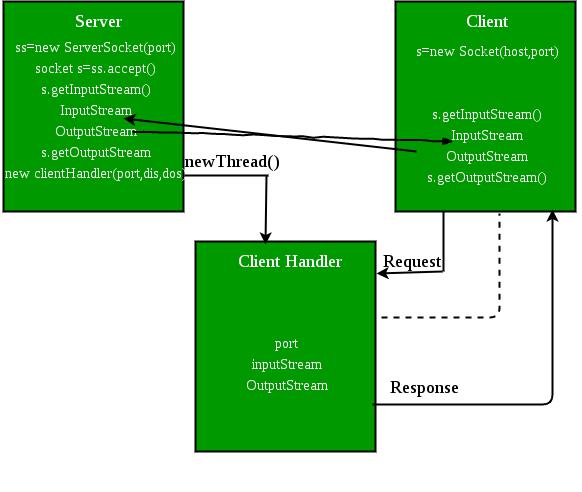
**4.A** Write a multi-threaded program in Java/c for chatting (multiuser andmulti-terminal) using threads

**Objectives:**

1. To learn about threading in Linux/Unix and Java and difference between them
2. Use of system call/library to write effective programs

**Theory:**

As normal, we will create two Java files, **Server.java** and **Client.java**. Server file contains two classes namely **Server** (public class for creating server) and **ClientHandler** (for handling any client using multi-threading). Client file contain only one public class **Client** (for creating a client). Below is the flow diagram of how these three classes interact with each other.



**Server class :** The steps involved on server side are similar to the article SocketProgramming InJava with a slight change to create the thread object after obtaining the streams and port number.

1. **Establishing the Connection:** Server socket object is initialized and inside awhile loop a socket object continuously accepts incoming connection.
2. **Obtaining the Streams:** The inputstream object and outputstream object isextracted from the current requests’ socket object.
3. **Creating a handler object:** After obtaining the streams and port number, a newClientHandler object (the above class) is created with these parameters.
4. **Invoking the start() method :** The start() method is invoked on this newlycreated thread object.

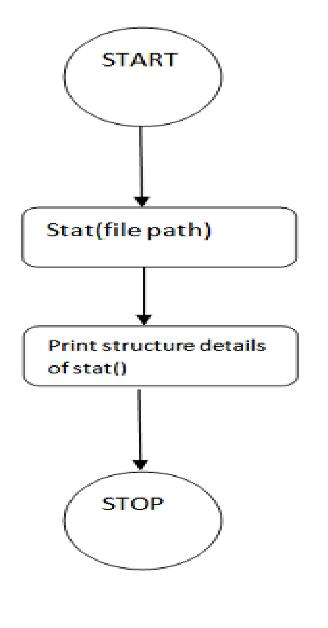
**ClientHandler class :** As we will be using separate threads for each request, letsunderstand the working and implementation of the ClientHandler class extending Threads. An object of this class will be instantiated each time a request comes.

1. First of all this class extends [Thread](https://www.geeksforgeeks.org/java-lang-thread-class-java/) so that its objects assumes all properties of Threads.
2. Secondly, the constructor of this class takes three parameters, which can uniquely identify any incoming request, i.e. a **Socket**, a D**ataInput Stream** to read from and a DataOutputStream to write to. Whenever we receive any request of client, the server extracts its port number, the DataInputStream object and DataOutputStream object and creates a new thread object of this class and invokes start() method on it.

***Note : Every request will always have a triplet of socket, input stream and output stream. This ensures that each object of this class writes on one specific stream rather than on multiple streams.***

1. Inside the **run()** method of this class, it performs three operations: request the user to specify whether time or date needed, read the answer from input stream object and accordingly write the output on the output stream object.

