Programming Assignment – 3
CS 2133 Computer Science II
Due: 10/14/2020 at 9:00 AM on CANVAS

Read through each problem and write a JAVA program to implement the solution.

1. Turn in individual JAVA files on Canvas.
2. ALL JAVA programs titled: LastName_ProblemX.java – where X is the problem number
3. There should be a separate JAVA file for each problem, e.g., If you will have 3 problems, you will submit 3 separate JAVA files.
4. Please add comments in the program detailing your solution for each problem. Your comments should contain a very brief (maximum 4 lines) description of each solution.
5. All programs will be tested for completeness on three different test cases.
6. **Unless otherwise specified please do not use pre-defined libraries or functions.**

Write a program that explores the concepts of inheritance and polymorphism to represent different types of cars in JAVA. Instead of creating a class for each type of car, we will try to capture the inherent relationships among them to create custom, flexible representations using object-oriented programming principles.

In this assignment, you will create five classes - a base class, three child classes and a tester class.

Part 1: Base class

The base class will be a `Vehicle` class that captures some relationships that are common to all cars such as

1. Make (String)
2. Model (String)
3. Year (int)
4. License Plate Number (String)
5. Miles Driven (int)

Therefore, each car can be represented as a base class `Vehicle` that contains these attributes. Create a base class that has the above attributes. We will have 2 constructors, a 0-parameter constructor (all instance data are initialized to unknown or 0) and one which receives a parameter for each instance data and assigns them all appropriately.

Part 2: Child Classes

Cars can be powered using gas, electricity, or both! You should create three child classes that extend the Vehicle class to denote each type of car. The `GasCar` class will have 2 instance variables: tank capacity (float) and fuel type (String) to denote gas vs diesel, the

`ElectricCar` class will have 1 instance variable: battery capacity (float) and the `HybridCar` class will have 3 instance variables: battery capacity (float), tank capacity (float) and fuel type (String). Similar to the base class, using function overloading, create 2 constructors.

Part 3: Polymorphism

Implement a function `getFuelEfficiency` that computes the fuel efficiency of each car. The function should take in 1 parameter: `new_miles`. For the `GasCar` class, it should return the miles per gallon (MPG) computed as $mpg = \frac{new\ miles}{tank\ capacity}$.

For the `ElectricCar` class, it should return the miles per gallon equivalent (MPGe) computed as $mpge = \frac{new\ miles * cost\ of\ 1kWh}{battery\ capacity}$. You can set `cost of 1 kWh = $2`.

For hybrid, you should return $mpg = \frac{mpg + mpge}{2}$. Finally, it should add the parameter `newMiles` to the attribute `Miles Driven`.

Part 4: Tester class

Create a tester class with a `main` function that creates an instance of each child class and computes fuel efficiency of each instance. You should verify your implementation by creating instances using the two different constructors. Also verify your implementation of the `getFuelEfficiency` function for each instance.