

Deliverables

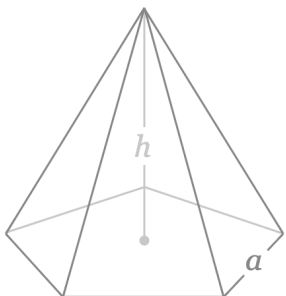
Your project files should be submitted to Web-CAT by the due date and time specified. You may submit your files to the skeleton code assignment until the project due date but should try to do this much earlier. The skeleton code assignment is ungraded, but it checks that your classes and methods are 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 (all three files must be submitted together):

- PentagonalPyramid.java
- PentagonalPyramidList.java
- PentagonalPyramidListApp.java

Specifications

Overview: You will write a program this week that is composed of three classes: the first class defines PentagonalPyramid objects, the second class defines PentagonalPyramidList objects, and the third, PentagonalPyramidListApp, reads in a file name entered by the user then reads the list name and PentagonalPyramid data from the file, creates PentagonalPyramid objects and stores them in an ArrayList, creates a PentagonalPyramidList object with the list name and ArrayList, prints the PentagonalPyramidList object, and then prints summary information about the PentagonalPyramidList object.

<p>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.</p> <p>Ref: Do a “Google” search on “pentagonal pyramid surface area” for details and an interactive solver.</p>		
	Surface Area (A)	$A = \frac{5}{4} \tan(54^\circ) a^2 + 5 \frac{a}{2} \sqrt{h^2 + \left(\frac{a \tan(54^\circ)}{2}\right)^2}$ $V = \frac{5}{12} \tan(54^\circ) h a^2$
	Volume (V)	
	Edge length (a)	
	Height (h)	

- **PentagonalPyramid.java** (assuming that you successfully created this class in Project 4, just copy the file to your new Project 5 folder and go on to PentagonalPyramidList.java on page 4. Otherwise, you will need to create PentagonalPyramid.java as part of this project.)

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 created in the previous project, except that in the `PentagonalPyramidApp` class you read in the values and then created and printed the object.

- **PentagonalPyramidList.java**

Requirements: Create an `PentagonalPyramidList` class that stores the name of the list and an `ArrayList` of `PentagonalPyramid` objects. It also includes methods that return the name of the list, number of `PentagonalPyramid` objects in the `PentagonalPyramidList`, total surface area, total volume, average surface area, and average volume for all `PentagonalPyramid` objects in the `PentagonalPyramidList`. The `toString` method returns a `String` containing the name of the list followed by each `PentagonalPyramid` in the `ArrayList`, and a `summaryInfo` method returns summary information about the list (see below).

Design: The `PentagonalPyramidList` class has two fields, a constructor, and methods as outlined below.

- (1) **Fields** (or instance variables): (1) a `String` representing the name of the list and (2) an `ArrayList` of `PentagonalPyramid` objects. These are the only fields (or instance variables) that this class should have, and both should be private.
- (2) **Constructor:** Your `PentagonalPyramidList` class must contain a constructor that accepts a parameter of type `String` representing the name of the list and a parameter of type `ArrayList<PentagonalPyramid>` representing the list of `PentagonalPyramid` objects. These parameters should be used to assign the fields described above (i.e., the instance variables).
- (3) **Methods:** The methods for `PentagonalPyramidList` are described below.
 - `getName`: Returns a `String` representing the name of the list.
 - `numberOfPentagonalPyramids`: Returns an `int` representing the number of `PentagonalPyramid` objects in the `PentagonalPyramidList`. If there are zero `PentagonalPyramid` objects in the list, zero should be returned.
 - `totalSurfaceArea`: Returns a `double` representing the total surface areas for all `PentagonalPyramid` objects in the list. If there are zero `PentagonalPyramid` objects in the list, zero should be returned.
 - `totalVolume`: Returns a `double` representing the total volumes for all `PentagonalPyramid` objects in the list. If there are zero `PentagonalPyramid` objects in the list, zero should be returned.
 - `averageSurfaceArea`: Returns a `double` representing the average surface area for all `PentagonalPyramid` objects in the list. If there are zero `PentagonalPyramid` objects in the list, zero should be returned.
 - `averageVolume`: Returns a `double` representing the average volume for all `PentagonalPyramid` objects in the list. If there are zero `PentagonalPyramid` objects in the list, zero should be returned.
 - `toString`: Returns a `String` (does not begin with `\n`) containing the name of the list followed by each `PentagonalPyramid` in the `ArrayList`. In the process of creating the return result, this `toString()` method should include a while loop that calls the `toString()` method for each `PentagonalPyramid` object in the list (adding a `\n` before and after each). Be sure to include appropriate newline escape sequences. For an example, [see lines 2 through 19](#) in the output below from `PentagonalPyramidListApp` for the `PentagonalPyramid_data_1.txt` input file. **[Note that the `toString` result should not**

include the summary items in lines 20 through 26 of the example. These lines represent the return value of the summaryInfo method below.]

- summaryInfo: Returns a String (does not begin with \n) containing the name of the list (which can change depending of the value read from the file) followed by various summary items: number of PentagonalPyramid objects, total surface area, total volume, average surface area, and average volume. Use "#,##0.0##" as the pattern to format the double values. For an example, see lines 20 through 26 in the output below from PentagonalPyramidListApp for the *PentagonalPyramid_data_1.txt* input file. The second example below shows the output from PentagonalPyramidListApp for the *PentagonalPyramid_data_0.txt* input file which contains a list name but no PentagonalPyramid data.

