**CS 162 Week 4 Lab Inheritance: Geometric Shapes**

|  |
| --- |
| **GeometricShape** |
| **-color**: str #The color of the shape(default : green)  **-filled**: bool #Indicates whether the shape is filled with a color(default : True) |
| **+GeometricShape(color: str, filled: bool)** #Creates a GeometricShape with the specified color and filled values  **+get\_color():** str #Returns the color of the shape  **+set\_color(new\_color: str):** None #Changes the color to the new color  **+is\_filled():** bool #Returns whether the shape is filled or not  **+set\_filled(filled: bool):** None #Changes the filled property  **+\_\_str\_\_():** str #Returns a string representation of this shape |

|  |
| --- |
| **Circle** |
| -**radius**: float #The radius of the circle |
| **Circle(radius: float, color: str, filled: bool)** #Creates a Circle as a subclass of GeometricShape with specified radius, color and filled values  **get\_radius():** float #Returns the radius of the circle  **set\_radius(new\_radius: float):** None #Changes the radius to the new size  **get\_area():** float #Calculates and returns the area of the circle  **get\_circumference():** float #Calculates and returns the circumference of the circle  **get\_diameter():** float #Calculates and returns the diameter of the circle  **\_\_str\_\_():** str #Returns a string representation of the circle |

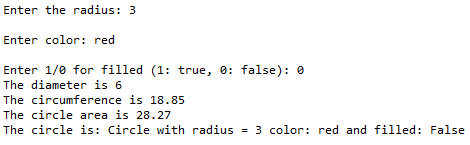
* Using the above UML, create the GeometricShape class
  + Implement each field and method as described
  + \_\_str\_\_() should return the following:

return "color: " + self.color + " and filled: " + str(self.filled)

* Using the above UML, create the Circle class as a subclass of GeometricShape
  + Implement each field and method as described
  + \_\_str\_\_() should return the string representation of the circle (use concatenation)
    - Hint: Use super().\_\_str\_\_() + …
    - For example:

Circle is : color = red filled = True radius = 3

* **Testing with Non-Defaults:** Write a test class called TestGeometrics. Create a unit test method that uses predetermined values for the radius of a circle, a color, and 1 or 0 to indicate whether the circle is filled (see sample data below). The unit test should create a Circle object with the radius and set the color and filled properties. The unit test should verify (hint: use self.assertEqual(expected, actual)) the data calculated by the classes under test, including the circle’s area, circumference, diameter, color and True or False (self.assertTrue(actual) or self.assertFalse(actual)) to indicate whether the circle is filled or not.
* **Testing with Defaults:** Write another unit test method in the TestGeometrics class that uses a predetermined value for the radius. This test should verify/assert that your default values in your super class work.
* Sample data for testing non-defaults:



* Sample data for testing defaults:

