Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year:2022), BSc. in CSE (Day)

LAB REPORT - 03

Course Title: Computer Networking Lab

Course Code: CSE-312 **Section:** PC-201 DB

Student Details

Name		Students Id
1.	Md. Romzan Alom	201902144

Lab Date: 10-12-2022

Submission Date: 29-12-2022

Course Teacher's Name: Rusmita Halim Chaity

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: ………………………………. **Signature:** …………………….

Comments: ………………………….. **Date:** …………………………..

1. TITLE OF THE LAB EXPERIMENT

Implementation simple arithmetic operations using multi-thread socket programming in Java.

2. OBJECTIVES/AIM

- To gather basic knowledge of socket programming using threading.
- To learn about step-by-step of multi-threading.
- To transfer any messages or data from client to server.
- To learn how to connect Client and Server.
- To learn how to handle multiple client requests using threaded socket programming.
- To learn how to solve simple arithmetic operation using threaded socket programming.

3. PROCEDURE / ANALYSIS / DESIGN

From this experiment we will create Server, Client and Client-Handler where we can create connection between client and server. Then client chooses operation and send to server. Next server takes the request and gives response where server say, “please enter your two number” then client gives two number and sends it server. Finally, server takes those number and operate them and returns the result to client. If client say “Exit” then server will end the connection.
The working System of Multi-Tread,

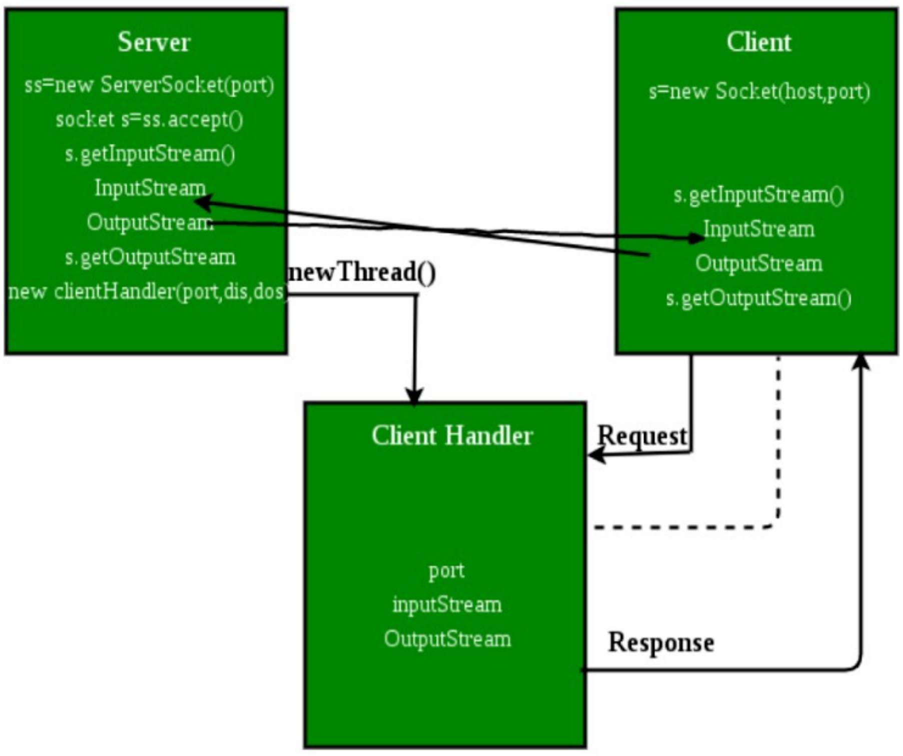


Figure_01: Flow Chart of Multi-Thread

4. IMPLEMENTATION

ServerThread:

```
package multithreading;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
public class ServerThread {
    public static void main(String[] args) throws IOException {
        ServerSocket Connection= new ServerSocket(2222);
        System.out.println("\nServer conncet at port: " + Connection.getLocalPort());
        System.out.println("Server is connecting......");
        System.out.println("Wait for the client");
        while(true){
            Socket take= Connection.accept();
            System.out.println("A Client is connected: " + take);
            DataOutputStream output = new DataOutputStream(take.getOutputStream());
            DataInputStream input = new DataInputStream(take.getInputStream());
            System.out.println("A new Thread assign");
            Thread thread =new ClientHandler(take,input,output);
            thread.start();
}} }
```

