

**HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION  
FACULTY FOR HIGH QUALITY TRAINING  
INFORMATION TECHNOLOGY**

---



# **THE FIRST PROJECT REPORT**

## **REMOTE DRAW**

**LECTURER NAME : Dr. Nguyen Dang Quang**  
**STUDENT NAME : Le Duc Thinh**  
**STUDENT ID : 17110076**  
**STUDENT NAME : Nguyen Hoang Danh**  
**STUDENT ID : 17110009**  
**CLASS : 17110CLA**

*Ho Chi Minh City, December 2019*



## Acknowledgment

We express our sincere thanks to **Mr. Nguyen Dang Quang**, our project in charge, who guided us through the project. He gave valuable suggestions and guidance for completing the project, helped us to understand the intricate issues involved in project-making besides effectively presenting it. These intricacies would have been lost otherwise. Our project has been a success only because of his guidance.

Projects are made within ten weeks, just enough to complete it. However, due to much new knowledge as well as the time we do through each week is not optimal, the project will have many errors, which is inevitable. We are looking forward to receiving all the comments of our teachers to help our limited knowledge better.

Sincerely thanks.

## Preface

The purpose and objective of this training and mainly the content is time-being, and with this training, we have gained some confidence regarding introducing the application. We also believe that way we gained some sorts of IT knowledge, and if we practice much and having some expertise in the field, then we will be able to survive smartly in today's competitive environment.

The effort to write the report is a partial fulfilment to complete the course. In the report, I try my best to represent all the content that we learned in a great deal in the program in a systematic and presentable order. I divided each of the topics as an individual chapter to reflect the entire topic more prominently and clearly. In reference, I have used the citation method in the entire report. Finally, I am very hopeful that the structure and topic of the report will be a useful material for all the reader, especially to the user.

## CONTENT

|                                      |    |
|--------------------------------------|----|
| Acknowledgment.....                  | 3  |
| Preface .....                        | 4  |
| List of images .....                 | 7  |
| List of tables .....                 | 7  |
| I. Project description .....         | 8  |
| 1. Objectives.....                   | 8  |
| 2. User benefits .....               | 8  |
| 3. Use case diagram .....            | 9  |
| 4. Use case description tables ..... | 11 |
| II. Task Assignment .....            | 13 |
| III. Design.....                     | 14 |
| 1. Process description.....          | 14 |
| 1.1. Rendering .....                 | 15 |
| 1.2. Remote drawing .....            | 20 |
| 2. Class Design.....                 | 21 |
| 2.1. Client .....                    | 22 |
| 2.2. Server .....                    | 31 |
| 3. Graphic User Interface .....      | 35 |
| IV. Test cases.....                  | 36 |
| V. Conclusion .....                  | 36 |
| 1. Student evaluation.....           | 36 |
| 2. Difficulties .....                | 37 |
| 3. Advantages.....                   | 37 |
| 4. Disadvantages .....               | 37 |

|                           |    |
|---------------------------|----|
| 5. Development ideas..... | 37 |
| References .....          | 38 |

## List of images

|  |    |
|--|----|
| Image 1 – Use Case Diagram .....                 | 10 |
| Image 2 – Draw unfill and fill rectangle .....   | 17 |
| Image 3 – Draw a list of graphics objects .....  | 20 |
| Image 4 – Application network architecture ..... | 21 |

## List of tables

|   |    |
|---|----|
| Table 1 – Use case Draw Shape description .....                     | 11 |
| Table 2 – Use case Pen description .....                            | 11 |
| Table 3 – Use case Invite description.....                          | 11 |
| Table 4 – Use case Join description .....                           | 11 |
| Table 5 – Use case Leave description .....                          | 12 |
| Table 6 – Use case Export description .....                         | 12 |
| Table 7 – Use case Chat description.....                            | 12 |
| Table 8 – Work Plan.....  | 13 |
| Table 9 – Work Assignment.....                                      | 14 |
| Table 10 – List of classes are used in the client application ..... | 22 |
| Table 11 – List of methods in DrawingObject class .....             | 23 |
| Table 12 – List of methods on Shape class .....                     | 23 |
| Table 13 – List of methods of Rectangle class.....                  | 24 |
| Table 14 – List of methods of Ellipse class.....                    | 25 |
| Table 15 – List of methods of Eraser class.....                     | 25 |
| Table 16 – List of methods of Stroke class .....                    | 26 |
| Table 17 – List of methods of Infrastructure class .....            | 26 |
| Table 18 – List of methods of Drawing class.....                    | 29 |

|   |    |
|---|----|
| Table 19 – List of methods of ReadThread class .....                | 31 |
| Table 20 – List of methods of WriteThread class .....               | 31 |
| Table 21 – List of classes are used in the server application ..... | 31 |
| Table 22 – List of methods of Artboard class .....                  | 32 |
| Table 23 – List of methods of Client class .....                    | 32 |
| Table 24 – List of methods of Datasource class.....                 | 33 |
| Table 25 – List of methods of UserThread class .....                | 33 |
| Table 26 – GUI explanation .....                                    | 35 |
| Table 27 – Test cases.....  | 36 |

## I. Project description

### 1. Objectives

The drawing board provides users abilities to customise their drawings; we could find some applications such as Windows Paint, Artweaver, GIMP, etc. However, these programs only available for one user at a time. **Remote Draw** 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 Remote Draw provides users with the abilities to draw, move, and modify graphics objects. Essential drawing functions are:

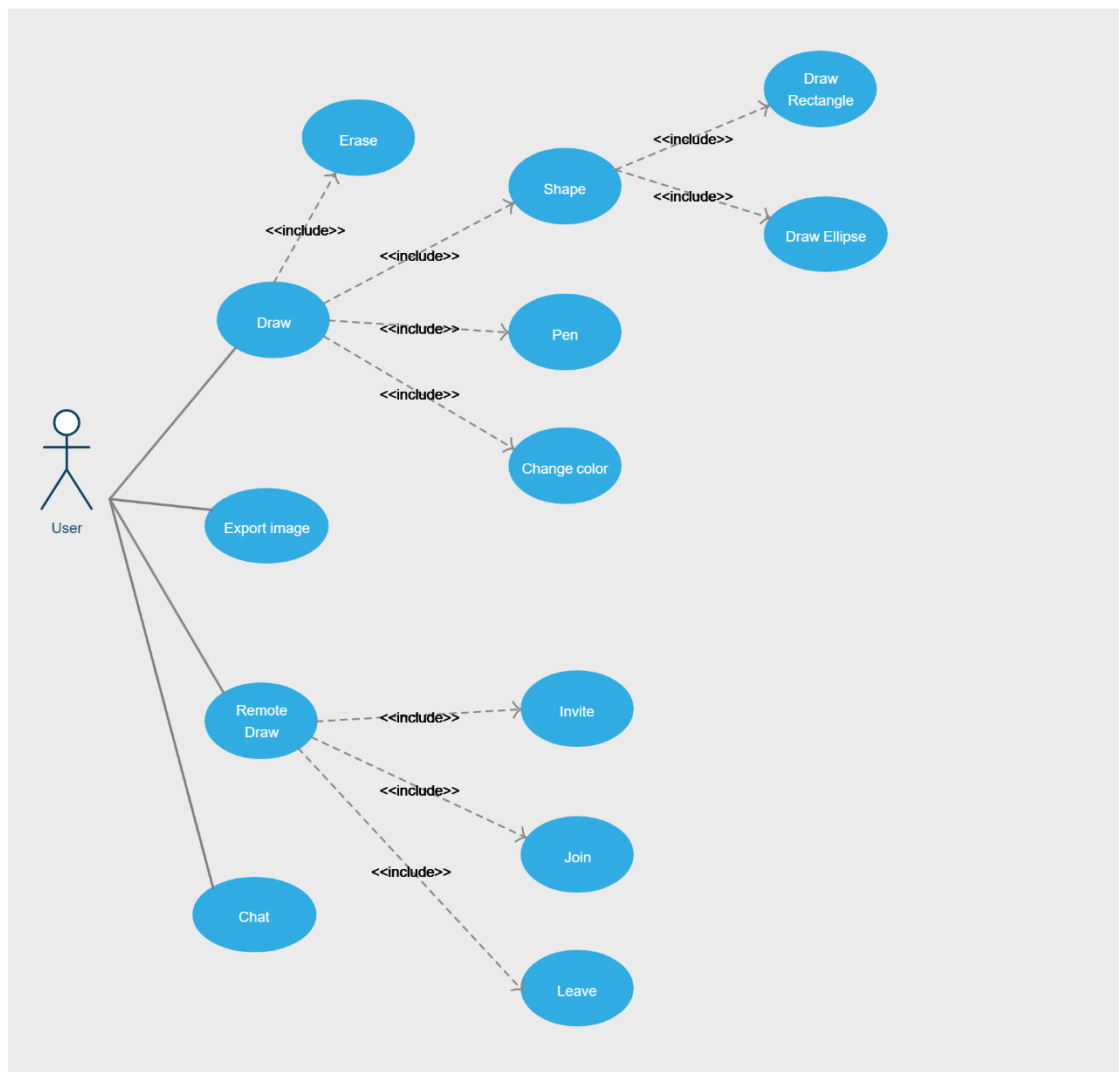
- Select: Select any shape and stroke on the artboard and move them to another position.
- Undo: Return to the previous action.
- Change colour: User can change colour with HSB, RGB and Web format.
- Change the thickness of a stroke or border width of a shape.



