Project: ConicalFrustum App

Deliverables

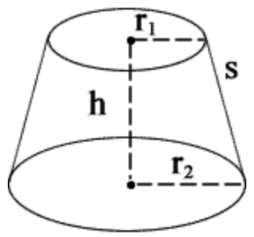
Your project files should be submitted to Web-CAT by the due date and time specified. You may submit your files to the skeleton code assignment until the project due date but should try to do this much earlier. The skeleton code assignment is ungraded, but it checks that your classes and methods are 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):

- ConicalFrustum.java
- ConicalFrustumApp.java

Specifications

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

A ***Conical Frustum*** is a Frustum created by slicing the top off a cone (with the cut made parallel to the base), forming a lower base and an upper base that are circular and parallel.		
	r_1 radius of top r_2 radius of bottom h height s slant height S lateral surface area V volume A total surface area	$V = \frac{\pi * h}{3}(r_1^2 + r_2^2 + (r_1 * r_2))$ $s = \sqrt{(r_1 - r_2)^2 + h^2}$ $S = \pi * (r_1 + r_2) * s$ $A = \pi * (r_1^2 + r_2^2 + (r_1 + r_2) * s)$

Source for figures and formulas: https://www.calculatorsoup.com/images/frustum001.gif

- **ConicalFrustum.java**

  Requirements: Create a ConicalFrustum class that stores the label, radius of top, radius of bottom, and height where the radii and height are non-negative. The ConicalFrustum class also includes methods to set and get each of these fields, as well as methods to calculate the volume, slant height, lateral surface area, and total surface area of a ConicalFrustum object, and a method to provide a String value that describes a ConicalFrustum object.
  Design: The ConicalFrustum class has fields, a constructor, and methods as outlined below.

  (1) **Fields** (instance variables): label of type `String`, radius1 of type `double`, radius2 of type `double`, and height of type `double`. Initialize the `String` to `""` and the `double` variables to 0 in their respective declarations. These instance variables should be private so that they are not directly accessible from outside of the ConicalFrustum class, and these should be the only instance variables (fields) in the class.

(2) **Constructor**: Your ConicalFrustum class must contain a public constructor that accepts four parameters (see types of above) representing the label, radius1, radius2, and height. Instead of assigning the parameters directly to the fields, the respective set method for each field (described below) should be called since they are checking the validity of the parameter. For example, instead of using the statement `label = labelIn;` use the statement `setLabel(labelIn);` Below are examples of how the constructor could be used to create ConicalFrustum 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.

```
ConicalFrustum example1 = new ConicalFrustum("Small", 0.5, 0.75, 0.25);

ConicalFrustum example2 = new ConicalFrustum(" Medium ", 5.1, 10.2, 15.9);

ConicalFrustum example3 = new ConicalFrustum("Large", 98.32, 199.0, 250.0);
```

(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 ConicalFrustum, which should each be public, are described below. See the formulas in the figure above and the Code and Test section below for information on constructing these methods.

- `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 "trimmed" `String` is set to the label field and the method returns `true`. Otherwise, the method returns `false` and the label is not set.
- `getRadius1`: Accepts no parameters and returns a `double` representing the radius1 field.
- `setRadius1`: Takes a `double` parameter and returns a `boolean`. If the `double` parameter is non-negative, then the parameter is set to the radius1 field and the method returns `true`. Otherwise, the method returns `false` and the radius1 field is not set.
- `getRadius2`: Accepts no parameters and returns a `double` representing the radius2 field.
- `setRadius2`: Takes a `double` parameter and returns a `boolean`. If the `double` parameter is non-negative, then the parameter is set to the radius2 field and the method returns `true`. Otherwise, the method returns `false` and the radius2 field is not set.
- `getHeight`: Accepts no parameters and returns a `double` representing the height field.
- `setHeight`: Accepts a `double` parameter and returns a `boolean` as follows. If the `double` parameter is non-negative, then the parameter is set to the height field and the method returns `true`. Otherwise, the method returns `false` and the height field is not set.
- `volume`: Accepts no parameters and returns the `double` value for the volume of the ConicalFrustum. [*Be sure to avoid integer division in your expression*.]
- `slantHeight`: Accepts no parameters and returns the double value for the slant height of the ConicalFrustum.

