**LAB #3 –Using Inheritance and Polymorphism**

**Due Date: Week 8**

**Value: 7%**

**Maximum points: 21 points**

Purpose: The purpose of this Lab assignment is to:

1. Practice the use of Inheritance
2. Practice the use of Polymorphism.

References: Read the Chapters #9, 10 . That material provides the necessary information you need to complete the exercises.

Be sure to read the following general instructions carefully:

- This lab should be completed individually by all the students.

- You will have to demonstrate your solution in a scheduled lab session and submitting the code **through dropbox link on eCentennial**.

You must name your Eclipse project according to the following rule:

**YourFullName\_COMP228\_sectionNumber\_LabNumber**

Example: **JohSmith\_COMP228\_002\_Lab3**

Each exercise should be placed in a separate package named *exercise1*, *exercise2*, etc.

Submit your assignment in a **zip file** that is named according to the following rule:

**YourLastName\_COMP228\_sectionNumber\_LabNumber.zip**

Example: **JohSmith\_COMP228\_002\_Lab3.zip**

## Exercise 1: (10 points) – Coffee Shop

Practice with concepts: Abstract methods and classes, Inheritance, Polymorphism, Protected access modifiers, static methods.

Concrete vs abstract. Direct vs indirect, is-a vs has-a, super, override, Object,

Write a Java application that allows user to place an order for tea or coffee.

Use the following concepts for implementation:

1. There must be at least one abstract method (2 points)
2. There must be at least one abstract class (2 points)
3. There must be at least two children classes and one parent class in the inheritance hierarchy (1 points)
4. There must be at least one protected instance variable (1 points)
5. There must be at least one static method (1 points)
6. There should be at least two unit tests (1 point)
7. Display concept of Polymorphism with your code. Provide comments to explain the Polymorphism ( 2 points)

All of the above requirements must be used properly in the code and not just declared in a class somewhere.

The following classes should be implemented: Drink, Coffee, Tea, Driver, CoffeeShopTests

Write a driver class to test this hierarchy. This application should interact with the user in your preferred way (JOptionPane or console). The concept of Polymorphism should be displayed here in the driver class.

Scenario to help display polymorphism (you can create your own scenario): 3 customers entered a coffee shop and placed an order for 1 cup of tea and 2 cups of coffee. The waiter will brew the drinks and deliver them all 3 drinks at the same time. Strore three drinks in an ArrayList called trayWithDrinks. You can pass the trayWithDrinks to a method that will print out the information about each drink in a polymorphic way.

**Exercise #2:** (**11** **points**)

Practice with Interfaces, Polymorphism, Constants

Must implement:

1. Interfaces (5 points)
2. Polymorphism (5 points)
3. Constants (1 points)

All of the above requirements must be used properly in the code and not just declared in a class somewhere.

Using interfaces, as you learned in this chapter, you can specify similar behaviors for possibly disparate classes.

Governments and companies worldwide are becoming increasingly concerned with carbon footprints (annual releases of carbon dioxide into the atmosphere) from buildings burning various types of fuels for heat, vehicles burning fuels for power, and the like.

Many scientists blame these greenhouse gases for the phenomenon called global warming. Create three small classes unrelated by inheritance—classes Building, Car and Bicycle. Give each class some unique appropriate attributes and behaviors that it does not have in common with other classes.

Write an interface CarbonFootprint with a getCarbonFootprint method. Have each of your classes implement that interface, so that its getCarbonFootprint method calculates an appropriate carbon footprint for that class (check out a few websites that explain how to calculate carbon footprints).

Write an application that creates objects of each of the three classes, places references to those objects in ArrayList<CarbonFootprint>, then iterates through the ArrayList, polymorphically invoking each object’s getCarbonFootprint method. For each object, print some identifying information and the object’s carbon footprint.

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Description** | **%** |
| Functionality | having everything working as per requirements described in each assignment | 45% |
| Implementation | having structures that we covered in the respective chapters for this lab | 40% |
| Validation, etc… | Required validations. Comments, correct naming of variables, methods, classes, etc. | 10% |
| User Friendliness and Innovative features | The flow is logical and intuitive for the user. New additional validations / structures / GUI / functionality that we haven’t covered in class yet. | 5% |
|  | **Total** | **100%** |