Project: Conical Frustum List 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.

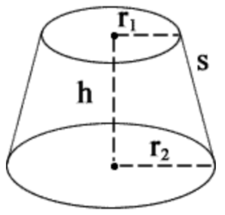
Files to submit to Web-CAT (all three files must be submitted together):

- ConicalFrustum.java
- ConicalFrustumList.java
- ConicalFrustumListApp.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, ConicalFrustumListApp, reads in a file name entered by the user then reads the list name and ConicalFrustum data from the file, creates ConicalFrustum objects and stores them in an ArrayList, creates a ConicalFrustumList object with the list name and ArrayList, prints the ConicalFrustumList object, and then prints summary information about the ConicalFrustumList object.

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

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

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

  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 these should be the only instance variables (fields) in the class.

(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.

- o `getLabel`: Accepts no parameters and returns a `String` representing the label field.
- o `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.
- o `getRadius1`: Accepts no parameters and returns a `double` representing the radius1 field.
- o `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.

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

  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.
  - `getName`: Returns a String representing the name of the list.
  - `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.
  - `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.
  - `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.
  - `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.

- `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.
- `toString`: Returns a String (does not 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 before and after each). Be sure to include appropriate newline escape sequences. For an example, see lines 3 through 21 in the output below from ConicalFrustumListApp for the *ConicalFrustum_data_1.txt* input file. [Note that the 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 `main`.]
- `summaryInfo`: Returns a String (does not begin or end 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 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 lines 24 through 29 in the output below from ConicalFrustumListApp for the *ConicalFrustum_data_1.txt* input file. The second example below shows the output from ConicalFrustumListApp for the *ConicalFrustum_data_0.txt* input file which contains a list name but no ConicalFrustum data.

Code and Test: Remember to import java.util.ArrayList. The `totalSurfaceArea` and `totalVolume` methods each requires that you use a **while loop** to retrieve each object in the ArrayList. As you implement your ConicalFrustumList class, you can compile it and then test it using interactions. Alternatively, (1) you can create a class with a simple main method that creates a ConicalFrustumList object and calls its methods; or (2) you can create the ConicalFrustumListApp described next.

- **ConicalFrustumListApp.java**

  Requirements: Create a ConicalFrustumListApp class with a `main` method that reads in the name of the data file entered by the user and then reads list name and ConicalFrustum data from the file, creates ConicalFrustum objects, stores them in a local ArrayList of ConicalFrustum objects, creates a ConicalFrustumList object with the name of the list and the ArrayList of ConicalFrustum objects, and then prints the ConicalFrustumList object followed summary information about the ConicalFrustumList object. **All input from the keyboard (System.in) and output to the screen (System.out) 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 read in the file name with a `Scanner` on `System.in`. A second `Scanner` should be created with the file name and then be used to read in the data file (see the **Code** section below for details). The first record (or line) in the file contains the name of the list. This is followed by the data for the ConicalFrustum objects. After each set of ConicalFrustum data is read in, a ConicalFrustum object should be created and stored in the local ArrayList of ConicalFrustum objects. After the file has been read in and the ArrayList has been populated, the main method should create a

ConicalFrustumList object with the name of the list and the ArrayList of ConicalFrustum objects as parameters in the constructor. It should then print the ConicalFrustumList object, then print the summary information about the ConicalFrustumList (i.e., print the value returned by the `summaryInfo` method for the ConicalFrustumList). The output from two runs of the main method in ConicalFrustumListApp is shown below. The first is produced after reading in the *ConicalFrustum_data_1.txt* file, and the second is produced after reading in the *ConicalFrustum_data_0.txt* file. Your program output should be formatted exactly as shown.

Line # Program output

```
----jGRASP exec: java ConicalFrustumListApp
Enter file name: conical_frustum_data_1.txt

Conical Frustum Test List

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


----- Summary for Conical Frustum Test List -----
Number of ConicalFrustum Objects: 3
Total Surface Area: 407,734.026
Total Volume: 18,023,600.645
Average Surface Area: 135,911.342
Average Volume: 6,007,866.882

 ----jGRASP: operation complete.
```

Line # Program output

```
----jGRASP exec: java ConicalFrustumListApp
Enter file name: conical_frustum_data_0.txt

Conical Frustum Empty Test List


----- Summary for Conical Frustum Empty Test List -----
Number of ConicalFrustum Objects: 0
Total Surface Area: 0.0
Total Volume: 0.0
Average Surface Area: 0.0
Average Volume: 0.0

 ----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 your `main` method `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 <u>four</u> lines contains the data from which a ConicalFrustum 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 ConicalFrustum 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 <u>four</u> lines (see note 3 below). As each of the lines is read from the file, the respective local variables for the ConicalFrustum data items (label, radius, and cylinder height) should be assigned, after which the ConicalFrustum object should be created and added to a local ArrayList of ConicalFrustum objects. The next iteration of the loop should then read the next set of three lines then create the next ConicalFrustum object and add it to the local ArrayList of ConicalFrustum 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 ConicalFrustum objects should be used to create a ConicalFrustumList object. The list should be printed by printing a leading \n and the ConicalFrustumList object. Finally, the summary information is printed by printing a leading \n and the value returned by the summaryInfo method invoked on the ConicalFrustumList object.

Test: You should test your program minimally (1) by reading in the *ConicalFrustum_data_1.txt* input file, which should produce the first output above, and (2) by reading in the *ConicalFrustum_data_0.txt* input file, which should produce the second output above. Although your program may not use all of the methods in ConicalFrustumList and ConicalFrustum, you should ensure that all of your methods work according to the specification. You can either user 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.

<u>General Notes</u>

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 ConicalFrustum class should do any input/output (I/O).

2. *ConicalFrustum_data_1.txt* and *ConicalFrustum_data_0.txt* – Be sure to download the data files and store them in same folder as your source files. You should 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.

3. Reading the data from the input file – Since each of the data items for label, radius1, radius2, and height are on separate lines, within the while loop, each line should be read with a .nextLine() on the file Scanner and then parsed into a double when appropriate. Below is an example where label, radius1, radius2, and height are local variables and the blank should be filled in with the variable for the Scanner for the input file.

```
label = _______.nextLine();
radius1 = Double.parseDouble(_______.nextLine());
radius2 = Double.parseDouble(_______.nextLine());
height = Double.parseDouble(_______.nextLine());
```