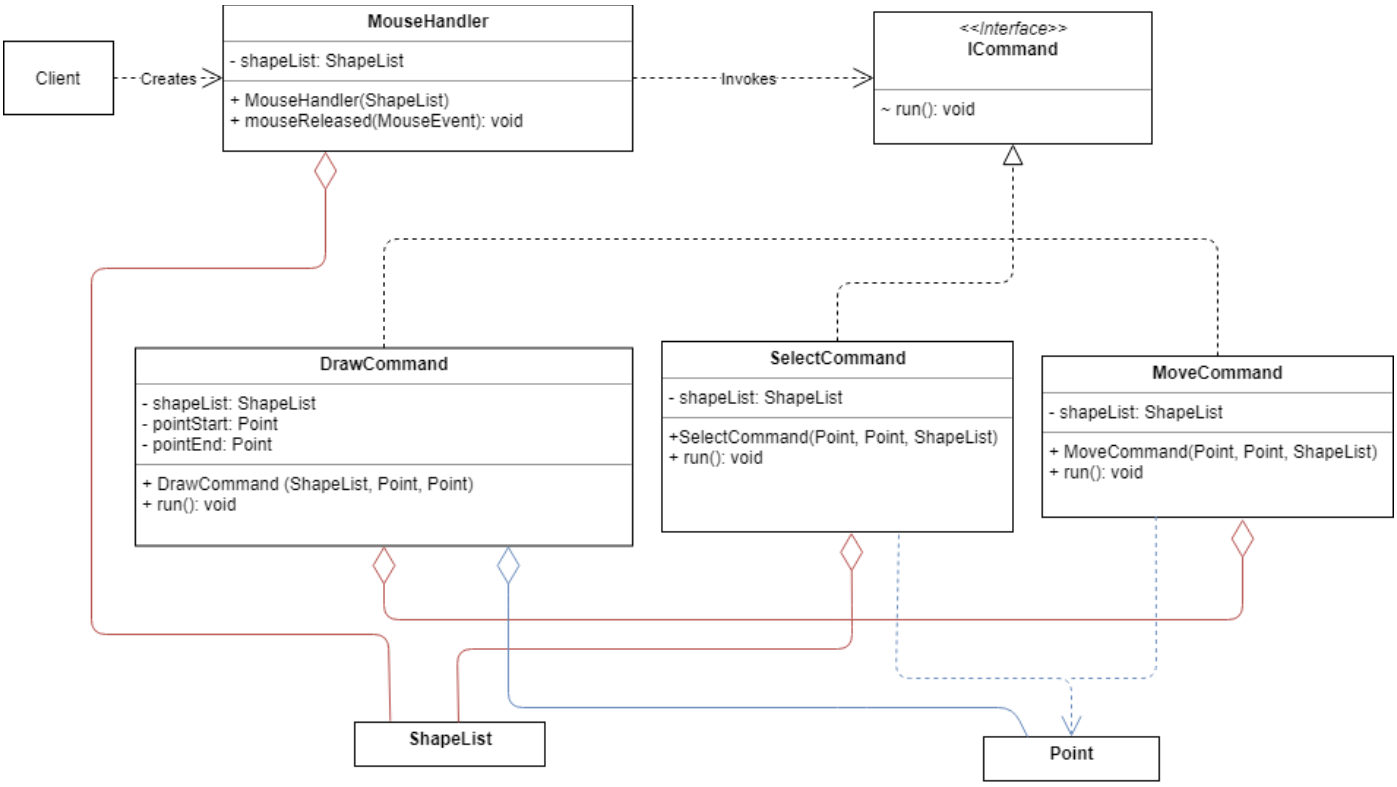
J-Paint Design Documentation

Andrew Richards

8/19/2019

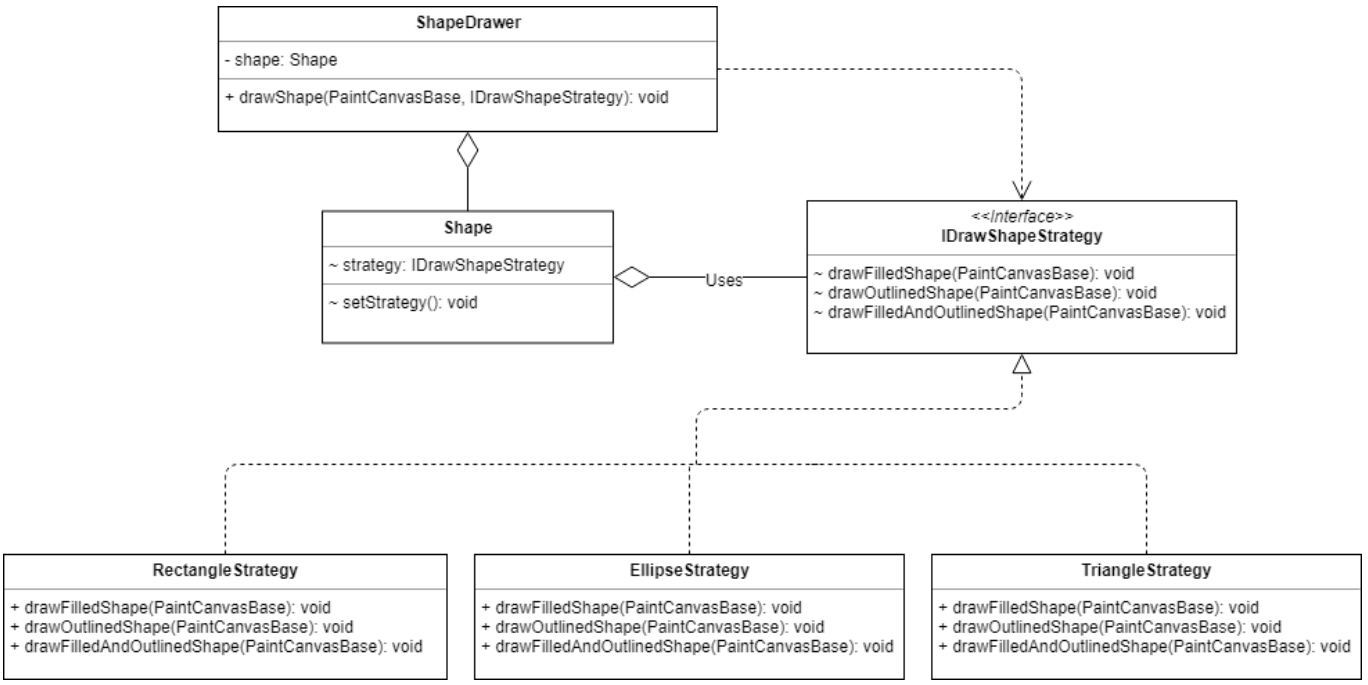
OVERVIEW

Command Pattern



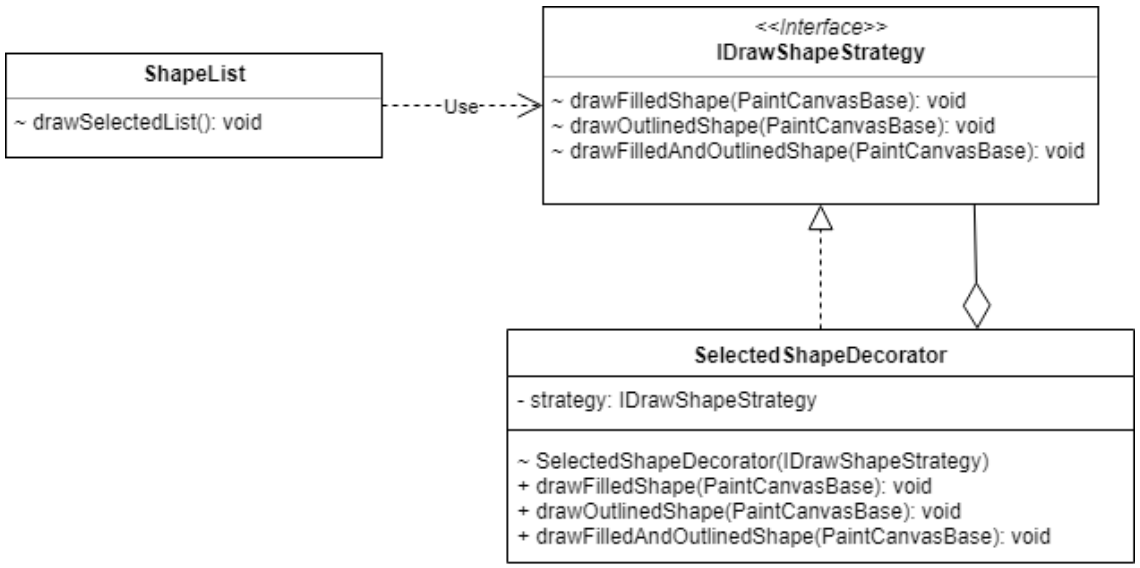
The command pattern uses the MouseHandler class to invoke ICommand, create the command, and call run from ICommand. Mousehandler, DrawCommand, SelectCommand, and MoveCommand have aggregation relationships with ShapeList. Point has an aggregation relationship with DrawCommand, and a dependency relationship with SelectCommand and MoveCommand.

