Lab 08 – Classes, Objects, and Methods

Introduction

This lab introduces you to fundamental concepts of Object Oriented Programming (OOP), arguably the dominant programming paradigm in use today. In the paradigm, a program consists of component parts (objects) that are independent of each other and that interact in order to achieve a desired result. At a very basic level, an object oriented program consists of the following types of things.

Classes:	The template or blueprint from which objects are created.
Objects:	The instantiation of a class. Once a class (the template) has been defined, many instances of it can be created.
Methods:	The named procedures or functions defined for an object. A method accepts input arguments and contains statements that are executed when the method is invoked. Some methods return values.
Instance Variables:	Variables that are initialized when an object is created. Each instance of a class can have distinct values for these variables.

In this lab, you will practice working with these fundamentals. A skeleton of code has been written for you defining the basic class and method structure that you are to use. You will then add code to the existing skeleton methods so that the classes function as intended. You are then to use a testing program illustrate how the component parts of what you have created interact.

Lab Objectives

By the end of the lab, you should be able to create classes utilizing:

- access modifiers;
- instance variables;
- methods which return values and void methods (which do not return any value);
- methods calling other methods;
- accessor and mutator methods;
- an **`equals()`** method.

What to Submit

The files **`Circle.java`** and **`CircleTester.java`** should be submitted for grading.

Instructions

The file **`Circle.java`** contains a partial definition for a class representing a circle. Save it to your computer and study it to see what methods it contains. Then complete the ***Circle*** class as described below. Note that you won't be able to test your methods until you complete or partially complete the test client class called **`CircleTester.java`**.

1. You must include a comment stating your name, the date, program purpose, and containing a statement of academic honesty. When writing your source code, you must also abide by the Java formatting and naming conventions.

2. Complete the **`Circle`** class following the instructions detailed below:

   a. Declare three *private double* instance variables: **`x`**, **`y`** and **`radius`**. The instance variables **`x`** and **`y`** represent the coordinates of the center of the circle. Note that you should not declare any other instance variables other than these three.

   b. Fill in the code for method **`toString`**, which should **return** a "string" containing values for the center of the circle, **`x`** and **`y`**, and the radius of the circle. The returned string value should be formatted as below:

   ```
   center: (x,y)
   radius: r
   ```

   c. Fill in the code for the accessor (getter) methods: **`getX`**, **`getY`** and **`getRadius`** properly. Note that accessors are also called "getter" methods.

   d. Fill in the code for the mutator (setter) methods: **`setX`**, **`setY`** properly. Mutators are also called "setter" methods.

   e. Fill in the code for the mutator(setter) method: **`setRadius`** properly. Note that:

      i. If the new radius value passed as a parameter to the method is **not valid,** than the method should leave the original radius *unchanged*.

      ii. A radius is valid only when it is greater than or equal to 0.

   f. Fill in the code for the method **`area`** that returns the area of the circle, computed by

   PI * radius^2

   g. Use the value of PI provided by the constant **`Math.PI`**.

   h. Fill in the code of the method **`perimeter`** that returns the perimeter of the circle computed by

   `2 * PI * radius`

   i. Fill in the code of the method **`diameter`** that returns the diameter of the circle.

   j. Fill in the code of the method **`isUnitCircle`** that returns **`true`** if the radius of the circle is **`1`** and is centered at the origin, i.e., the coordinates of the circle's center are (0, 0). Otherwise, this method returns **`false`**.

3. The **`CircleTester.java`** file contains a driver program to try and test out your **`Circle`** class. You must include a comment stating your name, the date, program purpose, and containing a statement of academic honesty at the top of this class. Using the comments in the **`CircleTester.java`** file, finish the file so that it does the following:

   a. Display the center and radius of **`circle1`**.

   b. Set the radius of **`circle2`** to **`5.3`**.

   c. Display the center and radius of **`circle2`**.

   d. Display the diameter, area and perimeter of **`circle1`**.

   e. Display the diameter, area and perimeter of **`circle2`**.

   f. Display a message that indicates whether or not **`circle1`** is a unit circle.

g. Display a message that indicates whether or not `circle2` is a unit circle.

4. Compile and test your `Circle` and `CircleTester` files, and verify your code is working properly. Then, set breakpoints in your methods, and the first line of the `CircleTester` class, and observe the flow of control within an Object-Oriented program.
5. Add to your `Circle` class the following methods:

a. **`public boolean equals(Circle anotherCircle)`**

This method returns `true` when the radius and centers of both circles are the same; otherwise, it returns `false`. This method can be implemented in one line.

b. **`public boolean isConcentric(Circle anotherCircle)`**

The method `isConcentric` returns true when the circle executing the method has the same center as `anotherCircle` but two circles are not equal (i.e. their `radius` is different).

Note: we will use the aforementioned definition of `isConcentric` for this lab; however, `isConcentric` may have different definitions outside of this lab exercise.

c. **`public double distance(Circle anotherCircle)`**

This method returns the distance between the centers of the circle executing the method and `anotherCircle`. Let (*x, y*) and (x_a, y_a) be the centers of the circle executing the method and `anotherCircle` respectively, the distance between their centers is computed by

$$\sqrt{(x - x_a)^2 + (y - y_a)^2}$$

The `Math` class contains methods for computing both square roots and powers. Below are the definitions of the methods.

```
//Returns a double value containing the square root of a double number.
double sqrt(double number)

//Returns a double value of the first argument raised to the power of
the second argument.
double pow(double base, double exponent)
```

d. **`public boolean intersects(Circle anotherCircle)`**

The method `intersects` returns `true` when the circle executing the method and `anotherCircle` have an intersecting area (one or more points enclosed by both circles); otherwise, it returns `false`.

Two circles intersect (by this lab's definition) if the distance between the centers of the two circles is less than the sum of their radius. Your method must call the method `distance` to obtain the distance between the two circles.

6. After you've added these methods, add at least three different tests for each method to your **`CircleTester`** class (you must have at least 12 new tests added to this class: 3 tests for the **`equals`** method, 3 tests for the **`isConcentric`** method, 3 tests for the **`distance`** method, and 3 tests for the **`intersects`** method), and verify you have successfully written the above methods. Remember, your methods must work properly with any input we can provide, so test rigorously.

Submission and Grading

After you have completed and thoroughly tested your programs, submit **`Circle.java`** and **`CircleTester.java`** to our course webpage in order to receive credit for the lab (if you have any questions about how to submit this assignment, then ask your lab instructor for help at least 24 hours before this assignment is due). Always double check that your submission was successful on our course website.

The lab will be graded according to the following guidelines.

- A score between 0 and 100 will be assigned.
- If the source file(s) are not submitted before the specified deadline, if they do not compile, or if the submitting student has an unexcused absence for a single lab period devoted to the assignment, then a grade of 0 will be assigned. (Note: students who show up to their first lab period and show their lab TA that they finished and submitted the week's lab assignment may be excused from their second lab period for that week.)
- If the required comment for all labs describing the program and the academic honesty statement is not included at the top of each submitted file, then 10 points will be deducted. Note: this required comment can be found in Lab 02.
- If a (single) source file name is incorrect, then 10 points will be deducted.
- The program will be evaluated using both the **`CircleTester.java`** file and a separate testing file. Multiple instances of the Circle class will be created and their methods invoked.