SOCKETS MINI-PROJECT REPORT

October 18, 2018

UFAZ University
Written by:
Afrasiyab Khalili
Orkhan Hashimli

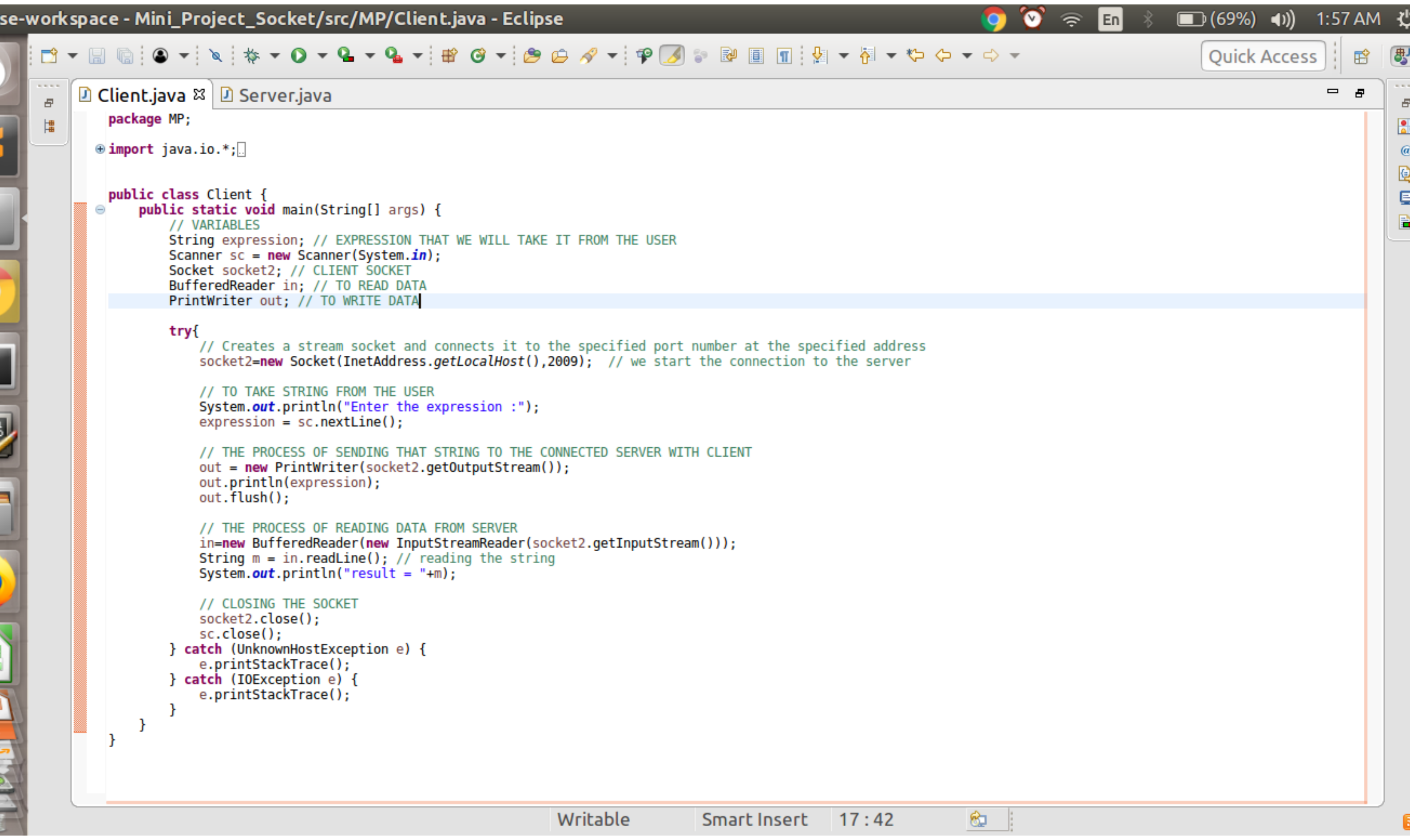
TASK:

We need to make a calculator that is working between Server and Client. There is some rules that we should to obey:
1. Shouldn't use the negative numbers.
2. Shouldn't have the first number lesser than the second one.
3. Second number shouldn't be equal to zero(zero division).
4. Should work in connected mode.

