Final Project (JPaint)

Written Report

Prepared by Disha Patel

On Mar 9th, 2018

Table of Contents

I. Extra Credit/ Unimplemented Features

Extra Credit Feature:

- Changes the color of primary and secondary for Shapes in ‘Select’ mode.

Unimplemented Feature:

- Undo
- Redo
- Copy/Paste
- Delete
- In Select mode, cannot drag multiple shapes. Only click event is present.
- Move shape is not implemented.

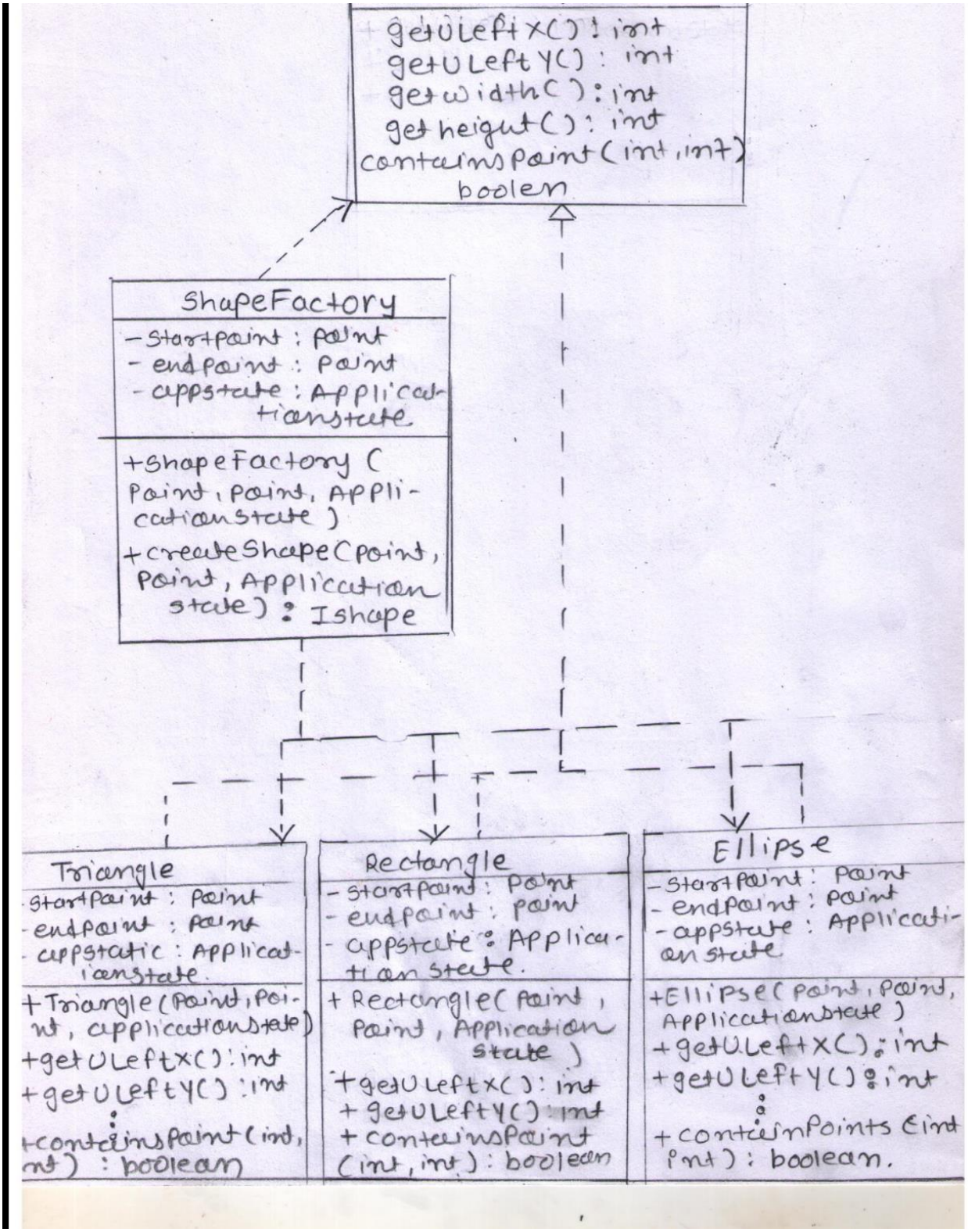
Bug

- Ideally for changing primary and secondary color, the shading type of shape should be ‘Outline and filled in’ but I can change the color for shading type ‘filled in’ and ‘outline’ when only last added shape have shading type ‘Outline and filled in’ and rest doesn’t have. So, it’s not checking for list of shapes the shading type. Only last shading types count in this.

