Assignment 2

Distributed Shared White Board

Student name: Yuanhang Liu

Student ID: 1200403

1 PROJECT GOAL

This project aims to implement a shared whiteboards server application that allows multiple users to perform basic drawings and inputting text messages on the shared paint area and communicate through the chat area from the client application. The first user acts as a manager and can kick other users out and perform file management. The project is developed using a client-server model, enabling the server to broadcast the newest paint on the painting area to all clients. Socket and thread technologies are the two fundamental technologies used in the project implementation for network communication and concurrency. Java programming language is used in project implementation.

2 ARCHITECTURES

2.1 CLIENT-SERVER ARCHITECTURE

The project was implemented using a client-server architecture. It allows the server to act as a broadcast station. So that no matter which client has performed any actions on the shared painting area, the action will be broadcast to every joined client to achieve the same action on the painting area, including the server. Whenever a new client has joined the server, the server will share the latest state of the shared painting area. By doing so, the concurrency state of the shared painting area can be achieved.

Figure 1. illustrates the Client-Server architecture described above:

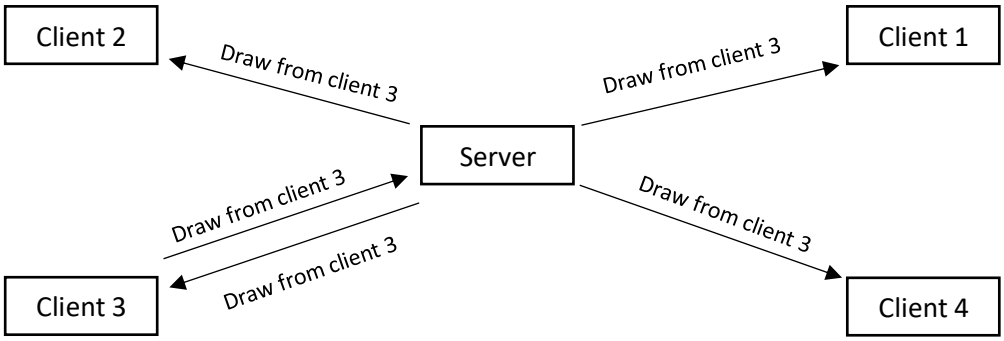

Figure 1. Client-server architecture dealing with drawing command

2.2 THREAD-PER-CONNECTION MULTI-THREAD SERVER ARCHITECTURE

This project multi-threaded server was implemented using thread-per-connection architecture. The decision of choosing this architecture is based on the successful implementation from the previous project. Stable communications between the server and clients can be guaranteed via establishing communication channels using TCP protocol. However, there is one major implementation difference between this project to the previous one.

The previous project only requires establishing one thread per connected client for communication, which is insufficient to handle the necessary message exchanges. It cannot support broadcast functionality from the server to all other connected clients. For this reason, the server application itself is implemented as one thread. As a result, every thread which handles per connected client can access the client thread list stored in the server thread to broadcast the message to all connected clients.

Figure 2. shows the thread-per-connection architecture used in the project:

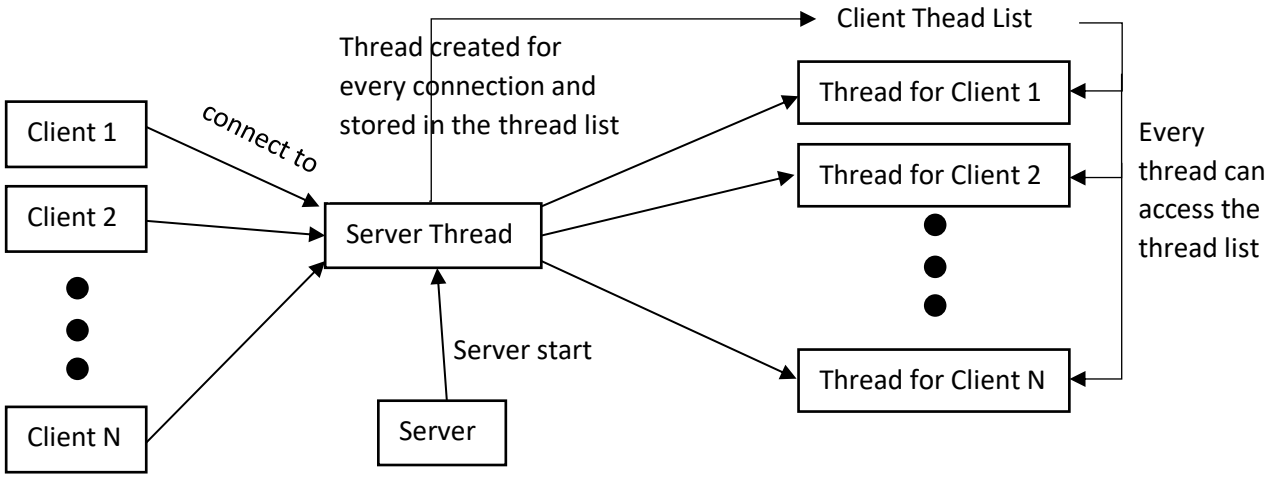

Figure 2. Thread-per-connection for the project

3 COMMUNICATION PROTOCOLS AND MESSAGE FORMATS:

All communications between server and clients are implemented using TCP sockets in this project. It provides a reliable communication mechanism to ensure every message transmission being delivered to another party successfully.

Message exchange protocol is mainly based on an external library called org.json[1], which allows message exchange between client and server to be done using JSON format. Furthermore, org.json library provides the functionality for converting, reading and writing from Strings format to JSON Object, which is also used in the dictionary library file manipulation.

As for the image transfer between the server and the newly joined user, the ImageIO class converts image format to a ByteArray format to transfer the image data from the server to the client. When the client has received the image in ByteArray format, the ImageIO class is able to convert the ByteArray formatted image back to the actual image format to display in the client application.

3.1 COMMAND TRANSMISSION:

JSON Object is mapped as below for message communications:

JSONObject message = new JSONObject();

message.put(queryCMD);

This is the command of the message. It includes the below types for initial communications:

- chat: The transmission is in regards to the chat messages
- quit: A client has quit the application
- logout: It is delivered to all the clients, notifies them which user has disconnected
- draw: It notifies all clients and the server that a drawing is performed.
- login: It notifies all clients and the server that a client has joined the shared whiteboard. It will only be delivered to all other clients until the server approves the login request.
- error: It notifies the client if there are any errors, such as using the same user name as the existing connected user.
- load: It notifies the newly joined clients to prepare to receive image files which is the current state of the whiteboard paint.

message.put(queryCMDType): It specifies the command type such as “line”, “circle” and “oval”.

message.put(queryColor): It specifies which colour the drawing is performed.

message.put(queryX1,Y1,X2,Y2): It records the drawing start point and endpoint.

Message.put(queryString): It specifies the content of the transmission, such as text string drawn on the whiteboard.

Figure 3. demonstrates how org.json library is used in the project:

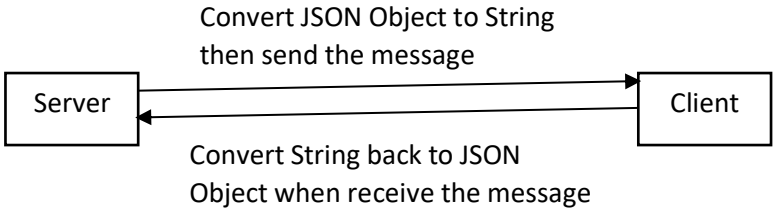

Figure 3. how JSON library is used

3.2 IMAGE TRANSMISSION:

Image transaction is done using the below format:

ImageIO.write(Image "png", ByteArrayOutputStream): Converts image to byte array.

ImageByte = ByteArrayOutputStream.toByteArray(): Convers byte array to byte for socket transimission

output.writeInt(ImageByte.length): Inform the client how long the image bytes are so that the client knows when to stop receive the message.

output.write(ImageByte): Transfer the image data in byte format.