Code and Test: Remember to import java.util.ArrayList. The each of the last five methods above requires that you use a loop (i.e., a while loop) to retrieve each object in the ArrayList. As you implement your PentagonalPyramidList class, you can compile it and then test it using interactions. However, it may be easier to create a class with a simple main method that creates a PentagonalPyramidList object and calls its methods.

- **PentagonalPyramidListApp.java**

Requirements: Create an PentagonalPyramidListApp class with a main method that reads in the name of the data file entered by the user and then reads list name and PentagonalPyramid data from the file, creates PentagonalPyramid objects, stores them in a local ArrayList of PentagonalPyramid objects, creates a PentagonalPyramidList object with the name of the list and the ArrayList of PentagonalPyramid objects, and then prints the PentagonalPyramidList object followed summary information about the PentagonalPyramidList object. **All input and output for this project must be done in the main method.**

- **Design:** The main method should prompt the user to enter a file name, and then it should read in the data file. The first record (or line) in the file contains the name of the list. This is followed by the data for the PentagonalPyramid objects. After each set of PentagonalPyramid data is read in, a PentagonalPyramid object should be created and stored in the local ArrayList of PentagonalPyramid objects. After the file has been read in and the ArrayList has been populated, the main method should create a PentagonalPyramidList object with the name of the list and the ArrayList of PentagonalPyramid objects as parameters in the constructor. It should then print the PentagonalPyramidList object, then print the summary information about the PentagonalPyramidList (i.e., print the value returned by the summaryInfo method for the PentagonalPyramidList). The output from two runs of the main method in PentagonalPyramidListApp is shown below. The first is produced after reading in the *PentagonalPyramid_data_1.txt* file, and the second is produced after reading in the *PentagonalPyramid_data_0.txt* file. Your program output should be formatted exactly as shown.

Line #	Program output
1	----jGRASP exec: java PentagonalPyramidListApp
2	Enter file name: PentagonalPyramid_data_1.txt
3	PentagonalPyramid Test List
4	
5	PentagonalPyramid "Ex1" with base edge = 1.0 and height = 2.0 units has:
6	surface area = 7.008203 square units
7	volume = 1.1469849 cubic units
8	
9	PentagonalPyramid "Ex 2" with base edge = 12.3 and height = 25.5 units has:
10	surface area = 1,086.4892066 square units
11	volume = 2,212.4737204 cubic units
12	
13	PentagonalPyramid "Ex 3" with base edge = 123.4 and height = 900.0 units has:
14	surface area = 305,081.9691528 square units
15	volume = 7,859,601.8538338 cubic units
16	
17	
18	----- Summary for PentagonalPyramid Test List -----
19	Number of PentagonalPyramid: 3
20	Total Surface Area: 306,175.467
21	Total Volume: 7,861,815.475
22	Average Surface Area: 102,058.489
23	Average Volume: 2,620,605.158
24	
	----jGRASP: operation complete.

Line #	Program output
1	----jGRASP exec: java PentagonalPyramidListApp
2	Enter file name: PentagonalPyramid_data_0.txt
3	PentagonalPyramid Empty Test List
4	
5	
6	----- Summary for PentagonalPyramid Empty Test List -----
7	Number of PentagonalPyramid: 0
8	Total Surface Area: 0.0
9	Total Volume: 0.0
10	Average Surface Area: 0.0
11	Average Volume: 0.0
12	
	----jGRASP: operation complete.

Code: Remember to import `java.util.ArrayList`, `java.util.Scanner`, and `java.io.File`, and `java.io.FileNotFoundException` prior to the class declaration. Your main method declaration should indicate that main throws `FileNotFoundException`. After your program reads in the file name from the keyboard, it should read in the data file using a `Scanner` object that was created on a file using the file name entered by the user.

```
... = new Scanner(new File(fileName));
```

You can assume that the first line in the data file is the name of the list, and then each set of three lines contains the data from which a `PentagonalPyramid` object can be created. After the name of the list has been read and assigned to a local variable, a while loop should be used to read in the `PentagonalPyramid` data. The boolean expression for the while loop should be

(_____ .hasNext()) where the blank is the name of the Scanner you created on the file. Each iteration through the loop reads three lines. As each of the lines is read from the file, the respective local variables for the PentagonalPyramid data items (label, baseEdge, and height) should be assigned, after which the PentagonalPyramid object should be created and added to a local ArrayList of PentagonalPyramid objects. The next iteration of the loop should then read the next set of three lines then create the next PentagonalPyramid object and add it to the local ArrayList of PentagonalPyramid objects, and so on. After the file has been processed (i.e., when the loop terminates after the hasNext method returns false), name of the list and the ArrayList of PentagonalPyramid objects should be used to create an PentagonalPyramidList object. The list should be printed by printing a leading \n and the PentagonalPyramidList object. Finally, the summary information is printed by printing a leading \n and the value returned by the summaryInfo method invoked on the PentagonalPyramidList object.

Test: You should test your program minimally (1) by reading in the *PentagonalPyramid_data_1.txt* input file, which should produce the first output above, and (2) by reading in the *PentagonalPyramid_data_0.txt* input file, which should produce the second output above. Although your program may not use all of the methods in PentagonalPyramidList and PentagonalPyramid, you should ensure that all of your methods work according to the specification. You can either use interactions in jGRASP or you can write another class and main method to exercise the methods. Web-CAT will test all methods to determine your project grade.

General Notes

1. All input from the keyboard and all output to the screen should be 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. Be sure to download the test data files (*PentagonalPyramid_data_1.txt* and *PentagonalPyramid_data_0.txt*) and store them in same folder as your source files. It may be useful to examine the contents of the data files. Find the data files in the jGRASP Browse tab and then open each data file in jGRASP to see the items that your program will be reading from the file. Be sure to close the data files without changing them.