




Clase 04

SOBRECARGA



¿Qué es la sobrecarga?

Es una técnica que nos permite definir varios miembros de una clase con el mismo nombre, siempre y cuando reciban un conjunto distinto de parámetros



MIEMBROS SOBRECARGABLES

01.

METODOS

02.

CONSTRUCTORES

03.

OPERADORES

04.

INDEXADORES

The background features four large, stylized geometric shapes in the corners, resembling arrowheads or chevrons. The top-left and bottom-right shapes are light gray, while the top-right and bottom-left shapes are dark gray. They are arranged symmetrically, pointing towards the center of the slide.

01.

SOBRECARGA DE MÉTODOS

SOBRECARGA DE MÉTODOS

Utilizamos la sobrecarga para crear varios métodos con el mismo nombre pero con distintos parámetros:

CANTIDAD DE PARÁMETROS

```
public int Sumar(int operando1,int operando2){}  
  
public int Sumar(int operando1,int operando2, int operando3){}
```

TIPO DE LOS PARÁMETROS

```
public int Sumar(int operando1,int operando2 ){}  
  
public int Sumar(float operando1,float operando2){}
```

ORDEN DE LOS PARÁMETROS

```
public int Sumar(int operando1,float operando2 ){}  
  
public int Sumar(float operando1,int operando2){}
```

BUENAS PRACTICAS



Utilizar nombres de parámetros descriptivos



Evitar variar arbitrariamente los nombres de los parámetros en las sobrecargas. Si un parámetro en una sobrecarga representa la misma entrada que un parámetro en otra sobrecarga, los parámetros deben tener el mismo nombre.



Evitar modificar el orden de los parámetros en miembros sobrecargados. Los parámetros con el mismo nombre deben aparecer en la misma posición en todas las sobrecargas.



NO tenga sobrecargas con parámetros en la misma posición y tipos similares pero con semántica diferente.



02.

SOBRECARGA DE CONSTRUCTORES

SOBRECARGA DE CONSTRUCTORES

La sobrecarga de constructores permite a los objetos inicializarse de distintas formas.

```
public Impresora( ){}  
  
public Impresora(string marca, string modelo){}  
  
public Impresora(string marca, string modelo, bool multifuncional, int hpm) :  
this (marca, modelo){}
```


The background features several large, stylized geometric shapes in shades of gray. On the left, a dark gray chevron points right. Above it, a light gray triangle points down. On the right, a dark gray triangle points down, and below it, a light gray chevron points left. The central text is positioned between these shapes.

03.

SOBRECARGA DE OPERADORES

OPERADORES

unarios	+ - ! ~ ++ -- true false	✓
binarios	+ - * / % & ^ << >> == != > < >= <=	✓
Unarios y binarios	=, &&, , ??, ?:, =>, <i>checked, unchecked, new, typeof, default, as, is</i>	✗

SOBRECARGA DE OPERADORES

Una clase puede proporcionar la implementación personalizada de una operación en caso de que uno o ambos operandos sean del tipo de la clase

```
class MiClase
{
    public static [tipo_dato] operator [operrador](MiClase c, [Tipo_Dato x]){}
}
```

The background features several large, stylized geometric shapes. On the left, a dark gray chevron points right. Above it, a light gray chevron points down. On the right, a dark gray chevron points down, and below it, a light gray chevron points left. These shapes are composed of parallel lines, creating a sense of depth and movement.

04.

OPERADORES DE CONVERSION

SOBRECARGA DE OPERADORES DE CONVERSIÓN

Una Clase puede definir una conversión implícita o explícita personalizada desde o hacia otro tipo.

IMPLICITA

```
[acceso] static implicit operator nombreTipo(Tipo a)
{
    //...
}
```

EXPLICITA

```
[acceso] static explicit operator nombreTipo(Tipo a)
{
    //...
}
```