Assignment 5 covered three previous assignments, for all of the three previous assignments I used the factory design pattern to create the shapes and the singleton design pattern to add styling to the shapes when they are outputted by the display method.

I used the factory method and not the abstract method for the simple reason that during a laboratory a demonstrator recommended to use the method over the abstract. Upon further investigation into the abstract factory it made more sense to use the factory method for simplicity. The abstract factory is more complex as it exists to create a family of related objects, thus requiring the creation of more classes for each different type of object. The factory method is used to create only one object type. In example, for the shape assignment, the factory method would create a factory for creating different shapes but the abstract factory would have two factories, one for rounded shapes (circles, ellipse) and a factory for straight shapes (rectangles), this is more confusing.

Implementing the factory method in assignments 2, 3 and 4 was not too difficult, since it is a design pattern what worked for one assignment could be copied over to another. Previously, In each assignment I would create the shape object in the main program, using the shape class of each assignment. In implementing the factory method, all that changed was I created a factory class containing a method which took a shape string argument and would create a shape object using the shape class, it would return this shape object, specified by the string argument. The main program now only needed to create a factory object and call the method, giving a shape as an argument, and it would receive the shape.

This is why the factory method is an appropriate for use, it made the creation of shapes in the main program easier to read, and made the entire program more modular, thus when a error did occur it was easier to find where it occurred. The factory method was appropriate over the abstract was since we only had to create shapes, while there are different shapes and shapes are the same type, a shape. If were asked to make separate shapes based on number of sides, it would make more sense to use abstract factory for the reason mentioned above, it is used to create a family of related objects, all the shapes of different sides are related by being shapes.

In order to implement styling I used the singleton design pattern. The main reason I used it was for the fact Mark suggested to use it in the tutorial. Since only one singleton object can exist which is accessed by all classes it makes sense to use it for styling since in the assignment only one style can exist and this style is applied to all shapes created.I could have used the factory method but it did not make sense to me how I would implement it and either way it seemed more complex, would it require a style object for each type of style? So a fill object and a stroke object?.

I implemented the style class by creating a singleton sealed class which allowed for only once instance of it to exist. The class contained 3 variables, stroke, fill and stroke width. The only method in the singleton class was the void "addStyle" method, it returned a string of the svg style parameter with the 3 variables in the string.

Since the assignment stated "When the shapes on the canvas are drawn, i.e.. Outputted as SVGs, then output methods must utilise the style object." the singleton object is created in the display method and thus the addStyle method in the singleton object is called in the display method, the addStyle method is not applied to a shape when it is created.

Shape objects when they are created contain a variable called "svgLine" which contains the svg code for the shape. Previously the display method would print the svg line of every shape on the canvas (off all shape objects created). Now when the display method displays the svg line of every shape on the canvas, it calls the addStyle method, which adds the styling to the svg line. This is how the output method implements svg styling.

Implementing the singleton pattern did not have a major impact on the re-design for assignments 2,3 and 4. It did require changing the display methods to now call the addStyle method using the singleton object. Since it is a design pattern, what worked in one assignment I just applied to the others.