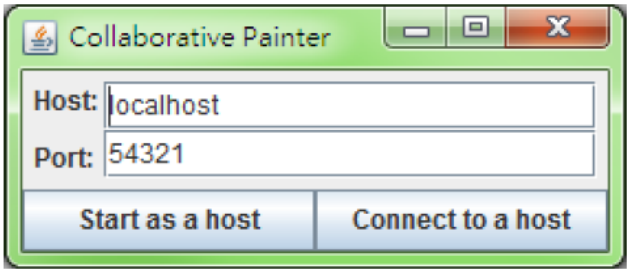
CSIS0396A / COMP2396A Assignment 6
Due: 30th Nov, 2015 23:59

In this assignment, you are required work individually or in a team of two students to implement a collaborative drawing program (`CoPainter.java`) in Java. This assignment can be separated into two parts. You are supposed to finish part 1 before Nov 20 (our next tutorial), and complete the rest in the week after.

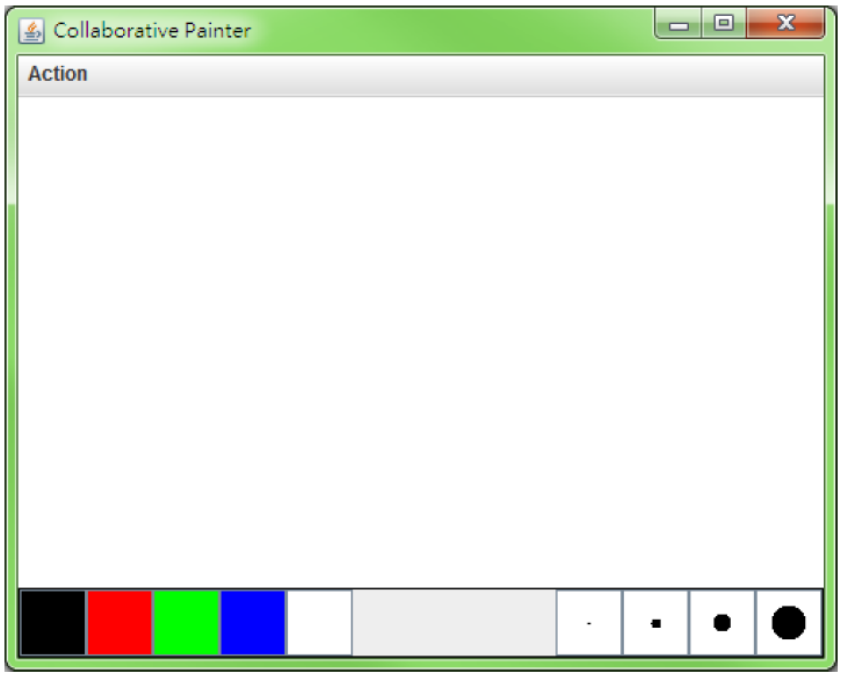
Part 1

GUI

When the program is executed (class **CoPainter**), the startup prompt appears:



In part 1, the above interface is not used. Upon clicking either of the buttons, the GUI will be replaced by the painter interface:



For the painter, it must consist of the following:

- An ***Action*** menu, with menu item **Clear**, **Save**, **Load** and **Exit**
- A color picker at the bottom left area, with at least **black**, **white** and one other color. You can freely design your color picker.

- A pen size picker at the bottom right area, with at least **3 different pen sizes**.

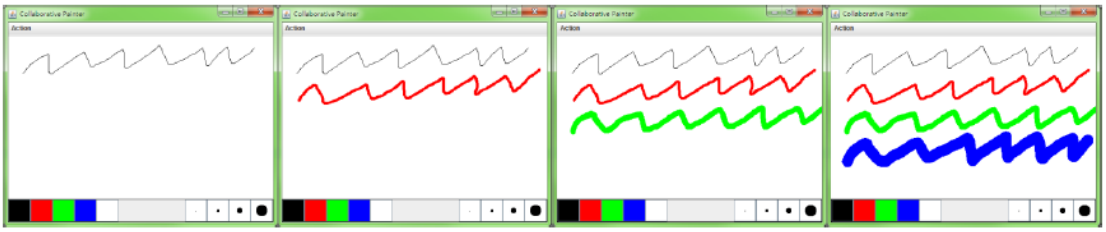
Note that buttons in color picker and pen size picker may contain text descriptions only. Bonus marks will be given if the buttons can contain graphical descriptions as in the diagram above.

