Zeyu Wang

1105913

COMP90015: Distributed Systems – Project 2

Distributed Shared White Board

1. **Introduction**

The task of this report is to design a distributed shared whiteboard. It has a login interface and a server. It allows multiple users to draw canvas at the same time. The drawing method can be different graphics, colors and curves. All whiteboard operations can be shared among different clients to realize information interaction. The program includes three GUI components-client whiteboard, client login interface and server.

2. **System Architecture**

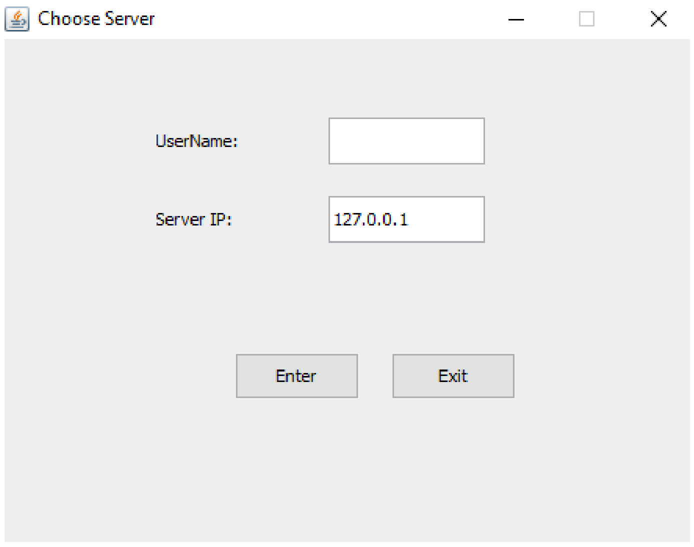
The whiteboard establishes client connections based on TCP Socket and Thread-per-Connection. A new thread is created every time it is transmitted. The transmission data uses JFrame encapsulated objects for direct object transmission. Every client will have an event listener . When the program starts, the TCP connection is created. The client will dynamically generate a request port at the software layer . It allows other client programs to connect for operations such as login, logout, data exchange, etc. The TCP/IP protocol stipulates that the port number of the receiving end remains unchanged. The port number of the sender is randomly generated when the TCP protocol message is sent for the first time. Server is used as an intermediate service provider to inherit two sessions. If the session is disconnected, TCP will automatically send a message to end the TCP link protocol.

3. Program Design

(1). GUI

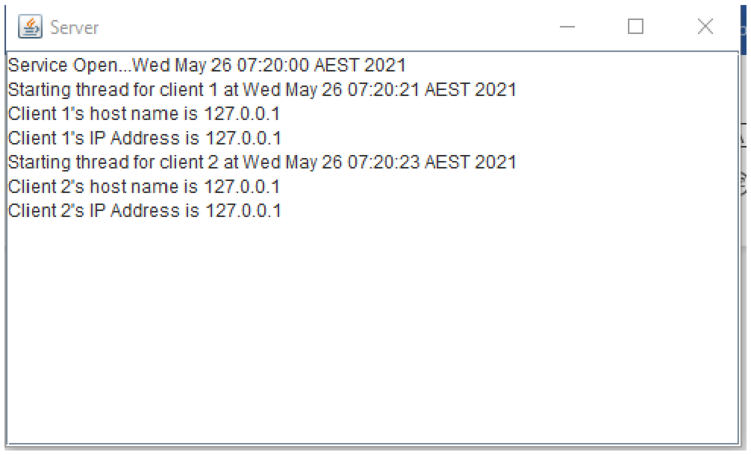
The program includes three GUI components-client whiteboard, client login interface and server. The GUI and drawing board of the project are realized through JavaFX Scene Builder. The whiteboard is divided into two projects. They are server and client. In addition, whenever the client performs an operation, a behavior report is fed back to the client's console. As shown below.

