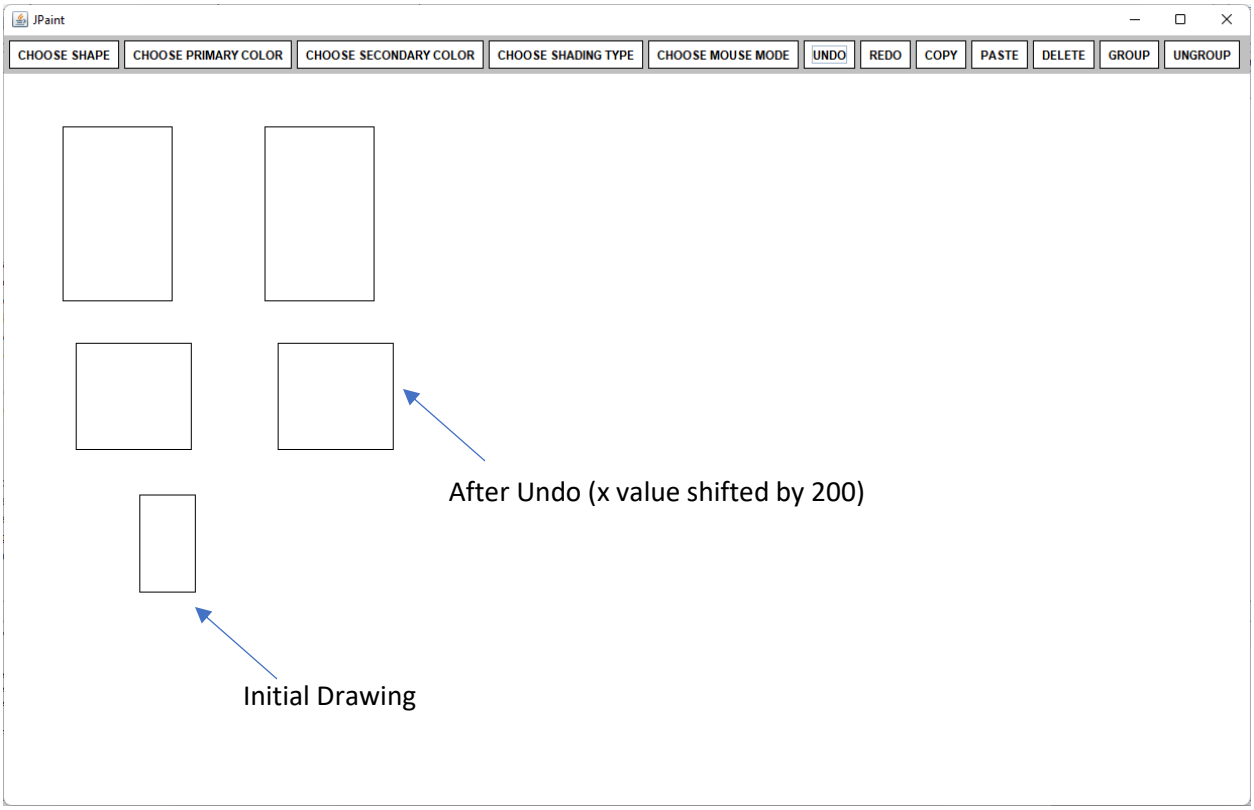
README

Project 1st Check-in

1. **features, bugs, extra credit, and miscellaneous notes.**

I couldn't implement the Undo/Redo correctly due to an issue with repaint() method. It clears out the graphics and when I redraw by retrieving the saved objects inside the stack it doesn't show the new draw. So, I commented out where I called the repaint method. Therefore, old rectangles are still remain on the canvas. To show that undo works I shifted starting "x" point by 200 pixels when I hit undo.



2. **Link to GitHub repo -** https://github.com/sanjayadpf/OOPFinalProject.git

Note: I have given the access. Please accept the invitation.

3. **List of Design Patterns**

Tried to implement Command pattern for Undo/Redo using the CommandHistory.java. Couldn't succeed because of the issue that I faced with repaint() method. Trying to figure out How to use the CommandHistory.java file with the actionlistener of the Buttons. Created a GenerateShape Class which implements ICommand and IUndoable. In there I'm adding the GenerateShape objects to the CommandHistory.

