## Components

AbstractShapeFactory.java  
I added a context placeholder and a get context function to get the current context. The context placeholder was used because the ShapeFactory parameter required a context. The getShapeFactory function will return Shape Factories based on the int receieved. It will then pass on the different arguments to style the circle and rectangles to be drawn.

ShapeFactory.java  
This class was changed a lot, adding on constructors to hold the arguments brought from the abstract shape factory. Previously the class took strings to decide what shape to draw, however for this version of the project, new methods were added to call upon the shapes to draw directly. getCircle will draw a circle and getRectangle will draw a rectangle. The paremeters for the methods were expanded to take on arguments for a stroke and fill color.

Shape.java  
This class was also expanded to hold additional arguments for stroke and fill color values. New methods were added so that these values can be called upon when drawing the shapes. getFill() returns the fill value and getStroke() returns the stroke value.

Circle.java  
This class was also modified to hold the stroke and fill values for its arguments. Instead of generating random colors, it now gets the stroke and fill values passed from the AbstractShapeFactory, calling it via the Shape class to draw and apply onto the drawn circles.

Rectangle.java  
This class was modified similar to the circle class. It now draws rectangles using the values acquired via the abstract shape factory, applying the respective stroke and fill color values.

MainActivity.java  
This class had new variables added to display the current style type. cStyle will hold the string for the current style, initialized to Black and Yellow, the first style. nStyle will hold the string for the next style, initialized to Black and Green. A new AbstractShapeFactory is instatiated instead of a ShapeFactory previously used in project 3. Then shapeFactory is created using the abstract shape factory. This allows all the previous methods to still function, such as the counter and the clearing methods. The shapeFactory is initialized to 1, meaning the app will start with the first style, Black and Yellow.

updateShapeFactory() method was created to rotate between the preset styles by using a switch to decide what styles is to be selected. The method increments the int value carried by the shapeFactory. It then updates the Shape count which updates the text in the textview, then returns the shapeFactory called. The method also updates cStyle and nStyle respectively.

A new button was added to call on the updateShapeFactory method to swtich between the styles.

The buttons for drawing circles and rectangles were updated slightly to use the getRectangle and getCircle methods.

## References

<http://www.tutorialspoint.com/design_pattern/abstract_factory_pattern.htm>

for review on abstract factory pattern. However, I think the example shown was vastly different than what was specified.

<https://youtu.be/tPFuVRbUTwA>

For a refresher on constructors, which is what I used to carry down the color values to apply to the shapes.

<http://stackoverflow.com/questions/9044769/how-to-draw-text-with-diferent-stroke-and-fill-colors?answertab=votes>

For styling the shapes with stroke and fill

## Approach

Initially I approached the project planning to create a style factory to adjust the stylings, but upon closer reading of the specifications, I changed my approach and used constructors to style the shapes. Getting the abstract shape factory to work with my previous code was easy, I just had to add another layer to call on the shape factory. After that I had to do a review about constructors to figure out how to pass on the color values to the shapes respectively. Then I used a switch statement to rotate between the stylings.

## UML Diagram

## C:\Users\cptns\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclipclip_image001.png