Tarea de Investigación.

1. **Use el LLM de Deepseek**

Usar propiedades con validación

Implementar un método para validar y actualizar la licencia

Hacer que la licencia dependa de la edad automáticamente

Agregar un método llamado ActualizarLicencencia

**Haz que TipoLicencia dependa de la edad automáticamente**

* Usa un **método privado** (DeterminarLicencia) que calcule la licencia válida según la edad.
* Actualiza la licencia **cada vez que cambie la edad** (en el constructor o con un método como ActualizarEdad).

**Conclusión:**

La mejor práctica es **centralizar la lógica de validación** en un método (como DeterminarLicencia) y **acoplarla a cambios en la edad**, evitando asignaciones directas sin validación.

Prompt:

using System;

namespace p1bpoo.MisClases

{

internal class Chofer : IPiloto

{

public string Nombre { get; set; }

public int Edad { get; private set; }

public string TipoLicencia { get; private set; }

public enum TipoLicencia

{

A, // Motocicletas

B, // Vehículos livianos

C, // Vehículos pesados

M, // Motocicletas

T // Transporte colectivo

}

public Chofer(string nombre, int edad)

{

Nombre = nombre;

Edad = edad;

TipoLicencia = DeterminarLicencia(edad).ToString();

}

// Método para actualizar edad y recalcular licencia

public void ActualizarEdad(int nuevaEdad)

{

Edad = nuevaEdad;

TipoLicencia = DeterminarLicencia(nuevaEdad).ToString();

}

private TipoLicencia DeterminarLicencia(int edad)

{

if (edad >= 25)

{

return TipoLicencia.T;

}

else if (edad >= 21)

{

return TipoLicencia.C;

}

else if (edad >= 18)

{

return TipoLicencia.B;

}

else if (edad >= 17)

{

return TipoLicencia.A;

}

else if (edad >= 16)

{

return TipoLicencia.M;

}

else

{

throw new ArgumentException("Edad no corresponde con ningún tipo de licencia.");

}

}

public void MostrarInformacion()

{

Console.WriteLine($"El piloto es: {Nombre}");

Console.WriteLine($"La licencia es: {TipoLicencia}");

Console.WriteLine($"Edad: {Edad}");

}

// Implementación explícita de la interfaz

void IPiloto.mostrarInformacion()

{

MostrarInformacion();

}

}

}

1. **Use el LLM llamado Gemini**

La clave está en los bloques set de las propiedades. Cada vez que se asigna un nuevo valor a Edad o TipodeLicencia, el código dentro del set se ejecuta.

Dentro de esos bloques set, llamo al método ValidarTipoLicencia(), que realiza la validación real.

Por lo tanto, si después de crear un objeto Chofer, intentas cambiar el valor de TipodeLicencia a través de la propiedad, el método ValidarTipoLicencia() se ejecutará automáticamente y lanzará una excepción si el nuevo tipo de licencia no es válido para la edad actual del conductor.

Espero que esta explicación más detallada haya aclarado tus dudas. Si tienes alguna otra pregunta, no dudes en preguntar.

using System;

namespace p1bpoo.MisClases

{

internal class Chofer : Interfases.IPiloto

{

private string v1;

private int v2;

private TipoLicencia \_tipoLicencia;

private int \_edad;

public string Nombre { get; set; }

public int Edad

{

get { return \_edad; }

set

{

\_edad = value;

ValidarTipoLicencia(); // Validar al cambiar la edad

}

}

public TipoLicencia TipodeLicencia

{

get { return \_tipoLicencia; }

set

{

\_tipoLicencia = value;

ValidarTipoLicencia(); // Validar al cambiar la licencia

}

}

public enum TipoLicencia

{

A, // Motocicletas

B, // Vehículos livianos

C, // Vehículos pesados

M, // Motocicletas

T // Transporte colectivo

}

private void ValidarTipoLicencia()

{

if (Edad >= 25 && \_tipoLicencia != TipoLicencia.T)

throw new ArgumentException("Para transporte colectivo (T) se requiere edad mínima de 25 años.");

else if (Edad >= 21 && \_tipoLicencia != TipoLicencia.C)

throw new ArgumentException("Para vehículos pesados (C) se requiere edad mínima de 21 años.");

else if (Edad >= 18 && \_tipoLicencia != TipoLicencia.B)

throw new ArgumentException("Para vehículos livianos (B) se requiere edad mínima de 18 años.");

else if (Edad >= 17 && \_tipoLicencia != TipoLicencia.A)

throw new ArgumentException("Para motocicletas (A) se requiere edad mínima de 17 años.");

else if (Edad >= 16 && \_tipoLicencia != TipoLicencia.M)

throw new ArgumentException("Para motocicletas (M) se requiere edad mínima de 16 años.");

}

public Chofer(string name, int edad, TipoLicencia licencia)

{

Nombre = name;

Edad = edad;

TipodeLicencia = licencia; // La validación ocurre aquí

}

public Chofer(string v1, int v2)

{

this.v1 = v1;

this.v2 = v2;

}

void IPiloto.mostrarInformacion()

{

Console.WriteLine("El piloto es: {0}", Nombre);

Console.WriteLine("La licencia es {0}", TipodeLicencia);

}

internal void mostrarInformacion()

{

throw new NotImplementedException();

}

}

}