ClientHandler:

```
package multithreading;
import java.io.*;
import java.net.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;
class ClientHandler extends Thread{
Socket com_tunnel;
DataInputStream dis_tunnel;
DataOutputStream dos_tunnel;
String received = "";
String toreturn = "";
```

```
public ClientHandler(Socket s, DataInputStream dis, DataOutputStream dos)
{
this.com_tunnel = s;
this.dis_tunnel = dis;
this.dos_tunnel = dos;
}
public void run(){
while(true){
try {
dos_tunnel.writeUTF("What you want:\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Division\
n5.Modulus\n6.Exit\n");
received = dis_tunnel.readUTF();
switch(received){
   case "1":
   {
       toreturn = Operations.add();
       dos_tunnel.writeUTF(toreturn);
       break;
   }
   case "2":
   {
       toreturn = Operations.sub();
       dos_tunnel.writeUTF(toreturn);
       break;
   }
   case "3":
   {
       toreturn = Operations.mul();
       dos_tunnel.writeUTF(toreturn);
       break;
   }
   case "4":
   {
       toreturn = Operations.div();
       dos_tunnel.writeUTF(toreturn);
       break;
   }
   case "5":
   {
       toreturn = Operations.mod();
       dos_tunnel.writeUTF(toreturn);
       break;
   }
   case "6":
   {
       System.out.println("Client " + this.com_tunnel + " sendsexits");
       System.out.println("Closing the connection");
       this.com_tunnel.close();
       break;
   }
  default:
   {
       System.out.println("Please, enter valid number.");
       break;
   }}}
catch (IOException ex) {
Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,null, ex);
}}
try {
this.dos_tunnel.close();
this.dis_tunnel.close();
} catch (IOException ex) {
Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE,null, ex);
}}}
```

ClientThread:

```
package multithreading;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;
public class ClientThread {
    public static void main(String args[]) throws IOException{
try{
Socket clientsocket = new Socket ("localhost", 2222);
System.out.println("Connected at server Handshaking port " + clientsocket.getPort());
System.out.println("Client is connecting at Communication Port " + clientsocket.getLocalPort());
System.out.println("Client is Connected");
Scanner scn = new Scanner(System.in);
DataOutputStream dos = new DataOutputStream(clientsocket.getOutputStream());
DataInputStream dis = new DataInputStream(clientsocket.getInputStream());
while(true){
String inLine = dis.readUTF();
System.out.println(inLine);
String outLine = scn.nextLine();
dos.writeUTF(outLine);
switch(outLine){
    case "1" :
    {  int c;
        System.out.println("Enter First number: ");
        int a= scn.nextInt();
        System.out.println("Enter Second number: ");
        int b= scn.nextInt();
        Operations.add(a, b);
        break; }
     case "2" :
    {  int c;
        System.out.println("Enter First number: ");
        int a= scn.nextInt();
        System.out.println("Enter Second number: ");
        int b= scn.nextInt();
        Operations.sub(a, b);
        break;}
     case "3" :
    {  int c;
        System.out.println("Enter First number: ");
        int a= scn.nextInt();
        System.out.println("Enter Second number: ");
        int b= scn.nextInt();
        Operations.mul(a, b);
        break;}
     case "4" :
    { int c;
        System.out.println("Enter First number: ");
        int a= scn.nextInt();
        System.out.println("Enter Second number: ");
        int b= scn.nextInt();
        Operations.div(a, b);
        break;}
      case "5" :
     {  int c;
        System.out.println("Enter First number: ");
        int a= scn.nextInt();
        System.out.println("Enter Second number: ");
        int b= scn.nextInt();
        Operations.mod(a, b);
        break;}
        case "6":
     { System.out.println("Closing the connecting "+ clientsocket);
        clientsocket.close();
        System.out.println("Connection Closed");
        break;}
    default:
     {System.out.println("Please, enter valid number.");
        break;
}}
String received = dis.readUTF();
System.out.println(received); }
dos.close();
dis.close();
clientsocket.close();
}catch (IOException ex){
System.out.println(ex);
}}}
```

