CS 342 **Project#4** **Spring 2022**

GUI Server Improvement

Due Dates: Friday April 29th @11:59pm

Since this is the end of the semester, there will be no late turn in or extensions given on this project.

Description:

In this project, you will be improving the GUI server and client sample application that we saw a few weeks back.

*****You may work in teams of two but do not have to***.**

Implementation Details:

You will make the following changes to the GUI server and client Maven project posted on BB:

- Each client will display a current list of other clients on the server.
  - this list must update for each client as others join and drop from the server.
- Each client will have the ability to text an individual client, group or clients or all of the clients at the same time.
- Your server code must be thread safe. Currently it is not.
  - You will use synchronization to accomplish this.

Extra Credit:

For 10 points extra credit, use FXML to create the client and server graphical interface. This would replace the way it is currently done in the sample app.

CS 342 Project#4 Spring 2022

Electronic Submission:

You must include a PDF file called Collaboration.pdf if you worked in a team of two. In that document, put both of your names and netIds as well as a description of who worked on what in the project. Submit it to the collaboration doc link on BB. Zip the Maven project and name it with your netid + Project4: for example, I would have a submission called mhalle5Project4.zip, and submit it to the link on Blackboard course website.

Assignment Details:

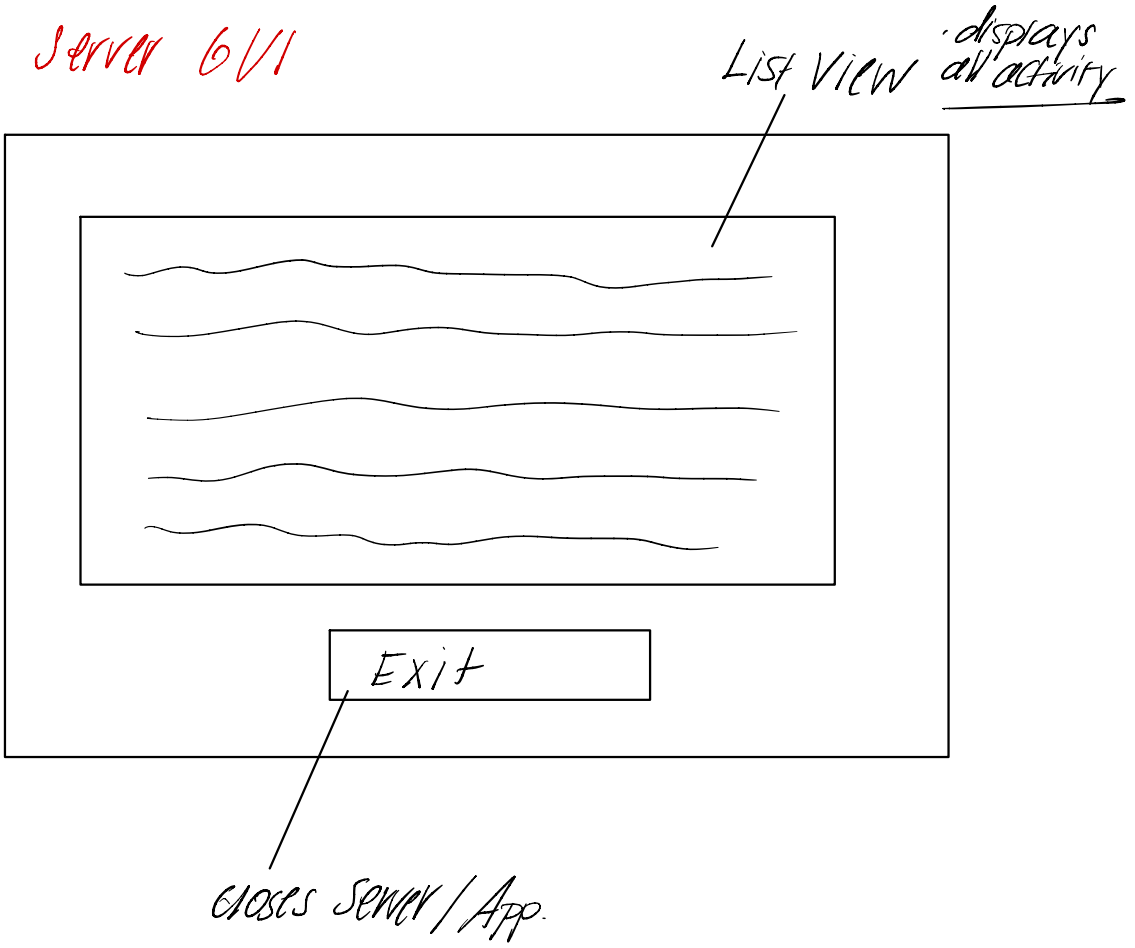
We will test all projects on the command line using Maven 3.6.3. You may develop in any IDE you chose but make sure your project can be run on the command line using Maven commands. Any project that does not run will result in a zero. If you are unsure about using Maven, come see your TA or Professor.

