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 1

CATEDRATICO:

ING. RULDIN EFRAIN AYALA RAMOS

ALUMNO:

LEBINSON DAVID GARCIA CASTILLO

CARNE:

0905-24-13926

Paso 1: Crear la clase `AutoDeCombustion`

La clase `AutoDeCombustion` hereda de la clase base `Vehiculo`. En esta clase, añadí una propiedad `TipoDeCombustible` que especifica qué tipo de combustible usa el auto (como gasolina, diésel, etc.). Esto me permitió dar más detalle a los autos en comparación con los otros vehículos. No modifiqué el comportamiento de los métodos heredados como `Acelerar()` y `Frenar()` porque el auto de combustión sigue la lógica estándar de estos métodos.

```
using p1bpoo.MisClases;

3 referencias
internal class AutoDeCombustion : Vehiculo
{
    private int capacidadTanque;
    private int nivelCombustible;
    private string tipoCombustible;

    1 referencia
    public AutoDeCombustion(int anio, string elColor, string elModelo, int capacidad, string tipo) : base(anio, elColor, elModelo)
    {
        capacidadTanque = capacidad;
        nivelCombustible = capacidad;
        tipoCombustible = tipo;
    }

    6 referencias
    public override void acelerar(int cuanto)
    {
        base.acelerar(cuanto);
        nivelCombustible -= 1;
        Console.WriteLine("Combustible restante: {0}%", nivelCombustible);
    }

    3 referencias
    public override void Frenar()
    {
        base.Frenar();
        nivelCombustible -= 1;
        Console.WriteLine("El auto de combustión está frenando.");
    }
```

Paso 2: Crear la clase `Camión`

En la clase `Camión`, que también hereda de `Vehiculo`, sobrescribí el método `Frenar()`. Esto es porque un camión, debido a su tamaño y peso, tarda más en detenerse que un auto o una motocicleta. Así que, en lugar de poner la velocidad a cero de inmediato como en la clase base, reduje la velocidad en 5 unidades por cada llamada al método `Frenar()`, pero también aseguré que la velocidad nunca fuera negativa, estableciendo un límite en 0.

```
using p1bpoo.MisClases;

3 referencias
internal class Camion : Vehiculo
{
    private int cargaMaxima;
    private int cargaActual;
    private bool tieneRemolque;

    1 referencia
    public Camion(int anio, string elColor, string elModelo, int cargaMax, bool remolque) : base(anio, elColor, elModelo)
    {
        cargaMaxima = cargaMax;
        cargaActual = 0;
        tieneRemolque = remolque;
    }

    6 referencias
    public override void acelerar(int cuanto)
    {
        base.acelerar(cuanto / 2);
        Console.WriteLine("El camión acelera más lento debido a la carga.");
    }

    1 referencia
    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.");
        }
    }
```

Paso 3: Crear la clase `Motocicleta`

Para la clase `Motocicleta`, que también hereda de `Vehiculo`, decidí sobrescribir el método `Frenar()` para reflejar que una motocicleta es más ágil y puede frenar más rápidamente que un camión. Por lo tanto, en lugar de reducir la velocidad gradualmente, lo hice a la mitad, simulando que el frenado de la motocicleta es más eficaz.

```
using p1bpoo.MisClases;

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

    1 referencia
    public Motocicleta(int anio, string elColor, string elModelo, int cilindraje, bool casco) : base(anio, elColor, elModelo)
    {
        this.cilindraje = cilindraje;
        tieneCasco = casco;
        encendida = false;
    }

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

    1 referencia
    public void Encender()
    {
        encendida = true;
        Console.WriteLine("Motocicleta encendida");
    }
}
```

Paso 4: Crear el archivo `Program.cs`

Finalmente, en el archivo `Program.cs`, instancié objetos de las tres clases derivadas (`AutoDeCombustion`, `Camion`, `Motocicleta`), sin interrumpir las dos que ya existían de (`vehculo y vehiculoElectrico`)y realicé pruebas con los métodos `Acelerar()` y `Frenar()`. Aceleré cada vehículo, luego apliqué el freno y observé cómo se comportaba cada uno.

```
using p1bpoo.MisClases;
using System;

namespace p1bpoo
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {

            AutoDeCombustion miAuto = new AutoDeCombustion(2023, "Rojo", "Toyota", 50, "Gasolina");
            Motocicleta miMoto = new Motocicleta(2022, "Negra", "Harley Davidson", 1200, true);
            Camion miCamion = new Camion(2020, "Azul", "Volvo", 10000, true);
            CarroElectrico miCarroElectrico = new CarroElectrico(2025, "Blanco", "Tesla");

            miAuto.InformacionVehiculo();
            miAuto.acelerar(30);
            miAuto.Frenar();

            miMoto.InformacionVehiculo();
            miMoto.acelerar(40);
            miMoto.Encender();

            miCamion.InformacionVehiculo();
            miCamion.acelerar(20);
            miCamion.Cargar(5000);

            miCarroElectrico.InformacionVehiculo();
            miCarroElectrico.acelerar(10);
            Console.WriteLine("Nivel de batería: " + miCarroElectrico.NivelBateria());

            Console.ReadLine();
        }
    }
}
```

Paso 5: el proyecto debe ser actualizado

Para actualizar el proyecto de la versión 7.0 a 9.0, abrí el archivo `.csproj` y cambié la línea `<TargetFramework>net7.0</TargetFramework>`a`<TargetFramework>net9.0</TargetFramework>`. Luego, actualicé las dependencias a través de la consola de NuGet con el comando `Update-Package` para asegurarme de que todo estuviera compatible con la nueva versión. Después de realizar esto, compilé el proyecto para verificar que no hubiese errores y corregí los que aparecieron para que el código fuera compatible con .NET 9.0.