HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION

FACULTY OF INTERNATIONAL EDUCATION

INFORMATION TECHNOLOGY

PROJECT REPORT

JAVA COLLABORATIVE WHITEBOARD

LECTURER NAME : Dr. Nguyen Dang Quang

STUDENT NAME : Nguyen Thai Bao

STUDENT ID : 21tm11044

CLASS : 22TM-IT

Ho Chi Minh City, December 2023

Acknowledgment

I express my sincere thanks to Mr. Nguyen Dang Quang, my project in charge, who guided me through the project. He gave valuable suggestions and guidance for completing the project, helped me to understand the intricate issues involved in project-making besides effectively presenting it. I am looking forward to receiving all the comments of my teachers to help my limited knowledge better. Sincerely thanks

Preface

After the process of create the application, I believe I have obtained some sort of IT knowledge and others soft skills. If I invest more effort in practicing and learning, then I can surrvive in the competitive coccupational arena in the future.

The report is an indispensable part of the course. In the report, I will try my best to represent systematically and logically all the content of the application and what I have learnt. I hope the report will be a useful material for all reader and user.

CONTENT

I. Project description

1. Objectives

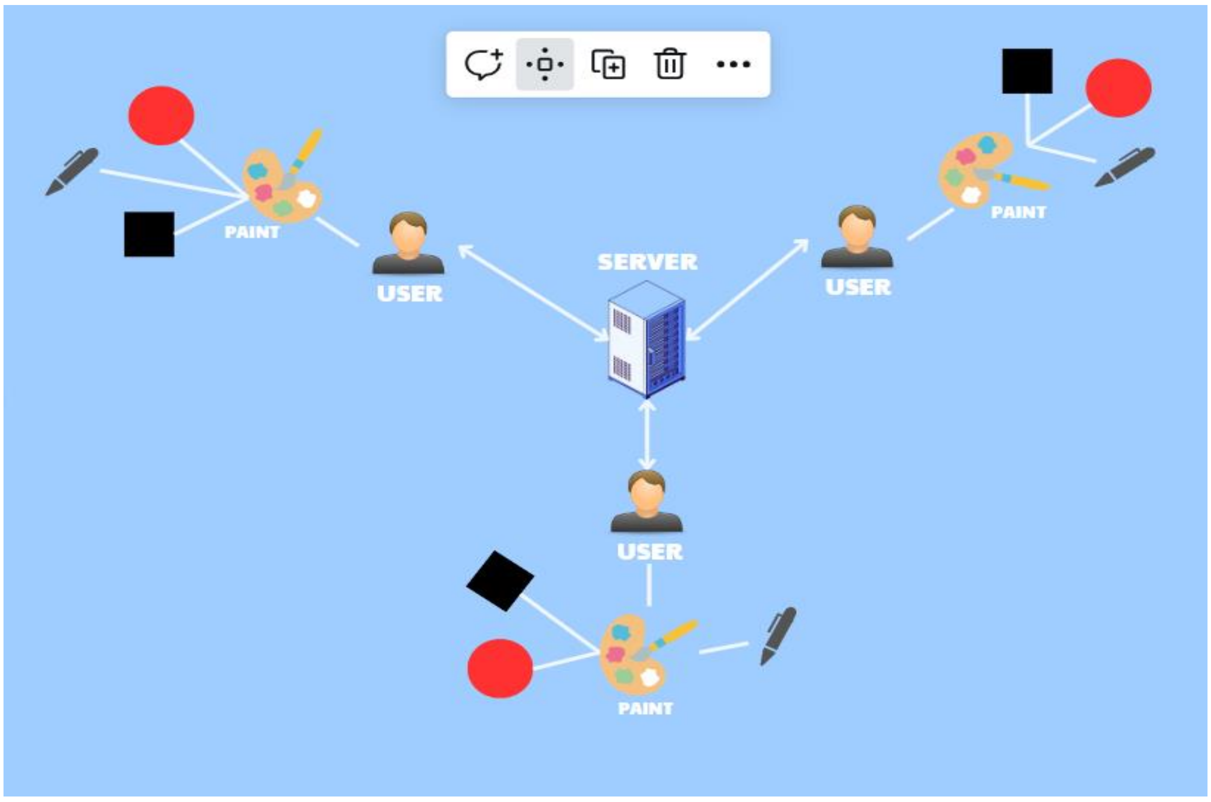
The collaborative white board provides users abilities to customise their drawings; I could find some applications such as Windows Paint, Artweaver, GIMP, etc. However, these programs only available for one user at a time. Collborative Whiteboard can allow multiple users to draw on board simultaneously through the network, which helps users spread their idea to other people more comfortable.