Login



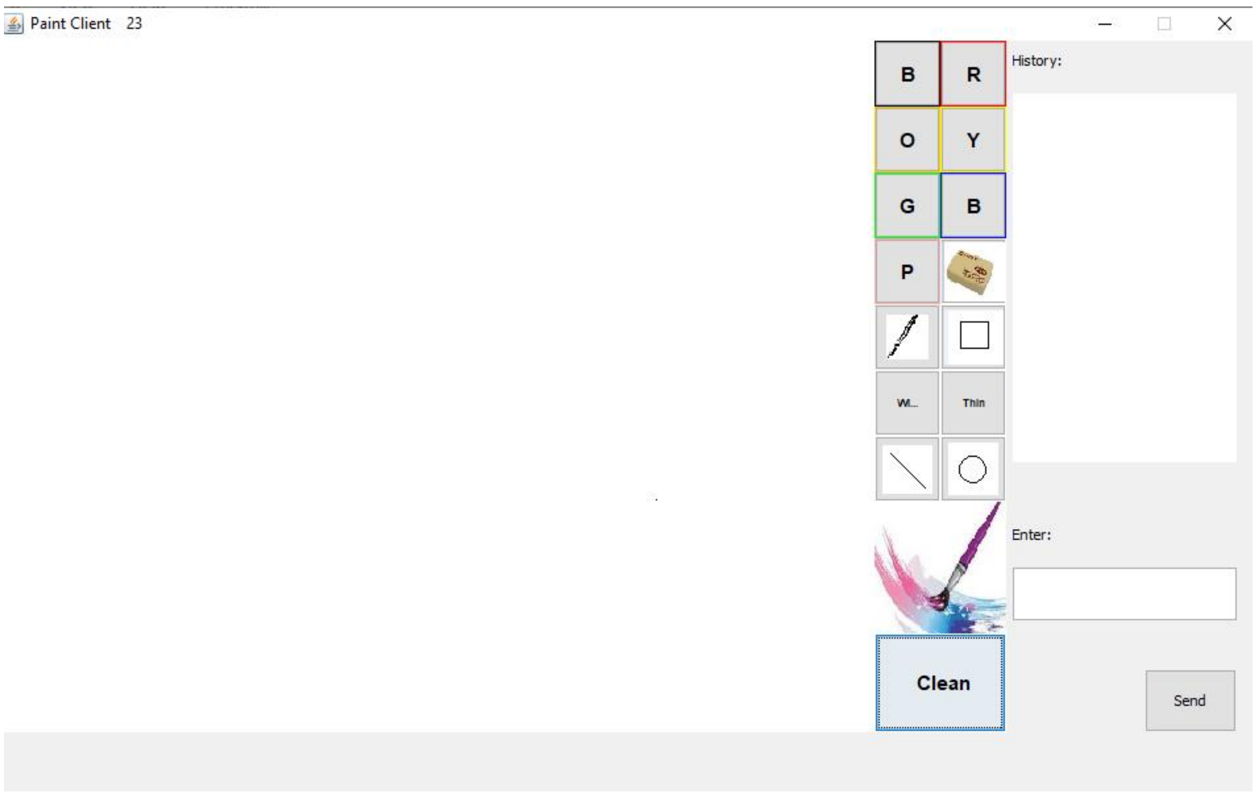
127.0.0.1

Server

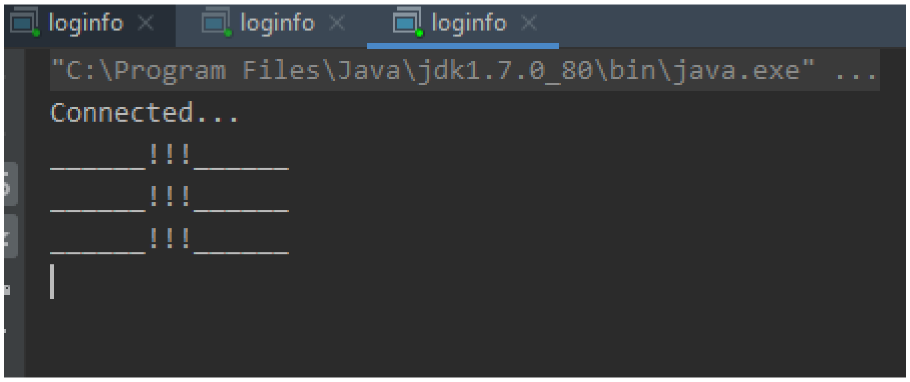


```
Service Open...Wed May 26 07:20:00 AEST 2021
Starting thread for client 1 at Wed May 26 07:20:21 AEST 2021
Client 1's host name is 127.0.0.1
Client 1's IP Address is 127.0.0.1
Starting thread for client 2 at Wed May 26 07:20:23 AEST 2021
Client 2's host name is 127.0.0.1
Client 2's IP Address is 127.0.0.1
```