These images show the essential elements that should appear in your GUIs, your implementation must look similar to them (except for the color and pen size picker). You should follow the placement of the elements in the GUIs, while the details like the borders and spacing may different.

Both GUIs must terminate the program completely if they are closed.

Drawing

Users draw on the canvas (the white area) by clicking and dragging on it. The current selected color and pen size will be used. Here is an example of someone drawing 4 lines on the canvas:



Menu items

There are 4 menu items:

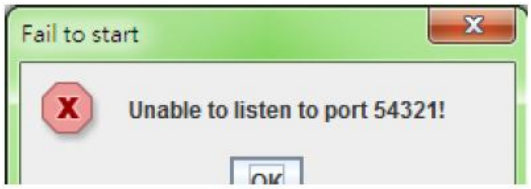
- **Clear**: clear the current drawing
- **Save**: save the current drawing in a file, you should use **JFileChooser** to pick the file
- **Load**: load a previously saved file and resume drawing, you should use **JFileChooser** to pick the file
- **Exit**: terminate the program completely

Part 2

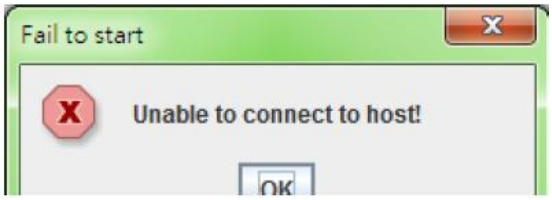
Startup prompt

The startup prompt is used to decide whether the painter interface works as a server (host) or a client.

If "**Start as a host**" is clicked, only the port number in the interface is used. The program will start listening to the specified port for incoming clients. At the same time, the painter interface will be shown (and replace the startup prompt). If the program cannot start a server at the specific port, an error message must be shown in a dialog box. The program will stay at the startup prompt after this.



If "**Connect to a host**" is clicked, the program will connect to the specified host and port. At the same time, the painter interface will be shown (and replace the startup prompt). If the program fails to connect to a server, an error message must be shown in a dialog box. The program will stay at the startup prompt after this.

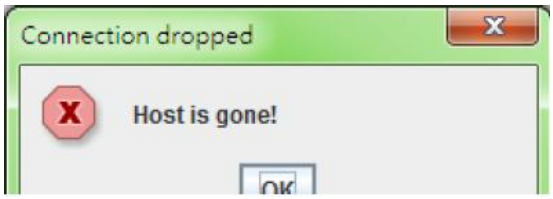


Painter behaviors

Here are the behaviors of the painter program:

- For the server painter, when a user finished drawing a line (e.g., after the user released the mouse button), the line will be sent to all clients. The clients will then display the line on canvas.
- For the client painter, when a user finished drawing a line, the line will be sent to the server. The server will then display the line on canvas. At the same time, the server will also forward the line to all clients to display.
- If the server painter used the Clear menu item, all the clients must also clear their drawing.
- If the server painter used the Load menu item, all the clients must also load the same drawing.
- When a client painter starts, it must show all the lines that is previously drawn on the server.
- Only the server painter can use the Clear and Load menu items. These menu items should either be disabled or hidden in the client painters.

- When a client painter terminates, the server and all other clients should not be affected.
- When a server painter terminates, all clients will also be terminated. An error message must be shown in the clients before so.



Exception Handling

You must handle as much exceptions as you can. You will lose marks if there is any exception uncaught.

Marking Scheme

Painter functionality: 35%
File access functionality: 30%
Server and client functionality: 35%
Bonus (buttons in color picker and pen size picker contain graphical descriptions): 5%

Submission: (You should submit a single compressed file containing all the above files through Moodle)
CoPainter.java (This is the main program)
All other files that you have created
Note: You do not need to generate and submit the JavaDoc
If you work in a team of 2 students, only ONE submission is required but please include a readme.txt file to list the names and student numbers of all members in the team.