- Erase: Clear everything when an eraser goes through, and the user can change the size of an eraser.
- Pen: Draw strokes to canvas.
- Shape: Draw rectangle and ellipse.
- Export: Export the artboard into the PNG format.

With the ability to connect to the Internet, Remote Draw application also allows users to invite one or more friends to draw together. Furthermore, after connecting, users and their friends can chat with each other to spread their ideas before drawing something.

### 3. Use case diagram



*Image 1 – Use Case Diagram*

#### 4. Use case description tables

*Table 1 – Use case Draw Shape description*

|  |  |
|--|--|
| Use case name                            | Draw Shape   |
| Description                              | Allows user to draw a rectangle or an ellipse on the board |
| Actor                                    | User   |
| Preconditions                            | Click the Shape button and choose a specific shape         |
| Conditions affecting termination outcome |  |

*Table 2 – Use case Pen description*

|  |   |
|--|---|
| Use case name                            | Pen   |
| Description                              | Allows user to draw any stroke on the board |
| Actor                                    | User  |
| Preconditions                            | Click Pen button                            |
| Conditions affecting termination outcome |   |

*Table 3 – Use case Invite description*

|  |  |                     |   |
|--|--|---------------------|---|
| Use case name                            | Invite   |                     |   |
| Description                              | Allows user to invite other people to draw together  |                     |   |
| Actor                                    | User   |                     |   |
| Business event                           | No.  | Agent               | System  |
|  | 1  | Click Invite button |   |
|  | 2  |                     | A connection is established between user and server |
|  | 3  |                     | The server sends a code to the user                 |
|  | 4  |                     | The system displays the code to the chatbox         |
| Preconditions                            | Server is running  |                     |   |
| Conditions affecting termination outcome | The server is running, the connection is established successfully<br>The server is terminated, connection failed |                     |   |

*Table 4 – Use case Join description*

|                |                                       |                   |  |
|----------------|---------------------------------------|-------------------|--|
| Use case name  | Join                                  |                   |  |
| Description    | Allows user to join to a remote board |                   |  |
| Actor          | User                                  |                   |  |
| Business event | No.                                   | Agent             | System   |
|                | 1                                     | Click Join button |  |
|                | 2                                     |                   | A connection is established between user and server              |
|                | 3                                     |                   | The server sends all information of the remote board to the user |

|  |  |  |   |
|--|--|--|---|
|  | 4  |  | Render the drawing information to the board |
| Preconditions                            | Server is running  |  |   |
| Conditions affecting termination outcome | The server is running, the connection is established successfully<br>The server is terminated, connection failed |  |   |

*Table 5 – Use case Leave description*

|  |                                 |                    |                          |
|--|---------------------------------|--------------------|--------------------------|
| Use case name                            | Leave                           |                    |                          |
| Description                              | Disconnect to server            |                    |                          |
| Actor                                    | User                            |                    |                          |
| Business event                           | No.                             | Agent              | System                   |
|  | 1                               | Click Leave button |                          |
|  | 2                               |                    | Disconnect to the server |
| Preconditions                            | User is connected to the server |                    |                          |
| Conditions affecting termination outcome |                                 |                    |                          |

*Table 6 – Use case Export description*

|  |  |                          |   |
|--|--|--------------------------|---|
| Use case name                            | Export   |                          |   |
| Description                              | Exports the current state of the board to a png file |                          |   |
| Actor                                    | User   |                          |   |
| Business event                           | No.  | Agent                    | System  |
|  | 1  | Click the export button  |   |
|  | 2  |                          | Show a dialogue for the user to choose the directory to store the image |
|  | 3  | Choose a specific folder |   |
|  | 4  |                          | Export the image  |
| Preconditions                            |  |                          |   |
| Conditions affecting termination outcome |  |                          |   |

*Table 7 – Use case Chat description*

|                |  |                                 |   |
|----------------|--|---------------------------------|---|
| Use case name  | Chat                                       |                                 |   |
| Description    | Allow users to communicate with each other |                                 |   |
| Actor          | User                                       |                                 |   |
| Business event | No.  | Agent                           | System  |
|                | 1  | Enter the message to a text box |   |
|                | 2  | Click Send button               |   |
|                | 3  |                                 | Send the message to the server to broadcast to other people that are connected to the same artboard |
|                | 4  |                                 | Display the message to the chatbox  |

|  |                                 |
|--|---------------------------------|
| Preconditions                            | User is connected to the server |
| Conditions affecting termination outcome |                                 |

## II. Task Assignment

*Table 8 – Work Plan*

| Student's name    | Evaluate contribution | Taskwork  |
|-------------------|-----------------------|---|
| Nguyen Hoang Danh | 100%                  | Select mode<br>Undo mode<br>Change colour mode<br>Pencil mode<br>Eraser mode<br>Shape mode<br>Export mode |
| Nguyen Hoang Danh | 100%                  | Create a connection to server   |
| Le Duc Thinh      | 100%                  | Create a server to store every graphic element that is drawn by a specified client                        |
| Le Duc Thinh      | 100%                  | Broadcast every update from a client to others  |
| Le Duc Thinh      | 100%                  | Chat function   |

