[CS209A-23Spring] Assignment 2 (100 points)

> Question Design: Yida Tao
>
> Code Sample: Zean HE
>
> Evaluation: Qiujiang CHEN
>
> Git & Code styles: Yunxiang YAN

Deadline: 23:55pm April. 25

Late submissions after the deadline will NOT be accepted.

Chat Room

In this assignment, you'll be implementing a chat room application in client/server mode. Users could use this application to chat with other(s), much like the way you use WeChat or QQ, but in a much simpler way. See below for details.

Implementation

We'll use **socket programming**, **multithreading**, and **JavaFX** to implement this application.

Server & Users (70 points)

The server is a single central place to host all users. It should maintain a list of users that are currently connected to the server.

Once a user connects to the server successfully, he/she will be able to see a list of other users that are currently available to chat (i.e., these users are also connected to the server at the same time). Then, the user could

- Select another user to start a one-on-one chat.
- Select multiple users to start a group chat.

During the chat, a user should be able to send text messages as well as emojis while the corresponding receiver(s) should receive exactly the same message.

GUI (15 points)

Primary GUI components of the application include:

- Main panel: shows a list of users available to chat
- Chat room: whenever users start a one-on-one chat or a group chat, there should be a separate chatroom window created for this new chat.

Application GUI should be implemented using only JavaFX. NO point will be given for this GUI part

- if you used only console for displaying all the information

- if you used other GUI frameworks, such as Vue or Swing.

Exception Handling (15 points)

Many things could go wrong in a C/S mode application. A server may crash due to internal bugs; a user may quit intentionally or accidentally (e.g., due to network problem) without notifying others. You should handle such exception cases in your program, so that both users and servers could react to unusual cases elegantly without affecting user experience.

For example, during a normal chat session, if one user accidentally goes offline (e.g., due to network problem), other user(s) in the same chat session would be notified properly without affecting their normal workflow. This means that for one-on-one chat, the remaining user should elegantly stay or quit the chat room without crashing or seeing explicit exception traces. For group chat, other users should be able to continue the chat without being interrupted.

If the server goes down, all active users should also be notified properly without crashing or seeing explicit exception traces.

Bonus (12 points)

- Account management & chat history (6 points): The server could support user registration and login. After login, users could see a list of available users, sorted by their recent chat activities. For example, if the user had a chat with A this morning and with B yesterday, then A should appear before B in the list. In addition, after selecting A, the user should be able to see their entire chat history within the chat room window.

- File transfer (6 points): users should be able to send files to others / receive files from others in the chat room.

Demo

We provide a skeleton JavaFX code for you to get started. Please click here.

Submission

Please put all project files into a single zip `A2-yourStudentID` and submit this `zip` to Sakai.

Evaluation

- **Functionalities**: You'll demonstrate your project during the lab session on April. 26 (week 11), and we'll check whether you've accomplished the required functionalities onsite.
- **Version Control**: You should use `GitHub` to manage the code changes of your project (see lab 1 for further details of how to use `git`). You should made **at least 2 commits**. Your remote repo on GitHub **should be set to `private` before A2 deadline, so that no one else will see your code.**
- **Coding Style**: You should pay attention to write readable and maintainable code along the way. See lab 1 for how to use `CheckStyle` for that purpose. **After the deadline of A2**, you can set your GitHub repo to `public`, and we'll check whether any of your commits have reduced `CheckStyle` warnings according to `google_checks.xml`.