Project: PentagonalPyramid App

Deliverables

Your project files should be submitted to Web-CAT by the due date and time specified. You may submit your files to the skeleton code assignment until the project due date but should try to do this much earlier. The skeleton code assignment is ungraded, but it checks that your classes and methods are named correctly and that methods and parameters are correctly typed. The files you submit to skeleton code assignment may be incomplete in the sense that method bodies have at least a return statement if applicable or they may be essentially completed files. In order to avoid a late penalty for the project, you must submit your completed code files to Web-CAT no later than 11:59 PM on the due date for the completed code. If you are unable to submit via Web-CAT, you should e-mail your files in a zip file to your TA before the deadline.

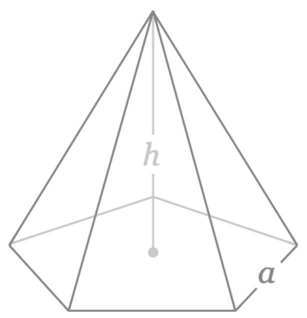
Files to submit to Web-CAT (both files must be submitted together):

- PentagonalPyramid.java
- PentagonalPyramidApp.java

Specifications

Overview: You will write a program this week that is composed of two classes: (1) one named PentagonalPyramid that defines PentagonalPyramid objects, and (2) the other, PentagonalPyramidApp, which has a main method that reads in data, creates a PentagonalPyramid object, and then prints the object.

A **PentagonalPyramid** has a pentagon base and five isosceles triangle faces, six vertices, and 10 edges as depicted below with base edge length *a* and height *h*. The formulas are provided to assist you in computing return values for the respective methods in the PentagonalPyramid class described in this project.
Ref: Do a "Google" search on "pentagonal pyramid surface area" for details and an interactive solver.

	Surface Area (*A*) Volume (*V*) Edge length (*a*) Height (*h*)	$A = \frac{5}{4}\tan(54°)\,a^2 + 5\frac{a}{2}\sqrt{h^2 + \left(\frac{a\tan(54°)}{2}\right)^2}$ $V = \frac{5}{12}\tan(54°)\,h\,a^2$

- **PentagonalPyramid.java**

  Requirements: Create a PentagonalPyramid class that stores the label, base edge (i.e., length of an edge in the pentagonal base), and height. The base edge and height must be greater than zero. The PentagonalPyramid class also includes methods to set and get each of these fields, as well as methods to calculate the surface area and volume of a PentagonalPyramid object, and a method to provide a String value of a PentagonalPyramid object (i.e., a class instance).

Design: The PentagonalPyramid class has fields, a constructor, and methods as outlined below.

(1) **Fields** (instance variables): label of type String, base edge of type double, and height of type double. Initialize the String to `""` and the double to 0 in their respective declarations. These instance variables should be private so that they are not directly accessible from outside of the PentagonalPyramid class, and these should be the only instance variables in the class.

(2) **Constructor**: Your PentagonalPyramid class must contain a public constructor that accepts three parameters (see types of above) representing the label, base edge, height. Instead of assigning the parameters directly to the fields, the respective set method for each field (described below) should be called. For example, instead of the statement `label = labelIn;` use the statement `setLabel(labelIn);` Below are examples of how the constructor could be used to create PentagonalPyramid objects. Note that although String and numeric literals are used for the actual parameters (or arguments) in these examples, variables of the required type could have been used instead of the literals.

```
PentagonalPyramid ex1 = new PentagonalPyramid("Ex 1", 1, 2);

PentagonalPyramid ex2 = new PentagonalPyramid(" Ex 2   ", 12.3, 25.5);

PentagonalPyramid ex3 = new PentagonalPyramid("Ex 3", 123.4, 900);
```

(3) **Methods**: Usually a class provides methods to access and modify each of its instance variables (known as get and set methods) along with any other required methods. The methods for PentagonalPyramid, which should each be public, are described below. See formulas in Code and Test below.

- `getLabel`: Accepts no parameters and returns a String representing the label field.
- `setLabel`: Takes a String parameter and returns a boolean. If the string parameter is not null, then the label field is set to the "trimmed" String and the method returns true. Otherwise, the method returns false and the label field is not set.
- `getBaseEdge`: Accepts no parameters and returns a double representing the base edge field.
- `setBaseEdge`: Accepts a double parameter and returns a boolean as follows. If the double is greater than zero, sets the base edge field to the double passed in and returns true. Otherwise, the method returns false and does not set the base edge field.
- `getHeight`: Accepts no parameters and returns a double representing the height field.
- `setHeight`: Accepts a double parameter and returns a boolean as follows. If the double is greater than zero, sets the height field to the double passed in and returns true. Otherwise, the method returns false and does not set the height field.
- `surfaceArea`: Accepts no parameters and returns the double value for the total surface area calculated using formula above and the values of the base edge and height fields. *See code and test below regarding the `Math.tan(x)` method.*
- `volume`: Accepts no parameters and returns the double value for the volume calculated using formula above and the values of the base edge and height fields.