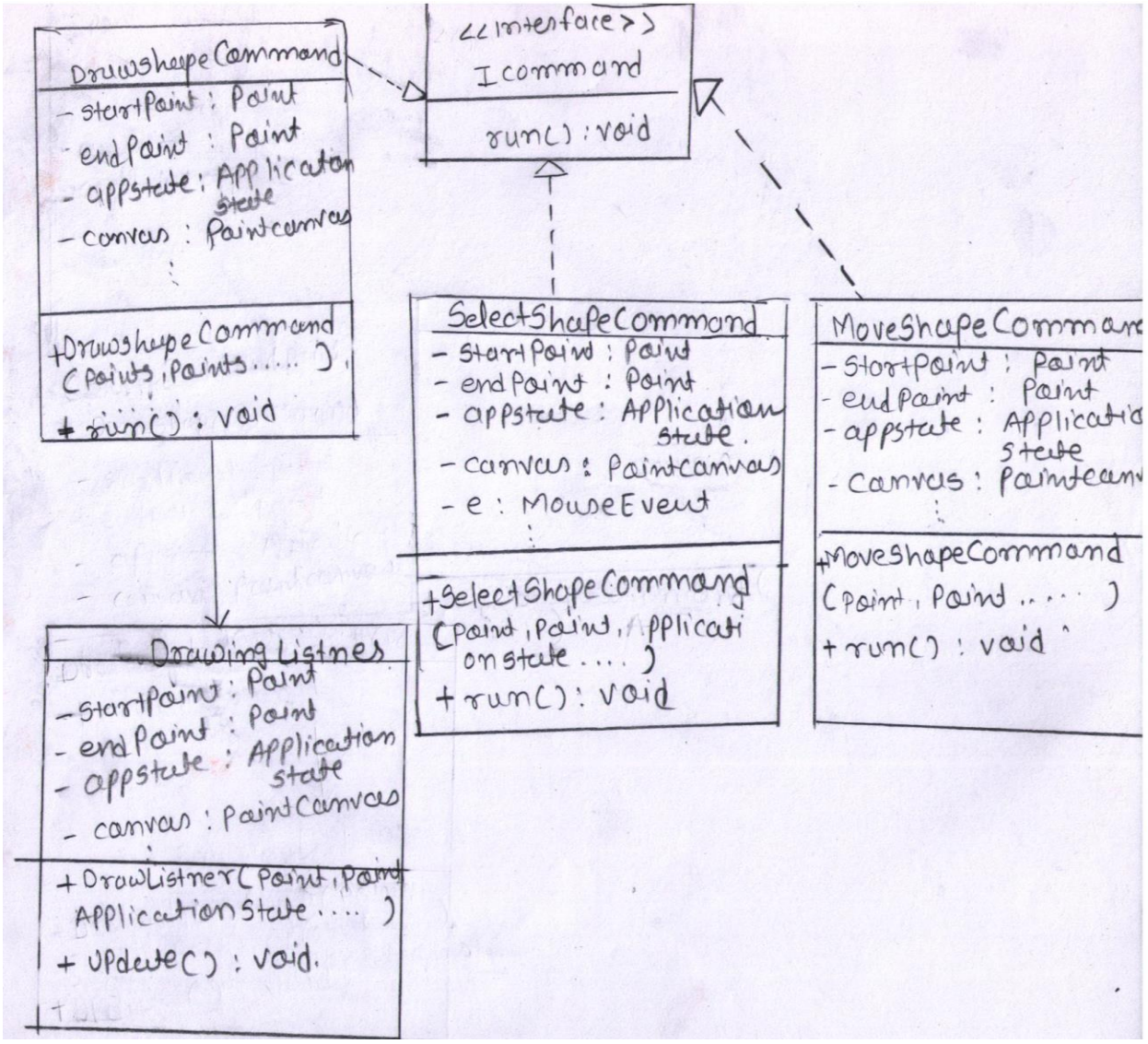
Disha Patel

II. Class Diagrams for the Model Classes

A. Static Design Pattern

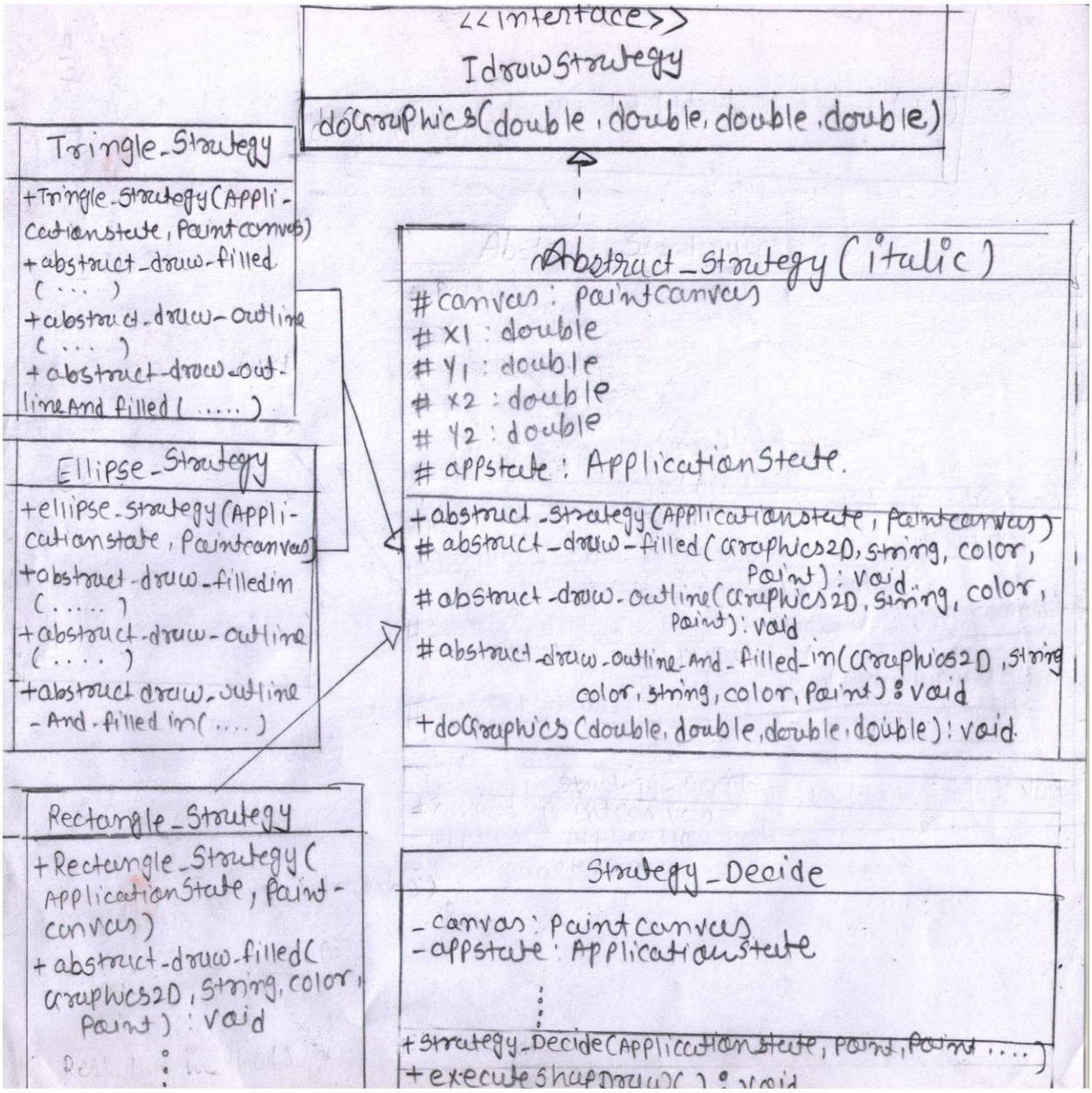


B. Command Design Pattern



Disha Patel

C. Strategy Design Pattern + Template Design Pattern



III. Time Recording Journal

Date	Design (any activity other than coding and debugging)	hr in Design	Coding and debugging	hr in coding	Dealing with BIG bugs	hr in big bug
9-Jan						
10-Jan						
11-Jan	wiring the code which was given by Prof. sharpe	2	was trying to connect IDialogChoice with event	5	was not able to wiring up the codes. Could not able to understand how to wire up.	8
12-Jan	wiring the code which was given by Prof. sharpe	2.5	was trying to connect IDialogChoice with event	5	was not aware on how to use IDialogChoice	8
13-Jan	wiring the code which was given by Prof. sharpe	3	was trying to make enum classes	10	was not aware on how to use IDialogChoice and how to connect with enum classes	10
14-Jan	wiring the code which was given by Prof. sharpe	2	was trying to make enum classes	10	was not aware on how to use IDialogChoice and how to connect with enum classes	10
