Assignment 2 – Communication using Java RMI

Due: **Friday February 17 at 11:59 pm**

**Programming problem (Pairs)**

Let’s implement a simple instant messaging (IM) application; following a client-server model. In such an application, a *single* server acts as an intermediary between *multiple* clients in the system.

When a client joins the IM application, she needs to decide on a unique user name to use for messaging. Once the server accepts the username, the client is registered, and can view the directory of users in the system and/or post messages to anyone in that directory.

The server object is in charge of managing all interactions between users, which in this simple application include:

- Registering new clients, while enforcing the unique username policy for each client.
- Receiving requests to send IMs to specific users.
- Pushing received instant messages to the correct user.

The client object only interacts with the server. These interactions include:

- Request to register user name in the application.
- Request to view the list of users in the directory.
- Sending messages to specific users in the directory.

A sample scenario in this application can be as follows:

<table>
<thead>
<tr>
<th>Time</th>
<th>Server</th>
<th>Client1(Alice)</th>
<th>Client2(Bob)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Connect to server</td>
<td>Connect to server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Register with name Alice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accept and add client to directory</td>
<td>Register with name Bob</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Request to see directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server sends the directory to Bob</td>
<td></td>
<td>Bob sees (Alice – Bob)</td>
<td>Request to send “Hello” to Alice</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server receives request and pushes it to Alice</td>
<td>Received the message:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bob says Hello</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Quit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Writing your application

For this problem, you should use the remote method invocation model (RMI) for communication. Check the following tutorial for the Java RMI. You might also need to read a little bit about the UnicastRemoteObject class, which will allow you to push messages to a client from the server.

Your implementation should include (at least) these four components,

1. The remote interface for the server.
2. The remote interface for the client.
3. The server implementation of its remote interface; i.e. writing the server-side class.
4. The client implementation of its remote interface; i.e. writing the client-side class.

You can have a fancy GUI for the IM, but there is no need. Everything can be done through the console using text commands. For example, you might only need these commands:

- `register-name`: to register the client’s user name.
- `get`: to get the list of names from the server’s directory.
- `snd-user1-text`: to instantly send a text message to user1.
- `quit`: to quit the application.

Testing your code

Start by implementing and testing everything locally. This can be done by setting the host name to null in your code.

Once you have a working and bug free version of your IM application, test the application on two machines; yours and your partner’s. Run the server and some client(s) on machine1, and run other client(s) on machine2, and start IM’ing.

Enjoy!

Deliverables

- Submit the code to the GitHub link emailed to you.
- Submit the following documents in your dedicated Google drive.
  - Your answers to the report questions attached below.
  - A screencast of you using the application on a single machine.
- Also, don’t forget to document the group discussions (in your group’s Google document), prior to the implementation. This should include:
  - The main ideas discussed within the group
  - The references/readings used during the discussion
  - The challenges faced while discussing an appropriate solution
Report for assignment 2

After finalizing the implementation of your IM application,

1. Copy below the interfaces that you used in your application.
2. How did you enforce the unique user name rule? What is the complexity (Big-Oh) of this process?
3. How is the server able to push a message to a specific client?
4. Did you design your code to handle any type of bug; such as network problems or user input errors?
   a. If yes, mention two of these errors, and copy and paste below parts of your code, in which you handle them.
   b. If no, what issues/bugs do you think might occur in your code? What do you think will happen to your IM application if such errors occur? For example, will it crash, or will the application just ignore the bug, ..etc.?