Whiteboard



Console feedback

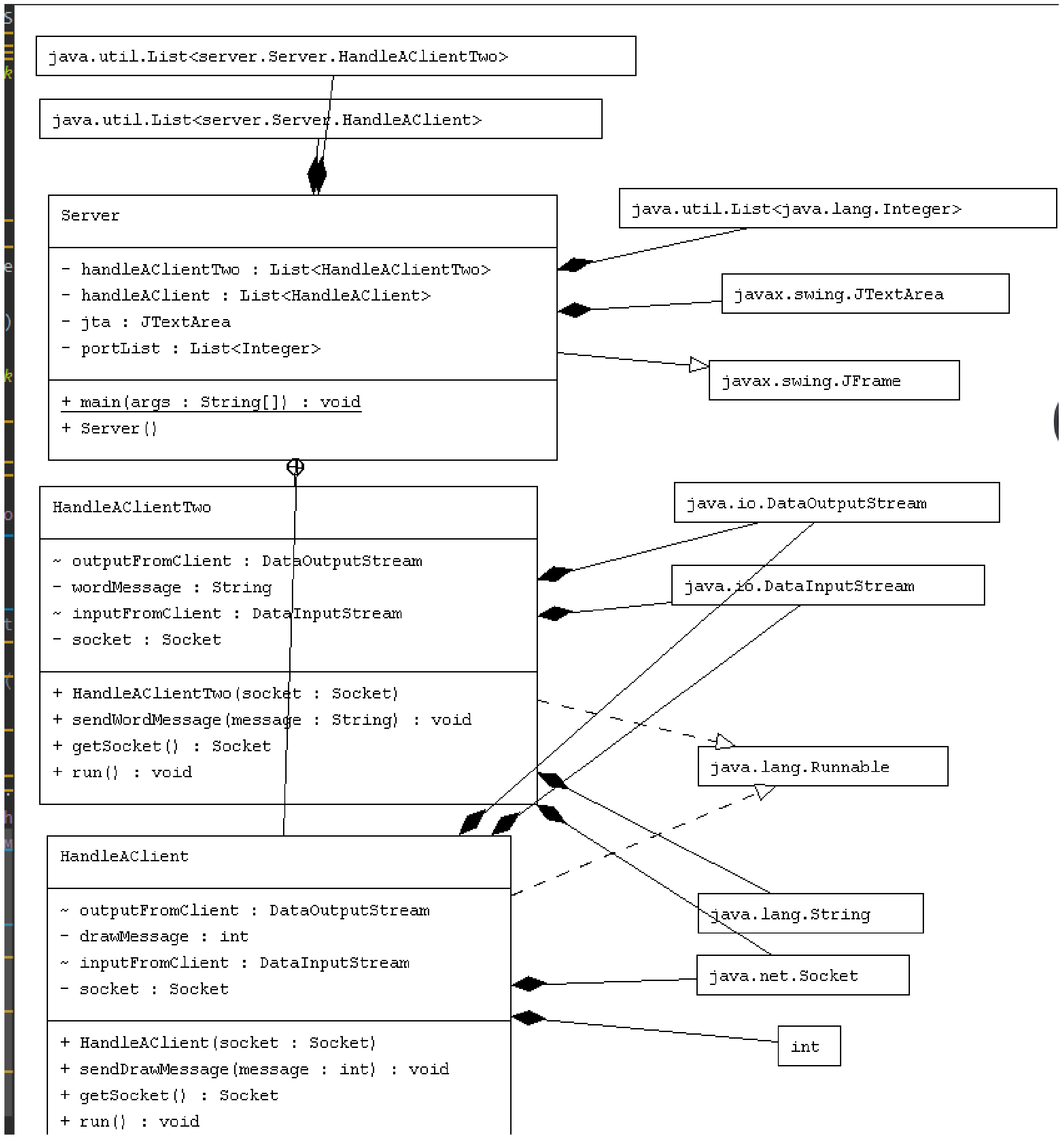


```
"C:\Program Files\Java\jdk1.7.0_80\bin\java.exe" ...
Connected...
______!!!______
______!!!______
______!!!______
```