Table 9 – Work Assignment

| Building an Remote Draw software using Java |   |            |            |  |            |            |            |            |            |            |            |      |       |
|---|---|------------|------------|--|------------|------------|------------|------------|------------|------------|------------|------|-------|
| No.   | Goal                                      | Schedule   |            |  |            |            |            |            |            |            |            | Danh | Thinh |
| 1   | Understand Requirement.                   | o          | o          |  |            |            |            |            |            |            |            | ✓    | ✓     |
| 2   | Describe the requirements of the project. | o          | o          |  |            |            |            |            |            |            |            | ✓    | ✓     |
| 3   | Learn Java language.                      |            | o          | o  | o          |            |            |            |            |            |            | ✓    | ✓     |
| 4   | Learn network programming.                |            | o          | o  | o          |            |            |            |            |            |            |      | ✓     |
| 5   | Building software architecture.           |            | o          | o  | o          |            |            |            |            |            |            | ✓    | ✓     |
| 6   | User interface design.                    |            |            | o  | o          |            |            |            |            |            |            |      | ✓     |
| 7   | Design classes.                           |            |            | o  | o          | o          |            |            |            |            |            | ✓    | ✓     |
| 8   | Build methods.                            |            |            |  | o          | o          |            |            |            |            |            | ✓    | ✓     |
| 9   | Program Implementation.                   |            |            |  |            | o          | o          | o          | o          | o          |            | ✓    | ✓     |
| 10  | Optimize Code.                            |            |            |  |            |            |            | o          | o          | o          |            | ✓    | ✓     |
| 11  | Testing.                                  |            |            |  |            |            |            | o          | o          | o          | o          | ✓    | ✓     |
| 12  | Write report.                             |            |            |  |            |            |            | o          | o          | o          | o          | ✓    | ✓     |
| Day   |   | 07/10/2019 | 14/10/2019 | 21/10/2019   | 28/10/2019 | 04/11/2019 | 11/11/2019 | 18/11/2019 | 25/11/2019 | 02/12/2019 | 09/12/2019 |      |       |
| Week  |   | 1          | 2          | 3  | 4          | 5          | 6          | 7          | 8          | 9          | 10         |      |       |
| Note  |   |            |            | o – Begin<br>o – Complete 50%<br>o – Complete 100% |            |            |            |            |            |            |            |      |       |

### III. Design

#### 1. Process description

Everything drawn on the application is objects; these objects have attributes that define them, such as position, colour, etc. The role of application is rendering all those

objects to a canvas. In java programming language, it provides us with a library called GraphicsContext. This library takes inputs which are the information of an object and draw it to a canvas. Canvas is also a library, its role is a UI element, and it works with GraphicsContext to display graphic elements.

## 1.1. Rendering

GraphicsContext and Canvas libraries help the application to draw every object to canvas, such as a line, a rectangle, an ellipse, a triangle, a circle, a polygon, etc. Remote Draw application focuses on three objects only that is a line, a rectangle, and an ellipse.

To perform all drawing actions easier, we created a class called **Drawing**. Then we need to import the necessary libraries to that class. This class receive information about [canvas](#) from User Interface.

```
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
```

Create a new GraphicsContext instance.

```
public Drawing(Canvas canvas) {
    this.canvas = canvas;
    this.graphicsContext = this.canvas.getGraphicsContext2D();
    this.graphicElements = new ArrayList<>();
    this.selectionGraphicElements = new ArrayList<>();
}
```

Each object is stored in an list. This list has two purpose, the first one is providing essential information to Drawing class and Drawing class can draw objects to canvas, the second one is to send this list over the Internet, other Remote Draw application will receive it and render all graphics elements.

After having the tool is GraphicsContext and Canvas library, we need objects. For example, we have a [Rectangle](#) class which is inherited from [Shape](#) abstract class. This class defines all the information of a rectangle.

```
public class Shape extends DrawingObject {
```

```

//region Private Attributes
private double x1;
private double y1;
private double x2;
private double y2;
boolean fill;
private String color;
private int thickness;
//endregion

//region Constructor
public Shape(double x1, double y1, double x2, double y2, boolean fill, String
color, int thickness) {
    this.x1 = x1;
    this.y1 = y1;
    this.x2 = x2;
    this.y2 = y2;
    this.fill = fill;
    this.color = color;
    this.thickness = thickness;
}
}

public class Rectangle extends Shape {

//region Private Attributes
private String type;
private double width;
private double height;
//endregion

//region Constructor
public Rectangle(double x1, double y1, double x2, double y2, boolean fill, String
color, int thickness, String type) {
    super(x1, y1, x2, y2, fill, color, thickness);

```



```

        this.type = type;
    }
}

```

Now we can draw rectangles to a canvas using below command

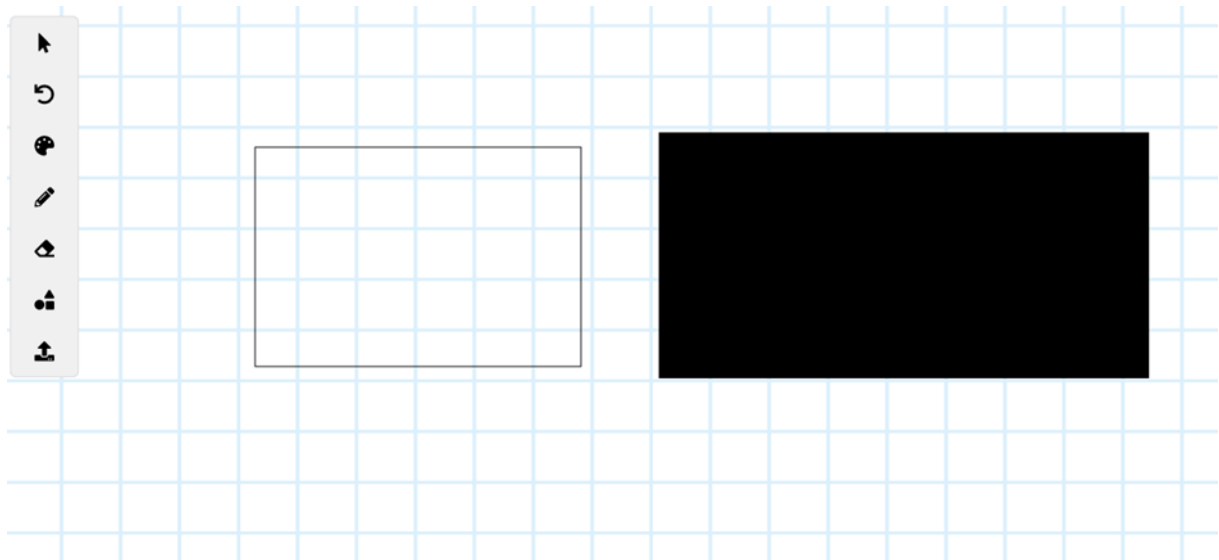
```

graphicsContext.fillRect(rectangle.getX1(), rectangle.getY1(), rectangle.getWidth(),
rectangle.getHeight());

graphicsContext.strokeRect(rectangle.getX1(), rectangle.getY1(),
rectangle.getWidth(), rectangle.getHeight());

```

And we have the result.



*Image 2 – Draw unfill and fill rectangle*

Similarly we can draw a list of graphics elements such as a ellipse and a stroke, note that a stroke is a set of lines. Everytime we draw an new object to the canvas, we add it to the list called `graphicsElements`, so that the Drawing class just need to render that list.

```

public void Render() {
    graphicsContext.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());

    for(Object object: graphicElements) {
        if(object instanceof Rectangle) {
            Rectangle rectangle = ((Rectangle) object);
            if(((Rectangle) object).isFill()) {
                graphicsContext.setFill(Color.valueOf(rectangle.getColor()));
            }
        }
    }
}