Project 2nd Check-in

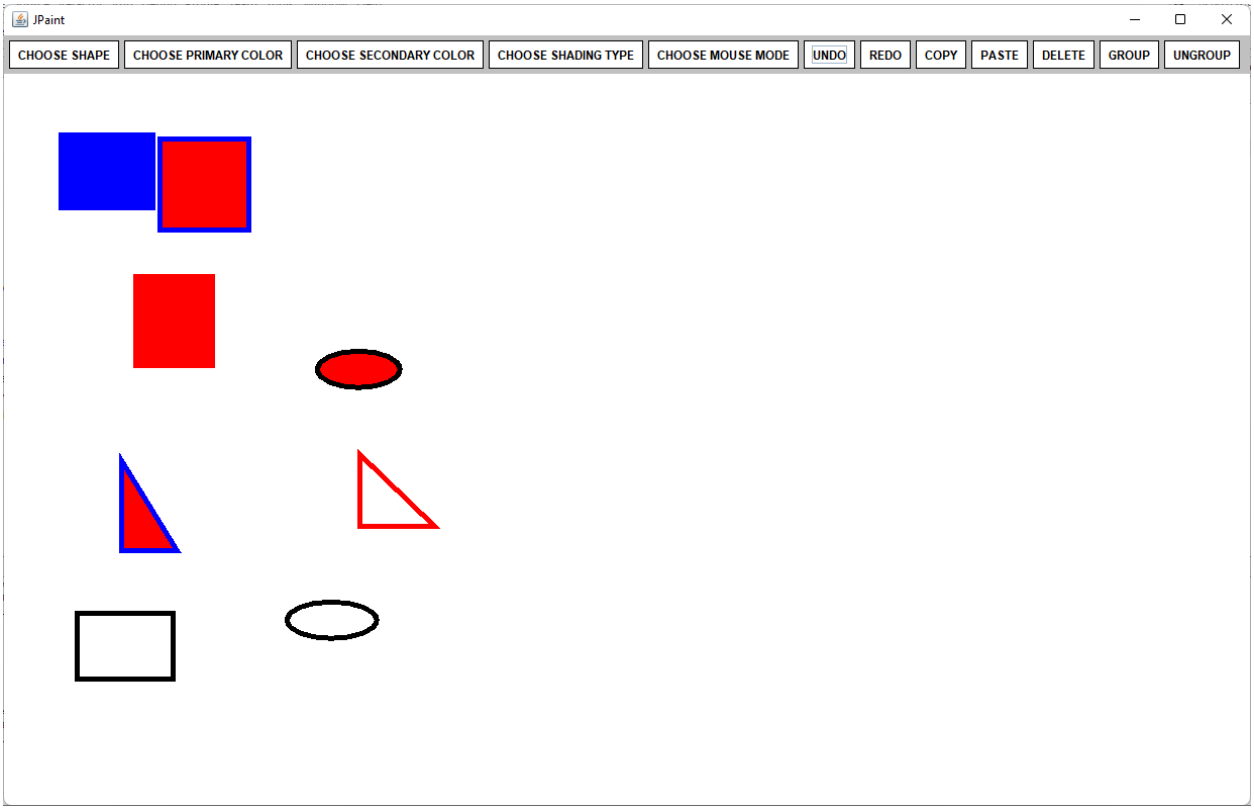
1. **features, bugs, extra credit, and miscellaneous notes.**

I have implemented the button functions CHOOSE SHAPE, CHOOSE PRIMARY COLOR, CHOOSE SECONDARY COLOR, CHOOSE SHADING TYPE. Couldn't implement the functions of mouse mode. So, the entire app is always on Draw mode. I struggled with selecting the shape. So, I uploaded the version which has no selection or move features. Major improvement is I managed to implement the Undo/Redo which I couldn't implement due to the issue I had with repaint() method. Anyway I tried fixing the issue with repaint() method as you suggested in the feedback last time. But I had no luck going forward with it. Instead, I used the setColor() method of Paint Canvas which I can set the Background color whatever the color I want. So I set it to "white" whenever I want to clear(easiest thing I should've done last check-in). I override the toString method of ColorInfo class. This was helpful for me when I was adding the ColorInfo objects to the JOptionPane with input dialogbox. In order to display the Color, toString() method has to override with the String we want to display. For this I referred following links.

https://stackoverflow.com/questions/27518555/pop-up-swing-window-with-both-option-and-textfield

