Assignment 4

Late homework assignments will not be accepted, unless you have a valid written excuse (medical, etc.). You must do this assignment alone. No team work or "talking with your friends" will be accepted. No copying from the Internet. Cheating means zero.

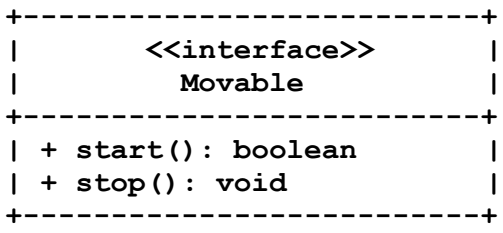
Create a new Eclipse workspace named "OOP" on the desktop of your computer. Create a Java Project named "Assignment3". Download the ass4-sample-code-for-student.zip from iSpace and unzip it. Next, we will import the sample code to your project (right click the src-> Import…-> File System-> locate to the unzipped folder->import all the questions to your project). Answer all the questions below. At the end, create a ZIP archive of all the src. In Eclipse, right click src-> Show In-> System Explorer, then zip up all folders inside the src folder. Rename your src.zip to "Assignment4_1234567890.zip" (replace 1234567890 with your student ID number). Upload the ZIP file onto AutoLab. For more details, refer to the file *How to Use Eclipse and submit to Autolab for student.pdf* on iSpace. Note: you must use the sample code, so AutoLab can work correctly. Here are a few extra instructions

- Put comments in your code (in English!) to explain WHAT your code is doing and to explain HOW your program is doing it.
- Make sure all your code is properly indented (formatted). Your code should be beautiful to read.

Failure to follow these instructions will result in you losing points.

question 1

Create a **`Movable`** interface with the following UML specification:

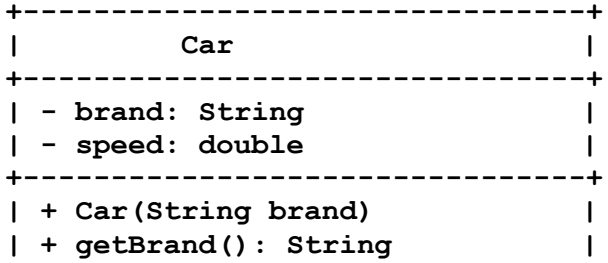


where:

- **`start()`**: starts to move with a certain speed. It returns true if the object starts successfully and false otherwise.
- **`stop()`**: stops moving

question 2

Create an abstract class **`Car`** with the following UML specification.



```
| + getSpeed(): double            |
| + setSpeed(double speed): void |
| + getSupply(): void             |
+---------------------------------+
```

Where

- **brand**: is the brand of the car
- **speed**: is the speed of the car
- **Car(String brand)**: initializes the brand of the car, and sets the speed of the car to zero.
- **getBrand()**: returns the brand of the car
- **getSpeed()**: returns the speed of the car
- **setSpeed()**: sets the speed of the car
- **getSupply**: refuels or recharges a car. However, since we don't know what type the car is, we don't know whether to go to a gas station or a charging station. Therefore, this method is abstract.

Further, a car is **Movable**, and it inherits the two methods of **Movable**:

- **start()**: since we don't know what type the car is, we don't know how fast it is. Therefore, this method is not implemented.
- **Stop()**: sets the speed of the car to zero.

question 3

Create a class **GasCar** with the following UML specification.

```
+---------------------------------+
|          GasCar                 |
+---------------------------------+
| - gasLevel: double              |
+---------------------------------+
| + GasCar(String brand)          |
| + testGasCar()                  |
+---------------------------------+
```

Where

- **gasLevel**: is the gas level of the car, ranging from 0 (empty) to 1 (full)
- **GasCar()**: initializes the gas level of the car to zero, in addition to the initialization of a general car.

Further, a **GasCar** is a **Car**, and it should override all the incomplete methods in **Car**:

- **`start()`**: tries to start the car. If the gas level is no more than zero, the car fails to start with an error message "Error: No gas!". Otherwise, the car starts with a speed of 60, and this operation decreases the gas level by 0.2. Then it prints a message "<brand> starts!". For example, if the car's brand is "BMW", then it prints "BMW starts!".

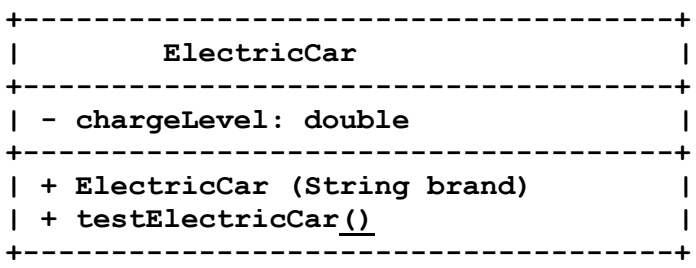
- **`getSupply()`**: refuels the gas car and sets its **`gasLevel`** to 1.

Your **`GasCar`** class should also have a static method called **`testGasCar()`** that contains the unit tests of at least the following methods:

- **`GasCar()`**
- **`getSupply()`**
- **`start()`**

question 4

Create a class **`ElectricCar`** with the following UML specification.



Where

- **`chargeLevel`**: is the battery level of the car, ranging from 0 (empty) to 1 (full)

- **`ElectricCar()`**: initializes the charge level of the car to zero, in addition to the initialization of a general car.

Further, a **`ElectricCar`** is a **`Car`**, and it should override all the incomplete methods in **`Car`**:

- **`start()`**: tries to start the car. If the charge level is no more than zero, the car fails to start with an error message "Error: Empty battery!". Otherwise, the car starts with a speed of 80, and this operation decreases the charge level by 0.1. Then it prints a message "<brand> starts!".

- **`getSupply()`**: recharges the electric car and sets its **`chargeLevel`** to 1.

Your **`ElectricCar`** class should also have a static method called **`testElectricCar()`** that contains the unit tests of at least the following methods:

- **`ElectricCar()`**
- **`getSupply()`**
- **`start()`**

Above all, your project should have a class **`Start`** with a **`main()`** method that launches the project. The main method should contain just two function calls: **`testGasCar()`** and **`testElectricCar()`**, and nothing else.