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.

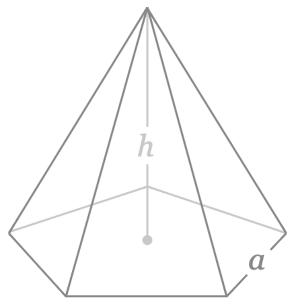
Files to submit to Web-CAT (all three files must be submitted together):

- PentagonalPyramid.java
- PentagonalPyramidList.java
- PentagonalPyramidListMenuApp.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, PentagonalPyramidListMenuApp, presents a menu to the user with eight options and implements these: (1) read input file (which creates a PentagonalPyramidList object), (2) print report, (3) print summary, (4) add a PentagonalPyramid object to the PentagonalPyramidList object, (5) delete a PentagonalPyramid object from the PentagonalPyramidList object, (6) find a PentagonalPyramid object in the PentagonalPyramidList object, (7) Edit a PentagonalPyramid in the PentagonalPyramidList object, and (8) quit the program. **[You should create a new "Project 6" folder and copy your Project 5 files** (PentagonalPyramid.java, PentagonalPyramidList.java, PentagonalPyramid_data_1.txt, and PentagonalPyramid_data_0.txt) **to it, rather than work in the same folder as Project 5 files.]**

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 (<u>assuming that you successfully created this class in Project 4 or 5, just copy the file to your new Project 6 folder and go on to PentagonalPyramidList.java on page 4. Otherwise, you will need to create PentagonalPyramid.java as part of this project</u>.)**

  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 <u>must be greater than zero</u>. 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 <u>these should be the only instance variables in the class</u>.

  (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 –** extended from Project 5 by **adding the last six methods below. (<u>Assuming that you successfully created this class in Project 5, just copy PentagonalPyramidList.java to your new Project 6 folder and then add the indicated methods. Otherwise, you will need to create all of PentagonalPyramidList.java as part of this project</u>.)**

  Requirements: Create a 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 <u>not</u> 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 16 in the output from PentagonalPyramidListApp for the *PentagonalPyramid_data_1.txt* input file. [Note that the toString result should **not** include the summary items in lines 18 through 24 of the example. These lines represent the return value of the summaryInfo method.]

- `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 18 through 24 in the output from PentagonalPyramidListApp for the *PentagonalPyramid_data_1.txt* input file. The second example shows the output from PentagonalPyramidListApp for the *PentagonalPyramid_data_0.txt* input file which contains a list name but no PentagonalPyramid data.

- **The following six methods are new in Project 6:**
  - `getList`: Returns the ArrayList of PentagonalPyramid objects (the second field above).
  - `readFile`: Takes a String parameter representing the file name, reads in the file, storing the list name and creating an ArrayList of PentagonalPyramid objects, uses the list name and the ArrayList to create a PentagonalPyramidList object, and then returns the PentagonalPyramidList object. See note #1 under Important Considerations for the PentagonalPyramidListMenuApp class (last page) to see how this method should be called.
  - `addPentagonalPyramid`: Returns nothing but takes three parameters (label, base edge, and height), creates a new PentagonalPyramid object, and adds it to the PentagonalPyramidList object.
  - `findPentagonalPyramid`: Takes a label of a PentagonalPyramid as the String parameter and returns the corresponding PentagonalPyramid object if found in the PentagonalPyramidList object; otherwise returns null. Case should be ignored when attempting to match the label.
  - `deletePentagonalPyramid`: Takes a String as a parameter that represents the label of the PentagonalPyramid and returns the PentagonalPyramid if it is found in the PentagonalPyramidList object and deleted; otherwise returns null. Case should be ignored when attempting to match the label; consider calling/using `findPentagonalPyramid` in this method.
  - `editPentagonalPyramid`: Takes three parameters (label, base edge, and height), uses the label to find the corresponding the PentagonalPyramid object. If found, sets the base edge and height to the values passed in as parameters, and returns true. If not found, returns false. This method should not change the label.

Code and Test: Remember to import java.util.ArrayList, java.util.Scanner, java.io.File, java.io.FileNotFoundException. These classes will be needed in the readFile method which will require a throws clause for FileNotFoundException. Some of the methods above require that you use a loop to go through the objects in the ArrayList. You may want to implement the class below in parallel with this one to facilitate testing. That is, after implementing one to the methods above, you can implement the corresponding "case" in the switch for menu described below in the PentagonalPyramidListMenuApp class.

