CICS-160 Fall 2023

Assignment 4: Using lists in Java

Due on Wednesday November 29

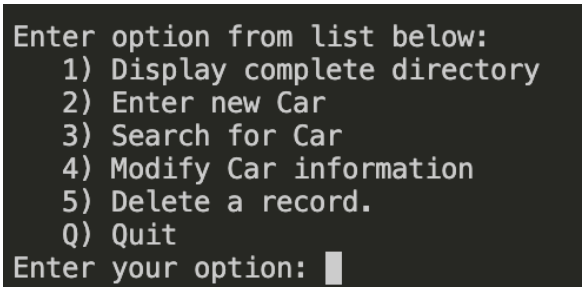
Learning Goals:

This assignment is designed to for you to demonstrate the ability to use lists in Java. We will do that by implementing the system you started to design for assignment #2. It will also give you great practice on dividing a project into classes, delegating operations to specific classes based on information hiding and data encapsulation, and testing.

Overview

For this project, you will be implementing a system that will present a user with a menu-driven system for adding information about people. This will be the system you designed for assignment 2, which is not exactly the same as the one you implemented in assignment 3 (similar, but not identical). If you want, you can use, as a starting point, any idea/concept/code we have used or talked about in class or labs for those classes. Your completed project will have the following classes: Car, GasolineCar, ElectricCar, FleetOfCars, and MainProgram.

The behavior of the system, when completed, will be as follows. The user will be presented with a menu of options from which to choose, each one of them implementing an operation on a list. The operations are: entering a new record; listing all records; displaying a particular record; modifying a particular record; and deleting records. A possible menu might look as displayed in the following image. Your code will start by validating the input entered by the user, asking them to enter another option if the value they enter does not match any of the options presented. All throughout the design, whenever letters are to be entered by the user, the program should behave correctly regardless of that input being uppercase or lowercase. After any of the menu operations is completed, the main menu is displayed again, continuing until the Quit option is selected.



The operations indicated in the menu should be performed by the following methods, using the indicated method names (otherwise the autograder will fail to find them).

All records, be them of type Car, ElectricCar, or GasolineCar, will be stored inside an object of type FleetOfCars. Class FleetOfCars has two attributes: a list of objects of type Car; and a count of how many records are in the list. The list inside FleetOfCars can be either an ArrayList or a LinkedList, your choice.

1. create toString() methods for all classes so that they respond as needed when they are printed. toString() is the Java equivalent to Python's __str__(). The format of the printing can be any that correctly displays all of the attributes of all of the objects stored inside in the FleetOfCars object.

2. Your program will prompt the user for the information to be stored, will create an object of type Car, and will call FleetOfCars.add to store that Car object.

3. Calls FleetOfCars.search(s), which returns an object of type FleetOfCars that contains only those Car objects that have s as their make and model. It then prints the content of the FleetOfCars object that search(s) returns.

4. Asks for the make and model in the record we are trying to modify. Then calls FleetOfCars.search(s), and displays the matching Car objects, one at a time, asking the user if they want to modify that record. If the user answers 'y' or 'Y', gets input from the user and then calls methods of class Car in order to modify the object. Assume that only attributes of class Car need to be modified (i.e. you do not need to provide for the modification of attributes added in ElectricCar or in GasolineCar).

5. Asks for the index of the record to delete. It then displays that index, and asks the user if they want to delete it. If the user answers 'y' or "Y", fleetOfCars.delete(i) is run, which deletes that object from the FleetOfCars object. If the index entered is invalid, prints an error message and deletes nothing.

Once the selected operation has been performed, the menu is re-displayed, and the process continues until the user selects to quit.

Your tasks

Based on the operations listed above, implement the following list of methods for classes Car, ElectricCar, GasolineCar, MainProgram, and FleetOfCars.

Car: (13 points total)

Car(String makeAndModel, int maximumNumberOfPassengers, int numberOfDoors)	5 points
setMakeAndModel(String m)	2 point
getMakeAndModel()→ string	2 point
setMaximumNumberOfPassengers(int m)	1 points
setNumberOfDoors(int m)	manual check
getMaximumNumberOfPassengers()→ int	1 points
getNumberOfDoors()→ int	manual check
toString()→ string	2 points

ElectricCar: (11 points total)

ElectricCar(String makeAndModel, int maximumNumberOfPassengers, int numberOfDoors, double batterySize)	5 points
toString()→ string	3 points
setBatterySize(double b)	2 point
getBatterySize()→ double	1 point

GasolineCar: (11 points total)

GasolineCar(String makeAndModel, int maximumNumberOfPassengers, int numberOfDoors, double gasTankSize	5 points
toString()→ string	3 points
setGasTankSize(double g)	2 points
getGasTankSize()→ double	1 points

FleetOfCars: (25 points total)

FleetOfCars()	1 point
search(String s)→FleetOfCars	15 points
add(Car/GasolineCar/ElectricCar x)	2 points
getSize() → int	2 points
delete(int i)	5 points
get(i) → Car	0 points, but used elsewhere

Things that will be checked manually: (40 points total)

Program has the complete functionality described above	
correct menu operation	10 points
correctly adds records	5 points
correctly searches for records	5 points
correctly modifies records	5 points
correctly deletes records	5 points
Class ElectricCar and class GasolineCar inherit from Car.	5 points
ElectricCar and GasolineCar do not implement things they can inherit from Car.	5 points.

Submit your work in five files: Car.java, ElectricCar.java, GasolineCar.java, MainProgram.java, and FleetOfCars.java.

Extra credit:

Code includes tests for FleetOfCars.delete()	5 points.
Code includes tests for FleetOfCars.add()	5 points