Universidad Mariano Gálvez de Guatemala

Facultad ingeniería

Ingeniería en sistemas

Curso: PROGRAMACION 1

Docente: Ruldin Ayala

Tarea: Herencia y Polimorfismo en C#

Autor: Eduardo Gabriel Visoni Morales

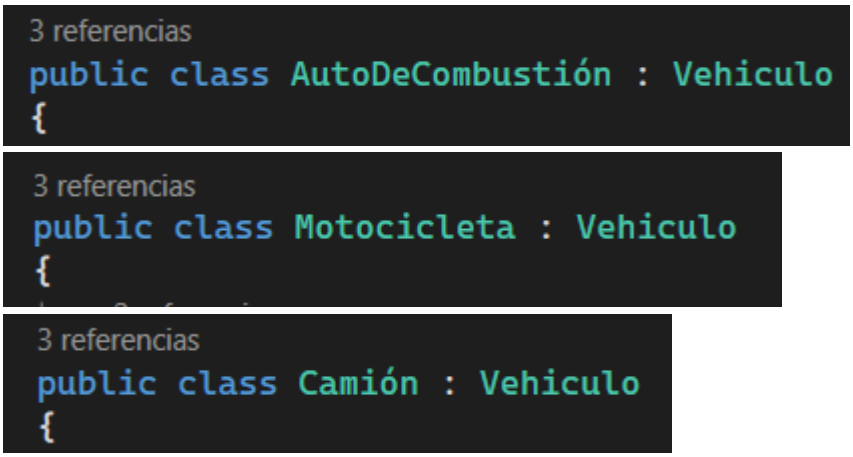
0905-22-1146

Jutiapa, 19 de marzo de 2025

Documentación Herencia y Polimorfismo en C#:

Parte 1: Crear 3 clases derivadas de la clase Vehículo:

Punto 1: De la clase Padre “Vehículo” hice Herencia para las siguientes clases “AutoDeCombustion”, “Motocicleta y “Camion” de la siguiente manera:



Punto 2: Cada clase tiene sus propiedades, además de las 3 que se Heredan a cada clase de la clase padre “vehículo”, se le hizo a cada una (AutoDeCombustion, Motocicleta, Camion) 3 propiedades adicionales diferentes una de la otra. También explico los ejemplos de encapsulación utilizados en cada una de las clases:

- Vehículo: Color, Modelo, Year

```
4 referencias
protected string Color { get; set; }
2 referencias
protected string Modelo { get; set; }
2 referencias
protected int Year { get; set; }

protected int velocidad = 0;

protected bool encendido = false;
```

En vehículo use “Protected” en todas las propiedades principales para que solamente las clases Heredadas descritas anteriormente pudieran usarse

- AutoDeCombustion: Sillón, asientos, Llantas

```
2 referencias
public string sillon { get; set; }
2 referencias
private int asientos { get; set; }
2 referencias
public string llantas { get; set; }

private bool lucesTraserasEncendidas = false;
2 referencias
protected string modelo { get; set; }
```

En AutoDeCombustion use “Public” en sillón y en llantas ya que, en la vida real, si se pueden cambiar por cualquier persona.

Ahora, use “Private” en asientos ya que es la cantidad de filas que hay en el carro de asientos, y si hay 3 filas no puedes “Agregar más filas”, al igual que en lucesTraserasEncendidas ya que es una propiedad específica para el carro cuando se frena y no quiero que nadie más pueda hacerlo.

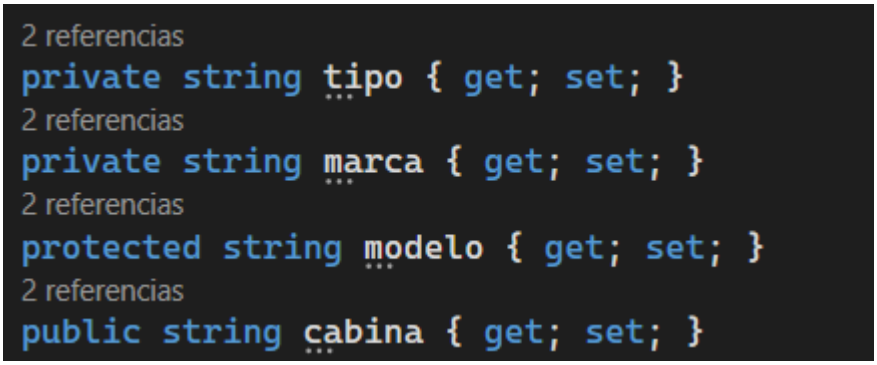
Y en modelo se usó “Protected” de la Herencia Padre.