Strategy Pattern



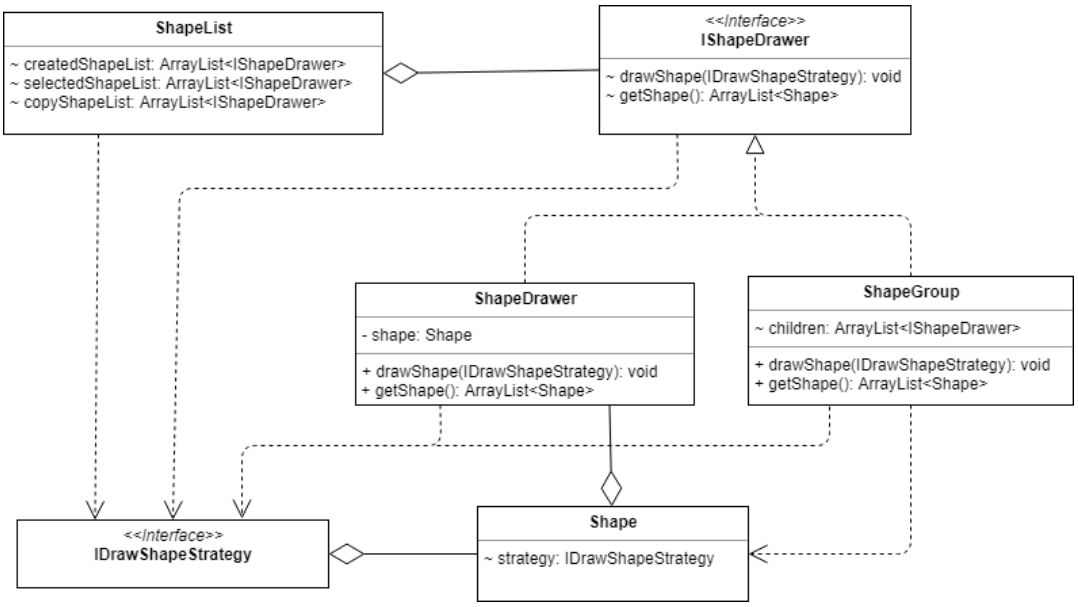
The Strategy pattern uses IDrawShapeStrategy interface to draw different shapes and shading types. The interface has an aggregation relationship with the Shape class. The draw methods are called in ShapeDrawer.

Decorator Pattern



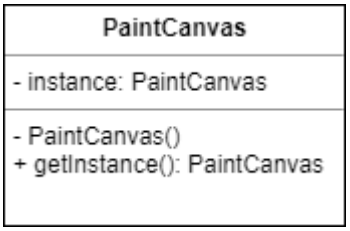
The Decorator pattern uses the SelectedShapeDecorator class which takes in IDrawShapeStrategy object into the constructor and then decorates it with a dashed outline strategy. The ShapeList class redefines the IDrawShapeStrategy field in Shape class for all shapes in selected list.

Composite Pattern



The Composite Pattern uses the ShapeGroup pattern which is a composite of IShapeDrawer objects. It has an array of ShapeDrawer objects which have a reference to Shape class.

Singleton Pattern



The Singleton pattern uses the PaintCanvas class. The only instance in created inside the class, and the constructor is set to private. There is a getter method for the instance.

COMMENTS

Shapes remain selected after move, copy, and paste. Group, ungroup, undo, and redo deselects the shape. Shape strategies are set in the Shape class, drawn with ShapeDrawer

class, and ShapeDrawer's draw method is called in the ShapeList class inside ShapeList's draw list methods.

Regarding challenges I faced, the composite Pattern was difficult to implement because I did not use an abstract IShape interface along with my Shape class. Originally my shape lists held shape objects instead of interface objects. I was unable to add my composite objects to my master, selected, and copied shape lists when grouping. I also originally tried making an abstract factory for my shapes. This did not end up making sense since I did not use separate classes for my shapes for the factory to create. When a new shape object is created, the constructor in the Shape class sets the characteristics of the shape based on the current application state. My factory turned into a separate class called ShapeDrawer for setting the drawing strategy and calling the draw shape method from IDrawShapeStrategy.

In order to solve the issue of my shape lists not holding interface objects, I added an IShapeDrawer interface to ShapeDrawer. I added IShapeDrawer objects to the lists instead of shape objects since ShapeDrawer holds a reference to the created shapes. This allowed me to add references to my lists for both standalone ShapeDrawer objects and composite ShapeGroup objects.