AP Computer Science — Project–Car Dealership

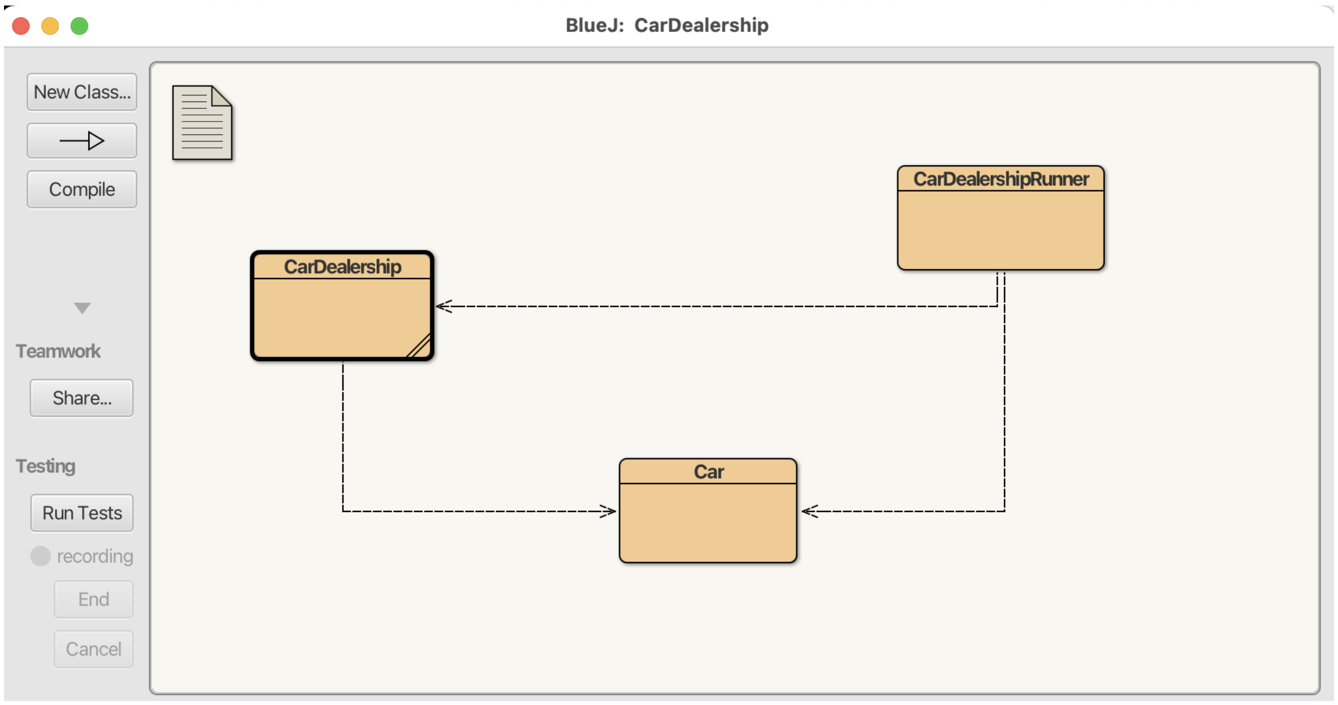
ASSIGNMENT OVERVIEW

In this assignment you'll be creating a small package of files which will model a Car Dealership. The package will include at least three files: `Car.java`, `CarDealership.java`, and `CarDealershipRunner.java`, all written by you, possibly with a partner.

This assignment is worth 50 points and is due on the *crashwhite.polytechnic.org* server at 23:59:59 on the date given in class.

BACKGROUND

A common use for software is managing inventory. A Car Dealership has to manage an inventory of cars, including adding to that inventory, selling cars, tracking sales, etc. This project provides you with the opportunity to practice your `ArrayList` manipulation skills.



This diagram from the BlueJ IDE shows graphically the relationship between the `Car` class, with arrows indicating that it is "used-by" the `CarDealership` and `CarDealershipRunner` classes.
The `CarDealershipRunner` class also uses the `CarDealership` class.

PROGRAM SPECIFICATION

Create a package of Java classes as described here.

1. Write a `Car` class that models a Car by its name, number of miles on the odometer, and selling price. Include:
   a. a constructor with all three of those values as parameters
   b. appropriately named getters for each instance variable
   c. a `setPrice()` method that allows the price of the car to be updated
   d. a `toString()` method that returns a String that allows for convenient printing.

