COMP90015 Distributed Systems
Assignment 2 Report

Liguo Chen
Student ID: 851090

May 25, 2021

Problem

In this project, a shared whiteboard will be built, which supports multiple users drawing simultaneously.

Some basic features include:

- draw line, oval, circle, and rectangle
- insert texts
- users can choose the favourite colors for drawing and inserted text

Some advanced features include:

- users can chat with each other
- manager can kick out ordinary users
- users can leave
- manager can save the whiteboard on local machine

System architecture

The system is Client - Server architecture.

All information about whiteboard, including active users, chat messages, content of the canvas, is stored in the central server.

Information exchange, for example manager kicks out a user, is done via the central server.

Communication

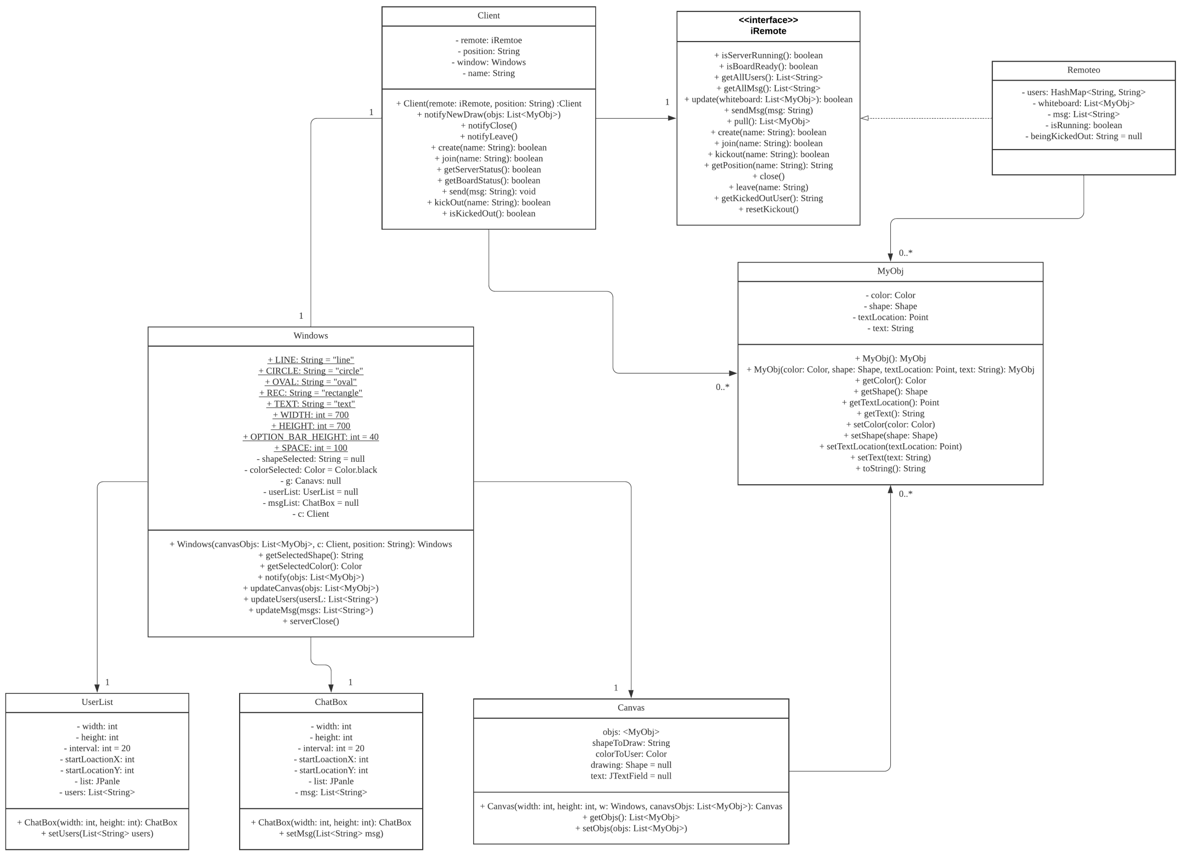
The Client - Server communication is done by using java RMI.

iRemote is the remote interface, which defines available methods the server prodvides. **Remoteo** is the remote object, which implements all the remote methdos defined in **iRemote**, and is the main server-side end-point for communication.