15-Jan	wiring the code which was given by Prof. sharpe	1	was trying to make enum classes	5	was not aware on how to use IDialogChoice and how to connect with enum classes	10
16-Jan	wiring the code which was given by Prof. sharpe	1	made paintCanvas new object and pass It down to IGUIwindow.	2	was not aware on how to use IDialogChoice and how to connect with enum classes	12
17-Jan						

18-Jan						
19-Jan	wriring provided by Prof. Sharpe(checking on lambda)	10	sample program was checking on lambda and dependency injection	2		
20-Jan	checking on lambda, dependency injection	12	checking on drop down event of canvas	3		
21-Jan	trying to understand how code is working	5	checking on drop down event of canvas	2.5		
22-Jan	trying to understand how code is working	5	checking on drop down event of canvas	10		
23-Jan	trying to understand how code is working	10	Swing and AWT checking	4		
24-Jan						
25-Jan						
26-Jan	checking on static factory for shapes	18	did sample program on static factory for shapes	3		
27-Jan	connect selected shape name from drop down to call specific class in Static factory	5	created shapes class for static factory and shapeFactory class	8	shapes were returning null due to missing return statement at the end of if else in factory.	3

28-Jan	connect selected shape name from drop down to call specific class in Static factory	5	take recently selected shapes and call appropriate class from the factory	10	was not able to call appropriate shape class from shape factory	8
29-Jan	connect selected shape name from drop down to call specific class in Static factory	3		16	once we reselect the shape from drop down, was not able to call the reselected class in factory	10
30-Jan	connect selected shape name from drop down to call specific class in Static factory	6		10	once we reselect the shape from drop down, was not able to call the reselected class in factory	16
31-Jan						
1-Feb						
2-Feb	reunderstanding of wiring up code given by Prof. Sharpe	10	able to call appropriate class after changing the selected shapes from drop down	12		
3-Feb	trying to understand property of Rectangle	5	checking and trying sample program for rectangle	10	was not able to understand how to call canvas class and how to use getGraphics method of canas class. Slowly understood I have to call the canvas class and	3
4-Feb	trying to understand	6	checking and trying sample	8		3

