Punto 1: Derivación de Clases

Para comenzar, creé una clase base llamada Vehiculo, que contiene las propiedades y métodos comunes a todos los vehículos. A partir de esta clase, derivé cuatro nuevas clases: CarroElectrico, AutoDeCombustion, Motocicleta y Camion. Gracias a esta derivación, pude reutilizar el código común y mantener una estructura organizada y eficiente.

Punto 2: Sobrescritura de Métodos y Encapsulación

Encapsulé la propiedad private int velocidad = 0; en la clase base Vehiculo para protegerla de modificaciones externas directas. Esto garantiza que la velocidad solo pueda ser modificada a través de métodos específicos, manteniendo la integridad del estado del objeto.

Luego, sobrescribí los métodos Acelerar, Frenar, Encender y Apagar en las clases derivadas para añadir comportamientos específicos a cada tipo de vehículo. Por ejemplo, en AutoDeCombustion, el método Acelerar reduce el nivel de combustible, mientras que en Motocicleta, este mismo método incrementa la velocidad más rápidamente.

❖ **AutoDeCombustion**

En la clase AutoDeCombustion, añadí las propiedades nivelCombustible, capacidadTanque y tipoCombustible para gestionar el nivel de combustible y el tipo de combustible del auto. Además, sobrescribí los métodos Acelerar, Frenar, Encender y Apagar para incluir la lógica específica de este tipo de vehículo.

Propiedades:

- nivelCombustible
- capacidadTanque
- tipoCombustible

Métodos sobrescritos:

- **Acelerar:** Disminuye el nivel de combustible al acelerar.
- **Frenar:** Disminuye el nivel de combustible al frenar.
- **Encender:** Cambia el estado del vehículo a encendido.
- **Apagar:** Cambia el estado del vehículo a apagado.

❖ **Motocicleta**

En la clase Motocicleta, añadí las propiedades cilindrada, tipoMoto y nivelAceite para gestionar características específicas de una motocicleta. Además, sobrescribí el método Acelerar para que la velocidad aumente más rápidamente en comparación con otros vehículos.

Propiedades:

- cilindrada

- tipoMoto
- nivelAceite

Métodos sobrescritos:

- **Acelerar:** Incrementa la velocidad más rápidamente.

❖ **Camión**

En la clase Camion, añadí las propiedades capacidadCarga, numeroEjes y pesoMaximo para gestionar características específicas de un camión. Además, sobrescribí los métodos Acelerar, Frenar, Encender y Apagar para incluir la lógica particular de este tipo de vehículo.

Propiedades:

- capacidadCarga
- numeroEjes
- pesoMaximo

Métodos sobrescritos:

- **Acelerar:** Incluye un mensaje adicional indicando que el camión está acelerando.
- **Frenar:** Incluye un mensaje adicional indicando que el camión está frenando.
- **Encender:** Cambia el estado del vehículo a encendido.
- **Apagar:** Cambia el estado del vehículo a apagado.

❖ **Programa Principal (Program.cs)**

En el programa principal, creé instancias de las clases derivadas (CarroElectrico, AutoDeCombustion, Motocicleta, Camion) y llamé a sus métodos para demostrar su funcionalidad. Este archivo contiene el método Main, que actúa como el punto de entrada del programa.

Resumen

A través de este código, probé cómo cada tipo de vehículo responde a acciones como acelerar, frenar, encender y apagar, verificando así que la herencia y sobrescritura de métodos funcionan correctamente.

En este proyecto, creé una jerarquía de clases utilizando la herencia en C#. La clase base Vehiculo define propiedades y métodos comunes a todos los vehículos. Las clases derivadas (CarroElectrico, AutoDeCombustion, Motocicleta, Camion) extienden la funcionalidad de la clase base con propiedades y comportamientos específicos.

El método Main en Program.cs demuestra cómo se pueden crear instancias de estas clases y utilizar sus métodos para simular el comportamiento de distintos tipos de vehículos. Este enfoque permite reutilizar el código y mantener una estructura organizada, facilitando la escalabilidad y mantenimiento del proyecto. 🚀