```

```

        graphicsContext.fillRect(rectangle.getX1(), rectangle.getY1(),
rectangle.getWidth(), rectangle.getHeight());
    } else {
        graphicsContext.setStroke(Color.valueOf(rectangle.getColor()));
        graphicsContext.setLineWidth(rectangle.getThickness());
        graphicsContext.strokeRect(rectangle.getX1(), rectangle.getY1(),
rectangle.getWidth(), rectangle.getHeight());
    }
}

if(object instanceof Stroke) {
    Stroke stroke = ((Stroke) object);
    graphicsContext.setStroke(Color.valueOf(stroke.getColor()));
    graphicsContext.setLineWidth(stroke.getPenSize());
    for(int i=0; i<stroke.getPath().size()-3; i+=2) {
        graphicsContext.strokeLine(stroke.getPath().get(i),
stroke.getPath().get(i+1), stroke.getPath().get(i+2), stroke.getPath().get(i+3));
    }
}

if(object instanceof Ellipse) {
    Ellipse ellipse = ((Ellipse) object);
    if(ellipse.isFill()) {
        graphicsContext.setFill(Color.valueOf(ellipse.getColor()));
        graphicsContext.fillOval(ellipse.getX1(), ellipse.getY1(),
ellipse.getWidth(), ellipse.getHeight());
    } else {
        graphicsContext.setStroke(Color.valueOf(ellipse.getColor()));
        graphicsContext.setLineWidth(ellipse.getThickness());
        graphicsContext.strokeOval(ellipse.getX1(), ellipse.getY1(),
ellipse.getWidth(), ellipse.getHeight());
    }
}

if(object instanceof Eraser) {

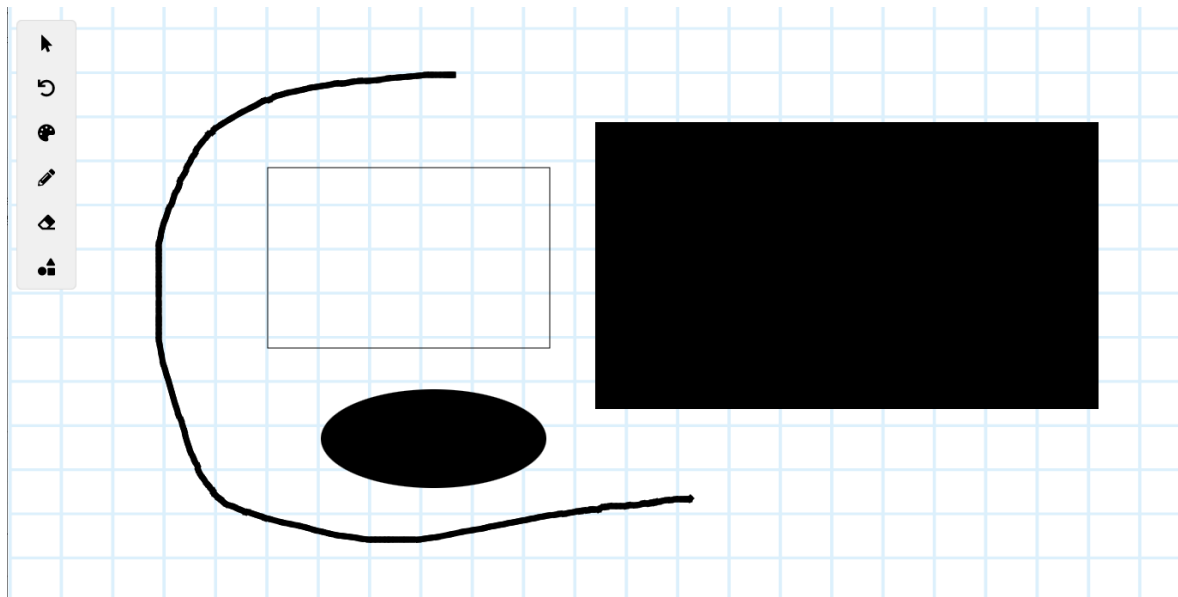
```

```

        Eraser eraser = ((Eraser) object);
        for(int i=0; i<eraser.getPath().size()-2; i+=2) {
            graphicsContext.clearRect(eraser.getPath().get(i),
eraser.getPath().get(i+1), eraser.getEraserSize(), eraser.getEraserSize());
        }
        if(!isFinishedErasing) {
            if(graphicElements.get(graphicElements.size() - 1) instanceof Eraser) {
                Eraser latestEraser = ((Eraser)
graphicElements.get(graphicElements.size() - 1));
                int size = latestEraser.getPath().size();
                graphicsContext.setLineWidth(3);
                graphicsContext.setStroke(Color.RED);
                graphicsContext.strokeRect(latestEraser.getPath().get(size-2),
latestEraser.getPath().get(size-1), latestEraser.getEraserSize(),
latestEraser.getEraserSize());
            }
        }
    }
}
}
}
}

```

And below is the result.



*Image 3 – Draw a list of graphics objects*

## 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.

Java.net package allows us to open a connection between a client to a server. The package contains two sets of APIs: the low-level API, and the high-level API. Low-level API deals with port, IP address while high-level deals with HTTP/HTTPS.

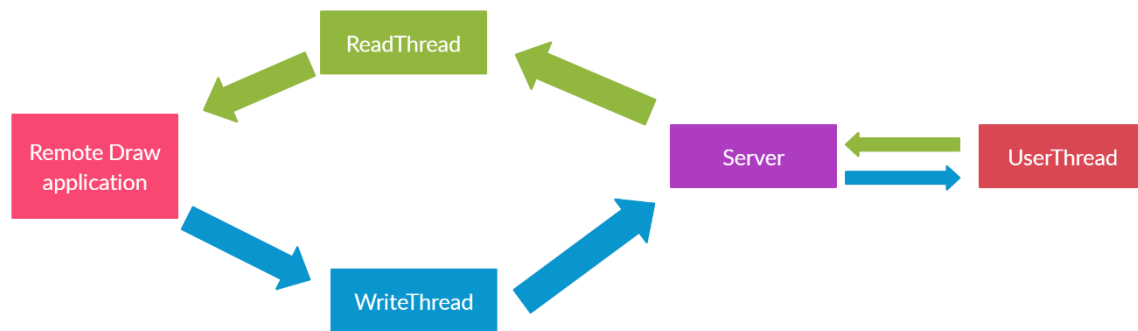
In this project, we use the low-level API to establish a connection between client and server. Java.net.socket library helps us to communicate with a server by an IP address and a port.

```
Socket socket = new Socket("127.0.0.1", 5000);
```

The above command is used to connect to a computer with the IP address is “127.0.0.1” which is the local computer, and the port is 5000. IP address specifies which computer on the Internet it should connect to and port clarifies which application in that computer it should connect to.

In the client-side, which is the Remote Draw application, we create two thread for reading data from the server and sending data to the server called `ReadThread` and `WriteThread`.

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



*Image 4 – Application network architecture*

So users want to draw with their friends, what information do they need to send to the server? Every time a user (host) wants to invite someone (guest) to draw with him, he needs to send all his current graphics elements to the server. The server stores that information plus a code, this code is unique for each host. The server will send this code back to the host and tell him that you need to share this code to your friends to start drawing together. Guest will send this code to the server and receive all drawing information of the host from the server. From now on, every modification of any graphics elements server will know and broadcast that changes to other users.

After successfully allowing users to draw remotely, we also allow users to send messages like a chat application to make their work more comfortable.

## ***2. Class Design***

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

We divided all classes in this application into two groups. The first group contains all classes of the client (Remote Draw), and the second one consists of all classes of the server.