	property of Rectangle		program for rectangle		Graphics2D can use like g,drawRec()…	
5-Feb	trying to understand property of Rectangle and Graphics and canvas class	9	checking and trying sample program for rectangle	8		3
6-Feb	trying to understand property of Rectangle and Graphics and canvas class	11	checking and trying sample program for rectangle	5		1
7-Feb						
8-Feb	Mouse Events checking	5	making small programs with mouseAdaptor to understand it	12		
9-Feb	thinking to add point class for getting point and how can connect those with shape classes	8	Rectangle was getting points but was unable to draw it	18	was not able to get startPoint and endPoint	15
10-Feb	Mouse Events methods and connectiong with shape classes	3	addMouseListener was added on wrong place hence was taking null value	14	addMouseListener added and got startPoint and endPoint	10
11-Feb	Mouse Events methods and connectiong with shape classes	2	posted on D2l for help on canvas	16	addMouseListener was added on wrong place. Hence was taking null	16
12-Feb	Trying to implement	9	thanks to D2l, got reply and	17	shapelist class was not able to implement	15

	observer design pattern		able to draw Rectangle for the first time			
13-Feb	Exam date					
14-Feb			made interface and class for rectangle			
15-Feb	Strategy Design pattern	5	implemented for Ellipse			
16-Feb	making draw for Ellipse	4	implemented for Triangle			
17-Feb	checking how to draw Triangle	2				
18-Feb	Array and Linked Listed(ShapeList Class)	10				
19-Feb	Array and Linked Listed(ShapeList Class)	10				
20-Feb	Observer Design Pattern with Strategy Design Pattern	17				
21-Feb	adapter design pattern for color	4	adaptor method array class with color name and color.awt object	14	unable to understand how can use adaptor method	20

22-Feb	adapter design pattern for color	4	adaptor method array class with color name and color.awt object	5		
23-Feb	Reflection used for changing color string to color object	3	reflection class to changing color String name into Color Class object	6	convert color string from appState to color object	3
24-Feb	how to implement outer line for all three shapes ,stock and methods of Graphics2D	4	setStroke(new BasicStroke(4)) added for outerline	4	how can make thik line for outer edge	4
25-Feb	undo redo	3	tried to implemene undo redo but could not	15	was trying to add in uiModule.addEvent(EventName.COPY, () -> applicationState.Copy()) but could not able to wire up the undo redo button with my canvas	15
26-Feb	copy paste	13	same with copy paste	15	could not able to wire up with the canvas	15
27-Feb						
28-Feb	iterator for arraylist and linked list for selection of shapes	3	loop on arraylist	16	was not able to loop the shapelist and store it in some object of ISHape	3
1-Mar	how to know shapes have been dragged or selected	4	mouseevents for click and pressed	5		
2-Mar	how to know while clicking on shapes, selected points are in which shape	4	shape's start and end point from shapelist linked list class	10	unable to ger start point end point for shapes in the arraylist. It got resolved once I added getter setter in Ishape	10

3-Mar	how to know while clicking on shapes, selected points are in which shape	5	shape's start and end point from shapelist linked list class	11		
4-Mar	change color of primary and secondary color	1	mouse right click event	8		
5-Mar	Select multiple shapes at a same time	7	selecting multiple shapes at a same time(dragging over many shapes)	8	mousedragging not able to add in code	12
6-Mar	Move selected shapes, copy paste short cuts	6	moving and short cuts	6	copy cut short cuts and wiring got failed	12
7-Mar	Templer Design pattern	3	Templer Design pattern	4		
8-Mar	Documentation	10				
9-Mar	Documentation + Class Diagrams	10				

IV. Time Summary

- Throughout the project, I spent most of the time on Design pattern implementation and learning on Swing, AWT, Collection classes and methods of JAVA. Below is the summary of weekly breakdown of the amount of time I spent each week working on this project.

Dates	Jan 9 – Jan 16	Jan 17 – Jan 23	Jan 24 – Jan 30	Jan 31 – Feb 6	Feb 7 – Feb 13	Feb 14 – Feb 20	Feb 21 – Feb 27	Feb 28 – Mar 6	
Week #	1	2	3	4	5	6	7	8	total
Design (hrs.)	11.5	42	37	41	27	48	31	33	270.5
Code (hrs.)	37	21.5	47	43	77	0	59	68	352.5
Big Bug (hrs.) - Bug Descriptions summary	58	0	37	10	56	0	57	37	255

V. Notes on Design Patterns

I have implemented Static Factory, Command, Strategy and Template design patterns for JPaint program.