Figure 4. demonstrates how image is transmitted in the project:

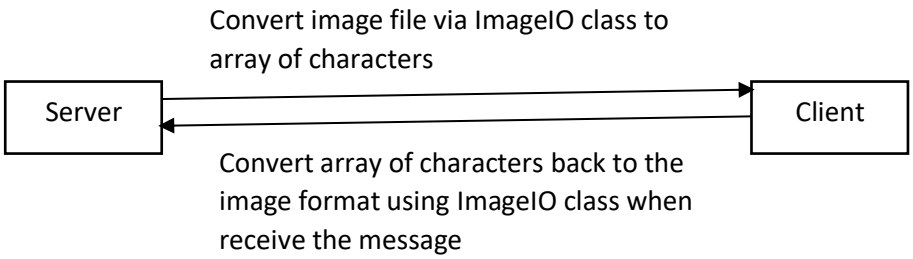

Figure 4. Image transimission

4 FAILURE MODEL:

Potential errors for both client and server-side have been appropriately handled in the project. A pop-up message window will appear and display the error message corresponding to the actual error the program encountered. The errors included but not limited to:

- Console input error, such as wrong parameters passed in command line
- Network communication error, such as socket closed and connection error
- Any other errors may occur runtime for both client and server applications

5 FUNCTIONALITIES AND GUI:

Both server and client are able to perform functions as listed below:

- Drawing
  - Line: It allows the user to draw a straight line on the whiteboard.
  - Circle: It allows the user to draw a circle shape on the whiteboard.
  - Oval: It allows the user to draw an oval shape on the whiteboard.
  - Rectangle: It allows the user to draw a rectangle shape on the whiteboard.
  - Text: It allows the user to insert a string of text anywhere on the whiteboard.
  - Clean: It allows the user to clean the whiteboard.
- Colour selection
  - It allows the user to paint with the selected colour on the whiteboard. The system provides 16 different colours for the user to choose.
- Chat room
  - It allows any user to send messages to connected users and the server. The chat room displays the user name followed by the message that the user has inputted.
- Online user list
  - It displays a connected user list.
- Quit
  - It allows the user to quit the program. The close button on the top right corner performs the same function as in this function.

Server exclusive functions:

- Accept or decline a user from joining the whiteboard.
- Kick a specific user from the online user list.
- File management
    - New: Creates a new file, which cleans the whiteboard.
    - Open: Opens an existing image file for the whiteboard.
    - Save: Saves to the existing image file. It will perform the SaveAs function if no image file is loaded or any image file is saved before the operation.
    - SaveAs: Saves the whiteboard image to a new file.
    - Close: Closes the File management menu.

The server user interface is shown as below: (The client GUI does not have the File and Kick button)

