Assignment 2 Share White Board

Ruipu Cui 1298198

Project Overview

The **Shared Whiteboard** project is a collaborative drawing application. The application include a server program and a client program and using Java RMI (Remote Method Invocation) for server-client communication. It enables multiple users to draw together in real time on shared whiteboards over a network. The white board supports multiple functionalities including drawing, user management, file management and some additional features. The drawing function enables users to draw in free line, straight line, rectangle, triangle, oval and select colour. The file management system allows manager to create, save, load and export the white board. The manager could use the user management system to approve and kick other participants. It also supports chat room and view all current collaborators.

Project Structure

In this section, I am going to briefly explain client system components, Thread synchronisation, Server system components and Server-client Communication.

....

Client System Components

CreateWhiteBoard class

The **CreateWhiteBoard** class is the main entry point for both creating and joining a shared whiteboard session in the application. It presents a launch UI for users to enter connection details and choose whether to start as a manager or participant. If acting as the manager, it starts the RMI registry, binds a new whiteboard server, registers the user, and launches the main UI. For participants, it attempts to connect to an existing server, submits a join request, and enters a waiting state until approved. The class also handles error cases such as connection failure, username conflicts, or being rejected by the manager, returning users back to the launch screen if necessary.

LaunchUI class

The LaunchUI class provides the initial graphical interface for users to start or join a shared whiteboard session. It collects user input such as IP address, port, username, and role (Manager or Participant), and validates the fields before proceeding. Once the "Enter" button is pressed, it triggers a callback (onLaunch) with a LaunchConfig object containing the connection details and user role. This class serves as a launcher for the application, directing users to the appropriate flow based on their selection.

MainClientUI class

The MainClientUI class is the central entry point for launching the shared whiteboard application's main user interface. It sets up the overall layout, which includes a toolbar, a tabbed pane for multiple whiteboards, and a sidebar with file controls, chat, and collaborator information. It dynamically loads all existing whiteboards from the remote server and allows real-time interaction through drawing tools and communication features. Additionally, the class initiates background polling threads to ensure up-to-date synchronisation (More details later). It supports both managers and participants.

RemoteHandler class

The RemoteHandler class is a utility responsible for establishing a connection to the remote RemoteWhiteBoards RMI service. It constructs the RMI URL based on the given IP address and port, then attempts to retrieve the remote object using Naming.lookup. To improve robustness, the class includes a retry mechanism that attempts the connection up to 5 times with short delays. This class abstracts away the RMI lookup logic and ensures reliable communication setup for clients connecting to the shared whiteboard server.

UserManagementDialog class

The UserManagementDialog class provides a modal user interface for managing participants in the shared whiteboard application. It displays two sections: a Waiting Room, where users pending approval are listed with an “Approve” button, and Current Users, showing active users with a “Kick” button. The dialog continuously updates these lists in real time using a background polling thread. It allows the whiteboard manager to control participant access dynamically

Other Client UI Classes

The **ChatPanel** class provides a scrollable text area to display messages, a text input field, and a send button for users to communicate in real time.
CollaboratorPanel class enable users to view all active collaborators in the share white board.
FilePanel class provides file management functionalities for manager of the share white board, it includes save white board, load white board and export white board.
ToolbarPanel class contains tool selection button, create new board button and user management button

....

Thread Synchronisation

To ensure responsive interaction, real-time updates, and consistent synchronisation across clients, the collaborative whiteboard system employs several background threads, each encapsulated in a dedicated Poller class. These threads perform periodic polling and repainting tasks without blocking the main UI thread. Below is a summary of their roles and functionalities:

1. ChatPoller
Responsible for continuously retrieving the latest chat messages from the server and updating the ChatPanel. This ensures all users see real-time communication without manual refresh.

2. CollaboratorPoller
Updates the CollaboratorPanel by polling the current list of active users from the server. It reflects changes like new users joining or participants being removed.

3. UserKickPoller
Monitors whether the current user has been kicked by the manager. If detected, it immediately returns the user to the launch UI and notifies them through a dialog.

4. UserManagementPoller
Used within the UserManagementDialog, this thread keeps the UI updated with the current waiting and approved users. It automatically refreshes the display when changes occur due to user approval or removal.

