**CS 1302 – Sample Test 1b Handout**

*Tear this page off to use for reference* ● *Turn this page in with test.*

**All problems on the test use this information:**

You will write a *Circle* class (can abbreviate, *C*) with the following characteristics:

* Has a center that is denoted by *x* and *y* (double), a *radius* (double), and a color (string) that are all supplied in a constructor.
* A constructor that accepts all 4 properties.*.*
* A constructor that accepts a *radius* only. In this case the *x* and *y* should be set 0.0 each and the color set to “orange”, using best practices.
* A constructor that accepts *x* and *y* only. In this case the *radius* should be set 1.0 and the color set to “purple”, using best practices.
* A getter and setter for the *radius* property only. (you do not need to write a getter and setter for the other properties; however, you can assume they exist).

It should also have these methods:

**Note: on an actual test, you will write 4 or 5 methods like the ones below, additional one are provided for practice.**

* *area* – returns the area of the circle. The area of a circle is: $a=πr^{2}$
* *moveXBy* – accepts a value to move *x* by this value. For example, if *x=3.0* and the argument is 2.0, then *x* will become 5.0.
* *moveXY* – accepts two values and moves *x* and *y,* respectively by these values.
* *swap* – swaps *x* and *y.*
* *distanceFrom –* accepts another *Circle* and returns the distance from the center of this circle to the input circle. The distance between two points: $(x\_{1},y\_{1})$ and $(x\_{2},y\_{2})$ is: $d=\sqrt{(x\_{1}-x\_{2})^{2}+(y\_{1}-y\_{2})^{2}}$
* *merge* – accepts another *Circle* and does the following: (a) this *x* is updated so that it is the average of this *x* and the *x* for the input circle, (b) this *y* is updated so that it is the average of this *y* and the *y* for the input circle, (c) this *radius* is updated so that it is the average of this *radius* and the *radius* for the input circle.
* *numSameColor* – accepts an array of *Circles* and returns the number of circles in the input array that have the same color as this circle.
* *beginsWithCharacter* – accepts a character and returns *true* if the color of this circle’s first character is the same as the input character; and *false* otherwise.
* *toString* – returns a string that shows *x* and *y* with two decimal places, and the color in the following format:

“Color: *color*, *x=#.##, y=#.##*”

For example, if the color is “Orange”, x=12.341234, y=9.82323, then the returned string would be:

“Color: Orange, x=12.34, y=9.82”

 **

|  |  |  |
| --- | --- | --- |
| **CS 1302 – Sample Test 1b** | **Name:** |  |
|  |  | Print: FirstName LastName |

***Closed Book Test. No Notes Allowed. Write answers on the test. This is longer than an actual test would be.***

1. (~50 points) Write the *Circle* class described on the handout.
2. (~20 points) Write a single line of code that:

|  |  |
| --- | --- |
| 1. Creates a *Circle* using the first constructor. Call it c*1*.
 |  |

|  |  |
| --- | --- |
| 1. Creates a *Circle* using the second constructor. Call it *c2*.
 |  |

|  |  |
| --- | --- |
| 1. Changes the *radius* of c*1* to 5
 |  |

|  |  |
| --- | --- |
| 1. Retrieves the area of the *c2*, storing the result in a variable.
 |  |

|  |  |
| --- | --- |
| 1. Finds the distance between *c1* and *c2* storing the result in a variable.
 |  |

|  |  |
| --- | --- |
| 1. Suppose you have *Circles*: *c5* and *c6.* Define an array to hold these circles and put the circles in the array. You can do this in one line or several.
 |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| 1. Find the number of circles in the array from the previous step that have the same color as *c1*, storing the result in a variable..
 |  |

1. Suppose you have a class, *CircleUtilities.* This class has no instance variables, it just has methods that accept arrays of *Circles*.

**Note: on an actual test, you will write 2 or 3 methods like the ones below, additional ones are provided for practice.**

1. (~10 points) Write a method for this class, *averageRadius,* that accepts an array of *Circle* objectsand returns the average radius of the circles in the input array. (**The solution does not provide a solution for this method)**
2. (~10 points) Write a method for this class, *smallestCircle,* that accepts an array of *Circle* objectsand returns the *Circle* with the smallest area. (**The solution does not provide a solution for this method)**
3. (~10 points) Write a method for this class, *closestCircle,* that accepts an array of *Circle* objects and another *Circle* object (call this *Circle, c*). The method should return the *Circle* in the array that is closest to *c.* (**The solution does not provide a solution for this method)**
4. (~10 points) Write a method for this class, *circleColors,* that accepts an array of *Circle* objectsand returns and array of only the color of each circle in the input array. (**This solution does not provide a solution for this method)**
5. (~10 points) Write a method for this class, *changeRadii,* that accepts an array of *Circle* objectsand a radius. This method should change the radius of each circle in the array, whose radius is less than the input radius, to the value of the input radius. For example, if a circle in the array has a radius of 2.0, and the input radius is 5.0, the this circle will have its radius changed to 5.0. (**This solution does not provide a solution for this method)**