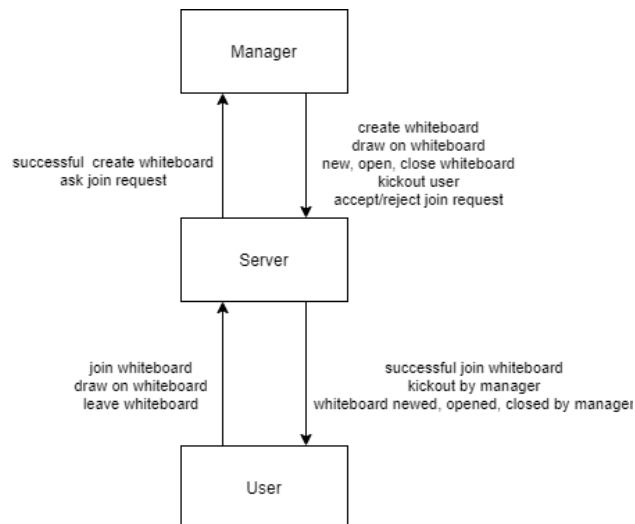# COMP90015 Projet2 2020S1 Report
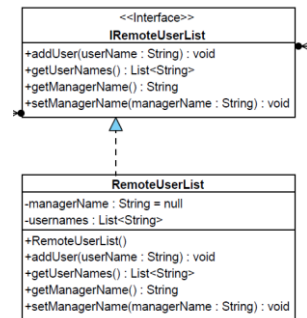
Xulin Yang 904904

System architecture

This the system architecture diagram. The system has a central server. The whiteboard and users' information are stored on the server. Different users like manager and user can use the whiteboard with different privileges. The server has RMI objects to be used by clients and also having TCP requests from clients' operations.

Communication protocols & message formats

The communication protocol is that I have used RMI to handle remote object. In the system I have one remote object for drawing shapes on the canvas, and one remote object
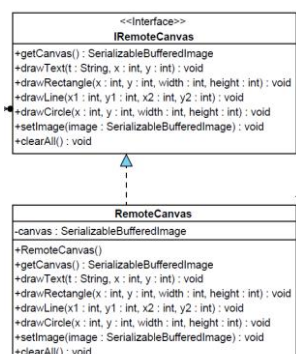
This the class for drawing on the whiteboard uses RMI for the system. The client can get the remote object by using the lookup remote object by the key from the rmiregistry. The client can draw on whiteboard by calling the remote methods defined in the IRemoteCanvas.

```
          <<Interface>>
         IRemoteUserList
+addUser(userName : String) : void
+getUserNames() : List<String>
+getManagerName() : String
+setManagerName(managerName : String) : void

          RemoteUserList
-managerName : String = null
-usernames : List<String>
+RemoteUserList()
+addUser(userName : String) : void
+getUserNames() : List<String>
+getManagerName() : String
+setManagerName(managerName : String) : void
```

This the class for holding the client's names information uses RMI for the system. The client can get the remote object by using the lookup remote object by the key from the rmiregistry. The client can get client's name who is using the whiteboard by calling the remote methods defined in the IRemoteCanvas.

The system also uses TCP as the communication between clients and server.
The server can process request:
1. Create Whiteboard
2. Join Whiteboard
3. Ask Join Whiteboard result
4. User leave Whiteboard
5. Manager close Whiteboard
6. Kick out user
7. Manager new a whiteboard
8. Manager open a whiteboard

The manager can process request:
1. Ask Join Whiteboard

The client can process request:
1. Kick out user
2. Manager closed the whiteboard
3. Manager new a whiteboard
4. Manager open a whiteboard

The message is JSON formatted. With {"request": task above, } and other key, value pair as data if needed.

Design diagrams

Class diagram

As you can see from the class diagram, in the root package, the JoinWhiteBoard, CreateWhiteBoard and Server are three classes for start the system for user, manager and the central server. The RegistryConstant contains the constant for the JoinWhiteBoard and CreateWhiteBoard to lookup the RMI remote canvas and userlists from the server. The util class contains some helper functions.

