Exercise Title: "Vehicle Inheritance Hierarchy"

Objective: The objective of this exercise is to reinforce the understanding of basic Object-Oriented Programming (OOP) concepts such as inheritance, abstraction, encapsulation, and polymorphism using C#.

Instructions: You are tasked with creating a simple inheritance hierarchy for different types of vehicles. Each vehicle should exhibit common properties and behaviors shared among vehicles of its type. Follow the instructions below to complete the exercise:

1. Create a C# console application project in your preferred development environment.
2. Define a base class called "Vehicle." This class should have the following properties and methods:
    - Properties:
        - **string Make** (with a getter and setter)
        - **string Model** (with a getter and setter)
        - **int Year** (with a getter and setter)
    - Methods:
        - A constructor that initializes the Make, Model, and Year properties.
        - A virtual method **Start()** that prints a message like "The vehicle is starting."
3. Create two derived classes, "Car" and "Motorcycle," that inherit from the "Vehicle" class.
    - Each derived class should have additional properties and methods specific to that type of vehicle:
        - For "Car":
            - Properties:
                - **int NumberOfDoors** (with a getter and setter)
            - Methods:
                - Override the **Start()** method to print a message specific to starting a car.
        - For "Motorcycle":
            - Properties:
                - **bool HasFairing** (with a getter and setter)
            - Methods:
                - Override the **Start()** method to print a message specific to starting a motorcycle.

4. Instantiate objects of both the "Car" and "Motorcycle" classes in your Main program.
5. Demonstrate the principles of encapsulation by setting and getting the properties of each object.
6. Demonstrate polymorphism by calling the **Start()** method on both the "Car" and "Motorcycle" objects. Even though they are of different types, the appropriate overridden method should be called based on the object type.
7. Compile and run your program to ensure that it works as expected.
8. Ensure that your code is well-commented and follows good coding practices.

Note: Feel free to add any additional properties or methods to enhance your vehicle classes, but be sure to include at least the specified properties and methods to cover the core OOP concepts.

Professor: Fabio Guilherme
Email: fabio.guilherme@gmail.com