COMP90015 ASSIGNMENT 2
DISTRIBUTED SHARED WHITE BOARD

A PREPRINT

Jingda Kang 1276802
University of Melbourne
jingdak@student.unimelb.edu.au

May 30, 2022

1 Introduction

There are many usage of shared whiteboards on the Internet. Shared whiteboards allow multiple users to draw simultaneously on a canvas with different kinds of tools, such as pencil, rubber, shapes etc, to cooperate to finish a paint work. In this assignment, I build a shared white board based on RMI in Java, which allows multiple users to work on it concurrently without much delay. The shared white board possess several functions such as free hand drawing with the mouse, drawing shapes, choosing colors, inserting text, etc. The issues I encountered during the development will be addressed in critical analysis.

2 Architecture

The communication between multiple users is implemented by using RMI (Remote Method Invocation). Java RMI is an extension of the Java object model to support distributed objects. The registry is created on localhost on my laptop in Java code. After the design of the remote interface and servant class which implements the interface, the clients could then look up to the servant with correct registry. Since the clients could directly talk to each other, I design all the function in servant to operate the white board from server perspective and pass the whiteboard to each client to present a local image. In the regard of all the information of the whiteboard should be synchronized in the whole running process, I build several threads running accompany with the program as while as it is working to pass and get synchronous message to and from the server, which is the key design to make the application run robustly. The class designed are presented in the diagram as in Figure 1.

According to the requirements, the first client that invoke CreateWhiteBoard should be the manager and has additional privileges. Thus in the design, I set two classes for two kinds of clients: manager and user. For manager, the GUI contains additional functions such as kicking out a certain user and open an existed canvas file. User GUI only inherits other functions from manager GUI. All GUI design in this project are using Java awt and Java swing.

The white board operations contain free hand drawing, shapes drawing including line, circle, triangle and rectangle, text inputting, line size selection and color selection. Details are presented in functions.

3 Functions

3.1 Basic Features

In the final work, the white board application allows multiple users to draw on a shared interactive white board without an obvious delay. The white boards presented to all clients are completely identical. The manager uses CreateWhiteBoard to start a new shared white board application, while other users use JoinWhiteBoard to make a request to join the shared white board. After the manager grant the permission, users could start working on the shared white board. The demo screenshot of the usage of the application is presented in Figure 2.

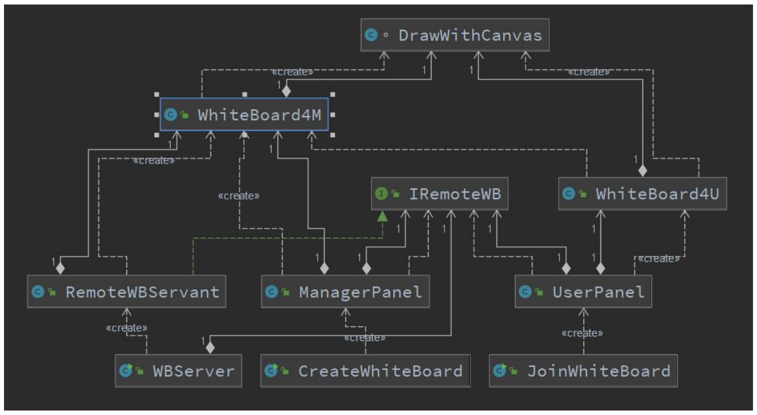

Figure 1: Class Interaction Diagram

The pencil function allows clients to draw free-hand paints anywhere in the canvas.

The eraser function allows clients to eraser the paint using a rubber in the canvas.

The line function allows clients to hold the mouse to draw a line in the canvas.

The circle function allows clients to hold the mouse to draw a circle in the canvas.

The rectangle function allows clients to hold the mouse to draw a rectangle in the canvas.

The text function allows clients to input a text with a click in the canvas.

The color section function allows clients to select a color from color chooser.

3.2 Advanced Features

The user list component presents the current clients that are working on the shared white board concurrently. The manager will be tagged to notify all the clients. Only the manager could kick out certain users on the user list. If a user was kicked out by the manager, the user panel will be closed and a message say ‘The manager has moved you out’ will prompt out. The user list will also update the user information. If the manager close the manager panel, then all the clients currently using the white board will automatically disconnect and be notified with a message saying ‘The manager has turned off the white board’.

