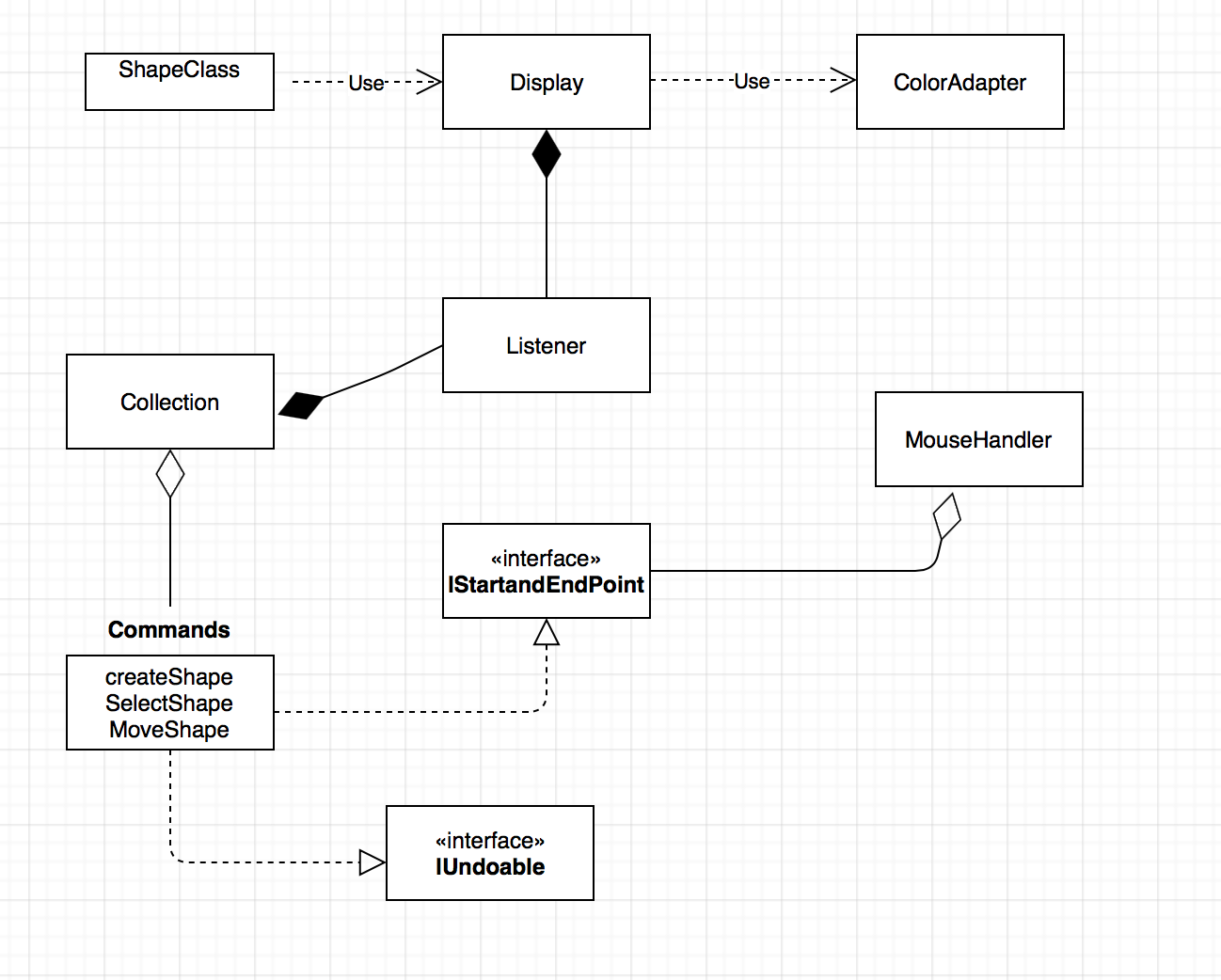
**Model Portion Diagram**

****

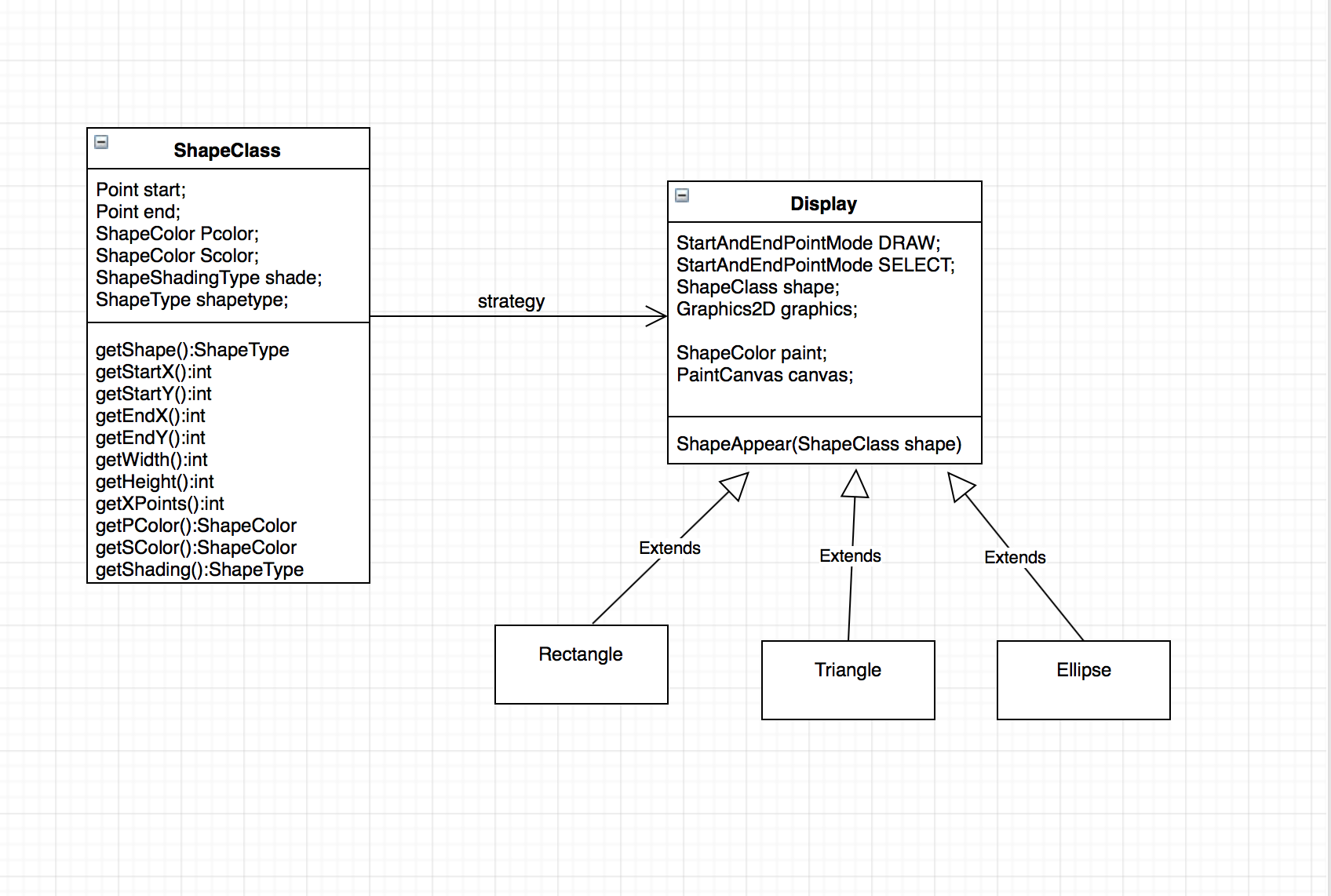
\*For detailed explanation of the relationships and methods, see the section on patterns used.

**Time Summary.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | total |
| Design | 1 | 2 | 1 | 1 | 0 | 4 | 2 | 2 | 2 | 0 | 0 | 15 |
| Code | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 2 | 1 | 1 | 0 | 9 |
| Bug | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 5 | 5 | 12 |

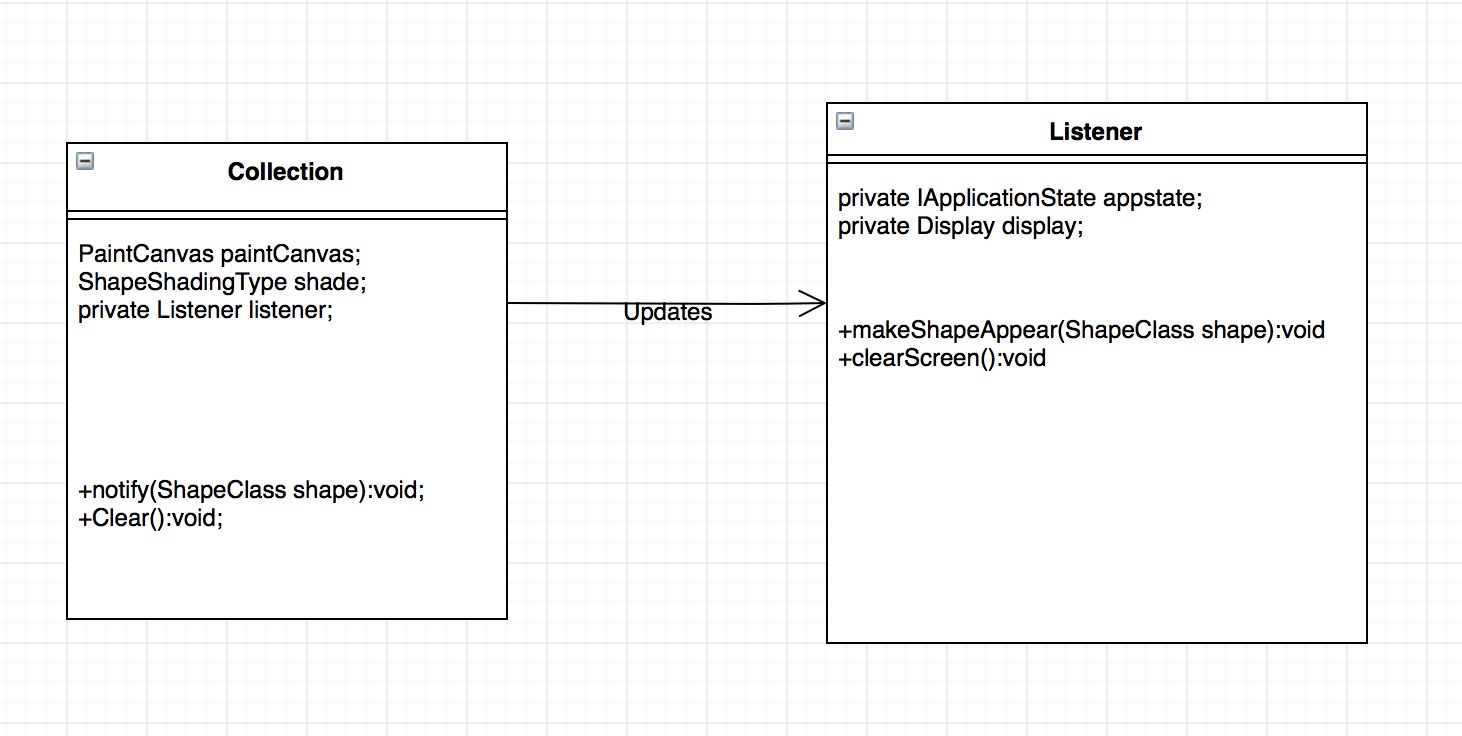
**Patterns**

Strategy:



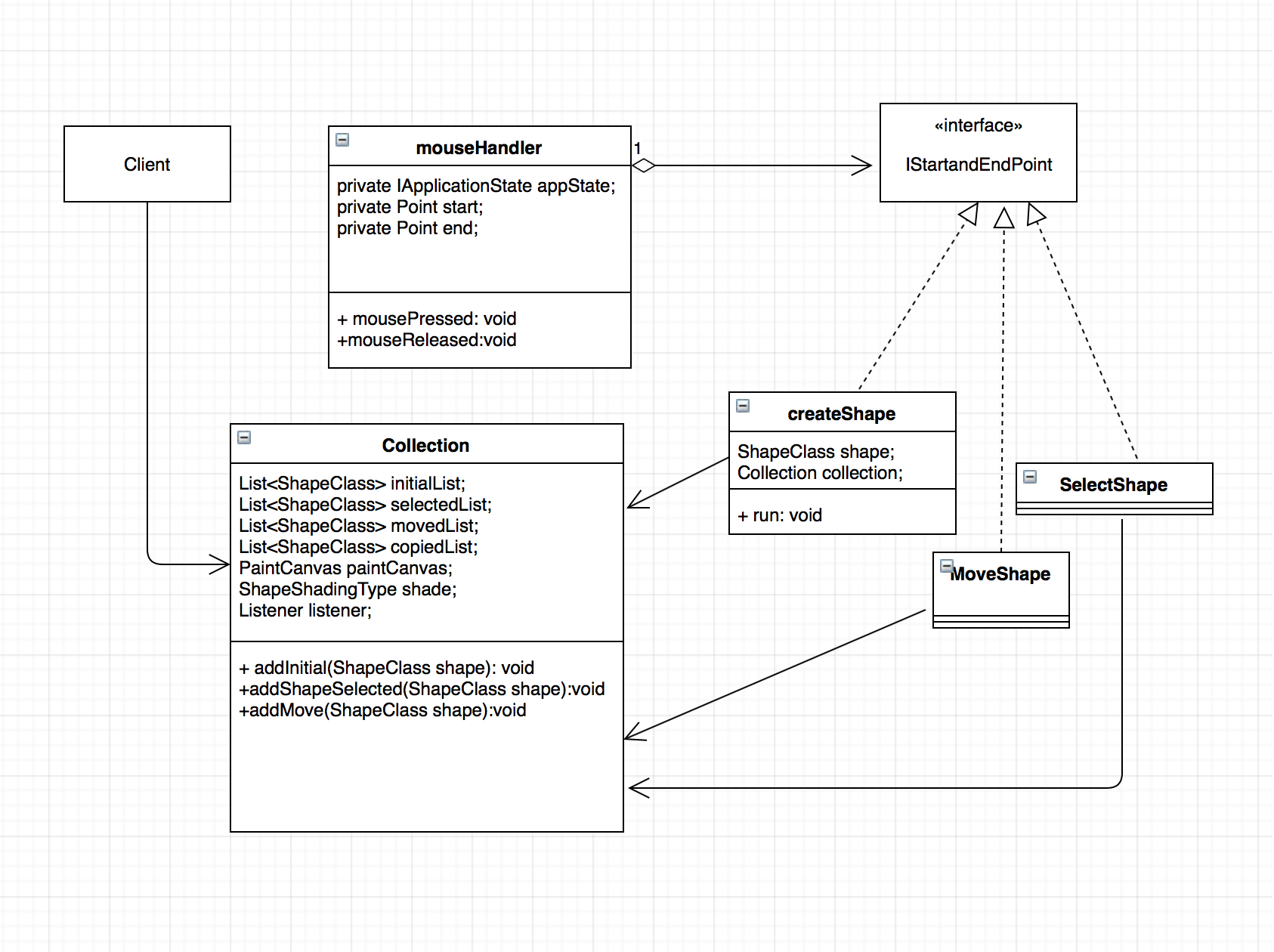
Instead of hard coding different shapes in different classes, the pattern allowed me to store the specific values a shape needs to be drawn, then dynamically call out the values based on the type of shape or color that is passed by the application user; the only thing that changes is the parameters of the ShapeClass, which is the mouse movements in this case.

2. Observer



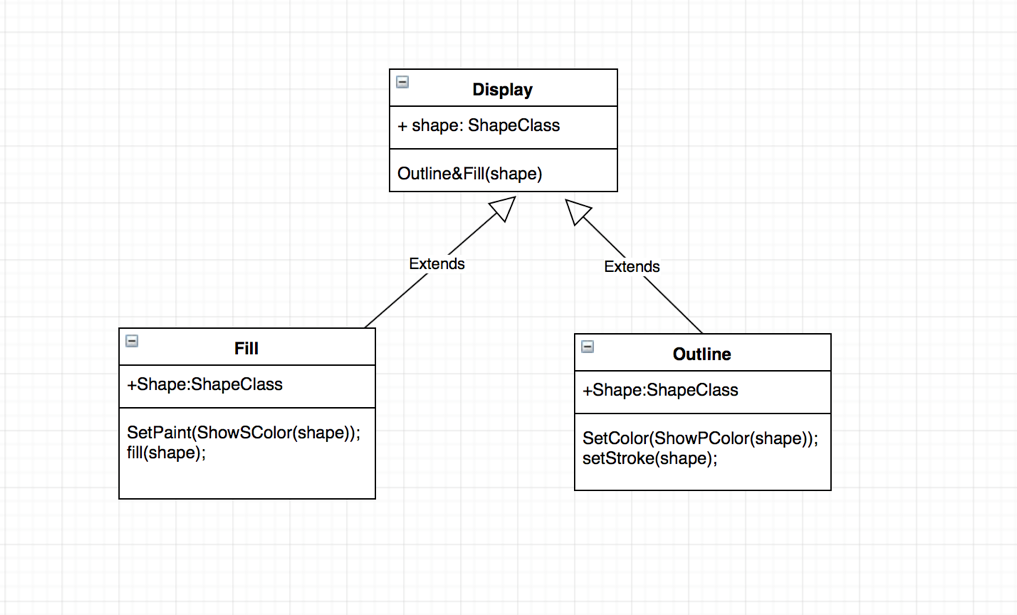
In this case Collection class is the subject that notifies Listener the observer that listens for updates when shapes are being added or removed from collection and displays shapes on PaintCanvas. This pattern allows for the publish-display relationship.

3.Command:



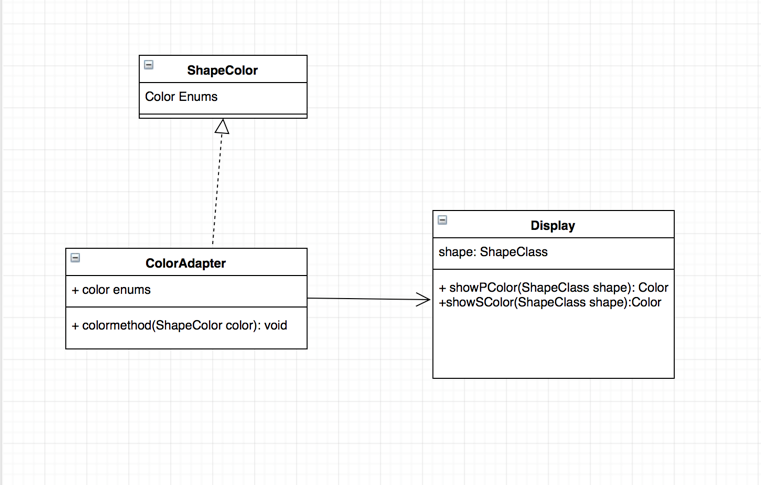
In this project, the commands could be stored as objects in a list, allowing for the functionality of the undo and redo buttons on the user interface.

4.Decorator:



sub-methods act as components to fulfill the operation of a larger method; in this case to outline AND fill a shape; similarly but not shown in diagram, Fill and Outline both use sub methods to get the color.

5. Adapter:



In this project, color enums were converted through Color Adapter as RGB values to be shown on Display.

**Specifics:**

-Please press mouse from bottom left corner and release to upper right corner to generate shapes.

-To select a shape, please select from its bottom left corner and release the mouse at is upper right corner to select, make sure the shape stays within the parameters of the mouse motion.

Same as when selecting shape. When in move mode, move mouse again from bottom left corner to upper right corner motion to generate shape at a desired location; note that the parameters of the new shape is not exact but colors and shapetype are the same. Select and Move can only operate on one shape at a time. But while in Draw mode, you can draw around as you want.

One major bug I had was having a return type in the shapeAppear method, which generated a composite of recs, triangle and ellipse. Originally thought was a lack of breaks, once the breaks were wrote in, due to having the return type, no shapes could be drawn. Caught the bug at the last minute and thus, a lot of time was spent not able to interact with the shapes.

Another stumbling block was the screen not clearing when a shape is deleted from its list, which took some work around that was not altogether elegant.