UNIVERDAD MARIANO GALVEZ DE GUATEMALA
FACULTAD DE INGENIERIA EN SISTEMAS DE INFORMACION Y CIENCIAS DE LA COMPUTACION
CENTRO UNIVERSITARIO CAMPUS JUTIAPA

FACULTAD:

INGENIERIA EN SISTEMAS

CURSO:

PROGRAMACION I

CATEDRATICO:

ING RULDIN EFRAIN AYALA RAMOS

ALUMNO:

ANTONY EZEQUIEL PINEDA PINEDA

CARNE:

0905-24-11747

Documentación del Proyecto: Herencia y Polimorfismo en C#

1. Clase Autocombustión

Esta clase hereda de la clase Vehiculo, y le agregamos propiedades específicas como la capacidad del tanque, el consumo de combustible y el tipo de combustible.

```
private int capacidadTanque;
private double consumoCombustible;
private string tipoCombustible;
```

Método CargarCombustible

```
1 referencia
public void CargarCombustible(int litros)
{
    capacidadTanque += litros;
    Console.WriteLine("Se han cargado {0} litros. Capacidad actual: {1}L", litros, capacidadTanque);
}
```

Este método se encarga de agregar combustible al auto. La cantidad de litros que se carga se suma a la capacidad total del tanque y se muestra un mensaje indicando cuántos litros se han agregado.

Método frenar sobrescrito

```
2 referencias
public override void frenar(int cuanto)
{
    velocidad -= cuanto;
    if (velocidad < 0) velocidad = 0;

    capacidadTanque -= 1;  // Simulamos el consumo de combustible al frenar
    if (capacidadTanque < 0) capacidadTanque = 0;

    Console.WriteLine("Frenaste. Velocidad: {0} KMS/H. Combustible restante: {1}L", velocidad, capacidadTanque);
}
```

Aquí sobrescribí el método frenar de la clase base. Al frenar, no solo reducimos la velocidad, sino que también disminuimos la capacidad del tanque de combustible. Es una simulación simple de cómo se consume algo de combustible cuando un auto frena. Esto se debe a que en la vida real, los autos de combustión utilizan energía (y por lo tanto combustible) incluso al frenar.

2.Clase Motocicleta

Al igual que el AutoDeCombustion, la motocicleta hereda de la clase Vehiculo, pero tiene propiedades propias como el cilindraje, casco y estado del motor.

Propiedades de la Clase Motocicleta

```
internal class Motocicleta : Vehiculo
{
    private bool tieneCasco;
    private int cilindraje;
    private bool encendida;
```

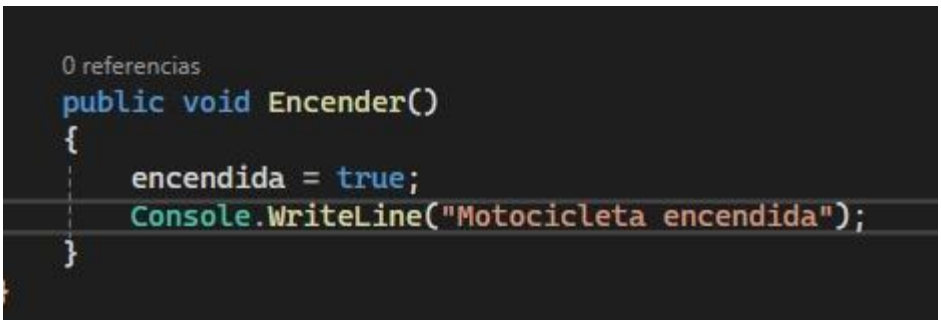
El cilindraje se refiere al tamaño del motor de la moto, algo fundamental para definir el tipo de moto (por ejemplo, una moto de 125cc vs una de 1000cc). La propiedad tieneCasco es bastante sencilla, y sirve para simular si el conductor de la moto está protegido o no. Y por último, encendida nos permite controlar si la moto está encendida o apagada

Método acelerar sobrescrito

```
5 referencias
public override void acelerar(int cuanto)
{
    base.acelerar(cuanto + 5);
    Console.WriteLine("La motocicleta acelera más rápido!");
}
```

Quise que la moto tuviera una aceleración más fuerte, porque las motos suelen ser más rápidas al arrancar en comparación con los autos. Aquí, le sumé 5 a la variable cuanto para darle un pequeño empuje extra en la aceleración. Al llamar a base.acelerar(cuanto + 5), estoy utilizando el método acelerar de la clase base (Vehiculo), pero con un valor ajustado. Este tipo de cambios son típicos en la programación orientada a objetos cuando queremos personalizar un comportamiento heredado de la clase base.