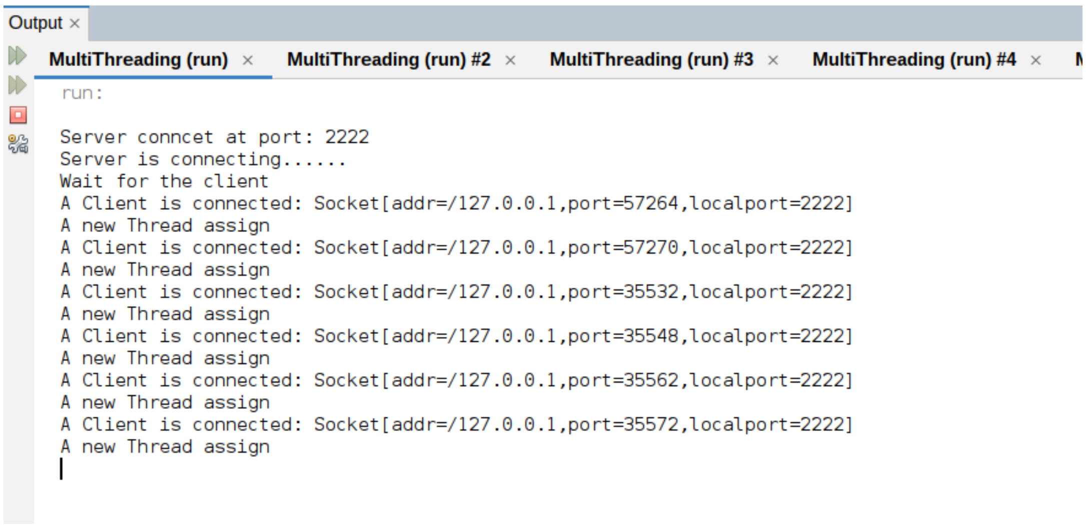
Operations:

```
package multithreading;
public class Operations {
public static String add(){
       System.out.println("");
    return null;
}
public static String add(int a,int b){
    int c;
    c= a+b;
    System.out.println("The addition is = "+c);
    return String.valueOf(c);
}
public static String sub(){
       System.out.println("");
    return null;
}
public static String sub(int a,int b){
    int c;
    c= a-b;
    System.out.println("The Subtraction is = "+c);
    return String.valueOf(c);
}
public static String mul(){
       System.out.println("");
    return null;
}
public static String mul(int a,int b){
    int c;
    c= a*b;
    System.out.println("The Multiplication is = "+c);
    return String.valueOf(c);
}
public static String div(){
       System.out.println("");
    return null;
}
public static String div(int a,int b){
    int c;
    c= a/b;
    System.out.println("The Division is = "+c);
    return String.valueOf(c);
}
public static String mod(){
       System.out.println("");
    return null;
```

```
}
public static String mod(int a,int b){
    int c;
    c= a%b;
    System.out.println("The Modulus is = "+c);
    return String.valueOf(c);
} }
```

5. TEST RESULT / OUTPUT

ServerSide:



Output ×
MultiThreading (run) ×
MultiThreading (run) #2 ×
MultiThreading (run) #3 ×
MultiThreading (run) #4 ×

