Deliverables

Your project files should be submitted to Web-CAT by the due date and time specified. You may submit your files to the <u>skeleton code</u> 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 <u>completed code</u> 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.
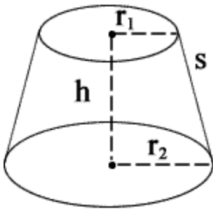
Files to submit to Web-CAT (all three files must be submitted together):
- ConicalFrustum.java
- ConicalFrustumList.java
- ConicalFrustumListMenuApp.java

Specifications

Overview: You will write a program this week that is composed of three classes: the first class defines ConicalFrustum objects, the second class defines ConicalFrustumList objects, and the third, ConicalFrustumListMenuApp, presents a menu to the user with eight options and implements these: (1) read input file (which creates a ConicalFrustumList object), (2) print report, (3) print summary, (4) add a ConicalFrustum object to the ConicalFrustumList object, (5) delete a ConicalFrustum object from the ConicalFrustumList object, (6) find a ConicalFrustum object in the ConicalFrustumList object, (7) Edit a ConicalFrustum in the ConicalFrustumList object, and (8) quit the program. **[You should create a new "Project 6" folder and copy your Project 5 files** (ConicalFrustum.java, ConicalFrustumList.java, conical_frustum_data_1.txt, and conical_frustum_data_0.txt) **to it, rather than work in the same folder as Project 5 files.]**

A *Conical Frustum* is a Frustum created by slicing the top off a cone (with the cut made parallel to the base), forming a lower base and an upper base that are circular and parallel.

r_1 radius of top	$V = \dfrac{\pi * h}{3}\ (r_1{}^2 + r_2{}^2 + (r_1 * r_2))$
r_2 radius of bottom	
h height	$s = \sqrt{(r_1 - r_2)^2 + h^2}$
s slant height	
S lateral surface area	$S = \pi * (r_1 + r_2) * s$
V volume	
A total surface area	$A = \pi * (r_1{}^2 + r_2{}^2 + (r_1 + r_2) * s)$

Source for figures and formulas: https://www.calculatorsoup.com/images/frustum001.gif

- **ConicalFrustum.java (<u>assuming that you successfully created this class in "Project: ConicalFrustum App" done previously, just copy the file to your new folder for this project and go on to ConicalFrustumList.java on page 4. Otherwise, you will need to create ConicalFrustum.java as part of this project.</u>)**

Requirements: Create a ConicalFrustum class that stores the label, radius of top, radius of bottom, and height where the radii and height are non-negative. The ConicalFrustum class also includes methods to set and get each of these fields, as well as methods to calculate the volume, slant height, lateral surface area, and total surface area of a ConicalFrustum object, and a method to provide a String value that describes a ConicalFrustum object.

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

(1) **Fields** (instance variables): label of type `String`, radius1 of type `double`, radius2 of type `double`, and height of type `double`. Initialize the `String` to `""` and the `double` variables to 0 in their respective declarations. These instance variables should be private so that they are not directly accessible from outside of the ConicalFrustum class, and <u>these should be the only instance variables (fields) in the class</u>.

(2) **Constructor**: Your ConicalFrustum class must contain a public constructor that accepts four parameters (see types of above) representing the label, radius1, radius2, and height. Instead of assigning the parameters directly to the fields, the respective set method for each field (described below) should be called since they are checking the validity of the parameter. For example, instead of using the statement `label = labelIn;` use the statement `setLabel(labelIn);` Below are examples of how the constructor could be used to create ConicalFrustum 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.

```
ConicalFrustum example1 = new ConicalFrustum("Small", 0.5, 0.75, 0.25);

ConicalFrustum example2 = new ConicalFrustum(" Medium ", 5.1, 10.2, 15.9);

ConicalFrustum example3 = new ConicalFrustum("Large", 98.32, 199.0, 250.0);
```

(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 ConicalFrustum, which should each be public, are described below. See the formulas in the figure above and the Code and Test section below for information on constructing these methods.
- `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 "trimmed" `String` is set to the label field and the method returns `true`. Otherwise, the method returns `false` and the label is not set.
- `getRadius1`: Accepts no parameters and returns a `double` representing the radius1 field.
- `setRadius1`: Takes a `double` parameter and returns a `boolean`. If the `double` parameter is non-negative, then the parameter is set to the radius1 field and the method returns `true`. Otherwise, the method returns `false` and the radius1 field is not set.

- o getRadius2: Accepts no parameters and returns a double representing the radius2 field.

- o setRadius2: Takes a double parameter and returns a boolean. If the double parameter is non-negative, then the parameter is set to the radius2 field and the method returns true. Otherwise, the method returns false and the radius2 field is not set.

- o getHeight: Accepts no parameters and returns a double representing the height field.