- `lateralSurfaceArea`: Accepts no parameters and returns the `double` value for the lateral surface area of the ConicalFrustum. Be sure to call your `slantHeight` method as appropriate.
- `totalSurfaceArea`: Accepts no parameters and returns the `double` value for the total surface area of the ConicalFrustum. Be sure to call your `slantHeight` method as appropriate.
- `toString`: Returns a `String` containing the information about the ConicalFrustum object formatted as shown below, including decimal formatting ("#,##0.0##") for the `double` values. Newline and tab escape sequences should be used to achieve the proper layout. In addition to the field values (or corresponding "get" methods), the following methods should be used to compute appropriate values in the `toString` method: `volume()`, `slantHeight()`, `lateralSurfaceArea()`, and `totallSurfaceArea()`. Each line should have no trailing spaces (e.g., there should be no spaces before a newline (\n) character). The `toString` value for `example1`, `example2`, and `example3` respectively are shown below (the blank lines are not part of the `toString` values).

```
ConicalFrustum "Small" with radius1 = 0.5, radius2 = 0.75, and height = 0.25 has:
   volume = 0.311 cubic units
   slant height = 0.354 units
   lateral surface area = 1.388 units
   total surface area = 3.941 square units

ConicalFrustum "Medium" with radius1 = 5.1, radius2 = 10.2, and height = 15.9 has:
   volume = 3,031.546 cubic units
   slant height = 16.698 units
   lateral surface area = 802.608 units
   total surface area = 1,211.172 square units

ConicalFrustum "Large" with radius1 = 98.32, radius2 = 199.0, and height = 250.0 has:
   volume = 18,020,568.788 cubic units
   slant height = 269.512 units
   lateral surface area = 251,739.485 units
   total surface area = 406,518.914 square units
```

Code and Test: As you implement your ConicalFrustum 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 ConicalFrustum in interactions (e.g., copy/paste the examples 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 a ConicalFrustum 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 ConicalFrustum then prints it out. This would be similar to the ConicalFrustumApp class you will below, except that in the ConicalFrustumApp class you will read in the values and then create and print the object.

- **ConicalFrustumApp.java**

    - **Requirements**: Create a ConicalFrustumApp class with a `main` method that reads in values for label, radius1, radius2, and height. After the values have been read in, `main` creates a ConicalFrustum object and then prints the object.

    - **Design**: The `main` method should prompt the user to enter the label, radius1, radius2, and height. After each numeric value is read (i.e., radius1, radius2, and height), if the value is less than zero, an appropriate message (see examples below) should be printed followed by a *return* statement to end `main`. Consider having three separate if statements. Assuming that radius1, radius2, and height are non-negative, a ConicalFrustum object should be created and printed. Below are four examples: (1) the user has entered a negative value for radius1, (2) the user has entered a negative value for radius2, (3) the user has entered a negative value for height, and (4) the user has entered the values from the first example above for label, radius1, radius2, and height. Your program input/output should be **exactly** as follows.

Line # Program input/output

```
Enter label, radius1, radius2, and height for a conical frustum.
   label: One with a bad radius1
   radius1: -1
Error: radius1 must be non-negative.
```

Line # Program input/output

```
Enter label, radius1, radius2, and height for a conical frustum.
   label: One with a bad radius2
   radius1: 10.5
   radius2: -2.5
Error: radius2 must be non-negative.
```

Line # Program input/output

```
Enter label, radius1, radius2, and height for a conical frustum.
   label: One with a bad height
   radius1: 25
   radius2: 26
   height: -1
Error: height must be non-negative.
```

Line # Program input/output

```
Enter label, radius1, radius2, and height for a conical frustum.
   label: Small
   radius1: 0.5
   radius2: 0.75
   height: 0.25
ConicalFrustum "Small" with radius1 = 0.5, radius2 = 0.75, and height = 0.25 has:
   volume = 0.311 cubic units
   slant height = 0.354 units
   lateral surface area = 1.388 units
   total surface area = 3.941 square units
```

Code: To read user input, your program should use the `nextLine` method of the `Scanner` class. Note that this method returns the input as a `String`. 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`. For the printed lines requesting input for label, radius1, radius2, and height, use a tab `"\t"` rather than three spaces. After you have created the object, it can be printed simply by using its variable name; i.e., when the object variable name is evaluated in the println statement, its `toString()` method is automatically called. So printing the object reference variable `myObj` is equivalent to printing the return value of the `myObj.toString()` method call.

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 ConicalFrustum, 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 a ConicalFrustum object. Web-CAT will test all of the methods specified above for ConicalFrustum 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 ConicalFrustum 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 `setRadius1(3.5)` is invoked, it sets the `radius1` field and returns `true`; whereas `setRadius1(-3.5)` will return `false` without setting the `radius1` field. If the caller knows that the value of `x` is non-negative in the method call `setRadius1(x)`, it can be assumed that the field was set appropriately and the return value can be safely ignored. Although caller can always ignore the return value; however, it is important to provide the boolean return value to indicate whether or not the `setRadius1` method did in fact set the field.

3. If the caller knows that the value of `x` is non-negative in the method call `setRadius1(x)`, it can be assumed that the field was set appropriately and the return value can be ignored safely.

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