



Module	Assessment Type
Distributed and Cloud Systems Programming	Individual Report

## Workshop 8

Student Id : 2049867  
Student Name : Roshan Parajuli  
Section : L5CG3  
Module Leader : Rupak Koirala  
Lecturer /Tutor : Saroj Sharma  
Submitted on : 2021-05-14

Table of Contents

Introduction .....1

Workshop.....1

Task 1 .....1

Task 2.....3

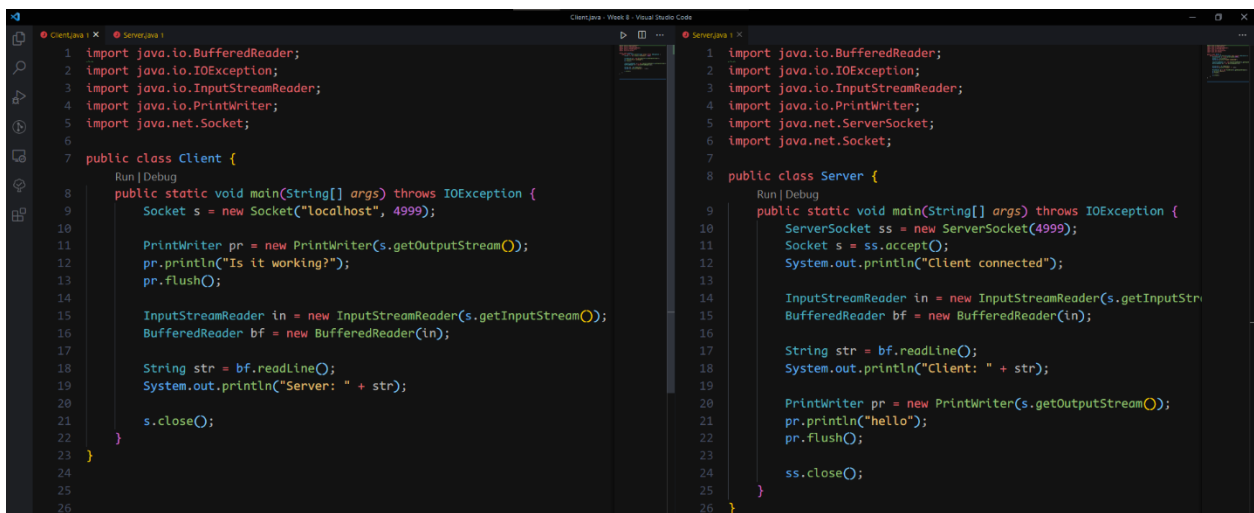
Conclusion .....6

## Introduction

Socket programming is the communication of two systems with one another and can either be connection-oriented or connectionless. For communication, the client in socket programming must know two information about the server i.e., IP Address and the port number. Once the java server is run and waiting for commands, client can then interact with it in any way that the server allows. The basic socket program is done in the workshop.

## Workshop

### Task 1:



For the communication, two files are set up. One is the server.java who acts as a server listening to commands from the client. In the main function of the server, ServerSocket is initialized. ServerSocket is a part of java.net package. The constructor of the server socket takes a port number as an argument to channel the communication through the port.

```
//establishing a connection with client
ServerSocket ss = new ServerSocket(4999);
```

In this case, the port number specified is 4999. It could have been any number as there are 65535 ports for communication between devices. Among them, the most popular ports ranges from 0-1023.

Meanwhile, In the client.java, a new socket is initialized which specifies localhost as the ip address (127.0.0.1) in the same port number as the server is listening to the commands in.

```
//establishing a connection with server
Socket s = new Socket("localhost", 4999);
```

The accept method in the server socket class will now accept the connection from the client.

```
Socket s = ss.accept();
System.out.println("Client connected");
```

After the connection is successfully established between the client and the server, the message "Client connected" is printed on the terminal.

```
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>javac Server.java && java Server
Client connected

C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>
```

```
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>javac Client.java && java Client

C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>
```

First of all, one way communication is set up. In the client part, PrintWriter class is initialized. It prints formatted representations of objects to a text-output stream. A message is attached to the reference variable of the class to send to the client and the output stream is flushed.

```
PrintWriter pr = new PrintWriter(s.getOutputStream());
pr.println("Is it working?");
pr.flush();
```

In the server, InputStreamReader is initialized which reads the output stream from the client and is passed to the buffered reader and then printed to the screen with the readLine() method. InputStreamReader allows to associate a stream that reads from the specified input. BufferedReader is an "abstraction" to help work with streams. Instead of reading character by character it returns the whole line until it finds a '\n'.

```
InputStreamReader in = new InputStreamReader(s.getInputStream());
BufferedReader bf = new BufferedReader(in);
String str = bf.readLine();
System.out.println("Client: " + str);
```

```
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>javac Server.java && java Server
Client connected
Client: Is it working?

C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>
```

```
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>javac Client.java && java Client

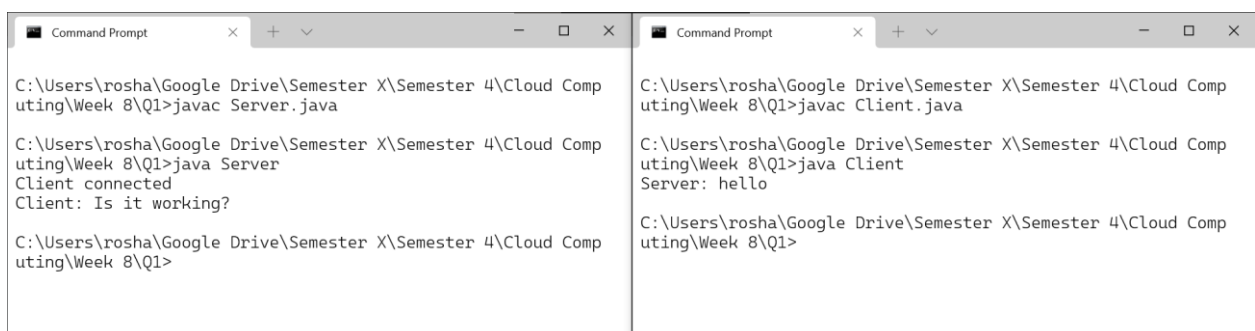
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>
```