https://www.codejava.net/java-se/swing/jcombobox-basic-tutorial-and-examples

Sample View of the App



2. **Link to GitHub repo -** https://github.com/sanjayadpf/OOPFinalProject.git

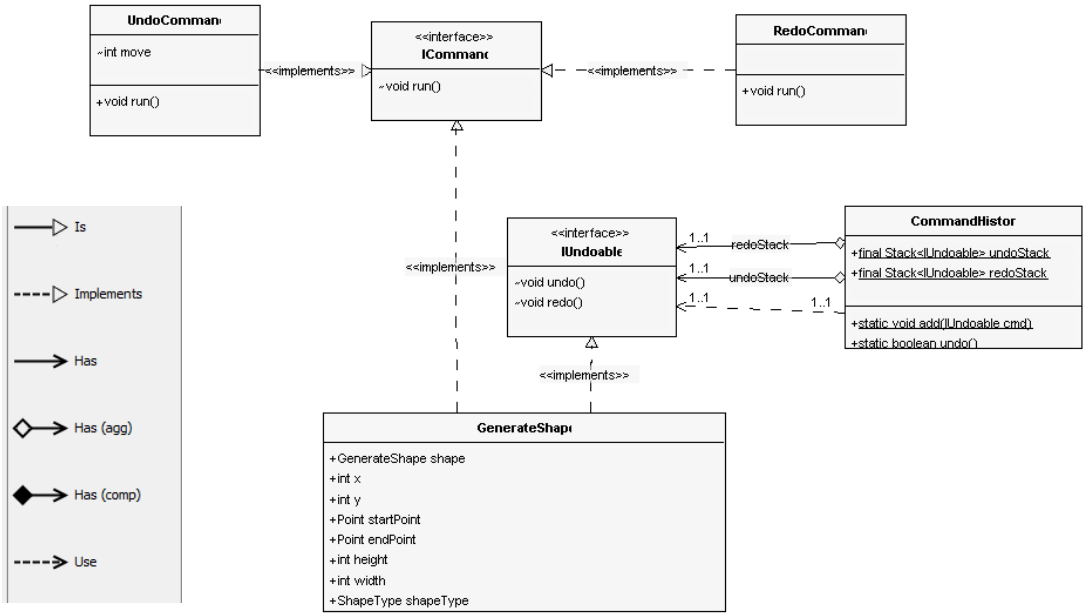
Note: Please contact me if you have any issues with accessing the repo. SMALLIKA@depaul.edu

3. List of Design Patterns

I implemented 3 design patterns discussed in the class up to now.

