Explicación y Mejor Práctica

Para asegurarnos de que la propiedad TipoLicencia se valide cada vez que se cambie, podemos usar un método de validación dentro del setter de la propiedad. Esto se puede lograr utilizando propiedades con lógica personalizada en C#.

Implementación

1. **Agregar un método de validación**: Define un método privado que valide la licencia en función de la edad.
2. **Modificar el setter de la propiedad TipoLicencia**: Llama al método de validación dentro del setter de la propiedad.

Código Modificado

```
using p1boo.Interfaces;

8 referencias
public class Chofer : Piloto
{
    private string tipoLicencia;
    3 referencias
    public string Nombre { get; set; }
    3 referencias
    public int Edad { get; set; }
    4 referencias
    public string TipoLicencia
    {
        get { return tipoLicencia; }
        set
        {
            ValidarLicencia(value, Edad);
            tipoLicencia = value;
        }
    }

    2 referencias
    public Chofer(string nombre, int edad, string tipoLicencia)
    {
        Nombre = nombre;
        Edad = edad;
        TipoLicencia = tipoLicencia; // Esto llamará al setter y validará la licencia
    }

    1 referencia
    private void ValidarLicencia(string licencia, int edad)
    {
        if (edad < 18 && licencia != "M")
        {
            throw new ArgumentException("Los menores de 18 años solo pueden tener licencia tipo M.");
        }
        // Agrega más validaciones según sea necesario
    }

    0 referencias
    public void MostrarInformacion()
    {
        Console.WriteLine("Nombre: {0}", this.Nombre);
        Console.WriteLine("Edad: {0}", this.Edad);
        Console.WriteLine("Licencia Tipo: {0}", this.TipoLicencia);
    }
}
```

Explicación

1. **Método <u>ValidarLicencia</u>**: Este método privado realiza la validación de la licencia en función de la edad del chofer.
2. **Setter de <u>TipoLicencia</u>**: El setter de la propiedad <u>TipoLicencia</u> llama al método <u>ValidarLicencia</u> antes de asignar el valor a la variable privada <u>tipoLicencia</u>.

Mejor Práctica

La mejor práctica es encapsular la lógica de validación dentro de la clase misma, asegurando que cualquier cambio en la propiedad pase por la validación necesaria. Esto se logra utilizando propiedades con lógica personalizada en C#.

CHAT UTILIZADO: <u>COMPILOT</u>

Para abordar la necesidad de validar y restringir cambios en la propiedad `TipoLicencia` de la clase `Chofer` en C#, es esencial implementar una estrategia que garantice la coherencia y validez de los datos. A continuación, se detallan dos enfoques recomendados, basados en consultas realizadas a modelos de lenguaje (LLMs) y en las mejores prácticas de programación orientada a objetos en C#.

1. Uso de Propiedades con Validación en el Setter

Implementar la validación directamente en el setter de la propiedad permite controlar y restringir los valores asignados en tiempo de ejecución. Este enfoque es útil cuando se requiere una validación inmediata cada vez que se modifica la propiedad.

IMPLEMENTACION:

```
public class Chofer
{
    private string tipoLicencia;
    private int edad;

    public string TipoLicencia
    {
        get => tipoLicencia;
        set
        {
            if (EsLicenciaValidaParaEdad(value, edad))
            {
                tipoLicencia = value;
            }
            else
            {
                throw new ArgumentException("La licencia no es válida para la edad del chofer
```

Copiar

```
    public int Edad
    {
        get => edad;
        set
        {
            edad = value;
            // Opcional: Revalidar TipoLicencia cuando la edad cambia
            if (!EsLicenciaValidaParaEdad(tipoLicencia, edad))
            {
                tipoLicencia = null; // O manejar según la lógica de negocio
            }
        }
    }

    private bool EsLicenciaValidaParaEdad(string licencia, int edad)
    {
        // Implementar la lógica de validación según los requisitos
        return true; // Ejemplo simplificado
    }
}
```

Ventajas:

- Validación inmediata al asignar un nuevo valor a la propiedad.Stack Overflow
- Encapsulación de la lógica de validación dentro de la clase.

Consideraciones:

- Es importante manejar las excepciones adecuadamente para evitar interrupciones inesperadas en la aplicación.
- Si la validación es compleja, puede ser preferible externalizarla a una clase de validación dedicada.

CHAT UTILIZADO: CHATGPT