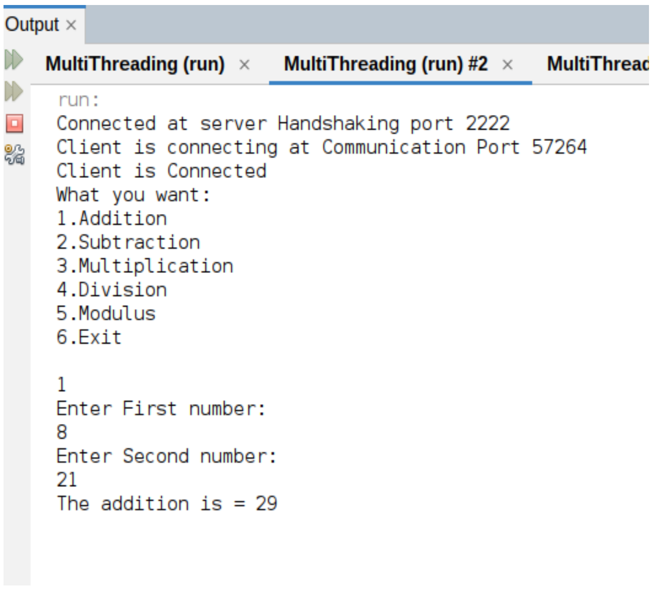
```
run:

Server conncet at port: 2222
Server is connecting......
Wait for the client
A Client is connected: Socket[addr=/127.0.0.1,port=57264,localport=2222]
A new Thread assign
A Client is connected: Socket[addr=/127.0.0.1,port=57270,localport=2222]
A new Thread assign
A Client is connected: Socket[addr=/127.0.0.1,port=35532,localport=2222]
A new Thread assign
A Client is connected: Socket[addr=/127.0.0.1,port=35548,localport=2222]
A new Thread assign
A Client is connected: Socket[addr=/127.0.0.1,port=35562,localport=2222]
A new Thread assign
A Client is connected: Socket[addr=/127.0.0.1,port=35572,localport=2222]
A new Thread assign
```

Figure_02: Output of ServerThread

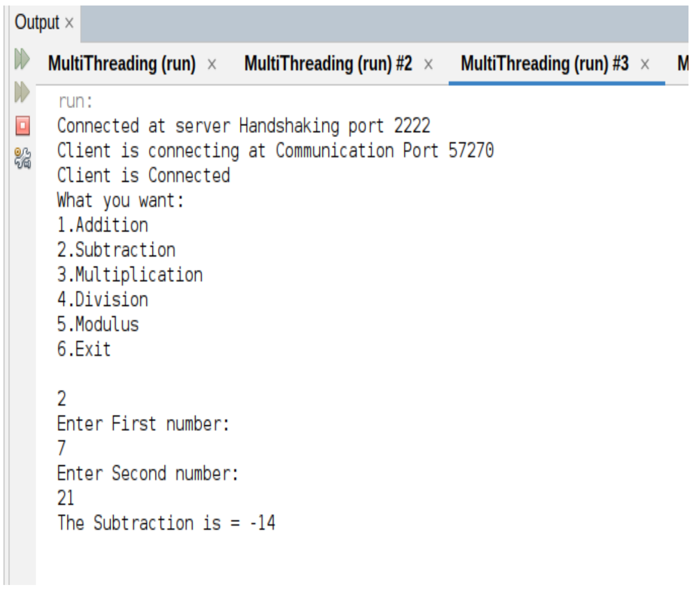
From this figure-02 the server say, Localhost is connect at port 2222 and wait for request different client.

