Pseudocode for Circle2D.java

* Global variables:
  + x for holding the center x value
  + y for holding the center y value
* Circle2D() blank constructor method
  + Call itself and passing the default values for center (0,0) and radius 1
* Circle2D(double, double, double) constructor
  + Make the global variables the values that are passed in
* getX() method
  + Return the value of x
* getY() method
  + Return the value of y
* getRadius() method
  + Return the value of radius
* getArea() method
  + Calculate the area with formula: pi \* r^2
  + Return the area that was calculated
* getPerimeter() method
  + Calculate the perimeter or circumference with formula: 2 \* pi \* r
  + Return the perimeter that was calculated
* contains(double, double) method
  + Calculate the distance with the formula: sqrt( (x2-x1)^2 + (y2 – y1)^2)
    - Use the X and Y value passed into the method
  + Check if the distance is less than the radius
    - If it is, return true because the point is in the circle
    - If not, return false
* contains(Circle2D) method
  + Calculate the distance with the formula: sqrt( (x2-x1)^2 + (y2 – y1)^2)
    - Use the x and y calling the getX() and getY() method from the Circle2D object that was passed through
  + Check if the distance + the radius of the Circle2D object passed through is less than the radius
    - If it is, return true because the circle is within the other circle
    - If it’s not, return false
* overlaps(Circle2D) method
  + Calculate the distance with the formula: sqrt( (x2-x1)^2 + (y2 – y1)^2)
    - Use the x and y calling the getX() and getY() method from the Circle2D object that was passed through
  + Get the sum of the radii from the Circle2D object and the current object.
  + Check if the distance is less than the sum of the radii
    - If it is, return true because the circles are overlapping
    - If it’s not, return false. Touching will return false because it’s not overlapping

Test plan:

I will write unit tests to test every every method, called Circle2DTests.java. Test cases are just each test that is done. For all test cases, testCircle1 is a circle with center (0,0) and radius 1. testCircle2 is a circle with a center (0,0) and a radius of 5.

Test cases:

|  |  |  |  |
| --- | --- | --- | --- |
| **Cases** | **Input** | **Expected Result** | **Actual Result** |
| testProperties() unit test | Assert that the testCircle1 center X is 0, Y is 0, and radius is 1. Then assert that testCircle2 center X is 0, center Y is 0, radius is 5 | The test passes | The test passed |
| testContains() unit test | Assert that testCircle1 doesn’t contain testCircle2. Assert that testCircle2 contains testCircle1. Assert that testCircle1 doesn’t contain a new circle with center (1.5,1.5) and radius 1. Assert that circle2 doesn’t contain a circle with center (1,3) and radius 2. Assert that testCircle1 contains point (0,0) and not point (-2,0). Then assert that testCircle2 contains points (1,4) and (-4,2) | The test passes | The test passed |
| testOverlap() unit test | Assert that a new circle with center (1.5,1.5) radius 2 overlaps testCircle1. Assert that a circle with center (1,3) radius 4 overlaps testCircle2.  Assert that a circle with center (10,0) radius 5 doesn’t overlap testCircle2. Assert that a circle with center (-9,0) radius 5 overlaps testCircle2 | The test passes | The test passed |
| testArea() unit test | Assert that the area of testCircle2 is 78.54 with a precision of .001. Assert that the area of testCircle1 is PI with precision .001 | The test passes | The test passed |
| testPerimeter() unit test | Assert that the perimeter of testCircle1 is 6.283 with precision .001. Assert that the perimeter of testCircle2 is 31.416 with precision of .001 | The test passes | The test passed |

Test Screenshots:

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

UML:

A screenshot of a cell phone screen with text

Description automatically generated

Note: Professor, I know you told me to make them in UML Sculptor instead of using ObjectAid so that I could make it by hand for planning. This UML diagram was made in <https://draw.io> using their UML template, before I created the class to organize the requirements for the project. Please let me know if this is not okay and I’ll use UML Sculptor.

Lessons Learned:

Over the course of this project, I learned the importance of unit testing and why you need to make them. They are used to debug classes and minimize as many logic errors as possible. Unit testing should be done on every class that is used in a program to make sure it functions properly.

Checklist:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
|  | **Source java files** | **Y** |  |
|  | **Compressed files:** | **Y** |  |
|  | FirstInitialLastName\_Project7\_Moss.zip | **Y** |  |
|  |  | **Y** |  |
|  | FirstInitialLastName\_Project7\_doc.zip | **Y** |  |
|  | **Program compiles** | **Y** |  |
|  | **Program runs** | **Y** |  |
|  | **Checklist is completed and included in the Documentation** | **Y** |  |
|  | **Documentation file:** | **Y** |  |
|  | **Comprehensive Test Plan** | **Y** |  |
|  | **Screenshots based on Test Plan** | **Y** |  |
|  | **UML Diagram** | **Y** |  |
|  | **Algorithms/Pseudocode** | **Y** |  |
|  | **Flowchart** | **Y** |  |
|  | **Lessons Learned** | **Y** |  |