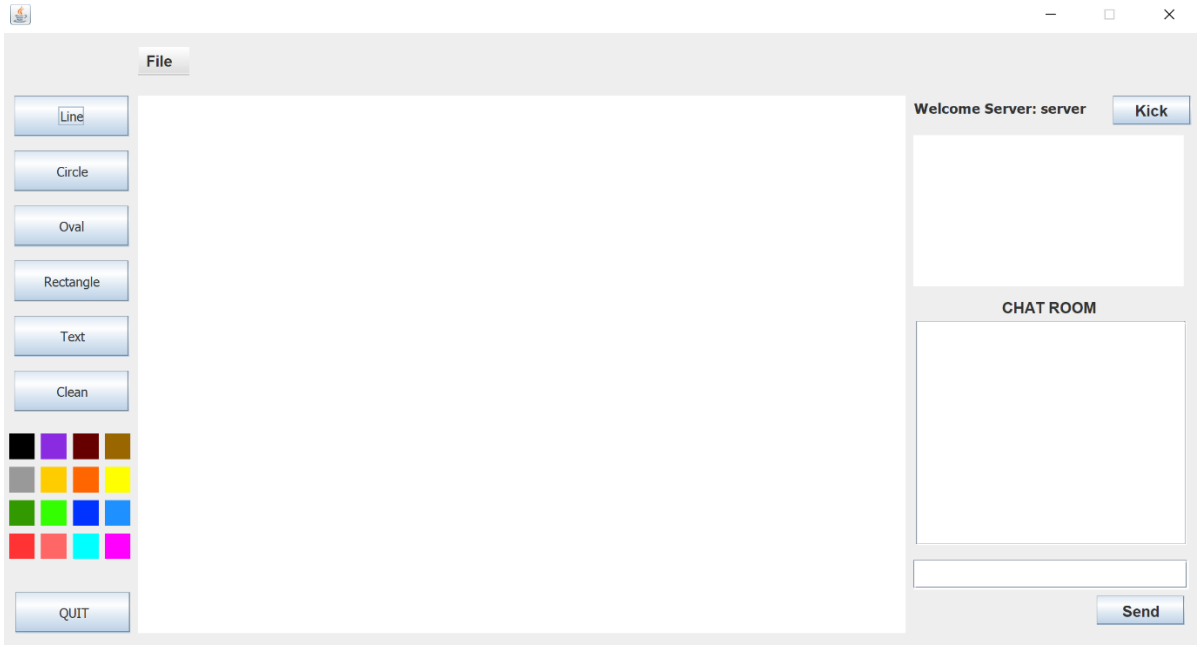

Figure 5. Server GUI design

6 OVERALL CLASS DESIGN, INTERACTION AND IMPLEMENTATION

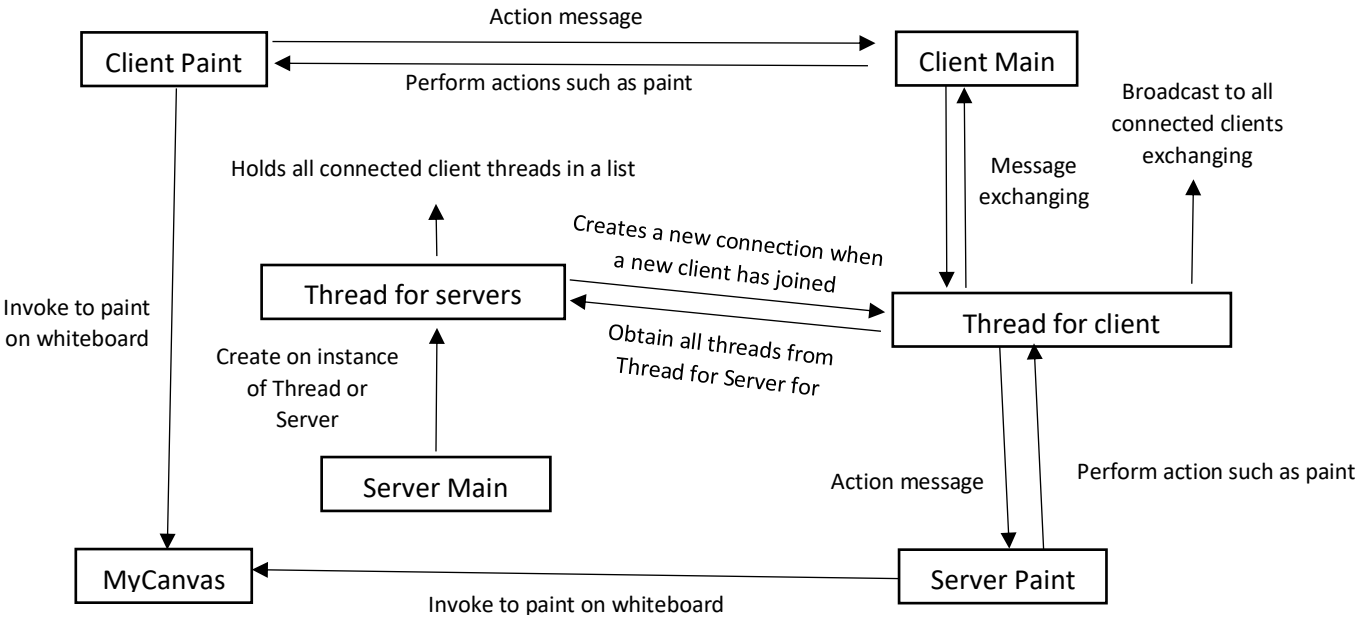

Figure 6. Overall Class design

6.1 SERVER-SIDE PROGRAMMING DESIGN DETAILS:

- ServerMain.java
  - It contains program entry class Main().
  - It creates an instance(a thread) of the ThreadForServers class.
- ThreadforServers.java
  - It runs as the socket server.
  - It holds a list of ThreadForClients. Whenever a client has joined the server, a thread is created for the client, and the thread will be stored in the list. When a client has disconnected from the server, the thread and the client socket connection will end and be removed from the thread list.
  - It holds a list of client names as references for all threads.
  - It contains a list of paint records that records all the paint history from both clients and the server.
- ThreadForClients.java
  - It is the thread implementation for thread-per-connection.
  - It records the user name for each thread. The server is able to identify each different thread via the user name variable.
  - It receives messages from the client and broadcast the message to all other clients by accessing the thread list from ThreadForServers class.
  - It invokes the paint and the chat function from the ServerPaint class once receiving messages from any clients.
  - It transmits the image of the current state of the whiteboard to the new joined client.
- ServerPaint

- It is the GUI implementation on the server-side.
- It handles all the GUI functions from the server-side and broadcast performed actions by invoking TheadForClients class and the thread list from the ThreadForServers class.
- It invokes MyCanvas class to perform all the drawings on the whiteboard.

6.2 CLIENT-SIDE PROGRAMMING DESIGN DETAILS:

- ClientMain.java
  - Contains program entry class Main().
  - It receives the messages and performs paint or chat actions by invoking the ClientPaint class.
- ClientPaint.java
  - It contains the GUI implementation on the client-side.
  - It handles all the GUI functions that performed from the client-side.
  - It messages the client actions to the server.
  - It invokes MyCanvas class to perform all the drawings on the whiteboard.
- MyCanvas.java
  - It contains the paint implementations via extending the Canvas class.

7 CRITICAL ANALYSIS

