Assignment2: distributed shared whiteboard

Student Name: Yutao Zhou
Student number: 1001087
Student Email: yutazhou@student.unimelb.edu.au

Introduction

In this assignment, I will design and implement a distributed shared whiteboard which can allow multiple users to draw and text simultaneously on a canvas. The technology I chose is socket and thread. The system architecture I chose is thread per connection. The socket I chose is TCP, which is more reliable than UDP. I chose JSON as exchange communication protocol between users and manager, so the message format is JSON format as well. I also store the whiteboard in two data types, JPG and JSON format. The exception handling is properly managed on both server and client side. To control concurrent operation of multiple threads, I use CopyOnWriteArrayList, which provides a safe concurrent access mechanism when multiple threads need to read the contents of a list at the same time.

The main System component

The main system component consists of Server Class, Connection Class, Whiteboard Class, Painter Class, and Client Class. The Server Class is used to monitor client connections and implement basic message broadcast functions, which means when server receives the message from the clients or wants to send a message to clients, server will broadcast this message to all clients. The Connection Class is used to handle connections between servers and clients. Client Class is used to connect to and communicate with the server. Whiteboard Class Mainly responsible for the construction of the whiteboard and the functional design of each component. Painter Class is responsible for the logic related to drawing on the whiteboard.

Class design

The figure1 is shown the overall class design for the distributed shared white board.
The first Class is Sever Class. It defines a main method, broadcast method and repost method. The main method is used to create a manager and keep listening whether client connects with server. The broadcast method is used to broadcast a message to all connected clients. The repost method is used to send messages to all connected clients except the specific sender.
The second method is Connection Class. Its main function is to continuously receive JSON exchange messages from the client and perform corresponding processing according to the command fields in the messages, such as DRAW, CHAT and so on. Also, it can send JSON messages to clients as well. For the JSON exchange protocol, I add org.json in pom.xml file and then import org.json.JSONObject in java.
The third Class is Client Class. It defines a main method and send method. The main method is to create a connection with the server, receive JSON messages from the server, and perform corresponding processing according to the command field in the message. The send method is to

send messages to servers. In the main method, commands such as ACK, NAK, KICK, USERS, DRAW, CHAT, NEW_BOARD, etc. are processed according to different command fields. These commands correspond to different interactive operations between the server and the client, such as confirming messages, rejecting messages, kicking out users, updating user lists, drawing operations, chat messages, creating new whiteboards, etc.
The fourth Class is Whiteboard Class. It designs whiteboard UI for the manager and users, which contains the design of all buttons and input boxes. For the save button design, the method will save the drawing record (JSON) to a file and saves the drawn image as a JPEG image. For the load button, the method will read the drawing record from the file and redraws the whiteboard content based on the record.
The fifth Class is Painter Class. It uses Graphics2D to draw graphics of different shapes, including straight lines, circles, ellipses, rectangles, and text, etc. and pass these operations to the manager in the form of JSON message, so that the manager can pass them to everyone, which achieves whiteboard sharing. It defines the draw method, updateAll method and save method. The draw method is used to draw graphics of a specified shape and send the drawn records to the server in the form of JSON. The updateAll method is used with load button in the Whiteboard Class, which can load drawing records (JSON format) to whiteboard. The save method is used with save button in the Whiteboard Class, which stores the whiteboard drawing as JPG.
The last two Class is Constants Class and Mypanel Class. Constants Class is to define a set of constants that are used to pass specific commands between the client and server. Mypanel Class extends JPanel and overrides the paintComponent method to draw all the drawing commands stored in the whiteboard record on the panel.

Figure2 shows the Interaction Diagram for distributed shared white board. Server run as a manager and create the whiteboard of manager. The manager listens and waits for clients to connect. When the client1 requests the connection with the manager, the manager will accept the connection and create a new thread for connection. Then, client1 creates a new whiteboard. The client2 connect with manger as well and create a new whiteboard. When the client1 drawing the circle on his own whiteboard by mouse listener and mouse motion listener, the whiteboard will send a JSON message of drawing a circle, that includes x-axis, y-axis, color and so on. The manager will update his own whiteboard with the JSON message and broadcast this message to all clients except client1, which avoids information repetition. Finally, the clients receive the message and update his whiteboard, which achieve the whiteboard sharing.