SOLUTION:

So, Our solution to that task is to make two 2 Classes:
1.Server Class
2.Client Class

1. CLIENT CLASS



```java
package MP;

import java.io.*;

public class Client {
    public static void main(String[] args) {
        // VARIABLES
        String expression; // EXPRESSION THAT WE WILL TAKE IT FROM THE USER
        Scanner sc = new Scanner(System.in);
        Socket socket2; // CLIENT SOCKET
        BufferedReader in; // TO READ DATA
        PrintWriter out; // TO WRITE DATA

        try{
            // Creates a stream socket and connects it to the specified port number at the specified address
            socket2=new Socket(InetAddress.getLocalHost(),2009);  // we start the connection to the server

            // TO TAKE STRING FROM THE USER
            System.out.println("Enter the expression :");
            expression = sc.nextLine();

            // THE PROCESS OF SENDING THAT STRING TO THE CONNECTED SERVER WITH CLIENT
            out = new PrintWriter(socket2.getOutputStream());
            out.println(expression);
            out.flush();

            // THE PROCESS OF READING DATA FROM SERVER
            in=new BufferedReader(new InputStreamReader(socket2.getInputStream()));
            String m = in.readLine(); // reading the string
            System.out.println("result = "+m);

            // CLOSING THE SOCKET
            socket2.close();
            sc.close();
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

So, In the **Client Class**, we need to take *String* from the user. Then, to send it to the **Server** side and get the result of calculation from the **Server** side.First, As you can see we have some variables:

(a) **String expression** - It is a string that will be taken from the user.

