Green University of Bangladesh

Department of Computer Science and Engineering
Faculty of Sciences and Engineering
Semester: (Fall, Year:2023), B.Sc. in CSE(Day)

Lab Report :04
Course Title: Computer Networking Lab
Course Code: CSE 312
Section: 211 D2

Experiment Name:

Implementation of socket programming using threading.

Student Details

Name	ID
Md Emon Hossain	201902009

Submission Date: December 13, 2023

Teachers Name: Tanpia Tasnim

Status

Marks:	Signature: ....
Comments:	Date: ....

0.1 Introduction

Mathematical operations are very useful to be implemented using TCP socket programming. Create two processes, a server and a client using JAVA codes. The client will send two separate integer values and a mathematical operator to the server. The server receives these integers and a mathematical operator, then performs a mathematical operation based on the user input. The server then sends this answer to the client, who then displays it. The client sends requests as many times as he wishes. However, The server can serve at most 5 clients in its lifetime. The individual client ends its connection by saying "ENDS".

0.2 Objectives

1. To Implementation of socket programming using threading.
2. To learn about step by step implementation for handling multiple client requests using threaded socket programming to perform different operations on a different different operand.

0.3 Implementation

Algorithm for Server-Side Socket Connection:

Step-1:

- Create a ServerSocket object named handshakeSocket, specifying the port number.

Step-2:

- Loop to accept up to 5 client connections:
  - Accept a client connection and create a Socket object named communication Socket.
  - Create DataOutputStream (dos) and DataInputStream (dis) objects for sending and receiving data.
  - Create a MathServerThread object, passing communicationSocket, dos, and dis.
  - Start the MathServerThread.

Step-3: In the MathServerThread:

- Loop to handle up to 5 requests from the connected client:
  - Receive two integers and an operator from the client using dis.
  - Perform the specified mathematical operation.
  - Display "ENDS" on the client side.
  - Send the result back to the client using dos.
  - If the client sends "ENDS," break from the loop.

Step-4:

- Close the handshakeSocket after serving 5 clients.

Algorithm for Client-Side Socket Connection:

Step-1:

- Create a Socket object, specifying the server's IP address and port number.

Step-2:

- Create DataOutputStream (dos) and DataInputStream (dis) objects for sending and receiving data.
  - Accept a client connection and create a Socket object named communication Socket.
  - Create DataOutputStream (dos) and DataInputStream (dis) objects for sending and receiving data.
  - Create a MathServerThread object, passing communicationSocket, dos, and dis.
  - Start the MathServerThread.

Step-3:

- Loop for up to 5 times:
  - Prompt the user to enter two integers and an operator.
  - Send the inputs to the server using dos.
  - Receive and display the result from the server using dis.
  - Display "ENDS" to indicate successful completion.
  - If the user inputs "ENDS," break from the loop.

Step-4:

- Close the connections and end the client-server communication.

Code:

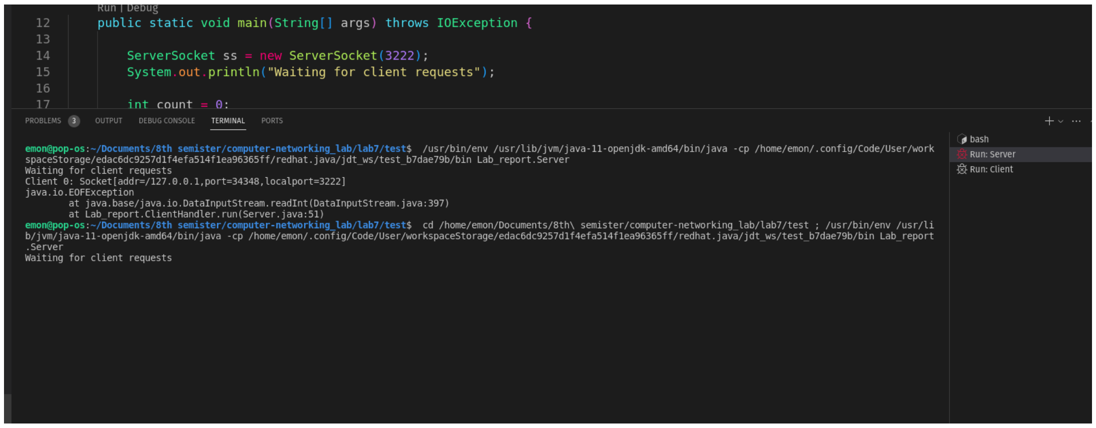
Client Code

```
package Lab_report;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;

public class Client {

    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 3222);

            DataInputStream dis = new DataInputStream(socket.getInputStream());
            DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
            Scanner scanner = new Scanner(System.in);

            int num1, num2;
            String operator;

            while (true) {
                System.out.print("Enter input 1 (or type 'ENDS' to exit): ");
                String input1 = scanner.nextLine();

                if (input1.equalsIgnoreCase("ENDS")) {
                    dos.writeUTF("ENDS");
                    break;
                }

                System.out.print("Enter␣input␣2␣");
                String input2 = scanner.nextLine();

                System.out.print("Enter␣operator␣(Sum,␣Subtract,␣Multiply,␣Divide,␣Modulus):␣");
                operator = scanner.nextLine();

                num1 = Integer.parseInt(input1);
                num2 = Integer.parseInt(input2);

                dos.writeInt(num1);
                dos.writeInt(num2);
                dos.writeUTF(operator);

                int result = dis.readInt();
                System.out.println("Result␣from␣server:␣" + result);
            }

            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Server Code

```

package Lab_report;


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

public class Server {

    public static void main(String[] args) throws IOException {

        ServerSocket ss = new ServerSocket(3222);
        System.out.println("Waiting␣for␣client␣requests");

        int count = 0;

        while (count < 5) {
            Socket s = ss.accept();
            System.out.println("Client␣" + count + ":␣" + s);

            DataInputStream dis = new DataInputStream(s.getInputStream());
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());

            Thread t = new ClientHandler(s, dis, dos);
            t.start();

            count++;
        }

        ss.close();
```

```java
    }
}

class ClientHandler extends Thread {

    final Socket soc;
    final DataInputStream input;
    final DataOutputStream output;

    public ClientHandler(Socket s, DataInputStream dis, DataOutputStream dos) {
        this.soc = s;
        this.input = dis;
        this.output = dos;
    }

    public void run() {
        try {
            while (true) {
                int num1 = input.readInt();
                int num2 = input.readInt();
                String operator = input.readUTF();

                int result = task(num1, num2, operator);

                output.writeInt(result);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private int task(int num1, int num2, String operator) {
        int result = 0;

        switch (operator.toLowerCase()) {
            case "sum":
                result = num1 + num2;
                break;
            case "subtract":
                result = num1 - num2;
                break;
            case "multiply":
                result = num1 * num2;
                break;
            case "divide":
                if (num2 != 0) {
                    result = num1 / num2;
                } else {
                    System.out.println("Cannot␣divide␣by␣zero.");
                }
                break;
            case "modulus":
                if (num2 != 0) {
                    result = num1 % num2;
                } else {
                    System.out.println("Cannot␣calculate␣modulus␣with␣zero.");
                }
                break;
            default:
                System.out.println("Invalid␣operator:␣" + operator);
        }

        return result;
        }
}
```

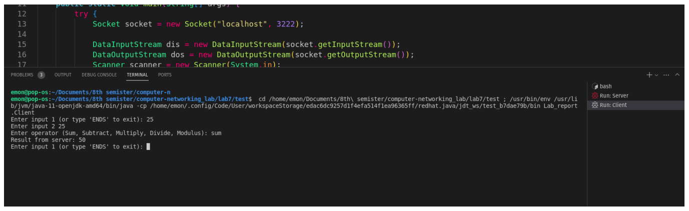


```
Run | Debug
public static void main(String[] args) throws IOException {

    ServerSocket ss = new ServerSocket(3222);
    System.out.println("Waiting for client requests");