## 2.1. Client

List of classes are used in the client application

*Table 10 – List of classes are used in the client application*

| No. | Class Name                               | Responsible               | Purpose   |
|-----|--|---------------------------|---|
| 1   | DrawingObject<br>Implement: Serializable | Le Duc Thinh              | We can make every child of this class such as shape and stroke be serialisable (convert to byte stream)   |
| 2   | Shape<br>Extend: DrawingObject           | Nguyen Hoang Danh         | An abstract class that defines every shape on the canvas such as rectangle and ellipse                    |
| 3   | Rectangle<br>Extend: Shape               | Nguyen Hoang Danh         | Define any rectangle that is drawn on the canvas  |
| 4   | Ellipse<br>Extend: Shape                 | Nguyen Hoang Danh         | Define any ellipse that is drawn on the canvas  |
| 5   | Eraser<br>Extend: Shape                  | Nguyen Hoang Danh         | Define any erasing path on the canvas   |
| 6   | Stroke<br>Extend: DrawingObject          | Nguyen Hoang Danh         | Define any path of a pen that is drawn on the canvas  |
| 7   | Infrastructure                           | Le Duc Thinh              | Define essential information about drawing and connection of the application                              |
| 8   | Drawing                                  | Nguyen Hoang Danh         | Store and draw graphic elements like shape and stroke to the canvas                                       |
| 9   | ReadThread                               | Le Duc Thinh              | Read every information coming from the server then transfer and store information to Infrastructure class |
| 10  | WriteThread                              | Le Duc Thinh              | Send requests from the client to server   |
| 11  | DrawingObjectsToStringConverter          | <a href="#">Reference</a> | Convert drawing objects to a byte stream  |

List of methods in DrawingObject class

*Table 11 – List of methods in DrawingObject class*

| No. | Method  | Purpose  | File name, Line            | Responsible  |
|-----|---|--|----------------------------|--------------|
| 1   | <b>equals(Object obj)</b><br>Input: obj.<br>Output: boolean.<br>Pseudocode: none. | Compare DrawingObjects   | DrawingObject.java<br>(8)  | Le Duc Thinh |
| 2   | <b>toString()</b><br>Input: none<br>Output: String<br>PseudoCode: none            | Print out the default information of an instance of DrawingObject to the console | DrawingObject.java<br>(13) | Le Duc Thinh |

List of methods in Shape class

*Table 12 – List of methods on Shape class*

| No. | Method  | Purpose  | File name, Line    | Responsible       |
|-----|---|--|--------------------|-------------------|
| 1   | <b>setX1(double x1)</b><br>Input: x1.<br>Output: none.<br>Pseudocode: none. | Assign the X position of the upper left corner of a shape  | Shape.java<br>(36) | Nguyen Hoang Danh |
| 2   | <b>getX1()</b><br>Input: none.<br>Output: double.<br>Pseudocode: none.      | Get the X position of the upper left corner of a shape     | Shape.java<br>(32) | Nguyen Hoang Danh |
| 3   | <b>setY1(double y1)</b><br>Input: y1.<br>Output: none.<br>Pseudocode: none. | Assign the Y position of the upper left corner of a shape  | Shape.java<br>(44) | Nguyen Hoang Danh |
| 4   | <b>getY1()</b><br>Input: none.<br>Output: double.<br>Pseudocode: none.      | Get the Y position of the upper left corner of a shape     | Shape.java<br>(40) | Nguyen Hoang Danh |
| 5   | <b>setX2(double x2)</b><br>Input: x2.<br>Output: none.<br>Pseudocode: none. | Assign the X position of the lower right corner of a shape | Shape.java<br>(52) | Nguyen Hoang Danh |
| 6   | <b>getX2()</b><br>Input: none.<br>Output: double.<br>Pseudocode: none.      | Get the X position of the lower right corner of a shape    | Shape.java<br>(48) | Nguyen Hoang Danh |
| 7   | <b>setY2(double y2)</b><br>Input: y2.<br>Output: none.<br>Pseudocode: none. | Assign the Y position of the lower right corner of a shape | Shape.java<br>(60) | Nguyen Hoang Danh |

|    |  |  |                 |                   |
|----|--|--|-----------------|-------------------|
| 8  | <b>getY2()</b><br>Input: none.<br>Output: double.<br>Pseudocode: none.                       | Get the Y position of the lower right corner of a shape  | Shape.java (56) | Nguyen Hoang Danh |
| 9  | <b>isFill()</b><br>Input: none.<br>Output: boolean.<br>Pseudocode: none.                     | Check whether the state of a shape is filled or unfilled | Shape.java (64) | Nguyen Hoang Danh |
| 10 | <b>setFill(boolean fill)</b><br>Input: fill.<br>Output: none.<br>Pseudocode: none.           | Set the fill value                                       | Shape.java (68) | Nguyen Hoang Danh |
| 11 | <b>getColor()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none.                    | Get the color of border and fill color of a shape        | Shape.java (72) | Nguyen Hoang Danh |
| 12 | <b>setColor(String color)</b><br>Input: color.<br>Output: String.<br>Pseudocode: none.       | Set the color of border and fill color of a shape        | Shape.java (76) | Nguyen Hoang Danh |
| 13 | <b>getThickness()</b><br>Input: none.<br>Output: int.<br>Pseudocode: none                    | Get the border width of a shape                          | Shape.java (80) | Nguyen Hoang Danh |
| 14 | <b>setThickness(int thickness)</b><br>Input: thickness<br>Output: none.<br>Pseudocode: none. | Set the border width of a shape                          | Shape.java (84) | Nguyen Hoang Danh |

List of methods of Rectangle class

*Table 13 – List of methods of Rectangle class*

| No. | Method   | Purpose   | File name, Line     | Responsible       |
|-----|--|---|---------------------|-------------------|
| 1   | <b>getWidth()</b><br>Input: none.<br>Output: double.<br>Pseudocode: none.  | Get the width of a rectangle  | Rectangle.java (33) | Nguyen Hoang Danh |
| 2   | <b>getHeight()</b><br>Input: none.<br>Output: double.<br>Pseudocode: none. | Get the height of a rectangle   | Rectangle.java (38) | Nguyen Hoang Danh |
| 3   | <b>isInsideRectangle(double x, double y)</b><br>Input: x, y.               | Check whether a specific position on the canvas is inside a rectangle | Rectangle.java (43) | Nguyen Hoang Danh |



|   |   |   |                        |                   |
|---|---|---|------------------------|-------------------|
|   | Output: boolean<br>Pseudocode: none   |   |                        |                   |
| 4 | <b>isOnRectangle(double x, double y)</b><br>Input: x, y.<br>Output: boolean<br>Pseudocode: none | Check whether a specific position on the canvas is on a rectangle | Rectangle.java<br>(50) | Nguyen Hoang Danh |

List of methods of Ellipse class

*Table 14 – List of methods of Ellipse class*

| No. | Method  | Purpose  | File name, Line      | Responsible       |
|-----|---|--|----------------------|-------------------|
| 1   | <b>getWidth()</b><br>Input: none.<br>Output: double.<br>Pseudocode: none.                         | Get the width of an ellipse  | Ellipse.java<br>(24) | Nguyen Hoang Danh |
| 2   | <b>getHeight()</b><br>Input: none.<br>Output: double.<br>Pseudocode: none.                        | Get the height of an ellipse   | Ellipse.java<br>(29) | Nguyen Hoang Danh |
| 3   | <b>isInsideEllipse(double x, double y)</b><br>Input: x, y.<br>Output: boolean<br>Pseudocode: none | Check whether a specific position on the canvas is inside an ellipse | Ellipse.java<br>(45) | Nguyen Hoang Danh |
| 4   | <b>isOnEllipse(double x, double y)</b><br>Input: x, y.<br>Output: boolean<br>Pseudocode: none     | Check whether a specific position on the canvas is on an ellipse     | Ellipse.java<br>(34) | Nguyen Hoang Danh |

List of methods of Eraser class

*Table 15 – List of methods of Eraser class*

| No. | Method   | Purpose                           | File name, Line     | Responsible       |
|-----|--|-----------------------------------|---------------------|-------------------|
| 1   | <b>getPath()</b><br>Input: none.<br>Output: List<Double>.<br>Pseudocode: none. | Get the erasing path of an eraser | Eraser.java<br>(25) | Nguyen Hoang Danh |
| 2   | <b>getEraserSize()</b><br>Input: none.<br>Output: double<br>Pseudocode: none   | Get eraser size                   | Eraser.java<br>(33) | Nguyen Hoang Danh |

