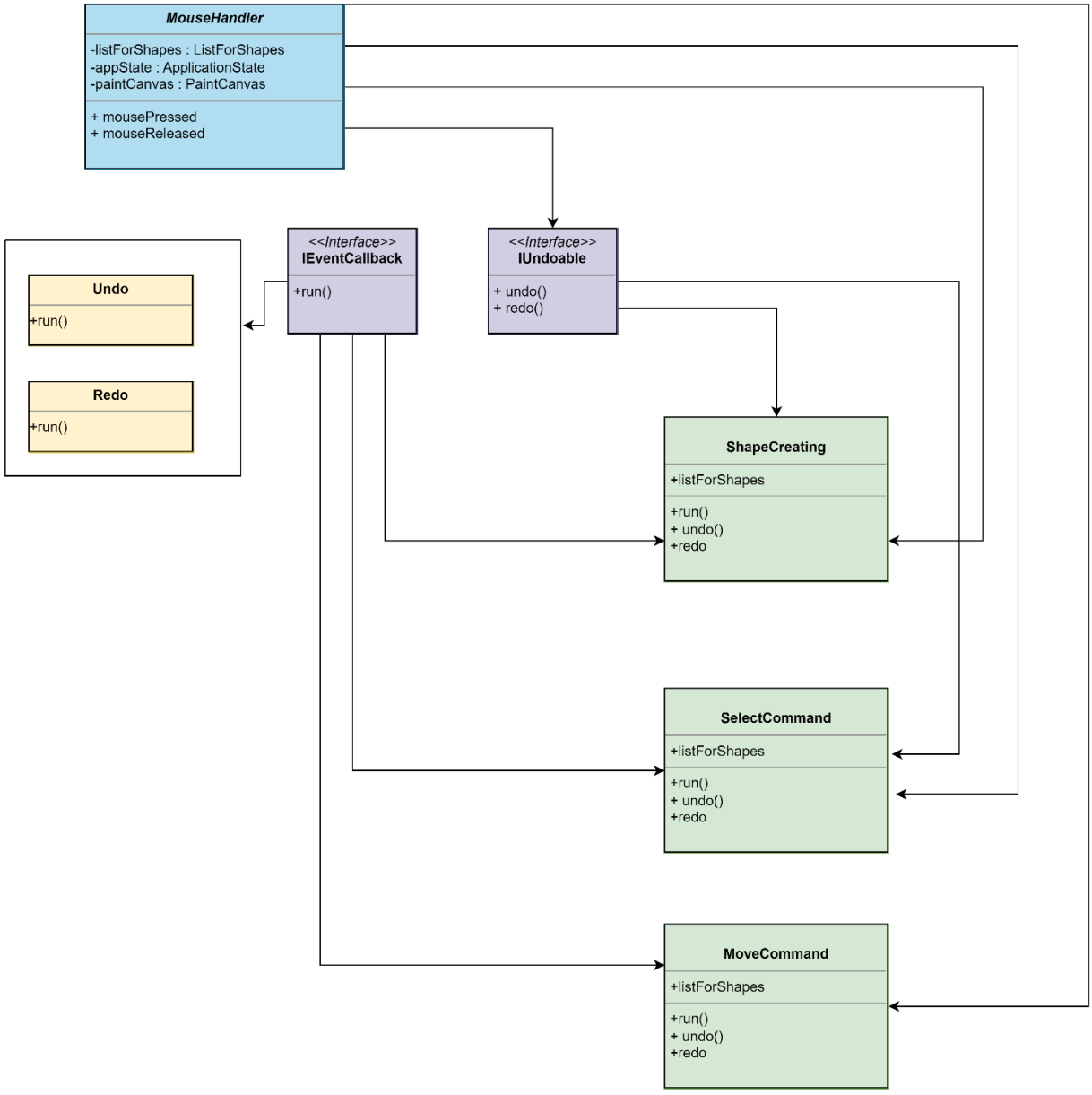
Command Pattern:

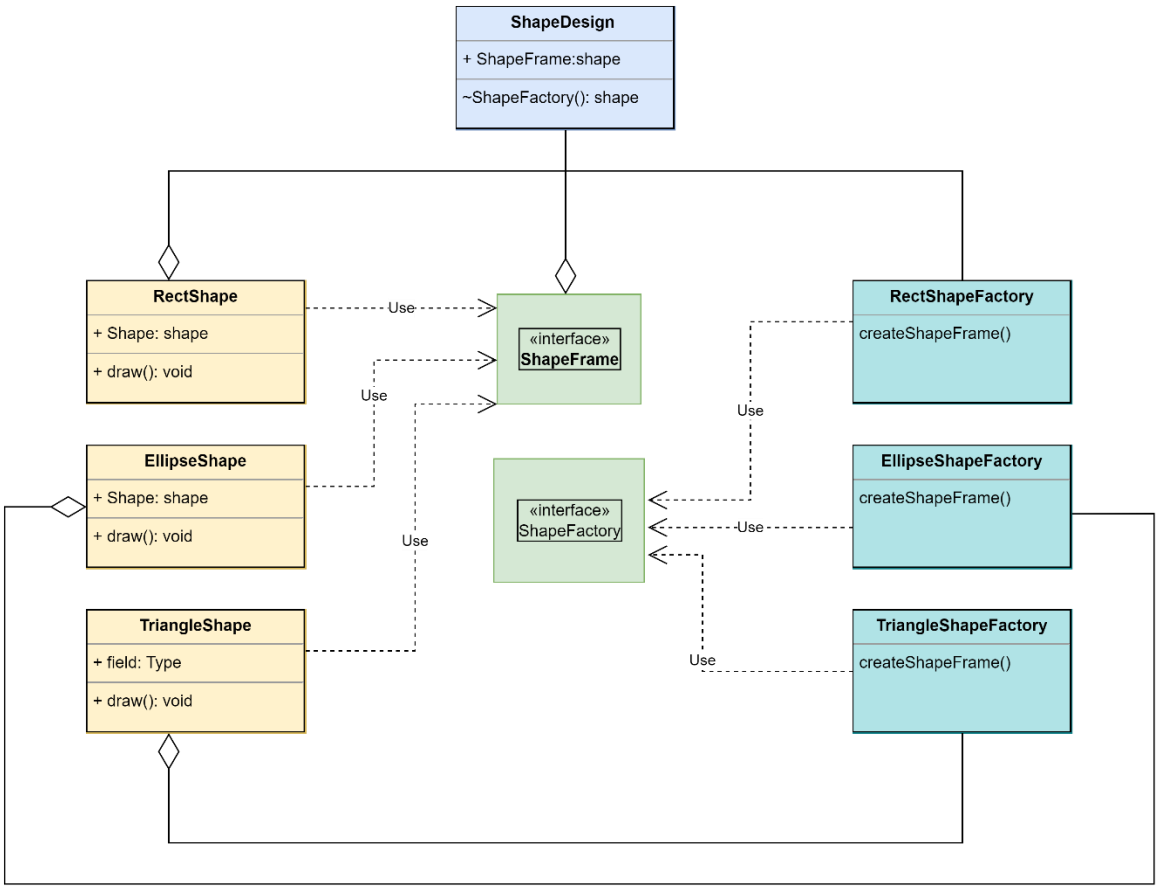
I have used Command pattern to execute most of the command in the JavaPaint application, here I have used the classes like MouseHandler, ShapeCreating, Undo, Redo, MoveCommand, SlectCommand. Because there are numerous commands that must be carried out when a specific action occurs, I decided to use this pattern. since depending on the input or button that is clicked, each command class can implement the IEventCallBack interface and use the method in a different way. The problem of what the mouse should perform based on the selected mouse mode was resolved by the command pattern. The draw command is executed if the mode is Draw. Similarly the select command is executed if the mode is Select. The issue of undoing and redoing instructions that had already been executed was likewise resolved by the Command Pattern