    int count = 0;
```

```
emon@pop-os:~/Documents/8th semister/computer-networking_lab/lab7/test$  /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/emon/.config/Code/User/work
spaceStorage/edac6dc9257d1f4efa514f1ea96365ff/redhat.java/jdt_ws/test_b7dae79b/bin Lab_report.Server
Waiting for client requests
Client 0: Socket[addr=/127.0.0.1,port=34348,localport=3222]
java.io.EOFException
        at java.base/java.io.DataInputStream.readInt(DataInputStream.java:397)
        at Lab_report.ClientHandler.run(Server.java:51)
emon@pop-os:~/Documents/8th semister/computer-networking_lab/lab7/test$  cd /home/emon/Documents/8th\ semister/computer-networking_lab/lab7/test ; /usr/bin/env /usr/li
b/jvm/java-11-openjdk-amd64/bin/java -cp /home/emon/.config/Code/User/workspaceStorage/edac6dc9257d1f4efa514f1ea96365ff/redhat.java/jdt_ws/test_b7dae79b/bin Lab_report
.Server
Waiting for client requests
```

Figure 1: server side run

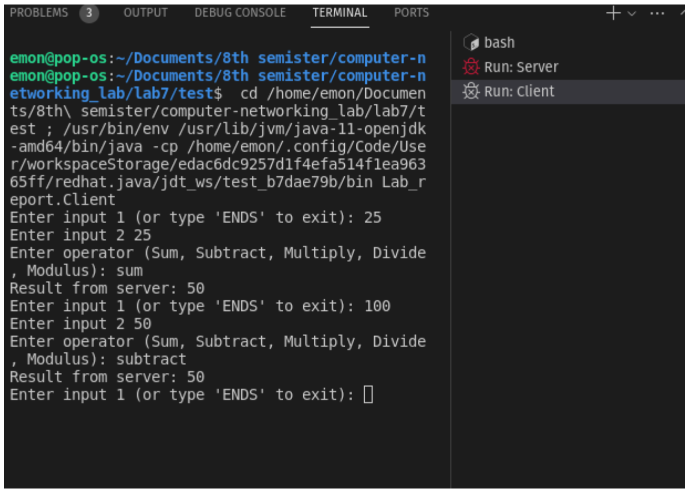
Final Result



```
try {
    Socket socket = new Socket("localhost", 3222);