List of methods of Stroke class

*Table 16 – List of methods of Stroke class*

| No. | Method  | Purpose  | File name, Line  | Responsible       |
|-----|---|--|------------------|-------------------|
| 1   | <b>getPath()</b><br>Input: none.<br>Output: List<Double>.<br>Pseudocode: none.                | Get the stroke path  | Stroke.java (31) | Nguyen Hoang Danh |
| 2   | <b>getColor()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none                      | Get color of a stroke  | Stroke.java (39) | Nguyen Hoang Danh |
| 3   | <b>getPenSize()</b><br>Input: none.<br>Output: int.<br>Pseudocode: none                       | Get the thickness of a stroke                                  | Stroke.java (47) | Nguyen Hoang Danh |
| 4   | <b>isOnStroke(double x, double y)</b><br>Input: x, y.<br>Output: boolean.<br>Pseudocode: none | Check whether a specific position on the canvas is on a stroke | Stroke.java (71) | Nguyen Hoang Danh |

List of methods of Infrastructure class

*Table 17 – List of methods of Infrastructure class*

| No. | Method  | Purpose   | File name, Line          | Responsible  |
|-----|---|---|--------------------------|--------------|
| 1   | <b>getColor()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none.     | Get selected color in color menu                  | Infrastructure.java (35) | Le Duc Thinh |
| 2   | <b>getThickness()</b><br>Input: none.<br>Output: int.<br>Pseudocode: none.    | Get thickness value selected from Draw Shape menu | Infrastructure.java (39) | Le Duc Thinh |
| 3   | <b>getMode()</b><br>Input: none.<br>Output: DrawingMode.<br>Pseudocode: none. | Get the selected function of the application      | Infrastructure.java (43) | Le Duc Thinh |
| 4   | <b>getPenSize()</b><br>Input: none.<br>Output: int.<br>Pseudocode: none.      | Get pen size value selected from Pen menu         | Infrastructure.java (47) | Le Duc Thinh |
| 5   | <b>getEraserSize()</b><br>Input: none.<br>Output: int.<br>Pseudocode: none.   | Get eraser size value selected from Eraser menu   | Infrastructure.java (51) | Le Duc Thinh |

|    |  |  |                             |              |
|----|--|--|-----------------------------|--------------|
| 6  | <b>setColor(String color)</b><br>Input: color.<br>Output: none.<br>Pseudocode: none.             | Set color used by the application                    | Infrastructure.java<br>(55) | Le Duc Thinh |
| 7  | <b>setThickness(int thickness)</b><br>Input: thickness.<br>Output: none.<br>Pseudocode: none.    | Set thickness value used by the application          | Infrastructure.java<br>(59) | Le Duc Thinh |
| 8  | <b>setMode(DrawingMode mode)</b><br>Input: mode.<br>Output: none.<br>Pseudocode: none.           | Set the current function of the application          | Infrastructure.java<br>(63) | Le Duc Thinh |
| 9  | <b>setPenSize(int penSize)</b><br>Input: penSize.<br>Output: none.<br>Pseudocode: none.          | Set the pen size value used by the application       | Infrastructure.java<br>(67) | Le Duc Thinh |
| 10 | <b>setEraserSize(int eraserSize)</b><br>Input: eraserSize.<br>Output: none.<br>Pseudocode: none. | Set the pen size value used by the application       | Infrastructure.java<br>(71) | Le Duc Thinh |
| 11 | <b>getCode()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none.                         | Get the code of the artboard according to the server | Infrastructure.java<br>(75) | Le Duc Thinh |
| 12 | <b>setCode(String code)</b><br>Input: code.<br>Output: none.<br>Pseudocode: none.                | Set the code to send to the server                   | Infrastructure.java<br>(79) | Le Duc Thinh |
| 13 | <b>getProtocol()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none                      | Get the current protocol of the communication        | Infrastructure.java<br>(83) | Le Duc Thinh |
| 14 | <b>setProtocol(String protocol)</b><br>Input: protocol.<br>Output: none.<br>Pseudocode: none.    | Set the current protocol of the communication        | Infrastructure.java<br>(87) | Le Duc Thinh |
| 15 | <b>getData()</b><br>Input: none.<br>Output: String<br>Pseudocode: none                           | Get the byte stream of the graphics elements         | Infrastructure.java<br>(91) | Le Duc Thinh |
| 16 | <b>setData(String data)</b><br>Input: data.  | Set the byte stream of the graphics elements         | Infrastructure.java         | Le Duc Thinh |

|    |   |  |                              |              |
|----|---|--|------------------------------|--------------|
|    | Output: none.<br>Pseudocode: none   |  | (95)                         |              |
| 17 | <b>getName()</b><br>Input: none.<br>Output: String<br>Pseudocode: none  | Get the name of the user                 | Infrastructure.java<br>(99)  | Le Duc Thinh |
| 18 | <b>setName(String name)</b><br>Input: name.<br>Output: none.<br>Pseudocode: none                                  | Set the name of the user                 | Infrastructure.java<br>(103) | Le Duc Thinh |
| 19 | <b>getResult()</b><br>Input: none.<br>Output: String<br>Pseudocode: none  | Get the result from the server           | Infrastructure.java<br>(107) | Le Duc Thinh |
| 20 | <b>setResult(String result)</b><br>Input: result.<br>Output: none.<br>Pseudocode: none                            | Set the result value                     | Infrastructure.java<br>(111) | Le Duc Thinh |
| 21 | <b>setNotification(String notification)</b><br>Input: notification.<br>Output: none.<br>Pseudocode: none          | Set notification for the application     | Infrastructure.java<br>(119) | Le Duc Thinh |
| 22 | <b>getNotification()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none                                   | Get notification from the server         | Infrastructure.java<br>(115) | Le Duc Thinh |
| 23 | <b>setIncomingMessage(String incomingMessage)</b><br>Input: incomingMessage.<br>Output: none.<br>Pseudocode: none | Set incoming message for the application | Infrastructure.java<br>(127) | Le Duc Thinh |
| 24 | <b>getIncomingMessage()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none                                | Get the incoming message from the server | Infrastructure.java<br>(123) | Le Duc Thinh |
| 25 | <b>setOutgoingMessage(String outgoingMessage)</b><br>Input: outgoingMessage.<br>Output: none.<br>Pseudocode: none | Set outgoing message for the application | Infrastructure.java<br>(135) | Le Duc Thinh |
| 26 | <b>getOutgoingMessage()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none                                | Get the outgoing message from the server | Infrastructure.java<br>(131) | Le Duc Thinh |

List of methods of Drawing class

*Table 18 – List of methods of Drawing class*

| No. | Method  | Purpose   | File name, Line       | Responsible       |
|-----|---|---|-----------------------|-------------------|
| 1   | <b>Render()</b><br>Input: none.<br>Output: none.<br>Pseudo ode:<br>for(object in graphicsElements)<br>draw the object to the canvas                             | Draw all graphic elements which are in the graphics elements list | Drawing.java<br>(41)  | Nguyen Hoang Danh |
| 2   | <b>Undo()</b><br>Input: none<br>Output: String<br>Pseudocode:<br>if the graphics elements list is not empty<br>remove the last element                          | Remove the last element in graphicsElements list                  | Drawing.java<br>(97)  | Nguyen Hoang Danh |
| 3   | <b>Clear()</b><br>Input: none.<br>Output: none.<br>Pseudocode:<br>Clear graphicsElements list<br>Render()   | Clear canvas  | Drawing.java<br>(107) | Nguyen Hoang Danh |
| 4   | <b>getSelectionGraphicElements()</b><br>Input: none.<br>Output: List<DrawingObject><br>Pseudocode: none.  | Get all graphics elements that are selected                       | Drawing.java<br>(115) | Nguyen Hoang Danh |
| 5   | <b>getGraphicElements()</b><br>Input: none.<br>Output: List<DrawingObject><br>Pseudocode: none.   | Get all graphics elements   | Drawing.java<br>(123) | Nguyen Hoang Danh |
| 6   | <b>setGraphicElements()</b><br>Input: List<DrawingObject>.<br>Output: none.<br>Pseudocode: none.  | Set graphics elements   | Drawing.java<br>(127) | Nguyen Hoang Danh |
| 7   | <b>initDrawRect(double x, double y, boolean fill, String color, int thickness)</b><br>Input: x, y, fill, color, thickness<br>Output: none.<br>Pseudocode: none. | Add a new rectangle to object list when the mouse is pressed      | Drawing.java<br>(172) | Nguyen Hoang Danh |
| 8   | <b>onDrawRect(double x, double y)</b><br>Input: x,y.<br>Output: none  | Draw the latest rectangle in the object list                      | Drawing.java<br>(182) | Nguyen Hoang Danh |

