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.
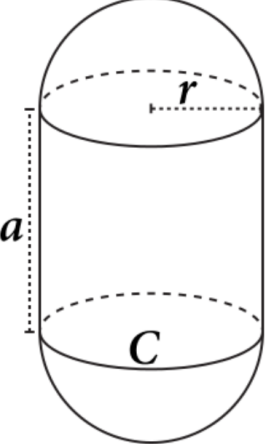
Files to submit to Web-CAT (all three files must be submitted together):
- Spherocylinder.java
- SpherocylinderList.java
- SpherocylinderListApp.java

Specifications

Overview: You will write a program this week that is composed of three classes: the first class defines Spherocylinder objects, the second class defines SpherocylinderList objects, and the third, SpherocylinderListApp, reads in a file name entered by the user then reads the list name and Spherocylinder data from the file, creates Spherocylinder objects and stores them in an ArrayList, creates a SpherocylinderList object with the list name and ArrayList, prints the SpherocylinderList object, and then prints summary information about the SpherocylinderList object.

A spherocylinder (or capsule) is a 3-dimensional object made up of two hemispheres connected by a cylinder as shown below. The formulas are provided to assist you in computing return values for the respective Spherocylinder methods described in this project.

The variables are abbreviated as follows:
- *r* is radius
- *a* is cylinder height
- *C* is Circumference
- *SA* is Surface Area
- *V* is Volume

$$C = 2\,\pi\,r$$

$$SA = 2\pi r(2r + a)$$

$$V = \pi r^2 \left(\frac{4}{3}r + a \right)$$

Source for figures and formulas: https://en.wikipedia.org/wiki/Capsule_(geometry)

- **Spherocylinder.java (assuming that you successfully created this class in Project 4, just copy the file to your new Project 5 folder and go on to SpherocylinderList.java on page 4. Otherwise, you will need to create Spherocylinder.java as part of this project.)**

    Requirements: Create a Spherocylinder class that stores the label, radius, and cylinder height where both radius and cylinder height are non-negative. The Spherocylinder class also includes methods to set and get each of these fields, as well as methods to calculate the circumference, surface area, and volume of a Spherocylinder object, and a method to provide a String value that describes a Spherocylinder object.

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

    (1) **Fields** (instance variables): label of type `String`, radius of type `double`, and cylinder 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 Spherocylinder class, and <u>these should be the only instance variables (fields) in the class</u>.

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

```
Spherocylinder example1 = new Spherocylinder("Small Example", 0.5, 0.25);

Spherocylinder example2 = new Spherocylinder(" Medium Example ", 10.8, 10.1);

Spherocylinder example3 = new Spherocylinder("Large Example", 98.32, 99.0);
```

    (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 Spherocylinder, 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.

    - `getRadius`: Accepts no parameters and returns a `double` representing the radius field.

    - `setRadius`: Takes a `double` parameter and returns a `boolean`. If the `double` parameter is non-negative, then the parameter is set to the radius field and the method returns `true`. Otherwise, the method returns `false` and the radius field is not set.

- o `getCylinderHeight`: Accepts no parameters and returns a `double` representing the cylinder height field.

- o `setCylinderHeight`: Accepts a `double` parameter and returns a `boolean` as follows. If the `double` parameter is non-negative, then the parameter is set to the cylinder height field and the method returns `true`. Otherwise, the method returns `false` and the cylinder height field is not set.

- o `circumference`: Accepts no parameters and returns the `double` value for the circumference of the Spherocylinder.

- o `surfaceArea`: Accepts no parameters and returns the `double` value for the total surface area of the Spherocylinder.

