CSCI 241 Assignment 9 Polygons

Due: ***Sunday, April 25, 2021 at 11:59 p.m.***
Note: due to the exam on April 19, you have 10 days to complete the entire assignment.

Objective:
Complete and test an instantiable Java class that keeps track of information for a polygon.

You will not get far without a working constructor! For 3 points, email your instructor the lines of code for your *default* constructor (lines copied as text into an email message) by 11:59 p.m. on Thursday, 4/22/2021.

Description:
Chapter 9 of your textbook introduces you to the idea writing your own instantiable classes. As you have learned, these are the types of classes which specify the design of an object.

1. Start your work by downloading the zip file of the BlueJ project named **`Assign9Shapes`** from D2L.
2. You will see a class already in your project named **`TestPolygon`.** This class holds a `main()` method which tests all the parts of the **`Polygon`** class (most of the code is currently commented out).
3. Add a new class to your project named **`Polygon`**. This is your *instantiable* class.

You SHOULD NOT make any changes to the **`TestPolygon`** class (except to uncomment each part as you are ready to test it). I will run this `main()` method to test your work done in the **`Polygon`** class, described here:

`Polygon` class

In your **`Polygon`** class, start by adding the following constants and variables:

Class constants
Using the appropriate naming convention for constants, create 2 class constants:

1. a `double` named `DEFAULT_SIDE_LENGTH` to hold the default length, which should hold 10.0.
2. an `int` named `DEFAULT_SIDE_COUNT` to hold the default number of sides, which should hold 4.

Class variables
We want to keep track of 2 kinds of information for our class: the number of **`Polygon`** objects created, and the number of times we change the length or number of sides of an already-created **`Polygon`**. Declare 2 private integer class variables to hold this information. Initialize both to zero (0). Name them `count` and `changes`.

Instance variables
Each polygon will be a *regular polygon,* i.e., all sides of the polygon have the same length. For each polygon, two ***private*** data fields are needed (please use the exact names given below):

1. `double length` holds the **`Polygon`**'s side length, in centimeters
2. `int sides` holds the **`Polygon`**'s number of sides, in centimeters

Class methods
Include 2 class methods which are "getters" (accessors) to return the values of the two class variables you created. Here are their prototypes:

1. `int getCount()`
2. `int getChanges()`

Constructors
This class needs 2 constructors. In addition to setting values into the instance variables, each constructor also needs code to increment the class variable named `count`.

1. A default (no parameter) version that sets all instance variables to the values specified in the corresponding constants

   For 3 points, copy your default constructor code into an email message to your instructor by 11:59 p.m. on Thursday, April 22.

2. An alternate version that takes 2 parameters, holding the side length and number of sides, respectively. The values of the parameters will be used to set the values of the corresponding instance variables. ***Make sure the number of sides is between 3 and 10, both inclusive.***

Instance methods

1. **"Getter" (Accessor) methods**
   Add a public "getter" method for each instance variable. Names *must* be:
   - `double getLength()`
   - `int getSides()`

2. **"Setter" (Mutator) methods**
   The class should have public "setter" methods for each of the instance variables. Along with changing the values in the instance variables to the values held in the parameters, the methods will also increment the `changes` class variable. The methods *must* be named:
   - `void setLength(double)`
   - `void setSides(int)`
     Note: this method should only allow sides between 3 and 10 (both inclusive). If the parameter value is lower than 3 or more than 10, it should print a descriptive error message and make no changes to the polygon.

3. **Methods to perform calculations and return calculated values**
   There are several basic calculations that can be performed on all Polygons. Here are the prototypes and descriptions for each of these *public* methods:

   - `double findPerimeter()`
     calculates the perimeter of the Polygon and returns it.

- `double findArea()`
  calculates the area of the Polygon and returns it. For an *n-sided* polygon, the area formula is (where N = # of sides, and S = side length):

$$area = \frac{N \times S^2}{4 \times \tan\frac{\pi}{N}}$$

  Note that the Math class contains a method to find the tangent of a number in radians.

4. **Methods to print and compare**
   Here are 2 more public methods to add. Neither is complicated, but each lets us see, or do, more interesting things with a given Polygon:

   - `String toString()`
     This method creates and returns a `String` that holds information about the polygon. **As with virtually all `toString()` methods, this method DOES NOT PRINT anything!** The information returned should include the polygon shape name, its number of sides and the side length. Here are some examples of what could be returned by this method:

```
square with 4 sides of length 10.00
pentagon with 5 sides of length 15.13
decagon with 10 sides of length 15.13
```

     I will leave it up to you to find the names of polygons with other numbers of sides… Please make sure that the side length is written with 2 digits after the decimal point. In case you wish to use the same formatting codes you use in a `System.out.printf()` statement to format a `String` object, you can use the `String.format()` method. It creates a `String` that has the formatting codes built into it. For example:

```
String formatted = String.format("%4.2f",3.192);
```

     would create the `String "3.19"`.

   - `boolean equals(Polygon)`
     This method checks to see if the current **`Polygon`** contains the same values as the one sent in the parameter. If so, it returns *true*, otherwise returns *false*. Remember, you should not use == to see if two doubles have the same value. Instead, see if the two values are *less than 0.01* apart from each other.

`TestPolygon` class

This class contains a `main()` method and other static methods. It will test all the methods you wrote in the `Polygon` class. By the time you are finished, all code should be uncommented to produce the expected output.

See the last page of this assignment description for output from my version. Try to match it as closely as possible.

Warning: Your **`Polygon`** class will be tested against both the `main()` method in **`TestPolygon`** and against a different `main()`, so make certain you have confidence in what you have written.

Submission Requirements

1. **Electronic:** Submit the completed implementation for the `Assign9Shapes` BlueJ project in your CS lab account..
2. **Canvas copy:** Submit your `Polygon.java` file to the associated assignment entry in Canvas. I do not need a copy of the `TestPolygon` class.

Grading Criteria

Your program ***must compile without errors*** to be accepted. The following criteria will be used to evaluate your program:

Appropriate and complete comments	6 pts
Early submission of default constructor code	3 pts
Adherence to style conventions (identifier names, indentation, etc.)	3 pts
Code organization/efficiency	6 pts
Correct program behavior (includes following specifications)	22 pts
Total points:	**40 pts**

```
Polygon 1:
length =   10.00
sides =  4
area =  100.00
perimeter =   40.00
square with 4 sides of length 10.00


Class variables:
1 polygons created
0 length/sides changes

Polygon 2:
area = 1759.71
perimeter =  151.23
decagon with 10 sides of length 15.12

Class variables:
2 polygons created
0 length/sides changes

Error:  sides must be between 3 and 10
area = 1759.71
perimeter =  151.23
decagon with 10 sides of length 15.12

Class variables:
2 polygons created
0 length/sides changes

After side decrement, poly2 = nonagon with 9 sides of length 15.12
After side increment, poly2 = decagon with 10 sides of length 15.12
Before checking equals(), print polygons
poly1: square with 4 sides of length 10.00
poly2: decagon with 10 sides of length 15.12
Polygons are NOT equal.
Step 15: pentagon with 5 sides of length 15.13
Step 17: decagon with 10 sides of length 15.13
length =   15.12
sides = 10
area = 1759.71
perimeter =  151.23
decagon with 10 sides of length 15.12

Before checking equals(), print polygons
poly1: decagon with 10 sides of length 15.13
poly2: decagon with 10 sides of length 15.12
Polygons are equal.
Class variables:
2 polygons created
5 length/sides changes

Class constants:
Default side length = 10.0
Default # of sides = 4
```