(b) **Scanner sc** - it is a Scanner type object that will help to scan the line.

(c) **Socket socket2** - it is a Socket object.

(d) **BufferedReader in** - Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

(e) **PrintWriter out** - Prints formatted representations of objects to a text-output stream.

In order to make sure that everything is OK, We'll do every step in **try and catch** blocks.We can have *UnknowHostException* - Thrown to indicate that the IP address of a host could not be determined or *IOException* - Signals that an I/O exception of some sort has occurred. This class is the general class of exceptions produced by failed or interrupted I/O operations exceptions that the program throws out.
STEP1:

(a) We are creating Socket object at Address XXXXXXX and connect it to specified port 2009.

(b) We are taking *String* from the user.

(c) And then, in order to send this *String* to the Server, we will use *out* - a new object in PrintWriter type, without automatic line flushing, from an existing OutputStream, *out.println(expression)* - prints a string, *out.flush()* - flushes the stream.

STEP2:
In that case we are getting data instead of sending it to somewhere.

(a) **in** - In order to get that data, We are creating an objcet in BufferedReader which will buffer the input from the specified input.

(b) **in.readLine()** - reads a line of text.

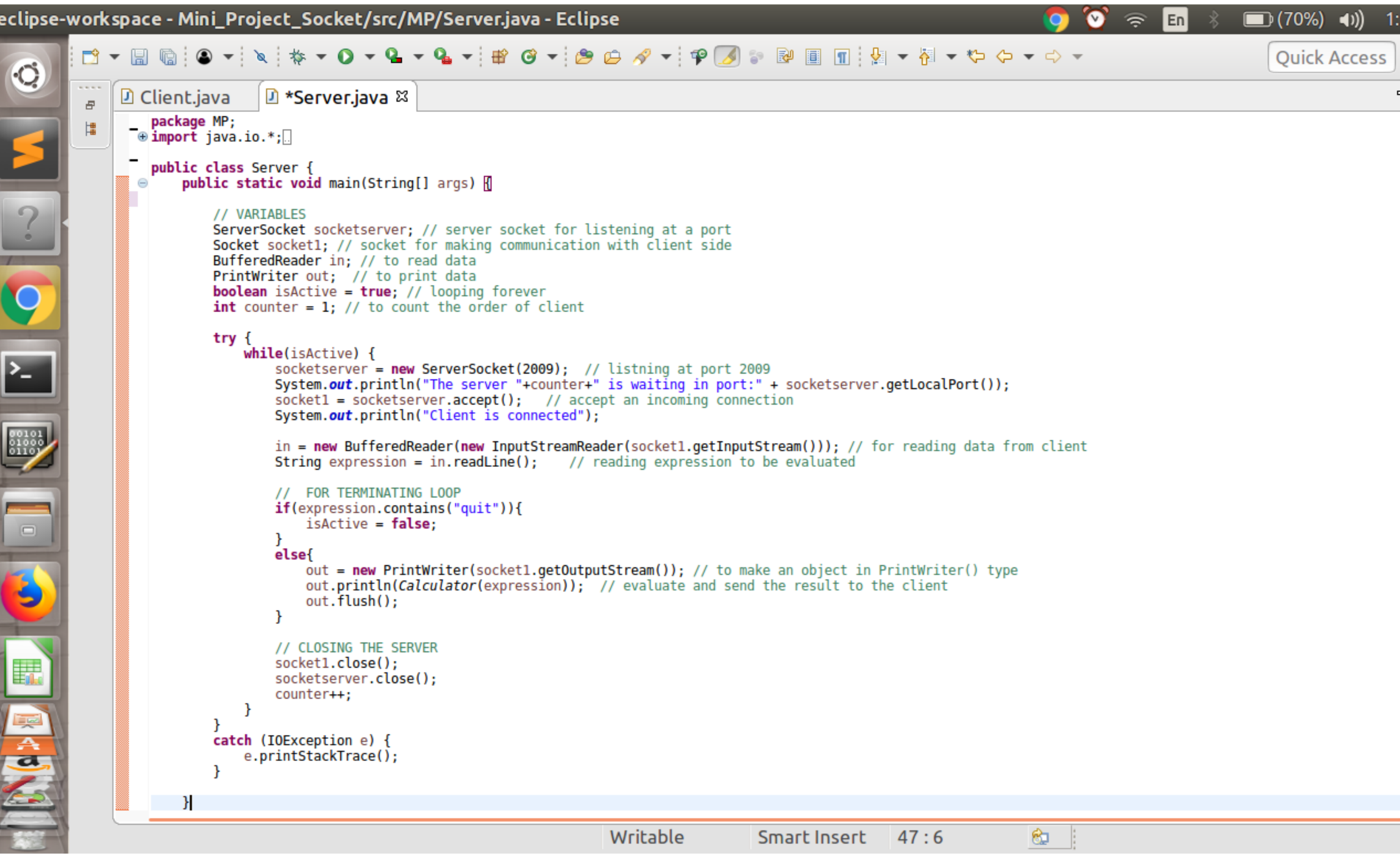
(c) Do not forget to close the servers(They are limited.).

2. **SERVER CLASS**
We have two methods in our Class.

1̂.**main()** method

2̂.**Calculator()** method

(a) The main() method:



eclipse-workspace - Mini_Project_Socket/src/MP/Server.java - Eclipse

