

# 1. Vehicle Showroom Management System

## Introduction

In this assignment, we will make a command line system for a vehicle showroom.

## Requirements/Features

- Add any type of vehicle in showroom.
- Remove any of the vehicles from showroom.
- Show the vehicle lists with details currently have in showroom.
- Show the list of vehicles with current expected visitor count

## Dependency/Tools used

- Language: **Java**
- IDE: **Eclipse**
- Framework/Concept: **OOP**

## Implementation

In this section, I will describe the whole implementation including all the classes, their attributes and methods.

I have used a total number of five classes (combining both super and subclasses). Details are available in Tables 1-5.

Table-1: Details of CarShowroom class

Class-1	
Class Name: CarShowroom	
Attributes	
Attribute Name	Type
<code>normalVehicleList</code>	<i>ArrayList of Normal Vehicles</i>
<code>sportsVehicleList</code>	<i>ArrayList of Sports Vehicles</i>
<code>heavyVehicleList</code>	<i>ArrayList of Heavy Vehicles</i>
<code>expectedVistorCount</code>	<i>int</i>
<code>visitorIncreaseBySports</code>	<i>int</i>
Methods	
Methods Name	Purpose
<code>addNormalVehicle()</code>	<i>adding normal cars in the list</i>

<code>addSportsVehicle()</code>	<i>adding normal cars in the list</i>
<code>addHeavyVehicle()</code>	<i>adding normal cars in the list</i>
<code>removeNormalVehicle()</code>	<i>Removing normal cars from the list</i>
<code>removeSportsVehicle()</code>	<i>Removing sports cars from the list</i>
<code>removeHeavyVehicle()</code>	<i>Removing heavy cars from the list</i>
<code>showAllVehicleList()</code>	<i>Showing all vehicles of the showroom</i>
<code>showNormalVehicleList()</code>	<i>Showing normal vehicles of the showroom</i>
<code>showSportsVehicleList()</code>	<i>Showing sports vehicles of the showroom</i>
<code>showHeavyVehicleList()</code>	<i>Showing heavy vehicles of the showroom</i>

Table-2: Details of Vehicle class

Class-2	
Class Name: Vehicle	
Attributes	
Attribute Name	Type
<code>modelNumber</code>	<i>String</i>
<code>engineType</code>	<i>String</i>
<code>enginePower</code>	<i>double</i>
<code>tireSize</code>	<i>double</i>
Methods	
Methods Name	Purpose
<code>printVehicle()</code>	<i>Printing details of a given vehicle</i>

Table-3: Details of NormalVehicle class

Class-3
Class Name: NormalVehicle
Type: Subclass (Extends Vehicle)
Attributes
<i>No extra attributes than the superclass</i>

Methods
<i>No extra or overridden methods</i>

Table-4: Details of SportsVehicle class

Class-4	
Class Name: SportsVehicle	
Type: Subclass (Extends Vehicle)	
Attributes	
Attribute Name	Type
turbo	String
Methods	
Methods Name	Purpose
printVehicle()	Overridden to print extra information of sports car

Table-5: Details of HeavyVehicle class

Class-5	
Class Name: HeavyVehicle	
Type: Subclass (Extends Vehicle)	
Attributes	
Attribute Name	Type
weight	int
Methods	
Methods Name	Purpose
printVehicle()	Overridden to print extra information of heavy vehicle

Now, in the main function, I have written all the codes in an infinite loop to give the user a friendly environment to use all the features i.e. adding vehicle, removing vehicle, showing vehicle list and expected visitor count on a continuous basis.

At first, I have created an object of the CarShowroom. I initialized the the `expectedVistorCount` and `visitorIncreaseBySports` number after adding a sports car as 30 and 20 respectively (as given in the assignment). **However, I did not hard code them for the sake of flexibility.** As soon as the program will begin, a message will prompted asking the user to select the desired option (**'1' for adding, '2' for removing, '3' for showing vehicles, '4' for showing expected visitor count**).

If the user chooses to add/remove any vehicle, he/she will further be asked which type of vehicle will be added/removed (**'N' for normal, '2' for sports, '3' for heavy vehicle**).

Everyime the user adds a sports car, the expected visitor count gets increased by `visitorIncreaseBySports` (=20). This is handled in the method called `showSportslVehicleList()`.

For removing, the user can provide the model number, and my system will search all the vehicles and remove the ones of such model. **This ensures further flexibility to the users.**

### Coding Practices Followed

- a. Hard coding is avoided as much as possible.
- b. Private types of variables and Getter/Setter methods are applied appropriately to ensure security and flexibility for further modifications.
- c. Codes are well-documented.

### Desired Clients

Various vehicle showrooms and rent-a-car services.

### GitHub Repo

The source code is publicly available at <https://github.com/TanvirIstiak60/CarShowroom>