ClientSide:



```
run:
Connected at server Handshaking port 2222
Client is connecting at Communication Port 57264
Client is Connected
What you want:
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modulus
6.Exit

1
Enter First number:
8
Enter Second number:
21
The addition is = 29
```

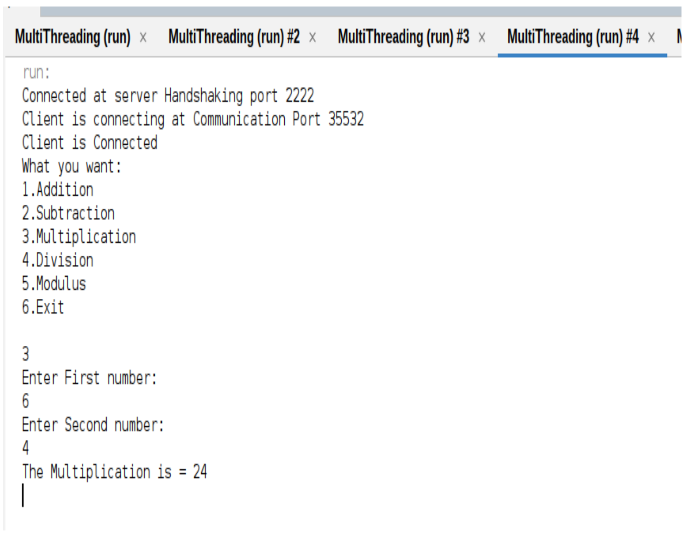


```
run:
Connected at server Handshaking port 2222
Client is connecting at Communication Port 57270
Client is Connected
What you want:
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modulus
6.Exit

2
Enter First number:
7
Enter Second number:
21
The Subtraction is = -14
```

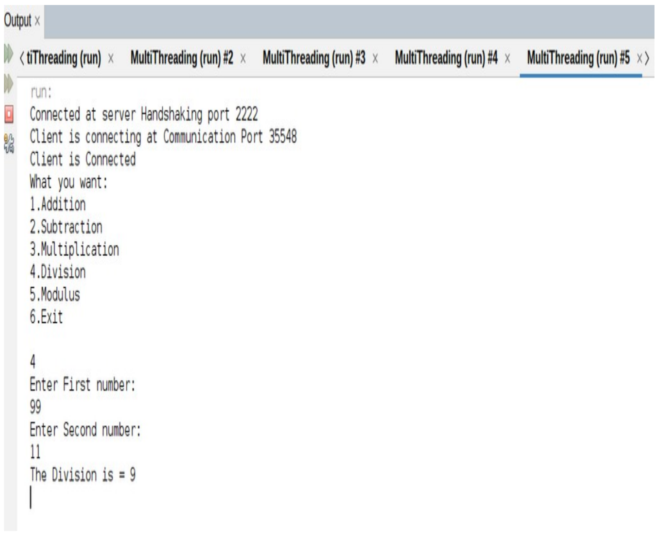
Figure_03: First Client operates addition

Figure_04: Second Client operates Subtraction



```
run:
Connected at server Handshaking port 2222
Client is connecting at Communication Port 35532
Client is Connected
What you want:
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modulus
6.Exit

3
Enter First number:
6
Enter Second number:
4
The Multiplication is = 24
```

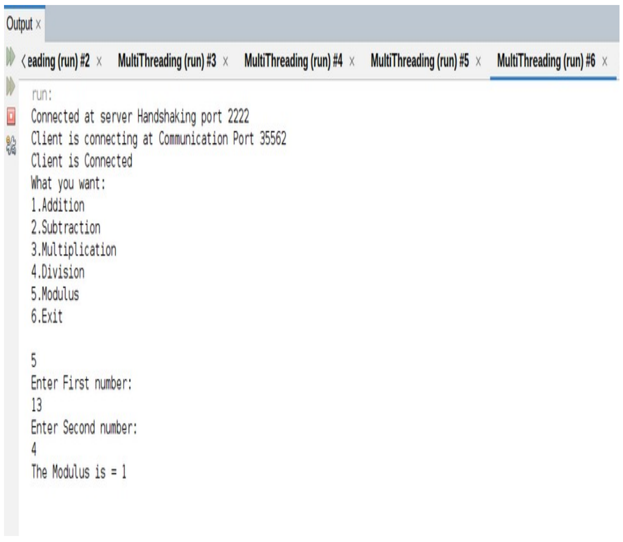


```
run:
Connected at server Handshaking port 2222
Client is connecting at Communication Port 35548
Client is Connected
What you want:
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modulus
6.Exit

4
Enter First number:
99
Enter Second number:
11
The Division is = 9
```

