Running Head: OBJECT COLLABORATION

Week 1 Application: Object Collaboration

Jered McClure

Walden University

Week 1 Application: Object Collaboration

 The Circle object defines the values of a given circle based on the radius value passed by the user. Radius can either be defined at object creation, or updated based on the setRadius() method. Values for the circle can be returned using the three getter methods: getRadius(), getDiameter(), and getArea(). Finally, all three attributes can be returned to the system console using the printCircle() method.

 When developing the circle object methods, the overall schema was fairly well defined based on the instructions given in the classroom. However, the data types and various intermediary methods were still required to match the overall goal of the Circle object. I chose to base all Circle attributes on the data type double. This allows for the greatest level of floating point numerical value. That being said, was another programmer to come along and wished to use the values based on a lesser data type, they would need to explicitly cast the Circle’s outputs to the new type.

 This drawback means that I could have implemented separate methods for returning each of the data types. If I do ever need to use this class again, I may, in fact, do so. However, for the scope of this particular project, I did not feel it was required.

 To test the circle object, the TestCircle class was created. TestCircle interfaces with two other classes, the Circle object and the Menu object. The Circle object is as mentioned above, the Menu object is a class that I have used in previous test cases. Essentially, it takes in user input, checks to see if it is what is expected, and outputs that input. It is used primarily in place of try-catch statements and in order to make the main method simpler, overall.

 The entire reason behind using the Menu class within TestCircle was to allow for user input via a while loop. TestCircle first outputs the default Circle values based on the default constructor. It then asks the user to input the radius of another circle. The Circle’s values are stored and returned via the setRadius and printCircle methods. A new circle is then instantiated and a value stored using the alternate constructor method. The new circle’s attributes are then output to the system console via the three getter methods. Next, the old circle and new circle are compared in the system console to ensure that they are indeed distinct objects. Finally, the user is given the option to loop back through the application or ending the application.

 By allowing for a loop which a user can input distinct values at each object method, the entire class can be tested without writing an inordinate amount of code. This does introduce other artifacts which can cause object error, however. If the Menu object is not inherently tested against false returns or throws an exception given invalid input, the testing of the Circle object could be faulty. As such, a programmer using the Menu object for the first time should test the object independently of the Circle object before using it to implement a test class. This does not mean that a programmer needs to test every java class, only those classes which are external to java’s internal library.

Appendix

UML Class Diagram



TestCircle Output

The circle's dimensions are: Radius - 1.0 Diameter - 2.0 Area - 3.141592653589793

What is the radius of another circle:

**25**

The circle's dimensions are: Radius - 25.0 Diameter - 50.0 Area - 1963.4954084936207

What is the radius of the new circle:

**50**

Using the getter methods directly the new circle's dimensions are: Radius - 50.0 Diameter - 100.0 Area - 7853.981633974483

Test the two circles independantly to make sure they are distinct.

aCircle - The circle's dimensions are: Radius - 25.0 Diameter - 50.0 Area - 1963.4954084936207

newCircle - The circle's dimensions are: Radius - 50.0 Diameter - 100.0 Area - 7853.981633974483

Would you like to find another Circle, please type Y or N:

**n**