****

**SCHOOL OF INFORMATION SCIENCES**

**B.Sc (Informatics) Year 3 Semester 2**

**INF 332 : OBJECT ORIENTED PROGRAMMING**

**CAT**

PATRICK MUCHERU

IF/45/14

**A JAVA PROGRAM THAT INVOLVES A CAR AND ITS ENGINE**

CODING STANDARDS

* Proper Naming Conventions : The program uses the industry standard naming convention. **P**ascal**N**aming**C**onvention for class names and **c**amel**C**ase naming convention for variable names. The variables are named according to the usage for ease of reading the code. Variable and method names are pronounceable and easy to understand. These names are meant to be self-explanatory and have meaningful distinctions from other named variables and methods. Constants have not been used at all within the source code
* Use of comments : Comments are used within the code to properly label code block features and their respective functions. This enables ease of reading of the code by other users.
* Use of Double instead of Float data types to offer more precision of mathematical operations.
* Indentation : Code indentation is adhered within the source code programs as per best coding practices and industry standard practice.
* Class members have public scope and can be accessed from other classes
* The source code has not used redundant initialization within the constructors.
* Only one for loop has been used for iteration purposes within the array that holded the car objects created by the user.

DESIGN PROCESS

Before I designed the system I first had to come up with a template for the entire program.

The main Design Element used on the program from the user’s end enables the user to choose from four different choices of creating a new car, deleting an existing car, retrieving an existing car from Array memory and finally Editing the car’s main features. To enable this I used a for loop to iterate the entire process for at most 25 entries into the array. However the array has no upper limit. The other main feature I employed was the Switch statements with four cases for the four main features of the program. If the user types in an invalid argument, the default statement alerts the user of an invalid entry.

The Program Source code imports ArrayList class for use with storing and manipulating object data as well as the Scanner class to collect user input from the keyboard to enable interactivity between the user and the program.

The following are the features of the main program in detail showing the specific aspects of the program as well as the design elements employed.

* I employed Inheritance by creating a Car class and a subclass of class Car called SuperCar.
* Create an instance of an object of class Car by inputting the car’s name, engine capacity/engine displacement of the car, the Fuel level/fuel capacity of the car and the car’s mileage.
* Once the car object is instantiated, the car starts automatically and requests the user to input the car’s average velocity in kilometres per hour as well as the time the car will be travelling in hours.

The car calculates the distance travelled by multiplying the velocity and the time travelled. The car’s engine adds this distance to its mileage and the mileage is updated in the stored array.

To calculate the fuel economy of the vehicle, the program creates an assumption that the fuel economy of the vehicle is equivalent to the car’s engine capacity multiplied by 4.

The programs outputs the results of running the car over a user specified time and a user specified speed and outputs to the user’s console the car’s index number on the array, name, engine displacement, mileage after run in kilometres , initial fuel volume in litres, fuel economy in litres per 100 kilometres, the volume of fuel used and finally the deficit volume of fuel left in the tank after the run.

* The second feature of the program enables the user to delete a previous entry of objects for class Car by entering the car number which is displayed after an Car object is instantiated and automatically started by start() method. However since we are using an array and objects deleted from the array let the other objects into the vacant space, one may require a unique identifier like a name or an id which may require a database, retrieving and deleting entries may require a bit of indexing logic.
* The third feature of the program enables the user to retrieve a Car object by using the array position and this feature outputs the car’s original number, the name, mileage, engine capacity/displacement and the amount of fuel left within the vehicle’s tank.
* The fourth and final feature enables the user to edit the elements within the specific car object the user may require to edit. This feature fetched the user’s specified Car object from the ArrayList and the following features enable the user to edit the Car’s name, Engine Displacement/ Capacity and the car’s mileage.
* The Engine class is abstracted by the Car class. The only way to instantiate an Engine object is by creating a new Car object.

CONSTRAINTS ENCOUNTERED

The main challenges that I encountered while designing and implementing the program was data manipulation of the objects within the array. I implemented an array together with an ArrayList because of functionality purposes. I needed to first implement my various ways to manipulate the data using the array but while on the way, I realised that it would be much easier to retrieve information about the Car objects from an Array List. It took a few hours to get through the various methods I would use from the imported ArrayList class from the java libraries but eventually I figure it would not cause any problem to have both an array and an ArrayList.