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

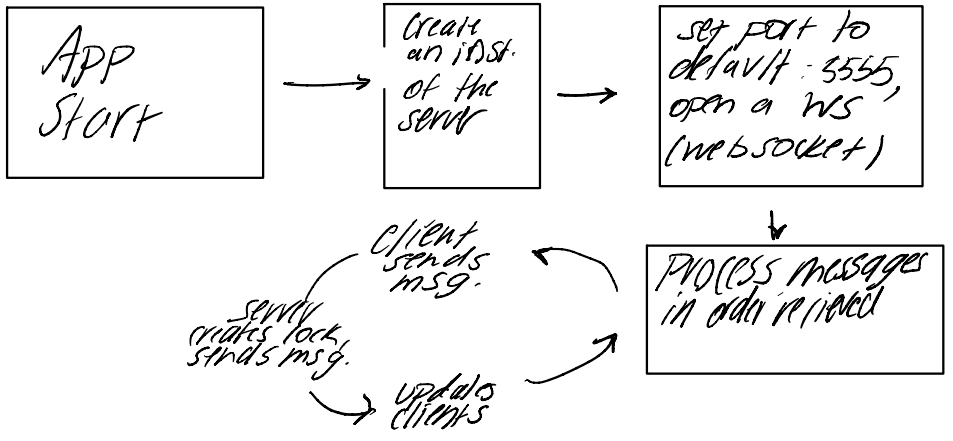
https://dos.uic.edu/conductforstudents.shtml.

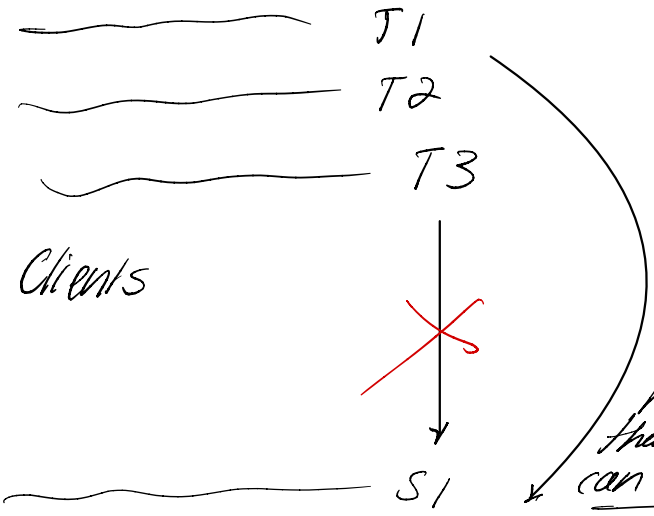
In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.

