Lab Assignment 04, Object-Oriented Programming, CSE 271, Spring 2020
Department of Computer Science and Engineering, Miami University

Constructor, Accessor, Mutator, Driver, Javadoc

In this lab, you will practice how to create a class and its methods, specifically, constructor, accessor and mutator. You will also implement a *Driver* 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 CarDriver.java) in this project.

Javadoc:
You must make Javadoc style comments for all methods and classes including parameter and return description. Follow the commenting style discussed in the lecture. 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.

- **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. The year cannot be greater than 2021 and smaller than 1885.
  - **float fuelLevel**: A float that holds the current fuel level of the car. The value of fuel level ranges between 0 and 1.0. You have to make sure the value is always within this range.
  - **int speed**: The speed field is an integer that holds the car's current speed in mile per hour. The speed cannot be a negative number. The maximum speed of a car is 250 mph. You have to make sure that the speed is within the valid range when a car accelerates or brakes.
  - **boolean start**: A boolean that is true if the car engine has been turned on and false, if it is off.

  The class should also have the following methods:

  - **public Car()**
    The *empty* constructor that initializes the instance variables to the default values (speed to zero, fuel level to 1.0, start to false, year model to 2020 and strings to null).
  - **public Car(String owner, String make, String model, int yearModel)**
    A *partial* constructor that initializes the instance variables using the received parameters. Initialize speed to zero, start to false, and fuel level to 1.0.
  - **public Car(String owner, String make, String model, int yearModel, float fuelLevel)**
    Another *partial* constructor that initializes the instance variables using the received parameters. Initialize speed to zero and start to false.
  - **public Car(String owner, String make, String model, int yearModel, float fuelLevel, int speed, boolean start)**
    The *workhorse* constructor that initializes the instance variables using the received parameters.
  - **public Car(Car anotherCar)**
    The *copy* constructor that initializes the instance variables using the values of anotherCar's instance variables.

Lab Assignment 04, Object-Oriented Programming, CSE 271, Spring 2020
Department of Computer Science and Engineering, Miami University

Constructor, Accessor, Mutator, Driver, Javadoc

- **Accessor and mutator 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 void setStart(boolean start)
  - public String getOwner()
  - public String getMake()
  - public String getModel()
  - public int getYearModel()
  - public float getFuelLevel()
  - public int getSpeed()
  - public boolean getStart()
- **public boolean accelerate()**
  The method increments the car's speed by 4 miles per hour and decreases the fuel level by 0.05. The car cannot accelerate if the engine is not on. Also, don't accelerate if the car does not have enough fuel (0.05 amount) to do that. If the current speed is the maximum, then acceleration won't increase the speed but will burn fuel. It returns true if it accelerates, increases the car speed by some amount, the car and false, otherwise.
- **public boolean brake()**
  The method decrements the car's speed by 3. You have to make sure that the car's engine is on to apply the brake and the speed cannot be negative. The speed cannot be negative as a result of a break. If you hit break while the car is running at 3 miles/hour or less then hitting the break should reduce the speed to 0 miles/hour. It returns true if it can apply brake and reduce the speed by some amount and false, otherwise.
- **public boolean isGasTankEmpty()**
  The method returns true if the fuel level of the car is less than 0.05. False, otherwise.
- **public boolean sameOwner(Car anotherCar)**
  The method returns true if the two cars have the same owner. False, otherwise. We assume that each person has a unique name.
- **public boolean equals(Car anotherCar)**
  It 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: Samuel, Make: Toyota, Model: Camry, Year: 2020, Speed: 75, Fuel Level: 0.7".

Lab Assignment 04, Object-Oriented Programming, CSE 271, Spring 2020
Department of Computer Science and Engineering, Miami University

Constructor, Accessor, Mutator, Driver, Javadoc

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 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 1 points)	7
Mutators (each worth 1 points) (total -3 if your methods don't throw IllegalArgumentException whenever and wherever applicable)	7
accelerate() method	6
brake() method	6
isGasTankEmpty() method	3
sameOwner(Car car) method	3
equals() method	4
toString() method	4
Driver	
Test constructors (each worth 1 points)	5
Test accessors and mutators (each worth 1 points)	14
Test rest 6 methods (each worth 1 points)	6
Javadoc	
Javadoc style comments (for each method 0.5 points and for the class 0.5)	13
Generate and submit correct Javadoc files	5
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.