Now that one way communication is succeeded, a two-way communication is practiced. In the same way that the client implemented the listening mechanism, server can implement the same. In the server, a message is sent with the outputstream.

```
PrintWriter pr = new PrintWriter(s.getOutputStream());  
pr.println("hello");  
pr.flush();  
  
ss.close();
```

And in the client, buffered reader on the input stream reader is reading the message and printing to the screen.

```
InputStreamReader in = new InputStreamReader(s.getInputStream());  
BufferedReader bf = new BufferedReader(in);  
  
String str = bf.readLine();  
System.out.println("Server: " + str);  
  
s.close();
```



```
Command Prompt
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>javac Server.java
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>java Server
Client connected
Client: Is it working?
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>

Command Prompt
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>javac Client.java
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>java Client
Server: hello
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1>
```

Connection is successfully established.

## Task 2:

In the same way that the task 1 is done, two files are set up. One is the client to ask for a number to send it to the server and another is the client to calculate the factorial and send it to the client.

In FactorialClient.java inside the main method,

```
System.out.println("Client Started");

Socket s = new Socket("localhost", 5123);

BufferedReader clientInput = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter a number. ");
int number = Integer.parseInt(clientInput.readLine());
PrintWriter out = new PrintWriter(s.getOutputStream(), true);

out.println(number);
```

a socket is initialized on a port number of 5123. A keypress was awaited by the BufferedReader and the number which was read was typecasted to an integer through the parseInt method of integer wrapper class. The printwriter class was then initialized and sent the number to the server. The “true” argument in the PrintWriter constructor is for the auto flush functionality. If the argument was not set to be true, out.flush() would have been provided for flushing the stream.

In FactorialServer.java inside the main method,

```
ServerSocket ss = new ServerSocket(5123);
Socket s = ss.accept();
System.out.println("Connection established");

BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream()));

int number = Integer.parseInt(in.readLine());
PrintWriter out = new PrintWriter(s.getOutputStream(), true);
out.println(" Server: The factorial of " + number + " is " + factorial(number));
```

ServerSocket is initialized on the same port number and the connection is accepted. The output stream from the client is fetched into the server and parsed to an integer.

```
public static int factorial(int num) {  
  
    int fac = 1;  
    for (int i = 1; i <= num; i++) {  
        fac = fac * i;  
    }  
    return fac;  
}
```

A function named factorial is created which takes a number as an argument and returns the factorial after calculation in the server.

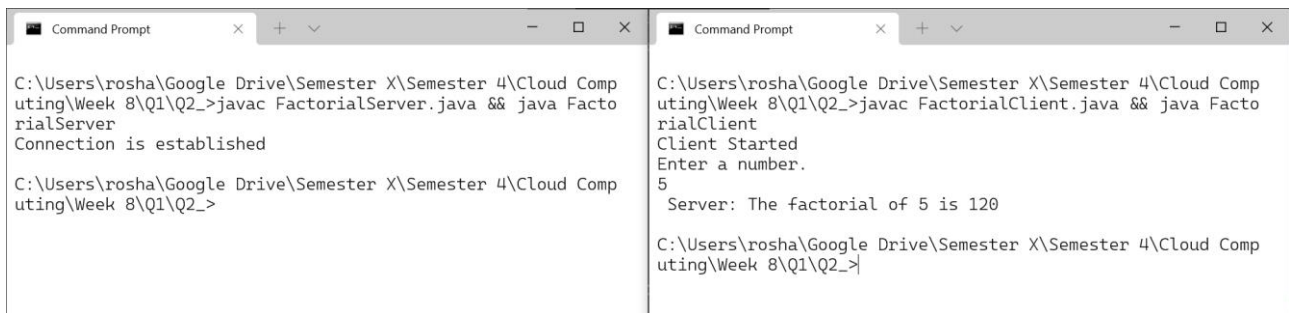
Similar with the client, PrintWriter class is initialized setting auto flush to be true and the result is passed to the client as a long string with a message attached with the factorial result.

```
PrintWriter out = new PrintWriter(s.getOutputStream(), true);  
out.println(" Server: The factorial of " + number + " is " + factorial(number));
```

Again, In the client.java,

```
BufferedReader in =  
    new BufferedReader(new InputStreamReader(s.getInputStream()));  
System.out.println(in.readLine());
```

BufferedReader uses the InputStreamReader to get the input stream from the server and prints the result to the console.



```
C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1\Q2->javac FactorialServer.java && java FactorialServer
Connection is established

C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1\Q2->

C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1\Q2->javac FactorialClient.java && java FactorialClient
Client Started
Enter a number.
5
Server: The factorial of 5 is 120

C:\Users\rosha\Google Drive\Semester X\Semester 4\Cloud Computing\Week 8\Q1\Q2->
```

## Conclusion

The workshop dealt with the basic socket programming tasks like, creating a socket, binding it to a specific IP address (localhost in this case) and port as well as sending and receiving HTTP packets. A simple two way communication program was made and then another one with the server calculating the factorial (doing the heavy duty) and returning the result to the client.