Distributed Shared White Board

Assignment 2 -Project

Student name: Feifan Cheng

Student ID: 1164589

COMP90015
Distributed Systems
Semester 1, 2021

Table of Contents

Introduction

This project aims to build a distributed shared white board that allowed concurrent clients to use this white board to draw straight line, oval, circle, rectangle and input the text in the white board. All shapes can be displayed on the whiteboard in real time in all the online client. In addition, for the first user to join, he can use the functions of re-creating, opening, saving, kicking out, etc. on the whiteboard.

This white board architecture use **TCP Socket** technology to achieve synchronous information transfer between different clients and servers and use **String** between client and server to transfer mutual commands and attributes of drawing board graphics. All the error will illustrate by using **JOptionPane** dialog. **Txt format file** is used to store the information of one white board.

System Implementation

This system mainly composes two parts: client and server.

Figure 1 illustrates the implementation of this white board. Firstly, the server will open **TCP** socket and keep listening the connection from the client. When the first user opens the client to connect server, he will become the manager of this system who is the same as orange cylindrical client in *figure 1*. Also, there will be multiple users to connect this server to use this white board, like the yellow cylindrical clients in the *figure 1*. Each time the server receives one request, like drawing an oval, from one client and one thread is built for this connection. After that, the server will send this response to all the other client by using the thread built between server and other client. In this case, the oval can be synchronized to the white boards of all other clients.

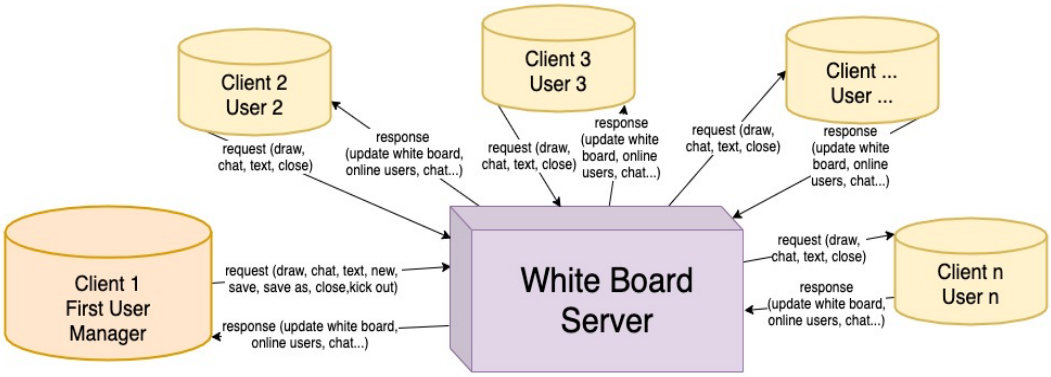

Figure 1. System implementation of Distributed Shared White Board

All the information communication between server and clients using **String**. For example, when user 2 wants to use the chat system to chat, the client will use **"S"** as the beginning of the **string** to mark this information. At the same time, user A's name, current time, the content of the chat will also be added in this **string** and send to the server. When the server recognizes the **string** beginning with **"S"** sent from client 2. It will add this **string** to the **string** used to store chat content in server

and use **"~"** symbol to separate each chat message. Then the server will synchronously send chat information updates to all the other clients and display them through their respective whiteboards.

