Question :Chat - Application – communication between Server and Client

Algorithm

Algorithm-Server

Name : Subhendu Maji

Reg. no : 18ETCS002121

Server Algorithm

Step 1 : Start

Step 2 : Declare objects.

2.1. Server socket, object of ServerSocket (constant)

2.2. ClientSocket : object of Socket (final)

2.3. BufferedReader : a object called in (used to read data from the clientSocket object).

2.4 PrintWriter Out : object of class PrintWriter. It is used to write data into the clientSocket object.

Step 3 : use accept() to wait for client for once receive create a instance of the socket.

Step 4 : Sender_thread will used to send messages to client

4.1. Recieve_thread will be used to receive messages from client

Step 5 : Start a server on a localhost port e.g. 5000

step 6 : catch any error using try-catch block.

step 7 : Stop.

Name: Subhendu Maji Reg. No.: 18ETCS002121

Algorithm - Client

Client - Algorithm

~~Step 1: Start~~

Step 2 : Connect to the port 5000 using socket

Step 3 : Listen for any message from server in a thread.

Step 4 : print the message if any message received.

Step 5 : print (server out of service) if not reachable on port}.

Step 6 : Catch any error using try-catch block.

Step 7 : Stop !

Source Code

Name: Subhendu Maji Reg. No.: 18ETCS002121

Server Code

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class Server {
    public static void main(String[] args) {
        final ServerSocket serverSocket;
        final Socket clientSocket;
        final BufferedReader in;
        final PrintWriter out;
        final Scanner sc = new Scanner(System.in);

        try {
            serverSocket = new ServerSocket(5000);
            clientSocket = serverSocket.accept();
            out = new PrintWriter(clientSocket.getOutputStream());
            in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

            Thread sender = new Thread(new Runnable() {
                String msg; // variable that will contains the data writter by the user

                @Override // annotation to override the run method
                public void run() {
                    while (true) {
                        msg = sc.nextLine(); // reads data from user's keybord
                        out.println(msg); // write data stored in msg in the clientSocket
                        out.flush(); // forces the sending of the data
                    }
                }
            });
            sender.start();
            Thread receive = new Thread(new Runnable() {
                String msg;

                @Override
                public void run() {
                    try {
                        msg = in.readLine();
                        // tant que le client est connecté
                        while (msg != null) {
                            System.out.println("Client : " + msg);
                            msg = in.readLine();
                        }

                        System.out.println("Client disconnected");

                        out.close();
                        clientSocket.close();
                        serverSocket.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            });
            receive.start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }}
```

Name: Subhendu Maji Reg. No.: 18ETCS002121

Client Code

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) {
        final Socket clientSocket;// socket used by client to send and recieve data from server
        final BufferedReader in; // object to read data from socket
        final PrintWriter out; // object to write data into socket
        final Scanner sc = new Scanner(System.in); // object to read data from user's keybord
        try {
            clientSocket = new Socket("127.0.0.1", 5000);
            out = new PrintWriter(clientSocket.getOutputStream());
            in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
            Thread sender = new Thread(new Runnable() {
                String msg;

                @Override
                public void run() {
                    while (true) {
                        msg = sc.nextLine();
                        out.println(msg);
                        out.flush();
                    }
                }
            });
            sender.start();
            Thread receiver = new Thread(new Runnable() {
                String msg;

                @Override
                public void run() {
                    try {
                        msg = in.readLine();
                        while (msg != null) {
                            System.out.println("Server : " + msg);
                            msg = in.readLine();
                        }
                        System.out.println("Server out of service");
                        out.close();
                        clientSocket.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            });
            receiver.start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
PS D:\RUAS-sem-06\DS\lab\lab_exam\lab\src> javac *.java
PS D:\RUAS-sem-06\DS\lab\lab_exam\lab\src>
```

Figure 1 creating class files that can be run on JVM

Test 1:

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\RUAS-sem-06\DS\lab\lab_exam\lab\src> java Server
hello
Client : hi
this is server
Client : this is client
abc123
Client : def456
hi from server side
Client : hello from client side
ok
Client : Okay
Bye
Client : Bye
```

```
PS D:\RUAS-sem-06\DS\lab\lab_exam\lab\src> java Client
Server : hello
hi
Server : this is server
this is client
Server : abc123
def456
Server : hi from server side
hello from client side
Server : ok
Okay
Server : Bye
Bye
```

Figure 2 Conversation between client and server

Test 2:

```
PS D:\RUAS-sem-06\DS\lab\lab_exam\lab\src> java Server
haaaaa
Client : laaaa
daskdjaks
Client : ssss
Client : asd
Client : asd
Client : asd
asd
asd
asd
ads
asClient :
Client : dasfds
Client : fds
Client : fds
Client : fd
adsdas
ads
sad
as
```

```
PS D:\RUAS-sem-06\DS\lab\lab_exam\lab\src> java Client
Server : haaaaa
laaaa
Server : daskdjaks
ssss
asd
asd
asd
Server : asd
Server : asd
Server : asd
Server : ads

dasfds
fds
fds
fd
Server : asadsdas
Server : ads
Server : sad
Server : as
```

Figure 3 Conversation between client and server