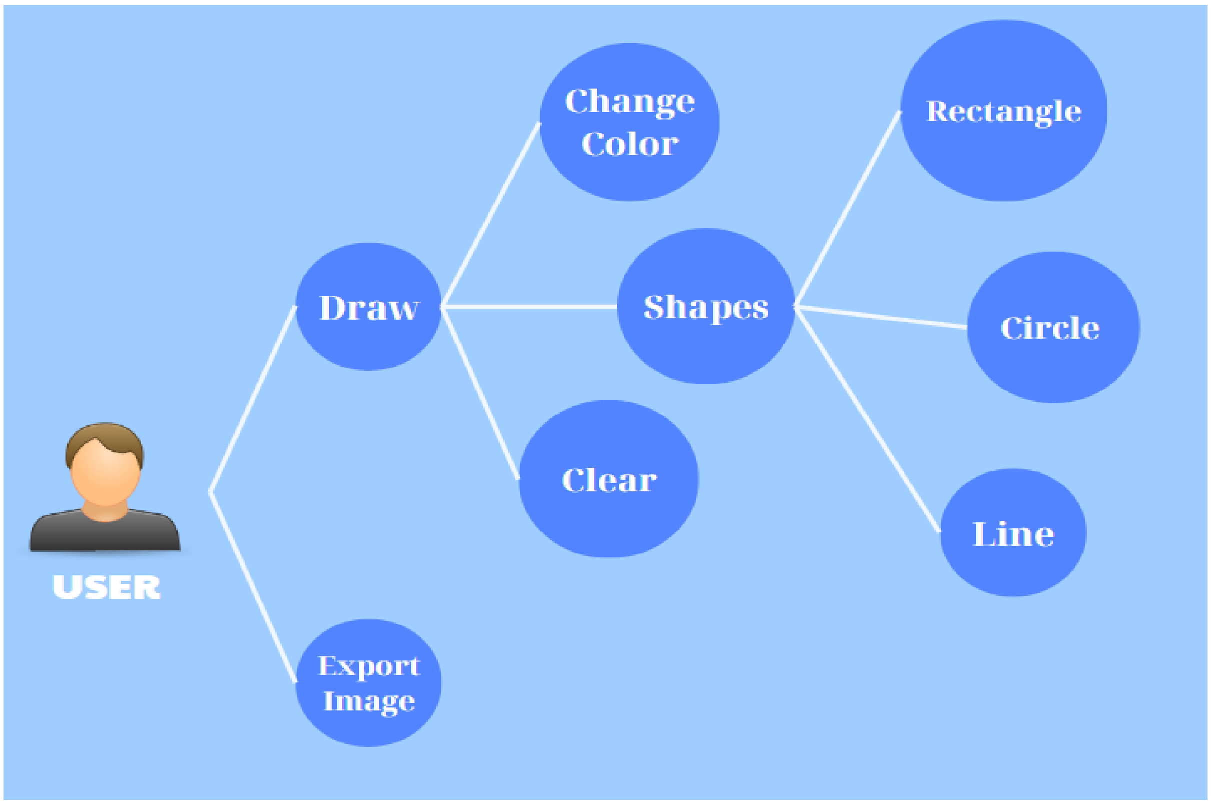
2. User benefits

Similarly to other drawing application programs, the Collaborative Whiteboard provides users with the abilitie to draw objects. Essential drawing functions are:

- Shape
- Select Color
- Line
- Clear

3. Use case diagram





4. Use case description tables

Table 1 – Use case Shapes description

Use case name	Shape
Description	Allows user to draw a rectangle or an ellipse on the board
Preconditions	Click one of the specific shape to draw that shape
Actor	User
Condition affecting Termination outcomes	

Table 2 – Use case Color description

Use case name	Color
Description	Allows user to choose the color of the shapes and lines
Preconditions	Click tho color button to choose the color
Actor	User
Condition affecting Termination outcomes	

Table 3 – Use case Clear description

Use case name	Clear
Description	Allows user to clear what were being drawn on the board
Preconditions	Click the Clear button to clear all shapes and lines
Actor	User
Condition affecting Termination outcomes	

II. Task Assignment

Table 4 – Work Plan

Student’s name	Evaluation contribution	Task work
Nguyen Thai Bao	100%	• Change color mode • Shape mode • Clear mode • Create a connection between server and clients • Broadcast every update from one client to the others

III. Design

1. Process description

Everything that being drawn on the application is objects; these objects have attributes that define them, such as position, colour, etc. The role of the application is rendering all those objects to a canvas. Firstly, I have to create a canvas class that extend Java Panel, which allows user to draw on the canvas.

1.1. Rendering

Jpanel is a component that include all the buttons and an drawing area, which appears as a GUI of the application.

```
public class DrawingCanvas extends Jpanel
```

To implement the drawing actions, I created a class called DrawingCanvas. Then I imported the nessary libraries to this class.

import javax.swing.*;

import java.awt.*;

import java.util.ArrayList;

Each object is stored in an list. The list has one purpose is to store all shapes' information from one client, then the list will be sent to the Server. Afterwards, the Server will send the list to other Clients to draw all shapes on their boards.

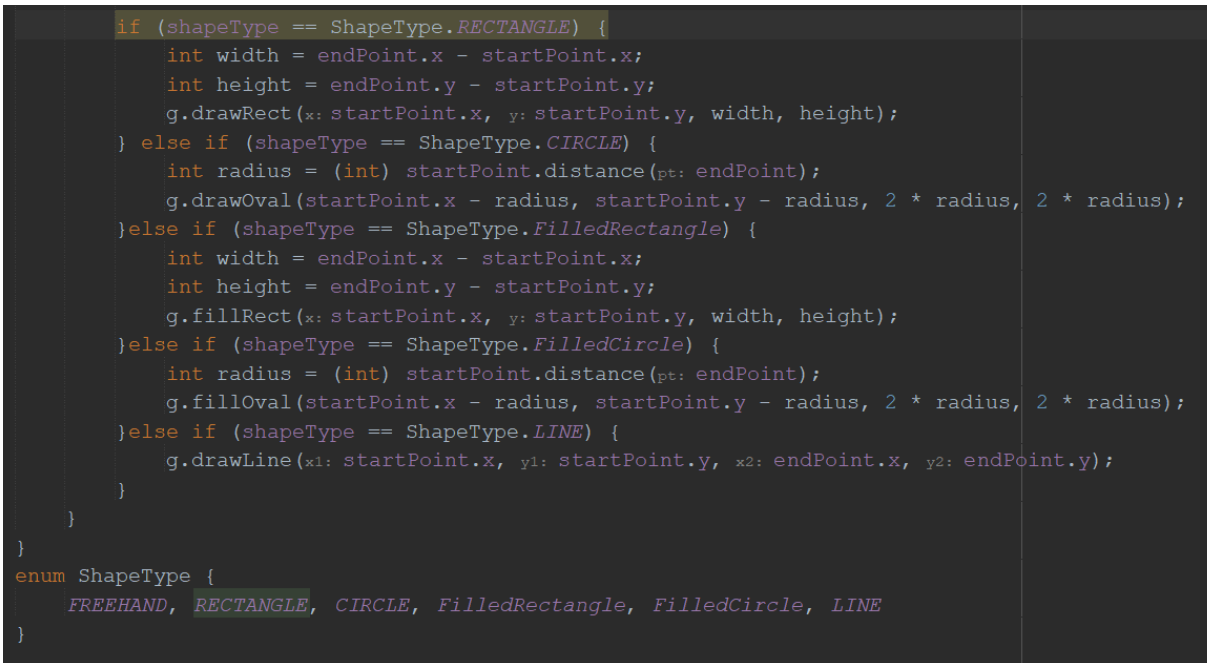
After having the GUI and a list to store the information, I created the objects by writing a new class called ShapeObject which implemented Serializable.

