# Lab 09.1: Working with Inheritance

## *Objective*

This lab will provide some practice in using inheritance. You will also experiment with overriding methods and using information from an inherited class.

## *Overview*

In this lab you will:

- Create a new class that extends an existing class
- Implement methods using overriding
- Add new functionality
- Create objects and test your new code

## *Step by Step Instructions*

### Exercise 1: Using the extends keyword

1. Create a new class named **Cube** that extends **Box** and place it in the **com.lq.exercises** package in the **ClassExercises** project.

2. A **Cube** is like a **Box** however all three dimensions of a **Cube** must be the same. We simulated this in our **Box** class by providing a constructor that would set all 3 sides to the same value. We will formalize this here. We don't really need any new attributes since we inherit everything we need from **Box**.

3. The first thing we need to do is define a constructor for **Cube**. Since a **Cube** must have all 3 sides set to the same value, our constructor only needs one parameter. We do not need to set any data in the **Cube** class so all we need is a call to our super class constructor [Hint: Use the `super` keyword as the first line in the **Cube** constructor].

4. **Cube** inherits all the `public` methods from **Box**. However, if we set the `length` of a **Cube** to a new value, we also want to guarantee that the `height` and `width` are updated as well. Override each of the following methods in the **Cube** class to guarantee this functionality. (Remember, an overridden method must have the exact same signature as the base class method it is overriding).

   a. `setLength()`
   b. `setWidth()`
   c. `setHeight()`

Here is what it might look like for one of the methods. Note that the author chose to only change the value if a different one has been passed in.

```
public void setHeight(double height) {
    if(height != super.getHeight()) {
        super.setHeight(height);
        super.setLength(height);
        super.setWidth(height);
    }
}
```

**Exercise 2: Adding Functionality to Cube**

5. Add two new methods to get and set the sides of the **Cube** to a value. These methods work in a similar manner to the **Box** methods but will only be available to cubes.

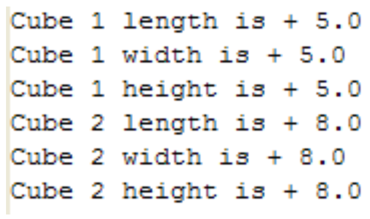
    a. Create a `setSide()` method that takes one parameter of type `double` and returns a `void`. Ensure that the sides are set to a valid value (>0).
    b. Create a `getSide()` method that takes no parameters and returns a `double`.

**Exercise 3: Try out the Cube class**

6. Create a new class named **CubeDriver** in the **com.lq.exercises** package. Ensure that this class has a `main()` method defined.

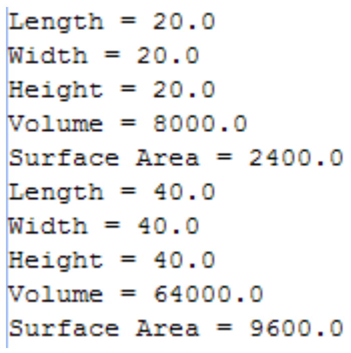
7. In the main method of **CubeTest**, create two **Cube** objects using the keyword `new`.
    a. A cube named **cube1** with all sides set to 5
    b. A cube named **cube2** with all sides set to 8

8. Using the get methods, print out the value for each attribute of both cubes. Ensure that the attributes were set correctly by the **Cube** constructor. Execute the `main()` method for **CubeDriver.** The output should be similar to the following. If there are any errors in your **Cube** code at this point, correct them and re-run the test.



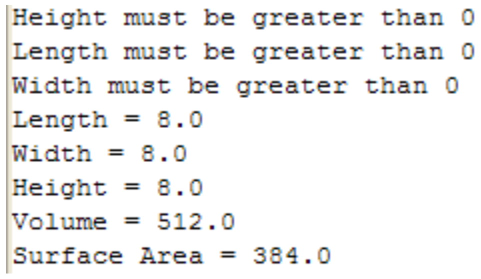
```
Cube 1 length is + 5.0
Cube 1 width is + 5.0
Cube 1 height is + 5.0
Cube 2 length is + 8.0
Cube 2 width is + 8.0
Cube 2 height is + 8.0
```

9. Ensure that your set methods work correctly. Call the `setLength()` method for **Cube** `cube1` and change the `length` to 20. Use the inherited `printBox()` method to ensure that all sides were set correctly. Do the same test again calling the `setSide()` method with a value of 40. Your output should look as follows:



```
Length = 20.0
Width = 20.0
Height = 20.0
Volume = 8000.0
Surface Area = 2400.0
Length = 40.0
Width = 40.0
Height = 40.0
Volume = 64000.0
Surface Area = 9600.0
```

10. Using the `setWidth`() method, change the width of **Cube** cube2 to -5. Did you receive the appropriate error message? Execute the `printBox()` method again and ensure that the value was not changed. Your output should look as follows:



```
Height must be greater than 0
Length must be greater than 0
Width must be greater than 0
Length = 8.0
Width = 8.0
Height = 8.0
Volume = 512.0
Surface Area = 384.0
```

**<u>Challenge Exercises:</u>**

Refactor your code to eliminate duplicate logic.

Experiment with different algorithms so that you don't receive 3 error messages when you try and change the size of a cube to an invalid value. There are many ways of handling this situation. Explore different techniques.

Is there any way that inheritance might be put to good use in your **Calculator** classes? Write a driver to exercise your Calculator classes.