Assignment 2: Java Client/Server Beaglebone Application

Name: *Stephen Holland*

Student num: *13210385*

Student Email: *stephen.holland6@mail.dcu.ie*

Description: My application has

A Client Class which has the Client GUI included. Which has a textField (for entering a message to be sent to server Note\* you need to press Enter to send your message) a TextArea (which will show all messages sent/retrieved from server) a Label for some description purpose. and a button to login.

A Server Class that can be run local or on the beagle bone. It has no GUI as this would not work when run on the beagle bone. Once started the server is constantly listening for a Client to communicate to it to be able to receive and send messages, username data using the ThreadedConnectionHandler Class. So an instance of the ThreadedConnectionHandler class is called from within the Server class.

A ThreadedConnectionHandler Class this is called from the Server class and the reason for this Class is for the multithreading purpose off allowing multiple Clients to connect to the server and work independently.

A ChatMessage Class this is used for the client to send an object to it. Then the ConnectionHandler can read this Object and convert it to a string to send back to Client so it can be displayed. That’s why the Class implements Serializable.

A ChatUser Class this does the same as ChatMessage except it is for the Username

Purpose.

A Message.bin file and MessageUser.bin file, these are to store the message and Username that is sent from client, which is then received for a new client to display the information. This is done by using FileOutputstream to write the information to the message.bin or messageUser.bin files to be stored.

A MyLeds Class which flashes the Led's on/off on the beaglebone when a message is sent or received.

So What Does my Application Actually DO?

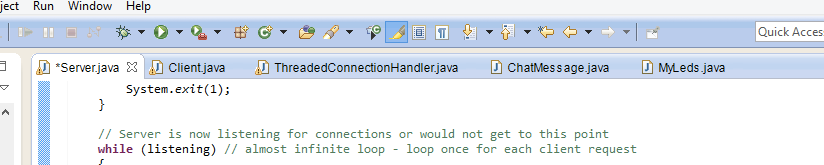
I have a functioning GUI that are threaded and can all work individually and not crash. (Shown by the use of Thread.sleep(4000) on the Run() ). The Client GUI does communicate with the Server either locally or through the BeagleBone and will send a message .I have it set up that when you first log in with a username there is no message to receive. You then type your message into the textField you then send it to the server to be saved. Then when you open up a new Client and log-in, that message will be displayed with the name of the user that sent that message. On the beaglebone a Led usr1 lights up when a message is sent the when a received by the client the Led goes off.

What Does my application NOT DO?

It does not have mailboxes that work in conjunction with Beaglebone Leds.

How to Use my Application:

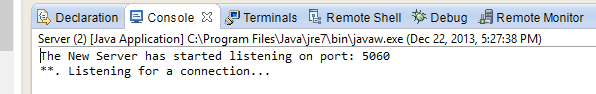
1. If running locally, Open the Five Classes in Eclipse. (Server.java , Client.java , ThreadedConnectionHandler , ChatMessage.java , ChatUser.java and MyLeds.java )

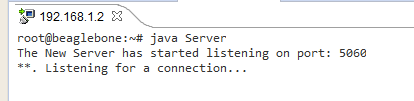


2. Run the Server.java class as i have an Object instance in the main so this will start our Server.

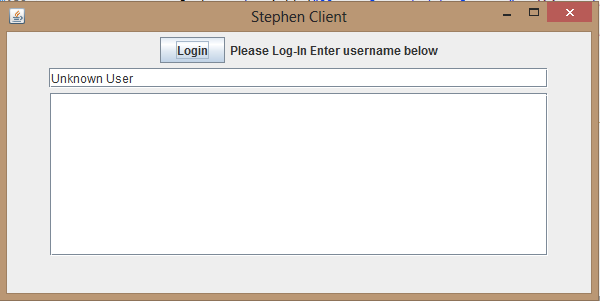
The (String sendMessageValueIP = “192.168.1.2”) variable depends on the server running on the beaglebone or locally.

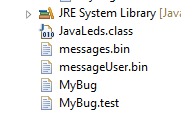
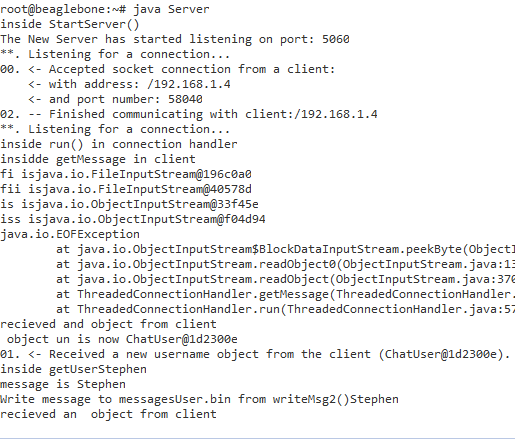
Note\* If running from beaglebone the Server.class, ThreadedConnectionHandler.class , ChatMessage.class , ChatUser.class and myLeds.class has to be added to the beaglebone. To start the Server.class on the beaglebone just use the command java Server then hit Enter.



