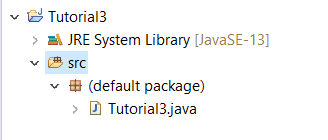
# **Lab 3\_1 Assignment**

## Part 1:

**Important:** Make sure you include the 3 header comments listed in previous lab assignments.  
For each project, once your program is completed and runs, copy the code into a text file named Tutorial3.txt (or Lab3A.txt or Lab3B.txt) and submit it in this assignment. Paste both classes for each program into the same document.

**Also, in Eclipse, go to the File tab and click Close All to close previous projects before starting a new one.**

1. Following the instructions from Lab1A, open Eclipse and start a new Java project named **Tutorial3**, and a new class named **Tutorial3**
2. Click on src in the left window again and go to **File 🡪 New 🡪** Class to create a second class named BoxRobot. Do NOT check the checkbox beside **public static void main (String [] args)** on this one. (Only one class should have a main method.)



1. On the BoxRobot tab, add a blank line above the **public class BoxRobot {** if you need to, and type **// This class will calculate the volume and surface area of a box**

(Documentation of a class’s purpose is important in the real world.)

1. Click inside the { } for the BoxRobot class to type the rest of the code for it.
2. First, we need to add the instance variables, which will be the dimensions of a box. Type   
   **private double length, width, height;**

The instance variables are private to keep their data from being changed by outside classes without following our class’s rules.

1. Note: Before reading the notes about stale data in Chapter 3, I would have added the instance variables for volume and surfaceArea, but I think we will just recalculate those whenever an outside class requests them. That way they will always be accurate.
2. Leave a blank line (for readability) and let’s create our first constructor. Remember, the constructor has the same name as the class, has no return type, and its job is to assign values to the instance variables (fields) and do any other necessary object set up. (Watch the first part of the 2nd video for an explanation of constructors.)
3. Type **public BoxRobot(double inlength, double inwidth, double inheight)** {

}

1. Inside the { } for the constructor we will add code to assign values to the instance variables. The name of the instance variable always comes first, and it receives a value from the parameter.

Type the following:  
**length = inlength;  
width = inwidth;**

**height = inheight;**

1. Now we are going to overload the constructor, which means we will make another constructor which has different parameters. This will be for the possibility that the box is a cube, which means we only need one side length for it, because all side are equal. Here is the next constructor:

**public BoxRobot(double side)**

**{**

**length = side;**

**width = side;**

**height = side;**

**}**Notice that it only has one parameter (side), and it sets all 3 instance variables equal to side, because they are all the same. The main program will be able to look at the number of parameters sent in a constructor call, and use that to decide which constructor to run.

1. Now we will create a couple of methods that will calculate and return the volume of the box and the surface area of the box.
2. After the second constructor, leave a blank line. Then type **public double getVolume() { }**

It doesn’t have any parameters because this method will compute the volume using the instance variables; then it will return that value.

1. Inside the { } type   
   **double volume;  
   volume = length \* width \* height;  
   return volume;**

(Or you could just type **return length \* width \* height;** to be more efficient.)

1. Next add another method to calculate surface area. Type **public double getSurfaceArea() { }**
2. The type **return 2\*length\*width + 2\*length\*height + 2\*width\*height;** inside the { }
3. Now we’re going to add a method that lets you take advantage of a **super-secret shortcut.** 😉  
   In your main method, if you have an object named robbie and you try to just print robbie (System.out.println(robbie)), all you will get is the weird memory address of the object – 0b0xxxx or something like that.  
   **However**, if you make a **public String toString** method (with no parameters) inside your BoxRobot class, when you try to print robbie back in the main method, it will print whatever is returned by the toString method instead of the 0b0xxxx.   
   Pretty cool! (Okay, it’s not that exciting, but it is pretty useful.)  
   **(Add this to your notes; you’ll need it in the future.)**
4. Add the following method: **public String toString() { }**
5. Inside the method type   
   **return "Length: " + length + "\t Width: " + width + "\t Height: " + height  
    + "\n\tVolume: " + getVolume() + "\tSurface Area: " + getSurfaceArea();**

This will create a String value with all the labels and values we need, and it will return that value to the sender. Notice that we call getVolume and getSurfaceArea as we create our string.

