Car Warehouse Database System

For this assignment, you are required to write a program that implements a simple Car Warehouse Database System. The aim of the Car Warehouse Database System is for a warehouse to keep a database of the car, and to perform operations such as searching for cars, adding/deleting cars, displaying the details of cars, etc.

The Car Warehouse Database System should provide the following features:

- maintains a list (using a Java Collection class) of Car objects
  - each Car object represents a single "car"
  - each "car" should contain the following information:
    - Registration Number (e.g. 8RT2WT)
    - Year Made (e.g. 2013)
    - Colour (e.g. white). A car can have up to 3 different colours.
    - Car Make (e.g. Toyota)
    - Car Model (e.g. Corolla)
    - Price (e.g. $15000)
  - cars can be added/deleted from the list
- lists the details of an existing car (or cars)
- produces a report of cars based on some criteria
- loads a list of cars from a text file
- saves the list of current cars to a text file

Program Logic

When the Car Warehouse Database System starts, it should automatically load a text file called "cars.txt" which contains details of all cars currently kept by the warehouse. The actual format of this text file is described later in this document. The data loaded should be stored in some appropriate data structures. No other reading from or writing to file is required while the program is in operation, until the user chooses to exit, at which point the program saves all the data in memory back to the same text file (cars.txt). In other words, the file I/O operations are performed automatically by the program, and require no direct interactions with the user.

When the program is running, it should repeatedly display the main menu with these options:

(1) Search Cars
(2) Add Car
(3) Delete Car
(4) Exit System

- Option (1) of the main menu allows the user to search for cars in the database. The user should be asked what criteria(s) he/she wants to use to search for cars. The following sub-menu and options should be displayed:

Car Searching Options:
(1) By Registration Number
(2) By Car Make and Car Model
(3) Back to Main Menu

Inputs other than 1-3 should be rejected, and an error message should be printed. The menu should be displayed repeatedly, until the user chooses Option (3).

- If the user chooses option (1), you should ask the user to input a registration number and then search the database for any car with the specific car registration number. Display all information of this car if found, or display “No such car with this Registration Number” if not found.

- If the user chooses option (2), you should ask the user to input a Car Make first and then ask the user to input a Car Model. The user can input a Car Model (e.g. “Corolla”) or “ANY”. If “ANY” is inputted, the program should display all car models of this Car Make from the database. Display all information of cars with this Car Make and this Car Model (or all Car Models if “ANY” is inputted) if found, or display “No such car with this Car Make and Car Model” if not found.

- If the user chooses option (3), the program should go back to the main menu.

➢ Option (2) of the main menu allows the user to add a new car to the database.

➢ Option (3) of the main menu allows the user to delete an existing car from the database. The user should be asked to search the car by Registration Number.

➢ Option (4) of the main menu exits the program. All the cars currently in memory are automatically saved back to "cars.txt".

➢ Inputs other than 1-4 to this main menu should be rejected, and an error message should be printed. The menu should be displayed repeatedly, until the user chooses Option (4).

Input File Format

The input data file (cars.txt) has the following format for each line: (Note that there is no empty space in between a word and a comma). If a car has less than 3 colours, the corresponding fields are empty.

CarReg,YearMade,Colour1,Colour2,Colour3,CarMake,CarModel,Price

- CarReg is a String with no empty space inside.
- YearMade is a positive integer.
- Colour1/Colour2/Colour3 are Strings (may contain empty space inside).
- CarMake is a String with no empty space inside.
- CarModel is a String with no empty space inside.
- Price is a positive integer.

Here is an example of a cars.txt file:

1YX98J,2014,Black,White,,Toyota,RAV4,25000
ABC132,2007,Yellow,,,Toyota,Corolla,5900
MARY,2017,Dark Grey,Blue,Red,BMW,116d,55132
38000,2012,Blue,Light Green,,Honda,Civic,9200

Important Assumptions

You should observe the following assumptions when implementing your program:

- All Car Registration Numbers are unique - if a car with the inputted Car Registration Number is already in the database, you cannot add this car and you should generate an error.
- There is no limit to how many cars can be stored.
- When performing searches, all search strings are not case-sensitive (E.g., there is no difference between “178XYZ”, “178xyz” or “178XyZ” for car registration number. There is no difference between “Toyota”, “TOYOTA” or “toyota” for Car Make. There is no difference between “YARIS” or “Yaris” or “yaris” for Car Model. There is no difference between “ANY”, “any” or “Any” for any car model.)
- The data file is always in the correct format - i.e. no need to validate the data when reading it in.
- All on-screen input/output should be formatted in a user-friendly manner. Sensible error messages should be displayed whenever appropriate (e.g. when searching for a car which is not in the database, or when trying to delete a car which does not exist, a “Not found” error message should be displayed.)