**Módulo: 02 - Características del Lenguaje C#**

**Tutorial: Interfaces**

**Ejemplo 1**

1. Crear la siguiente declaración de Inteface:

//1 . Declaracion de Interface y explicacion

interface IMiInterface

{

void Mostrar();

string Nombre { set; get; }

int Edad {get;}

}

1. Implementar la interface con un clase como la siguiente.

//2. Implementar una interfaz en una clase

public class Persona : IMiInterface

{

public void Mostrar()

{

Console.WriteLine(Nombre);

}

private string nombreValor;

public string Nombre

{

get

{

return nombreValor;

}

set

{

nombreValor = value;

}

}

private int edadValor;

public int Edad

{

get { return edadValor; }

}

}

1. Compilar y ejecutar.
2. Agregar un método a la interfaz y compilar. Observar el resultado.

class Empleado

**Ejemplo 2**

1. Crear un objeto para usar la interface implementada.

class Program

{

static void Main(string[] args)

{

//3. Uso de la Interfaz

Persona objPersona = new Persona();

objPersona.Nombre = "Alejo";

objPersona.Mostrar();

}

}

1. Compilar y ejecutar y ver el resultado.
2. Crear un variable interface.

class Program

{

static void Main(string[] args)

{

//3. Uso de la Interfaz

Persona objPersona = new Persona();

objPersona.Nombre = "Alejo";

objPersona.Mostrar();

//4. Variables Interfaz

IMiInterface i = objPersona;

i.Mostrar();

}

}

1. Compilar y ejecutar y ver el resultado.
2. Agregar el siguiente miembro a la clase persona. Y observar su visibilidad en la variable i y en la variable Persona.

public string Apellidos = "Mis Apellidos";

public void Mostrar()

{

Console.WriteLine(Nombre + " " + Apellidos);

}

1. Ejecutar y ver el resultado.

**Ejemplo 3**

1. Eliminar el miembro Apellido de la clase personas.
2. Compilar y ejecutar.
3. Crear la siguiente clase:

public class CaractPersonas

{

private string apellidosValor = "Apellidos";

public string Apellidos

{

get

{

return apellidosValor;

}

set

{

apellidosValor = value;

}

}

}

1. Modificar la declaración de la clase persona para que herede de la nueva clase.

public class Persona : CaractPersonas, IMiInterface

1. Compilar y ejecutar. Observar el resultado.
2. Crear un objeto de la siguiente manera, con el código asociado

//6. Variables Objeto

CaractPersonas a = objPersona;

a.Apellidos = "Otros Apellidos";

objPersona.Mostrar();

1. Compilar y ejecutar. Observar el resultado.

**Ejemplo 4**

1. Uso de interfaces provista por .NET. Definimos una clase para que pueda se clasificada por las herramientas provista por . NET, en este caso usaremos la interfaz IComparable, porque el método Sort (método para manejo de Listas) para clasificar llama al método CompareTo de IComparable, el cual lo vamos a implementar en nuestra clase.

public class Empleado : IComparable

{

public string Nombre;

public Empleado() { }

public Empleado(string nombre)

{

this.Nombre = nombre;

}

public int CompareTo(object obj)

{

if (obj is Empleado)

{

Empleado e1 = ((Empleado)obj);

return String.Compare(this.Nombre, e1.Nombre);

}

else

return 0;

}

}

1. Para usar la clase y hacer una ejemplo de clasificación se genera el siguiente código.

class Program

{

static void Main(string[] args)

{

// Una colección de datos del tipo Empleado

List<Empleado> empleados = new List<Empleado>();

// Añadimos varios empleados a la colección

empleados.Add(new Empleado("Pepe"));

empleados.Add(new Empleado("Bernardo"));

empleados.Add(new Empleado("Juan"));

empleados.Add(new Empleado("Ana"));

// La clasificamos

empleados.Sort();

// Mostramos los nombres clasificados

foreach (Empleado e1 in empleados)

Console.WriteLine(e1.Nombre);

Console.ReadLine();

}

}

1. Ejecutar y Compilar.
2. Si es desea hacer la clasificación de otra manera se debe modifica el método CompareTo, teniendo en cuenta de que un valor cero indica que los dos objetos son iguales, un valor menor que cero que nuestro objeto debe ir antes que el indicado en el parámetro y un valor mayor de cero para indicar que nuestro objeto debe ir después del indicado.