- o `volume`: Accepts no parameters and returns the double value for the volume of the Spherocylinder. [*Be sure to avoid integer division in your expression.*]

- o `toString`: Returns a `String` containing the information about the Spherocylinder 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 within the String but it should not begin or end with a newline. In addition to the field values (or corresponding "get" methods), the following methods should be used to compute appropriate values in the `toString` method: `circumference()`, `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 `example1`, `example2`, and `example3` respectively are shown below (the blank lines are not part of the `toString` values).

```
Spherocylinder "Small Example" with radius 0.5 and cylinder height 0.25 has:
    circumference = 3.142 units
    surface area = 3.927 square units
    volume = 0.72 cubic units

Spherocylinder "Medium Example" with radius 10.8 and cylinder height 10.1 has:
    circumference = 67.858 units
    surface area = 2,151.111 square units
    volume = 8,977.666 cubic units

Spherocylinder "Large Example" with radius 98.32 and cylinder height 99.0 has:
    circumference = 617.763 units
    surface area = 182,635.388 square units
    volume = 6,987,754.655 cubic units
```

Code and Test: As you implement your Spherocylinder 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 Spherocylinder 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 Spherocylinder 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 Spherocylinder then prints it out. This would be similar to SpherocylinderApp (Project 4), except that in the SpherocylinderApp class the values are read in and then the object is created and printed.

- **SpherocylinderList.java**

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

  Design: The SpherocylinderList 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 Spherocylinder objects. These are the only fields (or instance variables) that this class should have.
  (2) Constructor: Your SpherocylinderList class must contain a constructor that accepts a parameter of type String representing the name of the list and a parameter of type ArrayList< Spherocylinder> representing the list of Spherocylinder objects. These parameters should be used to assign the fields described above (i.e., the instance variables).

  (3) Methods: The methods for SpherocylinderList are described below.
  - `getName`: Returns a String representing the name of the list.
  - `numberOfSpherocylinders`: Returns an int representing the number of Spherocylinder objects in the SpherocylinderList. If there are zero Spherocylinder objects in the list, zero should be returned.
  - `totalSurfaceArea`: Returns a double representing the total surface areas for all Spherocylinder objects in the list. If there are zero Spherocylinder objects in the list, zero should be returned.
  - `totalVolume`: Returns a double representing the total volumes for all Spherocylinder objects in the list. If there are zero Spherocylinder objects in the list, zero should be returned.
  - `averageSurfaceArea`: Returns a double representing the average surface area for all Spherocylinder objects in the list. If there are zero Spherocylinder objects in the list, zero should be returned.
  - `averageVolume`: Returns a double representing the average volume for all Spherocylinder objects in the list. If there are zero Spherocylinder 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 Spherocylinder 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 Spherocylinder object in the list (adding a \n <u>before and after</u> each). Be sure to include appropriate newline escape sequences. For an example, see <u>lines 2 through 14</u> in the output below from SpherocylinderListApp for the *Spherocylinder_data_1.txt* input file. [Note that <u>the toString result should **not** include the summary items in lines 21

through 26 of the example. These lines represent the return value of the summaryInfo
method below which will be called from `main`.]

- o   `summaryInfo`: Returns a String (<u>does not begin or end with \n</u>) containing the name of
  the list (which can change depending of the value read from the file) followed by various
  summary items: number of Spherocylinders, 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 <u>lines 21 through 26</u> in the output below from
  SpherocylinderListApp for the *Spherocylinder_data_1.txt* input file. The second
  example below shows the output from SpherocylinderListApp for the
  Spherocylinder_data_0.txt input file which contains a list name but no Spherocylinder
  data.

Code and Test: Remember to import java.util.ArrayList. The `totalSurfaceArea` and
`totalVolume` methods each requires that you use a loop (i.e., a while loop) to retrieve each
object in the ArrayList. As you implement your SpherocylinderList class, you can compile it and
then test it using interactions. Alternatively, (1) you can create a class with a simple main method
that creates a SpherocylinderList object and calls its methods; or (2) you can create the
SpherocylinderListApp described next.

- **SpherocylinderListApp.java**

  Requirements: Create a SpherocylinderListApp class with a `main` method that reads in the
  name of the data file entered by the user and then reads list name and Spherocylinder data from
  the file, creates Spherocylinder objects, stores them in a local ArrayList of Spherocylinder
  objects, creates a SpherocylinderList object with the name of the list and the <u>ArrayList of
  Spherocylinder objects</u>, and then prints the SpherocylinderList object followed summary
  information about the SpherocylinderList object. **<u>All input and output for this project should
  be done in the main method</u>**.

