COMP90015 Distributed Systems
Assignment 2: Shared White Board

Wei Ge 1074198
The University of Melbourne

May 28, 2022

1 Introduction

This report will expound on a Shared White Board application under the Java RMI architecture. This project has two sides: the White Board Server and the User**.** By using the RMI, this application allows the concurrent users to remotely manipulate the whiteboard, check online users and use the chat room. In addition, the host user is allowed to kick the guest user, close the whiteboard, create a new whiteboard and save the whiteboard. Afterwards, all the exceptions, such as the IO exception and Null Pointer exception, are handled well through a user's UI notification.

2 Components of the System

2.1 Overall

This program contains several Classes:

- User Component
    - User (the super class of Guest and Host) & User UI (the super class of UIs)
    - Guest & Guest UI
    - Host & Host UI
    - ShapeDraw (the strategy to draw different shapes)
    - Canvas
    - UIListener
    - *iUser* (the interface of user, and is the Skeleton for Whiteboard in the RMI)

- White Board Component
    - WhiteBoardServer
    - WhiteBoard
    - *iWhiteBoard* (the interface of user, and is the Skeleton for Whiteboard in the RMI)

The domain model is shown in Figure 1.

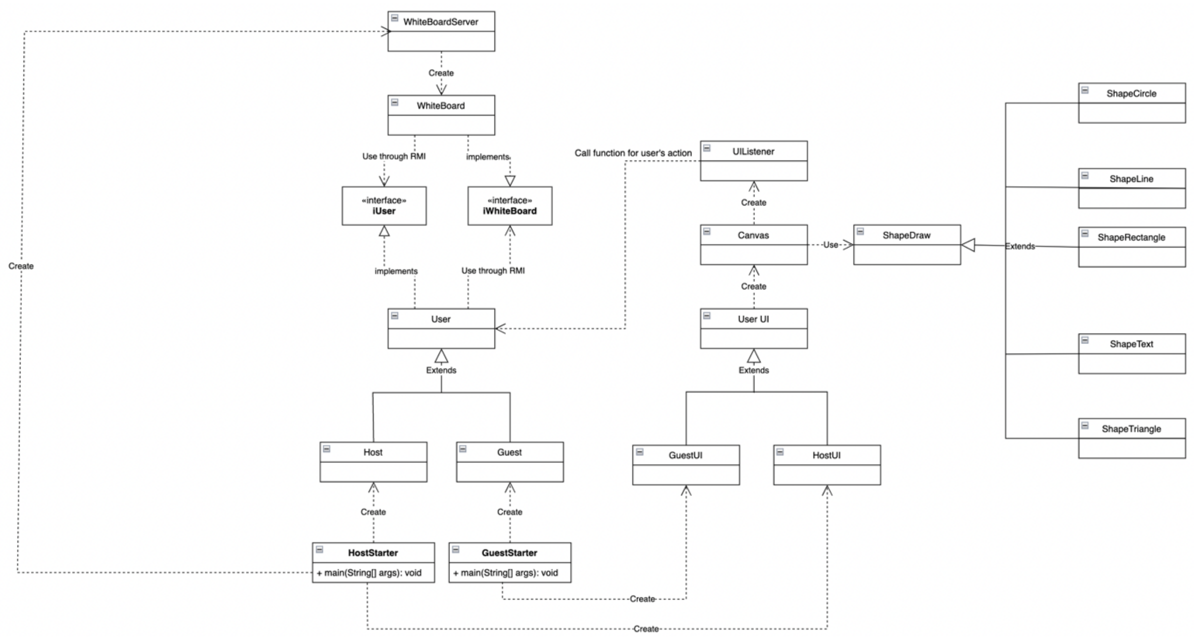


Figure 1: The System Domain Model

2.1 User Component

The typical user methods are packaged in the *User*. For example, both the host and the guest can send the message, draw the shape, quit the whiteboard, etc. In addition, the *User* implements the *iUser,* which includes the methods be called by the whiteboard. In other words, the *iUser* provides the *User*'s Skeleton to *WhiteBoard* in the RMI architecture.

Similar to the *User*, the common UI components are positioned in the *UserUI* followed by being extended by the *HostUI* and *GuestUI.*
Moreover, the *UserUI* contains the *Canvas* to show the shape, and the canvas is listened to by the *UIListener*. Different drawing logic is separated by strategy design pattern, the *Canvas* uses the strategies according to user's choice. Furthermore, all methods called by *UserUI* and *UIListener* are from the *User*, they will not directly communicate with the *WhiteBoard,* the *User* acts as a proxy between them.

