CSCI 2540 Assignment 2

100 points

Due date: Thursday, Sept. 15 by 11:59pm

For this assignment, you will be writing a class called **Point** and another class called **ExtendedCircle.** In addition, you need to write a class called **ExtendedCircleTest** class to test the Point class and the ExtendedCircle class.

You need to include all three class in the same package. Name your package as *assg2_yourPirateId*.

You are also required to use **Javadoc** comments at the beginning of your program, before each method, and for each constant.

(1) The Point class

The **Point** class represents a point in the format of (x, y).

The **Point** class should include the following variables and methods:

- It should include x and y coordinates as instance variables. Use *int* type for both variables.
- It should include two constructors, the default constructor and the two-parameter constructor. The default constructor will set both x and y coordinates to 0. The two-parameter constructor will include a given x and y coordinates as parameters.
- It should include get/set methods for both x and y coordinates.
- It should include *toString* method which returns a string in the format of "(x, y)".
- It should include *distance* method to calculate the distance between this point and another point. In other words, it will take one parameter of **Point** type and returns a *double* value. (You may use *Math.sqrt* method to calculate the square root).
- It should include the *equals* method with one parameter of **Object** type. It returns *boolean* value. Two points are equal if they have the same x and y coordinates.

(2) The ExtendedCircle class

The **ExtendedCircle** class represents a circle with a center point and a radius.

The **ExtendedCircle** class should include the following variables and methods:

- It should include two instance variables, the radius and the center point. Use *double* for radius and **Point** type for the center of the circle.
- It should include three constructors, the default constructor, the one-parameter constructor, and the two-parameter one.
  - The default constructor will set the radius to 1 and set the center to (0, 0).
  - The one-parameter constructor will have a parameter of *double* type. The parameter is the given radius. The center will be set to (0, 0).
  - The two-parameter constructor will have two parameters, one with *double* type and another one with **Point** type, which represent a given radius and a given center point.
- It should include get/set methods for both the radius and center variables.
- It should include *compArea* method that computes the area of this circle. This method does not have any parameter. It returns a *double* value. The formula to compute the area of a circle with radius of *r* is $PI*r^2$.
- It should include *compCircumference* method that computes the circumference of this circle. This method does not have any parameter. It returns a *double* value. The formula to compute the circumference of a circle with radius of *r* is *2*PI*r*.
- It should include the *toString* method which returns a string in the following format:

  Radius: the value of the radius
  Center: (x, y)

- It should include a method called *positionToCircle*. This method has one parameter of **Point** type. It returns an *int* value. It determines whether a given point is inside the circle (which returns -1), on the circle (which returns 0), or outside the circle (which returns 1. You can use the distance between the point and the center of the circle to determine.
- It should include a method called *shift*. This method has two parameters of *int* type. It returns a new **ExtendedCircle** object with the center shifted by certain value on x and y axis while the radius is the same. For example, if the parameters are 3 and 4, it means that the center is shifted by 3 on x-axis and 4 by y-axis. If the parameters are -3 and -4, it means that the center is shifted by -3 and -4 on x and y axis. The original circle should not be changed.
- It should include a method called *scale*. This method has one parameter of *double* type. It returns a new **ExtendedCircle** object with the radius scaled by a factor. For example, if the parameter is 2.0, it means the radius of the new circle is double of the original radius. If the parameter is 0.5, it means that radius of the new circle is half of the original radius. The original circle should not be changed.
- It should include a method called *overlap*. This method has one parameter of **ExtendedCircle** type and returns a *boolean* value. It tests whether this circle overlaps with a given circle. If the distance between the center point of the two circles is less

than the sum of the radius, it means the two circles overlap. Otherwise, they do not overlap.

- It should include the *equals* method. The parameter should be **Object** type and returns *boolean* value. Two circles are equal if they have the same radius and the same center point.

(3) The ExtendedCircleTest class

The **ExtendedCircleTest** class is to test the other two classes. It should include the *main* method. You need to create multiple objects of **Point** class and **ExtendedCircle** class. You need to test ALL the constructors and ALL the methods in both classes. You also need to test all the cases. For example, for the *equals* method, you need to have at least two test cases, one returns true and the other returns false. Similarly, for the *positionToCircle* method, you need to test at least three cases (inside the circle, on the circle, and outside the circle).

Do not ask user to enter anything from the keyboard for this program.

Print sufficient and proper messages to show what you are testing and what the result is.

Technical Notes

- **At the top of EVERY Java file you create, you must include a comment which states your name and what the program does.** Follow the standard "Javadoc" commenting style (/** ……. */) for formatting your comments.
- **YOU MUST PROVIDE COMMENTS for EVERY method that is included in your class.** Follow the standard "**Javadoc**" commenting style (/** ……. */) for formatting your comments and include @param for each parameter and @return if there is a return type that is not void. You should also include Javadoc comments before each constructor.
- You do not need to finish writing the entire class before you can start testing it.
- Programs that do not compile will receive an automatic grade of "F".

Submission instructions:

You need to submit your programs electronically on Canvas.

Please **use a named package for each of your assignment.** For example, for assignment 2, create a new package and **name your package as assg2_yourPirateId** (use lower case for your pirate id), such as assg2_smithj19. You also need to include a statement such as "package assg2_smithj19;" at the beginning of each of your .java file (it will be automatically generated in Eclipse if you create the class inside the given package). **Please follow this naming convention exactly for all future assignments. You will be deducted points for not doing so.**

When you submit your files to Canvas, please submit a zip file with your package folder inside the zip file. The package folder should include only .java files (make sure you include .java files, not .class files). The name of the folder should match with your package name. (You can use 7-zip software to zip files/folder). **Once your zip file is unzipped, it will generate a folder that matches your package name.**