Class Design

Client package

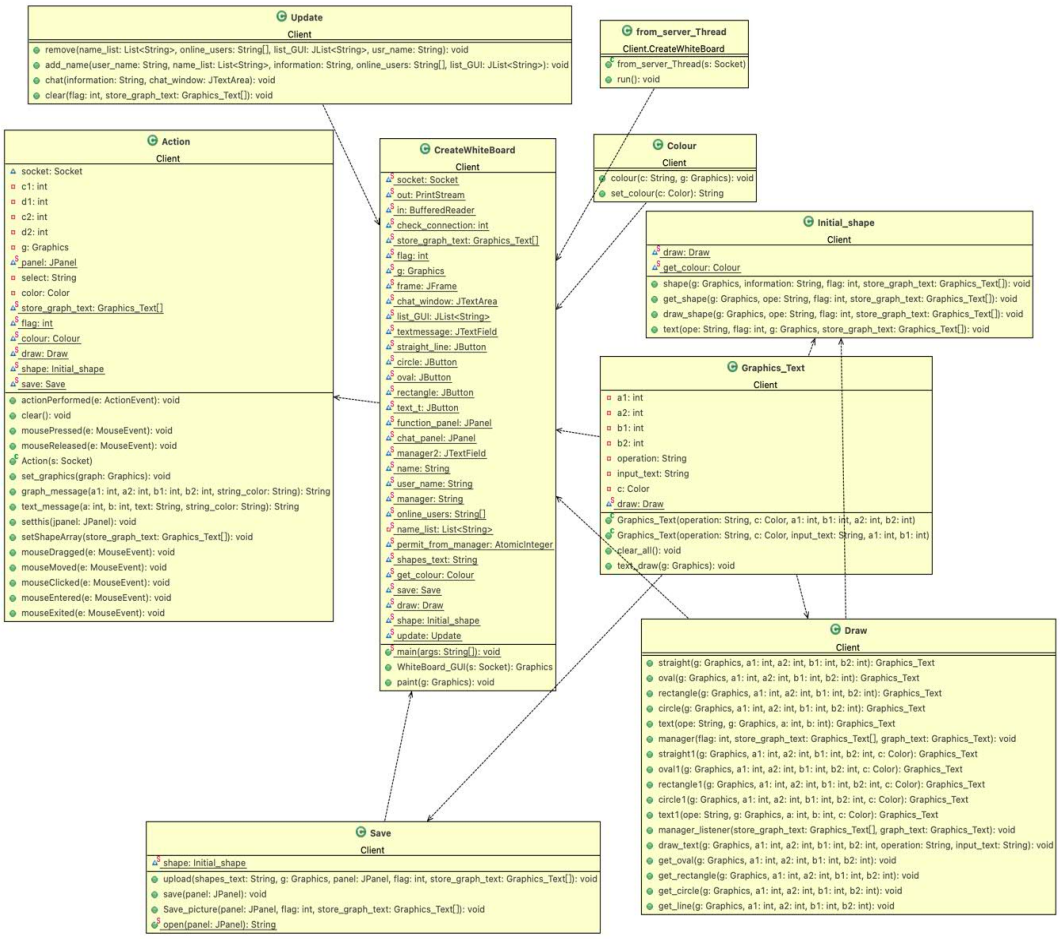

Figure 2. Client UML

CreateWhiteBoard class includes the main function and GUI of client which will receive the IP address, port number, user's name and illustrate the UI of client. Also, it will handle the thread receive from server and illustrate the response in its UI.

Action class will listen the operation of mouse, like clicking the button, drawing etc. in the white board, and convert these actions into marked strings sending to the server.

Graphics_Text class stores all the parameter of the shapes and texts.

Draw and **intial_shape**s class will draw and initialize all the shapes and text in the white board.

Update class is using to update the information, like online users, chat windows etc. in white board.

Save class achieve the function to save current board and open, upload previous board.

Colour class is to get and set the color as corresponding string.

from_server_thread receives thread from server and the run function was overrided to achieve the function of this white board system.

