COMP90015 Distributed Systems

Assignment 2 Shared White Board

Name: Shiqiang REN

Table of Contents

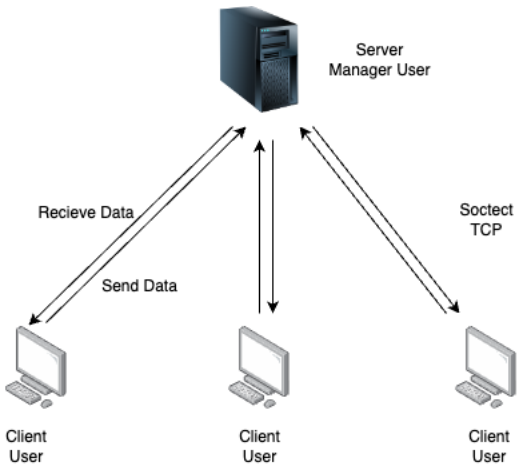
1 Project Description

This project is a distributed whiteboard application. The application allows all users to draw at the same time as handwriting and certain shapes including line, circle, oval, and rectangle. Also, users can type text on the board and select different colour to draw. In addition, the application offers some managing features for the manager user. A File menu and a remove button that allow the manage to kick out a user is provided to the manager user. Lastly, all users can chat with each other through the application.

2 The components of the system design.

In order to implement the features described above, this project has chosen design and technologies as following:

2.1 System architecture



The project adopts client-server distributed architecture, which server also plays as a manager user role. Java sockets are used to build the connection between the server and clients. To ensure the reliability of the drawing board data transmission, the project adopts TCP protocol. TCP protocol monitors the data transmitted and have the abilities to handle the errors that can possibly lead to the data inconsistency. As a result, the consistency of the drawing board data is achieved by connecting all users by sockets TCP.

In addition, JSON format data is used as message format. Compare with directly transmit the board data as picture format like PNG, JSON can better ensure that users get consistent data. Apart from this, JSON is smaller, well-performing and easy to extend and read.

2.2 Handling of concurrency

When a distributed system has a very large number of users, parallel design can improve the performance of the system. In this project, both the server and the client use a multi-threaded approach to send requests or receive the responds.

For the server (manager user), the main thread is handling to UI including the drawing board. A sub-thread messaging is handling to broadcast the data like white board, users list and chatting history to all users. When a general user connects with the sever, two connection sub-threads are created for sending data to the user and listening the data sent from the user.

For the client (general user), similar to the server, the UI thread is the main thread and the entry point to the program. Apart from this, there also have two connection sub-threads to handling the request data and respond data.

2.3 Other implementation details

Thread safety

As the project uses a multi-threaded concurrent design, the data safety issues of shared data between different threads need to be solved. In order to reduce the developing difficulty and complexity of the code, The data in the project with regard to data safety issues are all taken from the Java JUC package. The JUC's data structures are already thread-safe, which avoids using lock mechanisms such as the synchronized keyword to ensure security by self. For example, the project uses the thread safe HashMap (ConcurrentHashMap), List (CopyOnWriteArrayList) and Queue (LinkedBlockingQueue).

