

Clase

Plantilla que define características (propiedades) y comportamientos (métodos) de un objeto.

Ejemplo:

csharp

 Copy  Download

```
public class Persona
{
    // Propiedades
    public string Nombre { get; set; }
    public int Edad { get; }

    // Método
    public void Saludar()
    {
        Console.WriteLine($"Hola, soy {Nombre}");
    }
}
```

public class nombre

propiedades con tipo, nombre + lectura o escritura

procedimiento (void) nombre

Objeto

Instancia concreta de una clase.

Ejemplo:

csharp

 Copy  Download

```
Persona juan = new Persona();  
juan.Nombre = "Juan";  
juan.Saludar(); // Salida: "Hola, soy Juan"
```

estructura de un objeto = new persona()

invoco las propiedades y metodos del objeto

2. Propiedades y Encapsulamiento

Propiedades

- **get** : Lee el valor. que es el get
- **set** : Modifica el valor (puede incluir validaciones). que es el set

Ejemplo con encapsulamiento:

csharp

 Copy  Download

```
private string _nombre;  
public string Nombre  
{  
    get => _nombre;  
    set  
    {  
        if (string.IsNullOrEmpty(value))  
            throw new Exception("El nombre no puede estar vacío.");  
        _nombre = value;  
    }  
}
```

encapsulamiento, protege que nadie modifique desde afuera

esto es un tipo de validación que puede tener el set

Diferencia entre:

- **Atributo:** Variable privada (ej: `_nombre`). una buena practica para nombrar una propiedad privada es el guion bajo
- **Propiedad:** Método público para acceder/modificar el atributo (ej: `Nombre`).

3. Métodos y Niveles de Acceso

Tipos de Métodos

- **Función:** Devuelve un valor (`return`). funcion y su palabra reservada
- **Procedimiento:** No devuelve nada (`void`). procedimiento y su palabra reservada

Ejemplo:

csharp

 Copy  Download

```
// Función
public int Sumar(int a, int b) => a + b;

// Procedimiento
public void MostrarError(string mensaje)
{
    Console.WriteLine($"Error: {mensaje}");
}
```

si es publico o privado + tipo + nombre ()

si es publico o privado + tipo + nombre ()

Niveles de acceso:

desde donde se accede segun si es public / private / protected

- `public` : Accesible desde cualquier clase.
- `private` : Solo dentro de la misma clase.
- `protected` : Clase actual y clases hijas.

4. Herencia y Polimorfismo

Herencia

que es

Una clase hija hereda propiedades/métodos de una clase padre.

Ejemplo:

csharp

 Copy  Download

```
public class Vehiculo
{
    public string Patente { get; set; }
}
```

clase padre

```
public class Auto : Vehiculo
{
    public int Puertas { get; set; }
}
```


clase hija invoca clase padre con los :

Polimorfismo

Objetos diferentes responden al mismo método de manera única.

Ejemplo con método `virtual`:

csharp

 Copy

 Download

```
public class Animal
{
    public virtual void Sonido() => Console.WriteLine("Sonido genérico");
}
```

clase padre

coloca VIRTUAL para que se pueda modificar el metodo heredado

```
public class Gato : Animal
{
    public override void Sonido() => Console.WriteLine("Miau!");
}
```

OVERRIDE me va a permitir "reemplazar" o "modificar" el comportamiento de un miembro heredado en la clase hija

5. Clases Abstractas vs. Interfaces

Clase Abstracta

- No se puede instanciar.
- Puede tener métodos abstractos (obligatorios) o virtuales (opcionales).

Ejemplo:

csharp

Copy Download

```
public abstract class Figura
{
    public abstract double CalcularArea(); // Obligatorio implementar
}
```

clase abstracta representa un concepto

sus metodos tambien son abstractos

Interfaz

- Contrato que obliga a implementar métodos.
- Permite herencia múltiple.

en interfaz no se puede modificar solo se aplica el contrato

csharp

```
public interface IValidacion
{
    bool Validar(string dato);
}

public class ValidadorEmail : IValidacion
{
    public bool Validar(string email) => email.Contains("@");
}
```

6. Manejo de Errores (Try-Catch-Finally)

Casos comunes:

- Archivos no encontrados.
- Divisiones por cero.
- Conexiones fallidas.

sirven para prevenir errores y que el programa no se rompa

catch para prevenir que se rompa

finally, para que se pueda ejecutar todo

Ejemplo:

csharp

 Copy  Download

```
try
{
    int resultado = 10 / int.Parse("0");
}
catch (DivideByZeroException ex)
{
    Console.WriteLine("No se puede dividir por cero.");
}
catch (Exception ex)
{
    Console.WriteLine($"Error inesperado: {ex.Message}");
}
finally
{
    Console.WriteLine("Este bloque siempre se ejecuta.");
}
```


7. Parámetros en Métodos

Tipos:

- **Por valor:** Copia el dato original (no lo modifica).
- **ref :** Modifica el dato original (entrada/salida). para que sirve el parametro ref
- **out :** Solo salida (obligatorio asignar valor). para que sirve el parametro out

Ejemplo:

csharp

 Copy  Download

```
public void Duplicar(ref int numero) => numero *= 2;

// Uso:
int num = 5;
Duplicar(ref num); // num ahora es 10
```

8. Sobrecarga de Métodos

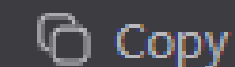
para que sirve (ejemplo de la cafeteria)

Múltiples métodos con el mismo nombre pero diferentes parámetros.

Ejemplo válido:

csharp

mismo nombre, distinto parametro



Copy



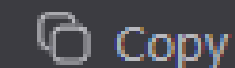
Download

```
public void Mostrar(int valor) { ... }  
public void Mostrar(string texto) { ... } // Distinto tipo de parametro
```

Ejemplo inválido:

csharp

ESTO ESTA ERRADO !! porque son los mismos parametros



Copy



Download

```
public void Mostrar(int a) { ... }  
public void Mostrar(int b) { ... } // Mismo tipo y cantidad
```

9. Enumeraciones (Enum)

Lista de valores fijos para mejorar legibilidad.

Ejemplo:

que es enum (valores fijos constantes, de preferencia POCOS)

csharp

 Copy  Download

```
public enum GrosorMina
{
    Fino = 1,
    Medio = 2,
    Grueso = 3
}
```

```
GrosorMina mina = GrosorMina.Medio;
```