- o setHeight: Accepts a double parameter and returns a boolean as follows. If the double parameter is non-negative, then the parameter is set to the height field and the method returns true. Otherwise, the method returns false and the height field is not set.

- o volume: Accepts no parameters and returns the double value for the volume of the ConicalFrustum. [*Be sure to avoid integer division in your expression.*]

- o slantHeight: Accepts no parameters and returns the double value for the slant height of the ConicalFrustum.

- o lateralSurfaceArea: Accepts no parameters and returns the double value for the lateral surface area of the ConicalFrustum. Be sure to call your slantHeight method as appropriate.

- o totalSurfaceArea: Accepts no parameters and returns the double value for the total surface area of the ConicalFrustum. Be sure to call your slantHeight method as appropriate.

- o toString: Returns a String containing the information about the ConicalFrustum 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: volume(), slantHeight(), lateralSurfaceArea(), and totallSurfaceArea(). Each line should have no trailing spaces (e.g., there should be no spaces before a newline (\n) character). The toString value for example1, example2, and example3 respectively are shown below (the blank lines are not part of the toString values).

```
ConicalFrustum "Small" with radius1 = 0.5, radius2 = 0.75, and height = 0.25 has:
    volume = 0.311 cubic units
    slant height = 0.354 units
    lateral surface area = 1.388 units
    total surface area = 3.941 square units

ConicalFrustum "Medium" with radius1 = 5.1, radius2 = 10.2, and height = 15.9 has:
    volume = 3,031.546 cubic units
    slant height = 16.698 units
    lateral surface area = 802.608 units
    total surface area = 1,211.172 square units

ConicalFrustum "Large" with radius1 = 98.32, radius2 = 199.0, and height = 250.0 has:
    volume = 18,020,568.788 cubic units
    slant height = 269.512 units
    lateral surface area = 251,739.485 units
    total surface area = 406,518.914 square units
```

Code and Test: As you implement your ConicalFrustum 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 ConicalFrustum in interactions (e.g., copy/paste the examples on page 2). 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 ConicalFrustum 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 ConicalFrustum then prints it out. This would be similar to the ConicalFrustumApp class you will below, except that in the ConicalFrustumApp class you will read in the values and then create and print the object.

- **ConicalFrustumList.java** – extend from "Project : Conical Frustum List App" by **adding the last six methods below. (Assuming that you successfully created this class in the previous project, just copy ConicalFrustumList.java to your new folder for this project and then add the indicated methods. Otherwise, you will need to create all of ConicalFrustumList.java as part of this project.)**

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

  Design: The ConicalFrustumList class has two fields, a constructor, and methods as outlined below. These instance variables should be private, and they should be the only instance variables (fields) in the class.

  (1) Fields (or instance variables): (1) a String representing the name of the list and (2) an ArrayList of ConicalFrustum objects. These are the only fields (or instance variables) that this class should have.
  (2) Constructor: Your ConicalFrustumList class must contain a constructor that accepts a parameter of type String representing the name of the list and a parameter of type ArrayList< ConicalFrustum> representing the list of ConicalFrustum objects. These parameters should be used to assign the fields described above (i.e., the instance variables).

  (3) Methods: The methods for ConicalFrustumList are described below.
    o `getName`: Returns a String representing the name of the list.
    o `numberOfConicalFrustums`: Returns an int representing the number of ConicalFrustum objects in the ConicalFrustumList. If there are zero ConicalFrustum objects in the list, zero should be returned.

- o   `totalSurfaceArea`: Returns a double representing the total surface areas for all ConicalFrustum objects in the list. If there are zero ConicalFrustum objects in the list, zero should be returned.

- o   `totalVolume`: Returns a double representing the total volumes for all ConicalFrustum objects in the list. If there are zero ConicalFrustum objects in the list, zero should be returned.

- o   `averageSurfaceArea`: Returns a double representing the average for the total surface area for all ConicalFrustum objects in the list. If there are zero ConicalFrustum objects in the list, zero should be returned.

- o   `averageVolume`: Returns a double representing the average volume for all ConicalFrustum objects in the list. If there are zero ConicalFrustum objects in the list, zero should be returned.

- o   `toString`: Returns a String (does <u>not</u> begin with \n) containing the name of the list followed by each ConicalFrustum 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 ConicalFrustum object in the list (adding a \n <u>before and after</u> each). Be sure to include appropriate newline escape sequences. For an example, see <u>lines 3 through 21</u> in the output from ConicalFrustumListApp (previous project) for the *conical_frustum_data_1.txt* input file. [Note that t<u>he toString result should **not** include the summary items in lines 24 through 29 of the example. These lines represent the return value of the summaryInfo method below which will be called from </u>`main`.]

