am Client
So, Your need help
Client SayYes
 Stop
stop
Client SayStop
BUILD SUCCESSFUL (total time: 2 minutes 29 seconds)
```

Client:



```
/**
 *
 * @author Fahad
 */
```

```
run:
client HandShack with Server:5000
client Communcation:51860
conncetion establish:
 Hello, Server .
server say:Yes
I am Client
server say:So, Your need help
Yes
server say: Stop
Stop
server say:stop
```

6. ANALYSIS AND DISCUSSION [3 marks]

Connection Establishment:

- The server starts and listens on a specific port (5000). The client initiates a connection to this port.
- On successful connection, both client and server print messages indicating the ports used for communication.

Communication:

- Data is exchanged between the client and server using DataInputStream and DataOutputStream.
- Both client and server continuously send and receive messages in a loop until the "Stop" message is received.

Synchronization:

- Proper synchronization is maintained to ensure messages are sent and received correctly without data loss.
- BufferedReader is used to read input from the console, enabling interactive communication.

Error Handling:

- Basic error handling is implemented using try-catch blocks to manage potential exceptions like IO errors.

7. SUMMARY:

In this lab, we successfully created a client-server application using Java sockets. We learned how to establish a connection, exchange messages, and terminate the connection gracefully. This exercise provides a foundational understanding of socket programming, which is essential for developing network-based applications. The main takeaways include the importance of proper synchronization and error handling in ensuring reliable communication between client and server.

[**EXTRA 1 mark** for skill and attitude on this Lab Report by the student]