# COMP90015 Assignment2

**Hongzhuan Zhu 1223535**

## System architecture

The structure of the system is the Client-Server structure,

The Server can mainly process threads per connection requests to complete requests from clients, including drawing requests, joining requests, viewing user requests, and chatting requests. The parsing of the command of the drawing board is also completed on the server-side and then updated(broadcasted) to all clients connected to the server.

In the client, it is mainly used to record local operations, and communicate messages to the server through each established connection, and in the client, it has the function of processing response messages to process those commands sent by the server and then update the UI.

Logically, the Manager component should belong to the Clinic side, because it can be considered as a Client with higher authority, but from the perspective of implementation and the tasks it undertakes, it belongs to the Server side, because it needs to process the request from the client and send a receipt to the client.

## Communication protocols

The main protocol implemented is TCP, which provides reliable communication between the client and server. The connection is completed through the socket, and a thread-per-connection is used to establish a 'tunnel' to complete the transmission of the command. The thread-per-connection will not use a new thread to handle a new request, only one thread to handle all requests based on this connection,  it can reduce the cost of thread across multiple requests from the same client process, therefore, for a long time duration connection, its performance is better.

## Message formats

The message exchange between client and server is consistent with two-part
**Command: Content** where '**:**' is a delimiter for the whole message and '**^**' is the terminator for drawing command.

Both sides will read the command and based on that process the content part.

Below is an example of the common command used in the program.

| Command: | Environment | Purpose: |
|---|---|---|
| begin | The first time to launch the client board | To send begin command to Server to get all users on the |

| | | |
|---|---|---|
| | | server. |
| feedback | After when user click-connection button | Check whether the username existed. |
| over | Close the window or kick by manager | To close the connection |
| request | When client request connection | To send request command to server let server check whether there is the same username |
| draw | Mouse action capture related to the drawing | To broadcast draw information between client and |
| updateUser | When a new client joins in | To update user list |
| clientout | When client left or removed | To update user list |
| postchat | When a client lick sends button | To broadcast chat content |
| new | When the manager click new or open | To clear canvas |

### Implementation details

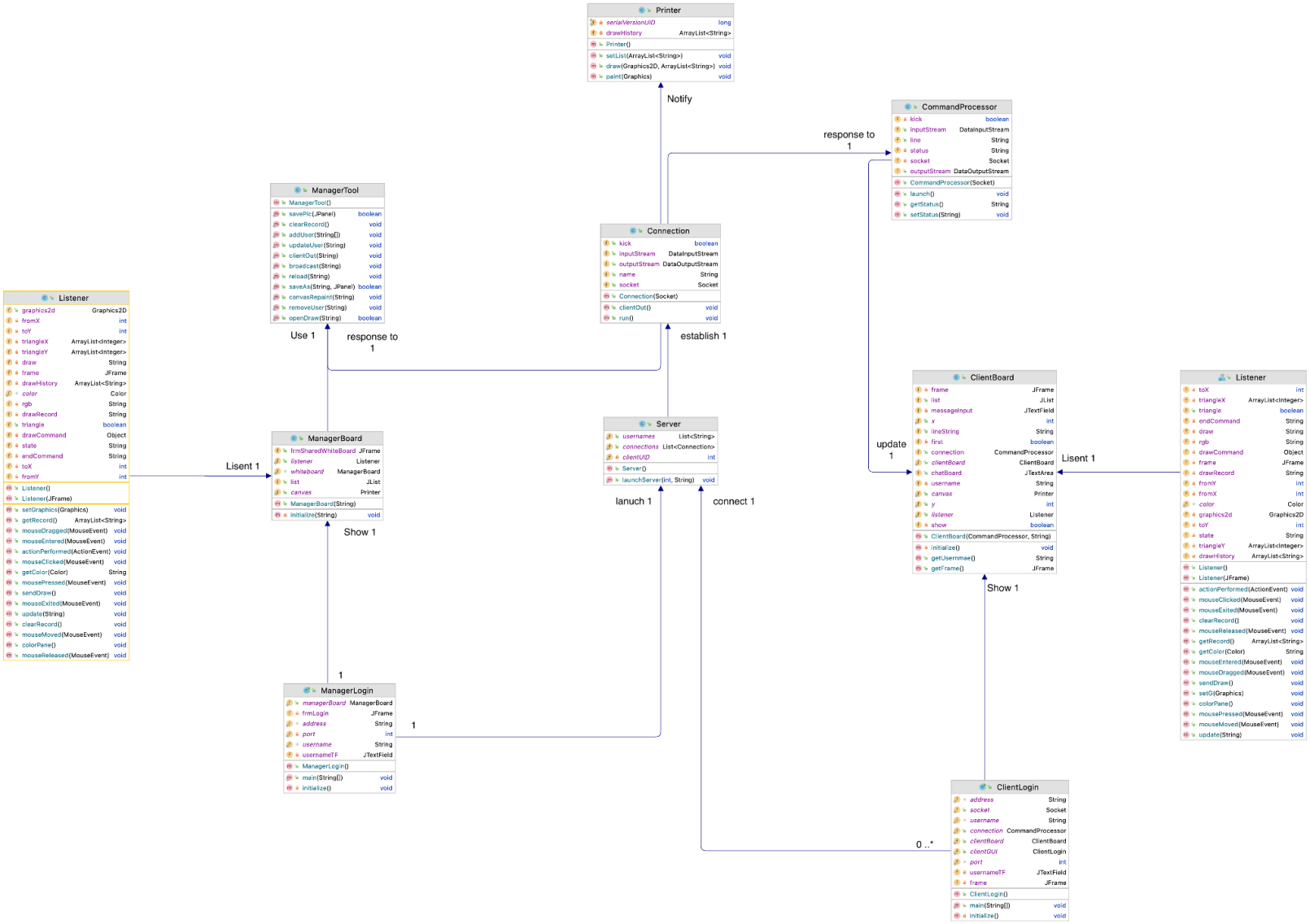
In terms of connectivity, the server will be automatically started when the Manager Login is started, and a new Connection will be created as a tunnel when the Client joins. Each end will have the mechanism to read incoming and write output and update related UI and files according to the command.

