CSC262 – Week3 Assignment

Due on: Tuesday April 30, 2019

Chapter 9: (20 points)

1- Use inheritance to implement the following classes:

A: A **Car** that is a **Vehicle** and has a name, a max_speed value and an instance variable called the number_of_cylinders in its engine. Add public methods to set and get the values of these variables. When a car is printed (using the toString method), its name, max_speed and number_of_cylinders are shown.

B: An **Airplane** that is also a **Vehicle** and has a name, a max_speed value and an instance variable called the number_of_engines it has. Add public methods to set and get the values of these variables. When an airplane is printed (using the toString method), its name, max_speed and number_of_engines are shown.

C: The **Vehicle** class that you create should have a constructor that receives values for its instance variables as parameters.

D: Write a VehicleDemo.java class that does the following:

1- Creates an instance of a Car and an Airplane class.
2- Assign values to the name, speed, number_of_cylinders (for the Car object) and number_of_engines (for the Airplane object) variables.
3- Compares which vehicle goes faster and prints the result.
4- Prints the instances of the car and airplane classes.

Chapter 10: (30 points)

2- Make new copies of the .java files that you created for the above problem and add the following changes to them:

A: Make the **Vehicle** an abstract class and add the following abstract method to it:
double runningCost(int hour); Which receives the hours of operation as a parameter and returns the running cost of the vehicle. The **Car** and **Airplane** classes will implement this method as follows:

1- For the **Car** class, define a private double constant called **COST_PER_CYLINDER_PER_HOUR** = 10.5. The runningCost of a **Car** will be equal to:
hours * COST_PER_CYLINDER_PER_HOUR * number_of_cylinders
2- For the **Airplane** class, define a private double constant called **COST_PER_ENGINE_PER_HOUR** = 25.3. The runningCost of an **Airplane** will be equal to:
hours * COST_PER_ ENGINE _PER_HOUR * number_of_engines

B: Write an interface called **maintainable**, which has the following method:
double maintenanceCost(double costPerUnit); It receives the cost per unit of an engine or cylinder and returns the maintenance cost. The **Car** and **Airplane** classes will implement this interface as follows:

1- For the **Car** class, the maintenance cost of a **Car** will be equal to:
   costPerUnit * number_of_cylinders
2- For the **Airplane** class, the maintenance cost of an **Airplane** will be equal to:
   costPerUnit * number_of_engines

C: Add a **String toString()** method to the **Vehicle** class. When a **Vehicle** is printed, its name and max_speed are shown. Also, the **String toString()** methods of the Car and Airplane classes will show the following:

1- When a car is printed, its name, max_speed, number_of_cylinders and **COST_PER_CYLINDER_PER_HOUR** are shown.
2- When an airplane is printed, its name, max_speed, number_of_engines and **COST_PER_ ENGINE _PER_HOUR** are shown.

D: Write a VehicleDemo.java class that does the following:

1- Creates an instance of a Car and an instance of an Airplane class.
2- Assigns values to the name, max_speed and number_of_cylinders instance variables of the Car object and the name, max_speed and number_of_engines instance variables of the Airplane object.
3- Calls all the methods of the Car and Airplane objects that return the values of their instance variables and prints the results.
4- Calls the **runningCost(5)** of the Car and Airplane objects and prints the result.
5- Calls the **maintenanceCost (30.0)** of the Car and **maintenanceCost (250.0)** of the Airplane objects and prints the result.
6- Defines a reference variable of type **Vehicle** and assigns it to an instance of a **Car** class. Example: **Vehicle v1 = new Car(…);**
7- Prints **v1**. Notice whether the toString() method of the **Vehicle** class is called or the toString() method of the **Car** class. Explain why it has been the case. In Object Oriented paradigm, what is this called?