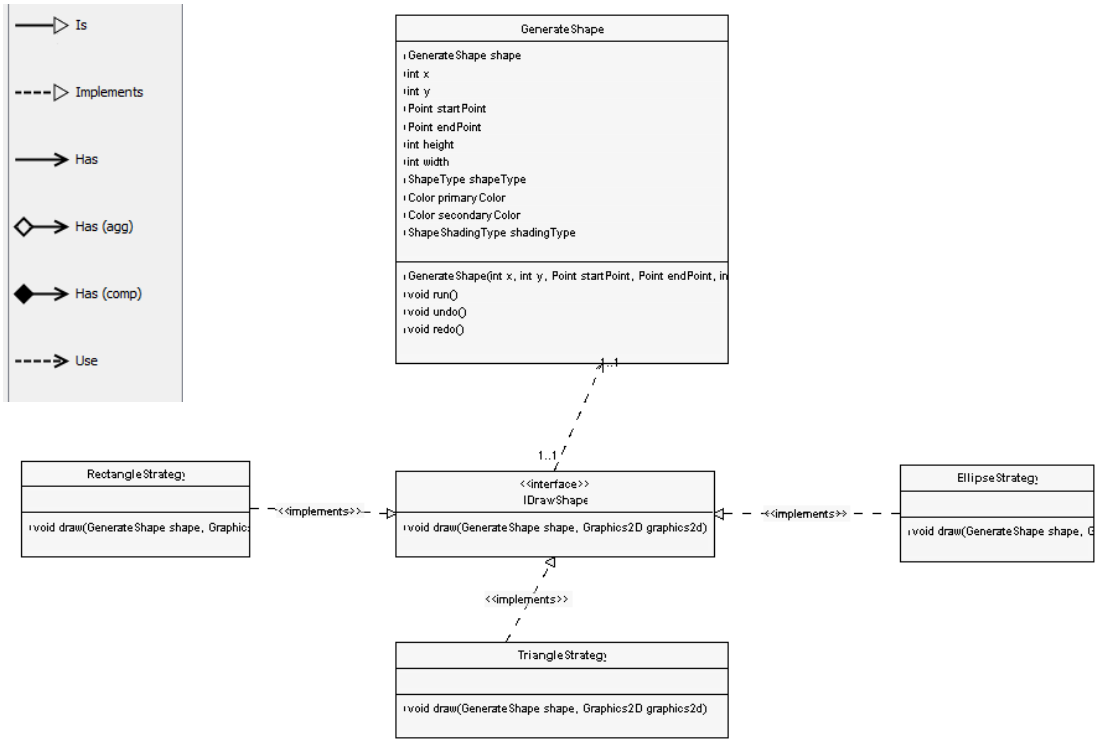
- Command Pattern

This is implemented along with the CommandHistory.java for the UNDO/REDO



- Strategy Pattern

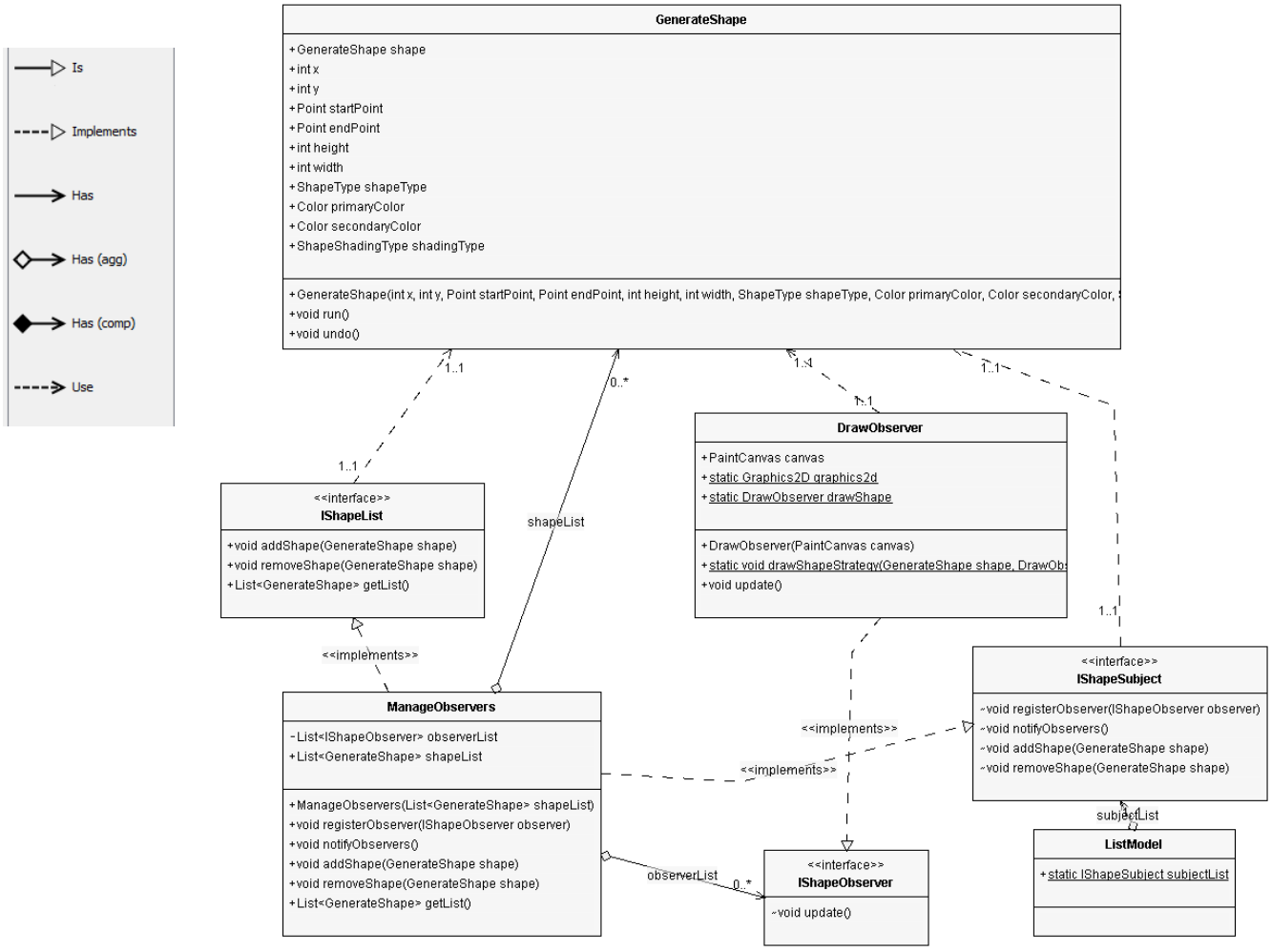
To draw the shape with same draw() method while the object assigned is changing in run time.



Note: Not all relationships are shown. Only the required ones.

- Observer Pattern

To notify the which shape currently is drawn.



Note: Observer is registered in the main method.

Future Development.

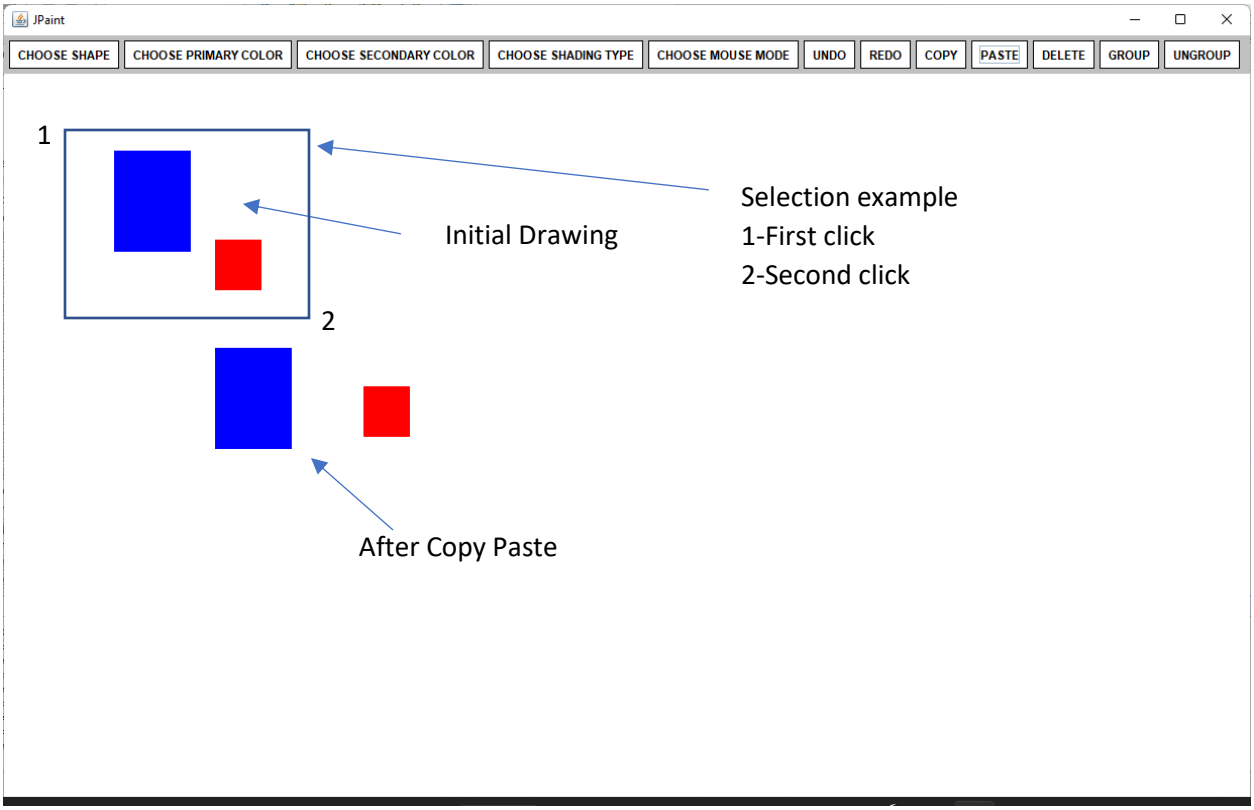
- Another Subject/Observer to implement the Mode selection or Static Factory Design.

Project 3rd Check-in

1. **features, bugs, extra credit, and miscellaneous notes.**

Implemented Select, Move (which wasn't working last time) new features Copy, Paste and Delete functioning as expected. Improved the single responsibility of DrawObserver Class based on the last feedback. Removed the Strategy selection from DrawObserver. App has the default mode Draw. When selecting first select the "Select" option from the Mouse Mode options and then select the shape. First click an upper Left Point. Then Click the Right Lower Point to make a proper selection. Following the same way select the shape first and then Click Copy and Paste to execute Copy and Paste Command. Paste will locate new shapes shifting 100,100 pixels from the original shape location. For Delete follow the same selection first and then Click the Delete Button. No Undo, Redo implemented for Select and Copy

Sample View of the App (Copy Paste)