Factory Method Pattern:

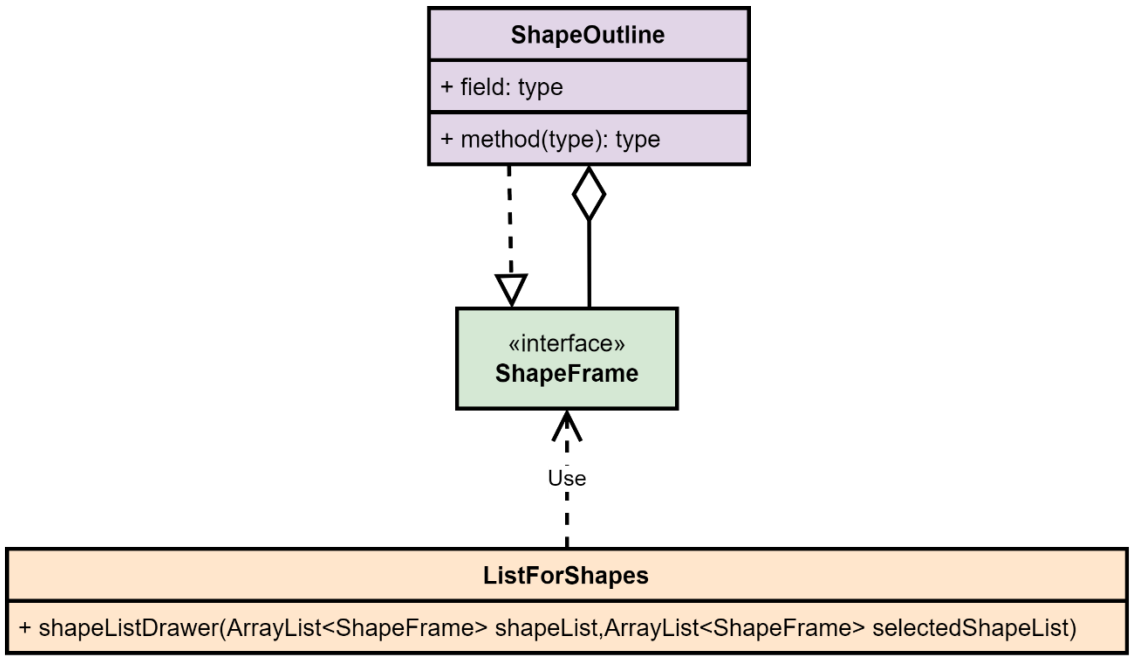
Implemented Factory method pattern which includes classes “RectShape”, “EllipseShape”, “TriangleShape”, “ShapeDesign”, “RectShapeFactory”, “EllipseShapeFactory”, “TriangleShapeFactory” and Interfaces “ShapeFrame”, “ShapeFactory”

Here ShapeDesign class now uses the ShapeFactory interface to create the ShapeFrame objects for each shape. The RectShapeFactory, EllipseShapeFactory, and TriangleShapeFactory classes implement the ShapeFactory interface and create the corresponding ShapeFrame objects.



Decorator pattern:

Implemented the Decorator pattern which includes classes ShapeOutline and ListForShapes and interface ShapeFrame.I have used the decorator pattern to draw the dashed outlines for the individual shapes. Because it allowed for the addition of new functionality to objects without affecting the other objects belonging to the same class, this design pattern was helpful.



Strategy Pattern:

Implemented Strategy pattern with classes FilledInStrategy , OutLineStrategy and OutlineAndFilledInStrategy and Interface ShadingStrategy. I have used the strategy pattern to implement all the ShadingTypes (FilledIn, Outline and OutlineAndFilledIn) by this pattern I have a have a single shared interface, but various classes that behave differently.