CSE 271 Lab 4 – Java Classes and Constructors
Spring 2022
Assigned: 2/17/2022
Due: 2/20/2022

Introduction:

In this lab, we are going to further hone our Java class making abilities. Specifically, we will create a class with methods, getters and setters, instance properties, and more importantly constructors. In this lab, we will create two classes, a Car.java and CarDriver.java. You will not be provided a driver class for this lab. You must provide JavaDoc for all methods and classes in this lab as well as line comments where needed. Create a project named **Lab4** and also create the Car.java and CarDriver.java classes in this project. **NOTE:** Please read this lab carefully. While it is similar to the previous lab assignment, there are differences you need to make in the Car class! Also note that when working with multiple constructors, do not forget to use the this() keyword to call the workhorse constructor either directly or indirectly.

Class Car:

Create a class named **Car** that has the following private Instance Properties:

- **int yearModel –** The yearModel field is an integer of the car's year. The year cannot be greater than 2022 and smaller than 1885.
- **String make –** The make field is a String of the car's make.
- **String model –** The model field is a String that holds the model of the car.
- **String owner –** The owned field is a String that holds the name of the owner of the car.
- **int speed –** The speed Instance Property is an integer that holds contains the car's current speed in miles per hour (mph). The speed cannot be a negative number. The maximum speed of a car is 250 mph. You have to ensure that the speed is within the valid range when a car accelerates or brakes.
- **double fuelLevel –** A double that holds the current fuel level of the car. This value ranges from 0 to 1.0. You have to make sure the value is always within this.
- **boolean start –** A boolean that is true if the car engine has been turned on and false if it is off.

The **Car** class should contain the following methods:

- **public Car() –** The default (or empty) constructor that initializes the instance properties to the default values (speed to 0, fuel level to 1.0, start to false, year model to 2022, and all strings to "").
- **public Car(String owner, String make, String model, int yearModel) –** A partial constructor that initializes the instance properties using the input parameters. Initialize the speed to 0, start to false, and fuel level to 1.0.

- **public Car(String owner, String make, String model, int yearModel, double fuelLevel) –** A partial constructor that initializes the instance properties using the input parameters. Initialize the speed to 0 and start to false.

- **public Car(String owner, String make, String model, int yearModel, double fuelLevel, int speed, boolean start) –** The workhorse constructor that initializes all instance properties with the input parameters.

- **public Car(Car anotherCar) –** The copy constructor that initializes the instance properties using the values of another Car's instance properties.

- **public boolean accelerate() –** This method increments the car's speed by 4 mph and decreases the car's fuelLevel by 0.05. The car cannot accelerate if the engine is not on. Also, do not accelerate if the car does not have enough fuel (at least 0.05). If the current speed is the maximum (250 mph) then the acceleration will not increase the speed but will decrement the fuel. The method returns true if the car accelerates (i.e., increases the car speed by some amount) and false, otherwise.

- **public boolean brake() –** This method decrements the car's speed by 3 mph. The car must have its engine on in order to apply the brake. The speed cannot be negative as a result of a brake. If you brake when the car's speed is 3 mph or less, then the brake call will reduce the speed to 0 mph. The method returns true if it can apply the brake and reduce the speed by some amount and false, otherwise.

- **public boolean isGasTankEmpty() –** This method returns true if the fuel level of the car is less than 0.05 and false, otherwise.

- **public boolean sameOwner(Car anotherCar) –** This method returns true if the two cars have the same owner and false, otherwise. We assume that each car owner as a unique name.

- **public boolean equals(Object o) –** Override the default Object class equals() method. This method returns true if the two cars have the same make, model, and yearModel and false, otherwise.

- **public String toString() –** Override the default Object class toString(). This method returns a string representation of the car that includes the owner, make, model, yearModel, fuelLevel, and speed of the car. An example return of this method call is as follows. "Owner: Samuel, Make: Toyota, Model: Camry, Year: 2020, Speed: 75, Fuel Level: 0.70".

- **A set of getters and setters for all Instance Properties.** Look at the Week 3 and 4 Lecture Code for examples.

Class CarDriver:

Create a class named **CarDriver** to test your **Car** class. The CarDriver class contains the **main** method to run your program. In this class, you have to test all the public interface methods of the Car class including constructors, getters, and setters. You may create multiple objects of the Car class for testing. When testing, you must make sure that you think about all possible scenarios for each of the methods. For example, when you test the accelerate() method, besides checking to see whether a car can accelerate in a normal scenario, you need to think about what are called edge cases. That is, 1) there is not enough fuel,

2) the speed is already at the maximum, 3) the engine is off, etc. Also, you need to think about scenarios where setter methods must throw an IllegalArgumentException.

Submission Instructions:

After you have completed the lab assignment, locate your source code (**Car.java and CarDriver.java**) in your workspace and submit it to the corresponding Lab 4 assignment's CODE plugin.

Rubric:

Task	Grade
Car	
Declare private Instance Properties	4
Create all constructors as listed in the requirements	15
List of getters correctly implemented	7
List of setters correctly implemented (-3 points if you do not handle invalid inputs with the IllegalArguementException when applicable)	7
Correctly implemented the accelerate() method	7
Correctly implemented the brake() method	7
Correctly implemented the isGasTankEmpty() method	7
Correctly implemented the sameOwner(Car anotherCar) method	7
Correctly implemented the equals() method	7
Correctly implemented the toString() method	7
Driver	
Test constructors	5
Test getters and setters	5
Test the remaining six methods	5
Program contained proper JavaDoc and individual comments where needed for both the Car and CarDriver classes and followed the Miami University coding guidelines	10
Total	**100**