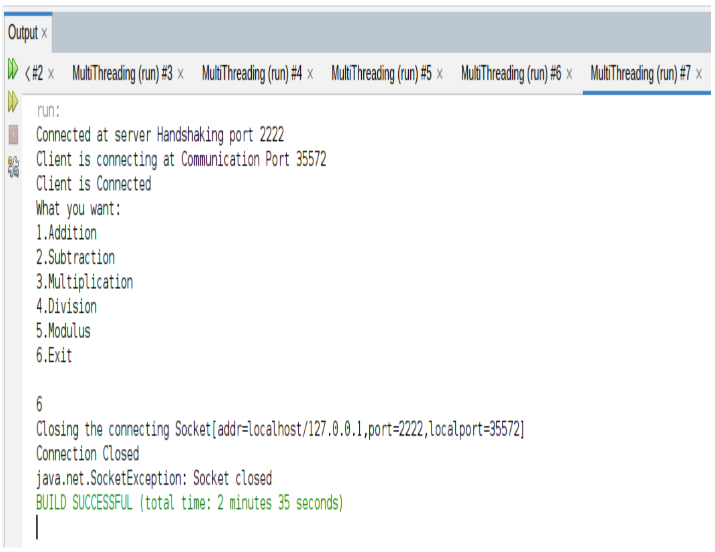
Figure_05: Third Client operates multiplication

Figure_06: Fourth Client operates division



```
run:
Connected at server Handshaking port 2222
Client is connecting at Communication Port 35562
Client is Connected
What you want:
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modulus
6.Exit

5
Enter First number:
13
Enter Second number:
4
The Modulus is = 1
```



```
run:
Connected at server Handshaking port 2222
Client is connecting at Communication Port 35572
Client is Connected
What you want:
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modulus
6.Exit

6
Closing the connecting Socket[addr=localhost/127.0.0.1,port=2222,localport=35572]
Connection Closed
java.net.SocketException: Socket closed
BUILD SUCCESSFUL (total time: 2 minutes 35 seconds)
```

Figure_07: 5th Client operates modulus

Figure_08: 6th Client operates exit

From those figure every client have six choices where 1 is Addition, 2 is Subtraction, 3 is Multiplication, 4 is Division, 5 is Modulus and 6 is Exit. If client choose 1 to 5 then client put two variables to operate. The figure-3 show addition operation. The figure-4 show subtraction operation. The figure-5 show multiplication operation. The figure-6 show division operation. The figure-7 show modulus operation. The figure-8 show exit operation.

6. ANALYSIS AND DISCUSSION

This experiment mainly based on Java. Based on the focused objective(s) to understand about multi-threaded socket programming. This task will help us to learn about the basic structure of multi threaded socket programming. From this experiment, we will create connection between client and server. Client gives data as input and server operate those data and gives response as output. Here, we use multiple client that's why we use client-handler to handle multiple client. We use simple arithmetic operation in compiler that's why it may have some compiler error. The main hard part of this experiment is successfully handle multiple client at the same time. We face so many problem for handle that part. This experiment will show real browsing system.

7. SUMMARY:

In this experiment we will create Server, Client and Client-Handler where we can create connection between client and server and server handle client using client-handler. Client gives data as input and server operate those data and gives response as output. In server side the client-handler handle client and operate basic operation (Addition, Subtraction, Multiplication, Division and Modulus). In this experiment, we will feel the real environment of internet. That's why this experiment is very interesting and helpful for future.