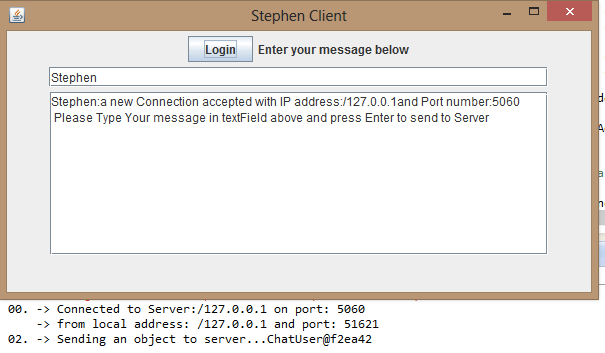
E.G Command prompt running server on beagle bone as follows……

3. Run the Client.java Class, as there is a GUI inside client i have called the Client GUI from Main (), the GUI will appear as follows. As can be seen from the console there is no IPAddress Passed in yet.

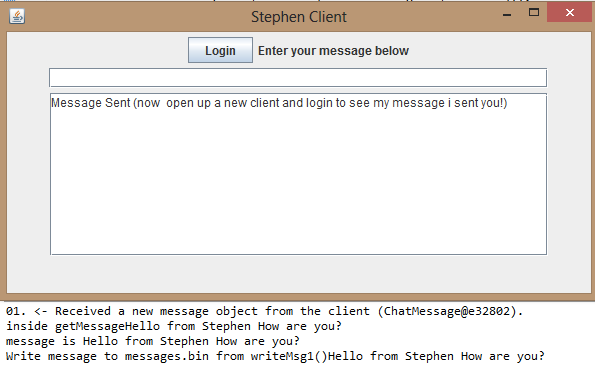


NOTE\* When the client.java is run first time two new files are created from the fileOutputStream called message.bin (which stores the message sent to server) and messageUser.bin (which stores the username sent to server which can later be retrieved. This code is run on the ThreadedconnectionHandler.java so the files will be created wherever the server is running.

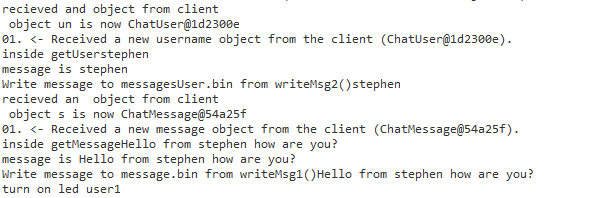
4. Next type in your Username to the textField and the press Log-in to login. The connection is made to the server and the username is simply displayed inside the textArea



5. Now you can type in your message you want to send to server inside the textField and press Enter to send.

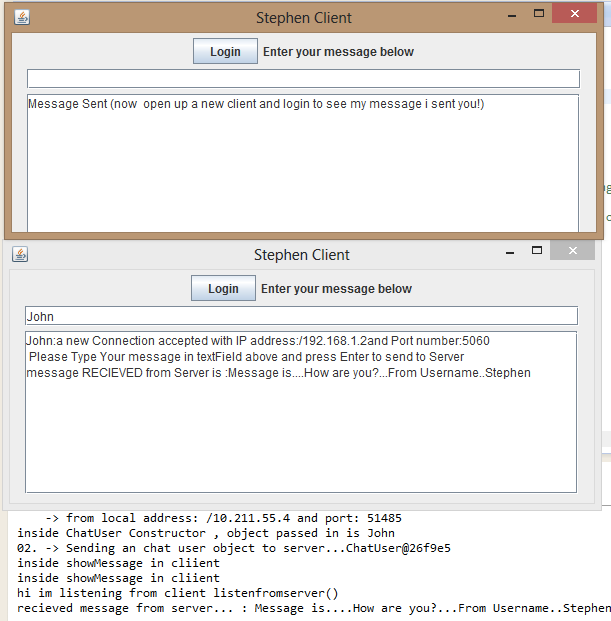


Server Terminal



6. Now open a new Client by just hitting the run button from Client.java, so two clients are running.

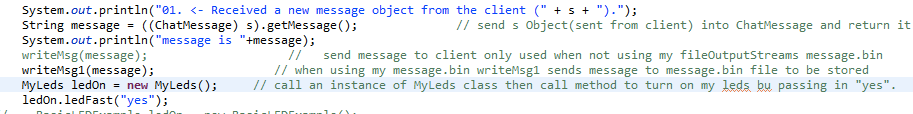
Note\* if needed you can uncomment and test the Thread.sleep(4000) on the Run() at this point

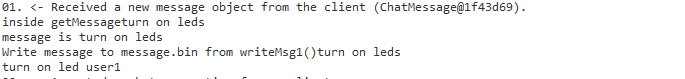
7. Next Log-in to the new Client with a new Username. and Log-in. And the message and username that was stored in message.bin and messageUser.bin files are shown in the textArea by using the getMessage() method in the ThreadedConnectionHandler that reads the files by the readCommand method. and then these files are sent back to the client for display on the txtArea. by using the send(String) method.

;



8. Led usr1 turns ON when message is sent to message.bin by calling the MyLeds class and passing in “yes” to the ledFast(String isOn) method.

Server Terminal



9. Led usr1 turns OFF when message is sent to Client class and passing in “no” to the

ledFast(String isOn) method.