- Motocicleta: Frenos, Motor, suspensión

```
2 referencias
public string frenos { get; set; }
2 referencias
private string motor { get; set; }
2 referencias
private string modelo { get; set; }
2 referencias
protected string suspension { get; set; }
```

Use “Public” en frenos ya que se puede cambiar por cualquier persona.
Use “Private” con Motor ya que no quiero que se le pueda cambiar el motor.
Use “Protected” con suspensión ya que quiero que, si se le cambia, solo se haga con suspensiones especificas de moto y su modelo, es decir, solo marcas específicas y direccionadas a esa moto, no “cualquier”

- Camion: Tipo, Marca, cabina



Use “Public” con cabina ya que hay cosas dentro de una cabina que pueden ser modificados

Use “Private” con tipo de camión, ya que si es camión de carga no se puede cambiar su tipo, y también con la marca ya que si es Toyota, será Toyota siempre

Use “Protected” con modelo ya que es Herencia de la clase Padre

Parte 2: Sobrescribir métodos en las clases derivadas

En las clases “vehículo”, agregue 3 métodos, los cuales debíamos sobrescribir en las clases derivadas y darles diferentes funciones a cada uno, los cuales son:

- Frenar

```
public virtual int frenar(int cuanto)
{
    if (!encendido)
    {
        Console.WriteLine("El vehículo está apagado. No se puede frenar.");
        return velocidad;
    }

    velocidad -= cuanto;
    if (velocidad < 0)
    {
        velocidad = 0;
    }
    Console.WriteLine("Vas a {0} KMS/Hora", velocidad);
    return velocidad;
}
```

Cree “Frenar” con un if, para validar si el vehículo no este encendido, retorne un mensaje diciendo que no se puede frenar.

Al igual, si esta encendido, use otro if para que cuando haya decremento de velocidad en “cuanto” muestre un mensaje diciendo los km/hr que se llega restando la velocidad inicial, con la velocidad que se quiere frenar y si el valor es menor a 0 se ajuste por los valores negativos

Luego de eso, sobrescribí ese método en AutoDeCombustion de la siguiente manera:

```
4 referencias
public override int frenar(int cuanto)
{
    velocidad -= cuanto;
    if (velocidad < 0)
    {
        velocidad = 0;
    }
    if (velocidad == 0)
    {
        Console.WriteLine("El vehículo está detenido. No se puede frenar.");
        return velocidad;
    }
    lucesTraserasEncendidas = true;
    Console.WriteLine("Ahora, las luces traseras están encendidas.");
    return velocidad;
}
1 referencia
public void ApagarLucesTraseras()
{
    lucesTraserasEncendidas = false;
    Console.WriteLine("Las luces traseras están apagadas.");
}
```

Use el contexto base e hice que existiera una función para que las luces traseras del carro se encendieran cuando se frenara con un valor “boll” declarado arriba para saber si el carro esta encendido o no, ya que, para esta función, las luces se prenden si se frena con el carro encendido.

También sobrescribí la misma función en Motocicleta de la siguiente manera:

```
public int frenar(int cuanto, string llanta)
{
    velocidad -= cuanto;
    if (velocidad < 0)
    {
        velocidad = 0;
    }

    if (llanta.ToLower() == "delantera")
    {
        Console.WriteLine("Frenando con la llanta delantera. Vas a {0} KMS/Hora", velocidad);
    }
    else if (llanta.ToLower() == "trasera")
    {
        Console.WriteLine("Frenando con la llanta trasera. Vas a {0} KMS/Hora", velocidad);
    }
    else
    {
        Console.WriteLine("Llanta no especificada correctamente. Vas a {0} KMS/Hora", velocidad);
    }
    return velocidad;
}
```

Lo que hice fue usar parte de la base de “Frenar” y usar if else para que se pueda determinar con que llanta se frena, si es la delantera o la trasera, y si se escribe algún otro valor a no ser “delantera” o “trasera”, que devuelva un mensaje diciendo que la llanta especificada no es correcta.

También sobrescribí la misma función en Camion de la siguiente manera:

```
public int frenar(int cuanto, string llanta)
{
    velocidad -= cuanto;
    if (velocidad < 0)
    {
        velocidad = 0;
    }

    if (llanta.ToLower() == "disco")
    {
        Console.WriteLine("Usando freno con disco. Vas a {0} KMS/Hora", velocidad);
    }
    else if (llanta.ToLower() == "aire")
    {
        Console.WriteLine("Usando freno de aire. Vas a {0} KMS/Hora", velocidad);
    }
    else if (llanta.ToLower() == "motor")
    {
        Console.WriteLine("Usando freno de motor. Vas a {0} KMS/Hora", velocidad);
    }
    else
    {
        Console.WriteLine("Freno no especificada correctamente. Vas a {0} KMS/Hora", velocidad);
    }
        return velocidad;
}
```

Lo que hice fue que al momento del camión frenar, por medio de if else, hacer que se pueda elegir el tipo de freno, ya sea por disco, aire o motor, y si se especifica otro valor a no ser ellos, deja un mensaje.