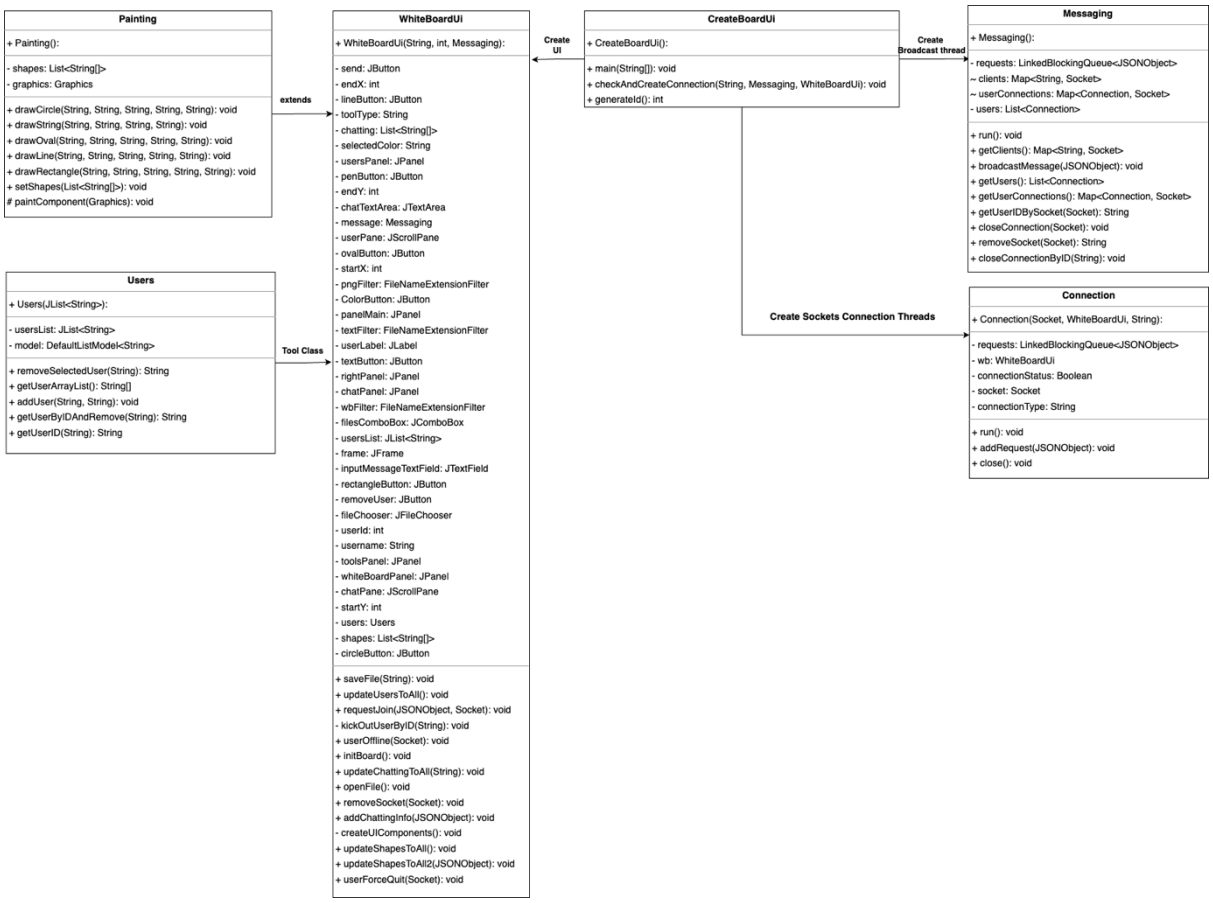
Transmission data

When a new user joins the whiteboard, previous chats are not synchronised and only the user's newly entered messages are transmitted, which reduces transmission pressure on the network.

The user list is only maintained by the server and the latest list information is broadcast to other users, as only the manager has the relevant authorisation.

