Homework 2

Before attempting this project, be sure you have completed all of the reading assignments, hands-on labs, discussions, and assignments to date.

Design a Java class named Polygon that contains:

- A private int data field named numSides that defines the number of sides of the polygon. The default value should be 4.
- A private double data field named sideLength that defines the length of each side. The default value should be 5.0.
- A private double data field named xCoord that defines the x-coordinate of the center of the polygon. The default value should be 0.0.
- A private double data field named yCoord that defines the y-coordinate of the center of the polygon. The default value should be 0.0.
- A private double data filed named perimeter that defines the perimeter of the polygon.
- A no argument constructor that creates a Polygon using the default number of sides, default side length, and default x- and y-coordinates.
- A constructor that creates a Polygon using a specified number of sides, side length, and x- and y-coordinates
- Getter methods for all data fields
- A getPerimeter() method that returns a double value representing the perimeter of the Polygon.
- A toString() method that displays the number of sides, side length, x-coordinate, and y-coordinates in String format

Be sure your code compiles.

Write a Java test program, named TestPolygon, to create 5 different polygons representing the 5 test cases you just created. When creating the five polygons, create one using the no argument constructor. For the remaining four, feel free to use any number of sides, side length and x-, and y-coordinates that are not equal to the default values and not equal to each other. For each of the five polygons, call all of the methods and display the results. For example for a Polygon with 3 sides, side length of 2.0 and x-coordinate and y-coordinates of 1.0, the following test data may result:

Output

```
toString(): (numsides=3, sideLength=2.0, xcoord=1.0,ycoord=1.0)
getNumSides(): 3
getSideLength(): 2.0
getXCoord(): 1.0
getYCoord(): 1.0
getPerimeter(): 6.0
```

Document your test cases in the form of table with columns indicating the input values, expected output, actual output and if the test case passed or failed. This table should contain 4 columns with appropriate labels and a row for each test case. An example template is shown below. Note that the

actual output should be the actual results you receive when running your program and applying the input for the test record.

Keep in mind, for five Polygons, you will have five different output results. Also, note there is no requirement to actually draw a Polygon.

Example test cases:

Input	Expected Output	Actual Output	Pass?
Constructor: numsides=3 sideLength=2.0 xcoord=1.0 ycoord=1.0	** Output ** toString(): (numsides=3, sideLength=2.0, xcoord=1.0,ycoord=1.0) getNumSides(): 3 getSideLength(): 2.0 getXCoord(): 1.0 getYCoord(): 1.0 getPerimeter(): 6.0	** Output ** toString(): (numsides=3, sideLength=2.0, xcoord=1.0,ycoord=1.0) getNumSides(): 3 getSideLength(): 2.0 getXCoord(): 1.0 getYCoord(): 1.0 getPerimeter(): 6.0	**Yes**
Test case 2 here			
Test case 3 here			
Test case 4 here			
Test case 5 here			

The google recommended Java style guide, provided as link in the week 2 content, should be used to format and document your code. Specifically, the following style guide attributes should be addressed:

- Header comments include filename, author, date and brief purpose of the program.
- In-line comments used to describe major functionality of the code.
- Meaningful variable names and prompts applied.
- Class names are written in UpperCamelCase.
- Variable names are written in lowerCamelCase.
- Constant names are in written in All Capitals.
- Braces use K&R style**.**

Submission requirements

Deliverables include all Java files (.java) and a single word (or PDF) document. The Java files should be named appropriately for your applications. The word (or PDF) document should include screen captures showing the successful compiling and running of each of the test cases. Each screen capture should be properly labeled clearly indicated what the screen capture represents. The test cases table should be included in your word or PDF document and properly labeled as well.

Submit your files to the Homework 2 assignment area no later than the due date listed in your LEO classroom. You should include your name and HW2 in your word (or PDF) file submitted (e.g. firstnamelastnamehw2.docx or firstnamelastnamehw2.pdf)

Grading Rubric:

In all programming assignments the following grading rubric will be used to determine your grade:

Attribute	**Exceeds**	**Meets**	**Does not meet**
Design (5 points)	(5 points) Exhibits proper use of parameters, and selection of data types all of the time. Employs correct and appropriate use of programming structures (loops, conditionals, classes etc.) all of the time. Efficient algorithms used all of the time.	(3-4 points) Exhibits proper use of parameters, and selection of data types most of the time. Employs correct and appropriate use of programming structures (loops, conditionals, classes etc.) most of the time. Efficient algorithms used most of the time.	(0-2 points) Rarely exhibits proper use of parameters, and selection of data types. Rarely employs correct and appropriate use of programming structures (loops, conditionals, classes etc.) Poorly structured and inefficient algorithms.
Functionality (10 points)	(9-10 points) Extra effort was apparent through the addition of significant and additional functionality beyond the scope of the assignment.	(7-8 points) Program fulfills most functionality. Most requirements were fulfilled. Screen captures provided demonstrating the successful compiling and running of the program.	(0-6 points) Program does not fulfill functionality. Few requirements were fulfilled.
Test cases (5 points)	(5 points) Test cases provide comprehensive	(3-4 points) Test cases provide coverage of most code paths.	(0-2 points) No or insufficient test cases

	coverage of all code paths. Discussion of run-time errors included.	Test cases results well documented providing pass/fail results for each test case.	Minimal supporting evidence provided to verify testing actually took place.
Java Style Guide (5 points)	(5 points) Code impeccably neat and well-organized. Extensive In-line comments providing additional insight into code design and functionality	(3-4 points) Header comments include filename, author, date and brief purpose of the program. In-line comments used to describe major functionality of the code. Meaningful variable names and prompts applied. Class names are written in UpperCamelCase. Variable names are written in lowerCamelCase. Constant names are in written in All Capitals. Braces use K&R style**.**	(0-2 points) Code rarely follows recommended Java style guide

Submission requirements

Your deliverables include all Java files (.java) and a single word (or PDF) document. The Java files should be named appropriately for your applications. Your word document should include screen shots showing the successful compiling and running of each application, and a detailed description of the test plan for your application. The test plan should include the input, expected output, actual output and if the test case passed or failed. Submit your files to the Homework 2 assignment area no later than the due date listed in the calendar. You should include your name and HW2 in your word (or PDF) file submitted (e.g. firstnamelastnamehw2.docx or firstnamelastnamehw2.pdf)