# Lab Three Assignment (Ch7-9, CS170 – Spring)

## *Due: 10:00 pm, Wednesday (4/3)*

**Submission requirement**: your lab must be completed as a project in Eclipse with required documentation for each source code by following the steps explained in **Steps to make a zipped Eclipse project file** below this lab spec. Submit it via Assignments link in Canvas. There is an explanation how to submit your lab after you click on the lab assignment specification link in Canvas.

**Source code documentation requirement**:

- Use of meaningful names for variables, classes, and methods and meaningful comments.
- Your name (last name, first name), class title and section #, the assignment #, and a brief description of the lab as comment lines must show up at the top of each source code.
- There will be no point if your program does not run. There will be points off if you don't follow the instruction and requirements to code your lab.

**Steps to make a zipped Eclipse project file**

1. Create your lab as a Java project and all of source code should be in this project.
2. Highlight the project title and make right mouse click, select **Properties**, and you will see the directory or folder in your computer where the project is saved.
3. Navigate to that directory, copy that folder to a different directory, say **C:/Temp**, and click on the right mouse button, select **Sent to**, and select the **Compressed/zipped folder**. Your file is ready for submission.
4. For test if your file can be opened in Eclipse, select **File**, then **Switch Workspace**, and select **Other…**, type a new directory as your new workspace for test, say, **Desktop/MyLabs,** Eclipse will create a new workspace. Select **File**, **Import…**, click on **General**, then **Existing Project into Workspace**, and click on **Next**, click on **Select archive file** button, navigate to the zipped project file, and then click on **Finish**. It should be executable now if your project is correct.

**Part I (p. 240)**: Exercise 12. Use principles of inheritance to write code calculating the area, surface area, and volume of rectangular objects. First, write an abstract super class **RegutangularShape**, then write a subclass **Rectangle** that inherits from the super class to compute areas of rectangles, including squares if there is only one data. Finally, write another subclass **Cuboid** that inherits from its super class **Rectangle** above to compute surface areas and volumes of cuboids, including 3 equal sided cube. Must apply code-reusability in coding and understand what the code-reusability first discussed in the text. Override **toString()** methods in all developed classes to return the proper data from each object. Write a driver code to test your program creating at least 4 different objects in computing their areas, surface areas, and/or volumes, as applicable to the instantiated objects. Document the source code as required above.

**Part II (p. 259)**: Exercise 8. Obtain example code files **Shape.java**, **Circle2.java**, **Sphere.java**, and **CircleShapeApp.java** from the downloaded files in **Ch8**. Compile and execute the example and understand the polymorphism it performs. Modify the application as follows:

a. Code a class called **Cone** inherited from class **Circle**. **Cone** will override the **computeArea()** and **computeVolume()** methods from its superclass to calculate the area and volume of a **Cone** object. Must apply code-reusability in the calculation formula.

b. Modify the driver code, **CircleShapeApp.java**, so it will also create an object of **Cone** with a height of 12.5 and diameter of 10.48. Modify the polymorphic calls to calculate and display the result of the calculations.

c. (**Optional extra-credit**): Instead of hard-coded data, modify your application to be interactive using **JOptionPane**, thus it will ask user to enter proper data for different object, continue to utilize the **Validator** class you have developed before to validate all wrong data entries, including negative and non-numerical data. Your program must continue to run until the user enters "n" to terminate the execution. You must also validate the user's entry so that only "y" and "n" are accepted.

d. Document your source code as required and save your work for the submission.