(2).UML

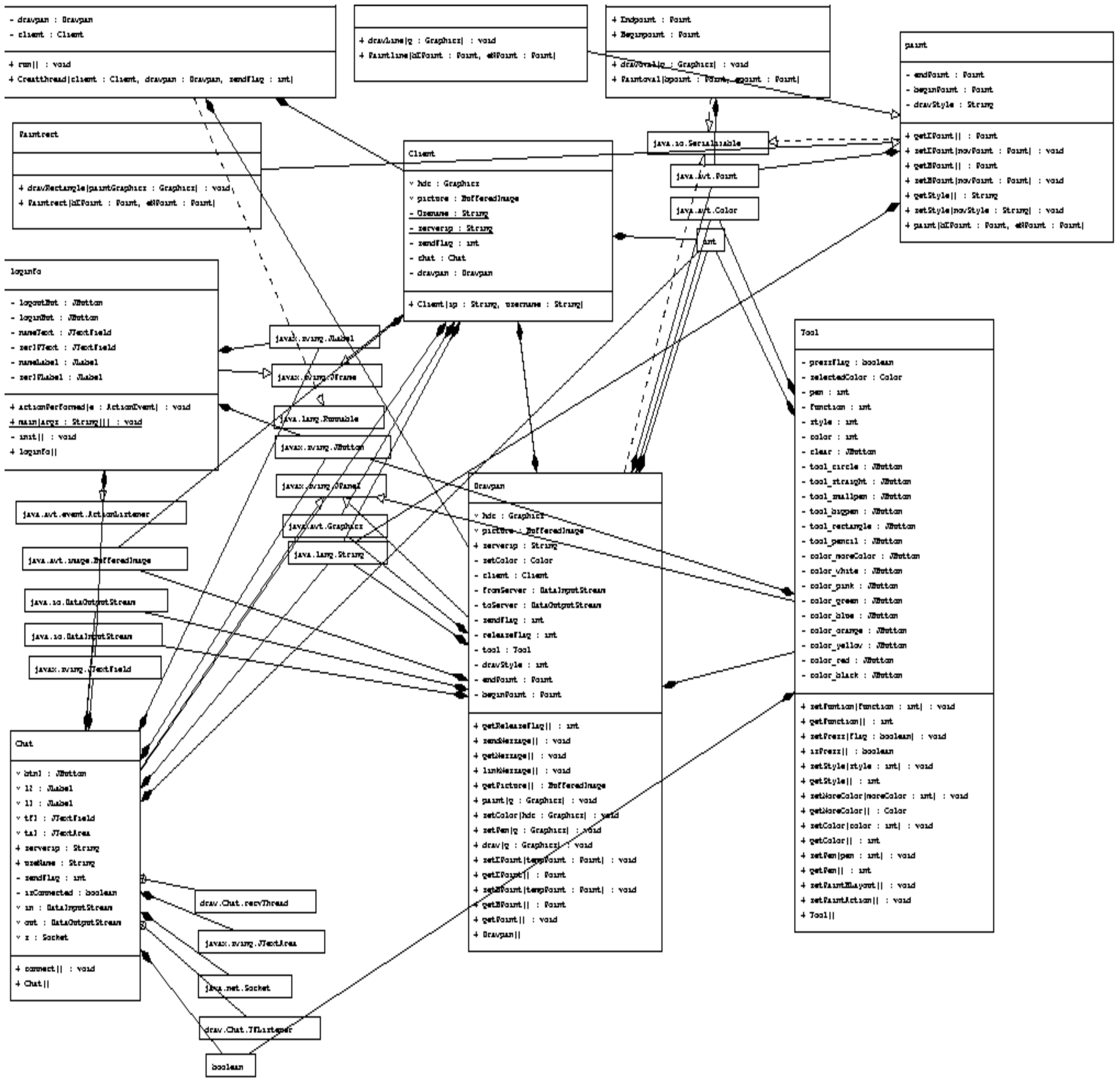
Server

Server side and client side respectively as an independent project. After the server starts, it will listen on the port and accept client connections. After the client logs in to the server, the server will start a new thread to listen to client messages and perform operations.


java.util.List<server.Server.HandleAClientTwo>
java.util.List<server.Server.HandleAClient>
Server
- handleAClientTwo : List<HandleAClientTwo>
- handleAClient : List<HandleAClient>
- jta : JTextArea
- portList : List<Integer>
+ main(args : String[]) : void
+ Server()
java.util.List<java.lang.Integer>
javax.swing.JTextArea
javax.swing.JFrame
HandleAClientTwo
~ outputFromClient : DataOutputStream
- wordMessage : String
~ inputFromClient : DataInputStream
- socket : Socket
+ HandleAClientTwo(socket : Socket)
+ sendWordMessage(message : String) : void
+ getSocket() : Socket
+ run() : void
java.io.DataOutputStream
java.io.DataInputStream
java.lang.Runnable
HandleAClient
~ outputFromClient : DataOutputStream
- drawMessage : int
~ inputFromClient : DataInputStream
- socket : Socket
+ HandleAClient(socket : Socket)
+ sendDrawMessage(message : int) : void
+ getSocket() : Socket
+ run() : void
java.lang.String
java.net.Socket
int

