CSE 271: Object-Oriented-Programming
Lab 04: Constructors, Mutator, Tester, Javadoc

Lab Overview:

In this lab, you will practice how to create a class and its methods, specifically, constructor, accessor (getter), and mutator(setter). You will also implement a Tester class with the main method and write Javadoc comments for the classes and methods. Create a project in Eclipse named **Lab_04.** You are going to design two classes (Car.java and TestCar.java) in this project.

Javadoc:

You have to make Javadoc style comments for all methods and classes including parameter and return description. Once you write all the Javadoc style comments generate Javadoc using Eclipse's "Generate Javadoc" option from the "project" menu. Include the Javadoc folder "doc" with your code submission.

Task1: Class Car:

Create a class named **Car** which has the following private instance variables (fields):

- **String owner** : This is a String object that holds the name of the owner of the car.
- **String make** : The make field references a String object that holds the make of the car.
- **String model :** The model field is a String object that holds the model of the car.
- **int yearModel** : The yearModel field is an int that holds the car's year model.
- **float fuelLevel**: A float that holds the current fuel level of the car. The value of fuel level is ranges between 0 and 10.0. You have to make sure the value is always within this range.
- **int speed** : The speed field is an int that holds the car's current speed. The speed cannot be a negative number. The maximum speed a car is 200 mph.

The class should also have the following Constructors:

- **public Car()**
  Initializes the instances variables to the default values (speed to zero, fuel level to 10 and strings to empty string).
- **public Car(String owner, String make, String model, in yearModel)**
  Initializes the instances variables using the received parameters. Initialize speed to zero and fuel level to 10.
- **public Car(String owner, String make, String model, in yearModel, int fuelLevel)**
  Initializes the instances variables using the received parameters. Initialize speed to zero.
- **public Car(String owner, String make, String model, in yearModel, int fuelLevel, int speed)**
  Initializes the instances variables using the received parameters.
- **public Car(Car car1)**
  Initializes the instances variables using car1 object's values.

The class should have the Accessor (getter) and mutator (setter) methods for all instance variables:

Important: For each mutator or setter method you need to throw an IllegalArgumentException if the parameter is not within the valid range of values as described above in the instance variables section.

- public void setOwner(String owner)
- public void setMake(String make)
- public void setModel(String model)
- public void setYearModel(int yearModel)
- public void setFuelLevel(float fuelLevel)
- public void setSpeed(int speed)
- public String getOwner()
- public String getMake()
- public String getModel()
- public int getYearModel()
- public float getFuelLevel()
- public int getSpeed()

Other methods you need to implement in the class Car:

- **public boolean accelerate()**
  The method increments the car's speed by 5 and decreases the fuel level by 0.5. The method does return true if it accelerates the car by 5 and false, otherwise. You have to check the fuel level to see whether it's possible to accelerate with the current fuel level.
- **public void brake()**
  The method decrements the car's speed by 5. The method does not return anything.
- **public boolean isGasTankEmpty()**
  The method returns true if the fuel level of the car is less than 0.5. False, otherwise.
- **public boolean sameOwner(Car car)**
  The method returns true if the two cars have the same owner. False, otherwise.
- **public boolean equals(Car car)**
  The method returns true if the two cars have the same make, model, and yearModel. False, otherwise.
- **public String toString()**
  The method returns a String representing the car object that includes owner, make, model, yearModel, fuelLevel, and speed of the car. When this method is called, it returns a String like "Owner: Robert, Make: Toyota, Model: Camry Year: 2019, Speed: 75, Fuel Level: 8.5".

Task2: Driver Class

Now write another class named **CarDriver** to test your Car class. The CarDriver class has the **main** method. In your CarDriver class you have to test all the methods of the Car class including constructors, accessors and mutators. You can create multiple objects of the Car class for testing. You must test all the methods you defined in Car class. When you test you need to make sure that you think about all possible scenarios for each of the methods. For example, when you test accelerate() method, besides testing to see whether a car can accelerate in a normal scenario, you need to think about edge cases: 1) there is not enough fuel, 2) the speed is already the maximum, 3) the engine is off, etc. Also, you need to think about scenarios where setter methods must throw an *IllegalArgumentException*.

Grading Rubric:

Car	
Declare private fields with appropriate type or class	2
Constructors (each worth 3 points)	15
Accessors (each worth 2 points)	12
Mutators (each worth 2 points)	12
accelerate() method	3
brake() method	3
isGasTankEmpty() method	3
sameOwner(Car car) method	3
equals() method	3
toString() method	3
Test constructors (each worth 1 points)	5
Test accessors and mutators (each worth 1 points)	12
Test rest 6 methods (each worth 1 points)	6
Javadoc style comments (for each method 0.5 points and for the class 0.5)	12
Submitted correct Javadoc files	6
Total	**100**

Important Note:

If you are done with the task within the lab then you need to show your work to one of the instructors present in the lab. Make sure the file name is correct and code is well commented. No late submission. You will get zero for the late submission.

Submission:

Submit java files and Javadoc folder "doc" to the appropriate submission folder on the Canvas by the due time. You can zip all the java files and "doc" folder together and submit one zip file.