A. Static Factory

Static Factory Design Pattern is used for selecting shapes like Rectangle, Triangle or Ellipse. It will simply return instance of the interface of the selected shape's class.

By using Static Factory Design Pattern, every time when we want to add some new types of shapes in the JPaint program, we can just add the new class. We need to update only at one place, which is factory class. So, we can easily maintain in future.

The ability of static factory methods to return the same object from repeated invocations allows classes to maintain strict control over what instances exist at any time.

Classes used:

- IShape: interface for the static factory class.
- Ellipse: this class have the details for ellipse startPoints, endpoints and containpoints() method to check whether points are inside the ellipse or outside.
- Rectangle: this class have the details for Rectangle startPoints, endpoints and containpoints() method to check whether points are inside the Rectangle or outside.
- Triangle: this class have the details for Triangle startPoints, endpoints and containpoints () method to check whether points are inside the Triangle or outside.
- ShapeFactory: This class contain simple if else statement which return the "IShape shape" object on chosen shapes.

B. Command Design Pattern

The command design pattern is implemented in this program for Draw, Select, Move, Undo, redo, some specific shorts cuts. We can store the shapes in Linked List or array List and then can perform operation of undo redo operation.
One more advantage of using command pattern over here is we can handle request at variant timed or in variant orders like draw, select, move etc.

Classes:

- ICommand: interface have only run() method without body .
- DrawShapeCommand: DrawShapeCommand implements the ICommand interface and hence it will provide body of run() method. The class is creating the object of static factory according to the shapes and then adding the shapes into linked list. After that, class is calling listener using update method to draw shapes on canvas.
- SelectShapeCommand: SelectShapeCommand is implementing ICommand interface. When the mouse got clicked, the SelectShapeCommand class check for the mouse clicked point and verify whether the points are part of any shapes or not.
    1. If the points are part of any shapes, then it will make dashed border against the shape and add that shape in selected shape array List.
    2. And if the shape has ShapeShadingType as “Outline_and_filled_in” then, right click will change the primary and secondary color for it.

- MoveShapeCommand: MoveShapeCommand is implementing the ICommand interface. When the shape got selected from SelectShapeCommand, we can redraw that shape by dragging mouse event.

C. Strategy design pattern

The strategy pattern allows us to change the behavior of an algorithm at runtime.

Firstly, I have done all codes with strategy pattern for drawing and set the colors of the shapes but then I found many pieces of code redundant. Like below was common code which was repeating in 3 classes Rectangle, Triangle and Ellipse.

```
public void addPoint(double x1, double y1, double x2, double y2) {
      Point p1 = new Point(x1,y1,x2,y2);

      Graphics2D g =    canvas.getGraphics2D();

        if(appState.getActiveShapeShadingType().name().equalsIgnoreCase("Filled_in"))
        {
            String fillColor = appState.getActivePrimaryColor().name();
            Color c = ColorSet.stringToColor(fillColor);
            g.setColor(c);
            g.fillPolygon(new int [] {(int) x1,(int) x2,(int) x1},new int [] {(int) y1,(int) y2,(int) y2},3);

        }
        else if (appState.getActiveShapeShadingType().name().equalsIgnoreCase("Outline"))
        {

            g.setStroke(new BasicStroke(4));
            String outerColor = appState.getActiveSecondaryColor().name();
            Color c1 = ColorSet.stringToColor(outerColor);
            g.setPaint(c1);
            g.drawPolygon(new int [] {(int) x1,(int) x2,(int) x1},new int [] {(int) y1,(int) y2,(int) y2},3);
        }
        else if (appState.getActiveShapeShadingType().name().equalsIgnoreCase("Outline_and_filled_in"))
        {
            String fillColor = appState.getActivePrimaryColor().name();
            Color c = ColorSet.stringToColor(fillColor);
            g.setColor(c);
            g.fillPolygon(new int [] {(int) x1,(int) x2,(int) x1},new int [] {(int) y1,(int) y2,(int) y2},3);
            g.setStroke(new BasicStroke(4));
            String outerColor = appState.getActiveSecondaryColor().name();
            Color c1 = ColorSet.stringToColor(outerColor);
            g.setPaint(c1);
            g.drawPolygon(new int [] {(int) x1,(int) x2,(int) x1},new int [] {(int) y1,(int) y2,(int) y2},3);
        }
```