In the ServerGUI package, there is a gui class for the server which is ServerGUI to be invoked when the server starts.

In the remote package, I have two major class uses RMI to be used during the client server communication. RemoteCanvas implement the IRemoteCanvas class with a SerializableBufferedImage class because the BufferedImage class in JAVA is not serializable thus it cannot be passed in the RMI communication. So, I serialize it to make the BufferedImage can be used as whiteboard canvas. The RemoteCanvas is a shared object for all clients of shared whiteboard to draw shapes on. The RemoteUserList implement IRemoteUserList class to be used to store all client's usernames including manager, users and users waiting for manager's acceptance/rejection. And there is a UserManager class for the server to perform the management actions on IRemoteUserList.

In the Communication package, I have ClientRequestsThread works at server to process client's requests and send responses by using the methods provided by CommunicationSocket class. ClientConnection works at the manager and the user side which can be used to establish the connection to the server by sending requests. ManagerCommunicationThread works at manager side to receive request from server and process and send responses to the server. UserCommunicationThread works at user side to receive request from server and process and send responses to the server. CommunicationConstant contains the constant use during the client and server communication.

In the WhiteBoard package, it contains the class used for the GUI and logic of the frontend application. WhiteBoardApplication is the GUI class used by the manager and the user. It initializes the UI components. If the client is manager, then the application has the functionality provided by the FileMenu class. The WhiteBoardApplication uses WhiteboardCanvasPanel to let client to draw shapes on it. The logic of drawing is handled by the PaintManager. TimeDialog is a class with Dialog but auto select cancel if the client makes no choices within the timeout period. The PaintConstant class contains the constant for

painting such as tool names. The WhiteBoardConstant class contains constants used by the application. The Util class has some helper functions for the application.

Sequence diagram

The first sequence diagram is the process for the server to parse the client's TCP requests. Firstly, it receives the request string. Secondly, it parses it as JSON object. Thirdly, it performs action according to the request message type.

The second sequence diagram is the process for the manager to parse the server's TCP requests. Firstly, it receives the request string. Secondly, it parses it as JSON object. Thirdly, it performs action according to the request message type. The manager returns acceptance/rejection to the user's request to ask to join the whiteboard.

The third sequence diagram is the process for the user to parse the server's TCP requests. Firstly, it receives the request string. Secondly, it parses it as JSON object. Thirdly, it performs action according to the request message type. The application will notify user with the notification from the server to the user.

The fourth sequence diagram is the process for the application to response when the user draws the shape on the whiteboard by clicking on the canvas. It receives the point on the canvas that the user clicked and invokes various methods in the PaintManager according to the tools that the client has selected.
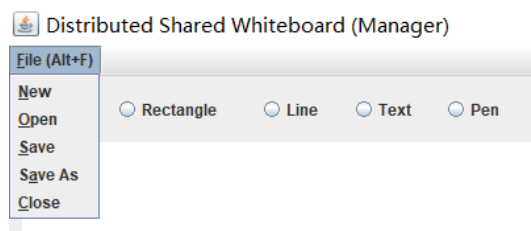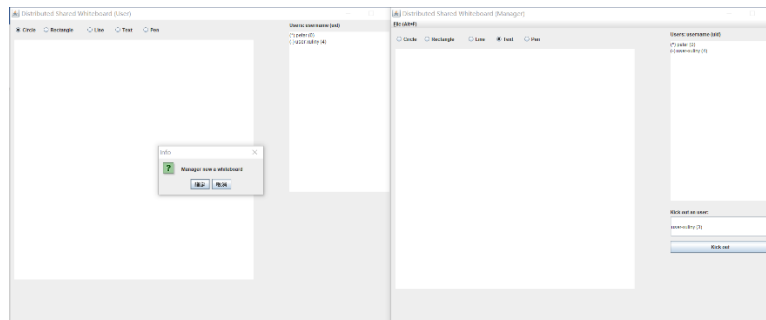
Implementation details

