First Look at Classes - Continuation

1 Introduction

In the previous lab you have created a Car Dealer Application that uses three car classes (Corolla, Camry, and Sienna) to create instances of those car types. As you probably noticed, Corolla, Camry, and Sienna classes have lots of very similar code among them. This is not good programming practice. If something must be changed in one of those functionalities, you must change in three places, making this action error prone.

To avoid this issue you will implement the concept of Class Inheritance in our project. To implement inheritance, you will create a super class called Car and place all the similar/equal functionality that Corolla, Camry and Sienna classes share in common and move to this Car class. Only keep the functionality that are specific for the type of car, such as setting the car price range, available colors, etc. However, the definition of those variables can be done in the Car class, and in each subclass you can set them to their respective values.

2 Super Class Car

2.1 Creating the Super Class Car

In NetBeans, create a new Java class (name it as "Car") in the same package that you have all the other car classes. Car class will be the super class of the Corolla, Camry, and Sienna classes. There is nothing different when creating a super class: no special setup, they are created in the same way as any class that you have created in this course so far.

2.2 Copying duplicated code into Car Class

Open up at least two car classes, such as Corolla and Camry, and compare their java code implementation. Look for duplicate portions of their code; both class fields (variables) and class methods.

After identifying what is common between those two classes, then move them to the Car Class.

Note: we have talked about public and private access permission in class fields and class methods. Since we want car-specific classes to have direct access to the Car class fields but we don't want other external classes besides the sub-classes to have direct access (such as main applications), then we can use a third access type option: protected. Use protected for the Car class fields.

```
public class Car {
    protected String model;
    protected int[] yearRange;
```

2.3 Modifying each car-specific class

You must do few things now in the Corolla, Camry, and Sienna classes to have them inherit the methods and fields found in the super class Car. Below is a step-by-step on what is needed.

a) At the top of each class, change

```
public class <class name here>
```

to

```
public class <class name here> extends Car
```

b) Delete the duplicated code that has been placed in the Car class already.
c) Modify the three constructors as discussed below:
   - In the default constructor, set all the class fields (remember the class fields are now defined in the Car super class, but setting them will be done in the sub-classes).
     - Examples:

```
model = "Camry";
yearRange = new int[]{2015, 2017};
```

   - For the non-default constructors, first make a call to the default constructor (to set all the class fields values to their default values) and then set values as specified by the non-default constructor input arguments. To call the default class constructor from within a non-default constructor use `this()` as shown below.
     - Example:

```
public Camry(int manufactureYear) {
    this();
    this.manufactureYear = manufactureYear;
}
```

If you did as expected, only code in each car-specific class should be their constructors.

If that's true, congratulations! Now you have a better code implementation, where duplicated code was eliminated. You have an easier code to modify in the future, or extend functionality (new car types), add new functionality to all the cars by just changing the Car class once instead of multiple similar changes in all the sub-classes, etc.

3 Testing your application

Based on the table below perform the following in the main application:

a) Create 3 Corolla car objects (real cars based on the Corolla blueprint class). Print their features in the terminal window.
b) Change one Corolla car to a new price of $18,000 and print again its features to verify if the new price has been accepted.
c) Change the second Corolla price to $1.00 and verify if the new price has been accepted.
d) Try to change the third corolla color to Pink. Should it be allowed to do so?
e) create 2 Camry car objects and play with their features.
f) create 2 Sienna car objects and play with their features..

Table 1. General Information on the Cars sold by the Dealer

	Car Model: Corolla Year Range: 2016 - 2018 Price Range: $18000 - $20000 Available Colors: Black, Blue Accessories: air-conditioning, luxury interior, remote starter Standard model: 2018, $18550, Black, air-conditioning
	Car Model: Camry Year Range: 2015 - 2017 Price Range: $20000 - $22000 Available Colors: Red, White Accessories: power locks, power windows Standard model: 2017, $21000, White, power locks
	Car Model: Sienna Year Range: 2015 - 2018 Price Range: $25000 $30000 Available Colors: Silver, Black Accessories: Media Player, heated-seat Standard model: 2018, $27000, Silver, no accessories