

# Encapsulamiento

8

Programación II y Laboratorio de Computación II

Edición 2018

# Encapsulamiento

- Se denomina encapsulamiento al ocultamiento del estado, es decir, de los datos miembro de un objeto de manera que solo se pueda cambiar mediante las operaciones definidas para ese objeto.
- Cada objeto está aislado del exterior, protegiendo a los datos asociados de un objeto contra su modificación por quien no tenga derecho a acceder a ellos.
- La encapsulación se encarga de mantener ocultos los procesos internos que necesita para hacer lo que sea que haga, dándole al programador acceso sólo a lo que necesita.



# Niveles de Encapsulamiento POO

- **Público:** todos pueden acceder a los datos o métodos de una clase que se definen con este nivel, este es el nivel más bajo, esto es lo que tu quieres que la parte externa vea.
- **Protegido:** podemos decir que estás no son de acceso público, solamente son accesibles dentro de su clase y por subclases.
- **Privado:** en este nivel se puede declarar miembros accesibles sólo para la propia clase.

# Propiedades

- Una propiedad es un miembro que proporciona un mecanismo flexible para leer, escribir o calcular el valor de un campo.
- Las propiedades se pueden usar como si fueran miembros de datos públicos, pero en realidad son métodos especiales denominados *descriptores de acceso*.
- Las propiedades permiten que una clase exponga una manera pública de obtener y establecer valores, a la vez que se oculta el código de implementación o validación.

# Propiedades

```
private int totalGoles;  
public int TotalGoles  
{  
    get  
    {  
        return totalGoles;  
    }  
    set  
    {  
        this.totalGoles = value;  
    }  
}
```



# Propiedades

- Para devolver el valor de la propiedad se usa un descriptor de acceso de propiedad get, mientras que para asignar un nuevo valor se emplea un descriptor de acceso de propiedad set.
- La palabra clave value se usa para definir el valor que va a asignar el descriptor de acceso set.
- Las propiedades pueden ser de lectura y escritura (en ambos casos tienen un descriptor de acceso get y set), de solo lectura (tienen un descriptor de acceso get, pero no set) o de solo escritura (tienen un descriptor de acceso set, pero no get).

# Propiedades

```
private int totalGoles;
private int totalPartidos;

public int PromedioGol
{
    get
    {
        int prom = totalGoles / totalPartidos;
        return prom;
    }
}
```

# Indexadores

- Los indexadores permiten a la instancia de una clase ser indexada tal cómo un array.
- La declaración de un indexador luce cómo una propiedad, sólo que:
  - Este recibe parámetros.
  - La palabra clave `this` se utiliza para su definición
- No es necesario indexarlos sólo con un entero.



# Enumerados

```
class EjemploIndexadores
{
    // Declaro un array
    private string[] palabras = new string[100];

    // Defino el indexador
    public string this[int i]
    {
        get { return palabras[i]; }
        set { palabras[i] = value; }
    }
}

//...
EjemploIndexadores ejemplo = new EjemploIndexadores();
ejemplo[0] = "Hola";
ejemplo[1] = "Chau";
Console.WriteLine(ejemplo[0]);
```

# Enumerados

- Son un conjunto propio de constantes con nombre.
- Estos tipos de datos permiten declarar un conjunto de nombres u otros valores literales que definen todos los valores posibles que se pueden asignar a una variable.
- Por dentro, estas constantes están asociadas con el tipo de dato int.
- Normalmente es mejor definir un enum directamente dentro de un espacio de nombres para que todas las clases del espacio de nombres puedan acceder a él con igual comodidad. Sin embargo, un enum también se puede anidar dentro de una clase o struct.

# Enumerados

```
public enum DiasDeLaSemana
{
    Domingo,
    Lunes,
    Martes,
    Miercoles,
    Jueves,
    Viernes,
    Sabado
}
// ...
DiasDeLaSemana dia = DiasDeLaSemana.Lunes;
int i = (int) DiasDeLaSemana.Lunes;

Console.WriteLine(dia); // Mostrará "Lunes"
Console.WriteLine(i);   // Mostrará 1
```



# Enumerados

- A un enumerado se le podrá asignar un valor entero a cualquiera de sus partes, cambiando la numeración por defecto.

```
public enum DiasDeLaSemana
{
    Domingo,
    Lunes = 10,
    Martes,
    Miercoles,
    Jueves,
    Viernes,
    Sabado
}
```

```
// ...
```

```
Console.WriteLine((int) DiasDeLaSemana.Lunes);    // Mostrará 10
```

```
Console.WriteLine((int) DiasDeLaSemana.Miercoles); // Mostrará 12
```