- o   `summaryInfo`: Returns a String (<u>does not begin or end with \n</u>) containing the name of the list (which can change depending of the value read from the file) followed by various summary items: number of ConicalFrustums, 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 <u>lines 24 through 29</u> in the output below from ConicalFrustumListApp for the *conical_frustum_data_1.txt* input file. The second example below shows the output from ConicalFrustumListApp for the *conical_frustum_data_0.txt* input file which contains a list name but no ConicalFrustum data.

<u>The following six methods are new in Project 6</u>:

- o   `getList`: Returns the ArrayList of ConicalFrustum objects (the second field above).

- o   `readFile`: Takes a String parameter representing the file name, reads in the file, storing the list name and creating an ArrayList of ConicalFrustum objects, uses the list name and the ArrayList to create a <u>new</u> ConicalFrustumList object, and then returns the ConicalFrustumList object. See note #2 under <u>Important Considerations</u> for the ConicalFrustumListMenuApp class to see how this method should be called.

- o   `addConicalFrustum`: Returns nothing but takes four parameters (label, radius1, radius2, and height), creates a new ConicalFrustum object, and adds it to the ConicalFrustumList object.

- o   `findConicalFrustum`: Takes a label of a ConicalFrustum as the String parameter and returns the corresponding ConicalFrustum object if found in the ConicalFrustumList object; otherwise returns null. <u>Letter case should be ignored when attempting to match the label (e.g., "Small Example" and "sMaLL EXAMPLE" should be a match)</u>.

- o `deleteConicalFrustum`: Takes a String as a parameter that represents the label of the ConicalFrustum. Returns the ConicalFrustum object if it is found in the ConicalFrustumList and deleted; otherwise returns `null`. <u>Letter case should be ignored when attempting to match the label; consider calling/using `findConicalFrustum` in this method</u>.
- o `editConicalFrustum`: Takes four parameters (label, radius1, radius2, and height), uses the label to find the corresponding the ConicalFrustum object. If found, sets the radius1, radius2, and height to the values passed in as parameters, and returns true. If not found, returns false.

Code and Test: You must import java.util.ArrayList, java.util.Scanner, java.io.File, java.io.FileNotFoundException as these classes will be needed in the readFile method, which will require a throws clause for FileNotFoundException. You may want to consider implementing the ConicalFrustumListMenuApp class below in parallel with the new methods in ConicalFrustumList. For example, after implementing the `readFile` method, you can implement the corresponding "case" in the switch for menu described below in the ConicalFrustumListMenuApp class that calls the method. This will allow you quickly test it by entering the 'R' and 'P' options to read in the file and print the list.

- **ConicalFrustumListMenuApp.java** (replaces ConicalFrustumListApp class from previous project)

  Requirements: Create a ConicalFrustumListMenuApp 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 ConicalFrustumList object, (2) print the ConicalFrustumList object, (3) print the summary for the ConicalFrustumList object, (4) add a ConicalFrustum object to the ConicalFrustumList object, (5) delete a ConicalFrustum object from the ConicalFrustumList object, (6) find a ConicalFrustum object in the ConicalFrustumList object, (7) Edit a ConicalFrustum object in the ConicalFrustumList 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 ConicalFrustumList 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	ConicalFrustum List System Menu
2	R – Read File and Create ConicalFrustum List
3	P – Print ConicalFrustum List
4	S – Print Summary
5	A – Add ConicalFrustum
6	D – Delete ConicalFrustum
7	F – Find ConicalFrustum
8	E – Edit ConicalFrustum
9	Q – Quit
10	Enter Code [R, P, S, A, D, F, E, or Q]:

Note that after an option code is entered, each sub-prompt has a leading tab (\t) (e.g., lines 2 and 3 below). The example below shows the screen after the user entered 'r' and when prompted, the file name. Notice the output from this action was "File read in and ConicalFrustum List created". This is followed by the prompt with the action codes waiting for the user to make the next selection. Use *conical_frustum_data_1.txt* from Project 5 to test your program.

Line #	Program output
1	Enter Code [R, P, S, A, D, F, E, or Q]: r
2	File name: conical_frustum_data_1.txt
3	File read in and ConicalFrustum List created
4	
5	Enter Code [R, P, S, A, D, F, E, or Q]:

The result of the user selecting 'p' to Print ConicalFrustum List is shown below and next page.

Line #	Program output
1	Enter Code [R, P, S, A, D, F, E, or Q]: p
2	
3	Conical Frustum Test List
4	
5	ConicalFrustum "Small" with radius1 = 0.5, radius2 = 0.75, and height = 0.25 has:
6	volume = 0.311 cubic units
7	slant height = 0.354 units
8	lateral surface area = 1.388 units
9	total surface area = 3.941 square units
10	
11	ConicalFrustum "Medium" with radius1 = 5.1, radius2 = 10.2, and height = 15.9 has:
12	volume = 3,031.546 cubic units
13	slant height = 16.698 units
14	lateral surface area = 802.608 units
15	total surface area = 1,211.172 square units
16	
17	ConicalFrustum "Large" with radius1 = 98.32, radius2 = 199.0, and height = 250.0 has:
18	volume = 18,020,568.788 cubic units
19	slant height = 269.512 units
20	lateral surface area = 251,739.485 units
21	total surface area = 406,518.914 square units
22	
23	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.