2.2 White Board Server Component

The host and guest are all the users to the whiteboard, and the host is the user with unique authority. In this project, the whiteboard server will be created by the host.

The *WhiteBoardServer* creates *WhiteBoard* to include remote methods for users to use, the methods include board messages, kick user, etc. The *iWhiteBoard* contains methods to be called the user, which is the Skeleton for *User*. Moreover, the *WhiteBoardServer* includes the user list and a shape's record list to synchronize the new user's canvas.

3 Technology and design

The **Graphics** is used in this project to text or draw different shapes on the canvas. Whenever the user clicks or drags on the canvas, the listener will capture the mouse's coordinate and other information, and then send them to the whiteboard. After several transfers, all the users will receive the records and do the corresponding drawing logic to draw through Graphics methods like *drawString()*, *drawOval()*, etc.

The canvas is deployed through **JPanel**, it provides the *paint()* and *repaint()* methods for the user to paint and repaint the canvas.

Moreover, the **JColorChooser** allows the users to choose the color they want. The **JFileChooser** is positioned to show the files' path to the host.

To communicate between the user and the whiteboard, the **RMI** architecture is positioned. Both the user side and the whiteboard side provides Skeleton for each other to be the Stub. Moreover, the basic function in RMI is finalized by multithread, which means the methods on the server already act under concurrency. The **synchronized** is used when a new user enrols into the server, preventing duplicate usernames from not being checked out.

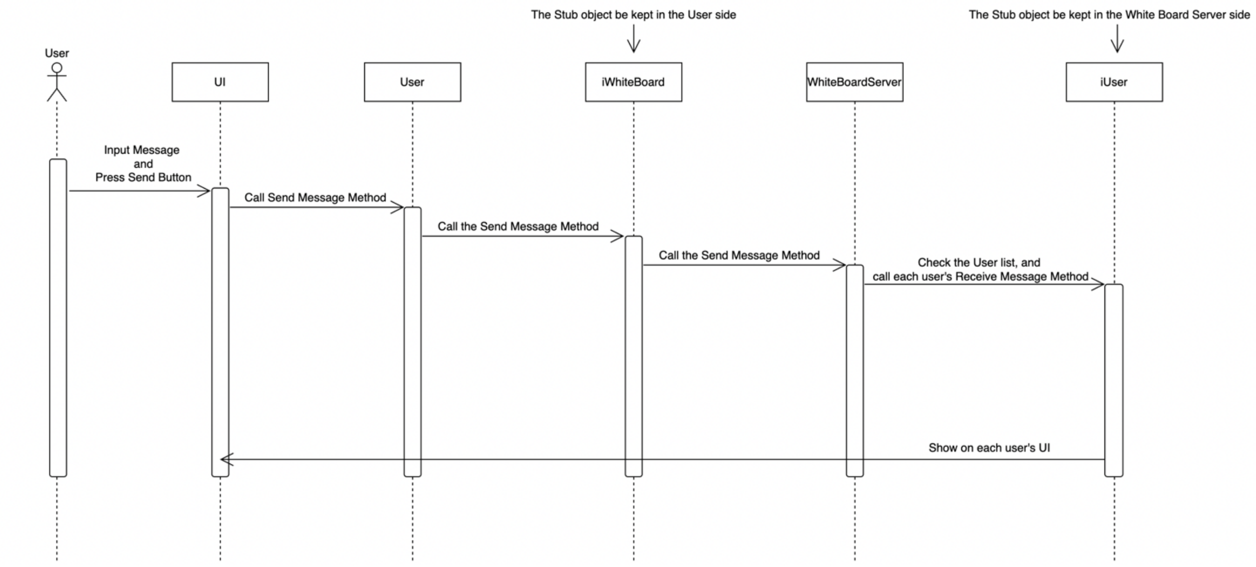
In addition, to avoid the blocking of UI, the GuestStarter and HostStart run the UI and other operations into different threads. The *UserUI* extends the **Thread**. And the *User* implements the **Runnable** because it already extends *UnicastRemoteObject.*

To transfer the information between different sides, the **JSONObject** is used. Each user's draw will generate a JSON record which contains the type of shape, the coordinate, the color, and the text if the shape has. Besides, the messages are delivered in **String** format.