Server package

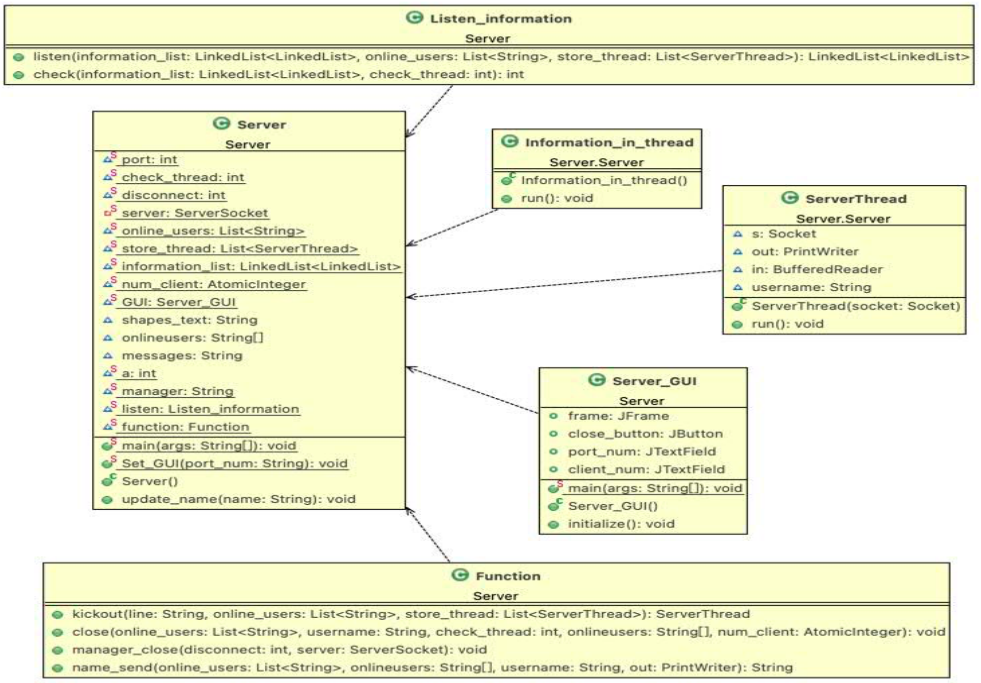

Figure 3. Server UML

Server class includes the main function and receive the port number. All the string command received from client will also handle and send to other designated clients in this class.

Server_GUI class achieves the UI of server. It shows the online client number and port number.

Function class handle the kick out, client or manager close the client function and send the online name list to all the clients.

Listen_information class uses a nested LinkedList storing string instructions that needs to be sent to all the other clients.

Information_in_thread and **ServerThread** creates Socket and thread to communicate with clients.

UI Implementation

Client UI

The client UI shows as follow:

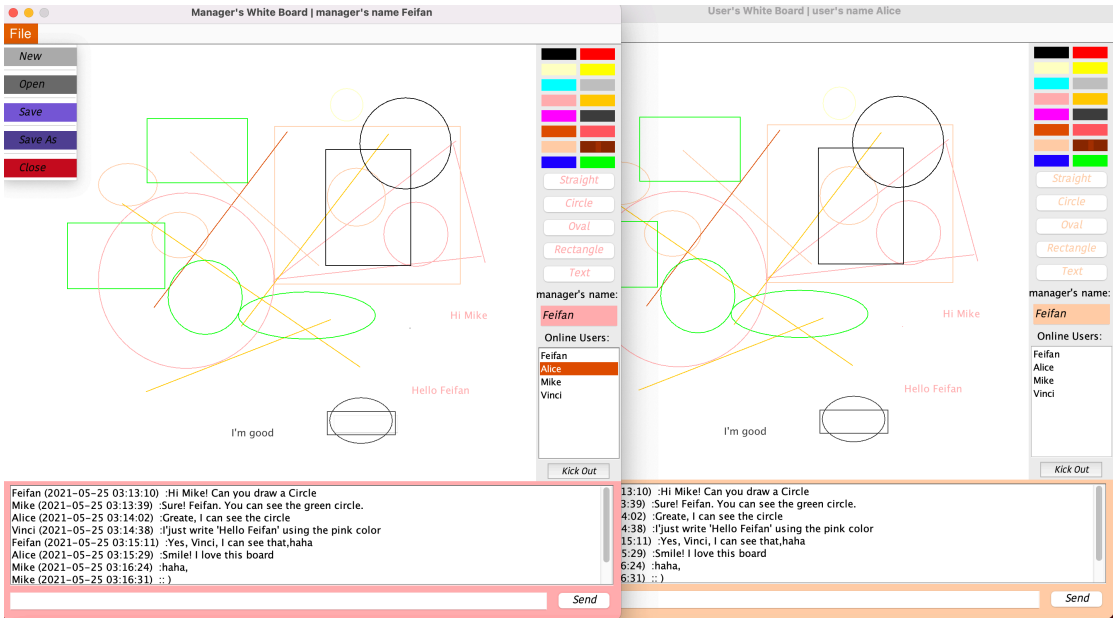

Figure 4. Client UI

We can see from the UI of the client in *figure 4*. At the top of UI, it illustrates the name of the current user and whether he is the first user to create the white board (the manager). Only the manager can use the "**New**", "**Open**", "**Save**", "**Save As**" button in **File** menu bar in the upper left corner. It separately means clear and create a new white board, open a previous white board, save current white board, save the current white board as a picture. When the manager clicks "**Close**" button, the whole system will close. But the other users click "**Close**" button, only he exits.

There are sixteen different colors to choose from on the right-hand side. When the user selects one color as the current drawing color. The color of the five shape buttons and manager name, and the background color of the chat panel will be set to this color. When user click "**Text**" button, he can type text he wants anywhere inside the white board by using the current selected color.

When the user selects one shape or text, he can use the mouse to draw the corresponding shape with the current color in his white board. And it will be synchronized to all other users' white boards.

On the right side, the names of current white board manger and all online users will be illustrated. On the manger's white board, he can use the "**Kick Out**" button to kick player other than himself.

In the lower part of white board is the chat panel. All users can chat with all online users here. Their name, current time and chat content will be displayed on everyone's white board simultaneously.

In the center of white board, the content of the drawing will be displayed to all users simultaneously.

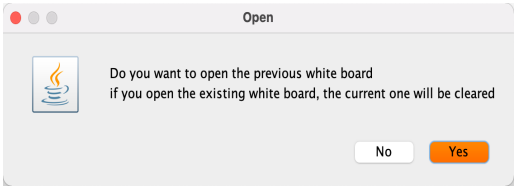

Figure 5. Notification UI

From *figure 5*, some notification UI will use **JOptionPane** dialog to notify the user. For example, if manager wants to open a previous white board. A dialog will notify the user that this operation will clear the current white board content.

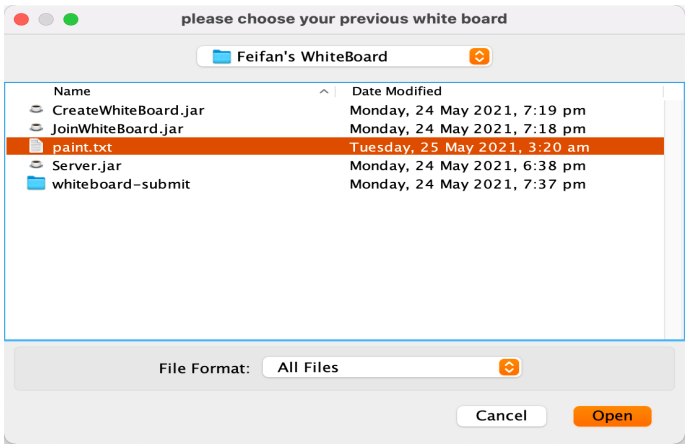

Figure 6. Choose file UI (Open, Save, Save As function)

From *figure 6*, if manager uses "**Open**", "**Save**", "**Save As**" button, choose file UI which use **JFileChooer** will help manager to select the path to open the previous white board file or save the current white board.

Server UI

The server UI shows as follow:

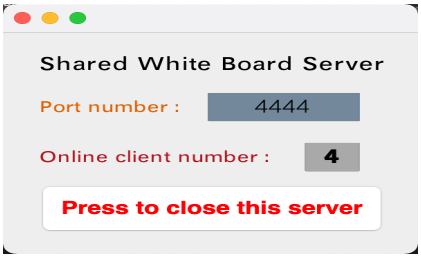

Figure 7. Server UI

From *figure 7*, the Server UI will display the port number of the current server and the online client number (with the change of login and logout of users, manager kicks). The "**Press to close this server**" button can exit and shut down the entire system.