5. WaitingApprovalPoller
Used for participants waiting for manager approval. It continuously checks if the user has been accepted or rejected and handles transition to the whiteboard UI or returns them to the launch screen accordingly.

6. WhiteboardPoller
Ensures that all tabs in the whiteboard UI reflect up-to-date canvas data by polling the server's whiteboard states. It also handles adding new boards when created by the manager.

7. WhiteboardRepaintPoller
Each WhiteBoardUI has its own repaint poller to continuously invoke repaint() at short intervals. This ensures that local drawing and remote updates are visually reflected in near real-time.

These threads operate as daemons where appropriate, with careful synchronisation and Swing-safe updates via SwingUtilities.invokeLater to prevent concurrency issues and maintain GUI consistency.

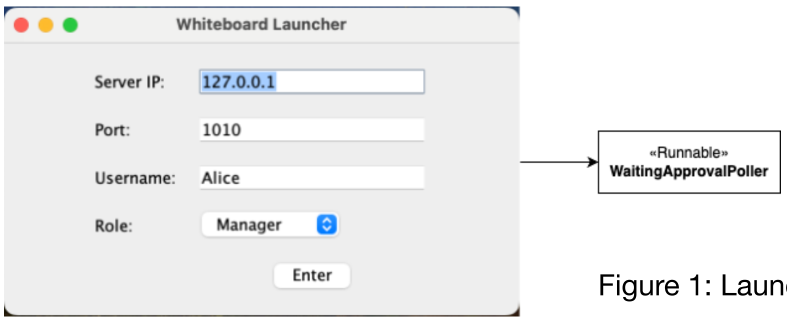


Figure 1: Launch UI

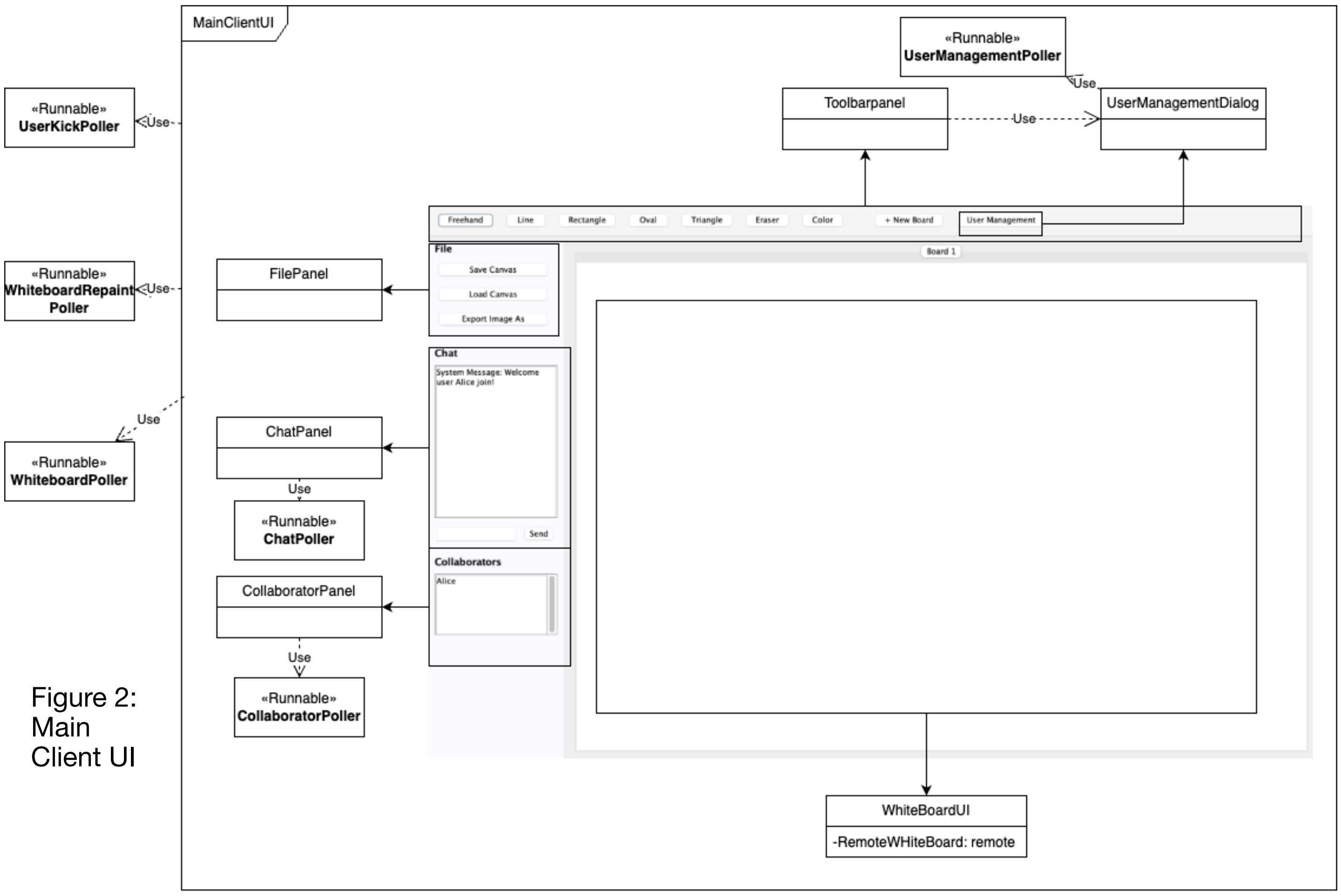


Figure 2: Main Client UI

Server System Components

RMIServer Class

The RMIServer class serves as the entry point for launching the RMI-based backend of the collaborative whiteboard application. It is responsible for:

- Starting the RMI Registry: It initialises the RMI registry on a specified port to allow remote object binding and discovery.

- Creating the WhiteBoards Server Object: It instantiates the WhiteBoards class, which implements the RemoteWhiteBoards interface and contains core server-side logic for managing users, whiteboards, chat, and collaboration.

- Binding the Remote Object: It binds the WhiteBoards instance to the RMI registry using a specified IP address and port, making it accessible to remote clients.

This setup allows clients to remotely access and interact with shared whiteboards through Java RMI.

DrawableShape Class

The DrawableShape abstract class defines the foundational structure for all drawable objects in the collaborative whiteboard application. It encapsulates common attributes and behaviours,

- Colour Management: Each shape has a customisable Colour field for rendering.

- Unique Identity: Each shape is assigned a UUID to support consistent identification, especially useful for syncing across clients.

- Drawing Interface: Declares the draw(Graphics2D g2) method, which must be implemented by subclasses to render the shape.

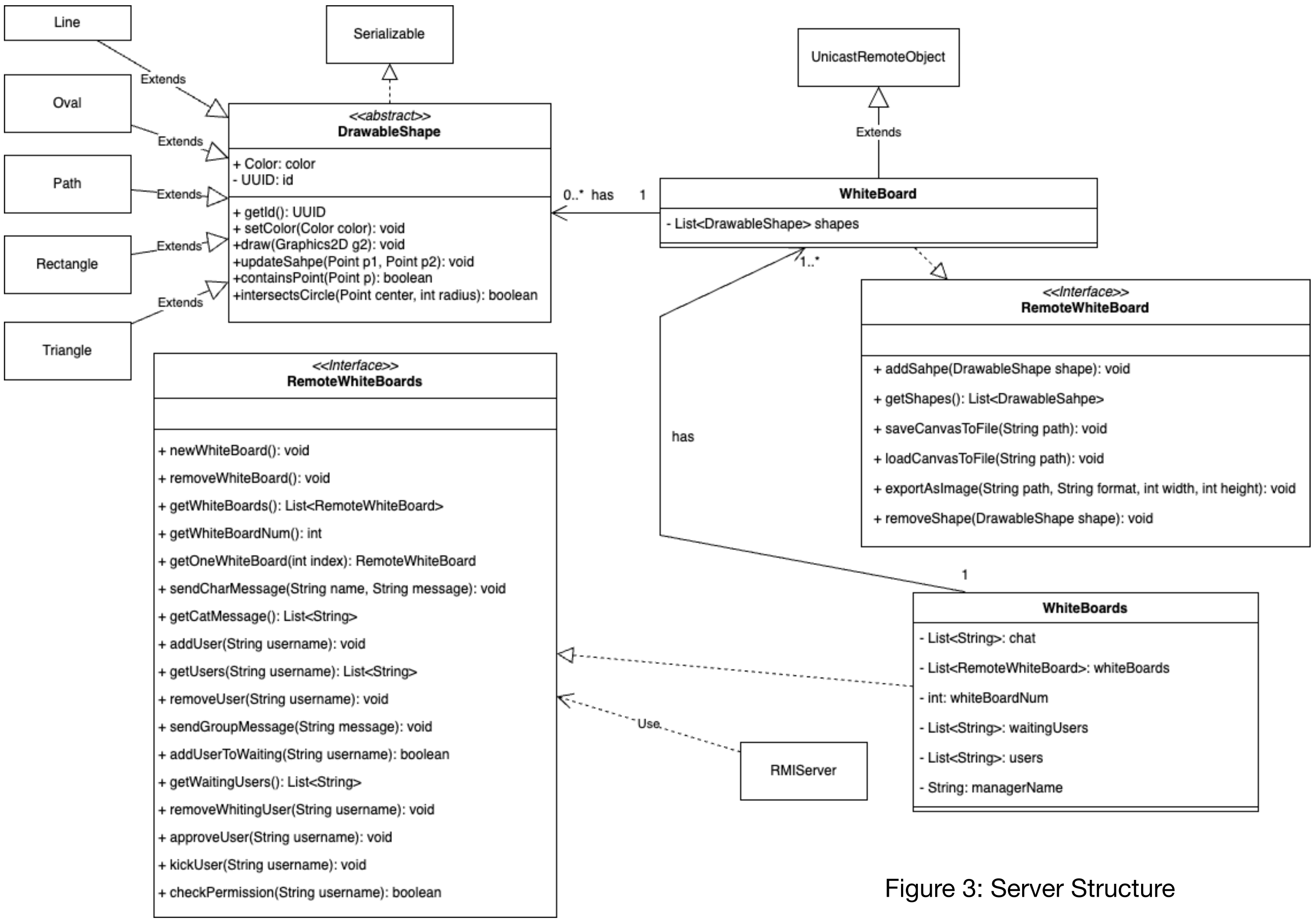
- Shape Interaction Methods: Defines abstract methods like containsPoint(Point p) and intersectsCircle(Point centre, int radius) for interaction logic (e.g., selection, erasing).

WhiteBoard

The WhiteBoard class serves as the core implementation of the RemoteWhiteBoard interface in the shared whiteboard system. It maintains a synchronised list of DrawableShape objects representing all visual elements on a canvas. As a remote object extended via UnicastRemoteObject, it enables real-time interaction and synchronisation of drawing data across multiple clients via Java RMI. Key functionalities include adding and removing shapes, retrieving shape data (getShapes()), and handling persistence operations such as saving and loading the canvas to/from a file. Additionally, it supports exporting the canvas as an image file in various formats. By encapsulating drawing state and operations, this class ensures consistent and centralised management of visual content in the distributed whiteboard application.

WhiteBoards

The WhiteBoards class is the central RMI-exposed server-side component responsible for managing multiple collaborative whiteboards and user sessions. It implements the RemoteWhiteBoards interface and extends UnicastRemoteObject to support remote method invocation. Internally, it maintains a list of RemoteWhiteBoardinstances (each representing a drawable canvas), active users, and users pending approval (in a waiting room). The class provides synchronised methods for creating new boards, retrieving board instances, managing chat messages, and handling user actions such as joining, approval, and kicking. It also designates one user as the manager and provides a permission check mechanism. By centralising board and session management, WhiteBoards enables consistent coordination and real-time collaboration among multiple clients in the distributed whiteboard system.



Line
Oval
Path
Rectangle
Triangle
Extends
Serializable
<<abstract>> DrawableShape
+ Color: color
- UUID: id
+ getId(): UUID
+ setColor(Color color): void
+draw(Graphics2D g2): void
+updateSahpe(Point p1, Point p2): void
+containsPoint(Point p): boolean
+intersectsCircle(Point center, int radius): boolean
UnicastRemoteObject
WhiteBoard
- List<DrawableShape> shapes
0..* has 1
1..*
has
1
<<Interface>> RemoteWhiteBoard
+ addSahpe(DrawableShape shape): void
+ getShapes(): List<DrawableSahpe>
+ saveCanvasToFile(String path): void
+ loadCanvasToFile(String path): void
+ exportAsImage(String path, String format, int width, int height): void
+ removeShape(DrawableShape shape): void
<<Interface>> RemoteWhiteBoards
+ newWhiteBoard(): void
+ removeWhiteBoard(): void
+ getWhiteBoards(): List<RemoteWhiteBoard>
+ getWhiteBoardNum(): int
+ getOneWhiteBoard(int index): RemoteWhiteBoard
+ sendCharMessage(String name, String message): void
+ getCatMessage(): List<String>
+ addUser(String username): void
+ getUsers(String username): List<String>
+ removeUser(String username): void
+ sendGroupMessage(String message): void
+ addUserToWaiting(String username): boolean
+ getWaitingUsers(): List<String>
+ removeWhitingUser(String username): void
+ approveUser(String username): void
+ kickUser(String username): void
+ checkPermission(String username): boolean
Use
RMIServer
WhiteBoards
- List<String>: chat
- List<RemoteWhiteBoard>: whiteBoards
- int: whiteBoardNum
- List<String>: waitingUsers
- List<String>: users
- String: managerName

Figure 3: Server Structure

Server-client Communication

Server-Client Communication Using Java RMI

In this collaborative whiteboard application, Java RMI (Remote Method Invocation) is used to enable seamless communication between the server and multiple clients. The communication is defined through two key remote interfaces: RemoteWhiteBoards and RemoteWhiteBoard, which expose the server's functionalities for remote access.

RemoteWhiteBoards Interface

This interface acts as the main coordination hub for the system. It enables clients to:

- Request the creation of new whiteboards (newWhiteBoard)
- Retrieve a list of available whiteboards (getWhiteBoards, getOneWhiteBoard)
- Send and receive chat messages (sendChatMessage, getChatMessages)

- Manage users, including adding to the waiting list (addUserToWaiting), approving or removing users (approveUser, kickUser), send messages to chat room(sendGroupMessage), retrieving user lists (getUsers, getWaitingUsers)
- Enforce access control using a permission check (checkPermission)

RemoteWhiteBoard Interface

Each individual whiteboard implements this interface, providing methods for:

- Drawing operations, including adding and retrieving shapes (addShape, getShapes)
- File operations to persist or restore board state (saveCanvasToFile, loadCanvasFromFile, exportAsImage)
- Erasing content by removing specific shapes (removeShape)

Communication Flow

- The server registers the RemoteWhiteBoards object in the RMI registry.
- Clients lookup this object using Naming.lookup and obtain a remote reference.
- From there, clients invoke methods as if they were local, but calls are transparently routed to the server.
- The server handles these calls, performs the requested actions (e.g. drawing, user management), and returns results or triggers updates.

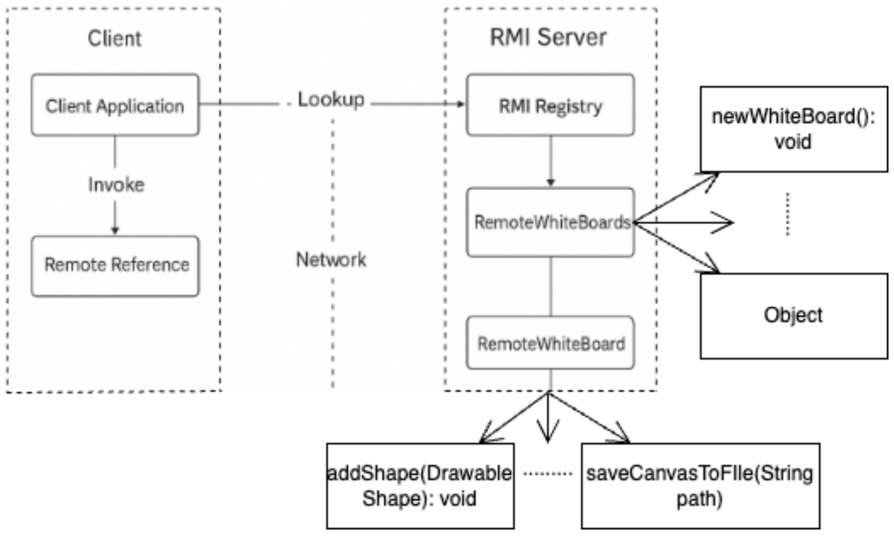


Figure 4: RMI Communication