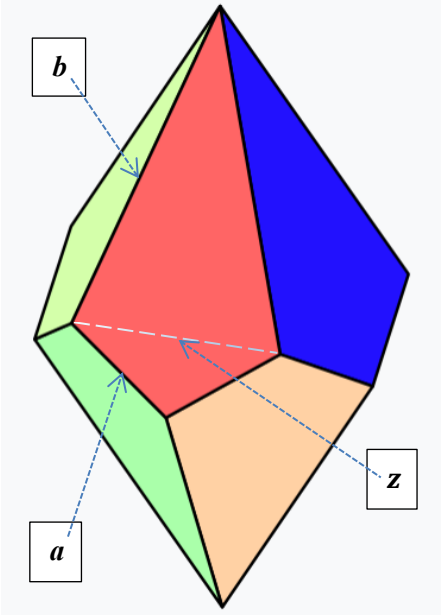
Deliverables

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):

- Trapezohedron.java
- TrapezohedronList.java
- TrapezohedronListApp.java

A **pentagonal trapezohedron** (or a pentagonal deltohedron) is a polyhedron composed of 10 kites as faces (with side lengths ***a*** and ***b***), 20 edges, and 12 vertices. The formulas are provided to assist you in computing return values for the respective methods in the Trapezohedron class described in this project. *(Sources: https://en.wikipedia.org/wiki/Pentagonal_trapezohedron#10-sided_dice https://rechneronline.de/pi/trapezohedron.php) To use calculator, see Test paragraph on page 5.*



Formulas for edge length antiprism (z), long edge length (b), surface area (*A*), and volume (*V*) are shown below where a is the short edge length, which will be read in.

$$z = a/((\sqrt{5}-1)/2)$$

$$b = ((\sqrt{5}+1)/2) * z$$

$$A = \sqrt{\frac{25}{2.0} * (5+\sqrt{5}) * z^2}$$

$$V = \frac{5.0}{12} * (3+\sqrt{5}) * z^3$$

Specifications

Overview: You will write a program this week that is composed of three classes: the first class defines Trapezohedron objects, the second class defines TrapezohedronList objects, and the third, TrapezohedronListApp, reads in a file name entered by the user then reads the list name and Trapezohedron data from the file, creates Trapezohedron objects and stores them in an ArrayList of

Trapezohedron objects, creates an TrapezohedronList object with the list name and ArrayList, prints the TrapezohedronList object, and then prints summary information about the TrapezohedronList object.

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

  Requirements: Create an Trapezohedron class that stores the label, color, and short edge length, which must be non-negative. The Trapezohedron class also includes methods to set and get each of these three fields, as well as methods to calculate the edge length antiprism, the long edge length, surface area, and volume of the Trapezohedron object, and a method to provide a String value of an Trapezohedron object (i.e., a class instance).

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

  (1) Fields (instance variables): label of type String, color of type String, and short edge of type double. Initialize the Strings to "" and the double to zero in their respective declarations. These instance variables should be private so that they are not directly accessible from outside of the Trapezohedron class, and these should be the only instance variables in the class.

  (2) Constructor: Your Trapezohedron class must contain a public constructor that accepts three parameters (see types of above) representing the label, color, and short edge. 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 Trapezohedron 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.

```
Trapezohedron ex1 = new Trapezohedron ("Ex 1", "red", 5.0);

Trapezohedron ex2 = new Trapezohedron (" Ex 2   ", "blue", 10.4);

Trapezohedron ex3 = new Trapezohedron ("Ex 3", "red, blue, tan", 24.5);
```

  (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 Trapezohedron, 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.
  - `getColor`: Accepts no parameters and returns a String representing the color field.

- `setColor`: Takes a String parameter and returns a boolean. If the string parameter is not null, then the "trimmed" String is set to the color field and the method returns true. Otherwise, the method returns false and the label is not set.
- `getShortEdge`: Accepts no parameters and returns a double representing the short edge field.
- `setShortEdge`: Accepts a double parameter and returns a boolean as follows. If the short edge is greater than zero, sets the short edge field to the double passed in and returns true. Otherwise, the method returns false and the short edge is not set.
- `edgeLengthAntiprism`: Accepts no parameters and returns the double value for the edge length antiprism calculated using the value for short edge (*a*) in the formula above.
- `longEdge`: Accepts no parameters and returns the double value for the long edge calculated using the formula above.
- `surfaceArea`: Accepts no parameters and returns the double value for the surface area calculated using the formula above.
- `volume`: Accepts no parameters and returns the double value for the volume calculated using the using the formula above.
- `toString`: Returns a String containing the information about the Trapezohedron object formatted as shown below, including decimal formatting (`"#,##0.0###"`) for the double values. The newline (\n) and tab (\t) escape sequences should be used to achieve the proper layout for the indented lines (use \t rather than three spaces for the indentation). In addition to the field values (or corresponding "get" methods), the following methods should be used to compute appropriate values in the toString method: edgeLengthAntiprism(), longEdge(), 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).

```
Trapezohedron "Ex 1" is "red" with 20 edges and 12 vertices.
   edge length antiprism = 8.0902 units
   short edge = 5.0 units
   long edge = 13.0902 units
   surface area = 622.4746 square units
   volume = 1,155.226 cubic units

Trapezohedron "Ex 2" is "blue" with 20 edges and 12 vertices.
   edge length antiprism = 16.8276 units
   short edge = 10.4 units
   long edge = 27.2276 units
   surface area = 2,693.074 square units
   volume = 10,395.7774 cubic units

Trapezohedron "Ex 3" is "red, blue, tan" with 20 edges and 12 vertices.
   edge length antiprism = 39.6418 units
   short edge = 24.5 units
   long edge = 64.1418 units
   surface area = 14,945.6145 square units
   volume = 135,911.1879 cubic units
```

Code and Test: As you implement your Trapezohedron 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 Trapezohedron in interactions (e.g., copy/paste the examples above 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 an Trapezohedron 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 Trapezohedron then prints it out.

- **TrapezohedronList.java**

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

Design: The TrapezohedronList 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 Trapezohedron objects. These are the only fields (or instance variables) that this class should have, and both should be private.

(2) Constructor: Your TrapezohedronList class must contain a constructor that accepts a parameter of type String representing the name of the list and a parameter of type ArrayList<Trapezohedron> representing the list of Trapezohedron objects. These parameters should be used to assign the fields described above (i.e., the instance variables).

(3) Methods: The methods for TrapezohedronList are described below.

- `getName`: Returns a String representing the name of the list.
- `numberOfTrapezohedrons`: Returns an int representing the number of Trapezohedron objects in the TrapezohedronList. If there are zero Trapezohedron objects in the list, zero should be returned.
- `totalSurfaceArea`: Returns a double representing the total surface area for all Trapezohedron objects in the list. If there are zero Trapezohedron objects in the list, zero should be returned.
- `totalVolume`: Returns a double representing the total volume for all Trapezohedron objects in the list. If there are zero Trapezohedron objects in the list, zero should be returned.

- `averageSurfaceArea`: Returns a double representing the average surface area for all Trapezohedron objects in the list. If there are zero Trapezohedron objects in the list, zero should be returned.
- `averageVolume`: Returns a double representing the average volume for all Trapezohedron objects in the list. If there are zero Trapezohedron 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 Trapezohedron 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 Trapezohedron 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 25 in the output below from TrapezohedronListApp for the *Trapezohedron_data_1.txt* input file. [Note that the toString result should **not** include the summary items in lines 27 through 33 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 on the value read from the file) followed by various summary items: number of Trapezohedron 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 27 through 33 in the output below from TrapezohedronListApp for the *Trapezohedron_data_1.txt* input file. The second example below shows the output from TrapezohedronListApp for the *Trapezohedron_data_0.txt* input file which contains a list name but no Trapezohedron data.

Code and Test: Remember to import java.util.ArrayList. Each of the four methods above that finds a total or average requires that you use a loop (i.e., a while loop) to retrieve each object in the ArrayList. As you implement your TrapezohedronList 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 an TrapezohedronList object and calls its methods.

- **TrapezohedronListApp.java**

  Requirements: Create an TrapezohedronListApp class with a main method that (1) reads in the name of the data file entered by the user and (2) reads list name and Trapezohedron data from the file, (3) creates Trapezohedron objects, storing them in a local ArrayList of Trapezohedron objects; and finally, (4) creates an TrapezohedronList object with the name of the list and the ArrayList of Trapezohedron objects, and then prints the TrapezohedronList object followed summary information about the TrapezohedronList 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 Trapezohedron objects. Within a while loop, each set of Trapezohedron data (i.e., label, color, and short edge) is read in, and then an Trapezohedron object should be created and added to the local ArrayList of Trapezohedron objects. After the file has been read in and the ArrayList has been populated, the main method should create an TrapezohedronList object with the name of the list and the ArrayList of Trapezohedron objects as parameters in the constructor.

It should then print the TrapezohedronList object, and then print the summary information about the TrapezohedronList (i.e., print the value returned by the summaryInfo method for the TrapezohedronList). The output from two runs of the main method in TrapezohedronListApp is shown below. The first is produced after reading in the *Trapezohedron_data_1.txt* file, and the second is produced after reading in the *Trapezohedron_data_0.txt* file. Your program output should be formatted exactly as shown on the next page.

Example 1

Line #	Program output
	----jGRASP exec: java TrapezohedronListApp
1	Enter file name: Trapezohedron_data_1.txt
2	
3	Trapezohedron Test List
4	
5	Trapezohedron "Ex 1" is "red" with 20 edges and 12 vertices.
6	edge length antiprism = 8.0902 units
7	short edge = 5.0 units
8	long edge = 13.0902 units
9	surface area = 622.4746 square units
10	volume = 1,155.226 cubic units
11	
12	Trapezohedron "Ex 2" is "blue" with 20 edges and 12 vertices.
13	edge length antiprism = 16.8276 units
14	short edge = 10.4 units
15	long edge = 27.2276 units
16	surface area = 2,693.074 square units
17	volume = 10,395.7774 cubic units
18	
19	Trapezohedron "Ex 3" is "red, blue, tan" with 20 edges and 12 vertices.
20	edge length antiprism = 39.6418 units
21	short edge = 24.5 units
22	long edge = 64.1418 units
23	surface area = 14,945.6145 square units
24	volume = 135,911.1879 cubic units
25	
26	
27	----- Summary for Trapezohedron Test List -----
28	Number of Trapezohedrons: 3
29	Total Surface Area: 18,261.163 square units
30	Total Volume: 147,462.191 cubic units
31	Average Surface Area: 6,087.054 square units
32	Average Volume: 49,154.064 cubic units
33	
	----jGRASP: operation complete.

Example 2

Line #	Program output
	----jGRASP exec: java TrapezohedronListApp
1	Enter file name: Trapezohedron_data_0.txt
2	
3	Trapezohedron Empty Test List
4	
5	
6	----- Summary for Trapezohedron Empty Test List -----
7	Number of Trapezohedrons: 0
8	Total Surface Area: 0.0 square units

```
Total Volume: 0.0 cubic units
Average Surface Area: 0.0 square units
Average Volume: 0.0 cubic units

 ----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 an Trapezohedron 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 Trapezohedron 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 Trapezohedron data items (label, color, edge) should be assigned, after which the Trapezohedron object should be created and added to a local ArrayList of Trapezohedron objects. The next iteration of the loop should then read the next set of three lines then create the next Trapezohedron object and add it to the local ArrayList of Trapezohedron 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 Trapezohedron objects should be used to create an TrapezohedronList object. Then the list should be printed by printing a leading \n and the TrapezohedronList object. Finally, the summary information is printed by printing a leading \n and the value returned by the summaryInfo method invoked on the TrapezohedronList object.

Test: You should test your program minimally (1) by reading in the *Trapezohedron_data_1.txt* input file, which should produce the first output above, and (2) by reading in the *Trapezohedron_data_0.txt* input file, which should produce the second output above. Although your program may not use all the methods in the TrapezohedronList and Trapezohedron classes, 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.

General Notes

1. All input from the keyboard and all output to the screen should done in the main method. Only one Scanner object on System.in should be created and this should be done in the main method. All printing (i.e., using the System.out.print and/or System.out.println methods) should be in the main method. Hence, none of your methods in the Trapezohedron class should do any input/output (I/O).

2. Be sure to download the test data files (*Trapezohedron_data_1.txt* and *Trapezohedron_data_0.txt*) and store them in same folder as your source files. It may be useful 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.