Lab Two Assignment (Ch5-6, CS170 – Spring)

Due: 10:00 pm, Wednesday (3/13)

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: Code an operation class **TempConvertWithValidate** using any version of Validator discussed in the text in Chapter 5, so it can verify all invalid data entries in the temperature conversion you did in Lab 1. Understand the examples of Validators first before you start to do this part of lab. In your code you will display a menu first as follows:

1. Convert temperature from Celsius to Fahrenheit
2. Convert temperature from Celsius to Fahrenheit
3. Quit

You will verify these 3 options and then in the method calls to do the temperature conversions, you will also verify the input data from the user ranged from -100 to 200. All other data entries than this range are considered as invalid and you must verify them using any versions of **Validator** class provided in the example code from the text.

You will code a driver class **TempCovertWithValidateApp** to test your application. Your program must continue to run until the user selects Quit to terminate the execution. Document each of your source code as required above.

Part II - Challenge question for extra-credit (Optional) from **Part I:** Code all operations including process of input data and the while loop for continuously run the program in the operation class, not in the driver class. Therefore, your driver class for test looks as follows:

```
public class TempCovertWithValidateApp {
    public static void main(String[] args) {
        int option = 0;
        TempCovertWithValidate temp = new TempConvertWithValidate();        //Create object
        Option = temp.showAndVerifyMenu();           //call method to display menu and verify the option
        if (option != 3) {                           //option must be 1 or 2
           temp.assignChoice(option);                //set the choice
           temp.inputData();                         //let user enters data and verify it
           temp.convert();                           //process the temp conversion based on the choice
           temp.display();                           //output the conversion result
        }
    }
}
```

You may submit **Part II only** for all credit you can earn. But I suggest you to code them separately first as your learning process. Document each of your source code as required above.

Part III: Use of constructor and method overloading to code a complete class called **CircularVolume** and the driver class called **CircleVolumeApp**. You will create a sphere object if there is only one data entered a radius; create an cylinder object if there are two data entered as radius and height; create an ellipsoid object if there are 3 data entered as a, b, c axis, respectively. You will call all three overloaded methods **computeVolume()** with different arguments. Fond formulas from the Internet search. Assume all data are doubles. All input data must be entered from user. You must verify all invalid data entries using any version of provided **Validator** class discussed in the textbook. Negative and non-numerical data are also considered as invalid and you must verify them. The result of a volume calculation will be displayed along with the object information including its name and input data. Document each of your source code as required above.