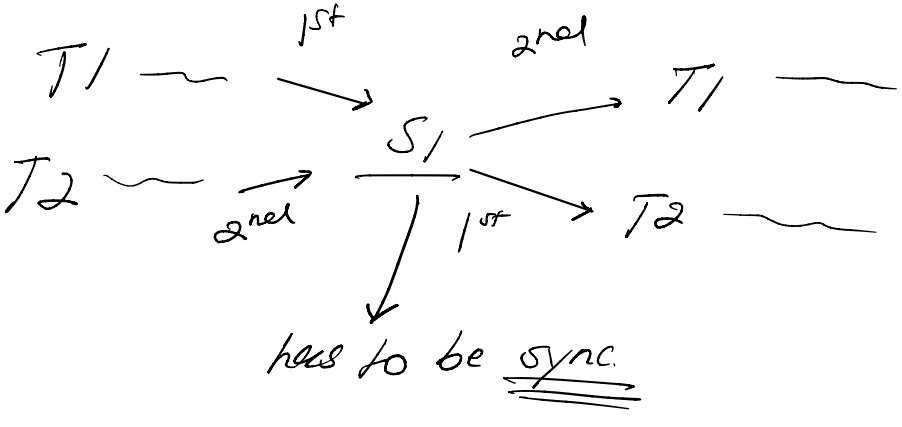
Server 6/1



Server Init. Process





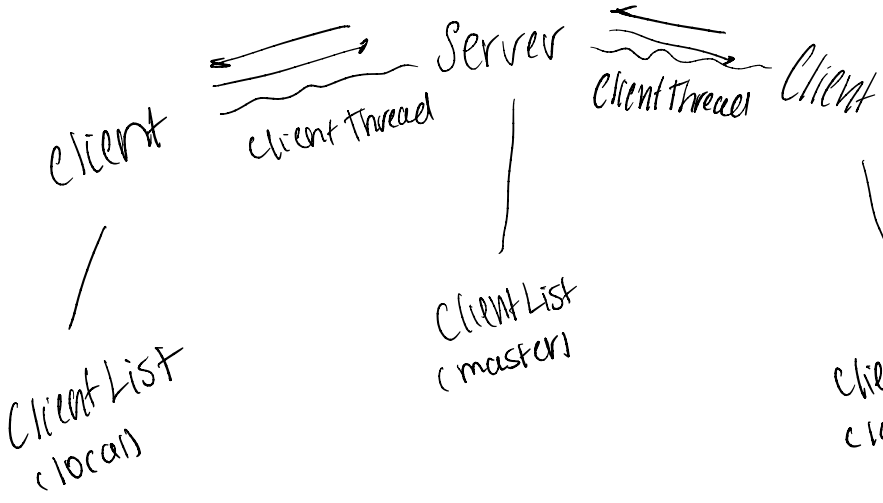


---

each Client → has a list of all the Clients

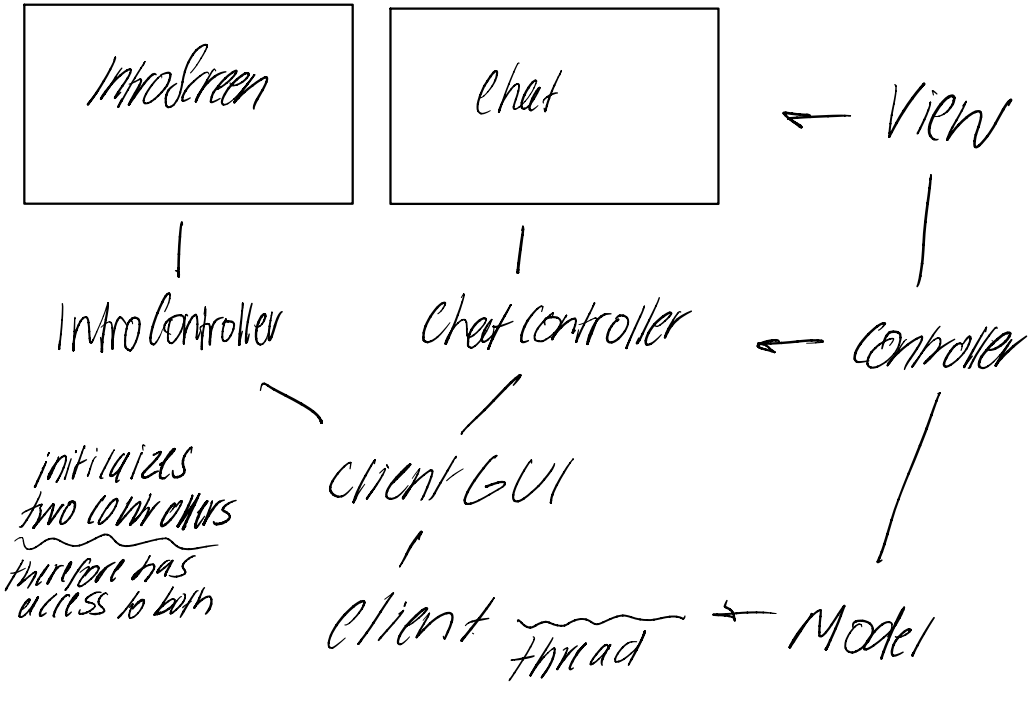
↓

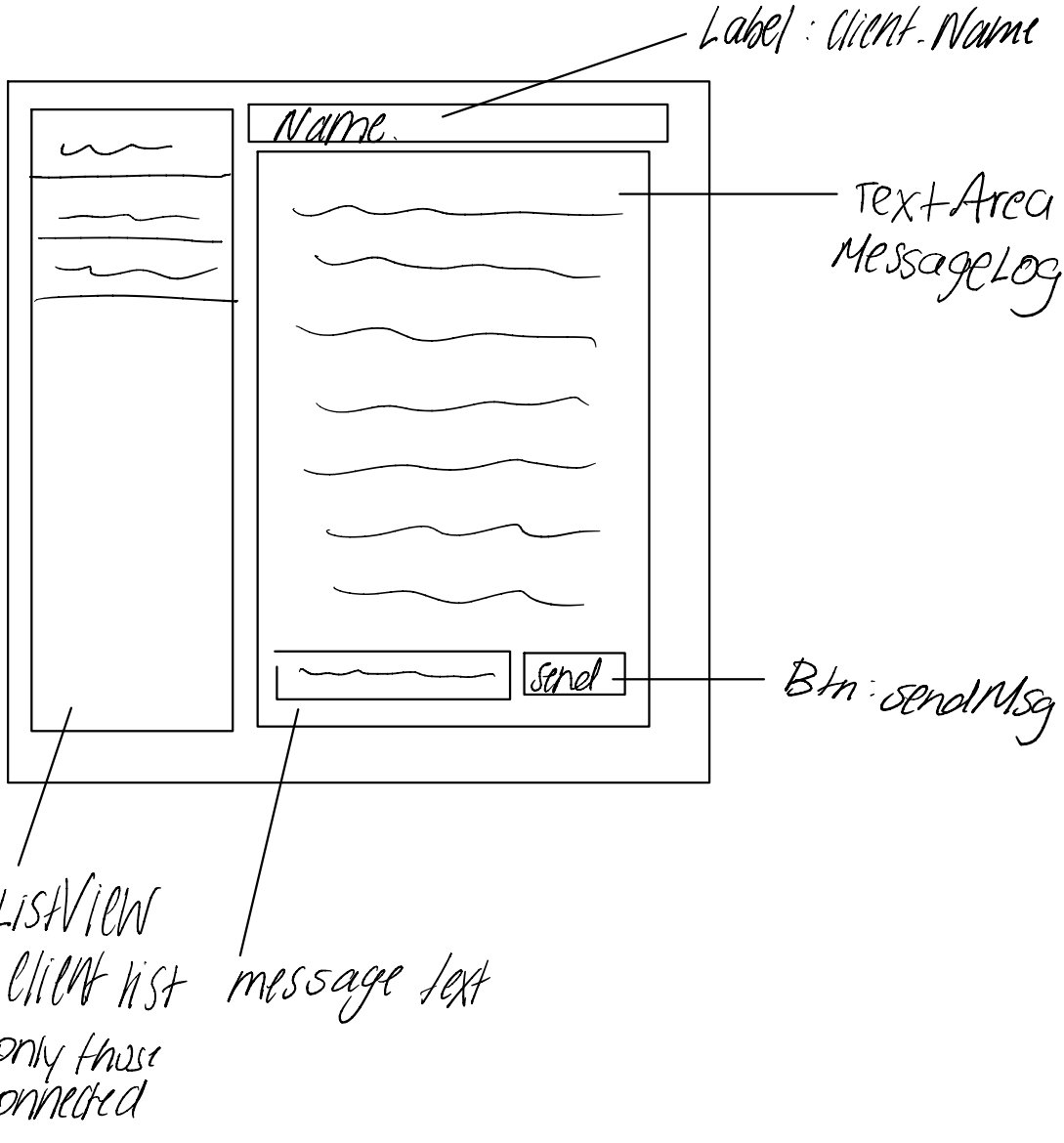
chat with each Client inv.

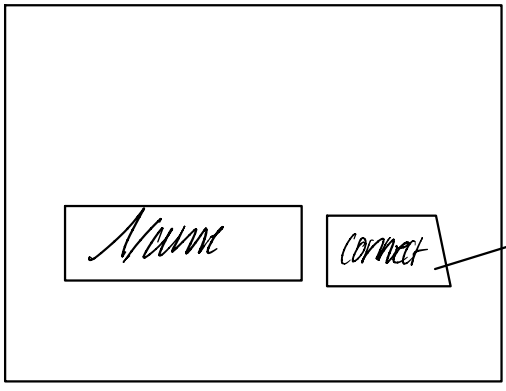


Client connects: passes their name / sends their name

Client GUI




Label : Client_Name
Name
TextArea
MessageLog
Send
Btn : sendMsg
ListView
• client list
• only those connected
message text



controller
↓
getsName;
opensConnection(name);
↓
client.start();
↓
clientThread.run();

---

sendMsg:
ChatController.sendMsg(msg)
↓
ClientGUI/ClientApp
↓
Client.Send(msg)
user.NotActive
abort
otherwise
send msg to server
↓
Server recieves a msg.
- recipent
- msg

↓
sends it to the recipent.

---

clientthread.run()
terminates
- remove from active clients
- update all clients active client lists.

---

msg = Object
Object = { String msg
String user }

Intro Screen

↓

connect

↓

show new screen / open connection

↓

assumes server is always there

↓

Introscreen controller func call to Client controller

↓

Client controller calls a function to display the new scene

↓

new scene is shown.