CE00527-2 Further Object Oriented Programming
Tutorial 1

In this tutorial we will review object-oriented principles by constructing classes to represent a Vehicle, and a class (Showroom) which is an aggregation of all the vehicles in the Car Showroom.

Part 1 – Vehicle Class - Basic Exercise

Design, build and test a class Vehicle to represent a vehicle that is shown in the Car Showroom.

The Vehicle class should have:

- String attributes to represent:
  - the manufacturer
  - the model
  - the name of the customer if it's sold
  - the VIN (vehicle identification number) of the vehicle (both letters and numbers)
  - the date of manufacture of the vehicle
  - the date it was sold (assume it is set to null initially)
- A boolean attribute to indicate whether it is has been sold (assume it is set to false initially)
- A char attribute to represent the tax band (A-M)
- An attribute representing the cost of the vehicle.

You should provide one or more appropriate constructor(s), methods to return the value of each attribute, a method to buy a vehicle, and a toString() method that returns relevant information about the vehicle as a String. The buy method should set the date sold and name of the customer attribute, and change the sold boolean attribute to **true**. Also include a method to return the CO2 emitted according to the tax band as a String - see below – include this in the toString() method.

Band	A	B	C	D	E	F	G
CO2	0-100	101-110	111-120	121-130	131-140	141-150	151-160

Make sure that you draw the UML class diagram before coding the class. When testing your class you should show the correct output from at least 2 bands.

Part 2 – Showroom class – a collection of vehicles - Basic Exercise

Design, build and test a class Showroom to manage the vehicles in the Showroom. You should be able to add a new vehicle to the end of the list, find a vehicle given its VIN, and display the details of all the vehicles. Make sure that you draw the UML diagram before coding the class; this diagram should show the relationship between the Showroom and Vehicle classes. Use an ArrayList to hold the Vehicle objects.

Part 3 –Adding more functionality to the Showroom Class - Advanced Exercise

Now modify your Showroom class to support the concept of a current vehicle; the current vehicle is the one that we are currently interested in. Add methods to return the current vehicle object, move on to the next vehicle and move to the previous vehicle.

Ensure that if the current vehicle is the first in the list then an attempt to move to the previous vehicle does not cause problems. Similarly ensure that an attempt to move to the next vehicle does not cause a problem if the current vehicle is the last in the list.

Modify your method to add a vehicle so that it adds the new vehicle at the position immediately after the current vehicle.

Part 4 – ShowroomDriver class - Basic Exercise

This class contains the main method. The class should construct an instance of the Showroom class and 4 instances of the Vehicle class which are unsold. Display the vehicles, then buy a couple of vehicles and display them again. The methods in the Showroom class should all be tested – which will lead to the testing of the methods in the Vehicle class.

You should put in your portfolio:

- **code listing for your final version the Vehicle class**
- **code listing for your final version the Showroom class**
- **code listing for your final version the ShowroomDriver class which tests each of these classes, with output (no user interaction is required, just hard-code the test data)**
- **a UML class diagram showing attributes, methods, and the relationship between the Vehicle and Showroom classes. This diagram should be saved as an image file which can be uploaded if required in blackboard**

Please note that the purpose of this exercise is to give you practice in OO Java programming, not to create a real Vehicle Showroom system! Please ensure your code conforms exactly to the description given.