- **PentagonalPyramidListMenuApp.java** (replaces PentagonalPyramidListApp class from Project 5)

  Requirements: Create a PentagonalPyramidListMenuApp class with a main method that presents the user with a menu with eight options, each of which is implemented to do the following: (1) read the input file and create a PentagonalPyramidList object, (2) print the PentagonalPyramidList object, (3) print the summary for the PentagonalPyramidList object, (4) add a PentagonalPyramid object to the PentagonalPyramidList object, (5) delete a PentagonalPyramid object from the PentagonalPyramidList object, (6) find a PentagonalPyramid object in the PentagonalPyramidList object, (7) edit a PentagonalPyramid object in the PentagonalPyramidList object, and (8) quit the program.

  Design: The main method should print a list of options with the action code and a short description followed by a line with just the action codes prompting the user to select an action. After the user enters an action code, the action is performed, including output if any. Then the line with just the action codes prompting the user to select an action is printed again to accept the next code. The first action a user would normally perform is 'R' to read in the file and create a PentagonalPyramidList object. However, the other action codes should work even if an input file has not been processed. The user may continue to perform actions until 'Q' is entered to quit (or end) the program. Note that your program should accept both uppercase and lowercase action codes.

  Below is output produced after printing the action codes with short descriptions, followed by the prompt with the action codes waiting for the user to make a selection.

Line #	Program output
1	`PentagonalPyramid List System Menu`
2	`R - Read File and Create PentagonalPyramid List`
3	`P - Print PentagonalPyramid List`
4	`S - Print Summary`
5	`A - Add PentagonalPyramid`
6	`D - Delete PentagonalPyramid`
7	`F - Find PentagonalPyramid`
8	`E - Edit PentagonalPyramid`
9	`Q - Quit`
10	`Enter Code [R, P, S, A, D, F, E, or Q]:`

  Below shows the screen after the user entered 'r' and then (when prompted) the file name. Notice the output from this action was "File read in and PentagonalPyramid List created". This is followed by the prompt with the action codes waiting for the user to make the next selection.

You should use the *PentagonalPyramid_data_1.txt* file from Project 5 to test your program.

```
Enter Code [R, P, S, A, D, F, E, or Q]: r
   File Name: PentagonalPyramid_data_1.txt
   File read in and PentagonalPyramid List created

Enter Code [R, P, S, A, D, F, E, or Q]:
```

The result of the user selecting ‘p’ to Print PentagonalPyramid List is shown below and next page.

```
Enter Code [R, P, S, A, D, F, E, or Q]: p
PentagonalPyramid Test List

PentagonalPyramid "Ex1" 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

Enter Code [R, P, S, A, D, F, E, or Q]:
```

The result of the user selecting ‘s’ to print the summary for the list is shown below.

```
Enter Code [R, P, S, A, D, F, E, or Q]: s

----- Summary for PentagonalPyramid Test List -----
Number of PentagonalPyramid: 3
Total Surface Area: 306,175.467
Total Volume: 7,861,815.475
Average Surface Area: 102,058.489
Average Volume: 2,620,605.158

Enter Code [R, P, S, A, D, F, E, or Q]:
```

The result of the user selecting ‘a’ to add a PentagonalPyramid object is shown below. Note that after ‘a’ was entered, the user was prompted for label, base edge, and height. Then after the PentagonalPyramid object is added to the PentagonalPyramid List, the message “*** PentagonalPyramid added ***” was printed. This is followed by the prompt for the next action. After you do an “add”, you should do a “print” or a “find” to confirm that the “add” was successful.

```
Enter Code [R, P, S, A, D, F, E, or Q]: a
   Label: Ex 4
   Base Edge: 10.5
   Height: 1000
   *** PentagonalPyramid added ***

Enter Code [R, P, S, A, D, F, E, or Q]:
```

Here is an example of the successful “delete” for a PentagonalPyramid object, followed by an attempt that was not successful (i.e., the PentagonalPyramid object was not found). You should do “p” to confirm the “d”.

```
Enter Code [R, P, S, A, D, F, E, or Q]: d
   Label: ex 2
   "Ex 2" deleted

Enter Code [R, P, S, A, D, F, E, or Q]: d
   Label: Fake
   "Fake" not found

Enter Code [R, P, S, A, D, F, E, or Q]:
```