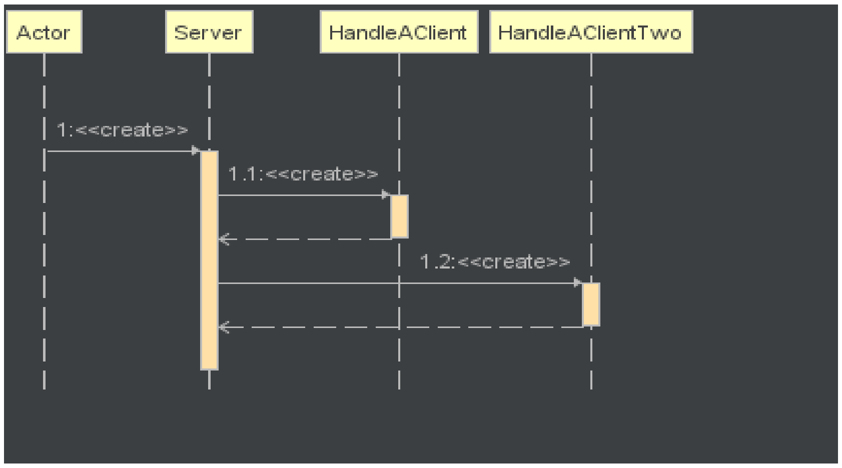
Client

The Project client consists of two parts: the drawing board and the Client login interface. The artboard is implemented by Graphics, and the event is captured by the listener and sent to the Server. The login interface is implemented using Jframe.