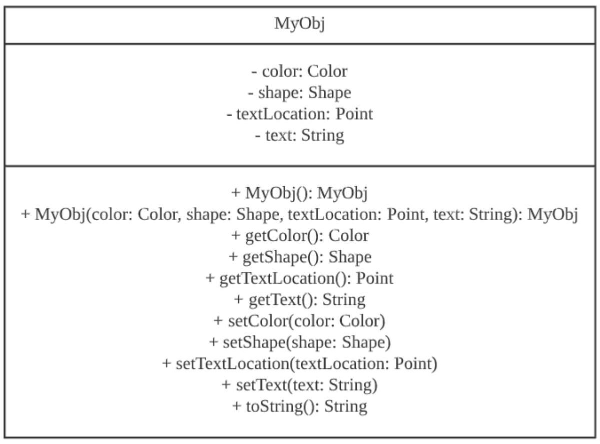
Design diagrams

Class diagram

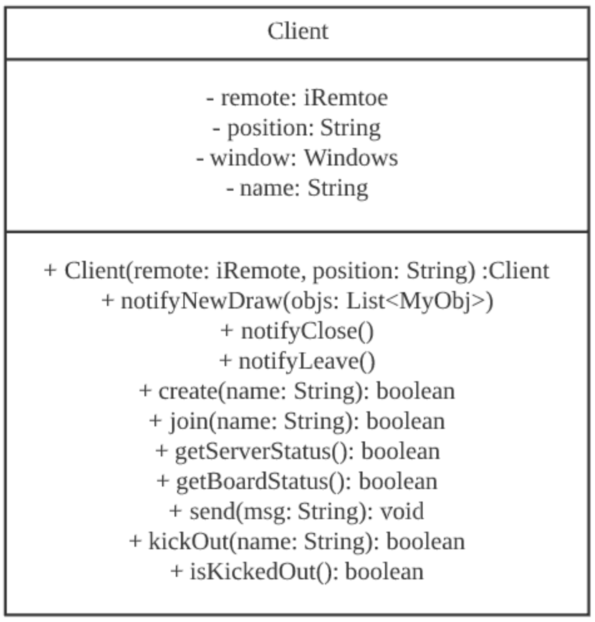
Overall


Client
- remote: iRemtoe
- position: String
- window: Windows
- name: String
+ Client(remote: iRemote, position: String) :Client
+ notifyNewDraw(objs: List<MyObj>)
+ notifyClose()
+ notifyLeave()
+ create(name: String): boolean
+ join(name: String): boolean
+ getServerStatus(): boolean
+ getBoardStatus(): boolean
+ send(msg: String): void
+ kickOut(name: String): boolean
+ isKickedOut(): boolean
1
1
<<interface>>
iRemote
+ isServerRunning(): boolean
+ isBoardReady(): boolean
+ getAllUsers(): List<String>
+ getAllMsg(): List<String>
+ update(whiteboard: List<MyObj>): boolean
+ sendMsg(msg: String)
+ pull(): List<MyObj>
+ create(name: String): boolean
+ join(name: String): boolean
+ kickout(name: String): boolean
+ getPosition(name: String): String
+ close()
+ leave(name: String)
+ getKickedOutUser(): String
+ resetKickout()
Remoteo
- users: HashMap<String, String>
- whiteboard: List<MyObj>
- msg: List<String>
- isRunning: boolean
- beingKickedOut: String = null
0..*
MyObj
- color: Color
- shape: Shape
- textLocation: Point
- text: String
+ MyObj(): MyObj
+ MyObj(color: Color, shape: Shape, textLocation: Point, text: String): MyObj
+ getColor(): Color
+ getShape(): Shape
+ getTextLocation(): Point
+ getText(): String
+ setColor(color: Color)
+ setShape(shape: Shape)
+ setTextLocation(textLocation: Point)
+ setText(text: String)
+ toString(): String
0..*
0..*
1
Windows
+ LINE: String = "line"
+ CIRCLE: String = "circle"
+ OVAL: String = "oval"
+ REC: String = "rectangle"
+ TEXT: String = "text"
+ WIDTH: int = 700
+ HEIGHT: int = 700
+ OPTION_BAR_HEIGHT: int = 40
+ SPACE: int = 100
- shapeSelected: String = null
- colorSelected: Color = Color.black
- g: Canavs: null
- userList: UserList = null
- msgList: ChatBox = null
- c: Client
+ Windows(canvasObjs: List<MyObj>, c: Client, position: String): Windows
+ getSelectedShape(): String
+ getSelectedColor(): Color
+ notify(objs: List<MyObj>)
+ updateCanvas(objs: List<MyObj>)
+ updateUsers(usersL: List<String>)
+ updateMsg(msgs: List<String>)
+ serverClose()
1
UserList
- width: int
- height: int
- interval: int = 20
- startLoactionX: int
- startLocationY: int
- list: JPanle
- users: List<String>
+ ChatBox(width: int, height: int): ChatBox
+ setUsers(List<String> users)
1
ChatBox
- width: int
- height: int
- interval: int = 20
- startLoactionX: int
- startLocationY: int
- list: JPanle
- msg: List<String>
+ ChatBox(width: int, height: int): ChatBox
+ setMsg(List<String> msg)
1
Canvas
objs: <MyObj>
shapeToDraw: String
colorToUser: Color
drawing: Shape = null
text: JTextField = null
+ Canvas(width: int, height: int, w: Windows, canavsObjs: List<MyObj>): Canvas
+ getObjs(): List<MyObj>
+ setObjs(objs: List<MyObj>)