This project is implemented based on two fundamental architectures, which are client-server architecture and thread-per-connection architecture. By using client-server architecture, the server program acts like a broadcast station, which delivers the messages to all the connected clients. In this project, the server is implemented as a thread which allows the client handling threads to access other threads via the server thread. As a result of doing this, each connection can broadcast to other clients. The thread-per-connection architecture enables the server to handle continuous communications from different clients the service simultaneously.

TCP socket is chosen over other technologies because of the successful implementation using TCP socket from the previous project. It can provide reliable communications between the server and clients. However, it requires extra data handling techniques for this project when dealing with the image data transfer. The initial implementation of image transmission is purely based on ImageIO function as it can transmit and read the image format straight away. Unfortunately, this method does not work using TCP socket as the TCP socket does not know when to stop reading the stream even if the transmission is finished. Closing the stream, on the other hand, will close the socket connection. To handle this issue, a message containing the image array's size is sent prior to the image array transmission so that the receiver can know when to stop reading the stream.

The 'synchronised' keyword is given to all threads to ensure the concurrent state of shared resources.

Using JSON format on message exchanging is mainly because it is easy to be mapped from a dictionary format to JSON format. JSON format has the "dictionary-like" data structure. JSON has the key-value pair, which can be used to store "commands" and "instructions". The JSON format message can be customised to different message length, which means the "instruction" can be configured as flexibly as possible and makes extension implementation a lot easier.

8 CONCLUSION

In summary, this project has implemented a distributed shared whiteboard that allows clients and the server to paint and chat on a single state whiteboard. Client-server and thread-per-connection architecture are used to achieve state sharing functions and message transmissions on the whiteboard application.

9 REFERENCE:

[1]"stleary/JSON-java", GitHub, 2021. [Online]. Available: https://github.com/stleary/JSON-java. [Accessed: 20- May- 2021].