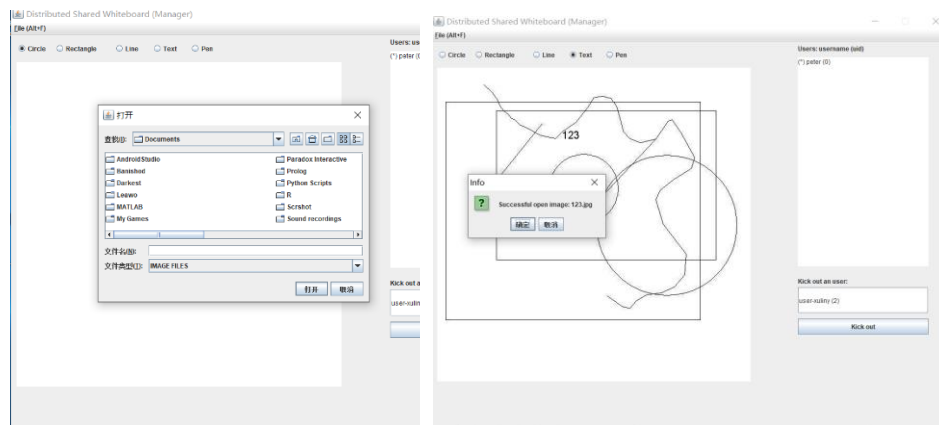This is the server GUI. To see all the users who are using the whiteboard.
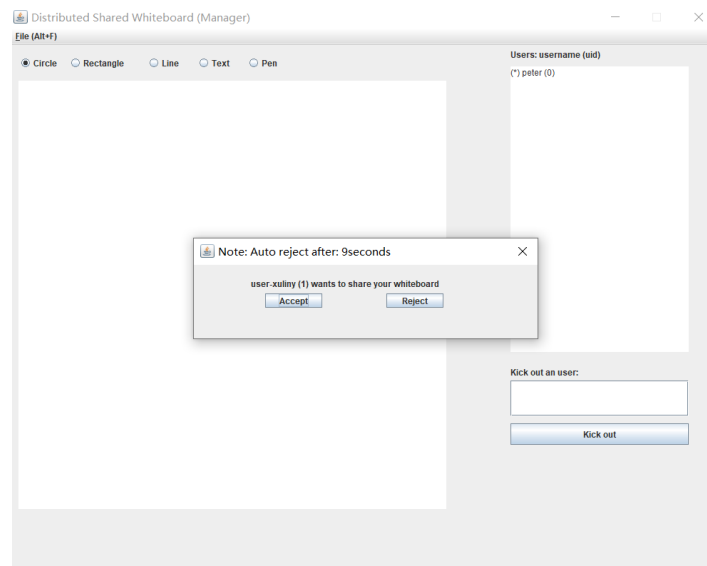
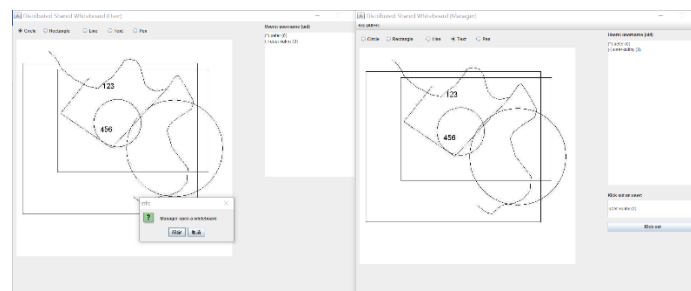The manager can accept or auto reject timeout in 10 seconds user's request to join the whiteboard.

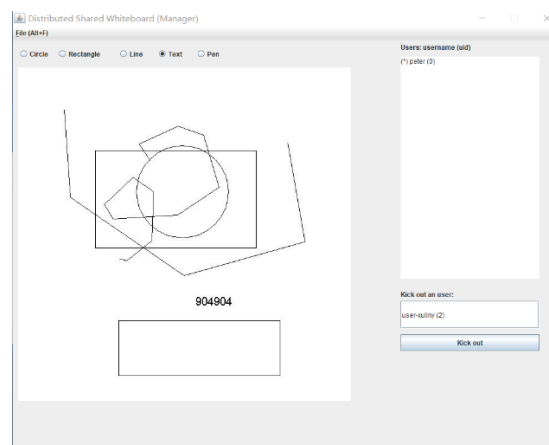The manager have the privileges to new/open/save/saveAs/close the whiteboard.

