Documentación del Proyecto Vehículos en C#

Introducción:

Este proyecto tiene como objetivo crear un sistema simple que simule vehículos de diferentes tipos (como autos eléctricos, autos de combustión, motocicletas y camiones). Para ello, utilicé algunos conceptos clave de la programación orientada a objetos, como la **herencia**, **polimorfismo** y **encapsulación**.

Paso 1: Configuración inicial del proyecto

Lo primero que hice fue crear el proyecto desde cero en Visual Studio Code. Primero, creé una carpeta llamada ProyectoVehiculos y luego utilicé el comando de dotnet para crear un proyecto de consola. Esto generó una estructura básica con un archivo principal llamado Program.cs, pero decidí limpiarlo porque iba a escribir mi propio código para probar las clases.

Después, decidí organizar mi código en una carpeta llamada Clases, donde iba a guardar todas las clases relacionadas con los vehículos. De este modo, pude mantener todo más ordenado y claro.

Paso 2: Creación de la clase base Vehiculo

La clase Vehiculo es la clase base de la que van a heredar todos los vehículos. Esta clase tiene algunos métodos comunes para todos los vehículos, como Acelerar, Frenar, Encender y Apagar.

Apliqué el concepto de **polimorfismo** al hacer estos métodos virtual, lo que significa que las clases que hereden de Vehiculo pueden sobrescribirlos con sus propias implementaciones, si así lo desean.

- El método Acelerar aumenta la velocidad del vehículo.
- El método Frenar reduce la velocidad del vehículo.
- Los métodos Encender y Apagar simplemente simulan encender y apagar el vehículo.

Paso 3: Creación de las clases derivadas

Aquí es donde comenzó a tomar forma el proyecto. Como la clase Vehiculo es la base, decidí crear clases más específicas para los diferentes tipos de vehículos. Estas clases derivadas heredan de Vehiculo y sobrescriben algunos de los métodos para tener comportamientos más específicos. Aquí te cuento cómo lo hice con cada uno:

1. **CarroEléctrico**: Esta clase representa a un vehículo eléctrico. Sobrescribí el método Acelerar para que cada vez que se acelere, también se reduzca la carga de la batería en un 1%. También agregué un método específico llamado CargarBateria, que simula la carga de la batería.
2. **AutoDeCombustion**: Este es un vehículo que usa gasolina. Al igual que en el caso del CarroElectrico, sobrescribí el método Acelerar, pero en lugar de perder batería, el auto de combustión pierde combustible al acelerar. También implementé un método CargarCombustible para recargar el tanque de gasolina.

3. **Motocicleta**: Para esta clase, quise que la motocicleta fuera más rápida al acelerar. Por eso, sobrescribí el método Acelerar para que aumente la velocidad más rápidamente que los otros vehículos. Además, agregué un método UsarCasco, que simula el uso de un casco de seguridad, algo que es importante para las motocicletas.
4. **Camion**: Finalmente, el camión tiene una característica especial: carga mercancías. En esta clase, sobrescribí el método Acelerar y agregué un nuevo método CargarMercancia, que simula la carga de mercadería en el camión. Además, al igual que en los otros vehículos, la velocidad del camión puede aumentar o disminuir con los métodos heredados de Vehiculo.

Paso 4: Creación del archivo Program.cs

Una vez que todas las clases estaban listas, era hora de probarlas. En el archivo Program.cs, instancié objetos de cada una de las clases derivadas: CarroElectrico, AutoDeCombustion, Motocicleta y Camion. Llamé a sus métodos para ver cómo se comportaban. También me aseguré de probar los métodos específicos de cada clase, como CargarBateria, CargarCombustible y CargarMercancia, para verificar que todo estuviera funcionando correctamente.

Resumen de lo que hice:

- Primero, creé una clase base llamada Vehiculo con métodos genéricos para acelerar, frenar, encender y apagar un vehículo.
- Luego, creé cuatro clases derivadas (CarroElectrico, AutoDeCombustion, Motocicleta y Camion) que heredan de

Vehiculo y sobrescriben los métodos según sus necesidades específicas.

- Apliqué **polimorfismo** al sobrescribir métodos como Acelerar, Frenar, y Encender en las clases derivadas para que tuvieran un comportamiento único.
- También utilicé **encapsulación** al declarar propiedades como nivelCombustible y cargaBateria como privadas, asegurándome de que solo pudieran ser modificadas a través de métodos específicos de cada clase.
- Finalmente, probé todas las clases en Program.cs para asegurarme de que todo funcionara correctamente.