|    |  |  |                       |                   |
|----|--|--|-----------------------|-------------------|
|    | Pseudocode: none   | corresponding to the position of the mouse when it is being dragged  |                       |                   |
| 9  | <b>initDrawEllipse(double x, double y, boolean fill, String color, int thickness)</b><br>Input: x, y, fill, color, thickness<br>Output: none.<br>Pseudocode: none. | Add a new ellipse to object list when the mouse is pressed   | Drawing.java<br>(222) | Nguyen Hoang Danh |
| 10 | <b>onDrawEllipse(double x, double y)</b><br>Input: x,y.<br>Output: none<br>Pseudocode: none  | Draw the latest ellipse in the object list corresponding to the position of mouse when it is being dragged | Drawing.java<br>(232) | Nguyen Hoang Danh |
| 11 | <b>initDrawStroke(double x, double y, String color, int penSize)</b><br>Input: x, y, fill, color, penSize<br>Output: none.<br>Pseudocode: none.                    | Add a new stroke to object list when the mouse is pressed  | Drawing.java<br>(272) | Nguyen Hoang Danh |
| 12 | <b>onDrawStroke(double x, double y)</b><br>Input: x,y.<br>Output: none<br>Pseudocode: none   | Draw the latest stroke in the object list corresponding to the position of mouse when it is being dragged  | Drawing.java<br>(285) | Nguyen Hoang Danh |
| 13 | <b>initDrawEraser(double x, double y, double eraserSize)</b><br>Input: x, y, eraserSize<br>Output: none.<br>Pseudocode: none.                                      | Add a new eraser to object list  | Drawing.java<br>(296) | Nguyen Hoang Danh |
| 14 | <b>onDrawEraser(double x, double y)</b><br>Input: x,y.<br>Output: none<br>Pseudocode: none   | Clear every stroke corresponding to the position of mouse when it is being dragged                         | Drawing.java<br>(309) | Nguyen Hoang Danh |

List of methods of ReadThread class

*Table 19 – List of methods of ReadThread class*

| No. | Method   | Purpose                             | File name, Line             | Responsible     |
|-----|--|-------------------------------------|-----------------------------|-----------------|
| 1   | <b>run()</b><br>Input: none.<br>Output: none.<br>Pseudocode: none.         | Listen to data sent from the server | ReadThrea<br>d.java<br>(26) | Le Duc<br>Thinh |
| 2   | <b>getSocket()</b><br>Input: none.<br>Output: Socket.<br>Pseudocode: none  | Get current socket                  | ReadThrea<br>d.java<br>(82) | Le Duc<br>Thinh |
| 3   | <b>setSocket()</b><br>Input: none.<br>Output: Socket.<br>Pseudo code: none | Update socket                       | ReadThrea<br>d.java<br>(86) | Le Duc<br>Thinh |

List of methods of WriteThread class

*Table 20 – List of methods of WriteThread class*

| No. | Method  | Purpose                             | File name, Line              | Responsible     |
|-----|---|-------------------------------------|------------------------------|-----------------|
| 1   | <b>run()</b><br>Input: none.<br>Output: none.<br>Pseudocode: none.        | Listen to data sent from the client | WriteThrea<br>d.java<br>(26) | Le Duc<br>Thinh |
| 2   | <b>getSocket()</b><br>Input: none.<br>Output: Socket.<br>Pseudocode: none | Get current socket                  | WriteThrea<br>d.java<br>(82) | Le Duc<br>Thinh |
| 3   | <b>setSocket()</b><br>Input: none.<br>Output: Socket.<br>Pseudocode: none | Update socket                       | WriteThrea<br>d.java<br>(86) | Le Duc<br>Thinh |

## 2.2. Server

List of classes are used in the server application

*Table 21 – List of classes are used in the server application*

| No. | Class Name | Responsible     | Purpose   |
|-----|------------|-----------------|---|
| 1   | Artboard   | Le Duc<br>Thinh | Store information of an artboard such as code, drawingObjects and clients who are connecting to that artboard |

|   |            |              |   |
|---|------------|--------------|---|
| 2 | Client     | Le Duc Thinh | Store information of a client who is connecting to the server                       |
| 3 | Datasource | Le Duc Thinh | An instance provides the server with the ability to share resources between threads |
| 4 | UserThread | Le Duc Thinh | Handle every request come from clients  |

List of methods of Artboard class

*Table 22 – List of methods of Artboard class*

| No. | Method  | Purpose                                   | File name, Line       | Responsible  |
|-----|---|---|-----------------------|--------------|
| 1   | <b>getCode()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none.          | Get the code of an artboard               | Artboard.java<br>(16) | Le Duc Thinh |
| 2   | <b>setCode()</b><br>Input: String.<br>Output: none.<br>Pseudocode: none.          | Set the code of an artboard               | Artboard.java<br>(20) | Le Duc Thinh |
| 3   | <b>getDrawingObjects()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none | Get graphics elements of an artboard      | Artboard.java<br>(24) | Le Duc Thinh |
| 4   | <b>setDrawingObjects()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none | Set graphics elements of an artboard      | Artboard.java<br>(28) | Le Duc Thinh |
| 5   | <b>getClients ()</b><br>Input: none.<br>Output: List<Client>.<br>Pseudocode: none | Get all clients connected to the artboard | Artboard.java<br>(32) | Le Duc Thinh |

List of methods of Client class

*Table 23 – List of methods of Client class*

| No. | Method   | Purpose                  | File name, Line     | Responsible  |
|-----|--|--------------------------|---------------------|--------------|
| 1   | <b>getClientName()</b><br>Input: none.<br>Output: String.<br>Pseudocode: none. | Get the name of a client | Client.java<br>(7)  | Le Duc Thinh |
| 2   | <b>setClientName()</b><br>Input: String.                                       | Set the name of a client | Client.java<br>(11) | Le Duc Thinh |



|   |   |                                |                     |                 |
|---|---|--------------------------------|---------------------|-----------------|
|   | Output: none.<br>Pseudocode: none.  |                                |                     |                 |
| 3 | <b>getClientThread()</b><br>Input: none.<br>Output: UserThread.<br>Pseudocode: none | Get the UserThread of a client | Client.java<br>(15) | Le Duc<br>Thinh |
| 4 | <b>setClientThread()</b><br>Input: UserThread.<br>Output: none.<br>Pseudocode: none | Get the UserThread of a client | Client.java<br>(19) | Le Duc<br>Thinh |

List of methods of Datasource class

*Table 24 – List of methods of Datasource class*

| No. | Method   | Purpose   | File name,<br>Line          | Responsible     |
|-----|--|---|-----------------------------|-----------------|
| 1   | <b>getArtboards()</b><br>Input: none.<br>Output: List<Artboard>.<br>Pseudocode: none.      | Get a list of artboards that are stored in the server | Datasource<br>.java<br>(23) | Le Duc<br>Thinh |
| 2   | <b>getConnectedClients()</b><br>Input: none.<br>Output: List<Client>.<br>Pseudocode: none. | Get a list of clients that are stored in the server   | Datasource<br>.java<br>(31) | Le Duc<br>Thinh |