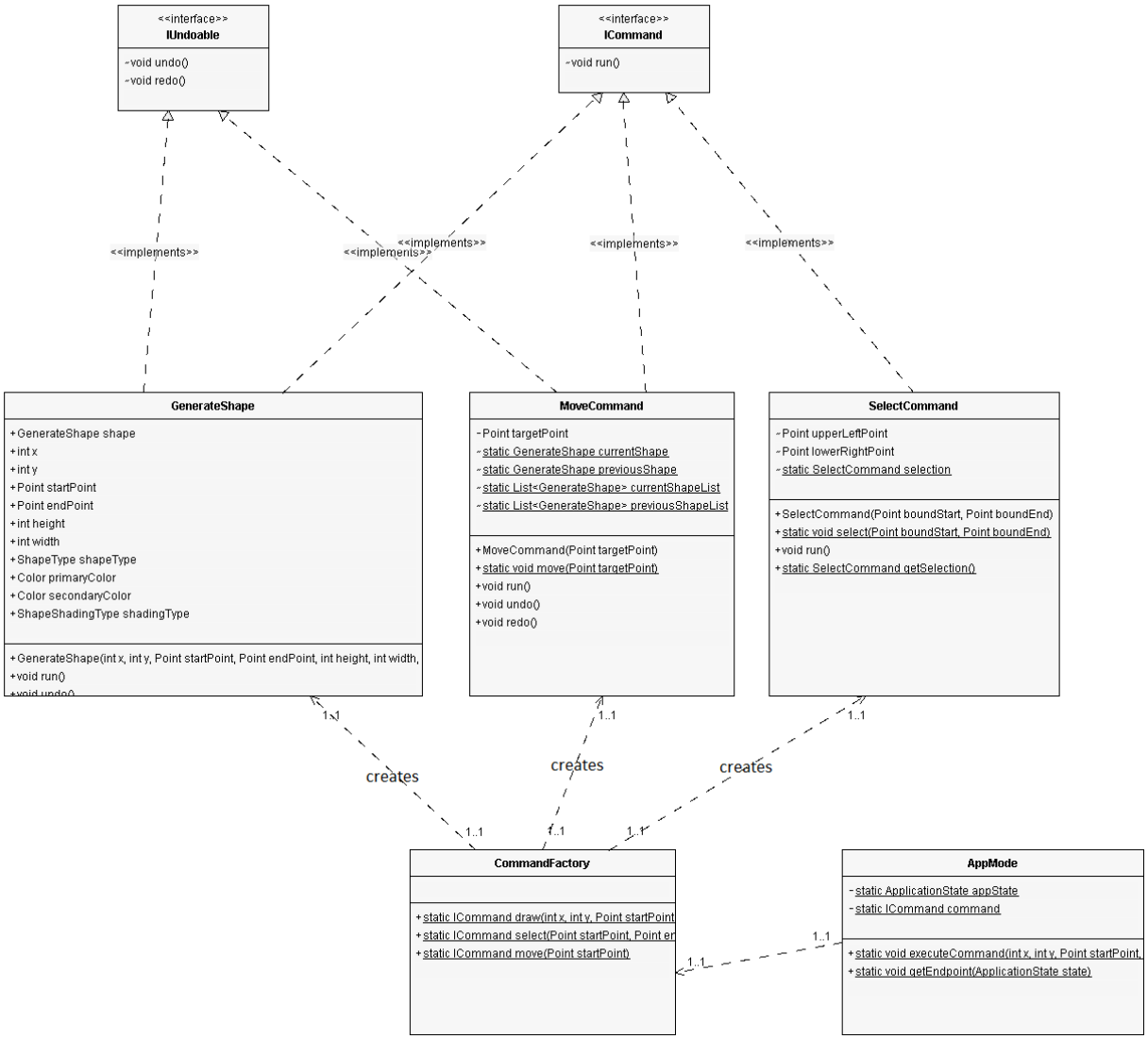
2. **Link to GitHub repo -** https://github.com/sanjayadpf/OOPFinalProject.git

Note: Please contact me if you have any issues with accessing the repo. SMALLIKA@depaul.edu

3. List of Design Patterns

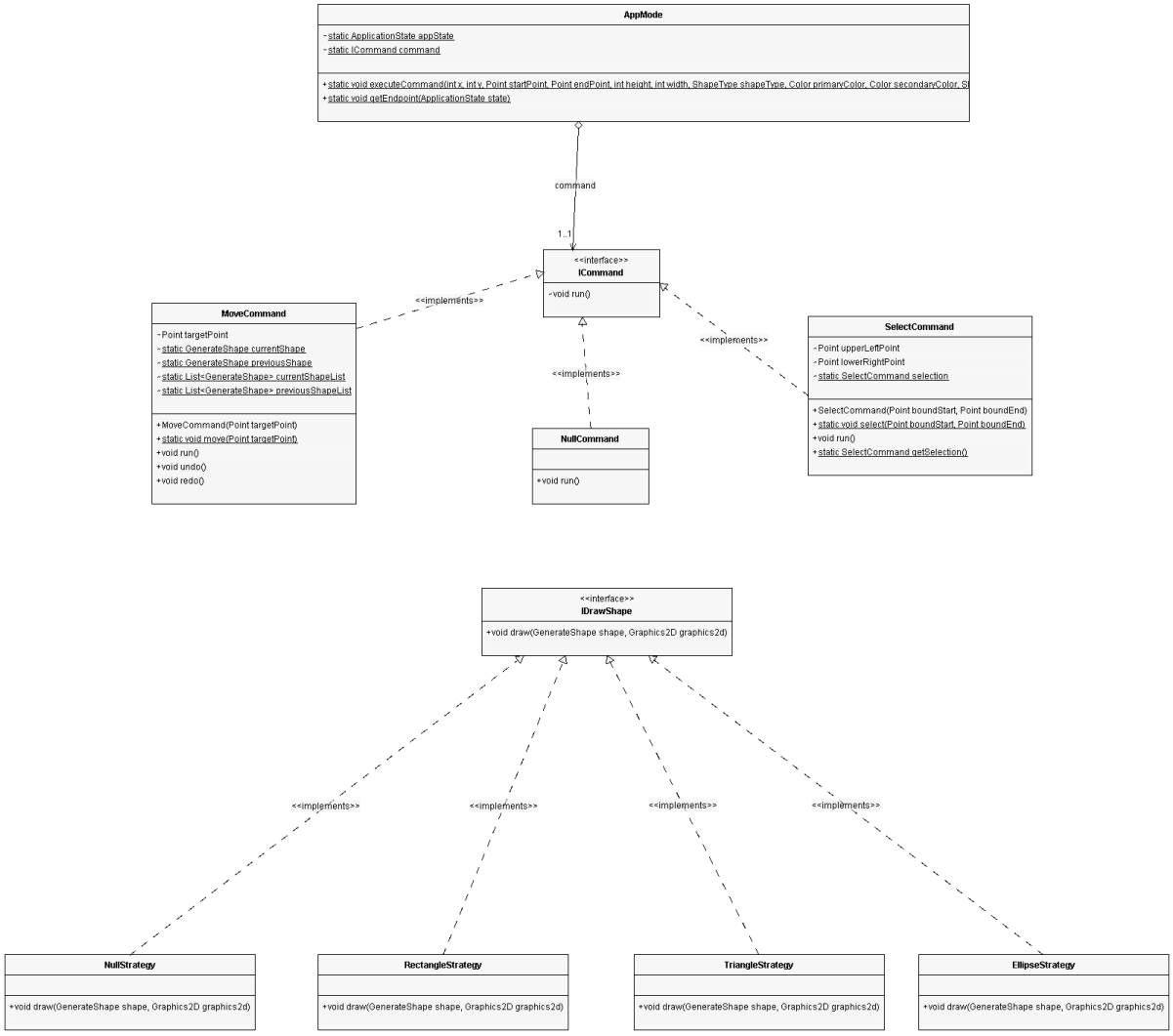
- Included two new Design Patterns with theses two project now has 5 design patterns. New design patterns are indicated in red color.
    (1) Command Pattern (Redo/Undo/Copy/Paste)
    (2) Strategy Pattern (For Selecting shape strategy)
    (3) Observer Pattern (To notify which shape is currently drawing)
    (4) Static Factory (For mode selection Select Move Draw)
    (5) Null Object (Inside CommandFactory in case the Command Object for Draw Select Move is not received and incase right strategy is not received)

