CS3330 LAB 4, DUE 48 HOURS AFTER YOUR LAB ENDS

Objectives:

- Practice Composition and Aggregation
- ArrayList Data Collection

Submission Info:

Use the online submission system available at https://submit.cs.missouri.edu/app

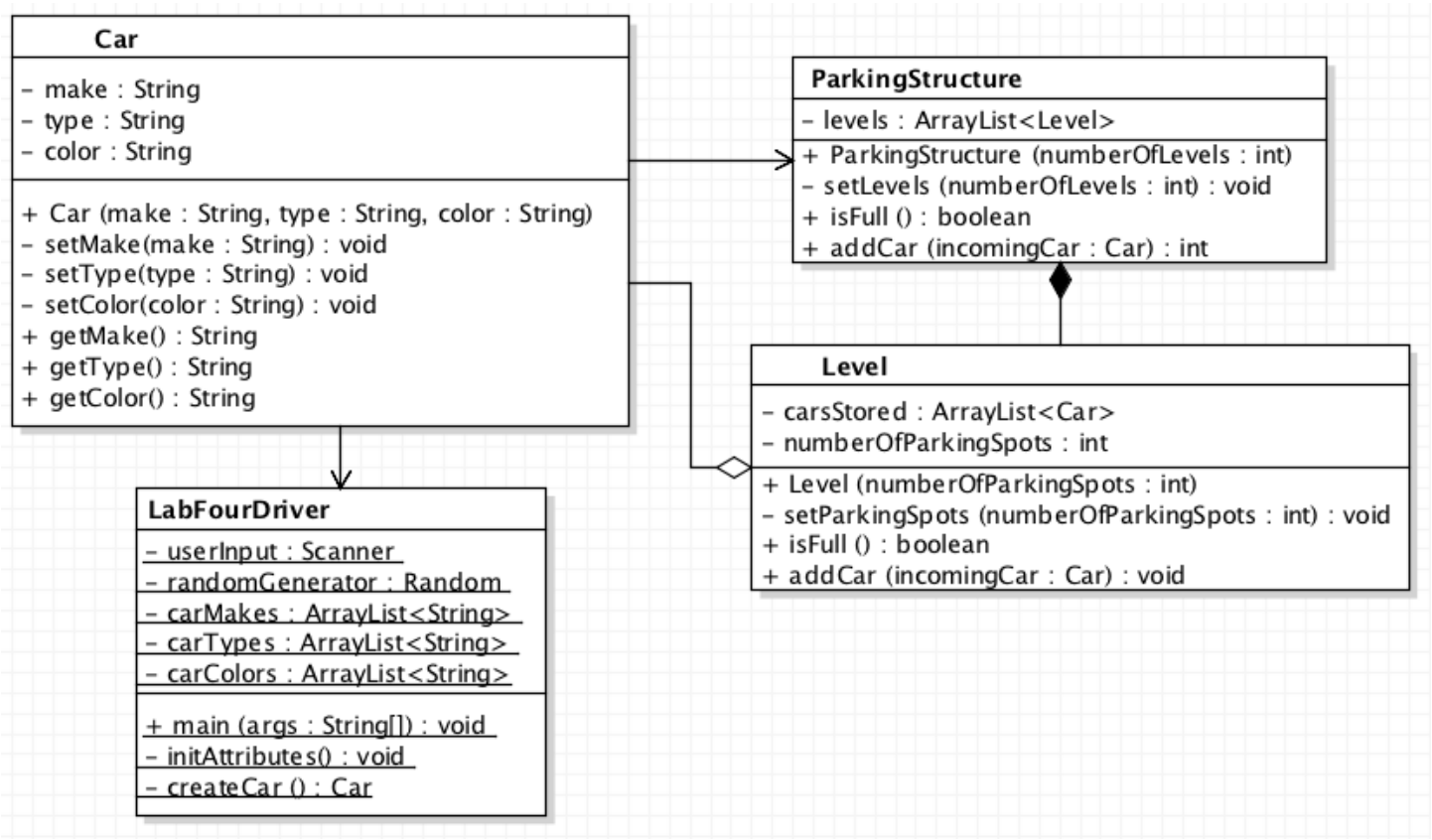
Lab Material:

Download the main method, this LAB 4 document on blackboard. Please copy the contents from the main code given into a LabFourDriver class.