List of methods of UserThread class

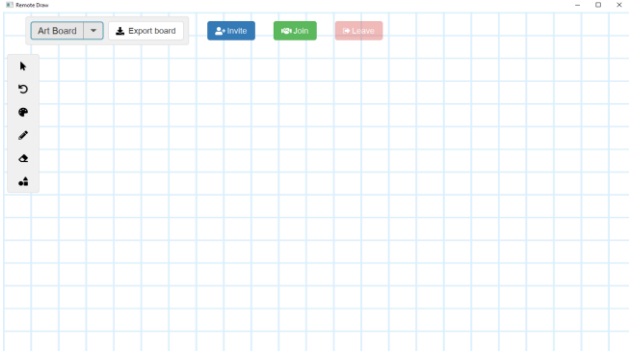
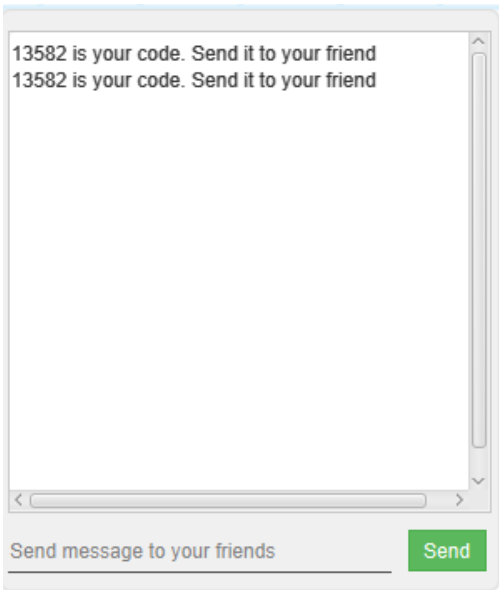
*Table 25 – List of methods of UserThread class*

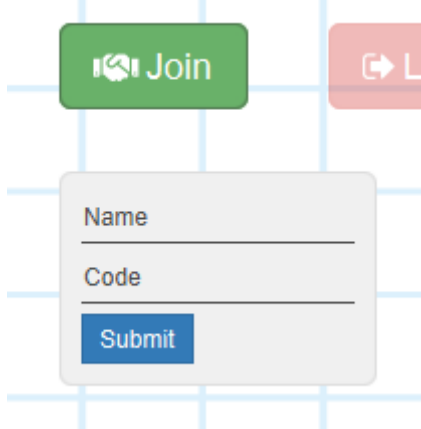
| No. | Method   | Purpose   | File name,<br>Line           | Responsible     |
|-----|--|---|------------------------------|-----------------|
| 1   | <b>run()</b><br>Input: none.<br>Output: none<br>Pseudocode: none.                              | Listen to any request from clients  | UserThrea<br>d.java<br>(20)  | Le Duc<br>Thinh |
| 2   | <b>inviteAction(String[] message)</b><br>Input: message.<br>Output: none.<br>Pseudocode: none. | Create an artboard with a unique code<br>Add graphics elements to that artboard<br>Send the code to the client      | UserThrea<br>d.java<br>(111) | Le Duc<br>Thinh |
| 3   | <b>joinAction(String[] message)</b><br>Input: message.<br>Output: none.<br>Pseudocode: none.   | Find the artboard with the code that equals to the code from the client<br>Send all graphics elements to the client | UserThrea<br>d.java<br>(139) | Le Duc<br>Thinh |

|   |  |  |                          |              |
|---|--|--|--------------------------|--------------|
| 4 | <b>leaveAction(Artboard artboard, Client client)</b><br>Input: artboard, client.<br>Output: none.<br>Pseudocode: none.                         | Close the connection between client and server             | UserThread.java<br>(184) | Le Duc Thinh |
| 5 | <b>updateAction(Artboard artboard, Client client)</b><br>Input: artboard, client.<br>Output: none.<br>Pseudocode: none.                        | Tell the client that it needs to update graphics elements. | UserThread.java<br>(204) | Le Duc Thinh |
| 6 | <b>broadcastMessage(Artboard artboard, Client client, String data)</b><br>Input: artboard, client, data.<br>Output: none.<br>Pseudocode: none. | Broadcast message to other clients                         | UserThread.java<br>(211) | Le Duc Thinh |

### 3. Graphic User Interface

Table 26 – GUI explanation

| No. | GUI   | Purpose  | Brief Explanation   |
|-----|---|--|---|
| 1   | <b>Main window</b><br> | The main window of the application                                   | <b>Le Duc Thinh</b><br>On the left side is the menu for drawing functions such as Select, Undo, etc.<br>On the top is the menu for export the artboard to png file, three buttons which help connect to other users.<br>After choosing one of the drawing function like Pen, the user can click and hold on the blue grid to start drawing. |
| 2   | <b>Chat box</b><br>  | Allows user to communicate with others                               | <b>Le Duc Thinh</b><br>User can type anything to the text field and then click send button. A message will be displayed in the text area above the text field and send button   |
| 3   | <b>Join form</b>  | Allows user to send the code to the server to join with other people | <b>Le Duc Thinh</b><br>After filling all field user can click submit button. The application will send the code to the server and tell  |

|  |   |  |   |
|--|---|--|---|
|  |  |  | the server to send information of graphics elements of an artboard containing that code back to the application |
|--|---|--|---|

## IV. Test cases

*Table 27 – Test cases*

| No. | Test cases  | Purpose  | Brief Explanation   |
|-----|---|--|---|
| 1   | <b>Test case 1:</b><br>Input:<br>Set the IP address for the socket is “127.0.0.1” and the port is the port of the server.<br>Result:<br>Two apps connect successfully.<br>Every graphics element is updated successfully                    | Test the communication between client and server                       | Open two application in the same computer and start drawing.  |
| 2   | <b>Test case 2:</b><br>Input:<br>Set the IP address for the socket is IP address of another computer and the port is the port of the server.<br>Result:<br>Two apps connect successfully.<br>Every graphics element is updated successfully | Test the communication between client and server in different computer | Place the server on another computer.<br>Open two application in a different machine and start drawing. |

## V.Conclusion

### 1. Student evaluation

- Almost requirements are met.
- Design the application with Object-Oriented programming paradigm.

- Simple design GUI for easy using.
- The code is quite clean and reusable
- Our application allows many people to draw together.
- Our application cannot allow the user to type in the canvas and import an image to the canvas.

## *2. Difficulties*

- Learning new technology is a problem for us because it slows down the project progress
- Multi-threading programming is also a problem because we do not have enough knowledge and practice.

## *3. Advantages*

- Quite clean code
- Meets the requirements of the project
- Simple GUI, users easy to use this application to draw and invite other people to join with them.
- Reuse, recycling, and maintainability

## *4. Disadvantages*

- Multiple threads in this application are working but not in the perfect way. Therefore, user experience (UX) may be a problem.
- The application cannot allow users to type text on the canvas.
- The application cannot import images to canvas.

## *5. Development ideas*

- Instead of allowing a user with the basic drawing functions, we can upgrade the drawing functions to become more technical and professional like Photoshop or Corel Draw.

## References

Examples about chat application using java socket:

<https://cs.lmu.edu/~ray/notes/javanetexamples/>

<https://www.geeksforgeeks.org/multi-threaded-chat-application-set-1/>

How to install JavaFX 13 in IntelliJ IDEA:

<https://openjfx.io/openjfx-docs/#install-javafx>

How to use canvas in javaFX:

<https://examples.javacodegeeks.com/desktop-java/javafx/javafx-canvas-example/>

Multi-threading programming:

<https://dzone.com/articles/java-thread-tutorial-creating-threads-and-multithr>

Socket in java:

<https://www.geeksforgeeks.org/socket-programming-in-java/>