    DataInputStream dis = new DataInputStream(socket.getInputStream());
    DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
    Scanner scanner = new Scanner(System.in);
```

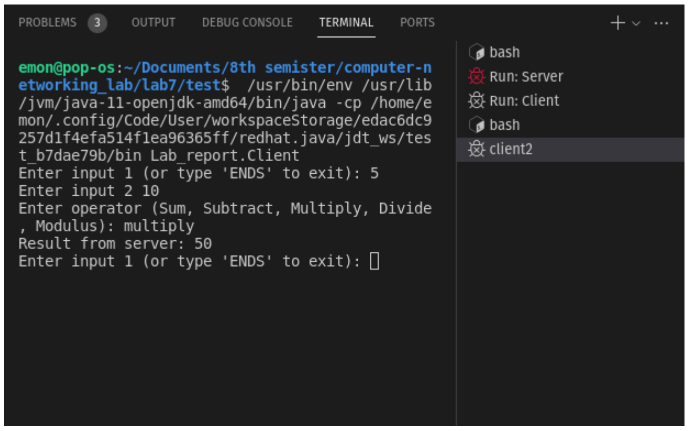
```
emon@pop-os:~/Documents/8th semister/computer-n
emon@pop-os:~/Documents/8th semister/computer-networking_lab/lab7/test$  cd /home/emon/Documents/8th\ semister/computer-networking_lab/lab7/test ; /usr/bin/env /usr/li
b/jvm/java-11-openjdk-amd64/bin/java -cp /home/emon/.config/Code/User/workspaceStorage/edac6dc9257d1f4efa514f1ea96365ff/redhat.java/jdt_ws/test_b7dae79b/bin Lab_report
.Client
Enter input 1 (or type 'ENDS' to exit): 25
Enter input 2 25
Enter operator (Sum, Subtract, Multiply, Divide, Modulus): sum
Result from server: 50
Enter input 1 (or type 'ENDS' to exit): 
```

Figure 2: Client 1 sum



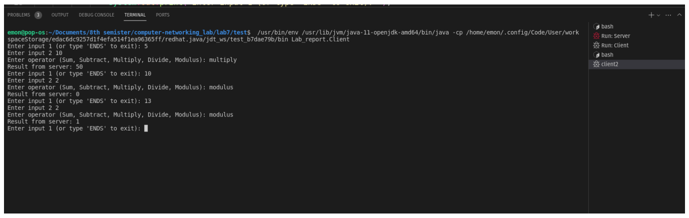
```
emon@pop-os:~/Documents/8th semister/computer-n
emon@pop-os:~/Documents/8th semister/computer-n
etworking_lab/lab7/test$  cd /home/emon/Documen
ts/8th\ semister/computer-networking_lab/lab7/t
est ; /usr/bin/env /usr/lib/jvm/java-11-openjdk
-amd64/bin/java -cp /home/emon/.config/Code/Use
r/workspaceStorage/edac6dc9257d1f4efa514f1ea963
65ff/redhat.java/jdt_ws/test_b7dae79b/bin Lab_r
eport.Client
Enter input 1 (or type 'ENDS' to exit): 25
Enter input 2 25
Enter operator (Sum, Subtract, Multiply, Divide
, Modulus): sum
Result from server: 50
Enter input 1 (or type 'ENDS' to exit): 100
Enter input 2 50
Enter operator (Sum, Subtract, Multiply, Divide
, Modulus): subtract
Result from server: 50
Enter input 1 (or type 'ENDS' to exit): 
```

Figure 3: Client 1 substract



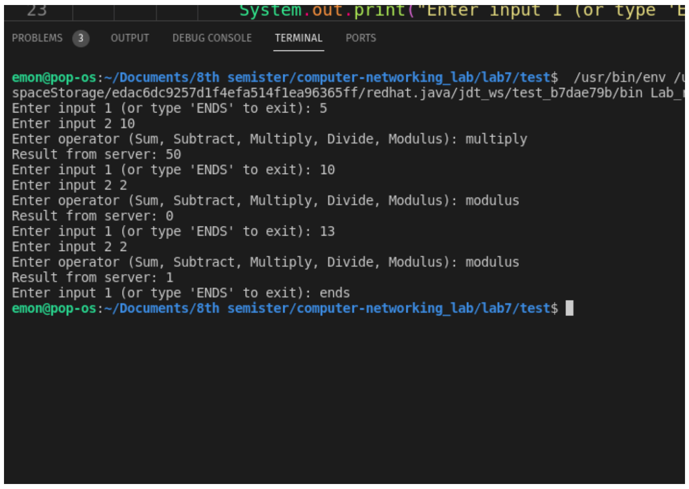
```
emon@pop-os:~/Documents/8th semister/computer-n
etworking_lab/lab7/test$  /usr/bin/env /usr/lib
/jvm/java-11-openjdk-amd64/bin/java -cp /home/e
mon/.config/Code/User/workspaceStorage/edac6dc9
257d1f4efa514f1ea96365ff/redhat.java/jdt_ws/tes
t_b7dae79b/bin Lab_report.Client
Enter input 1 (or type 'ENDS' to exit): 5
Enter input 2 10
Enter operator (Sum, Subtract, Multiply, Divide
, Modulus): multiply
Result from server: 50
Enter input 1 (or type 'ENDS' to exit): 
```

Figure 4: Client 2 Multiply