2. Write a `CarDealership` class that models a Car Dealership, including:
   a. A constructor that requires no parameters. The inventory for the dealership will be initialized, but no cars initially added. Additionally, the revenue of the dealership sales should be tracked as cars are sold.
   b. A `toString` method that successively calls the `toString` method on each car in the lot.
   c. An `addInventory` method that accepts a `Car` as a parameter and adds it to the dealership, placing the `Car` at the last position in the inventory of cars.
   d. A `find` method that takes a `Car`'s name as a parameter and finds the position of that car in the inventory.
   e. A `sell` method that accepts an integer and "sells" the car at that position, removing it from the inventory and adding the price of the car to the revenue for the dealership.
   f. A `getRevenue` method that returns the current total revenue for the dealership.
   g. A `discount` method that accepts a mileage amount and a discount percentage as parameters, and goes through all the cars in the inventory that have at least that many miles and reduces the price of those cars by the indicated percentage.
   h. A `swap` method that accepts two integers for positions and swaps the two `Car` objects located at those positions.
   i. A `priceRange` method that accepts two double values (lowest and highest) and returns an `ArrayList` of all `Car` names that are within that price range.

3. A `CarDealershipRunner` class with a main method that demonstrates the capabilities of the `CarDealership` class by creating a `CarDealership` object and manipulating it.

DELIVERABLES

`CarDealership.zip`

This single file will be a zipped directory (folder) of your project. It will include as a minimum the three files listed above along with any other classes you create during the development of your program.

To submit your assignment for grading, copy your file to your directory in `/home/studentID/forInstructor/` at *crashwhite.polytechnic.org* before the deadline.

ASSIGNMENT NOTES

- This project is mostly an implementation challenge. Three classes have been identified for you in the statement of this problem along with specific constructor and method specifications. The methods are representative of the kinds of things a class needs to be able to do.

- It's not uncommon to have a larger class whose role is to manipulate objects contained in an Array or ArrayList. Here, the `CarDealership` class is manipulating `Car` objects, and the methods you write will perform specific tasks on an `ArrayList` of `Car` objects.

- As can be seen from the specifications above, a number of different methods need to be written for the `CarDealership`. It makes sense to write and test these methods one at a time, particularly as some methods might be called by other methods. A standard approach to developing this project would be:
  1. Begin writing the `Car` class.
  2. Begin writing the `CarDealershipRunner` class with a main method that tests the `Car` class.
  3. Continue developing the `Car` class, and continue writing tests in the `CarDealershipRunner` to test those methods.
  4. Once `Car` has been developed, begin writing the `CarDealership` class.
  5. Continue writing methods for the `CarDealershipRunner` class, along with additional tests that demonstrate the effects of those methods.
  6. Continue until all methods for all classes have been written and tested.

  A sample test run is included at the end of this document.

GETTING STARTED

1. With paper and pencil, and possibly in collaboration with a partner, sketch out the basic features of each class, including instance variables, constructors, accessor methods, and mutator methods.

2. Consider writing some pseudocode that you can use to begin implementing those classes.

3. Create a new project in BlueJ, VS Code, or some other IDE that will allow you to manage this assignment.

4. Begin writing following the strategies described above in the Assignment Notes.

5. Once a day or so, archive/zip your project folder and save a backup copy of it on another device or machine: a flash drive, your home folder on the *crashwhite.polytechnic.org* server, etc.

6. When your program is completed (but before the deadline), copy a final archived package (**CarDealership.zip**) to the server as indicated above.

QUESTIONS FOR YOU TO CONSIDER (NOT HAND IN)

1. This project has a "manager" class, `CarDealership`, that has methods to manipulate an ArrayList of objects belonging to the `Car` class. Java already has methods to manipulate an ArrayList of objects directly—`get()`, `set()`, etc.—so why would we go to the trouble of writing a separate `CarDealership` class?

2. The `CarDealership` class itself is run by the `CarDealershipRunner` class via its `main()` method. How many `main` methods is a Java program allowed to have?

SAMPLE TEST RUN

```
Printing out inventory:
CarDealership[inventory=
Car[name=RAV4,milesTravelled=0,price=40000.0],
Car[name=Honda CR-V,milesTravelled=10000,price=30000.0],
Car[name=Ford Bronco,milesTravelled=5000,price=50000.0],
revenue=0.0]

Swapping first and last cars:
CarDealership[inventory=
Car[name=Ford Bronco,milesTravelled=5000,price=50000.0],
Car[name=Honda CR-V,milesTravelled=10000,price=30000.0],
Car[name=RAV4,milesTravelled=0,price=40000.0],
revenue=0.0]

Getting a list of cars w/ prices between 25000 and 40000:
Car[name=Honda CR-V,milesTravelled=10000,price=30000.0]
Car[name=RAV4,milesTravelled=0,price=40000.0]

Discounting cars with >= 8000 miles by 10%:
CarDealership[inventory=
Car[name=Ford Bronco,milesTravelled=5000,price=50000.0],
Car[name=Honda CR-V,milesTravelled=10000,price=27000.0],
Car[name=RAV4,milesTravelled=0,price=40000.0],
revenue=0.0]

Looking for the RAV4 on the lot:
It's location is: 2

Selling the RAV4:

Selling the car at position 0:

Identifying the total revenue for cars sold:
90000.0

Printing out the final inventory and sales:
CarDealership[inventory=
Car[name=Honda CR-V,milesTravelled=10000,price=27000.0],
revenue=90000.0]
```