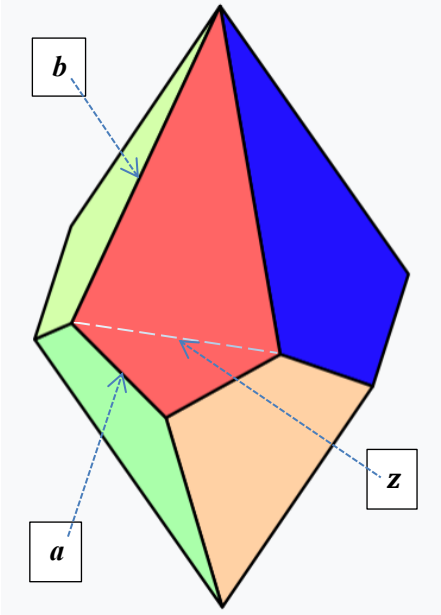
Deliverables

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
- TrapezohedronListMenuApp.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, TrapezohedronListMenuApp, presents a menu to the user with eight options and implements these:

(1) read input file (which creates a TrapezohedronList object), (2) print report, (3) print summary, (4) add a Trapezohedron object to the TrapezohedronList object, (5) delete a Trapezohedron object from the TrapezohedronList object, (6) find a Trapezohedron object in the TrapezohedronList object, (7) Edit a Trapezohedron in the TrapezohedronList object, and (8) quit the program. **[You should create a new "Project 6" folder and copy your Project 5 files** (Trapezohedron.java, TrapezohedronList.java, Trapezohedron_data_1.txt, and Trapezohedron_data_0.txt) **to it, rather than work in the same folder as Project 5 files.]**

- **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 above.
  - `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 –** extended from the previous project by **adding the last six methods below. (Assuming that you successfully created this class in the previous project, just copy TrapezohedronList.java to your new Project folder and then add the indicated methods. Otherwise, you will need to create all of TrapezohedronList.java as part of this project.)**

  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.

- **The following six methods are new in Project 6:**
  - `getList`: Returns the ArrayList of Trapezohedron 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 Trapezohedron objects, uses the list name and the ArrayList to create a TrapezohedronList object, and then returns the TrapezohedronList object. See note #1 under Important Considerations for the TrapezohedronListMenuApp class (last page) to see how this method should be called. The `readFile` method header should include the `throws FileNotFoundException` clause.
  - `addTrapezohedron`: Returns nothing but takes three parameters (label, color, and short edge), creates a new Trapezohedron object, and adds it to the TrapezohedronList object.
  - `findTrapezohedron`: Takes a label of a Trapezohedron as the String parameter and returns the corresponding Trapezohedron object if found in the TrapezohedronList object;

otherwise returns null. This method should ignore case when attempting to match the label.

- `deleteTrapezohedron`: Takes a String as a parameter that represents the label of the Trapezohedron and returns the Trapezohedron if it is found in the TrapezohedronList object and deleted; otherwise returns null. This method should use the String equalsIgnoreCase method when attempting to match a label in the Trapezohedron object to delete.
- `editTrapezohedron`: Takes three parameters (label, color, and short edge), uses the label to find the corresponding the Trapezohedron object. If found, sets the color and short edge to the values passed in as parameters, and returns true. If not found, simply returns false. Note that this method should not set 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 TrapezohedronListMenuApp class.

- **TrapezohedronListMenuApp.java** (replaces TrapezohedronListApp class from the previous project)

Requirements: Create a TrapezohedronListMenuApp 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 TrapezohedronList object, (2) print the TrapezohedronList object, (3) print the summary for the TrapezohedronList object, (4) add a Trapezohedron object to the TrapezohedronList object, (5) delete a Trapezohedron object from the TrapezohedronList object, (6) find a Trapezohedron object in the TrapezohedronList object, (7) Edit a Trapezohedron object in the TrapezohedronList 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 TrapezohedronList 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 (see items 3 and 4 in the **Code and Test** section below).
Below is output produced by running TrapezohedronListMenuApp and printing the action codes with short descriptions followed by the prompt with the abbreviated action codes waiting for the user to select from the menu.

Example 1

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

Below shows the output after the user entered 'r' and then (when prompted) the file name. Notice the output from this action was "File read in and Trapezohedron 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 *Trapezohedron_data_1.txt* file from Project 5 to test your program.

Example 2

Line #	Program output
1	Enter Code [R, P, S, A, D, F, E, or Q]: r
2	File name: Trapezohedron_data_1.txt
3	File read in and Trapezohedron List created
4	
5	Enter Code [R, P, S, A, D, F, E, or Q]:

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

Example 3

Line #	Program output
1	Enter Code [R, P, S, A, D, F, E, or Q]: p
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

```
   short edge = 24.5 units
   long edge = 64.1418 units
   surface area = 14,945.6145 square units
   volume = 135,911.1879 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.

Example 4

Line #	Program output
1	`Enter Code [R, P, S, A, D, F, E, or Q]: s`
2	
3	`----- Summary for Trapezohedron Test List -----`
4	`Number of Trapezohedrons: 3`
5	`Total Surface Area: 18,261.163 square units`
6	`Total Volume: 147,462.191 cubic units`
7	`Average Surface Area: 6,087.054 square units`
8	`Average Volume: 49,154.064 cubic units`
9	
10	`Enter Code [R, P, S, A, D, F, E, or Q]:`

The result of the user selecting 'a' to add a Trapezohedron object is shown below. Note that after 'a' was entered, the user was prompted for label, color, and short edge. Then after the Trapezohedron object is added to the Trapezohedron List, the message "*** Trapezohedron 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.

Example 5

Line #	Program output
1	`Enter Code [R, P, S, A, D, F, E, or Q]: a`
2	`   Label:   My New Label 1`
3	`   Color:      Blue`
4	`   Short Edge: 22`
5	`   *** Trapezohedron added ***`
6	
7	`Enter Code [R, P, S, A, D, F, E, or Q]:`

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

Example 6

Line #	Program output
1	`Enter Code [R, P, S, A, D, F, E, or Q]: d`
2	`   Label: Ex 2`
3	`   "Ex 2" deleted`
4	
5	`Enter Code [R, P, S, A, D, F, E, or Q]: d`
6	`   Label: Ex 10`
7	`   "Ex 10" not found`

8	
9	`Enter Code [R, P, S, A, D, F, E, or Q]:`

In the example below, there is a successful "find" for a Trapezohedron object, followed by an attempt that was not successful (i.e., the Trapezohedron object with label Fake was not found), and finally an invalid code Y was entered.

Example 7

Line #	Program output
1	`Enter Code [R, P, S, A, D, F, E, or Q]: f`
2	`   Label: ex 3`
3	`Trapezohedron "Ex 3" is "red, blue, tan" with 20 edges and 12 vertices.`
4	`   edge length antiprism = 39.6418 units`
5	`   short edge = 24.5 units`
6	`   long edge = 64.1418 units`
7	`   surface area = 14,945.6145 square units`
8	`   volume = 135,911.1879 cubic units`
9	
10	`Enter Code [R, P, S, A, D, F, E, or Q]: f`
11	`   Label: Fake`
12	`   "Fake" not found`
13	
14	`Enter Code [R, P, S, A, D, F, E, or Q]: Y`
15	`   *** invalid code ***`
16	
17	`Enter Code [R, P, S, A, D, F, E, or Q]:`

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

Example 8

Line #	Program output
1	`Enter Code [R, P, S, A, D, F, E, or Q]: e`
2	`   Label: ex 3`
3	`   Color: orange`
4	`   Short Edge: 34`
5	`   "ex 3" successfully edited`
6	
7	`Enter Code [R, P, S, A, D, F, E, or Q]: e`
8	`   Label: another fake label`
9	`   Color: blue`
10	`   Short Edge: 35`
11	`   "another fake label" not found`
12	
13	`Enter Code [R, P, S, A, D, F, E, or Q]:`

Finally, entering a 'q' should quit the application with no message.

Example 9

Line #	Program output
1	`Enter Code [R, P, S, A, D, F, E, or Q]: q`

2	----jGRASP: operation complete.

Code and Test: This class should import java.util.Scanner, java.util.ArrayList, and java.io.FileNotFoundException. Carefully consider the following **important considerations** as you develop this class.

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

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

   The 'R' option in the menu should invoke the readFile method on your TrapezohedronList object. This will return a new TrapezohedronList object based on the data read from the file, and this new TrapezohedronList object should replace (be assigned to) your original TrapezohedronList 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. After you read in the code as a String, you should invoke the toUpperCase() method on the code and then convert the first character of the String to a char as shown in the two statements below.

```
code = code.toUpperCase();
char codeChar = code.charAt(0);
```

4. For the menu, your switch statement expression should evaluate to a char and each case should be a char.

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 Trapezohedron List" option, you should be able to print the TrapezohedronList 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 TrapezohedronList 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.

Although your program may not use all the methods in your Trapezohedron and TrapezohedronList classes, you should ensure that all 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 TrapezohedronList 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.