4 Process

4.1 Interaction

While the RMI has already finished the TCP and Socket communication in its fundamental, it is easy to interact between the users and the server. The *iWhiteBoard* contains methods to board information to all users or specified users. After establishing the RMI connection, the host and the guest can transfer information by calling the *iWhiteBoard* methods. For example, the processing to send the message is shown in Figure 2.



The Stub object be kept in the User side
The Stub object be kept in the White Board Server side
User
UI
User
iWhiteBoard
WhiteBoardServer
iUser
Input Message and Press Send Button
Call Send Message Method
Call the Send Message Method
Call the Send Message Method
Check the User list, and call each user's Receive Message Method
Show on each user's UI

Figure 2: Chat Sequence Diagram

Moreover, the process of drawing shapes is shown in Figure 3. After the user draws a new shape on the canvas, the listener will call methods to generate a corresponding record and then send the record to the whiteboard server. After that, the whiteboard server will board the new record to all users and store the record in the server. When a new user joins the server, the whiteboard will send the whole records list to the user to synchronize its canvas.

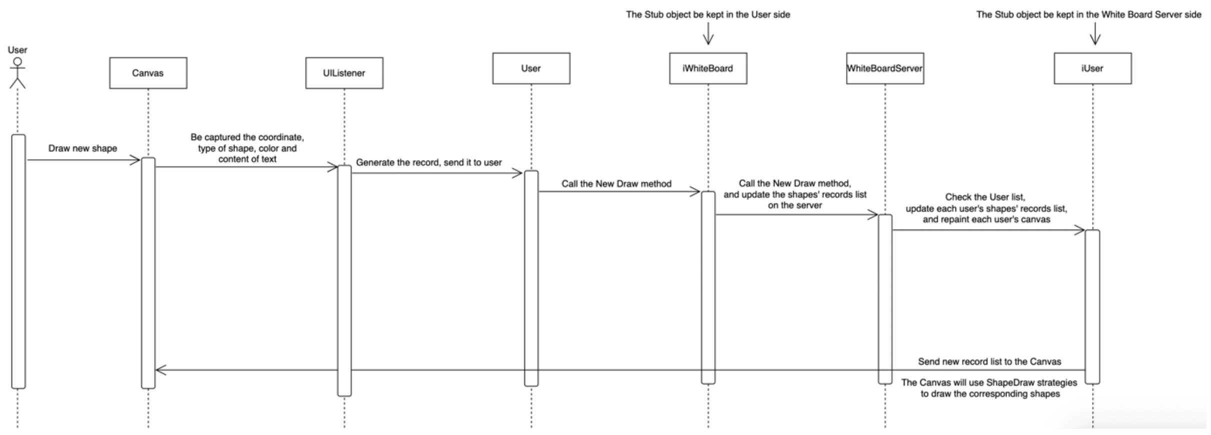

The Stub object be kept in the User side
The Stub object be kept in the White Board Server side
User
Canvas
UIListener
User
iWhiteBoard
WhiteBoardServer
iUser
Draw new shape
Be captured the coordinate, type of shape, color and content of text
Generate the record, send it to user
Call the New Draw method
Call the New Draw method, and update the shapes' records list on the server
Check the User list, update each user's shapes' records list, and repaint each user's canvas
Send new record list to the Canvas
The Canvas will use ShapeDraw strategies to draw the corresponding shapes

Figure 3: Draw Sequence Diagram

4.2 Authority

To identify the users, the Authority attribute is used. There are two types of authorities deployed as Host and Guest through the Enum in java. When the host level methods like kicking guests are used, the whiteboard will check the caller’s authority.