Here is an example of the successful “find” for a PentagonalPyramid object, followed by an attempt that was not successful (i.e., the PentagonalPyramid object was not found).

```
Enter Code [R, P, S, A, D, F, E, or Q]: f
   Label: ex 3
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

Enter Code [R, P, S, A, D, F, E, or Q]: f
   Label: Another Fake
   "Another Fake" not found

Enter Code [R, P, S, A, D, F, E, or Q]:
```

Here is an example of the successful "edit" for a PentagonalPyramid object, followed by an attempt that was not successful (i.e., the PentagonalPyramid object was not found). In order to verify the edit, you should do a "find" for "medium" or you could do a "print" to print the whole list.

```
Enter Code [R, P, S, A, D, F, E, or Q]: e
   Label: Ex 3
   Base Edge: 250
   Height: 1000
   "Ex 3" successfully edited

Enter Code [R, P, S, A, D, F, E, or Q]: e
   Label: Ex 33
   Base Edge: 1.5
   Height: 4.5
   "Ex 33" not found

Enter Code [R, P, S, A, D, F, E, or Q]:
```

Finally, below is an example of entering an invalid code, followed by an example of entering a 'q' to quit the application with no message.

```
Enter Code [R, P, S, A, D, F, E, or Q]: k
   *** invalid code ***

Enter Code [R, P, S, A, D, F, E, or Q]: q

```

Code and Test:

Important considerations: This class should import java.util.Scanner, java.util.ArrayList, and java.io.FileNotFoundException. Carefully consider the following information as you develop this class.

1. At the beginning of your main method, you should declare and create an ArrayList of PentagonalPyramid objects and then declare and create a PentagonalPyramidList object using the list name and the ArrayList as the parameters in the constructor. This will be a PentagonalPyramidList object that contains no PentagonalPyramid objects. For example:

```
String _______ = "*** no list name assigned ***";
ArrayList<PentagonalPyramid> _________ = new  ArrayList<PentagonalPyramid>();
PentagonalPyramidList _________ = new PentagonalPyramidList(_________,_________);
```

   The 'R' option in the menu should invoke the readFile method on your PentagonalPyramidList object. This will return a new PentagonalPyramidList object based on the data read from the file, and this new PentagonalPyramidList object should replace (be assigned to) your original PentagonalPyramidList object variable in main. Since the readFile method throws FileNotFoundException, your main method needs to do this as well.

2. **Very Important**: **You should declare only one Scanner on System.in for your entire program, and this should be done in the main method.** That is, all input from the keyboard (System.in) must be done in your *main* method. Declaring more than one Scanner on System.in in your program will likely result in a very low score from Web-CAT.
3. For the menu, your switch statement expression should evaluate to a char, and each case should be a char; alternatively, your switch statement expression should evaluate to a String with a length of 1, and each case should be a String with a length of 1.

After printing the menu of actions with descriptions, you should have a *do-while* loop that prints the prompt with just the action codes followed by a switch statement that performs the indicated action. The *do-while* loop ends when the user enters 'q' to quit. You should strongly consider using a *for-each* loop as appropriate in the new methods that require you to search the list. You should be able to test your program by exercising each of the action codes. After you implement the "Print PentagonalPyramid List" option, you should be able to print the PentagonalPyramidList object after operations such as 'A' and 'D' to see if they worked. You may also want to run in debug mode with a breakpoint set at the switch statement so that you can step-into your methods if something is not working. In conjunction with running the debugger, you should also create a canvas and drag the items of interest (e.g., the Scanner on the file, your PentagonalPyramidList object, etc.) onto the canvas and save it. As you play or step through your program, you will be able to see the state of these objects change when the 'R', 'A', and 'D' options are selected.

Although your program may not use all of the methods in your PentagonalPyramid and PentagonalPyramidList classes, you should ensure that all of your methods work according to the specification. You can run your program in the canvas and then after the file has been read in, you can call methods on the PentagonalPyramidList object in interactions or you can write another class and main method to exercise the methods. Web-CAT will test all methods to determine your project grade.