Line #	Program output
1	Enter Code [R, P, S, A, D, F, E, or Q]: s
2	
3	----- Summary for Conical Frustum Test List -----
4	Number of ConicalFrustum Objects: 3
5	Total Surface Area: 407,734.026
6	Total Volume: 18,023,600.645
7	Average Surface Area: 135,911.342
8	Average Volume: 6,007,866.882
9	
10	Enter Code [R, P, S, A, D, F, E, or Q]:

The result of the user selecting 'a' to add a ConicalFrustum object is shown below. Note that after 'a' was entered, the user was prompted for label, radius1, radius2, and height. Then after the ConicalFrustum object is added to the ConicalFrustum List, the message "*** ConicalFrustum added ***" was printed. Note that lines 2 - 6 below have leading tabs (\t). 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.

Line #	Program output
1	Enter Code [R, P, S, A, D, F, E, or Q]: a
2	Label: Test of Add
3	Radius1: 5.0
4	Radius2: 6.0
5	Height: 7.0
6	*** ConicalFrustum added ***
7	
8	Enter Code [R, P, S, A, D, F, E, or Q]:

Here is an example of the successful "delete" for a ConicalFrustum object, followed by an attempt that was not successful (i.e., the ConicalFrustum object was not found). Do "p" to confirm the "d".

Line #	Program output
1	Enter Code [R, P, S, A, D, F, E, or Q]: d
2	Label: Medium
3	"Medium" deleted
4	
5	Enter Code [R, P, S, A, D, F, E, or Q]: d
6	Label: not a real label
7	"not a real label" not found
8	
9	Enter Code [R, P, S, A, D, F, E, or Q]:

Here is an example of the successful "find" for a ConicalFrustum object, followed by an attempt that was <u>not</u> successful (i.e., the ConicalFrustum object was not found).

Line #	Program output
1	Enter Code [R, P, S, A, D, F, E, or Q]: f
2	Label: small
3	ConicalFrustum "Small" with radius1 = 0.5, radius2 = 0.75, and height = 0.25 has:
4	volume = 0.311 cubic units
5	slant height = 0.354 units
6	lateral surface area = 1.388 units
7	total surface area = 3.941 square units
8	
9	Enter Code [R, P, S, A, D, F, E, or Q]: f
10	Label: not a real label
11	"not a real label" not found
12	
13	Enter Code [R, P, S, A, D, F, E, or Q]:

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

Line #	Program output
1	Enter Code [R, P, S, A, D, F, E, or Q]: e
2	Label: small
3	Radius1: 1
4	Radius2: 2
5	Height: 3
6	"small" successfully edited
7	
8	Enter Code [R, P, S, A, D, F, E, or Q]: e
9	Label: even smaller
10	Radius1: .5
11	Radius2: .5
12	Height: .5
13	"even smaller" not found
14	
15	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.

Line #	Program output
1	Enter Code [R, P, S, A, D, F, E, or Q]: c
2	*** invalid code ***
3	
4	Enter Code [R, P, S, A, D, F, E, or Q]: q

Code and Test: <u>Important considerations</u>

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

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

    ```
    String _____ = "*** no list name assigned ***";
    ArrayList<ConicalFrustum> _____ = new ArrayList<ConicalFrustum>();
    ConicalFrustumList _____ = new ConicalFrustumList(_____,_____);
    ```

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

3. **<u>Very Important</u>**: **<u>You should declare only one Scanner on System.in for your entire program, and this should be done in the main method.</u>** 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.

4. 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.

5. 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 ConicalFrustum List" option, you should be able to print the ConicalFrustumList 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 drag the items of interest (e.g., the Scanner on the file, your ConicalFrustumList object, etc.) onto the canvas and save it. As you play or step through your program, you'll be able to see the state of these objects change when the 'R', 'A', and 'D' options are selected.

    a. For option P, when you print the ConicalFrustumList object (e.g., cList) be sure to append a leading "\n" in println:

```
System.out.println("\n" + cList);
```

b. For option S, when you print the summary for ConicalFrustumList object (e.g., cList) be sure to append a leading and trailing "\n" in the .println:

```
System.out.println("\n" + cList.summaryInfo() + "\n");
```

Although your program may not use all of the methods in your ConicalFrustum and ConicalFrustumList 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 ConicalFrustumList 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.