```java
package MP;
import java.io.*;

public class Server {
    public static void main(String[] args) {

        // VARIABLES
        ServerSocket socketserver; // server socket for listening at a port
        Socket socket1; // socket for making communication with client side
        BufferedReader in; // to read data
        PrintWriter out;  // to print data
        boolean isActive = true; // looping forever
        int counter = 1; // to count the order of client

        try {
            while(isActive) {
                socketserver = new ServerSocket(2009);  // listning at port 2009
                System.out.println("The server "+counter+" is waiting in port:" + socketserver.getLocalPort());
                socket1 = socketserver.accept();   // accept an incoming connection
                System.out.println("Client is connected");

                in = new BufferedReader(new InputStreamReader(socket1.getInputStream())); // for reading data from client
                String expression = in.readLine();    // reading expression to be evaluated

                //  FOR TERMINATING LOOP
                if(expression.contains("quit")){
                    isActive = false;
                }
                else{
                    out = new PrintWriter(socket1.getOutputStream()); // to make an object in PrintWriter() type
                    out.println(Calculator(expression));  // evaluate and send the result to the client
                    out.flush();
                }

                // CLOSING THE SERVER
                socket1.close();
                socketserver.close();
                counter++;
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }

    }
```

Writable   Smart Insert   47 : 6

In order to make sure that everything is OK, We'll use every step in **try** and **catch** blocks.There can be *IOException* - Signals that an I/O exception of some sort has occurred. This class is the general class of exceptions produced by failed or interrupted I/O operations. So we have infinity loop(till *isActive* is true) we can continue making connection with **Client** until he doesn't want.

STEPS:

i. Creating *SocketServer* at port 2009.

ii. To declare *socket1* that listens for a connection to be made to this socket and accepts it.

iii. *in* - In order to get that data, We are creating an objcet in BufferedReader which will buffer the input from the specified input.

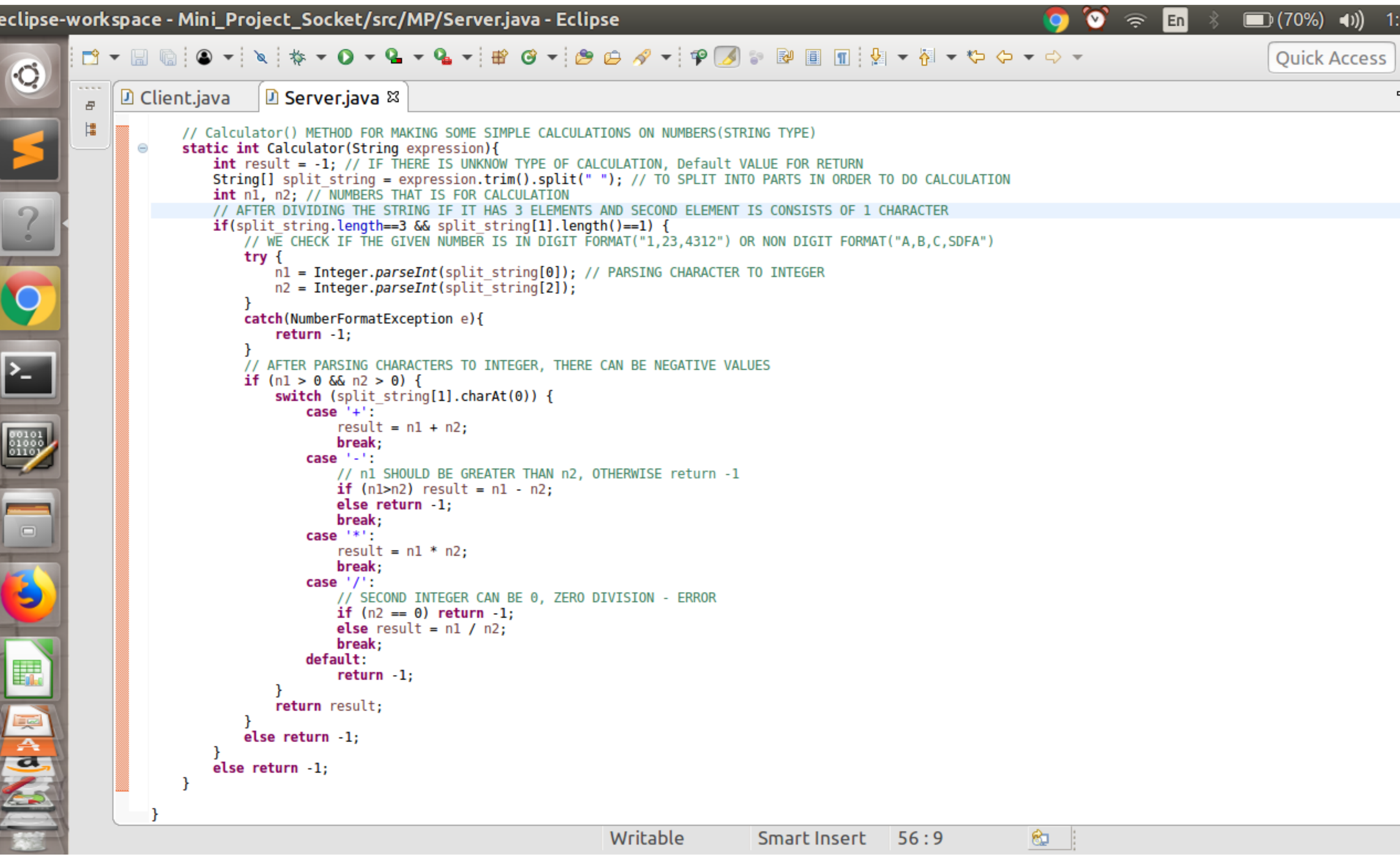