4.3 Handle exception

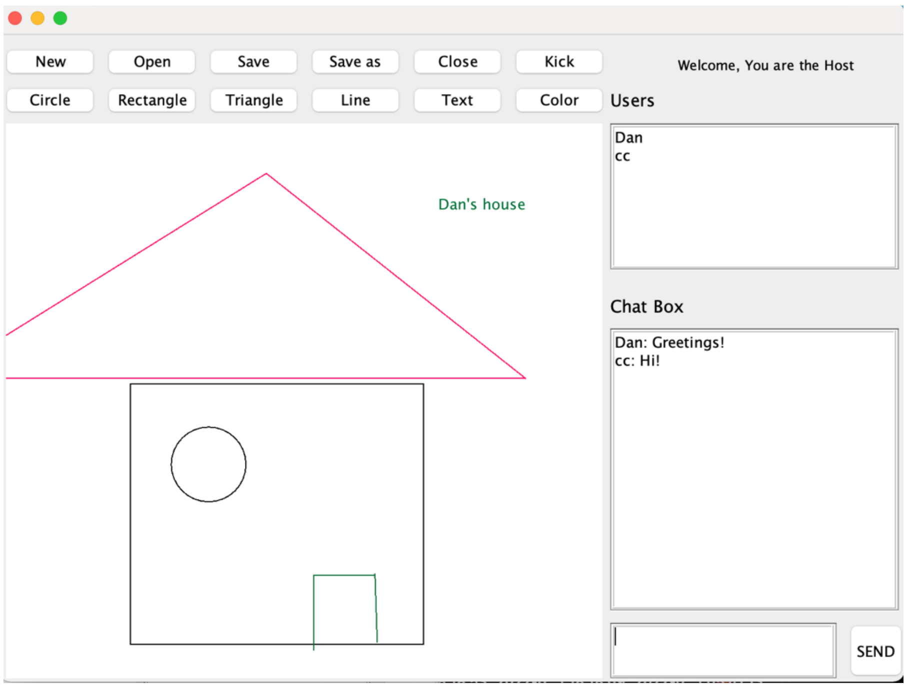
While the application has users, invalid input is inevitable. This White Board application has a complete set of tests to ensure the program is running correctly. The invalid operation and reaction are shown in Table 1.

Invalid Operation	Reply
Wrong port address or port number	Notify the user through pop-up window, and then close the user's program.
Missing username	Generate a default username for the user. 'Host' for host user and 'Guest' for guest user
RMI's Remote Error	Notify the user through pop-up window.
Kick non-existed guest or host (Host User)	Notify the user through pop-up window.
Can not 'save' or 'save as' the file (Host User)	Notify the user through pop-up window.
File not found or file's format can not be loaded	Notify the user through pop-up window.
Host disconnects	All guests receive the 'Host close window' notification through pop-up window and close their program.
Guest disconnects	The guest will be removed from the user list. When the user re-join the server again, its canvas will be synchronized.

Table 1: Invalid Operation and Reaction

4.4 The UIs

The demo of Host UI is shown in Figure 4.

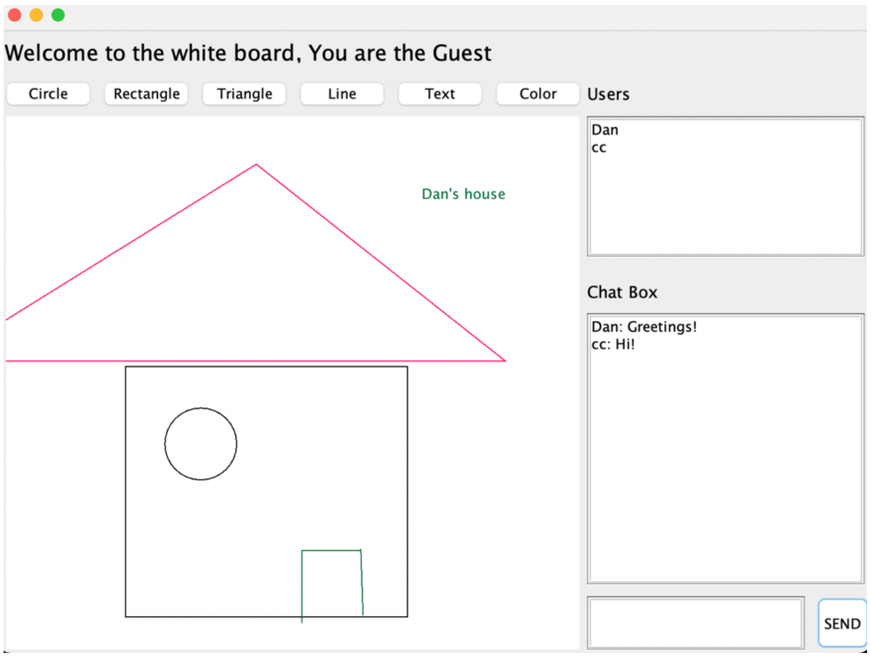


Dan
cc

Dan: Greetings!
cc: Hi!

Figure 4: Host UI demo

The demo of Server UI is shown in Figure 5.



Dan
cc

Dan: Greetings!
cc: Hi!

Figure 5: Guest UI demo

5 Analysis

In this project, the whiteboard server is deployed by the host. For further improvement, it is better to deploy separately. That will provide more fault tolerance when the host is crashed. Moreover, the message is transferred in String format in the chat part. It can be improved as Json format for further development while Json format has better scalability.