In terms of painting update, the captured mouse action and the current state are recorded every time by the listener and added to the drawHistrocy list and the record file. In other clients, the repaint in the Printer is called again, where the paint function has already been overridden. So in this way, a real-time update of the canvas is ensured. When a new user joins, all the contents of the draw record in the record file will be obtained and redrawn.

### New innovations

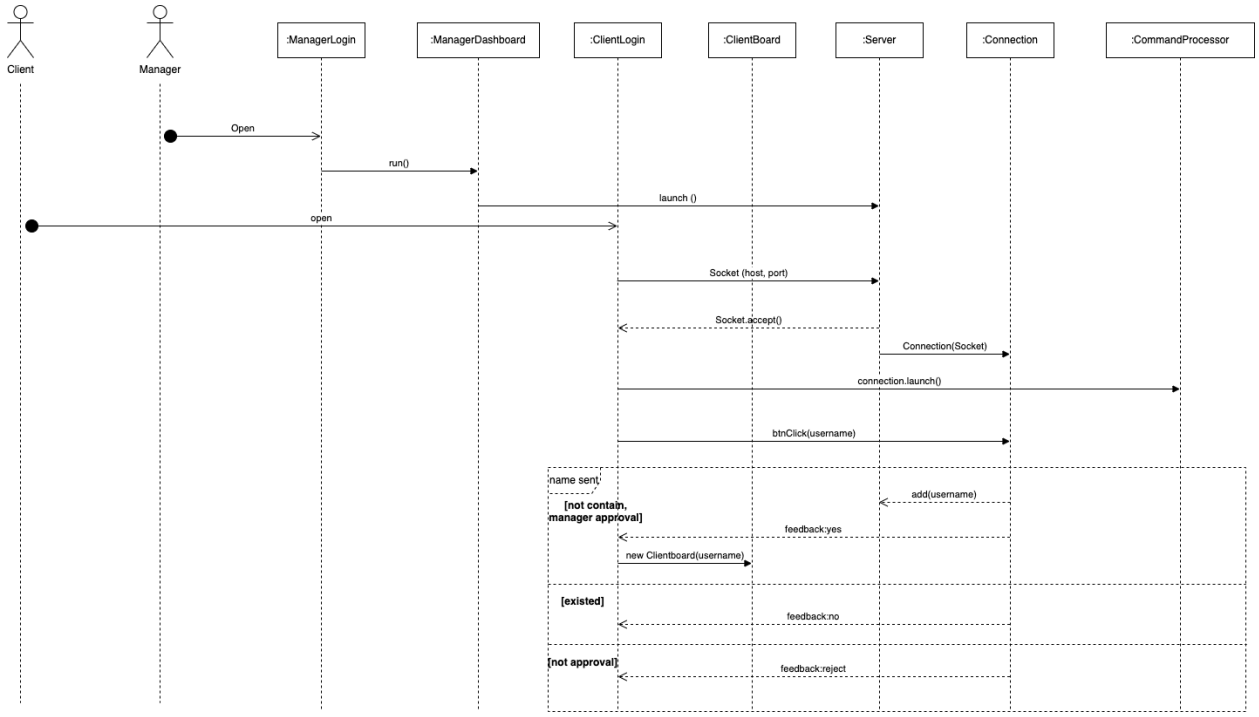
1. The Chat board, which provides the inference allow all client to chat.
2. Client Management, which allows the manager to kick the user
3. File System, which allows the manager to open, save, new, and save as canvas.

# Class diagrams



## Main Interaction diagram

### Login and reload canvas




### Draw and update