3 Class design and interaction diagram

3.1 Server class design

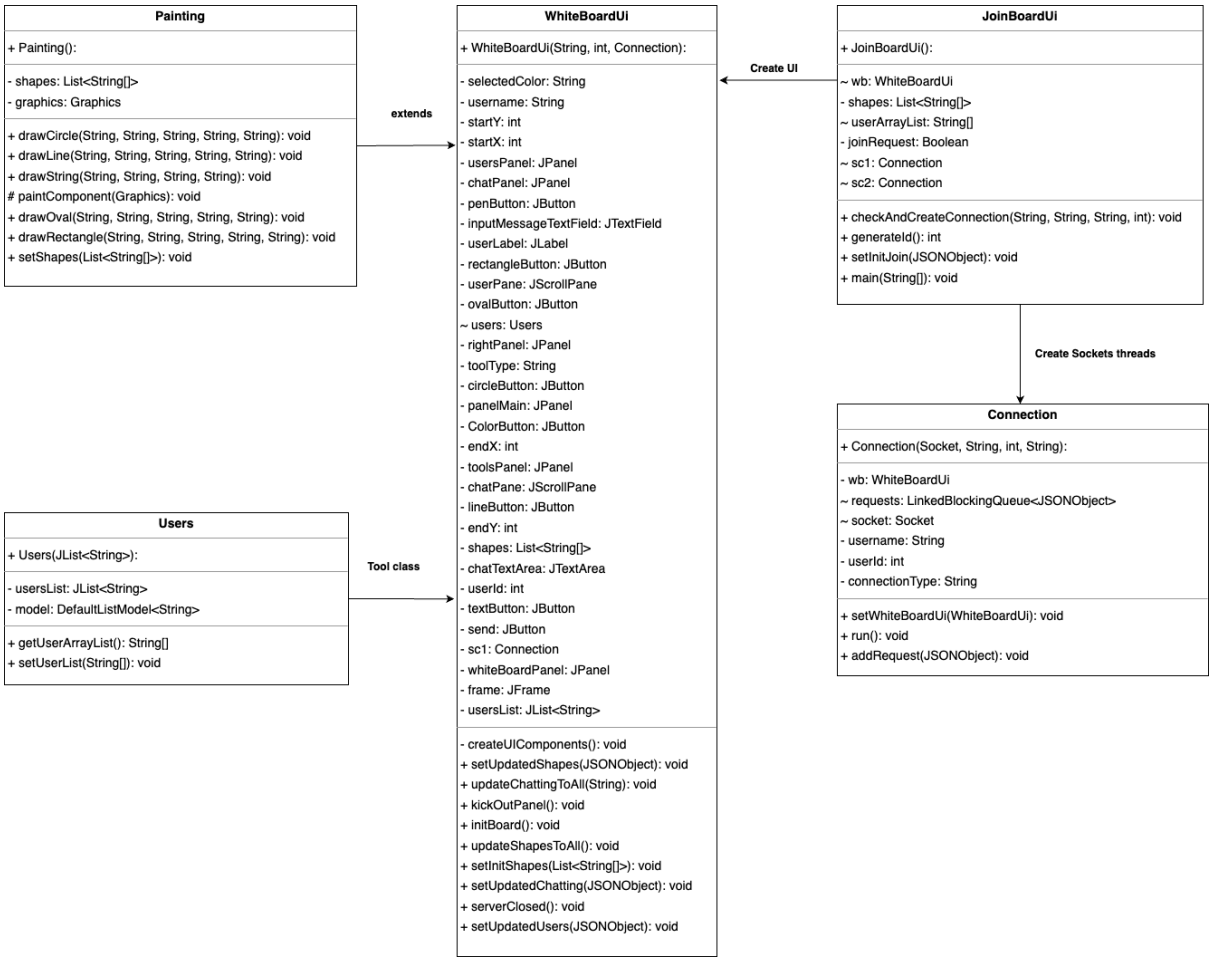


CreateBoardUi is the main thread and the entry point for the program, used to collect the manager's name and create a manager ID, then create the UI (WhiteBoardUi) object and listen to the user's sockets connection. In addition, it also creates Messaging thread to waiting for the data that need to broadcast to all users.

When the program UI is created, WhiteBoardUi object is created as well as its inherited classe (Painting Class) object. WhiteBoardUi handles listening and responding to a range of events, such as mouse dragging and button clicking. And it is also responsible for rendering data to the UI etc. Painting only handles drawing-related events.

When the user built the sockets connection between it and the server, two Connection sub-threads are created, one to listen data from the user and one to send data to the user. Lastly, the Users class is a tool class, which is handle the user list content.

