**Java 1: Assignment #4**

Due: TBD

Please remember to read over the assignment rubric before answering these questions. Each question should be written in its own file for upload. When all questions are answered place the source code for each question in a folder, zip it up, then upload it to the assignment 4 upload area on the LMS. If there are any concerns, let me know right away. Feel free to work with others to solve the questions but please code them on your own.

**Question 1: [100 marks]**

**(This question alone accounts for all the marks of Assignment 4)**

Bugatti is a car manufacturer, which produces super cars. You are tasked to write a program which will allow a user to enter a given car type either being a gas car or diesel car. Before writing any classes create three separate packages in a new project. The packages should be called: **CarModels**, **EngineModels**, and **View**. The program specifications are as follows:

* An abstract class named **Engine**, which has the following details: (Place this class in the EngineModels package)
  + A number of spark plugs. Default = 4
  + A number of valves. Default = 4
  + A number of pistons. Default = 4
  + A default constructor with the following details:
    - No parameters.
    - The constructor should redirect to the other constructor mentioned below with the defaults specified.
  + A constructor accepting all the properties of the **Engine** class as parameters.
  + Getters for all the data members of the class, which are protected.
  + A public abstract method, which does not return a value, named engineUtilization.
  + An overridden to string method displaying all the properties of the **Engine** class.
* A class named **GasEngine**, which extends the **Engine** class and has the following details: (Place this class in the EngineModels package)
  + A default constructor with the following details:
    - No parameters.
    - Calls the super class constructor directly.
  + A constructor accepting all the properties of the **Engine** class as parameters. This constructor must pass these values to the super class constructor.
  + The overwritten engine utilization method should display the following the user:
    - Intake stroke – fuel is mixed with air.
    - Compression stroke – piston goes up, mixture of fuel and air is compressed.
    - Ignition stroke – fuel/air is ignited through the use of a spark plug.
    - Exhaust stroke – piston goes up, pushes exhaust through the exhaust valve.
  + An overridden equal’s method implemented using all the super class’s data properties in order to check if two gas engines are equal.
* A class named **DieselEngine**, which extends the **Engine** class and has the following details: (Place this class in the EngineModels package)
  + A default constructor with the following details:
    - No parameters.
    - Calls the super class constructor directly.
  + A constructor accepting all the properties of the **Engine** class as parameters. This constructor must pass these values to the super class constructor.
  + The overwritten engine utilization method should display the following the user:
    - Intake stroke – intake valve opens, air in, piston goes down.
    - Compression stroke – piston goes up, air compressed (heated in excess of 540°C).
    - ICombustion stroke – fuel is injected (right time), ignition, piston goes down.
    - Exhaust – piston goes up, pushes exhaust through the exhaust valve.
  + An overridden equal’s method implemented using all the super class’s data properties in order to check if two diesel engines are equal.
* An abstract class named **Car**, which has the following details: (Place this class in the CarModels package)
  + A name. Default = “Test Car”
  + A color. Default = “White”
  + A number of doors. Default = 4
  + A number of wheels. Default = 4
  + The wheel size in inches. Default = 20
  + The car’s top speed in km/h. Default = 100
  + The engine type coming from the abstract **Engine** class.
  + A default constructor with the following details:
    - The default engine will come from the default constructor of the **GasCar** class or the **DieselCar** class. Thus, the constructor should have one parameter of the **Engine** class.
    - The constructor should also redirect to the other constructor mentioned below with the defaults specified.
  + A constructor accepting all the properties of the **Car** class as parameters.
  + Getters and setters.
    - All properties should have getter methods, which are final.
    - Only name, color, and wheel size in inches should have setter methods, which are also final.
  + An overridden to string method displaying all the properties of the **Car** class.
  + Lastly, this class should implement the Comparable<Car> interface allowing the compare to method to be overridden.
    - This method should compare a car based on its top speed.
    - To understand additional details of how to do this perform some research. It is a lot easier than you may think.
