Assignment 2 – Distributed White Board

System architecture

To implement the distributed white board application, a central server architecture was used to maintain a consistent state of the white board which is shared between users of the white board. The system is consisted of two parts, the client application and the server:

The server

The server is used to host sessions of white board (**the server is not the manager**) that can be shared among users. The server maintains a list of sessions that is currently active and can be joined, once a session is finished (the host closes the white board), it is removed from the session list. Each session keeps track of the host and clients in the session, only host can perform privileged operations such as kick a client, open a new white board, close the session.

Client application

The client application can connect to the server with a client specified username, after connecting to the server, the client can see a list of sessions that are active on the server. The client can choose to join one of them and become a participant, or create a room and become the host/manager of a new white board session. After the client chooses one of the above options, the client application will display the white board that is shared for this white board session. All clients in the same session will see the same white board.

Communication protocols and message formats

The server and the client application use socket to communicate with each other. The server opens a server socket that is bind to the port specified by the command line argument. For non-persistent request such as retrieving a list of sessions active on the server, the clients and server will close the connection(socket) once request is responded by the server. For a request related to a white board session, the server and client application will keep the connection(socket) open until either the client leaves the session, or the host closes the session. JSON formatted string is used to communicate between the client application and the server, messages are formatted in the following ways:

For **non-persistent** requests the server needs to reply with a response, they have the following format:

```
// A request to retreive all rooms/sessions in the server
{
    "operation": "Rooms",
}
// The response from the server
{
    "operation": "Rooms",
    "status": "success",
    "rooms": [{
        "roomId" : 0,
        "host": "name"
    }] // List of rooms
}
```

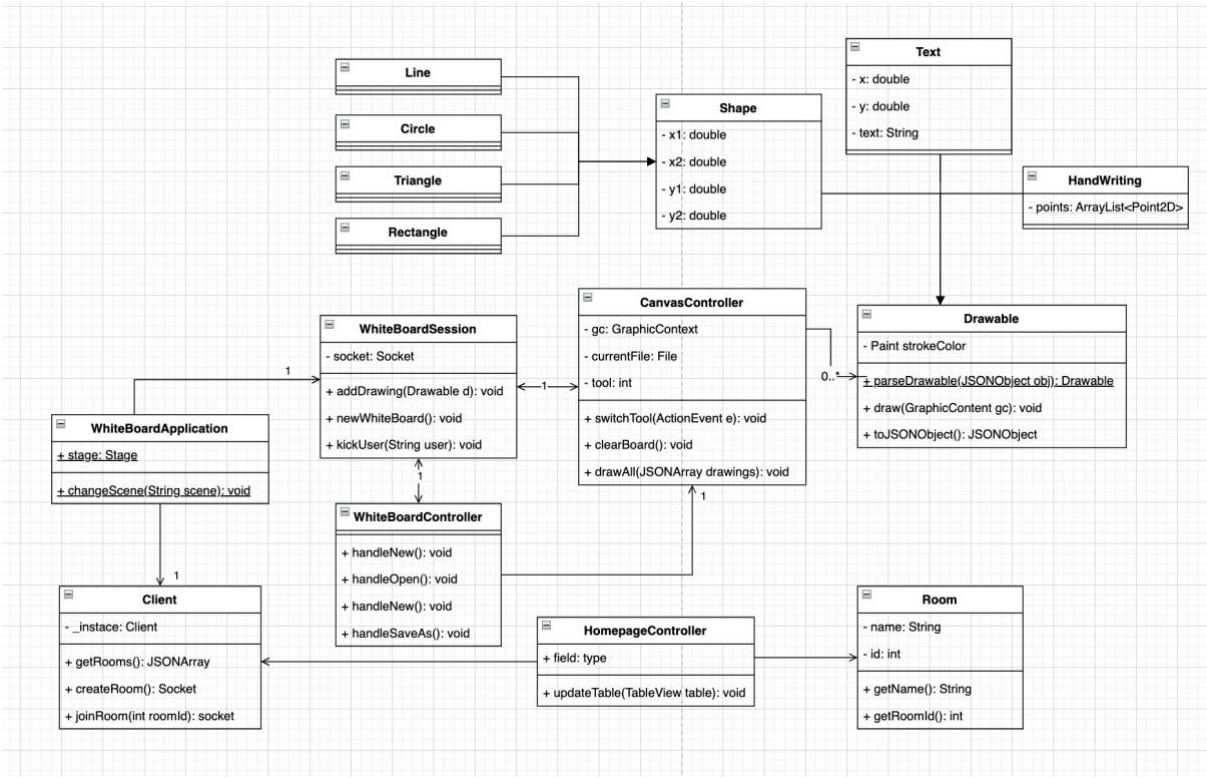
The messages used in the **persistent** connection is slightly different, the server and the client only send message to each other, and the other side do not need to reply the message. Each message has the following format:

```
// A message to draw an object
{
    "operation": "Draw",
    "drawing": "drawable object"
}
```

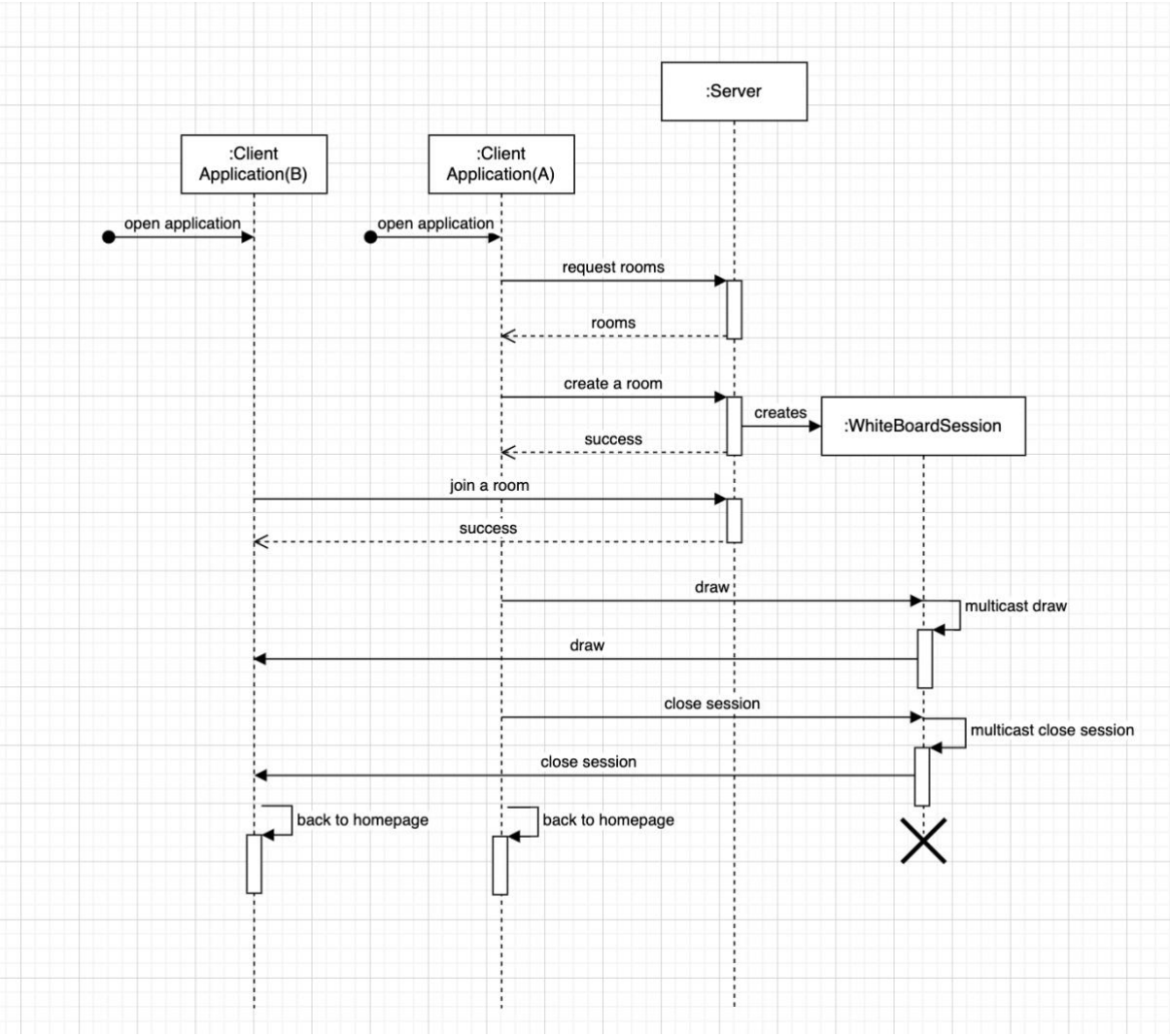
```
// A message to send a chat
{
    "operation": "Chat",
    "clientName": "name",
    "text": "text",
}
```

Design diagrams (class and interaction)

The diagram below is the UML diagram for the Client application, the entry point for the client program is at the WhiteBoardApplciation class, it creates the stage (a javafx window) that displays all the UI elements, it also maintain the session which is connected to the server.



The picture below demonstrates how the client application interacts with the server, a session object is instantiated on a create white board request and destroyed after a close session message, the main purpose of the session object is for management and multicast purpose (detailed explanation in later section).