**Flowchart:**

****

**Data Dictionary:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr Number | Variable/Function | Datatype | Use |
|  |  |  |  |
| Class Server |  |  |  |
|  |  |  |  |
| 1 | Ar | Vector | Store which clients are there. |
|  |  |  |  |
| 2 | i | int | Count of clients. |
|  |  |  |  |
| 3 | ss | ServerSocket | Create a socket for server side |
|  |  |  | communication. |
|  |  |  |  |
| 4 | s | Socket | Socket is created. |
|  |  |  |  |
| 5 | dis | DataInputStream | Input data. |
|  |  |  |  |
| 6 | t | Thread | Used to create new thread for each |
|  |  |  | client. |
|  |  |  |  |
| 7 | mtch | ClientHandler | Object of ClientHandler type. Used to |
|  |  |  | handle client. |
|  |  |  |  |
| Class |  |  |  |
| ClientHandler |  |  |  |
|  |  |  |  |
| 1 | scn | Scanner | Usedfor any input. |
|  |  |  |  |
| 2 | name | String | Store name of client. |
|  |  |  |  |
| 3 | dis | DataInputStream | Input a message. |
|  |  |  |  |
| 4 | dos | DataOutputStream | Output to standart ouput the message |
|  |  |  | input. |
|  |  |  |  |
| 5 | received | String | Store message. |
|  |  |  |  |
| 6 | isloggedin | boolean | If the client is logged or not. |
|  |  |  |  |

**Program:**

**For Server:**

import java.io.\*;

import java.util.\*;

import java.net.\*;

public class Server

{

static Vector<ClientHandler> ar = new Vector<>(); static int i = 0;

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

ServerSocket ss = new ServerSocket(1234);

Socket s;

while (true)

{

s = ss.accept();

System.out.println("New client request received : " + s);

DataInputStream dis = new DataInputStream(s.getInputStream());

DataOutputStream dos = new DataOutputStream(s.getOutputStream());

System.out.println("Creating a new handler for this client...");

ClientHandler mtch = new ClientHandler(s,"client " + i, dis, dos);

Thread t = new Thread(mtch);

System.out.println("Adding this client to active client list");

ar.add(mtch);

t.start();

i++;

}

}

}

class ClientHandler implements Runnable

{

Scanner scn = new Scanner(System.in);

private String name;

final DataInputStream dis;

final DataOutputStream dos;

Socket s;

boolean isloggedin;

public ClientHandler(Socket s, String name, DataInputStream dis, DataOutputStream dos)

{

this.dis = dis;

this.dos = dos;

this.name = name;

this.s = s;

this.isloggedin=true;

}

@Override

public void run()

{

String received;

while (true)

{

try

{

received = dis.readUTF();

System.out.println(received);

if(received.equals("logout"))

{

this.isloggedin=false;

this.s.close();

break;

}

StringTokenizer st = new StringTokenizer(received, "#");

String MsgToSend = st.nextToken();

String recipient = st.nextToken();

for (ClientHandler mc : Server.ar)

{

if (mc.name.equals(recipient) && mc.isloggedin==true)

{

mc.dos.writeUTF(this.name+" : "+MsgToSend);

break;

}

}

} catch (IOException e){e.printStackTrace();}

}

try

{

this.dis.close();

this.dos.close();

}catch(IOException e){e.printStackTrace();}

}

}

**For Client:**

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class Client

{

final static int ServerPort = 1234;

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

{

final Scanner scn = new Scanner(System.in); InetAddress ip = InetAddress.getByName("localhost"); Socket s = new Socket(ip, ServerPort);

final DataInputStream dis = new DataInputStream(s.getInputStream());

final DataOutputStream dos = new DataOutputStream(s.getOutputStream()); Thread sendMessage = new Thread(new Runnable() {

@Override

public void run()

{

while (true)

{

String msg = scn.nextLine();

try

{

dos.writeUTF(msg);

} catch (IOException e){e.printStackTrace();}

}

}

});

Thread readMessage = new Thread(new Runnable()

{

@Override

public void run()

{

while (true)

{

try

{

String msg = dis.readUTF();

System.out.println(msg);

}catch (IOException e){e.printStackTrace();}

}

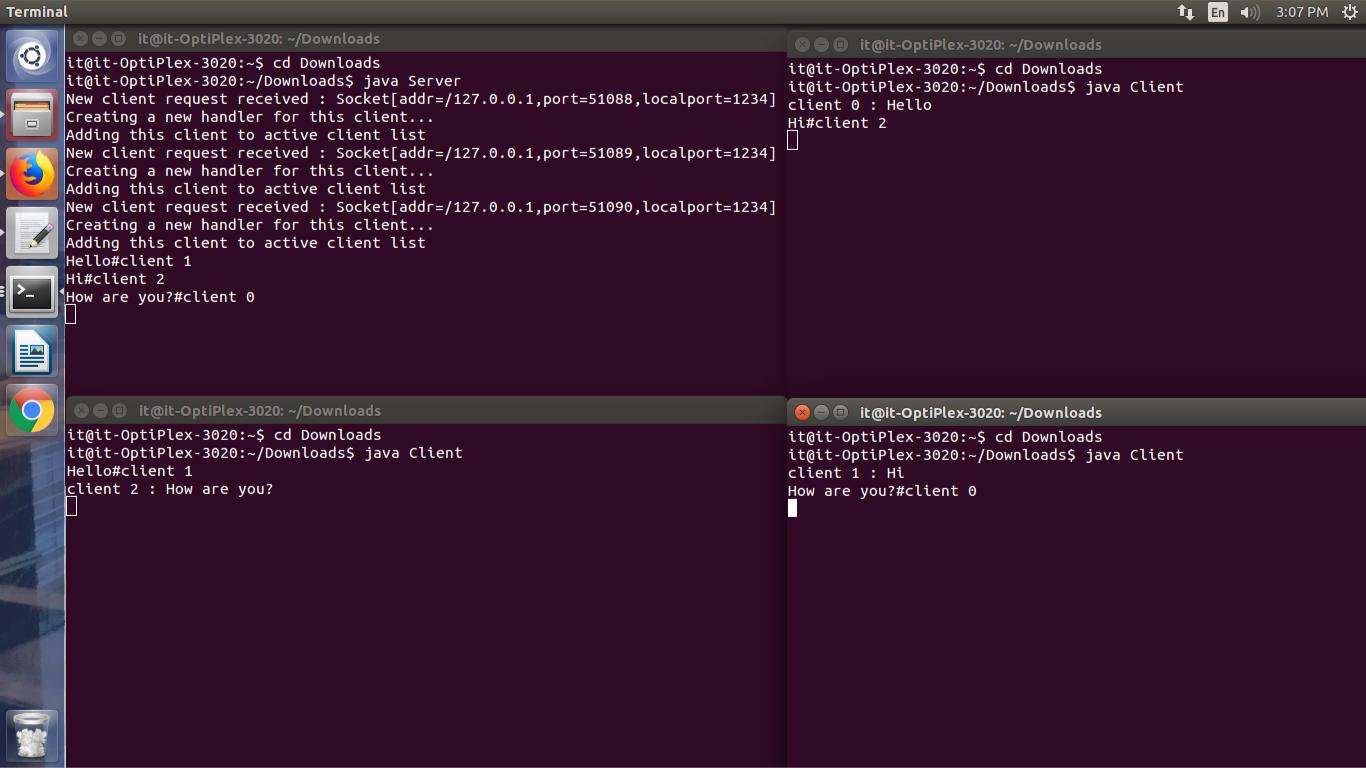
sendMessage.start();

readMessage.start();

}

}

**Output:**

****

**Conclusion:**

Various concepts and effective programming in Java using threads and sockets was studied. The concept of threading and multithreading understood.

**References:**

https://www.geeksforgeeks.org/multithreading-in-java/