- Acelerar

```
public virtual int acelerar(int cuanto)
{
    if (!encendido)
    {
        Console.WriteLine("El vehículo está apagado. No se puede acelerar.");
        return velocidad;
    }

    velocidad += cuanto;
    Console.WriteLine("Vas a {0} KMS/Hora", velocidad);
    return velocidad;
}
4 referencias
```

Acelerar fue creado en clase.

Luego de eso, sobrescribí ese método en AutoDeCombustion de la siguiente manera:

```
public override int acelerar(int cuanto)
{
    if (!encendido)
    {
        Console.WriteLine("El vehículo está apagado. No se puede acelerar.");
        return velocidad;
    }

    if (velocidad + cuanto >= 50)
    {
        velocidad += cuanto * 2;
        Console.WriteLine("Con turbo, vas a: {0} KMS/Hora", velocidad);
    }
    else
    {
        velocidad += cuanto;
        Console.WriteLine("Vas a {0} KMS/Hora", velocidad);
    }
    return velocidad;
```

La función que tiene este método de acelerar es que, además de usar la base, ahora si la velocidad llega a ser mayor o igual a 50KM/Horas, el carro activará el turbo y la velocidad será duplicada.

También sobrescribí la misma función en Motocicleta de la siguiente manera:

```
5 referencias
public override int acelerar(int cuanto)
{
    int incremento = cuanto * 2;
    velocidad += incremento;
    Console.WriteLine("La motocicleta va a {0} KMS/Hora", velocidad);
    return velocidad;
}
```

Simplemente la moto acelera más rápido que un carro, por lo que la velocidad es relativamente duplicada, y llega a velocidades altas, más rápido que un carro.

También sobrescribí la misma función en Camion de la siguiente manera:

```
public override int acelerar(int cuanto)
{
    int incremento = cuanto / 2;
    velocidad += incremento;
    Console.WriteLine("El camion es mas lento, va a {0} KMS/Hora", velocidad);
    return velocidad;
}
```

Lo hice de esa manera ya que el camión de hecho va mas lento que Vehículo, auto de combustión y motocicleta entonces hice que fuera a la mitad de velocidad que se le asigna, para que parezca que va más lento.

-Encender

```
public virtual void encender()
{
    if (encendido)
    {
        Console.WriteLine("El vehículo ya está encendido.");
        return;
    }

    else
    {
        encendido = true;
        Console.WriteLine("El vehículo está encendido.");
    }
}
```

4 referencias

Lo coloque en la clase padre, Vehículo, declarando en el comienzo de la clase un valor tipo bool, para saber si esta encendido.

También sobrescribí la misma función en AutoDeCombustion, Motocicleta y Camion de la siguiente manera:

En cada una de las clases sobrescribí el método e hice que en cada vehículo de transporte se mostrara un mensaje diferente en la pantalla TFT (Pantalla donde se muestra la información de velocidad, km, gasolina etc) [Las imágenes están en este orden, de arriba hacia abajo:

```
public override void encender()
{
    encendido = true;
    Console.WriteLine("El vehículo está encendido.");
    Console.WriteLine("Pantallas TFT del carro: SEJA BEM-VINDO A MAIS UMA VIAJEM!!");
}
```

```
public override void encender()
{
    encendido = true;
    Console.WriteLine("La motocicleta está encendida.");
    Console.WriteLine("Pantallas TFT de la moto: BIENVENUE DANS UN AUTRE VOYAGE!!");
}
```

```
public override void encender()
{
    encendido = true;
    Console.WriteLine("El camion está encendido.");
    Console.WriteLine("Pantallas TFT del camion: BIENVENIDO A OTRO VIAJE!!");
}
```

Resumen General:

En resumen, todo lo que hice fue simplemente agregar 3 clases más al programa, que las herede de la clase padre “Vehículo” y a cada una le agregue propiedades diferentes una de la otra. Al igual, en la clase padre hice 3 funciones nuevas (además de acelerar) y las sobrescribí en las 3 clases nuevas.

En las propiedades de cada clase Utilice diferentes tipos de encapsulación para cumplir con las instrucciones y para que tenga sentido si lo usamos en un contexto de la vida real. Además, en los métodos sobrescritos, trate de adaptarme a la realidad lo máximo posible, y usar funciones que realmente tengan sentido y puedan usarse, tanto cosas que se pueden “Realizar” (acelerar) tanto como cosas que se pueden “Mostrar” (un mensaje en pantalla, etc.)

Trate de hacer el interfaz del CMD lo mas entendible posible, para que cuando se muestre la información sea entendible. Espero este proyecto haya cumplido con todas los requerimientos, me esforcé para poder hacerlo y no solamente eso, pero también comprender la lógica de cómo funciona POO, usando Herencias, Polimorfismo y usar correctamente la encapsulación.