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.

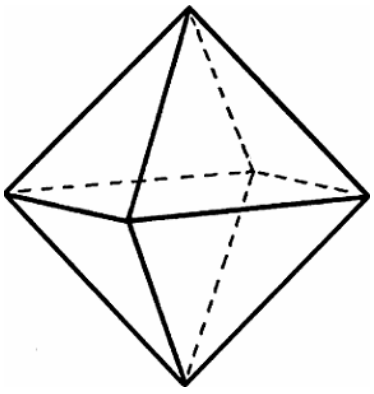
Files to submit to Web-CAT (all three files must be submitted together):

- Octahedron.java
- OctahedronList.java
- OctahedronListApp.java

Specifications

Overview: You will write a program this week that is composed of three classes: the first class defines Octahedron objects, the second class defines OctahedronList objects, and the third, OctahedronListApp, reads in a file name entered by the user then reads the list name and Octahedron data from the file, creates Octahedron objects and stores them in an ArrayList of Octahedron objects, creates an OctahedronList object with the list name and ArrayList, prints the OctahedronList object, and then prints summary information about the OctahedronList 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 (assuming that you successfully created this class in the previous project, just copy the file to your new project folder and go on to OctahedronList.java on page 4. Otherwise, you will need to create Octahedron.java as part of this project.)**

  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.

- **OctahedronList.java**

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

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

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

  (3) Methods: The methods for OctahedronList are described below.
  - `getName`: Returns a String representing the name of the list.
  - `numberOfOctahedrons`: Returns an int representing the number of Octahedron objects in the OctahedronList. If there are zero Octahedron objects in the list, zero should be returned.
  - `totalSurfaceArea`: Returns a double representing the total surface area for all Octahedron objects in the list. If there are zero Octahedron objects in the list, zero should be returned.
  - `totalVolume`: Returns a double representing the total volume for all Octahedron objects in the list. If there are zero Octahedron objects in the list, zero should be returned.
  - `averageSurfaceArea`: Returns a double representing the average surface area for all Octahedron objects in the list. If there are zero Octahedron objects in the list, zero should be returned.
  - `averageVolume`: Returns a double representing the average volume for all Octahedron objects in the list. If there are zero Octahedron objects in the list, zero should be returned.
  - `averageSurfaceToVolumeRatio`: Returns a double representing the average of the surface to volume ratios for all Octahedron objects in the list (i.e., the sum of the surface to volume ratios for all Octahedron in the list divided by the number of Octahedron objects). If there are zero Octahedron objects in the list, zero should be returned.
  - `toString`: Returns a String (does <u>not</u> begin with \n) containing the name of the list followed by each Octahedron 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 Octahedron 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 19 in the output below from OctahedronListApp for the *Octahedron_data_1.txt* input file. [Note that the toString result should **not** include the summary items in lines 21 through 27 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 of the value read from the file) followed by various summary items: number of Octahedron objects, total surface area, total volume, average surface area, average volume, and average surface to volume ratio. Use "#,##0.0##" as the pattern to format the double values. For an example, see lines 21 through 27 in the output below from OctahedronListApp for the *Octahedron_data_1.txt* input file. The second example below shows the output from OctahedronListApp for the *Octahedron_data_0.txt* input file which contains a list name but no Octahedron data.

Code and Test: Remember to import java.util.ArrayList. Each of the five 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 OctahedronList 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 OctahedronList object and calls its methods.

- **OctahedronListApp.java**

  Requirements: Create an OctahedronListApp 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 Octahedron data from the file, (3) creates Octahedron objects, storing them in a local ArrayList of Octahedron objects; and finally, (4) creates an OctahedronList object with the name of the list and the ArrayList of Octahedron objects, and then prints the OctahedronList object followed summary information about the OctahedronList 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 Octahedron objects. Within a while loop, each set of Octahedron data (i.e., label, and axes a, b, and c) is read in, and an Octahedron object should be created and added to the local ArrayList of Octahedron objects. After the file has been read in and the ArrayList has been populated, the main method should create an OctahedronList object with the name of the list and the ArrayList of Octahedron objects as parameters in the constructor. It should then print the OctahedronList object, and then print the summary information about the OctahedronList (i.e., print the value returned by the summaryInfo method for the OctahedronList). The output from two runs of the main method in OctahedronListApp is shown below. The first is produced after reading in the *Octahedron_data_1.txt* file, and the second is produced after reading in the *Octahedron_data_0.txt* file. Your program output should be formatted exactly as shown on the next page.

Line #	Program output
	----jGRASP exec: java OctahedronListApp
1	Enter file name: Octahedron_data_1.txt
2	
3	Octahedron Test List
4	
5	Octahedron "Ex 1" is "orange" with 12 edges of length 5.2 units.
6	surface area = 93.6693 square units
7	volume = 66.2832 cubic units
8	surface/volume ratio = 1.4132
9	
10	Octahedron "Ex 2" is "blue" with 12 edges of length 20.4 units.
11	surface area = 1,441.6205 square units
12	volume = 4,002.066 cubic units
13	surface/volume ratio = 0.3602
14	
15	Octahedron "Ex 3" is "orange and blue" with 12 edges of length 104.5 units.
16	surface area = 37,828.8557 square units
17	volume = 537,950.8703 cubic units
18	surface/volume ratio = 0.0703
19	
20	
21	----- Summary for Octahedron Test List -----
22	Number of Octahedrons: 3
23	Total Surface Area: 39,364.145
24	Total Volume: 542,019.22
25	Average Surface Area: 13,121.382
26	Average Volume: 180,673.073
27	Average Surface/Volume Ratio: 0.615
28	
	----jGRASP: operation complete.

Line #	Program output
	----jGRASP exec: java OctahedronListApp
1	Enter file name: Octahedron_data_0.txt
2	
3	Octahedron Empty Test List
4	
5	
6	----- Summary for Octahedron Empty Test List -----
7	Number of Octahedrons: 0
8	Total Surface Area: 0.0
9	Total Volume: 0.0
10	Average Surface Area: 0.0
11	Average Volume: 0.0
12	Average Surface/Volume Ratio: 0.0
13	
	----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 Octahedron 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 Octahedron 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 Octahedron data items (label, color, edge) should be assigned, after which the Octahedron object should be created and added to a local ArrayList of Octahedron objects. The next iteration of the loop should then read the next set of three lines then create the next Octahedron object and add it to the local ArrayList of Octahedron 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 Octahedron objects should be used to create an OctahedronList object. Then the list should be printed by printing a leading \n and the OctahedronList object. Finally, the summary information is printed by printing a leading \n and the value returned by the summaryInfo method invoked on the OctahedronList object.

Test: You should test your program minimally (1) by reading in the *Octahedron_data_1.txt* input file, which should produce the first output above, and (2) by reading in the *Octahedron_data_0.txt* input file, which should produce the second output above. Although your program may not use all of the methods in the OctahedronList and Octahedron 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 Octahedron class should do any input/output (I/O).

2. Be sure to download the test data files (*Octahedron_data_1.txt* and *Octahedron_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.