When I first used the for loop, I used it within the switch statements but I soon realised that I would only be iterating within the loop and I would not be able to access the other main features of the program. I edited out the for loop and used the switch statements within the for loop without disrupting the overall functionality of the main aspects of the program. Using the switch statements within the for loop enabled easier iteration as the main user menu will be displayed once the previous iteration is finished.

The switch statement needed a catch method however I was able to use the default case for exceptions that may result from invalid entries.

Developing the Assumption used for fuel economy was an average mathematical operation of common fuel economy figures from car review website [autotrader.com](https://www.autotrader.com/)

The coding process involved using web based IDE [Online Java - IDE, Code Editor, Compiler](https://www.online-java.com/). It proved somewhat troublesome to create different java class files to run the program however I used one java file with one main method to combat this challenge. After a few trials I realised that I could only run the program as one file by refusing to declare the access modifier of the classes.

While coding the constructor, I realised I needed to declare all the variables added within the brackets as well as adding the this keyword in case the fields are overshadowed by the constructor parameter.

RATIONALE FOR CHOICES MADE

Creating the java program for a car and its engine was intriguing to me as i wanted to learn how to effectively use abstraction in Java. It seemed easier to accomplish considering it would need less time to work on and also that cars are interesting engineering technologies.   
The choice I made to use abstraction was because the user does not necessarily need to create a new Engine object separately as they could easily create a Car object which could in itself instantiate its own engine without involving the user while only taking the user’s desired engine capacity.

EVIDENCE OF VALIDATION

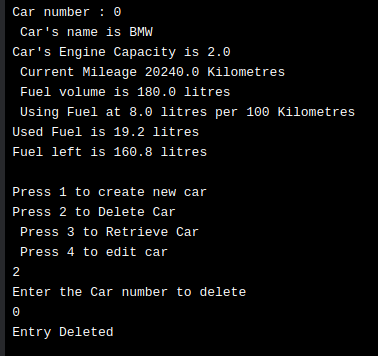
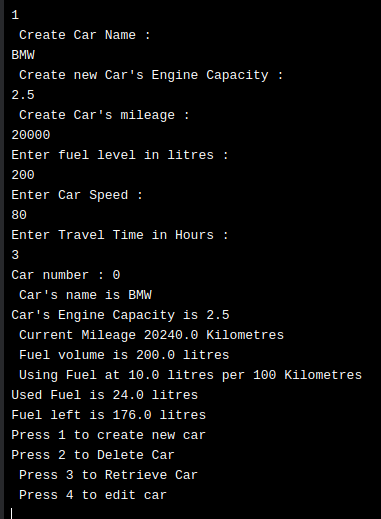


Image 1 : Create Car object Image 2 : Delete Car object Entry

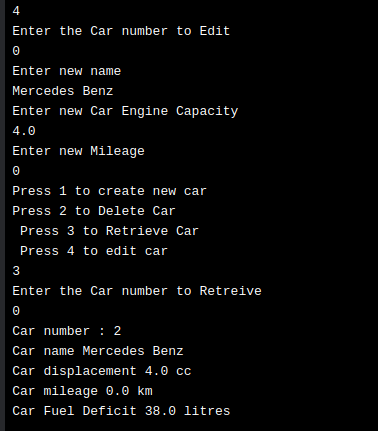
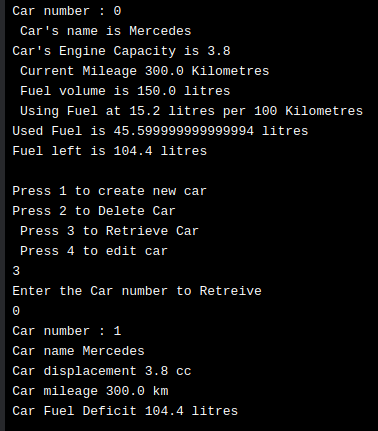


Image 3 : Retrieve Car object Image 4 : Edit Car object Entry