The Developed Application JPaint is like Microsoft Paint. JPaint allows us to draw three kinds of shapes, those are triangle, ellipse, and rectangle. We can perform select, copy, paste, move, delete, and clear operation on this application.

## Features

* **Draw:** It allows us to draw the three shapes such as triangle, ellipse, and rectangle on the canvas. The Default shape is ellipse and default Start and End Point mode is Draw, it calls the ShapeDrawOperation..JPaint allows us to draw shapes with different colors. The primary color is for shape’s outline and secondary color is shape’s fill. We can draw the shapes on three different shadings like Filled-in, Outline, and Outline and Filled-in. Filled-in allows us to draw filled-in shape with secondary color. Outline is to draw the outline of shape using the primary color. And the Outline and Filled-in allows us to draw shape that is outline and filled-in using both primary color and secondary color.
* **Select:** The select command in the JPaint application is used to select particular shapes to copy, paste, move, and delete. It is included in the Start and End Point menu. If you want to perform select operation you must set Start and End Point as select mode.
* **Move:** If you want to move a particular shape in canvas, then we use move feature. To perform this action you must set Start and End Point as move mode. Move will move only selected shapes while in select mode. This feature is works for triangle, rectangle, and ellipse.
* **Copy:** To perform copy operation first you must select shape in select mode, click on Copy button on canvas to activate copy, it will copy the selected shapes to array list in the ShapeList class. This copy only the selected shapes.
* **Paste:** To perform paste operation first you must select shape in select mode, click on paste button on canvas to activate paste, it will create a duplicate copy of the selected shapes.
* **Undo:** This feature allows us to undo the previous operation, this can be done by using the OperationHistory class. To perform this feature just click Redo button on canvas.
* **Redo:** This is quite opposite to Undo feature, this allows us to redo the previous operation using the OperationHistory class. To perform this feature just click Redo button on canvas.
* **Delete:** To perform delete, select shapes to which you want to delete in select mode, click on delete button from canvas to delete the selected shapes.
* **Clear All:** If you want to clear the canvas, then simply click on clear all button on canvas, it will clear the entire canvas. We can perform this feature in any mode.

# Notes on Design Patterns

This application is developed by using the java language, it utilized five design patterns Factory, Observer, Command, Singleton, and Strategy.

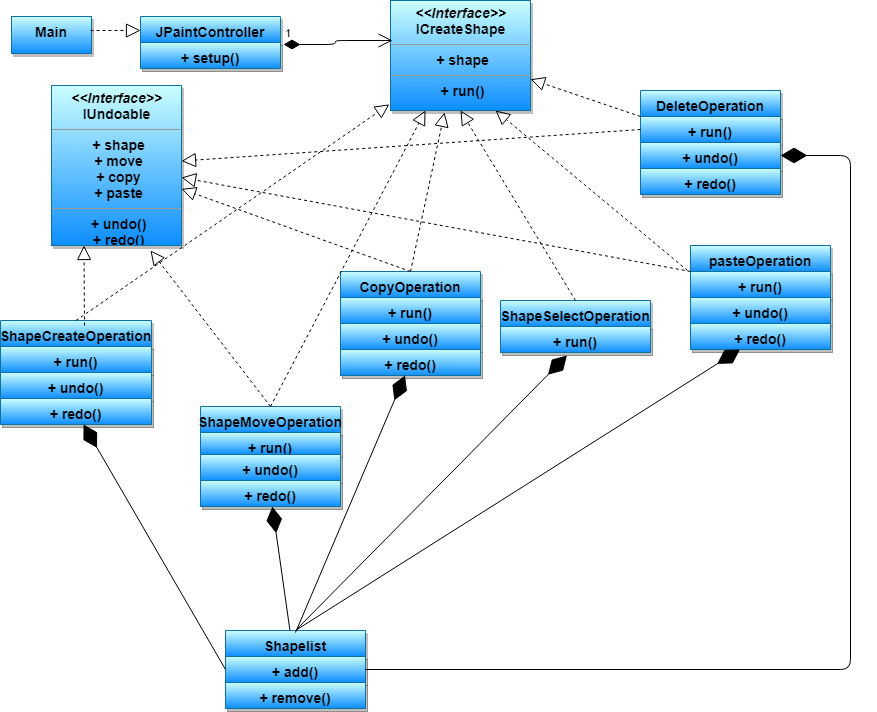
**Factory:** The Factory pattern comes under the creational class design pattern. Factory pattern is often defined as “A method whose sole responsibility is to abstract the creation process of an object and return that object.” Factory pattern is used to create shapes like triangle, rectangle, and ellipse dynamically. Factory pattern provides flexibility, loose coupling, and also allows us to make changes to the applications without changing the code. Based on the shape type selected by the application state, the pattern will create a new shape object, which is invoked by the ShapeCreateOperation Dynamically at run-time. Object creation is taken care of by the classes DrawTriangleShape, DrawRectangleShape, DrawEllipseShape, and interface IDrawShape. This pattern allows us to select which strategy to run at runtime.

**Observer:** The Observer pattern comes under the behavioral object design pattern. This pattern useful in notifying the paint canvas that a new object had been added to the internalShapeList so the new shapes could be either drawn or removed to or from the canvas. This pattern allows the objects to be notified when changes are made. The Observer pattern delivers many advantages in the development of the JPaint application like loose coupling, data transfer. It can be added or removed as needed. To implement Observer pattern we need to implement two interfaces, IShapeListObserver and IShapeListSubject, and add the methods to the subject registerObserver, notifyObserver. This is used in ShapeList. Additionally, it was used to notify the mouse adapter. I used the Observer pattern in the MouseObserver class.

**Strategy:** The Strategy pattern also comes under the behavioral object design pattern. It basically encapsulates algorithms and allows them to be interchangeable at run-time. The abstraction is partial in an interface, and implementation details are found in the classes. In this application strategy used to draw shapes. The classes are DrawRectangleStategy, DrawTriangleStrategy, and DrawEllipseStrategy and the interface is IDrawShape. The pattern has one to many relationships and allows us to add more shapes.

**Command:** It relies heavily on the Command Pattern, a behavioral design pattern, that provides loose coupling and encapsulates a request to either draw, select, move, copy, paste, undo, redo only exposing the run() method and encapsulating everything else. Using the command pattern in this project, made it easier for me to add new commands throughout the development life cycle without having to change existing code making it easier to build out new future features. The command pattern hides the inner details about the command exposing only the call to execute the command. This allowed me to change the functionality of the actual command without having to modify the invocation code. Classes involved include: IOperation interface, ShapeCreateOperation, ShapeMoveOperation, ShapeSelectOperation, PasteOperation, CopyOperation, DeleteOperation, Redo/Undo Operation, ClearAllOperation and mouse adapters which create and run a command dynamically. All command classes include a run method with details the functionality of the command.

## UML class diagram



There are seven classes are used in the UML class diagram such as shapeCreateOperation, shapeMoveOperation, shapelist, ShapeSelectOperation, pasteOperation, deleteOperation, copyOperation, All classes have attributes and operations. The operations are run (). Undo (), redo (), add (), remove (). The shapelist class is an associate with other classes. All classes are implements with two interfaces such as ICreateShape and IUndoable except shapelist class. The ICreateShape interface has a shape and run method. The IUndoable method has undo and redo operations. It has variables such as shape, move, copy and paste.