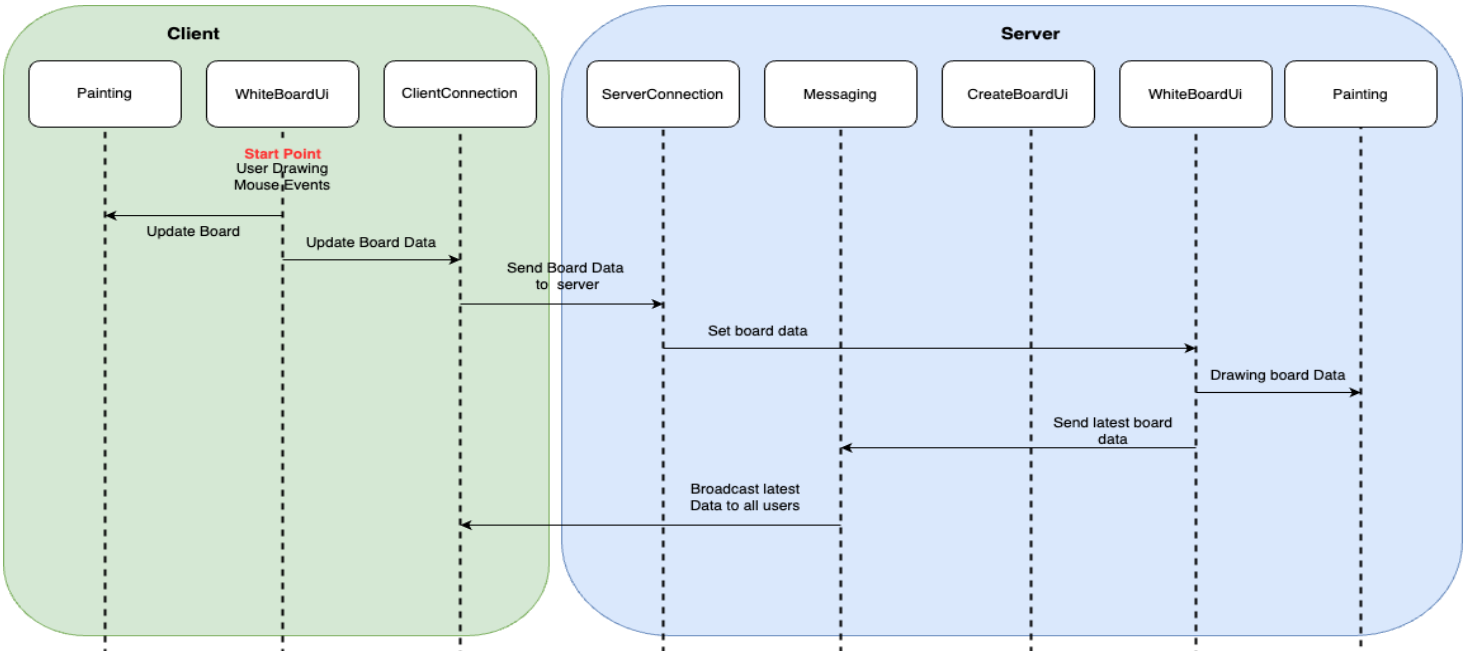
3.2 Client class design



Client classes are very similar to the server classes. JoinBoardUi is the main thread and the entry point for the program. other differences between client classes and server classes are as following: The client does not need to send data to all other clients, only to the server, so there is no Messaging class. Moreover, client's WhiteBoardUi does not include UI and functions related to removing users and File features.

3.3 Interaction diagram

Scenario 1 - User draws on the board

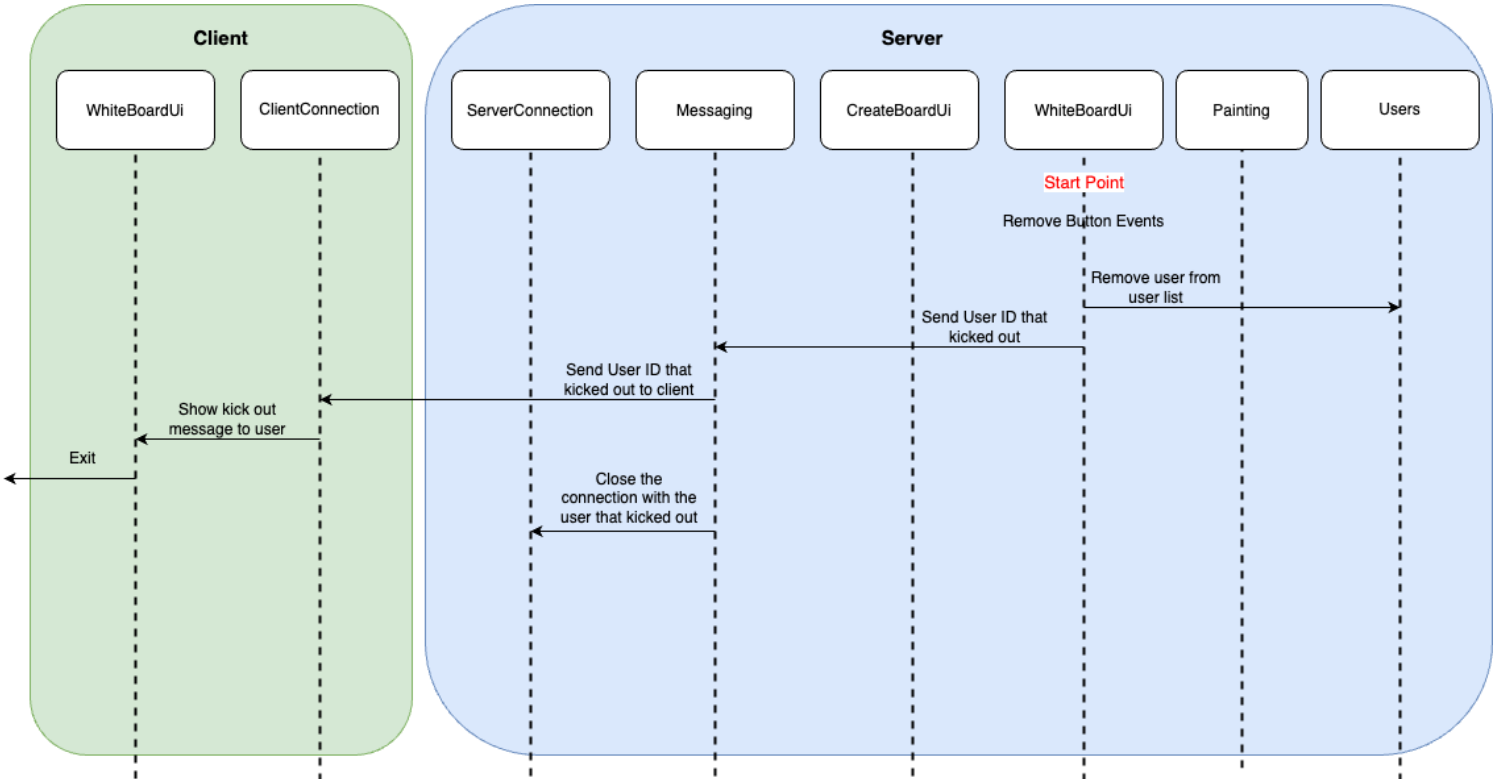


When client's WhiteBoardUi listened to the user's mouse drag event, it will add the coordinate data to the whiteboard data Shapes. After the data has been updated, on the one hand it will be transferred to Painting Class to draw the graphic onto the whiteboard. On the other hand, the latest data will be sent to the server through the socket's connection.

Server's WhiteBoardUi receives the latest whiteboard data from the user and sends it to Server's Painting to draw on its own whiteboard, and then broadcast this data through Messaging Class to all other users.

As a result, all other users receive the updated whiteboard data and draw it to their own whiteboard, and therefore the whiteboard data is synchronised among all users.

Scenario 2 - Manager kicks out one user



When server's WhiteBoardUi listened to the user's button click event, it will get the Id of selected user and send this data to Users Class that remove the relevant user from the list. At the same time, the user Id that need to be removed also send to the Messaging Class and broadcast it all users and then the server will close the removed user's Connections.

When user's Connection Class get the user data that need to be kicked out, it will check if the ID matches its own. If that so, the client will show kicking out message and exit. If not, the client will ignore it.

4 Analysis

The project is a whiteboard application based on a distributed system, which implements client-server architecture. Manager server as the central node of the system can better manage and handle the requests and resources of the clients. This allows to ensure the consistency of data, such as whiteboard drawing data across all users. In addition, the central node ensures a higher level of security and provides permission authentication functions such as approving joining and kicking out users. Regarding to the communication between the server and clients, project adopts TCP protocol to ensure the data is highly reliable and consistent. The server and clients build connections by the sockets which ensure reliable data transmission and is easy for development.

On the other hand, the current project implementation may not be beneficial to form clusters of servers and thus make the system highly available. This is because the server also integrates managers-related features. It is possible to take this part of the functionality away from the server and treat the manager as a client rather than a server. Alternatively, the project may use a P2P architecture where each user has the ability to act as a server as well as a client. However, this option may also result in a single node being too complex.

In summary, the project is a whiteboard application which all users can draw on the board and chat with each other. The project adopts client-server architecture and is based on Java sockets and TCP protocol. The data transmission is highly reliable and the data among all users is consistent. However, the project can continue to extract manager's features in the future and therefore make the whole system highly available by forming clusters.