(4) Static Factory (For mode selection Select Move Draw)



(5) Null Object (Inside CommandFactory in case the Command Object for Draw Select Move is not received and incase right strategy is not received)

- Used in AppMode Class (draw select move selection) – if command Object is null. NullCommand object will be created.
- Used in Strategy selection inside the Strategy selection class – if strategy Object is not received Null Strategy object will be created.



Future Development

Planning to implement Temporary selection box (pop-up) so that user can clearly see the selected region. Also Undo Redo for Select and Copy.

Project 4th Check-in

1. **features, bugs, extra credit, and miscellaneous notes.**

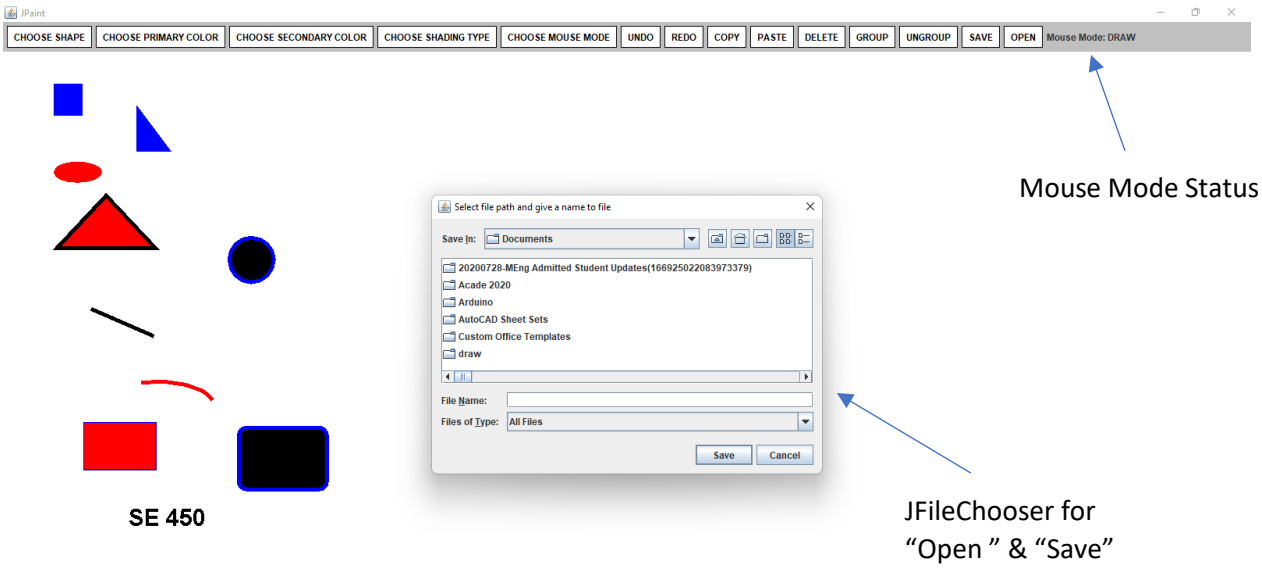
Features

I have implemented 10 shapes (including rendering text) as new shapes. Most challenging shapes were Arc, Round rectangle and text rendering. These shapes need more than two inputs (more than two mouse click points). As an example, the Arc needs start point angle and the end point angle. My solution was popping up an JOptionPane dialog box to get these additional values as keyboard inputs. I keep a flag for these objects since it is not required to re-enter theses values when moving and copying. Also, the user can save the entire drawing by simply clicking the "Save" button. JFileChooser window will pop-up and user have to select a file location from the File Chooser window and give a name to the current drawing. Then the drawing will be saved as a JSON object in the disk once you hit the "Save" button in the Filechooser. To load the drawing from the disk user has to click the "Open" button then it will again pop-up the JFileChooser window and user has to select the saved JSON object from the disk and click open on JFileChooser window. Then the saved drawing will be rendered on the application. In order to get it work this feature you have to add the provided .jar fille to your dependencies. Please kindly do so otherwise program wouldn't work. Another small feature that I added was showing the current MouseMode of the app on the button panel. I thought this will be useful since user instantly knows what mode that the app is currently in. Also, I have implemented the bounding box which wasn't a part of my last check-ins.

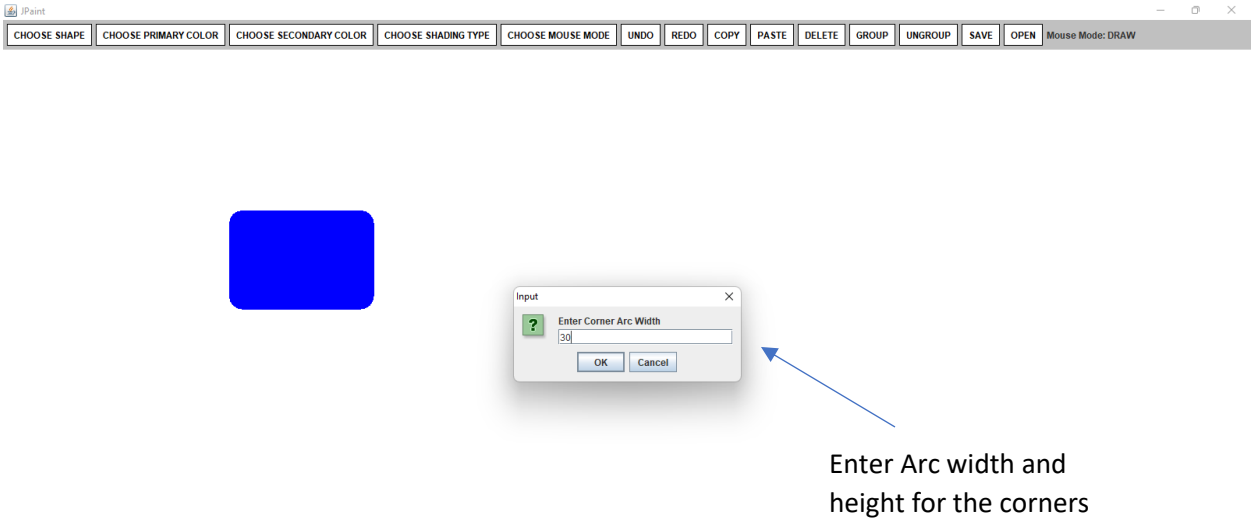
Bugs

For some shapes bounding box may appeared in some weird places but the Move, Copy functions will work. Group is working but there is no large bounding box combining the grouped objects. Once you hit the group you can perform the Copy, Move to the selected objects. First select one by one then hit "Group". The you can perform Copy, Paste and Move to those grouped objects.

Sample View of the App (With some of the new features)



Drawing Round Rectangle Example



2. **Link to GitHub repo -** https://github.com/sanjayadpf/OOPFinalProject.git

Important: Please add .jar file in the lib folder to your project

Note: Please contact me if you have any issues with accessing the repo. SMALLIKA@depaul.edu

3. List of Design Patterns

New Design Pattern included is Singleton. JPaintManager is the class where all the Lists are maintained and the Lists are initiated only one time. So I made the JPaintManager instance singleton(only create one time initializing all the lists. Some Lists are set in main. Others are inside the constructor).

Following is the list of Design Patterns (All 6 now) implemented in the entire project. New contents are indicated in red color.

(1) Command Pattern (Redo/Undo/Copy/Paste/Save/Open/Group/UnGroup)
(2) Strategy Pattern (For Selecting shape strategy, new strategies for new shapes)
(3) Observer Pattern (To notify which shape is currently drawing)
(4) Static Factory (For mode selection Select Move Draw)
(5) Null Object (Inside CommandFactory in case the Command Object for Draw Select Move is not received and incase right strategy is not received)
(6) Singleton Pattern (To create the JPaintManager instance and initialize its attributes needed for the JPaint app.)

(6) Singleton Pattern (To create the JPaintManager instance and initialize its attributes needed for the JPaint app.)