Implementation details

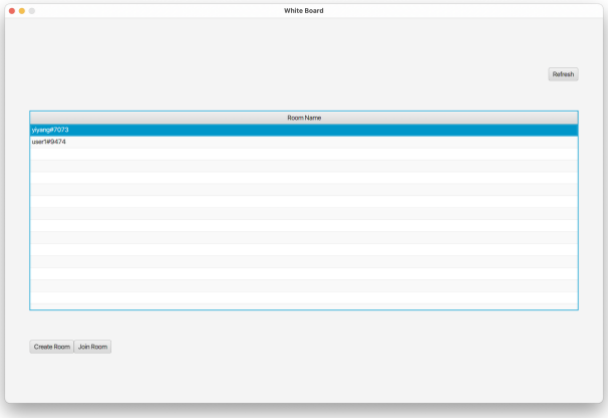
The server

The main thread of the server accepts incoming request from the clients, it is used for the clients to retrieve room lists, create room, join room. Once the client has joined a session, it is handled by a WhiteBoardSession object, it runs on a separate thread and has the following functionality:

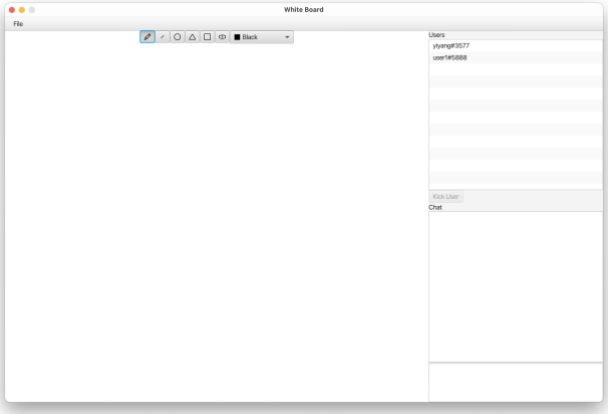
1. Maintain the current state of the white board for this session
2. Maintain a list of clients in the current session, and the host of the session
3. Maintain a list of clients that wants to join the session (not accepted by the host)
4. Once a client joins the session, session object will notify that client the current state of the white board
5. Once any client performs a normal operation (draw, chat), it will be multicast to other clients in the session
6. Once the host performs a privileged operation (kick, new white board, open white board, close session), it's effect will take place in the session object and multicast to other clients
7. Once clients in the session leaves(kicked or closed application) or joins the session, the session object will also multicast a message to notify an update on the users in the session

The client application

The client application uses JavaFx to develop the GUI, it has a homepage which displays all the white board rooms that are currently open (see picture below). Client can join one of the rooms that is currently open. Client can also create their own room and become the manager.

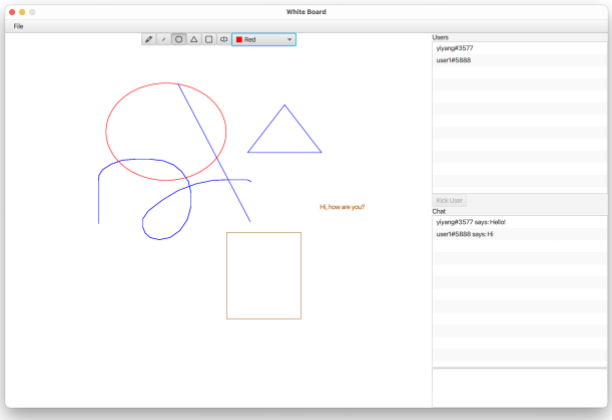


After join/create room, client can see the white board on the left, a side panel which contains a list of users and a chat box on the right.

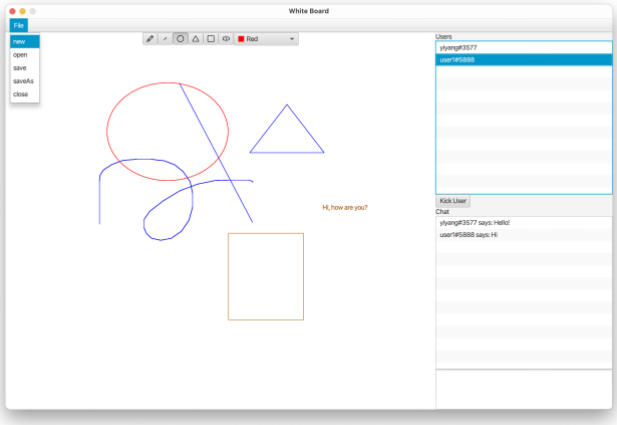


(The interface for host)

Clients can draw or enter text into the chat, it will be shared to other clients in the room, not that the drawings are consistent for all clients but the chat will only display chat messages that are sent after client joins the room.



And the host can perform previlleged operation such as kick user, or have access to the file option menu(new, open, save, saveAs)



There are other UI elements such as dialog for host to accept the user, dialog for session closed notification and a restricted version of the white board for participants (no privileged operation).

New innovations

Hand writing

The white board allows user to draw freely, not restricted to those pre-defined shapes, it is a common functionality that most white board application provides.

Multiple White Boards

Use the approach that the server maintains list of sessions, there can be multiple sessions open at the same time, thus there can be multiple white boards shared among different groups of people.

Hosting behind the NAT

If the server is hosted on a public IP address, such that everyone can access to the server. Then everyone can host a session easily, simply create a room and it can be join by other people across the internet. Without the central server approach, it would be hard for people behind the NAT (no public IP address) to share the white board to other clients, because others may not be able to access the host's IP address.