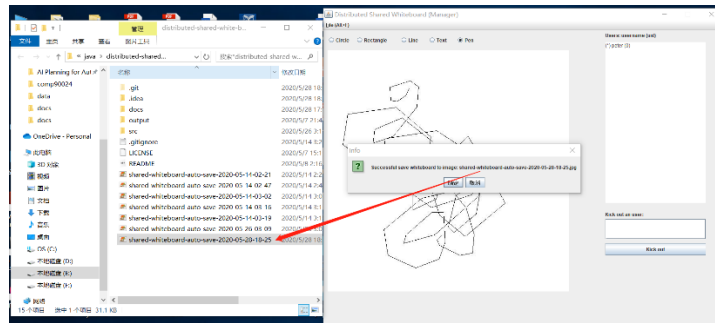When the manager new a whiteboard, all the users will be notified.
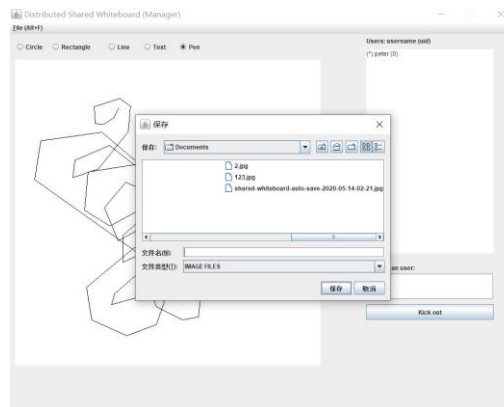

Manager can open a saved whiteboard.
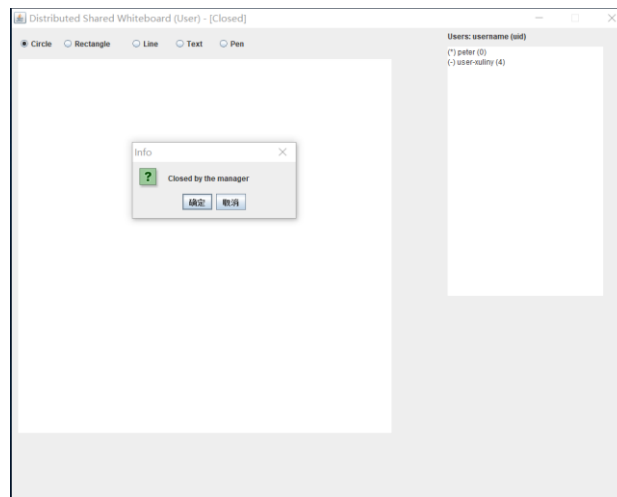

And the user will be notified

The manager can save the whiteboard to a the default directory with an auto generated file name.
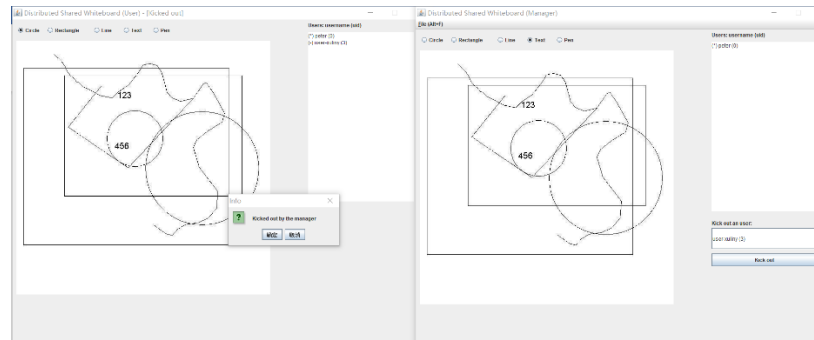

The manager can save the whiteboard by specifying the path and the file name.

Users will be notified if the manager has closed the whiteboard.

All the clients can draw circle, rectangle, lines, text on the canvas and updated simultaneously on all the clients.

User will be notified if kickout by the manager.