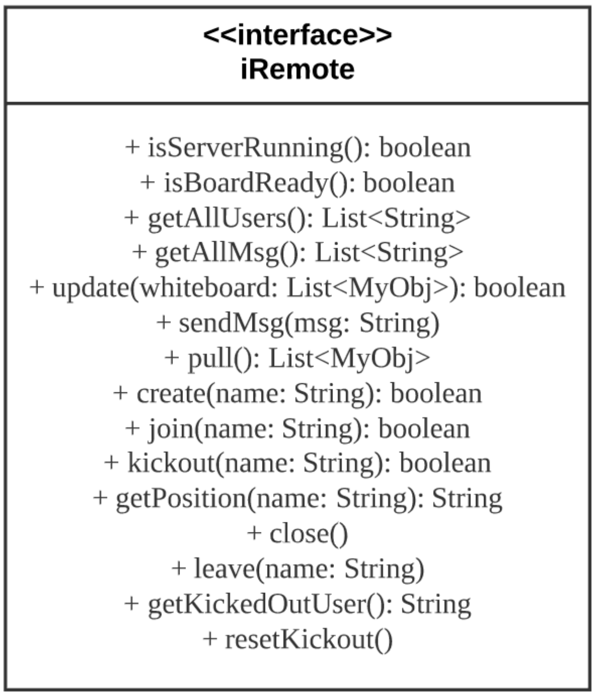
MyObj



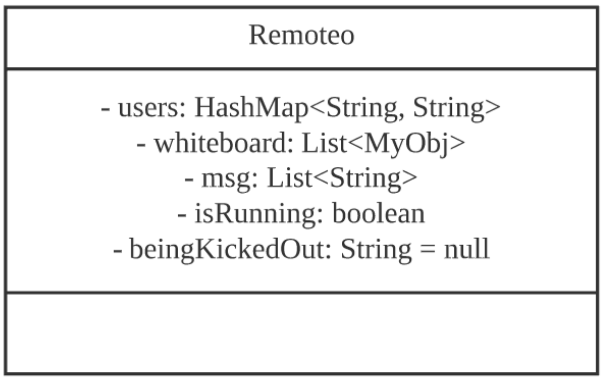
Client


Client
- remote: iRemtoe
- position: String
- window: Windows
- name: String
+ Client(remote: iRemote, position: String) :Client
+ notifyNewDraw(objs: List<MyObj>)
+ notifyClose()
+ notifyLeave()
+ create(name: String): boolean
+ join(name: String): boolean
+ getServerStatus(): boolean
+ getBoardStatus(): boolean
+ send(msg: String): void
+ kickOut(name: String): boolean
+ isKickedOut(): boolean