iv. *expression* - String type.

v. if the *expression* consists of *quit* the whole actions will terminate, because *isActive* will be **false**, so you couldn't enter to while() loop. Else, we will continue our connection.

out - a new object in PrintWriter type, without automatic line flushing, from an existing OutputStream, *out.println(Calculator(expression)* - prints a string, *out.flush()* - flushes the stream.

vi. Do not forget to close the servers(They are limited).

(b) The Calculator() method:


eclipse-workspace - Mini_Project_Socket/src/MP/Server.java - Eclipse
(70%)
Quick Access
Client.java
Server.java
Writable
Smart Insert
56 : 9

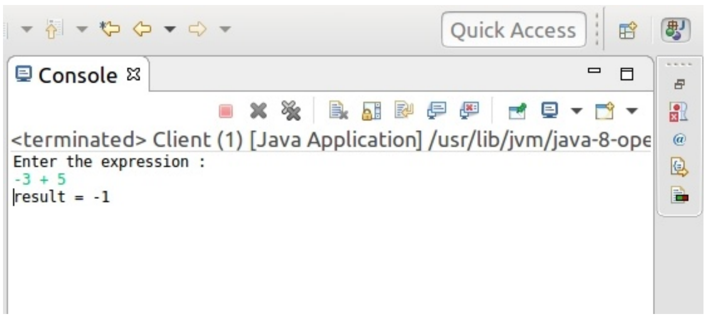
```
    // Calculator() METHOD FOR MAKING SOME SIMPLE CALCULATIONS ON NUMBERS(STRING TYPE)
    static int Calculator(String expression){
        int result = -1; // IF THERE IS UNKNOW TYPE OF CALCULATION, Default VALUE FOR RETURN
        String[] split_string = expression.trim().split(" "); // TO SPLIT INTO PARTS IN ORDER TO DO CALCULATION
        int n1, n2; // NUMBERS THAT IS FOR CALCULATION
        // AFTER DIVIDING THE STRING IF IT HAS 3 ELEMENTS AND SECOND ELEMENT IS CONSISTS OF 1 CHARACTER
        if(split_string.length==3 && split_string[1].length()==1) {
            // WE CHECK IF THE GIVEN NUMBER IS IN DIGIT FORMAT("1,23,4312") OR NON DIGIT FORMAT("A,B,C,SDFA")
            try {
                n1 = Integer.parseInt(split_string[0]); // PARSING CHARACTER TO INTEGER
                n2 = Integer.parseInt(split_string[2]);
            }
            catch(NumberFormatException e){
                return -1;
            }
            // AFTER PARSING CHARACTERS TO INTEGER, THERE CAN BE NEGATIVE VALUES
            if (n1 > 0 && n2 > 0) {
                switch (split_string[1].charAt(0)) {
                    case '+':
                        result = n1 + n2;
                        break;
                    case '-':
                        // n1 SHOULD BE GREATER THAN n2, OTHERWISE return -1
                        if (n1>n2) result = n1 - n2;
                        else return -1;
                        break;
                    case '*':
                        result = n1 * n2;
                        break;
                    case '/':
                        // SECOND INTEGER CAN BE 0, ZERO DIVISION - ERROR
                        if (n2 == 0) return -1;
                        else result = n1 / n2;