* A class named **GasCar** which extends the **Car** class and has the following properties: (Place this class in the CarModels package)
  + A gas tank volume in liters.
  + A default constructor with the following details:
    - One parameter accepting the gas tank volume in liters.
    - Calls the super class default constructor passing in a new **GasEngine** object with no parameters specified for the gas Engine.
    - Set the data member of the **GasCar** class.
  + A constructor accepting all the properties of the **GasCar** class and **Car** class as parameters.
    - This constructor should pass the relevant information to the super class.
    - Set the data member of the **GasCar** class.
  + A getter and setter for the gas tank liters volume.
  + An overridden equal’s method implemented using the following properties:
    - The car’s name.
    - The number of doors.
    - The number of wheels.
    - The engine.
  + An overridden to string method displaying all the properties of the **GasCar** class.
* A class named **DieselCar** which extends the **Car** class and has the following properties: (Place this class in the CarModels package)
  + A gas tank volume in liters.
  + A default constructor with the following details:
    - One parameter accepting the gas tank volume in liters.
    - Calls the super class default constructor passing in a new **DieselEngine** object with no parameters specified for the gas Engine.
    - Set the data member of the **GasCar** class.
  + A constructor accepting all the properties of the **DieselCar** class and **Car** class as parameters.
    - This constructor should pass the relevant information to the super class.
    - Set the data member of the **GasCar** class.
  + A getter and setter for the gas tank liters volume.
  + An overridden equal’s method implemented using the following properties:
    - The car’s name.
    - The number of doors.
    - The number of wheels.
    - The engine.
  + An overridden to string method displaying all the properties of the **DieselCar** class.
* A class named Bugatti which has the following properties: (Place this class in the View package)
  + An array list of gas cars.
  + An array list of diesel cars.
  + A default constructor with the following details:
    - No parameters.
    - Creates new empty array list objects for the gas cars array list and the diesel cars array list.
  + A method named add gas car with the following details:
    - Accepts a GasCar as a parameter.
    - The method will then add the incoming gas car to the gas car array list.
    - This method does not return a value.
  + A method named add diesel car with the following details:
    - Accepts a DieselCar as a parameter.
    - The method will then add the incoming gas car to the diesel car array list.
    - This method does not return a value.
  + A method named remove gas car with the following details:
    - Accepts a GasCar as a parameter.
    - The method will then remove the incoming gas car from the gas car array list.
    - This method does not return a value.
  + A method named remove diesel car with the following details:
    - Accepts a DieselCar as a parameter.
    - The method will then remove the incoming gas car from the diesel car array list.
    - This method does not return a value.
  + A method named display gas car list, which simply displays the details of all the gas cars.
  + A method named display diesel car list, which simply displays the details of all the diesel cars.
  + A method named display all cars by top speed.
    - This method should make a new array list of the cars type and adds all the gas cars and diesel cars to this list.
    - Next the method should create a normal array of the car type adding all the cars from the array list of the cars type.
    - Lastly, the method should then use the Arrays.sort method to sort the array and finally display the sorted array to the user.
* A class named Question3 which has the following properties: (Place this class in the View package)
  + Just the main method which:
    - Creates a default gas car and diesel car.
    - Create a gas car and diesel car with parameters.
    - Displays each engine type utilizations from the instances of the default cars created.
    - Creates a new **Bugatti** object which be used to add the created gas cars and diesel cars to.
    - After the four cars have been added, call the display methods in the **Bugatti** object for the gas cars, diesel cars, and displaying the cars by their top speed.
    - Next, the remove methods should be called from the **Bugatti** object removing only one of each cars, doesn’t matter which.
      * Note, do not pass the reference of the already created cars, pass in a new car of each type to the remove method with duplicated parameters to demonstrate the method is removing the cars based on their data properties and not their reference.
    - Lastly, after the cars have been removed call all the display methods once again.

Here is an example of the output:



