Lab 3 – Instant Messaging with RMI

CC5212-1

March 25, 2015

We're gonna build an instant messaging service using Java RMI. There will be a central directory listing all user names to co-ordinate everything. Everyone sets up an RMI server on their local machine to receive and print messages. Everyone uploads their username and the details of their server to the central directory. Then users can choose who to message.

- Download `http://aidanhogan.com/teaching/cc5212-1/code/mdp-lab3.zip`.
- First we'll quickly test the connectivity in the lab using `org.mdp.cli.UserDirectoryClient`. There are two todos.
  - Replace `"replace_me"` with the location of the central directory on the board.
  - Replace `"ahogan"` and `"Aidan Hogan"` with your own username and real name. Don't worry about IP or port for now.
  - Run the class. It will add you to the directory, delete you again and list the current users at each step. This is just to test that you can connect.
- Next we are going to implement the message server you are going to run on your local machine. Go to `org.mdp.im.InstantMessagingServer` and implement the `message(.,.)` method. *What should it do?*
- Next head to `org.mdp.cli.InstantMessagingApp`. A bunch of input/output stuff is done for you. What you need to do is implement the methods at the bottom. You may find useful code examples at the end of Monday's slides: `http://aidanhogan.com/teaching/cc5212-1/MDP2015-03.pptx`.
  - `startRegistry(.)` is done for you. It opens an RMI registry on your local machine on the port you decide.
  - Once you have a registry, you can register a skeleton for the `InstantMessagingServer` you just wrote in it. In `bindSkeleton(.)`, you will add `skel` to the registry you just opened with key `skelname`. Now users will be ready to find your server and call the method to message you.
  - Next you need to implement the methods to send messages to other users. In `messageUser(.,.,.)`, you'll need to open up a user's registry and then retrieve an `InstantMessagingStub` instance from the registry using the given `stubName`. Finally, on that instance, you can call `message(.,.)` to message that user. Return the time returned by the remote user.
  - The last step is to open the connection to the directory. This is done in `connectToDirectory(.,.)` for you. It opens another registry, this time for the central directory, finds the stub for the directory class, and returns it.
- Now we have code to receive messages locally, to setup a local registry that others can connect to, to connect to others' registries, to locate and call their message servers, and as well, to connect to a local directory where we can add our details and find out the details of others.
- Please review the `main` method before you start. It primarily just handles inputs and outputs.

- Also note down your local IP address. Run `cmd` and type `ipconfig` (look for the value on the `IPv4 Address` line).
- Finally try running the `InstantMessagingApp`: first set the argument `-n [registry location]` where the location will be given on the board. Then run the application. Enter a username, your real name, your IP address and a port (1985 is a good choice). Add yourself to the directory, get the list of names from the directory), try message some users (if you're not the first; if you are, you can still message yourself).
- OPTIONAL: In `messageUser(.,.,.)` it opens a connection every time you message. Maybe you can cache connections instead?
- Submit the `InstantMessagingApp` and `InstantMessagingServer` classes in a zip to ucursos before the Monday lecture.