                        break;
                    default:
                        return -1;
                }
                return result;
            }
            else return -1;
        }
        else return -1;
    }
}
```

We are taking *String expression* as input variable and return integer value as a

result value. When something goes wrong, method will return -1. First, We are dividing the given String into the parts(in the correct case it should have 3 parts otherwise return -1). After splitting,we should have 3 parts:
1.First digit
2.operator
3.Second digit
Then, We know that they all are in the *String* type. But we can not do calculations on String type :-). So we are trying to parse this strings into digits(if they are not even integers like "a","123a" return -1). Second condition they should be positive integers(both of them if not, return -1). Then we can do our basic calculations just taking 2nd element of string array. In the case '-', Third condition, number1 should be greater than number2 otherwise return -1.In the case '/',Fourth condition, number2 shouldn't be 0,if it is 0, return -1.If there is no problem with calculation, it will return *result*.

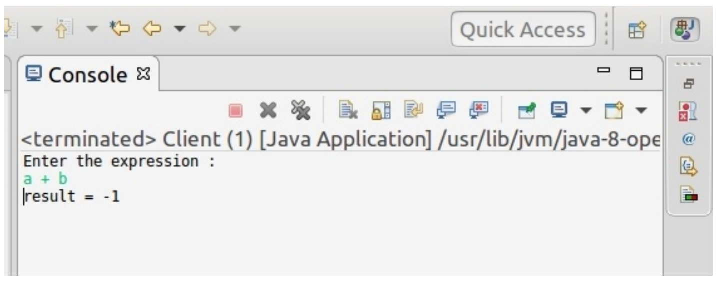
3. **SOME OUTPUTS FROM OUR PROGRAM**:
   CLIENT CONSOLE:

   (a) When n2 > n1:



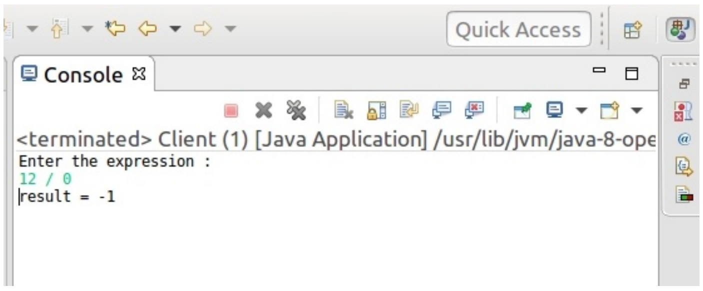
```
<terminated> Client (1) [Java Application] /usr/lib/jvm/java-8-ope
Enter the expression :
-3 + 5
result = -1
```

   (b) When Non integer format:



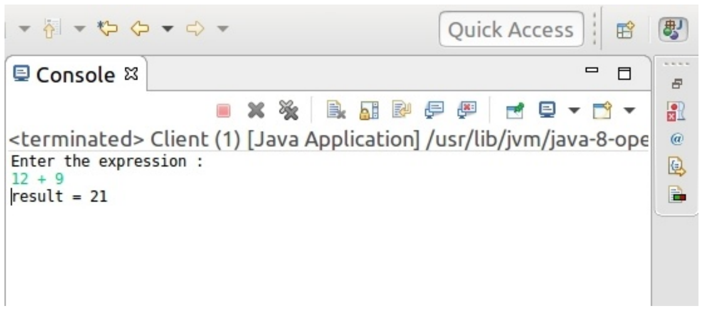
```
<terminated> Client (1) [Java Application] /usr/lib/jvm/java-8-ope
Enter the expression :
a + b
result = -1
```

(c) When n2 equals to zero:



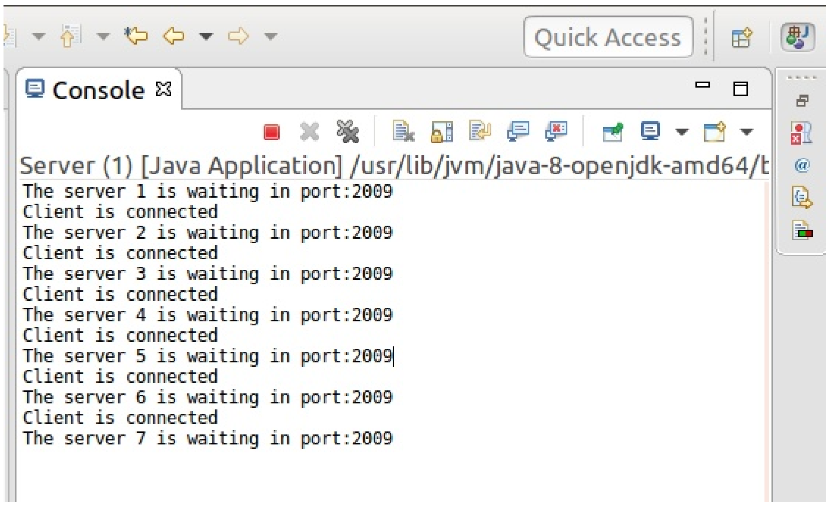
```
<terminated> Client (1) [Java Application] /usr/lib/jvm/java-8-ope
Enter the expression :
12 / 0
result = -1
```

(d) When the normal case:



```
<terminated> Client (1) [Java Application] /usr/lib/jvm/java-8-ope
Enter the expression :
12 + 9
result = 21
```

SERVER CONSOLE:



```
Server (1) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/t
The server 1 is waiting in port:2009
Client is connected
The server 2 is waiting in port:2009
Client is connected
The server 3 is waiting in port:2009
Client is connected
The server 4 is waiting in port:2009
Client is connected
The server 5 is waiting in port:2009
Client is connected
The server 6 is waiting in port:2009
Client is connected
The server 7 is waiting in port:2009
```