Project: Octahedron 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.

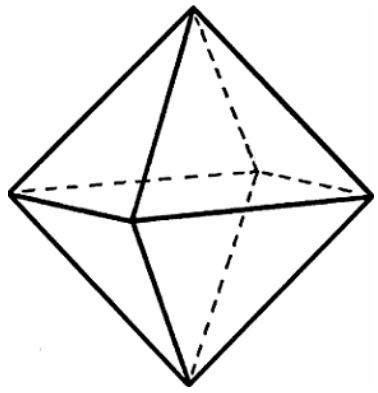
Files to submit to Web-CAT (both files must be submitted together):

- Octahedron.java
- OctahedronApp.java

Specifications

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

An **Octahedron** is a is a polyhedron with eight faces, twelve edges, and six vertices. The term is most commonly used to refer to the regular octahedron, a Platonic solid composed of eight equilateral triangles, four of which meet at each vertex. The formulas are provided to assist you in computing return values for the respective methods in the Octahedron class described in this project. Source: Wikipedia and Merriam-Webster.



Formulas for surface area (A) and volume (V) are shown below where a is the length of an edge.

$$A = 2\sqrt{3}\,a^2$$

$$V = \frac{\sqrt{2}}{3}\,a^3$$

- **Octahedron.java**

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

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

(1) **Fields** (instance variables): label of type String, color of type String, and 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 Octahedron class, and these should be the only instance variables in the class.

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

```
Octahedron ex1 = new Octahedron ("Ex 1", "orange", 5.2);

Octahedron ex2 = new Octahedron (" Ex 2   ", "blue", 20.4);

Octahedron ex3 = new Octahedron ("Ex 3", "orange and blue", 104.5);
```

(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 Octahedron, 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.
- `getEdge`: Accepts no parameters and returns a double representing the edge field.
- `setEdge`: Accepts a double parameter and returns a boolean as follows. If the edge is non-negative, sets the edge field to the double passed in and returns true. Otherwise, the method returns false and the edge is not set.
- `surfaceArea`: Accepts no parameters and returns the double value for the surface area calculated using the value for edge.
- `volume`: Accepts no parameters and returns the double value for the volume calculated using the value for edge.

- `surfaceToVolumeRatio`: Accepts no parameters and returns the double value calculated by dividing the surface area by the volume.
- `toString`: Returns a String containing the information about the Octahedron 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: surfaceArea(), volume(), and surfaceToVolumeRatio(). 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).

```
Octahedron "Ex 1" is "orange" with 12 edges of length 5.2 units.
   surface area = 93.6693 square units
   volume = 66.2832 cubic units
   surface/volume ratio = 1.4132

Octahedron "Ex 2" is "blue" with 12 edges of length 20.4 units.
   surface area = 1,441.6205 square units
   volume = 4,002.066 cubic units
   surface/volume ratio = 0.3602

Octahedron "Ex 3" is "orange and blue" with 12 edges of length 104.5 units.
   surface area = 37,828.8557 square units
   volume = 537,950.8703 cubic units
   surface/volume ratio = 0.0703
```

Code and Test: As you implement your Octahedron 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 Octahedron 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 Octahedron 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 Octahedron then prints it out. This would be similar to the OctahedronApp class you will create below, except that in the OctahedronApp class you will read in the values and then create and print the object.

- **OctahedronApp.java**

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

Design: The main method should prompt the user to enter the label, color, and edge. After a value is read in for edge, if the value is less than zero, an appropriate message

(see examples below) should be printed followed by a *return* from main. Assuming that edge is non-negative, an Octahedron object should be created and printed.
Below is an example where the user has entered a negative value for 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.

Line #	Program input/output
1	`Enter label, color, and edge length for an octahedron.`
2	`   label: Negative Edge Value`
3	`   color: red`
4	`   edge: -10.5`
5	`Error: edge must be non-negative.`

Line #	Program input/output
1	`Enter label, color, and edge length for an octahedron.`
2	`   label: Ex 1`
3	`   color: orange`
4	`   edge: 5.2`
5	
6	`Octahedron "Ex 1" is "orange" with 12 edges of length 5.2 units.`
7	`   surface area = 93.6693 square units`
8	`   volume = 66.2832 cubic units`
9	`   surface/volume ratio = 1.4132`

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 edge, use a tab "\t" rather than three spaces. After reading in the values, create the new Octahedron, say octa, then print it: `System.out.println("\n" + octa);`

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 Octahedron, you should ensure that all of 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 Octahedron object. Web-CAT will test all of the methods specified above for Octahedron 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 System.out.println methods) should be in the main method. Hence, none of your methods in the Octahedron 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 setEdge(3.5) is invoked, it returns true to let the caller know the edge field was set; whereas setEdge(-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 setEdge(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.