COM90015 Distribute System

Assignment-2

Share Whiteboard

Name: Yicong Li

Student ID: 1307323

**Introduction**

This project is to design a shared whiteboard that multiple users draw at the same time. There are two different roles in the system: manager and user. The manager is allowed to draw several different shapes including rectangle, circle, triangle, line and pen, select different colours and input a text and also include several advanced features such as new, save and open. The user is only allowed draw shapes, choose different colours and input a text. All these functionalities will discuss later.

**Diagram**

**Diagram, schematic

Description automatically generated**

**Class Design**

**Common Package**

**GUIDesign**

This is my GUI class, which include all my swing components such as all required buttons, canvas and etc. The good thing is that if I want to user any component, I could just inheritance this class and call its method from this class. It will reduce amount of duplicate code and make more readable because the user and manager both need this class.

**UIListener**

This is my buttonListener class, which include my button events in this class such as shape buttons, colour button, and add text button. When I click any of these buttons, then it will actionPerformance method, I pass this method to the UIlistener, and then it will analyse my button name and then implement the correct command. Firstly, it will set the cursor status first, then I try to draw the shape based on the button to canvas component. The canvas is also passed to my UIlistener. Then the user could draw anything that he selects. In addition, Once the draw is finished, then encapsulate these data and then send them to the server. The server will do further commands such as broadcast to all other users based on the request type. Then it will call redraw method in the UIListener. The redraw method is to parse all json data to readable data for draw shape. The good thing is that if I want to use any component, I could just inherit this class and call its method from this class. It will reduce the amount of duplicate code and make it more readable because the user and manager both need this class.

**Manager Package**

**ConnectionManager**

This class is my broadcast class, it is to broadcast all connections based on specific commands. the commands include "Draw/" and "Clear/". The canvasrepaint method is to add a new shape' information to the shape list. The newUser method is to justify if the user is allowed to join by the manager. if it is ok, respond to a message on the terminal on the server-side. However, if it's no, then close that connection and eliminate that process. the broadcast method is to let all connections draw shapes. The updatePaintingBoardcast method is to broadcast draw requests and their shape data to all users. The clearPaint method is to broadcast clear requests to all users. The systemOverBoardcast method is to exit requests to all users.

**ManagerApp**

This is one of my main functions in the manager side. The main process will run the server. When the system is running, it will create a new thread for running managerframe instance. In the managerframe instance, it has a socket for setting up TCP connection with server. Therefore, the manager will automatically connect to server if the ip address and port number are valid.

**ManagerFrame**

The constructor is for setting up TCP connection with server. In addition, the createUI method is to design manager layout. All the layout function inherits from GUIDesign class. We could direct call its method without create a new instance. All functionalities will pass to UILIsitener for handling button event.

**Server**

This class is my server class for setting up a TCP connection because I used serversocket to set it up. It’s a TCP connection architecture. The thread architecture that I used is a thread per connection. The server will receive the request from its socket, and then pass it to the serverconnection instance. the serverconnection instance will do further handling.

**ServerConnection**

The constructor is my communication objects between users and the server. in the class i use DataInputStream as my reader and DataOutputStream as my writer. This class also has a runnable interface extended from the Thread class. I rewrite the run method and then receive all the requests from the users. When this class receive a "Request" command, then just it will invoke three static methods from connectionmanager class. firstly, store username and connection as a combination, then justify if the user is allowed to join by the manager. finally, broadcast draw request and its shape data to all users. In addition, when this class receive a "Draw" method, then it will invoke the redraw method from UIListener to parse all the shape data in JSON format into readable data. then broadcast to all the users.

**Tools**

This class is button events for newbutton, savebutton, openbutton and exitbutton. The newCanvas() method is to clear all the stored shape data from the list, Once the list is empty, it will other methods to repaint the empty list to canvas. The saveCanvas() is to save all shape's information in JSON format in the .txt file line by line. The openCanvas is to read the previously-stored .txt file and then parse data into readable data. When the parse is finished, it will repaint all the data into the canvas.

**User Package**

**UserApp**

This is one of my main functions in the user side. When the system is running, it will create a new thread for running userframe instance. In the userframe instance, it has a socket for setting up TCP connection with server.

**UserFrame**

The constructor is for setting up a TCP connection with the server. the createUI method is to design a user layout. All the layout function inherits from GUIDesign class. We could direct call its method without creating a new instance. All functionalities will pass to UILIsitener for handling button events. Once the user has connected with the server, then it will automatically send a request -> (“Request/”) to the server using DataOutStream. In addition, it will create a thread for receiving the response data from the server. If this thread receives a “Draw” string, then it will call the redraw method to parse all the shape data in JSON format to readable data and then return on canvas. If this thread receives a “Clear” string, then it will call canvas.repaint() method because the shape data list has already been empty. If this thread receives an “Exit” string, then this process will be exited.

**Communication Protocol**

The communication protocol that I used in my project is TCP connection because TCP connection is easier to implement and fulfil in this project. The advantage feature of implementing a TCP connection will not lose information during the transmission. Using UDP may lose information during the transmission. The RMI method is a good option but a little bit hard to understand and implement.

**Message Format**

The message format that I used for transmission in my project is a combination method. I used string and JSON format together. For example, Draw/{"Shape":"Line","StartX":153,"FontSize":2,"StartY":172,"Color":"0 0 0","EndX":392,"EndY":172} . the “Draw” is the request type that I send to the server. After the slash symbol is the content of this request type. In this case, this is a draw request, Therefore, the rest of the JSON format is all draw method detail.

**Advanced Features**

I implemented six different advanced features in my application. The first one is the newButton. It’s aiming to remove all data and return an empty whiteboard to all the users. The second one is the storeButton, which means saving painting data into a .txt file. The third method that I implemented is the openButton. It’s to open a .txt file that I just save and read all data from this .txt file and then repaint to all users. Fourthly, I implemented exitButton to close all the processes at the same time. Fifthly, I implemented the functionality of removing the existing users. When the manager clicks twice by mouse on one user, then it will close that user process by socket transmission. Click the refresh button updates the display userlist. The last one is to design a pen that allows users to draw any shape.