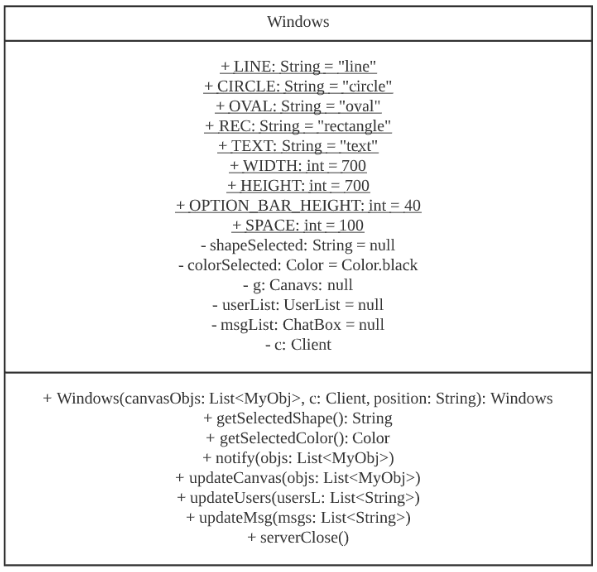
iRemote


<<interface>>
iRemote
+ isServerRunning(): boolean
+ isBoardReady(): boolean
+ getAllUsers(): List<String>
+ getAllMsg(): List<String>
+ update(whiteboard: List<MyObj>): boolean
+ sendMsg(msg: String)
+ pull(): List<MyObj>
+ create(name: String): boolean
+ join(name: String): boolean
+ kickout(name: String): boolean
+ getPosition(name: String): String
+ close()
+ leave(name: String)
+ getKickedOutUser(): String
+ resetKickout()

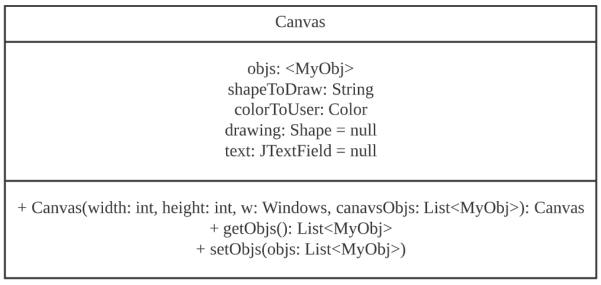
Remoteo


Remoteo
- users: HashMap<String, String>
- whiteboard: List<MyObj>
- msg: List<String>
- isRunning: boolean
- beingKickedOut: String = null

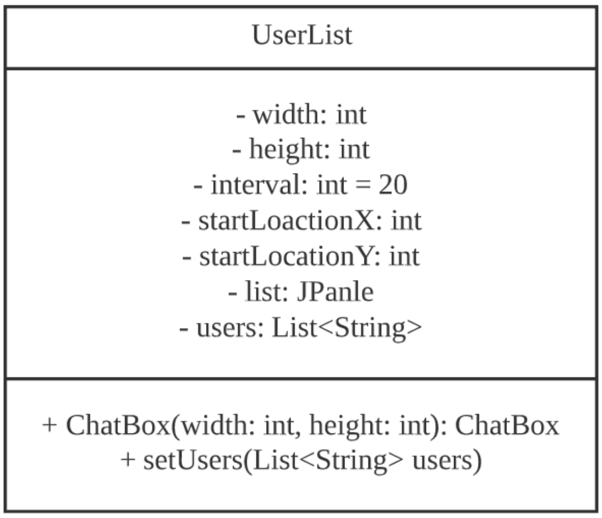
Windows


Windows
+ LINE: String = "line"
+ CIRCLE: String = "circle"
+ OVAL: String = "oval"
+ REC: String = "rectangle"
+ TEXT: String = "text"
+ WIDTH: int = 700
+ HEIGHT: int = 700
+ OPTION_BAR_HEIGHT: int = 40
+ SPACE: int = 100
- shapeSelected: String = null
- colorSelected: Color = Color.black
- g: Canavs: null
- userList: UserList = null
- msgList: ChatBox = null
- c: Client
+ Windows(canvasObjs: List<MyObj>, c: Client, position: String): Windows
+ getSelectedShape(): String
+ getSelectedColor(): Color
+ notify(objs: List<MyObj>)
+ updateCanvas(objs: List<MyObj>)
+ updateUsers(usersL: List<String>)
+ updateMsg(msgs: List<String>)
+ serverClose()

