Programming Final Project Report

Mohamed Mohab Elbattawy 7206 Group 4 Section 2

Ahmed Hany Ismail 7248 Group 1 Section 1

Youssef Sameh Fayek 6911 Group 1 Section 2

Mostafa Mohamed Magdy Daebis 6874 Group 4 Section 1

************USE CASE UML ATTACHED IN THE PROJECT FILES***********

******** CLASS DIAGRAM UML IN THE PROJECT FILE DUE TO SIZE********

1.How to use:

The user has to first select a color and then choose which shape he wants to draw, should the user wish to do any of the other functions he simply has to click on the radio button containing the text with the wanted function.
Note: copy and pasting should be done at the same time, the user cannot paste the shape if he clicked on another radio button.

2.Operations:

All of these Operations check that the cursor is placed on an object prior to it's execution using contains(x,y) method which returns false if the cursor isn't placed on the shape and do not execute

2.1 Move:

The move is done by changing the starting coordinates of the Rectangle, Circle and Square (x,y).

The Line is moved by changing the starting and ending coordinates (x1,y1,x2,y2).

The Triangle is moved using built in java function translate() specific to objects of type Polygon.

2.2 Resize:

Square, Circle and Rectangle are all done by adjusting the length and width according to the mouse wheel action, the Line is done by adjusting the coordinates of the starting point to increase according to mouse movement and the ending coordinates decrease according to mouse movement.

Triangle resizing is done by decreasing x[0] (bottom left coordinate), y[1] (the only upper triangle coordinate) and increasing the other coordinates except for x[1] which is left unchanged. The coordinates will be explained in detail in the Rectangle Part.

2.3 Copy:

The user clicks on the shape with the left mouse button and it's then added to the clipboard, the user then right clicks to paste the image (the copy radio button should still be clicked and not change modes).

2.4 Modify Color:

The color of the shape is modified according to new color selected and then pressing on the shape.

2.5 Delete:

The shape is deleted after being clicked on.

2.6 Undo-Redo:

Redo does nothing if no action has been undone and the undo does nothing if no single action has happened at least once.

3.Shapes:

3.1 Rectangle:

Rectangle is made by Rectangle2D.Float, the move is done by moving the starting position(x,y) and the resizing is done by increasing/decreasing the length and width according to mouse wheel motion.

3.2 Circle:

Circle is made by Ellipse2D.Float and making the width=height to be made a Circle otherwise it will be an ellipse, the move is done by moving the starting point (x,y) and the resizing is done by increasing/decreasing the length and width according to mouse wheel motion.

3.3 Square:

Square is made by Rectangle2D.Float and making the width=length, moving and resizing is done in the same manner as the rectangle.

3.4 Line:

Line is made by Line2D.Float, it has starting and ending points (x1,y1,x2,y2) and those are changed to move, starting points x1,y1 increase with mouse wheel movement while x2,y2 decrease with mouse wheel movement.

3.5 Triangle:

Triangle is made using Polygon of 3 points, there's an array of x points and an array of 3 points, each have 3 points(array of x points, array of y points, int of number of points). Coordinates x[0] and y[0] are the starting points which represent the bottom left corner, x[2] and y[2] present the bottom right corner while the x[1] and y[1] represent the upper corner, this logic is reversed if the triangle is drawn upside down. The special point is x[1] which is the middle between x[0] and x[2] as it's calculated and not actually drawn by the user (logic wise).

The move is done using translate() function and the resizing is done by increasing the coordinates except x[0] and y[1] which decrease with the mouse wheel movement.

4.Design Patterns Implemented:

1.Factory Pattern

2.Singleton Pattern

3.Prototype Pattern

4.Memento Pattern

5.Iterator Pattern

6.Adapter Pattern

7.Strategy Pattern

1.Factory Pattern:

Used when there's a selection of objects and we need a specific object (rectangle from shapes for example), implemented in ShapeMaker, ShapeMover, ShapeResizer, and the Adapter.

2.Singleton Pattern:

Used on GUI to ensure there's only 1 active window, implemented in the ''paint'' GUI class.

3.Prototype Pattern:

Used to copy the LinkedList (ShapeList) and ColorShape List, ColorShape used to copy the shape while the LinkedList needed for the undo/redo (you need the copy to undo/redo but the memento is the one responsible for the undo/redo itself).

4.Memento Pattern:

Required for undo/redo, Classes are Originator,CareTaker and Memento, CareTaker has an arraylist of Memento, the Memento has an arraylist of the saved shapes which is used in the undo (go back in the arraylist) and redo (go forward in the arraylist), Originator sets and modifies the Memento.

5.Iterator Pattern:

Used to Iterate in the LinkedList containing the shapes, checks if there is next element in linkedlist, implemented in all operation Classes (MoveAdapter,ResizeAdapter etc…), it's "called ListIterator".

6. Strategy Pattern:

Determines the best type of function (move,draw etc…) according to shape chosen or clicked on. The Shape is determined and then the most suitable function is called upon (example when clicked on rectangle the move rectangle function is called and not move square).

7. Adapter Pattern:

Made in order not to have many if or switch cases inside the code and the classes deal with only 1 function (Adapters) that determine the function required and implements the required functionality only, implemented in all Adapter Classes.

5. Algorithms:

All Adapter Classes interact with CareTaker Class to store the current data needed for undo/redo

5.1 Choosing the mode:

The Program starts with the paint class which is the GUI window, each radio button is given a name and according to the button clicked the program enters one of it's partitions (move , copy, draw etc…) , the class adapter factory is the one responsible for deciding which functions to execute based on the input it receives (the input here is the name of the radio button according to the user click).

5.2 Drawing:

Should the user click on any of the 5 draw radio buttons of draw he is taken to the class DrawAdapter, the Class decides what shape it will be drawing according to the string argument that it is given, the string is the name of the shape as we previously stated all radio buttons are given names. As soon as the mouse is pressed current coordinates are registered and the ShapeMakerFactory is used, the ShapeMakerFactory is the class responsible for making the shapes indirectly as it sends the coordinates that were given to the appropriate class LineMaker,RectangleMaker etc… according to the String that was attached with the coordinates and that's why the ShapeMakerFactory takes the coordinates as well as the shape name, the start points (x1,y1) are obtained from the mouse pressed event while the end points (x2,y2) are obtained from the mouse dragged event. These 5 classes LineMaker,SquareMaker,TriangleMaker,CircleMaker,RectangleMaker are the classes which have the math and logic behind the drawing, the Line has a start and end point, the Circle, Square and Rectangle only have a starting point but have width and height while the triangle is a polygon that has an array of x points and y points and the number of points, the logic behind the triangle making was explained in the Triangle Section. Every drawn shape is stored in an arraylist (Shapelist).

**** Before any of these upcoming algorithms are done it is 1st checked that when the mouse is pressed the cursor was on an object and not on a blank space using contains(x,y) method and iterating the entire linkedlist using the Iterator to validate each shape***

5.3 Moving:

Following the same Adapter principles, the AdapterFactory enters the MoveAdapter after clicking the radio button move and the MoveAdapter sends to the ShapeMoverFactory which decides which move algorithm to use based on the shape clicked. The classes with the move makers are LineMaker, SquareMaker, RectangleMaker, CircleMaker, TriangleMaker. These are the classes that contain the logic behind the moving of each line which was explained earlier in the report.

5.4 Resizing:

Same Principles as Moving, the Adapter Factory enters the ResizerAdapter, and then the ShapeResizerFactory. The Resize Classes are RectangleResizer, SquareResizer, CircleResizer, TriangleResizer, LineResizer. The resizing of each shape was explained early in the report.

5.5 Copying:

left click is MouseEventButton==1,right click is MouseEventButton==3

Same Principles as before, AdapterFactory enters the CopyAdapter, the left mouse click is responsible for validating that the mouse is on a shape (Boolean found becomes true). The right click is responsible for making a copy and pasting it at the new coordinates. The logic behind the new coordinate allocation differs from object to object as defines by the SquarePaster, Circle Paster etc… Classes. ShapeList LinkedList is updated with the new copied shape after pasting and repainting is done.

5.6 Undo-Redo:

Using Memento Pattern, when Undo is done Previous Memento is the new list while the redo makes the next memento the new list, when a new shape is drawn the current memento is updated.

5.7 Modify Color:

Global Variable next color is changed according to the Color Palette, and the click on the shape changes the current shape color (this.shape.setcolor) to the new color and repainting is done.

5.8 Delete:

The Clicked shape is deleted from the LinkedList using .remove and repainting is done.

6.Solid Principles:

All Classes were Single Responsibilities closed for modification open for extension, No excess classes were made. Liskov's principles were used as super classes were used most of the time.