- **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 Spherocylinder objects. After each set of Spherocylinder data is read in, a
  Spherocylinder object should be created and stored in the local ArrayList of Spherocylinder
  objects. After the file has been read in and the ArrayList has been populated, the main method
  should create a SpherocylinderList object with the name of the list and the ArrayList of
  Spherocylinder objects as parameters in the constructor. It should then print the
  SpherocylinderList object, then print the <u>summary</u> information about the SpherocylinderList (i.e.,
  print the value returned by the `summaryInfo` method for the SpherocylinderList). The output
  from two runs of the main method in SpherocylinderListApp is shown below. The first is
  produced after reading in the *Spherocylinder_data_1.txt* file, and the second is produced after
  reading in the *Spherocylinder_data_0.txt* file. Your program output should be formatted exactly
  as shown.

Line #	Program output
	----jGRASP exec: java SpherocylinderListApp
1	Enter file name: spherocylinder_data_1.txt
2	
3	Spherocylinder Test List
4	
5	Spherocylinder "Small Example" with radius 0.5 and cylinder height 0.25 has:
6	circumference = 3.142 units
7	surface area = 3.927 square units
8	volume = 0.72 cubic units
9	
10	Spherocylinder "Medium Example" with radius 10.8 and cylinder height 10.1 has:
11	circumference = 67.858 units
12	surface area = 2,151.111 square units
13	volume = 8,977.666 cubic units
14	
15	Spherocylinder "Large Example" with radius 98.32 and cylinder height 99.0 has:
16	circumference = 617.763 units
17	surface area = 182,635.388 square units
18	volume = 6,987,754.655 cubic units
19	
20	
21	----- Summary for Spherocylinder Test List -----
22	Number of Spherocylinders: 3
23	Total Surface Area: 184,790.426
24	Total Volume: 6,996,733.041
25	Average Surface Area: 61,596.809
26	Average Volume: 2,332,244.347
27	
	----jGRASP: operation complete.

Line #	Program output
	----jGRASP exec: java SpherocylinderListApp
1	Enter file name: spherocylinder_data_0.txt
2	
3	Spherocylinder Empty Test List
4	
5	
6	----- Summary for Spherocylinder Empty Test List -----
7	Number of Spherocylinders: 0
8	Total Surface Area: 0.0
9	Total Volume: 0.0
10	Average Surface Area: 0.0
11	Average Volume: 0.0
12	
	----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 a Spherocylinder 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

Spherocylinder 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 (<mark>see note 3 below</mark>). As each of the lines is
read from the file, the respective local variables for the Spherocylinder data items (label, radius,
and cylinder height) should be assigned, after which the Spherocylinder object should be created
and added to a local ArrayList of Spherocylinder objects. The next iteration of the loop should
then read the next set of three lines then create the next Spherocylinder object and add it to the
local ArrayList of Spherocylinder 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
Spherocylinder objects should be used to create a SpherocylinderList object. The list should be
printed by printing a leading \n and the SpherocylinderList object. Finally, the summary
information is printed by printing a leading \n and the value returned by the summaryInfo method
invoked on the SpherocylinderList object.

Test: You should test your program minimally (1) by reading in the *Spherocylinder_data_1.txt*
input file, which should produce the first output above, and (2) by reading in the
Spherocylinder_data_0.txt input file, which should produce the second output above. Although
your program may not use all of the methods in SpherocylinderList and Spherocylinder, 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 System.out.println methods) should
    be in the main method. Hence, none of your methods in the Spherocylinder class should do
    any input/output (I/O).

2. Be sure to download the test data files (*Spherocylinder_data_1.txt* and
    Spherocylinder_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.

3. Reading the data from the input file – Since each of the date items for label, radius, and
    cylinder height are on separate lines, within the while loop each line should be read with a
    .nextLine() on the file Scanner and then parsed into a double when appropriate. Below is an
    example where label, radius, and cylinderHeight are local variables and the blank should be
    filled in with the variable for the Scanner for the input file.

    ```
    label = _____.nextLine();
    radius = Double.parseDouble(_____.nextLine());
    cylinderHeight = Double.parseDouble(_____.nextLine());
    ```