Project: Trapezohedron 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.

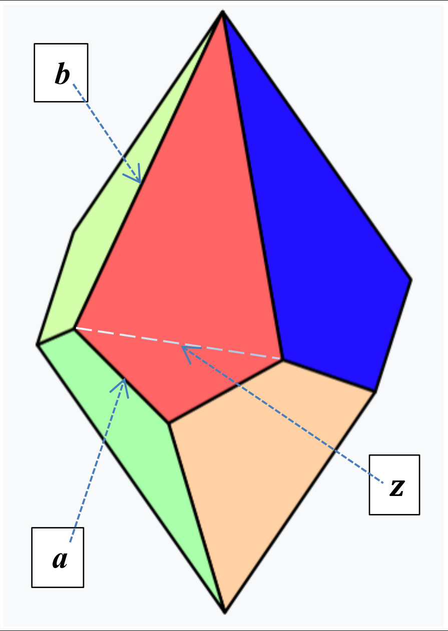
Files to submit to Web-CAT (both files must be submitted together):

- Trapezohedron.java
- TrapezohedronApp.java

Specifications

Overview: You will write a program this week that is composed of two classes: (1) one named Trapezohedron that defines Trapezohedron objects, and (2) the other, TrapezohedronApp, which has a main method that reads in data, creates an Trapezohedron object, and then prints the object.

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.

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

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

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

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

- **Trapezohedron.java**

  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.
  - 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 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.
  - o `getColor`: Accepts no parameters and returns a String representing the color field.
  - o `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. This would be similar to the TrapezohedronApp class you will create below, except that in the TrapezohedronApp class you will read in the values and then create and print the object.

- **TrapezohedronApp.java**

Requirements: Create an TrapezohedronApp class with a main method that reads in values for label, color, and edge. After the values have been read in, main creates an Trapezohedron object and then prints a new line and the object.

Design: The main method should prompt the user to enter the label, color, and short edge. After a value is read in for the short edge, if the value is less than or equal to zero, an appropriate message (see examples below) should be printed followed by a *return* from main. Assuming that the short edge is positive, an Trapezohedron object should be created and printed. Below is an example where the user has entered a negative value for short edge followed by an example using the values from the first example above for label, color, and edge. Your program input/output should be **exactly** as follows.

Example #0

Line #	Program input/output
	`----jGRASP exec: java TrapezohedronApp`
1	`Enter label, color, and short edge length for a trapezohedron.`
2	`   label: Ex0`
3	`   color: white`
4	`   short edge: -9.8`
5	`Error: short edge must be greater than zero.`
6	
	` ----jGRASP: operation complete.`

Example #1

Line #	Program input/output
	`----jGRASP exec: java TrapezohedronApp`
1	`Enter label, color, and short edge length for a trapezohedron.`
2	`   label: Ex1`
3	`   color: red`
4	`   short edge: 5.0`
5	
6	`Trapezohedron "Ex1" is "red" with 20 edges and 12 vertices.`
7	`   edge length antiprism = 8.0902 units`
8	`   short edge = 5.0 units`
9	`   long edge = 13.0902 units`
10	`   surface area = 622.4746 square units`

```
  volume = 1,155.226 cubic units

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

Code: Your program should use the nextLine method of the Scanner class to read user input. Note that this method returns the input as a String, even when it appears to be numeric value. Whenever necessary, you can use the Double.parseDouble method to convert the input String to a double. For example, `Double.parseDouble(s1)` will return the double value represented by String s1, assuming s1 represents a numeric value. For the printed lines requesting input for label, color, and short edge, use a tab "\t" rather than three spaces. After reading in the values, create the new Trapezohedron, say trap, then print it: `System.out.println("\n" + trap);`

Test: You should test several sets of data to make sure that your program is working correctly. Although your main method may not use all the methods in Trapezohedron, you should ensure that all your methods work according to the specification. You can use interactions in jGRASP or you can write another class and main method to exercise the methods. The viewer canvas should also be helpful, especially using the "Basic" viewer and the "toString" viewer for an Trapezohedron object. Web-CAT will test all the methods specified above for Trapezohedron to determine your project grade. You may also find it useful to use the calculator at *https://rechneronline.de/pi/trapezohedron.php*. When using the calculator, set the *Trapezohedron* field to 5- (pentagonal), set *short edge* to a positive value, and then *Round to* 4 decimal places.

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 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. When a method has a return value, you can ignore the return value if it is no interest in the current context. For example, when setShortEdge(3.5) is invoked, it returns true to let the caller know the edge field was set; whereas setShortEdge(-3.5) will return false since the edge field was not set. So, if the caller knows that x is positive, then the return value of setShortEdge(x) can safely be ignored since it can be assumed to be true.

3. Even though your main method may not be using the return value of a method, you can ensure that the return value is correct using interactions.