General Lab Info:

You will create 4 separate java classes in a project that you create.

UML CLASS DIAGRAMS:



LabFourDriver.java

initAttributes()

- Assigns the attribute randomGenerator to a new instantiated Random object with a seed of 1337.
- Assigns the userInput attribute to a new instantiated a Scanner Object.
- Assigns the attribute carMakes to a new instantiated ArrayList<String>.
- Manually fill the carMakes ArrayList with the following makes in the order listed:
    - “Ford”, “Toyota”, “Dodge”, “Honda”.
- Assigns the attribute carTypes to a new instantiated ArrayList<String>.
- Manually fill the carTypes ArrayList with the following models in the order listed:
    - “Sedan”, “Truck”, “Van”, “Hatchback”.
- Assigns the attribute carColors to a new instantiated ArrayList<String>.
- Manually fill the carColors ArrayList with the following colors in the order listed:
    - “Blue”, “Green”, “Red”, “Purple”, “Orange”, “Yellow”.

createCar()

Return a newly instantiated car object created with randomly generated make, type, and color. Randomly generate a make from carMakes ArrayList, Randomly generate a type from carTypes ArrayList, and randomly generate a color from carColors ArrayList.

Car.java

Create the constructors, setters, and getters. Use the diagram above to help guide creation of this class.

Level.java

Level(numberOfParkingSpots: int)

Passes the passed parameter numberOfParkingSpots to the setter setParkingSpots.

setParkingSpots(numberOfParkingSpots: int)

Instantiates the carsStored to a new ArrayList<Car>. Assigns the attribute numberOfParkingSpots to the passed in parameter numberOfParkingSpots.

isFull()

Checks if the size of carStored is greater than or equal to the numberOfParkingSpots; returns true, else returns false.

addCar(incomingCar: Car)

Adds the incomingCar Car object to carStored attribute.

ParkingStructure.java

ParkingStructure(numberOfLevels: int)

Call the setter setLevels with the passed parameter numberOfLevels

setLevels(numberOfLevels: int)

Instantiates the levels attribute to a new ArrayList<Level>. Verify that the passed in parameter numberOfLevels is greater than zero. If not, manually add two new Levels to the newly instantiated levels attribute with **10** parking spots at each level . Else, loop through numberOfLevels adding new Levels with **10** parking spots at each level..

Note: *Check the lab lecture on how to create a ArrayList<Level> properly.*

isFull()

Checks to make sure no parking spots is available. If one level does have an open spot; return false, else true.

addCar(incomingCar: Car)

Passes the passed incomingCar object to a level with **an empty spot** and returns the level the car was stored in the parking structure.

PROGRAM OUTPUT (MUST BE EXACTLY AS BELOW -- WE SEEDED THE RANDOM), INPUT IS IN BOLD

Also, output is shown in columns so everything can be seen on one page.

