Assignment on Networking
Marks: 10

In this assignment, you are required to build a client-server communication mechanism with additional features as directed. This assignment will be like a simplified mail server. You can reuse the codes in your project as you want by proper modification.

You are given a Java Project, open the project in Intellij. If you use any other IDE beside Intellij, then copy the files of the project and open up a new project in **that IDE** in order to complete the assignment.

You are given the following files as starter code:

1. **Message.java** - The main data object which is exchanged in the network.
2. **Server.java** - The server part of the network, which connects to different clients at the same time by multithreading.
3. **Client.java** - The client part of the network, which connects to the server through multiple instances by the help of multithreading.
4. **NetworkUtil.java** - A networkutil class that can be used by you to exchange data between the clients and the servers. This is under a package called **utility**.

The message class is made serializable so that every networking data can be exchanged between the servers and the clients without any hassle. **You should always use objects of this class in order to exchange information, even for sending a simple string.**

The whole premise of this assignment is to achieve two major tasks:

1. Create a **Client-to-Client communication** scheme, where one client can send a message to another by mentioning the name of the recipient. This has to be done by **leveraging the server,** that is the server itself will act as a mediator between the two parties. So naturally, all the control information (the messages, who to send the data to, table to track the clients etc) will be managed by the server.
2. Create a **Client-Server server module** that would enable the clients to receive the inbox of the respective client. That is if a client wants to see his/her inbox, he/she can use a certain command to tell the server that they want to see the stored inbox. The server will store all the inbound messages for a particular client and upon request will deliver the whole list to the client.

You are required to complete the following tasks:

1. Create a **NetworkInformation** class with the following information:

a. NetworkUtil object.
b. An **arraylist of strings** called **Inbox** in order to store the inbound messages from other clients.
c. Required **constructors and other methods** as per necessary (this is upto you to design and implement as you require).

2. Modify the **Client.java** class to include the **ReadThreadClient** and **WriteThreadClient** objects in the constructor of the client class. (see other networking codes to have an idea on what I am talking about).
3. Write the **ReadThreadClient.java** class where in the **run()** method of the thread, you will receive a message object from the **server**. Your job is to:
    a. Check whether the message is from the server or from the client.
    b. If it is from client, then simply print the message in the following way:

        From: Fahim Message: How are you!!
        Here the **From** field is for the **sender** from which a client is receiving a message from, and the **Message** field just prints the string message. Note that in the whole application, there is no message body with "," as delimiter. This is done in order to simplify the tasks for you.
    c. If the message is from the **server**, you just received the list of messages in your inbox and you are required to print the messages.
        i. You should first get the messages in the form of a single string. The messages will all be separated by **"~"** delimiter and you have to use the split**("~")** method of the string library to separate the messages in the form of a list.
        ii. Then simply print out the messages in the following format (Here the client which is receiving the inbox list from the server is **b**):

        Your Inbox:
        From: a **Message**: how are you!
        From: a **Message**: chill
4. Write the **WriteThreadClient.java** class where in the **run()** method of the thread you would simply:
    a. Take input from the user in the following manner: **Client_name, message body** eg: Suppose the sender is **Sheldon and the Recipient is Leonard**. So the message would be:

        Leonard, How are you!!

        Where the first part before the comma is the name of the **receiver** in the exact format as they have registered and the second part is the body of the message that

you want to send.

Note again that there should not be any comma in the message body to avoid complications.

b. Use the **NetworkUtil object** to directly send the message to the server.
c. For getting the inbox from the server, you should simply send the message in the following format:

Server,inbox

Mind that there is No Spaces between the comma, it should be written as it is.

5. **You should look at the existing networking codes to get an idea on how to implement these two classes.**
6. Modify **the Server.java** file to include a **clientNetworkInformationMap** Hashmap that has **Name of the client as key** and **NetworkInformation** Object as the value for each of the clients. This is made in order to store the **NetworkUtil** object of each of the clients as well as for storing the **ArrayList** of strings for incoming messages for each of the clients as well. After declaring the Hashmap, create an instance of it in the constructor.
7. Modify the **Serve** method of the **Server class** as required. The **NetworkUtil** object has already been made. Create an instance/object of the **NetworkInformation** class. The name of the client will be obtained whenever a client connects to the server via the socket. Code for capturing the name of the client is also written for you. Your job is to put the **name of the client as key and the newly created NetworkInformation object as value in the Hashmap** that you have created.
8. Finally Include the **ReadThreadServer** object/instance and pass the **clientNetworkInformationMap** and the **NetworkInformation** object to the constructor of the ReadThreadServer Class.
9. Create the **ReadThreadServer** class. This is the **most important** and the largest class that you will be working on the whole assignment. Remember, unlike the code examples we have seen in the class, this is much simpler in the sense that both reading and writing operation on the server side is done via this class.
    a. You should write the appropriate constructor as mentioned in **point 8**.
    b. Inside the run method, you are to retrieve the message object sent from the client via the **NetworkInformation** object. After getting the object, do proper checking if these are instances of the **Message** data type or not. Do necessary type casting (follow example codes for this) and get the **Sender, Receiver** and the **Message Body from the client.**
    c. Obtain two **NetworkUtilInformation** objects from the **clientNetworkInformationMap** using the sender and receiver name to make the workflow easy.

d. Now if the **receiver is another client**, do the following:
    i. Store the message (since this is an incoming message to the receiver) in the **ArrayList** of the **networkInformation object** of the receiver that you have obtained in **C.**
    ii. Simply write the message object to the receiver using the **networkInformation object** of the receiver that you have obtained in **C**.
e. Now if the receiver is the **server** itself and the **message body** says "**inbox**" in the following format:

    Server,inbox

    1. Get the Inbox arraylist from the **networkInformation object** of the sender.
    2. Preprocess the whole arraylist by converting it to a **single string with ~ as the delimiter between the strings** to be able to send the string to the client side.
    3. Create the required message object with sender as **"server"**, receiver as **"client"** and the message body as the inbox messages in the form of a long string separated by **"~" delimiter.**
    4. Send the object by the **networkInformation object** of the sender so that the client can appropriately receive the message.

Probable Marks Distribution:

- Client-Client Communication: 5
- Inbox Mechanism: 5

Video Description Link: Demo video

****Not allowed to use any online tools (chat gpt, gpt-4) or plagiarize from friends/seniors/online sources. If found during viva, there will be negative marking****