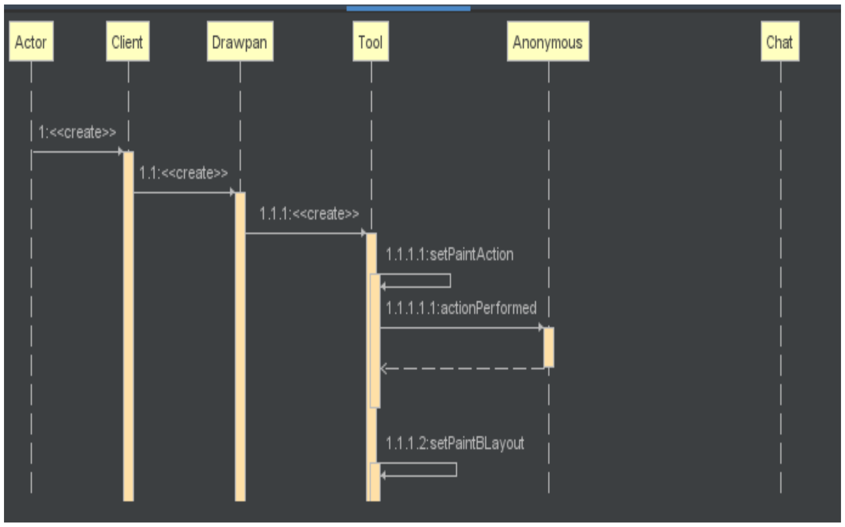


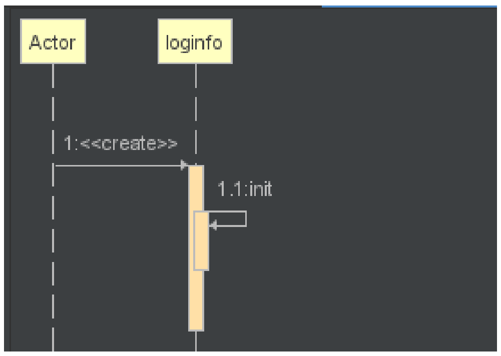
(3). Interaction Diagram

Server



Client





4. Critical Analysis

1. Creativity elements

1. The user interface has a dialog box to provide communication with other users.

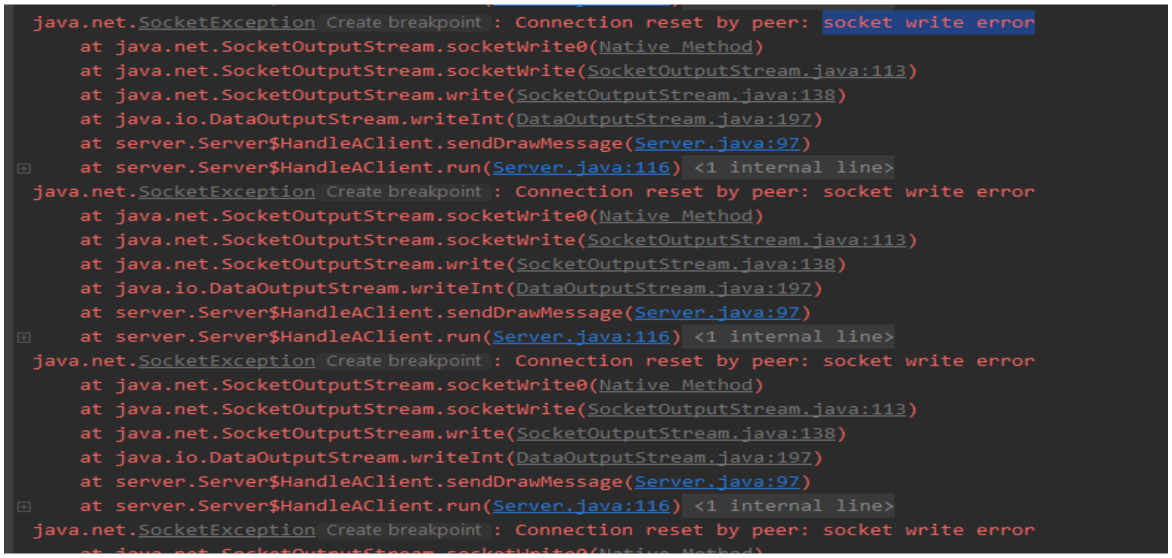
2. When creating a TCP connection, the client will dynamically generate a request port in the software layer and ensure data transmission through the server.

3. The user can also use any color based on the palette in the drawing board.

4. Eraser function. Users can eliminate some lines or graphics on the whiteboard.

2. Known Deficiencies

In the process of using TCP protocol to create the whiteboard, the problem of package loss in network transmission was discovered. When TCP transfers data, there will be breakpoints or transmission errors when the local network is unstable. The compiler will throw an error as shown below: Connect reset by peer: Socket write error. Because the network is unstable, one end of the server or client exits. However, the connection was not closed when exiting. Therefore, if the other end is reading data from the connection, this exception is thrown (Connection reset). In other words, it is caused by read and write operations after the connection is broken.



```
java.net.SocketException Create breakpoint : Connection reset by peer: socket write error
    at java.net.SocketOutputStream.socketWrite0(Native Method)
    at java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:113)
    at java.net.SocketOutputStream.write(SocketOutputStream.java:138)
    at java.io.DataOutputStream.writeInt(DataOutputStream.java:197)
    at server.Server$HandleAClient.sendDrawMessage(Server.java:97)
    at server.Server$HandleAClient.run(Server.java:116) <1 internal line>
java.net.SocketException Create breakpoint : Connection reset by peer: socket write error
    at java.net.SocketOutputStream.socketWrite0(Native Method)
    at java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:113)
    at java.net.SocketOutputStream.write(SocketOutputStream.java:138)
    at java.io.DataOutputStream.writeInt(DataOutputStream.java:197)
    at server.Server$HandleAClient.sendDrawMessage(Server.java:97)
    at server.Server$HandleAClient.run(Server.java:116) <1 internal line>
java.net.SocketException Create breakpoint : Connection reset by peer: socket write error
    at java.net.SocketOutputStream.socketWrite0(Native Method)
    at java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:113)
    at java.net.SocketOutputStream.write(SocketOutputStream.java:138)
    at java.io.DataOutputStream.writeInt(DataOutputStream.java:197)
    at server.Server$HandleAClient.sendDrawMessage(Server.java:97)
    at server.Server$HandleAClient.run(Server.java:116) <1 internal line>
java.net.SocketException Create breakpoint : Connection reset by peer: socket write error
```

3. Future improvements

Due to thread transmission, packet loss sometimes occurs. Therefore, it is planned to add a TCP KeepAlive keep-alive mechanism. Whether it is the server or the client, after one party turns on the KeepAlive function, it will automatically send a heartbeat packet to the other party within the specified time, and the other party will automatically reply after receiving the heartbeat packet to tell the other party that I am still online. This method can be used to reduce data packet loss. When the server or client is disconnected due to the network.