class ShapeObject implements Serializable

ShapeObject class defined all the information of the objects such as color, location and their type of shape. In terms of the type of shape, I create an "enum" in order to help the programe distinguish shapes easier.

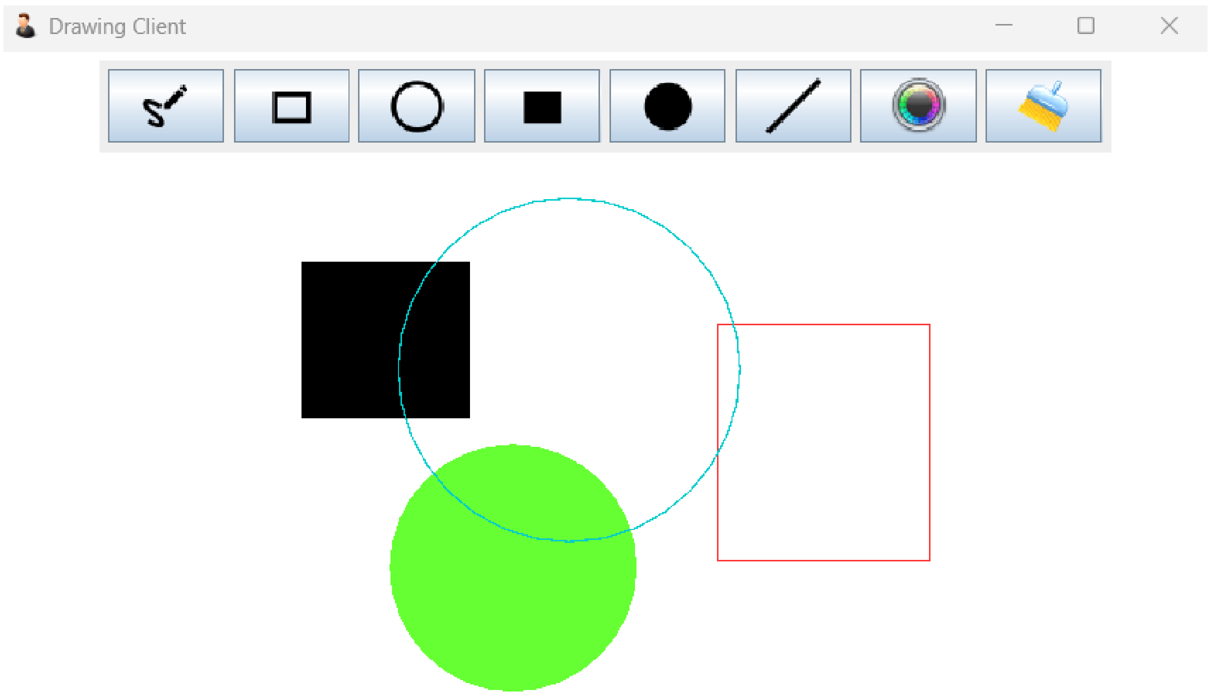
enum ShapeType {

  FREEHAND, RECTANGLE, CIRCLE, FilledRectangle, FilledCircle, LINE }



```
        if (shapeType == ShapeType.RECTANGLE) {
            int width = endPoint.x - startPoint.x;
            int height = endPoint.y - startPoint.y;
            g.drawRect(x: startPoint.x, y: startPoint.y, width, height);
        } else if (shapeType == ShapeType.CIRCLE) {
            int radius = (int) startPoint.distance(pt: endPoint);
            g.drawOval(startPoint.x - radius, startPoint.y - radius, 2 * radius, 2 * radius);
        }else if (shapeType == ShapeType.FilledRectangle) {
            int width = endPoint.x - startPoint.x;
            int height = endPoint.y - startPoint.y;
            g.fillRect(x: startPoint.x, y: startPoint.y, width, height);
        }else if (shapeType == ShapeType.FilledCircle) {
            int radius = (int) startPoint.distance(pt: endPoint);
            g.fillOval(startPoint.x - radius, startPoint.y - radius, 2 * radius, 2 * radius);
        }else if (shapeType == ShapeType.LINE) {
            g.drawLine(x1: startPoint.x, y1: startPoint.y, x2: endPoint.x, y2: endPoint.y);
        }
    }
}
enum ShapeType {
    FREEHAND, RECTANGLE, CIRCLE, FilledRectangle, FilledCircle, LINE
}
```

Now we can draw all the optional shapes to the canvas, and we have the result:

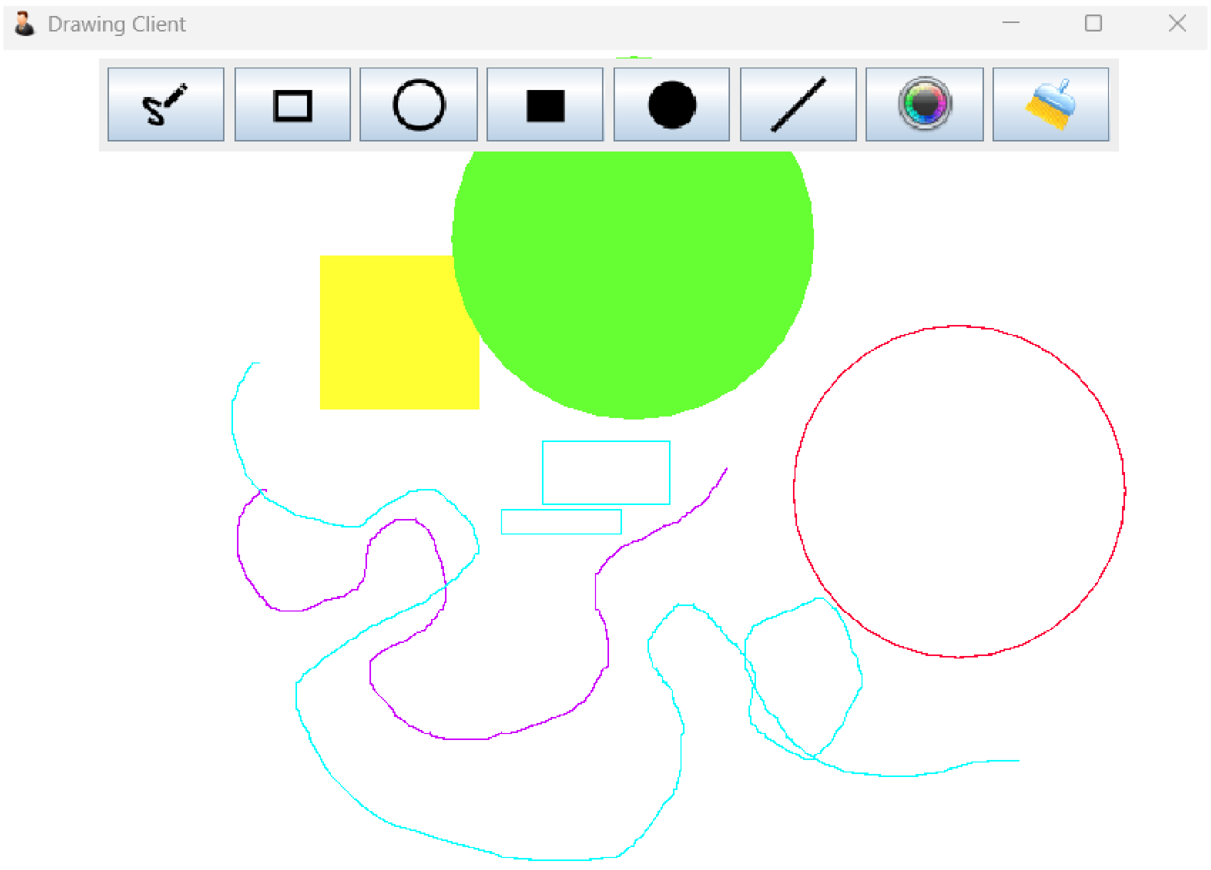


For Freeline drawing, I also create a arraylist(freehandLines) to store all the points of the freeline because the line is a sequence of points. The arraylist is included in a class called DrawingObject implementating Serializable.

```
public class DrawingObject implements Serializable
```

```
private ArrayList<ArrayList<Point>> freehandLines;
```

And the result is:



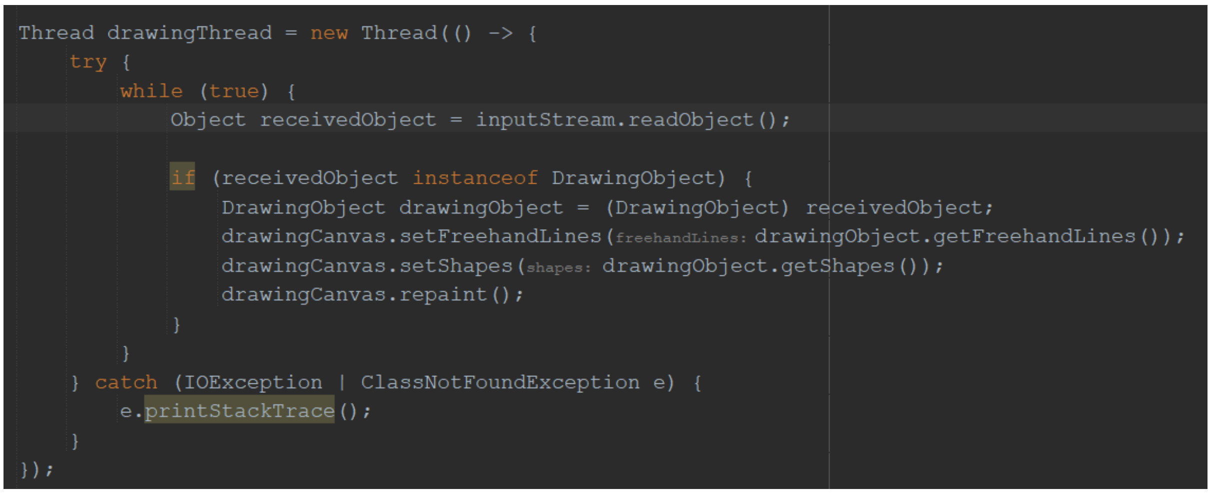
1.2 Remote drawing

To allow users to draw together, we need a server to be a middle man between users. Server receives graphics elements from a user and broadcast it to other users who are connecting to the same artboard.

In this project, I use basic Java Socket Programming technique to create a Server. Moreover, the Server also has an arraylist to handle a list of clients wwho want to connect.

private static final int PORT = 5000;

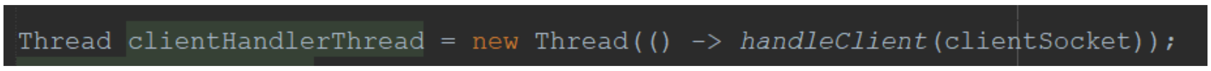
In the client-side, which is the Remote Draw application, I create a thread for reading data from the server and sending data to the server called drawingThread.