1. We have now made a fully functional BoxRobot class that can receive values, compute volume and surface area, and then return that information in a nicely formatted way. Let’s start working on our main method in the Tutorial3 class now.
2. Click on the Tutorial3 tab. Inside the main method we will declare and instantiate a BoxRobot object named robbie. Type **BoxRobot robbie = new BoxRobot(10, 15, 10);** This creates robbie and calls the first constructor because we are sending in 3 parameters. robbie’s length, width & height will be 10, 15 & 10, respectively.
3. Now let’s create a second BoxRobot object named moe. moe is a cube, so we only need to send one side length in for him. Type **BoxRobot moe = new BoxRobot(20);**Since only one parameter is sent in, the second constructor will be called, and length, width & height will all equal 20.
4. Now let’s get robbie’s surface area. Type **System.out.println("robbie Surface Area: " +** **robbie.getSurfaceArea());**
5. Then we’ll print moe’s volume. Type **System.out.println("moe Volume: " + moe.getVolume());**
6. And last, but not least, we will use our super-secret shortcut to print all the information about an object that we put together in the toString method.  
   Type **System.out.print(**"**robbie:** "**);**   
    **System.out.println(robbie);**
7. Them type **System.out.print(**"**moe:** "**);**   
    **System.out.println(moe);**
8. Run and test your program.

## Part 2:

1. Create a Java project called **Lab3A** and a class named **Lab3A**.
2. Create a second new class named **Employee**.
3. In the **Employee** class:
   1. Add the following private instance variables:
      1. name (String)
      2. job (String)
      3. salary (double)
   2. Add a constructor that receives 3 parameters (one for each instance variable) and sets each instance variable equal to the corresponding variable. (Refer to the Tutorial3 program constructor if needed to remember how to do this.)
   3. Add a public String method named **getName** (no parameter) that returns name.
   4. Add a public double method named **getSalary** (no parameter) that returns salary.
   5. Add a public void method named **giveRaise** with a double parameter named **raisePercent**. It should give the employee a raise by changing the salary to be equal to salary \* (1 + **raisePercent**).
   6. Add a public String method **toString** (no parameters). It should create a String variable with the employee’s instance data. (I will give you the code here, but add this to your notes, because you will do this on future assignments.)

**String result = "Name: " + name + "\nJob: " + job + "\nSalary: " + salary;**

**return result;**

1. Back in the main **Lab3A** class.
   1. Declare and instantiate an **Employee** object named **helen** sending the following parameters to the constructor ("Helen Lee", "Accountant", 45000.00)
   2. Declare and instantiate an **Employee** object named **brandon** sending the following parameters to the constructor ("Brandon Charles", "Analyst", 48000.00)
   3. Call **getName** for the **helen** object and print the returned value.   
      ( System.out.println(helen.getName()); )
   4. Call **getSalary** for the **helen** object and print the returned value.
   5. Call **getName** for the **brandon** object and print the returned value.
   6. Give **helen** a 15% raise by calling the **giveRaise** method for **helen** and sending 0.15 as the parameter. (You don’t print anything here.)
   7. Give **brandon** a 10% raise.
   8. Use our super-secret toString shortcut to print all of **helen**’s information.

( System.out.println(helen); )

Add **System.out.println();** right after it to add a blank line.

* 1. Use the same coding to print all of **brandon**’s info.
  2. If you run this, you’ll see that the salaries print a lot of decimal places after the raises. Let’s go back to the **toString** method and fix this. (Add this to your notes for future use.)

**String result = "Name: " + name + "\nJob: " + job + "\nSalary: " + String.format("$%,.2f",salary);**

Your output should be:

Helen Lee

45000.0

Brandon Charles

Name: Helen Lee

Job: Accountant

Salary: $51,750.00

Name: Brandon Charles

Job: Analyst

Salary: $52,800.00

## Part 3:

1. Create a Java project called **Lab3B** and a class named **Lab3B**.
2. Create a second new class named **Song**.
3. In the **Song** class:
   1. Add the following private instance variables:
      1. name (String)
      2. artist (String)
      3. songLength (double)
   2. Add a constructor that receives 3 parameters (one for each instance variable) and sets each instance variable equal to the corresponding variable.
   3. Add a public String method named **getName** (no parameter) that returns name.
   4. Add a public double method named **getLength** (no parameter) that returns length.
   5. Add a public String method **toString** (no parameters). It should create a String variable with the song’s instance data with labels. It should then return that string.
4. Back in the main **Lab3B** class.
   1. Declare and instantiate a Song object named **song1** sending the following parameters (“Radioactive”, “Pentatonix”, 4.24)
   2. Declare and instantiate a Song object named **song2** sending the following parameters (“Piano Man”, “Billy Joel”, 3.30)
   3. Declare and instantiate a Song object named **song3** sending the following parameters (“Imagine”, “Pentatonix”, 3.51)
   4. Pretend you don’t know the song lengths and write the statements to find the song with the longest length (using the getLength() method). Print your results with the name and length.
   5. Print the data from each song using the toString method.