- `toString`: Returns a String containing the information about the PentagonalPyramid object formatted as shown below, including decimal formatting `("#,##0.0######")` for the double values. Newline and tab escape sequences should be used to achieve the proper layout. In addition to the field values (or corresponding "get" methods), the following methods should be used to compute appropriate values in the `toString` method: `surfaceArea()` and `volume()`. Each line should have no trailing spaces (e.g., there should be no spaces before a newline (\n) character). The `toString` value for `ex1`, `ex2`, and `ex3` respectively are shown below (the blank lines are not part of the `toString` values).

```
PentagonalPyramid "Ex 1" with base edge = 1.0 and height = 2.0 units has:
   surface area = 7.008203 square units
   volume = 1.1469849 cubic units

PentagonalPyramid "Ex 2" with base edge = 12.3 and height = 25.5 units has:
   surface area = 1,086.4892066 square units
   volume = 2,212.4737204 cubic units

PentagonalPyramid "Ex 3" with base edge = 123.4 and height = 900.0 units has:
   surface area = 305,081.9691528 square units
   volume = 7,859,601.8538338 cubic units
```

Code and Test: Math.tan(x) expects x to be in radians rather than degrees. In the formula, 54 degrees can be converted to radians using Math.toRadians(54). The following combines the two methods: `Math.tan(Math.toRadians(54))`
As you implement your PentagonalPyramid class, you should compile it and then test it using interactions. For example, as soon you have implemented and successfully compiled the constructor, you should create instances of PentagonalPyramid in interactions (e.g., copy/paste the examples above). Remember that when you have an instance on the workbench, you can unfold it to see its values. You can also open a viewer canvas window and drag the instance from the Workbench tab to the canvas window. After you have implemented and compiled one or more methods, create a PentagonalPyramid object in interactions and invoke each of your methods on the object to make sure the methods are working as intended. You may find it useful to create a separate class with a main method that creates an instance of PentagonalPyramid then prints it out. This would be similar to the PentagonalPyramidApp class you will create below, except that in the PentagonalPyramidApp class you will read in the values and then create and print the object.

- **PentagonalPyramidApp.java**

Requirements: Create a PentagonalPyramidApp class with a main method that reads in values for label, base edge, and height. After the values have been read in, main creates a PentagonalPyramid object and then prints a new line and the object.

Design: The main method should prompt the user to enter the label, base edge, and height. After a value is read in for base edge, if the value is less than or equal to zero, an appropriate message (see examples below) should be printed followed by a *return* from main. Similarly, after a value is read in for height, if the value is less than or equal to zero, an appropriate message (see examples below) should be printed followed by a *return* from main. Assuming that base edge and height are positive, a PentagonalPyramid object should be created and printed.

Below is an example where the user has entered a non-positive value for base edge. Your program input/output should be **exactly** as follows.

Line # | Program input/output

```
Enter label, base edge, and height for a pentagonal pyramid.
   label: PentagonalPyramid with a bad base edge value
   base edge: -5
Error: base edge must be greater than 0.
```

Below is an example where the user has entered a non-positive value for height. Your program input/output should be **exactly** as follows.

Line # | Program input/output

```
Enter label, base edge, and height for a pentagonal pyramid.
   label: PentagonalPyramid with a bad height value
   base edge: 10.0
   height: 0
Error: height must be greater than 0.
```

Below is an example where the user has the values from ex1, the first example, above for label, base edge, and height. Your program input/output should be **exactly** as follows.

Line # | Program input/output

```
Enter label, base edge, and height for a pentagonal pyramid.
   label: Ex 1
   base edge: 1
   height: 2

PentagonalPyramid "Ex 1" with base edge = 1.0 and height = 2.0 units has:
   surface area = 7.008203 square units
   volume = 1.1469849 cubic units

```

Code: Your program should use the `nextLine` method of the `Scanner` class to read user input. Note that this method returns the input as a `String`. Whenever necessary, you can use the `Double.parseDouble` method to convert the input String to a double. For example: `Double.parseDouble(s1)` will return the double value represented by `String s1`. For the printed lines requesting input for label, base edge, and height, use a tab `"\t"` rather than three spaces. After you have created the object, it can be printed simply by using its variable name; i.e., when the object variable name is evaluated in the println statement, its `toString()` method is automatically called. Thus, printing the object reference variable `myObj` is equivalent to printing the return value of the `myObj.toString()` method call.

Test: You should test several sets of data to make sure that your program is working correctly. Although your main method may not use all the methods in PentagonalPyramid, you should ensure that all of your methods work according to the specification. You can use interactions in jGRASP or you can write another class and main method to exercise the methods. The viewer canvas should also be helpful, especially using the “Basic” viewer and the “toString” viewer for an PentagonalPyramid object. Web-CAT will test all of the methods specified above for PentagonalPyramid to determine your project grade.

General Notes

1. All input from the keyboard and all output to the screen should done in the main method. Only one Scanner object on System.in should be created and this should be done in the main method. All printing (i.e., using the System.out.print and System.out.println methods) should be in the main method. Hence, none of your methods in the PentagonalPyramid class should do any input/output (I/O).
2. When a method has a return value, you can ignore the return value if it is no interest in the current context. For example, when setBaseEdge(3.5) is invoked, it returns true to let the caller know the base edge field was set; whereas setBaseEdge(-3.5) will return false indicating the edge field was not set. If the caller knows that x is positive, then the return value of setBaseEdge(x) can safely be ignored since it can be assumed to be true.
3. Even though your main method (or other methods) may not be using the return value of a method such as setBaseEdge(x), you can ensure that the return type is correct by using interactions.