TEST 1

```
Enter number of parking structure levels: 5

Dodge, Sedan, Yellow.
Car Stored at level 1.

Honda, Hatchback, Orange.
Car Stored at level 1.

Honda, Hatchback, Green.
Car Stored at level 1.

Dodge, Hatchback, Green.
Car Stored at level 1.

Ford, Sedan, Purple.
Car Stored at level 1.

Honda, Hatchback, Orange.
Car Stored at level 1.

Dodge, Hatchback, Green.
Car Stored at level 1.

Honda, Truck, Purple.
Car Stored at level 1.

Ford, Sedan, Purple.
Car Stored at level 1.

Toyota, Hatchback, Orange.
Car Stored at level 1.

Ford, Van, Yellow.
Car Stored at level 2.

Toyota, Sedan, Red.
Car Stored at level 2.

Dodge, Hatchback, Red.
Car Stored at level 2.

Dodge, Truck, Purple.
Car Stored at level 2.

Dodge, Van, Purple.
Car Stored at level 2.

Honda, Sedan, Green.
Car Stored at level 2.

Dodge, Hatchback, Blue.
Car Stored at level 2.

Dodge, Hatchback, Blue.
Car Stored at level 2.

Dodge, Truck, Orange.
Car Stored at level 2.

Dodge, Van, Orange.
Car Stored at level 2.

Toyota, Hatchback, Purple.
Car Stored at level 3.

Dodge, Sedan, Green.
Car Stored at level 3.

Toyota, Sedan, Red.
Car Stored at level 3.

Toyota, Hatchback, Blue.
Car Stored at level 3.

Dodge, Sedan, Green.
Car Stored at level 3.

Toyota, Van, Blue.
Car Stored at level 3.

Honda, Hatchback, Yellow.
Car Stored at level 3.

Dodge, Sedan, Yellow.
Car Stored at level 3.

Dodge, Truck, Orange.
Car Stored at level 3.

Dodge, Truck, Orange.
Car Stored at level 3.

Honda, Hatchback, Green.
Car Stored at level 4.

Ford, Sedan, Green.
Car Stored at level 4.

Dodge, Sedan, Yellow.
Car Stored at level 4.

Honda, Hatchback,
Yellow.
Car Stored at level 4.

Ford, Van, Purple.
Car Stored at level 4.

Toyota, Sedan, Red.
Car Stored at level 4.

Dodge, Sedan, Blue.
Car Stored at level 4.

Dodge, Truck, Purple.
Car Stored at level 4.

Toyota, Hatchback,
Purple.
Car Stored at level 4.

Dodge, Hatchback, Red.
Car Stored at level 4.

Ford, Van, Yellow.
Car Stored at level 5.

Honda, Truck, Yellow.
Car Stored at level 5.

Ford, Sedan, Yellow.
Car Stored at level 5.

Honda, Hatchback,
Yellow.
Car Stored at level 5.

Honda, Hatchback, Green.
Car Stored at level 5.

Toyota, Van, Purple.
Car Stored at level 5.

Ford, Hatchback, Green.
Car Stored at level 5.

Ford, Hatchback, Yellow.
Car Stored at level 5.

Toyota, Truck, Red.
Car Stored at level 5.
```

		Ford, Truck, Orange. Car Stored at level 5.

TEST 2

Enter number of parking structure levels: **-3** Dodge, Sedan, Yellow. Car Stored at level 1. Honda, Hatchback, Orange. Car Stored at level 1. Honda, Hatchback, Green. Car Stored at level 1. Dodge, Hatchback, Green. Car Stored at level 1. Ford, Sedan, Purple. Car Stored at level 1. Honda, Hatchback, Orange. Car Stored at level 1. Dodge, Hatchback, Green. Car Stored at level 1. Honda, Truck, Purple. Car Stored at level 1. Ford, Sedan, Purple. Car Stored at level 1. Toyota, Hatchback, Orange. Car Stored at level 1. Ford, Van, Yellow. Car Stored at level 2. Toyota, Sedan, Red. Car Stored at level 2. Dodge, Hatchback, Red. Car Stored at level 2. Dodge, Truck, Purple. Car Stored at level 2. Dodge, Van, Purple. Car Stored at level 2.	Honda, Sedan, Green. Car Stored at level 2. Dodge, Hatchback, Blue. Car Stored at level 2. Dodge, Hatchback, Blue. Car Stored at level 2. Dodge, Truck, Orange. Car Stored at level 2. Dodge, Van, Orange. Car Stored at level 2.

GRADE GUIDE

30 possible points with a possible 5 bonus points extra

If your program does not compile, produce any input/output (I/O) because most of the source code is commented out then your lab will receive a grade of zero points. If your lab has any runtime errors (Such as NullPointerException or ArrayIndexOutOfBounds), the lab will also receive **ZERO** points. **If you don't have header comments inside of ALL OF YOUR CLASS FILES, you will receive ZERO points as well. NO EXCEPTION!!!!!**

Grading Rubric

5 points: Javadoc and Commenting
5 points: Car Class
7 points Total: Level Class
- **2 points:** setParkingSpots
- **3 points:** isFull
- **2 points:** addCar

8 points Total: ParkingStructure Class
- **3 points:** setLevel
- **3 points:** isFull
- **2 points:** addCar

2 points: createCar method
3 points: initAttributes method

BONUS (5 points - ALL OR NOTHING)

Convert all ArrayList<Car> and ArrayList<Level> in your program into HashMap<Car> and HashMap<Level> respectfully. Change the program as necessary to accommodate the new data structure, output is the same as above. **Also, you must explain why a programmer would use a HashMap over an ArrayList in this situation, Big-O Notation comparison is required for each of these Collections (Data Structures) in your explanation. Add your explanation at the top of your LabFourDriver.java in the form of comments.**