There was a minor change in each class of the code. All those have a different input parameter as well as methods which we are calling using Graphics2D class. For example, below codes were changing for 3 classes.

- Rectangle:

```
g.fillRect(p1.getULeftX(), p1.getULeftY(), p1.getWidth(),
p1.getHeight());
g.drawRect(p1.getULeftX(), p1.getULeftY(), p1.getWidth(),
p1.getHeight());
```

- Triangle:

```
g.fillPolygon(new int [] {(int) x1,(int) x2,(int) x1},new int [] {(int)
y1,(int) y2,(int) y2},3);
```

```
g.drawPolygon(new int [] {(int) x1,(int) x2,(int) x1},new int [] {(int)
y1,(int) y2,(int) y2},3);
```

- Ellipse :

```
g.fillOval(p1.getULeftX(), p1.getULeftY(), p1.getWidth(),
p1.getHeight());
g.drawOval(p1.getULeftX(), p1.getULeftY(), p1.getWidth(),
p1.getHeight());
```

One more advantage of using Strategy Design pattern was we can add new shape class without modifying anything in present code. In case we don't want to provide color selection for some newly added shapes in JPaint program, for instance line, then using Strategy design pattern we can directly implement the IdrawStategy interface and add appropriate methods for newly added class.

D. Template Design Pattern

To overcome code redundancy problem, I added Template Design pattern. In this method, abstract class have implemented interface and one method which have everything common for all 3 classes.

By using this, code was reduced in each shape classes and modification in future becomes easy.

Classes:

- IdrawStategy: IdrawStategy is the interface for Strategy.
- Abstract_strategy: Abstract strategy is the abstract class which implements IdrawStategy. The class have 4 methods, in which only one is defined over here. Rest 3 is dependent on behavior of sub class.
- Strategy Decide: Starategy_Decide class is use for calling the command design patterns classes. Draw, select and Move Classes called from here.
- RECTANGLE_Strategy: RECTANGLE_Strategy extend the Abstract_strategy and override the methods of Abstract_strategy as per RECTANGLE_Strategy required.
- Ellipse_Strategy: Ellipse_Strategy extends Abstract_strategy and implements 3 methods in own way to draw ellipse.
- Traingle_Strategy: Traingle_Strategy extends Abstract_strategy and implements 3 methods in own way to draw Triangle.

VI. Successes and Failures

A. Successes

1. Able to draw different shapes with primary and secondary color.
2. I can draw shapes with different shading types
3. Able to store the IShape object in linked list and Array list
4. In the 'select' mode, able to click on the shapes and can change primary and secondary color using right click mouse event.
5. Did lots of google and learned about swing and awt packages of java from online examples.
6. Dependency injection and object-oriented concepts became clear.
7. Array list and linked list took lots of time for implementation and understanding.

B. Failures

1. Design pattern:
    - Observer design pattern was difficult to add in the code as I have implemented Strategy design pattern for drawing shapes.
    - Have tried to implement Adaptor design pattern for color class but could not make it due to trouble with collection classes of java.
2. Shape Dragging is not possible to implement.
3. For the changing of primary color to secondary or vice versa, if added last shape have shading type 'outline and filled in'. I can change the color of shapes which doesn't have shading type 'outline and filled in'. (bug in code)
4. I could not wire up undo, redo and delete buttons of canvas.
5. Moving of shapes is not implemented as i could not understand how to write code for that piece of functionality.

VII. Final Product: