CSci 1133, Spring 2015
Lab Exercise 13

More on Inheritance

In this Lab exercise, you will extend the graphics classes you created in the previous Lab. If you did not complete the Workout problem from Lab 12, you should first complete it before starting these exercises.

Warm-up

1). **Adding and Removing Shapes**

The `Display` class created in Lab 12 has an instance variable named `elements` that was initialized as a null list. The `elements` list will be used to keep track of shapes that are currently being displayed. Add the following mutator methods to the `Display` class:

> `add(shape)` : Takes a shape object as an argument and adds it to the end of the `elements` list. After the shape has been added, it should be drawn on the display using the object's `draw` method.
>
> `remove(shape)` : Takes a shape object as an argument and removes it from the `elements` list. After the shape has been removed from the list, clear the display (using the appropriate Turtle method), then redraw each shape in the `elements` list. The shapes should be drawn in order of appearance in the list.

Be sure to test each of these new methods using Python in interactive mode. Add a few circles, then remove them.

Stretch

1). **Rectangle Class**

Add another geometric shape class to draw rectangles. The `rectangle` class should be derived from the `shape` base class. The constructor should have arguments for the x and y location, plus the width and height of the rectangle object. Also include a `draw` method that will draw a rectangle based on its x, y, width, height, filled and fillColor attributes (refer to the `circle` class).

Test your new class by using the `add` and `remove` methods in the Display object to display and remove a few rectangle objects.

2). **Random Shapes**

Now modify the `mouseEvent` handler function in the `Display` class to randomly construct and display either a circle or rectangle object. Use the pseudo-random number generator to determine a "50-50 probability" outcome and then decide to construct either a randomly sized/colored circle or a randomly sized/colored rectangle at the x,y mouse event coordinates based on that outcome. Be sure to use the `Display.add` method to add the object to the display instead of the `draw` object method!

Workout

1). **Selecting Objects**

Add a method to both the `circle` and `rectangle` classes named `isIN` that will take x,y coordinate values and return `True` if the point x,y is *within* the boundaries of the object.

[Hint: The x,y "location" of the object is where it was drawn. For a circle, this is the bottom-center point of the circle]

2). **Displaying and Removing Shapes**

Modify the `mouseEvent` handler function in the `Display` class to remove a shape if the mouse button is pressed while the pointer is "inside" a currently displayed shape. If it's not inside any displayed object, construct and display either a randomly sized/colored circle or a randomly sized/colored rectangle per the instructions in Stretch problem 2.

Test your program by creating and removing objects from the display using the mouse.

Challenge

Add a `Button` class that is a child of a rectangle and adds both a label (string) and a button handler (function reference). Either bind the handler function to the object using the constructor or a mutator method (or both!)

Now create a button-handler function to toggle a state variable (in the Display class) that will direct the construction of rectangles or circles. In other words, choose which type of shape to draw based on the state variable rather than a "flip of the coin".

Modify the `mouseEvent` handler in the `Display` class to recognize a button-press and call the bound handler function.

Finally, construct and add a Button to the display.