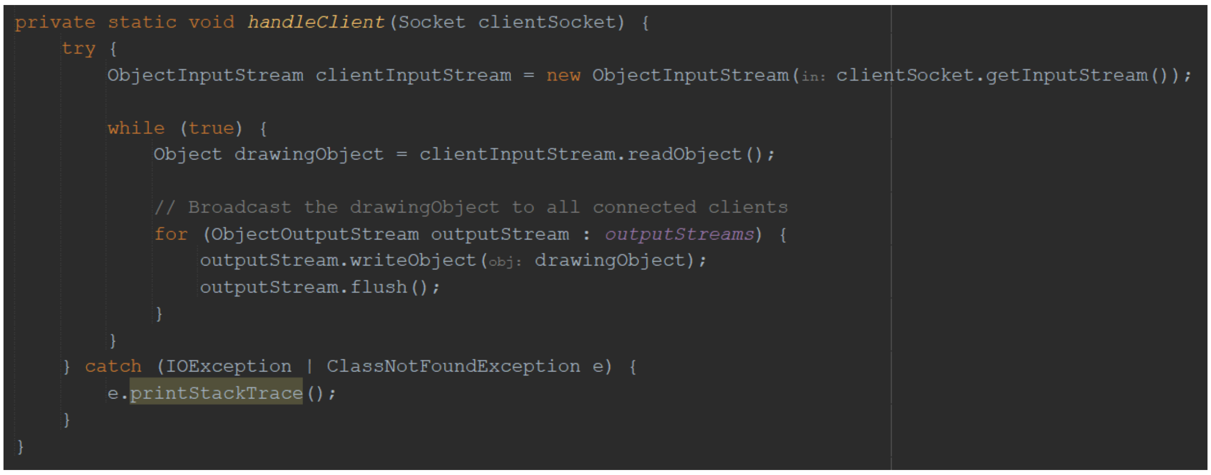
```java
Thread drawingThread = new Thread(() -> {
    try {
        while (true) {
            Object receivedObject = inputStream.readObject();

            if (receivedObject instanceof DrawingObject) {
                DrawingObject drawingObject = (DrawingObject) receivedObject;
                drawingCanvas.setFreehandLines(freehandLines: drawingObject.getFreehandLines());
                drawingCanvas.setShapes(shapes: drawingObject.getShapes());
                drawingCanvas.repaint();
            }
        }
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }
});
```

In the server-side, which is a console application, I create a thread called clientHandlerThread to handle request from a specific client.



```java
Thread clientHandlerThread = new Thread(() -> handleClient(clientSocket));
```



```java
private static void handleClient(Socket clientSocket) {
    try {
        ObjectInputStream clientInputStream = new ObjectInputStream(in: clientSocket.getInputStream());

        while (true) {
            Object drawingObject = clientInputStream.readObject();

            // Broadcast the drawingObject to all connected clients
            for (ObjectOutputStream outputStream : outputStreams) {
                outputStream.writeObject(obj: drawingObject);
                outputStream.flush();
            }
        }
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }
}
```

2. Class Design

In other to handling all features from drawing and remote drawing, object-oriented programming is the best technique helping us to implement those features. Below is the detail of all classes that I defined myself in this project, including the methods and the purposes of them.

2.1 Client

Tabel 5-List of classes are used in the client applicatiion.

No.	Class Name	Purpose
1.	ShapeObject Implement: Serializable	Serialize shapes as objects and convert objects to byte stream
2.	DrawingObject Implements: Serializable	Serialize lists of objects and points which will be sent over a network
3.	DrawingCanvas Extends: JPanel	Create a GUI for user to interact with the application by using mouse

Enum	Purpose
ShapeType	Help defining and working more effective with a collection of shapes

Tabel 6-List of methods are used in the ShapeObject class.

No.	Method	Purpose	File Name
1.	ShapeObject()	Play a role as a constructor	ShapeObject.java
2.	Draw()	Draw the shape depend on the Enum	ShapeObject.java

Tabel 7-List of methods are used in the DrawingObject class.