```
emon@pop-os:~/Documents/8th semister/computer-networking_lab/lab7/test$  /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/emon/.config/Code/User/work
spaceStorage/edac6dc9257d1f4efa514f1ea96365ff/redhat.java/jdt_ws/test_b7dae79b/bin Lab_report.Client
Enter input 1 (or type 'ENDS' to exit): 5
Enter input 2 10
Enter operator (Sum, Subtract, Multiply, Divide, Modulus): multiply
Result from server: 50
Enter input 1 (or type 'ENDS' to exit): 10
Enter input 2 2
Enter operator (Sum, Subtract, Multiply, Divide, Modulus): modulus
Result from server: 0
Enter input 1 (or type 'ENDS' to exit): 13
Enter input 2 2
Enter operator (Sum, Subtract, Multiply, Divide, Modulus): modulus
Result from server: 1
Enter input 1 (or type 'ENDS' to exit): 
```

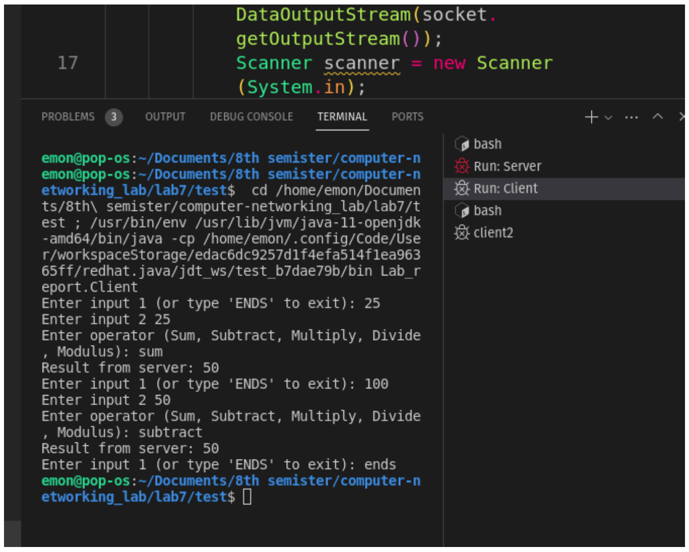
Figure 5: Client 2 Modulus



```
System.out.print("Enter input 1 (or type 'E
```

```
emon@pop-os:~/Documents/8th semister/computer-networking_lab/lab7/test$  /usr/bin/env /
spaceStorage/edac6dc9257d1f4efa514f1ea96365ff/redhat.java/jdt_ws/test_b7dae79b/bin Lab_
Enter input 1 (or type 'ENDS' to exit): 5
Enter input 2 10
Enter operator (Sum, Subtract, Multiply, Divide, Modulus): multiply
Result from server: 50
Enter input 1 (or type 'ENDS' to exit): 10
Enter input 2 2
Enter operator (Sum, Subtract, Multiply, Divide, Modulus): modulus
Result from server: 0
Enter input 1 (or type 'ENDS' to exit): 13
Enter input 2 2
Enter operator (Sum, Subtract, Multiply, Divide, Modulus): modulus
Result from server: 1
Enter input 1 (or type 'ENDS' to exit): ends
emon@pop-os:~/Documents/8th semister/computer-networking_lab/lab7/test$ 
```

Figure 6: client2 Types ends for stop



```
DataOutputStream(socket.
getOutputStream());
Scanner scanner = new Scanner
(System.in);
```

```
emon@pop-os:~/Documents/8th semister/computer-n
emon@pop-os:~/Documents/8th semister/computer-n
etworking_lab/lab7/test$  cd /home/emon/Documen
ts/8th\ semister/computer-networking_lab/lab7/t
est ; /usr/bin/env /usr/lib/jvm/java-11-openjdk
-amd64/bin/java -cp /home/emon/.config/Code/Use
r/workspaceStorage/edac6dc9257d1f4efa514f1ea963
65ff/redhat.java/jdt_ws/test_b7dae79b/bin Lab_r
eport.Client
Enter input 1 (or type 'ENDS' to exit): 25
Enter input 2 25
Enter operator (Sum, Subtract, Multiply, Divide
, Modulus): sum
Result from server: 50
Enter input 1 (or type 'ENDS' to exit): 100
Enter input 2 50
Enter operator (Sum, Subtract, Multiply, Divide
, Modulus): subtract
Result from server: 50
Enter input 1 (or type 'ENDS' to exit): ends
emon@pop-os:~/Documents/8th semister/computer-n
etworking_lab/lab7/test$ 
```

Figure 7: client 1 Types ends for stop

0.4 Conclusion

In this Lab report, implemented solution comprises a TCP server that handles multiple client connections concurrently, allowing clients to send mathematical operation requests. The server performs the requested operations, sends the result back to the client, and displays "ENDS" after each successful completion. The server serves up to 5 clients in its lifetime and then shuts down.