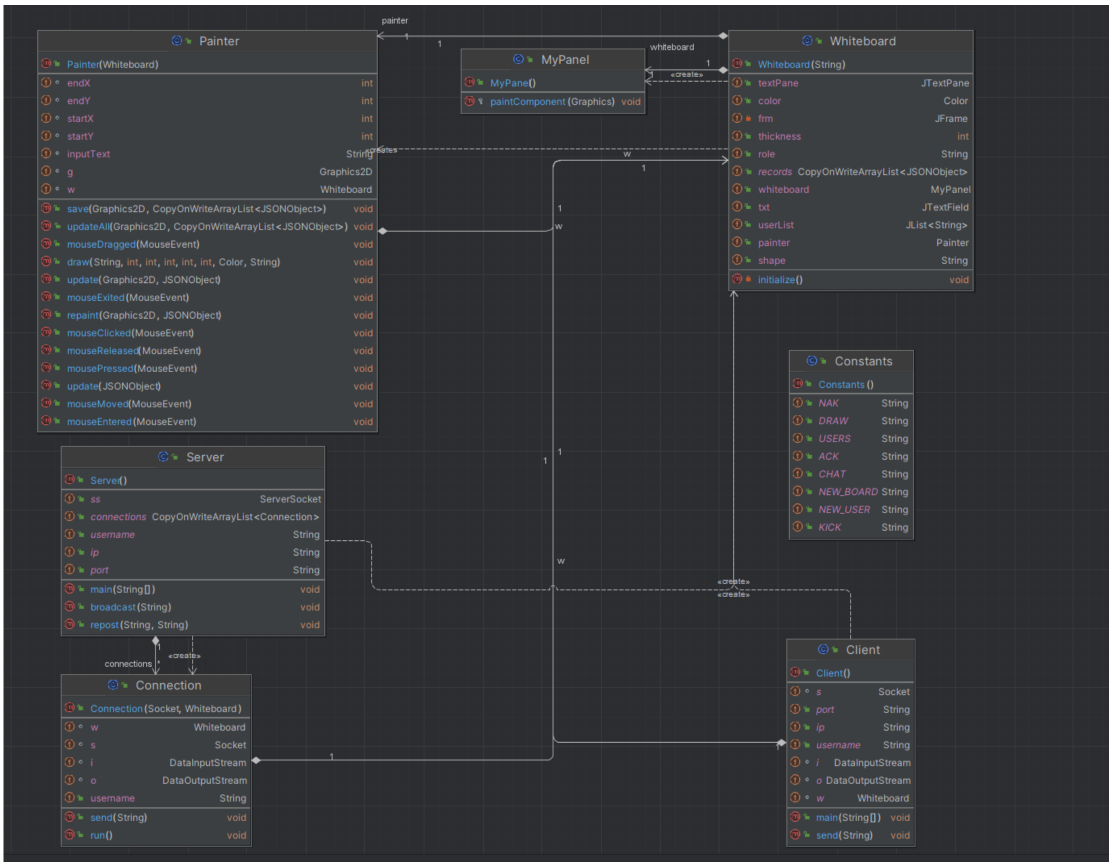

Figure1. the overall class design for distributed shared white board

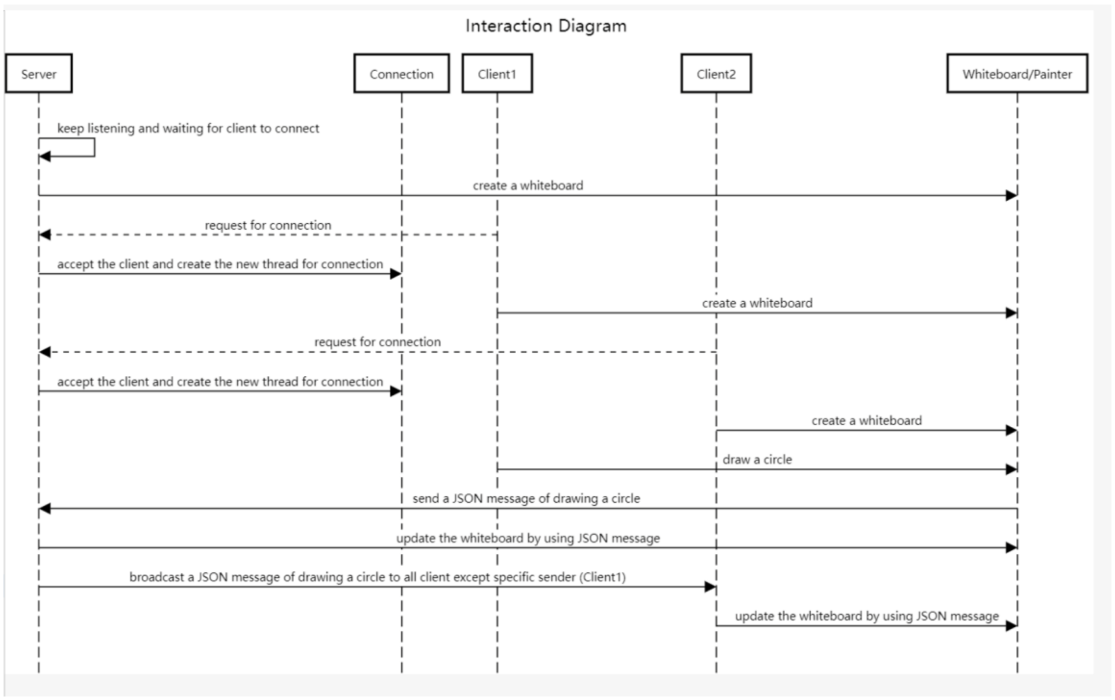
Interaction Diagram
Server
Connection
Client1
Client2
Whiteboard/Painter
keep listening and waiting for client to connect
create a whiteboard
request for connection
accept the client and create the new thread for connection
create a whiteboard
request for connection
accept the client and create the new thread for connection
create a whiteboard
draw a circle
send a JSON message of drawing a circle
update the whiteboard by using JSON message
broadcast a JSON message of drawing a circle to all client except specific sender (Client1)
update the whiteboard by using JSON message

Figure2. Interaction Diagram for distributed shared white board

Critical analysis (Excellence)

In this assignment, I still use thread per connection to realize the interaction between server and client, mainly for two reasons. The first is because thread per connect is simpler to implement and has clearer logic than RMI. The second reason is that for thread per connection, the client's connection is independent, and each connection does not affect each other. The choice of socket is TCP because TCP is more stable than UDP. TCP requires a three-way handshake to establish a connection, and because UDP does not require a connection, packet loss is likely to occur during data transmission. In the choice of communication protocol, I used JSON because the format of JSON is more intuitive and can represent relatively complex data structures, while simple string is less readable, relatively complex when used to represent relatively complex data structures and difficult to design. In handling multi-threaded concurrency control, I used CopyOnWriteArrayList, which can send messages while receiving messages, which is very suitable for multi-person drawing board scenarios.

Creativity

When designing the whiteboard, I used JSlider to adjust the thickness of the lines. When designing line colors, I used JColorChooser to select all colors of lines and fonts, not just 16 colors.

Conclusion

The distributed shared white board is designed with TCP sockets, Thread-per-connection architecture, JSON communication protocol. It mainly supports multiple users to use whiteboard at the same time, share the screen, chat, etc. It allows the manager to complete the new, saving, saving as, closing, and opening of whiteboard. In the future, I would like to add more functions, like operation history, voting and so on.