No.	Method	Purpose	File Name
1.	DrawingObject()	Play a role as a constructor	DrawingObject.java
2.	getFreehandLines()	Get the list of points	DrawingObject.java
3.	getShapes()	Get the list of shapes	DrawingObject.java
4.	Getcolor()	Get the information of the color	DrawingObject.java

Tabel 8-List of methods are used in the DrawingCanvas class.

No.	Method	Purpose	File Name
1.	DrawingCanvas()	Play a role as a constructor	DrawingCanvas.java
2.	mouseDragged()	Update continuously the location of points for freehand draw	DrawingCanvas.java
3.	mousePressed()	Set the location of the start point when the mouse is pressed	DrawingCanvas.java
4.	mouseReleased()	Set the location of the end point when the mouse is released	DrawingCanvas.java
5.	setupButton()	Set up the GUI for users' interaction	DrawingCanvas.java
6.	setCurrentShapeType()	Assign the type of shape after user click a specific shape button	DrawingCanvas.java
7.	addShape()	Add new shape to the Canvas	DrawingCanvas.java
8.	setOutputStream()	Assign the output stream	DrawingCanvas.java
9.	setCurrentColor()	Assign the shapes or lines' color	DrawingCanvas.java
10.	clearCanvas()	Clear the Canvas	DrawingCanvas.java
11.	sendDrawingObject()	Send and broadcast the shapes, lines, color to the stream	DrawingCanvas.java
12.	setFreehandLines()	Assign the freehand line	DrawingCanvas.java
13.	setShapes()	Assign the shape	DrawingCanvas.java
14.	paintComponent (Graphics)	A method that help drawing the freehand line	DrawingCanvas.java

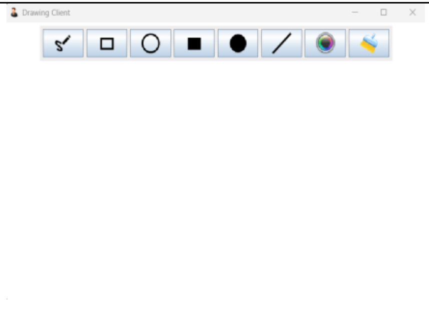
2.2. Server

Tabel 9-List of methods are used in Server class

No.	Method	Purpose	File Name
1.	Main()	Create a Server and accept new client	TCP_Server.java
2.	handleClient()	Control the clients who connect to the server	TCP_Server.java

3. Graphic User Interface

Tabel 10-GUI of the Client

GUI	Purpose	Explaination
	The main wwindow of the application	The menu on the top of the window include a group of buttons that user can choose to draw a specific shape or line.

IV. Conclusion

1. Student evaluation

- Alomost requirements are met.
- Design the application with Object-Oriented programming paradigm.
- The design of GUI is simple and easy to use.
- The application can allow multiple clients to draw together as long as they are connected the the Server.
- The code is not so clean and reusable.

2. Difficulties

- Java language is quite new to me so it takes a lot of time to be learnt. As a result, it slows down the project progress.
- Socket programming is a big problem because I was struggle in how to broadcast components among connected-clients.

3. advantages

- Sipmle GUI, easy to use.
- Allow multiple clients to connect and draw in real-time.

4. Disadvantages

- Lost some basic function such as undo and redo.
- The application cannot import images to Canvas.
- The application cannot alloew users to type text on Canvas.

5.Development ideas

- The application can be upgraded to allow users chatting and dowloading image to their computer.

References

Java Socket Programming

https://www.javatpoint.com/socket-programming

https://www.baeldung.com/a-guide-to-java-sockets

https://www.youtube.com/watch?v=1jXYeRCI98&list=PLyxSzL3F7485hAzjQI7JrkcoaYmqoPn-5

Java Language

https://www.youtube.com/watch?v=xfOp0izFnu0&list=PLyxSzL3F748401hWFgJ8gKMnN6MM8QQ7F