POLIMORFISMO

Sobrecarga de Métodos.-

La sobrecarga de métodos se trata de crear dos o más métodos, con en el mismo nombre.

Ejemplo.-

```
using System;
public class numeros {
    private int x;
    //Aqui tenemos el constructor de numeros
    public numeros () {
        x = 0;
    }
    //Aqui sobrecargamos el constructor de numeros aumentado un parametro
    public numeros (int y) {
        x = y;
    }
    //Aqui tenemos el metodo inc que permite incremetar a la variable x + 5
    public void inc(){
        x += 5;
        Console.WriteLine("Valor de X = "+x);
    }
    //Aquí tenemos la sobrecarga del método inc que permite incrementar a la
    //variable + x
    public void inc(int x){
        this.x += x;
        Console.WriteLine("Valor de X = "+this.x);
    }
    //Aquí tenemos la sobrecarga del método inc que permite incrementar a la
    //variable + otra clase numeros
    public void inc(numeros num)
    {
        this.x += num.x;
        Console.WriteLine("Valor de X = " + this.x);
    }
    public void mostrar(){
        Console.WriteLine("Valor de X = "+x);
    }
    //Aqui tenemos el metodo modulo que permite sacar el modulo de la variable % 10
    public int modulo() {
        return x % 10;
    }
    //Aquí tenemos la sobrecarga del método modulo que permite sacar el modulo de la
    //variable % y
    public int modulo(int y)
    {
        return x % y;
    }
}
```

Como se puede ver en el ejemplo anterior la sobrecargar de métodos nos permite escribir métodos (Procesos, Funciones y Métodos Especiales como constructores) con el mismo nombre, pero la única condición que nos pide es colocar distintos parámetros a cada método.

Sobrecarga de Operadores.-

La sobrecarga de operadores nos permite sobrecarga o cambiar la funcionalidad de un operador:

- **UNARIOS:** ++, --, !, ~
- **BINARIOS:** +, -, *, /, %, &, |, ^, <<, >>, <, >, <=, >=, ==, !=

Los siguientes operadores deben ser declarado en conjunto son:

- Tanto el mayor que y el menor que deben ser declarados a la vez ">" y "<"
- Tanto el mayor igual que y el menor igual que deben ser declarados a la vez ">=" y "<="
- Tanto el igual a y el distinto de deben ser declarados a la vez "==" y "!="

Las siguientes líneas de comandos son para la instancia de la sobrecarga de operadores unarios:

- ! y ~

```
public static TipoDatoRetornar operator OperadorUnarioSobrecargar (NomClase var){
       :
       return TipoDato;
}
```

- ++ y --

```
public static NomClase operator OperadorUnarioSobrecargar (NomClase var){
:
return var;
}
```

Y en el principal o otra clase se invoca de la siguiente manera

- ! y ~

```
TipoDato=(! o ~) NomVarClase;
```

- ++ y --

```
NomVarClase(++ o --);
```

Las siguientes líneas de comandos Las siguientes líneas de comandos son para la instancia de la sobrecarga de operadores Binarios:

```
public static TipoDatoRetornar operator OperadorBinarioSobrecargar (NomClase var,
TipoDato var1){
       :
       return TipoDato;
}
```

Y en el principal o otra clase se invoca de la siguiente manera

```
TipoDato = NomVarClase OPERADOR TipoDato;
```

Usando el ejemplo anterior sobrecargaremos los siguientes operadores:

- ++: para leer un nuevo numero
- ~: para mostrar
- *: para multiplica por la variable y, si es que el número es mayor 15
- == y != comparar si 2 clases de números son iguales

```
using System;
public class numeros {
    :
    //Sobrecargando ++
    public static numeros operator ++ (numeros x){
        x.x = int.Parse(Console.ReadLine());
        return x;
    }
    //Sobrecargando ~
    public static double operator ~(numeros num)
    {
```

```
        Console.WriteLine("Valor de X = " + num.x);
        return 5.5;
    }
    //Sobrecargando *
    public static int operator *(numeros num, int y) {
        if (num.x > 15)
        {
            num.x*= y;
        }
        return num.x;
    }
    //Sobrecargando ==
    public static bool operator ==(numeros n, numeros m)
    {
        if (n.x == m.x) {
            return true;
        }
        return false;
    }
    //Sobrecargando !=
    public static bool operator !=(numeros n, numeros m)
    {
        if (n.x != m.x)
        {
            return true;
        }
        return false;
    }
}
using System;
class Program
{
    static void Main(string[] args){
        numeros a = new numeros();
        numeros b = new numeros(7);
        Console.WriteLine("SOBRECARGA DE METODOS ----------------------------");
        Console.Write("Numeros A: ");
        a.mostrar();
        Console.Write("Numeros B: ");
        b.mostrar();
        Console.Write("Numeros A + 5: ");
        a.inc();
        Console.Write("Numeros B + y: ");
        b.inc(9);
        Console.Write("Numeros A + B: ");
        a.inc(b);
        Console.WriteLine("Numeros A % 10: " + a.modulo());
        Console.WriteLine("Numeros B % y: " + b.modulo(3));
        Console.WriteLine("SOBRECARGA DE OPERADORES --------------------------");
        a++;
        Console.Write("Numeros A: ");
        double x = ~a;
        Console.WriteLine("Resultado de la Sobrecarga Operador *: "+a * 5);
        if (a == b) {
            Console.WriteLine("Son iguales las clases numeros");
        }
        if (a != b){
            Console.WriteLine("Son Diferetes las clases numeros");
        }
        Console.ReadKey();
    }
}
```