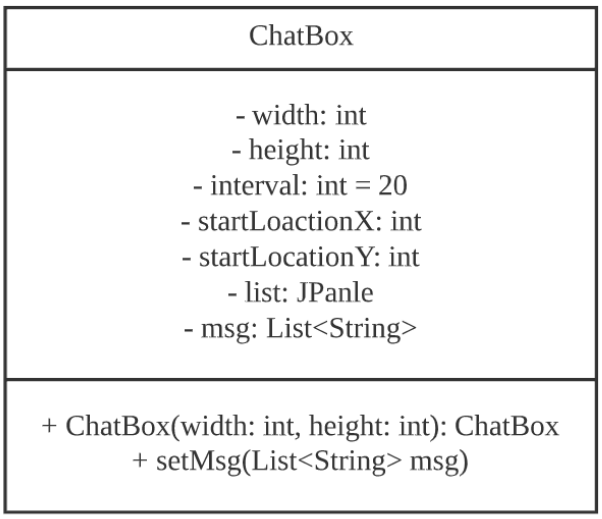
Canvas



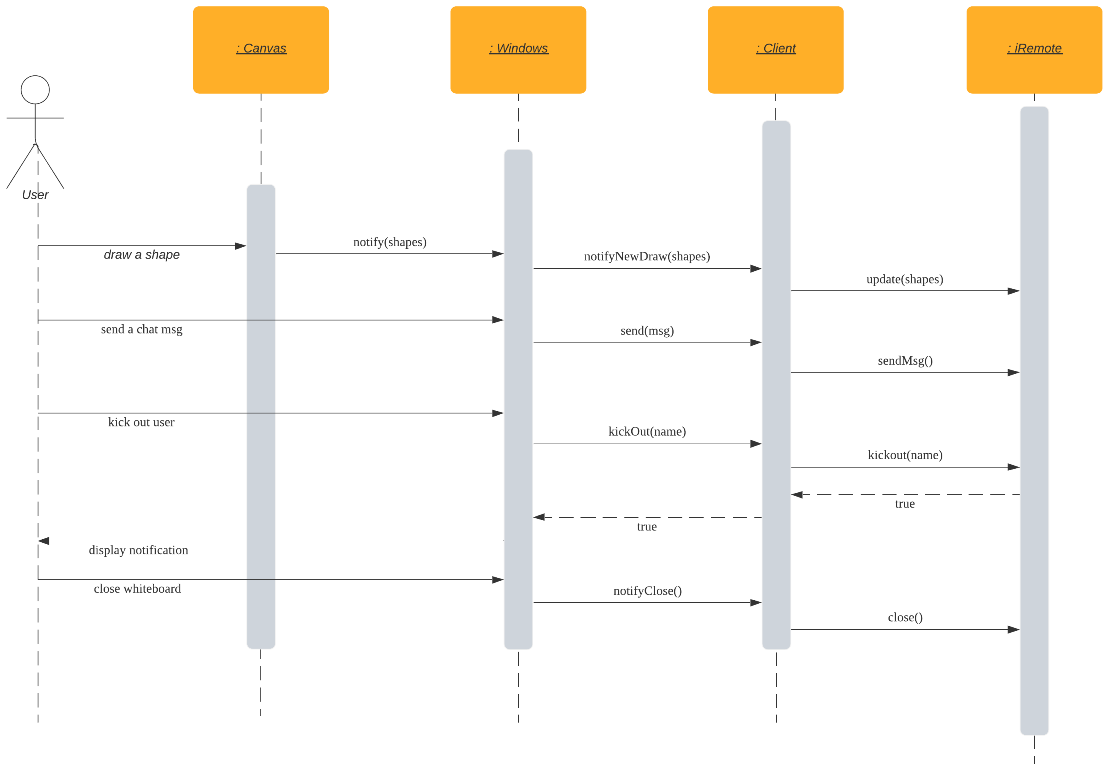
UserList



ChatBox



Sequence Diagram



Implementation details

MyObj represents an object in the canvas. It could be a shape, or an inserted text. To support java RMI communication, **MyObj** is serializable.

Client represents a user in the system. It is responsible for notifying server about modifications and getting the latest information from server and passing down to window for rendering.

Windows represents the client GUI window. It communicate with **Client** to fulfill all the required features of the system.

Canvas represents the canvas object in the client GUI. When a new shape is drawn or a new text is inserted, it will notify **Windows**

UserList represents a component in the GUI and its main job is to display a list of active users in the system

ChatBox is a GUI component similar to **UserList**. Instead of displaying a list of users, **ChatBox** shows a list of chat message sent by all users.