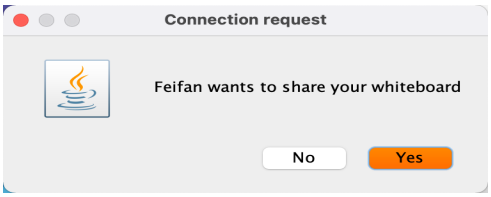

Figure 8. Join UI

When someone wants to join the white board, a dialog will illustrate, only clicking "**Yes**", this user can join the current white board.

Error Handling

A dialog UI (**JOptionPane**) will show all the errors to user when they use this shared white board.

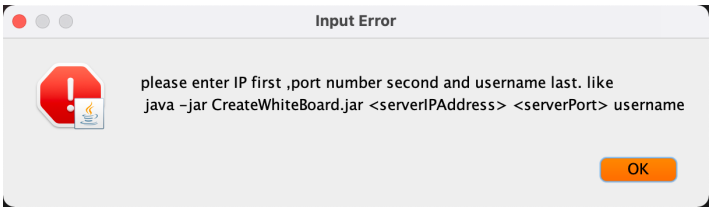

Figure 8. Error UI (wrong argument input)

Like one input error shows in *figure 8*, if users input the wrong command jar. Line. A dialog will tell the user to input the right format command lint to launch his client.

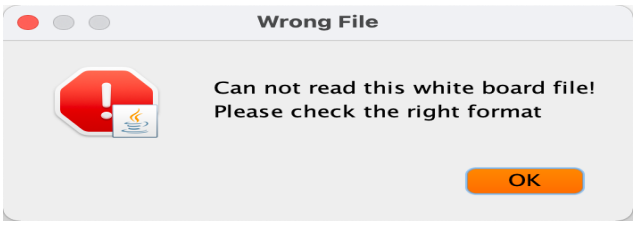

Figure 9. Error UI (wrong argument input)

From *figure 9*, an error dialog shows the wrong white board file format when the manager wants to open a previous white board file.

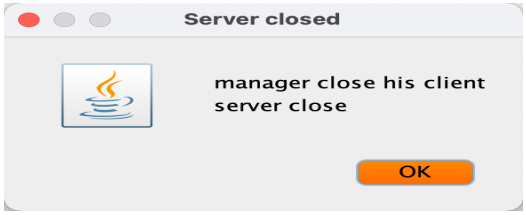

Figure 10. Error UI (wrong string mark receive from server)

In addition, from *figure 10*, when the manger closes his client, I set to send string of "**managerclose**" as a mark to client. Because in this project, when the manager closes his client, all this system will close. I have not written any code to handle this "**managerclose**" mark in client class. When the client receives this string mark. It will throw an Exception error. At this time, I set to use **JOptionPane** dialog to show that the manager closes his client and shut down the current client system.

Also, there are some other errors happened in this system, like the error happened when sending the message, save the file, etc. All these errors will illustrate in these dialogs.

Analysis

The overall system user **TCP socket**, because it is more reliable and stable. By using TCP technique,

two or more clients could use this distributed shared white board to draw the simple shapes, texts they want and chat with other users simultaneously. This TCP system could easily handle all these operations.

The simple **string** is used for message exchange. Because for the drawing shapes and chatting function, only using string to record the coordinate, texts, colors and some mark strings is enough. It will be simpler and easier to operate. Then each client updates the corresponding operation synchronously on its own white board according to the received string.

A **txt format file** is used to save the graphic information on white board. Because String is used to save the information of the coordinate, texts, colors and mark string of each graphic on the white board. The string is easily saved in a txt format file. And when manager wants to read the saved previous white board information next time. It is very easy to read this string from a **txt format file**.

Conclusion

All the basic and advanced features are achieved in this system. Users can use 16 colors to draw oval, straight lines, circles, rectangles and type and texts they want anywhere on the white board. Similarly, all users can use the chat panel area to chat with each other. The first person who creates the whiteboard becoming the manager. He can create a new whiteboard, save the current whiteboard, open the previously saved whiteboard, and can also have the privilege of kicking out other users.