INSTITUTO KHIPU

Semestre III

Sesión 10

PROGRAMA DE ESTUDIOS

DESARROLLO DE SISTEMAS DE INFORMACIÓN

HERRAMIENTAS DE PROGRAMACIÓN – C#

Tema:

PROPIEDADES EN C# - PROPIEDAD GET, PROPIEDAD SET

PROPIEDADES EN C#

Las propiedades en C# son una manera sencilla y controlada de acceder a los datos de una clase, sin necesidad de acceder directamente a los campos (o variables) internos. Esto nos permite tener control sobre cómo se leen o modifican los valores dentro de una clase. Para comprender mejor este tema, es importante ver cómo funcionan los accesores get y set.

¿Por qué usar propiedades en lugar de campos directamente?

En lugar de hacer que los campos de una clase sean públicos (lo que permitiría a cualquier parte del programa modificarlos directamente), se utilizan propiedades para proteger esos datos. Las propiedades permiten:

1. **Validar** los datos antes de asignarlos.
2. **Ocultar** la lógica interna (encapsulación).
3. **Controlar** cómo se acceden o modifican los datos.

¿Cómo funcionan las propiedades con get y set?

1. Propiedad con get: para obtener un valor.

El accesor get nos permite leer el valor de un campo de la clase. En otras palabras, cuando llamamos a la propiedad para obtener su valor, el código dentro de get se ejecuta.

Ejemplo 01

```
public class Persona
{
    private string nombre; // Campo privado

    // Propiedad para obtener el valor de 'nombre'
    public string Nombre
    {
        get { return nombre; }
    }

    public Persona(string nombre)
    {
        this.nombre = nombre;
    }
}
```

Aquí, nombre es un campo privado (no se puede acceder directamente desde fuera de la clase), pero podemos usar la propiedad Nombre para obtener su valor. Si tenemos un objeto de tipo Persona, podemos hacer lo siguiente:

```
Persona persona = new Persona("Ana");
Console.WriteLine(persona.Nombre);  // Imprimirá "Ana"
```

El accesor get ejecuta return nombre; lo que nos permite ver el valor del campo privado nombre desde fuera de la clase.

2. Propiedad con set: para asignar un valor.

El accesor set nos permite **modificar** el valor de un campo. El valor que se pasa a la propiedad cuando intentamos modificarla está representado por la palabra clave value.

Ejemplo 02

```
public class Persona
{
    private string nombre; // Campo privado

    // Propiedad para obtener y modificar el valor de 'nombre'
    public string Nombre
    {
        get { return nombre; }
        set { nombre = value; }
    }

    public Persona(string nombre)
    {
        this.nombre = nombre;
    }
}
```

Aquí, la propiedad Nombre tiene tanto un get como un set. Esto significa que podemos obtener y modificar el valor del campo privado nombre. Por ejemplo:

```
Persona persona = new Persona("Ana");
Console.WriteLine(persona.Nombre);  // Imprime "Ana"
persona.Nombre = "Carlos";          // Cambia el nombre a "Carlos"
Console.WriteLine(persona.Nombre);  // Imprime "Carlos"
```

El código dentro de set asigna el nuevo valor (en este caso, "Carlos") al campo privado nombre.


```
Persona persona = new Persona("Ana");
Console.WriteLine(persona.Nombre);  // Imprime "Ana"
persona.Nombre = "Carlos";          // Cambia el nombre a "Carlos"
Console.WriteLine(persona.Nombre);  // Imprime "Carlos"
```

El código dentro de set asigna el nuevo valor (en este caso, "Carlos") al campo privado nombre.

¿Por qué es útil el accesor set?

- **Validación**: Puedes asegurarte de que se asignen valores correctos a los campos. Por ejemplo, si queremos que la edad siempre sea un número positivo:

Ejemplo

```
public class Persona
{
    private int edad; // Campo privado

    public int Edad
    {
        get { return edad; }
        set
        {
            if (value >= 0)  // Validar que la edad no sea negativa
            {
                edad = value;
            }
            else
            {
                Console.WriteLine("La edad no puede ser negativa.");
            }
        }
    }
}
```

Con este accesor set, evitamos que se asigne un valor negativo a la edad.

Csharp

```
Persona persona = new Persona();
persona.Edad = -5;  // Imprime "La edad no puede ser negativa."
persona.Edad = 25;  // Ahora la edad es 25.
Console.WriteLine(persona.Edad);  // Imprime "25"
```

Propiedades de solo lectura o solo escritura

Propiedades de solo lectura: Si solo tienes un accesor get y no un set, la propiedad se vuelve de solo lectura. Es decir, puedes obtener el valor pero no cambiarlo.

```
public class Persona
{
    private string nombre;

    public string Nombre
    {
        get { return nombre; }
    }
}
```

Propiedades de solo escritura: Si solo tienes un accesor set y no un get, la propiedad es de solo escritura, lo que significa que se puede asignar un valor, pero no se puede leer.

```
public class Persona
{
    private string nombre;

    public string Nombre
    {
        set { nombre = value; }
    }
}
```

Propiedades automáticas

En muchos casos, no necesitamos hacer nada especial en los accesores get o set. Para estos casos, podemos utilizar **propiedades automáticas**, que son una forma más simple de definir propiedades.

```
public class Persona
{
    public string Nombre { get; set; } // Propiedad automática
}
```

Aquí, el compilador se encarga de crear automáticamente el campo privado para Nombre. Esto simplifica el código cuando no necesitas validaciones adicionales.

Resumen:

- **Propiedades** en C# son una forma de controlar cómo se acceden y modifican los datos de una clase.
- El accesor **get** se utiliza para obtener el valor de un campo privado.
- El accesor **set** se utiliza para asignar un valor a un campo privado.
- Puedes agregar **validaciones** dentro del set para asegurar que los datos sean correctos.
- Existen **propiedades automáticas** que simplifican la creación de propiedades cuando no necesitas lógica adicional.

INSTITUTO

KHIPU