The chat box component provides the real-time communication between all the clients. Client could type the message in the input box and click send to present the message in the chat box. All the clients could view the chat box content.

The manager has the privilege to open an image on the canvas by clicking the open file button. The image file selected will over the current image on canvas. After that all clients could draw on the canvas as usual.

The manager has the privilege to save the current canvas in an image file by clicking the save as button. The current canvas will be screenshot and saved in a selected file for future usage.

All the clients could choose the line size by clicking the line size button.

4 Critical Analysis

In the development process I have encountered and resolved many issues which occupy a really long time in this project. For example, in the early stage of the development the clients have problem with look up to the registry to use servant functions despite that all registry has been started well in the server. It turns out to be my directory name contains space so the client couldn’t figure out the RMI address using TCP protocol on my localhost.

Another key issue related with the synchronization between server white board and local white board. The updating process between two sides have to be strictly separated to make sure override won’t happen on the white board. Thus, I design an additional variable for each whiteboard to tag the update status.

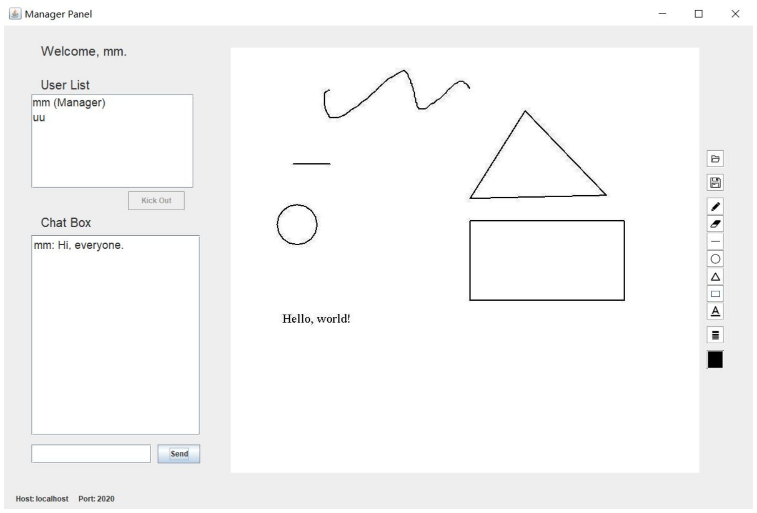

Figure 2: Manager GUI

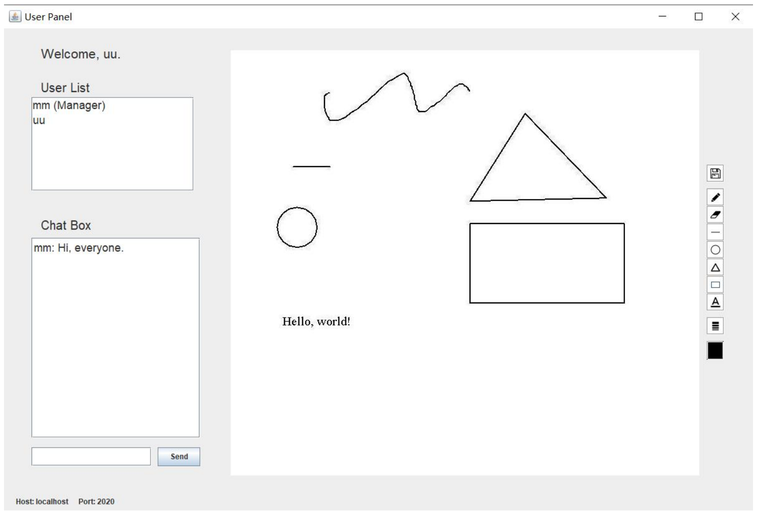

Figure 3: User GUI

5 Conclusion

In conclusion, I implement RMI in this project and design a distributed shared white board for multiple clients to work together on a canvas without an appreciable delay. I also build contact exception handle model in the project to clearly showcase the unexpected accidents happened at run time.