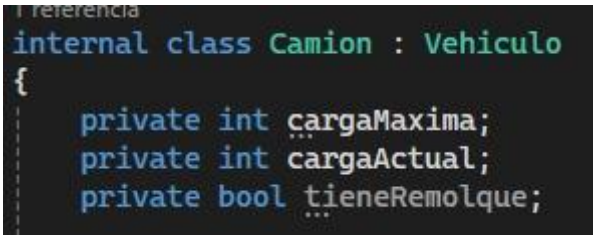
Método encender



Este método es solo una simulación de lo que pasa cuando se enciende una moto. Lo pensé como algo sencillo, pero esencial. Sin este método, la moto no podría comenzar a moverse o acelerar, así que se convierte en algo clave en el funcionamiento de la moto.

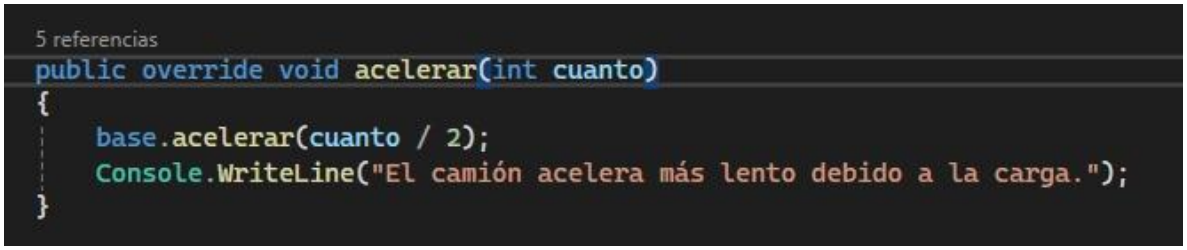
3.Clase Camión

Finalmente, en el caso del Camión, decidí agregar características relacionadas con la carga, ya que los camiones son conocidos por transportar grandes volúmenes de peso.



La cargaMaxima es lo que define el límite de peso que un camión puede llevar. La cargaActual es el peso real que lleva el camión en ese momento. Y tieneRemolque es una propiedad que puede cambiar según el tipo de camión, pues algunos camiones tienen remolques adicionales que aumentan su capacidad de carga.

Método acelerar sobrescrito



Al sobrescribir el método acelerar para el camión, quise reflejar que los camiones, debido a su peso, son más lentos para ganar velocidad. Por eso, reduje la aceleración a la mitad. Esto es solo una simulación simple, pero me ayudó a crear un comportamiento más realista para el camión.

Método Cargar

```
0 referencias
public void Cargar(int peso)
{
    if (cargaActual + peso <= cargaMaxima)
    {
        cargaActual += peso;
        Console.WriteLine("Carga actual: {0}kg", cargaActual);
    }
    else
    {
        Console.WriteLine("No se puede cargar más, el camión está lleno.");
    }
}
}
```

El método Cargar es un ejemplo de cómo simular el proceso de cargar un camión. Verifica si el peso que quiero agregar no supera la cargaMaxima. Si no se excede, se agrega el peso a la carga actual; de lo contrario, muestra un mensaje diciendo que no hay más espacio para cargar.

Comentario

El proceso de desarrollar estas clases fue en gran parte un ejercicio de crear vehículos más específicos sin perder de vista las funcionalidades comunes. Usando la herencia, pude crear vehículos distintos pero con una estructura similar, donde cada uno tiene sus propias características pero hereda comportamientos comunes de la clase Vehículo.

Lo más importante fue asegurarme de que cada clase reflejara comportamientos realistas. Por ejemplo, la moto acelera más rápido que el coche debido a su menor tamaño, el camión tiene restricciones en su aceleración por su peso y el auto de combustión tiene que lidiar con el consumo de combustible al frenar. Todo esto se hizo de manera sencilla, pero fiel a las características de cada tipo de vehículo.

Al final, la clave fue la reutilización de código a través de la herencia, lo que me permitió hacer que cada clase tuviera lo que necesitaba sin repetir demasiado código. ¡Esto es lo bonito de la programación orientada a objetos!