



PRÁCTICA 10: HERENCIA EN PROGRAMACIÓN ORIENTADA A OBJETOS

Objetivos:

- ✓ Construir clases base y clases derivadas
- ✓ Programar aplicaciones para implementar herencia con programación orientada a objetos

1. Ejecutar Visual Studio .NET
2. Crear un nuevo proyecto de tipo Aplicación de Windows Forms (Windows Form App)
3. Crear un formulario para cada uno de los siguientes ejemplos.

Ejemplo: creación e implementación de una jerarquía de herencia

1. Crear la clase **Empleado**:
 - a. Menú Project (Proyecto)
 - b. Opción Add Class... (Agregar Clase...)
 - c. Asignar a la clase el nombre de **Empleado.cs**
 - d. Clic en el botón Add (Agregar)
 - e. Agregar el siguiente código a la clase **Empleado**

```
class Empleado
{
    //campos
    private string nombre;
    private string dui;
    private string nit;

    //método constructor
    2 references
    public Empleado(string nombre, string dui, string nit)
    {
        this.nombre = nombre;
        this.dui = dui;
        this.nit = nit;
    }

    //propiedades
    2 references
    public string Nombre { get => nombre; set => nombre = value; }
    2 references
    public string Dui { get => dui; set => dui = value; }
    2 references
    public string Nit { get => nit; set => nit = value; }
}
```

2. Crear la clase **EmpleadoAsalariado**:

- a. Menú Project (Proyecto)
- b. Opción Add Class... (Agregar Clase...)
- c. Asignar a la clase el nombre de **EmpleadoAsalariado.cs**
- d. Clic en el botón Add (Agregar)
- e. Agregar el siguiente código a la clase **EmpleadoAsalariado**

```
class EmpleadoAsalariado : Empleado
{
    //campo
    private decimal sueldo;

    //método constructor
    1 reference
    public EmpleadoAsalariado(string nombre, string dui, string nit, decimal sueldo)
        : base(nombre, dui, nit)
    {
        this.sueldo = sueldo;
    }

    //propiedad
    1 reference
    public decimal Sueldo { get => sueldo; set => sueldo = value; }
}
```

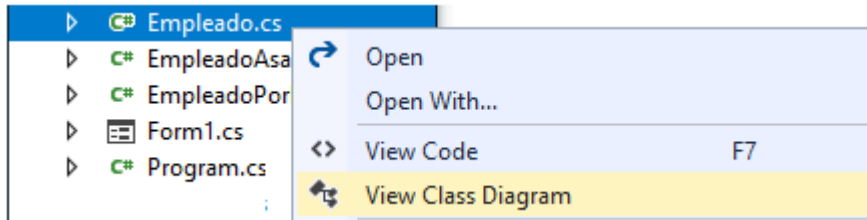
3. Crear la clase **EmpleadoPorHora**:

- a. Menú Project (Proyecto)
- b. Opción Add Class... (Agregar Clase...)
- c. Asignar a la clase el nombre de **EmpleadoPorHora.cs**
- d. Clic en el botón Add (Agregar)
- e. Agregar el siguiente código a la clase **EmpleadoPorHora**

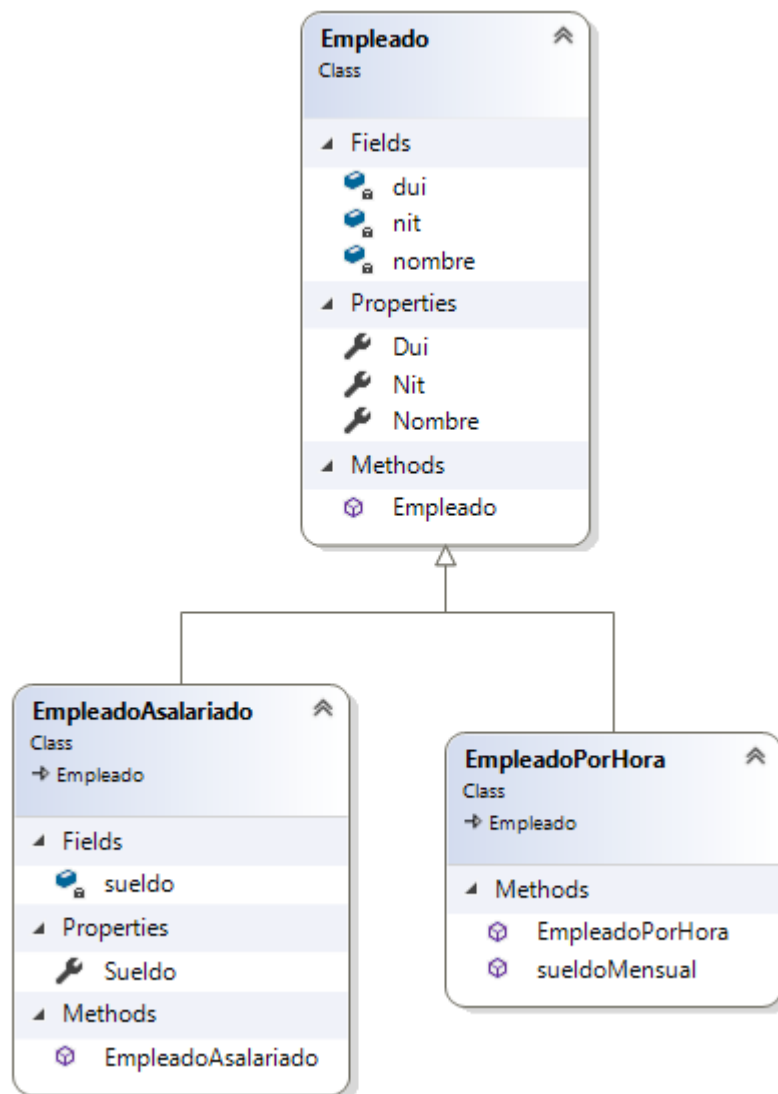
```
class EmpleadoPorHora : Empleado
{
    //método constructor
    1 reference
    public EmpleadoPorHora(string nombre, string dui, string nit)
        : base(nombre, dui, nit)
    {
    }

    //método
    1 reference
    public decimal sueldoMensual(int horas, decimal valor)
    {
        return horas * valor;
    }
}
```

4. Para visualizar el diagrama de clase, en el Explorador de Soluciones, hacer clic derecho sobre la clase **Empleado** y luego hacer clic en la opción **View Class Diagram** (Ver Diagrama de Clases)



- a. Diagrama de la jerarquía de clases



5. Crear el formulario empleados, con la siguiente estructura:

Empleados

Nombre

DUI

NIT

Tipo

☒ Asalariado

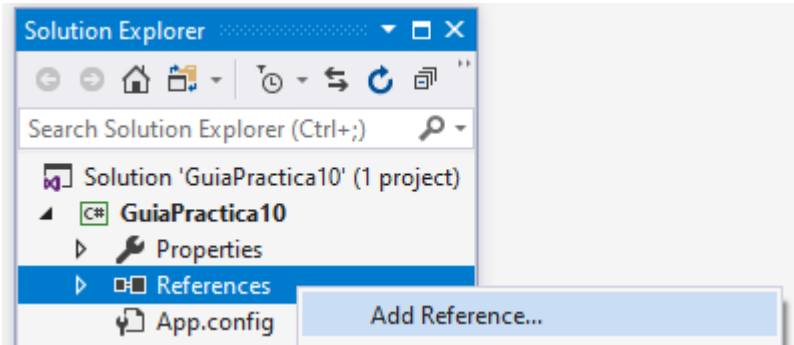
☐ Por hora

Agregar

Nombre	DUI	NIT	Tipo	Sueldo a pagar
--------	-----	-----	------	----------------

Control	Name	Text
TextBox1	txtNombre	
MaskedTextBox1	mktDui	
MaskedTextBox2	mktNit	
RadioButton1	radAsalariado	Asalariado
RadioButton2	radPorHora	Por hora
Button1	btnAgregar	Agregar
DataGridView1	dgvEmpleados	

Agregar una referencia a Visual Basic, en el panel Add Reference...



Seleccionar Microsoft.VisualBasic

<input checked="" type="checkbox"/>	Microsoft.VisualBasic	10.0.0.0
-------------------------------------	-----------------------	----------

Agregar el siguiente código al evento clic del botón Agregar

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    if (txtNombre.Text == String.Empty)
    {
        MessageBox.Show("Ingrese el nombre");
        txtNombre.Focus();
    }
    else if (!mktDui.MaskFull)
    {
        MessageBox.Show("Ingrese correctamente el DUI");
        mktDui.Focus();
        mktDui.SelectionStart = 0;
        mktDui.SelectionLength = mktDui.TextLength;
    }
    else if (!mktNit.MaskFull)
    {
        MessageBox.Show("Ingrese correctamente el NIT");
        mktNit.Focus();
        mktNit.SelectionStart = 0;
        mktNit.SelectionLength = mktNit.TextLength;
    }
    else
    {
        if (radAsalariado.Checked)
        {
            decimal sueldo;
            if (decimal.TryParse(Microsoft.VisualBasic.Interaction.InputBox(
                "Ingrese el sueldo"), out sueldo))
            {
                EmpleadoAsalariado emp = new
                    EmpleadoAsalariado(txtNombre.Text, mktDui.Text, mktNit.Text, sueldo);
                dgvEmpleados.Rows.Add(emp.Nombre, emp.Dui, emp.Nit, "Asalariado", emp.Sueldo);
                dgvEmpleados.ClearSelection();
                txtNombre.Clear();
                mktDui.Clear();
                mktNit.Clear();
                txtNombre.Focus();
            }
            else
            {
                MessageBox.Show("Ingrese correctamente el sueldo a pagar");
            }
        }
    }
}
```

```

else
{
    EmpleadoPorHora emp = new EmpleadoPorHora(txtNombre.Text, mktDui.Text, mktNit.Text);
    int horas;
    decimal valor;
    if (int.TryParse(Interaction.InputBox(
        "Ingrese el numero de horas"), out horas) &&
        decimal.TryParse(Interaction.InputBox(
            "Ingrese el valor de la hora"), out valor))
    {
        decimal pago;
        pago = emp.sueldoMensual(horas, valor);
        dgvEmpleados.Rows.Add(emp.Nombre, emp.Dui, emp.Nit, "Por hora", pago);
        dgvEmpleados.ClearSelection();
        txtNombre.Clear();
        mktDui.Clear();
        mktNit.Clear();
        txtNombre.Focus();
    }
    else
    {
        MessageBox.Show("Ingrese correctamente las horas y el valor a pagar");
    }
}
}
}

```

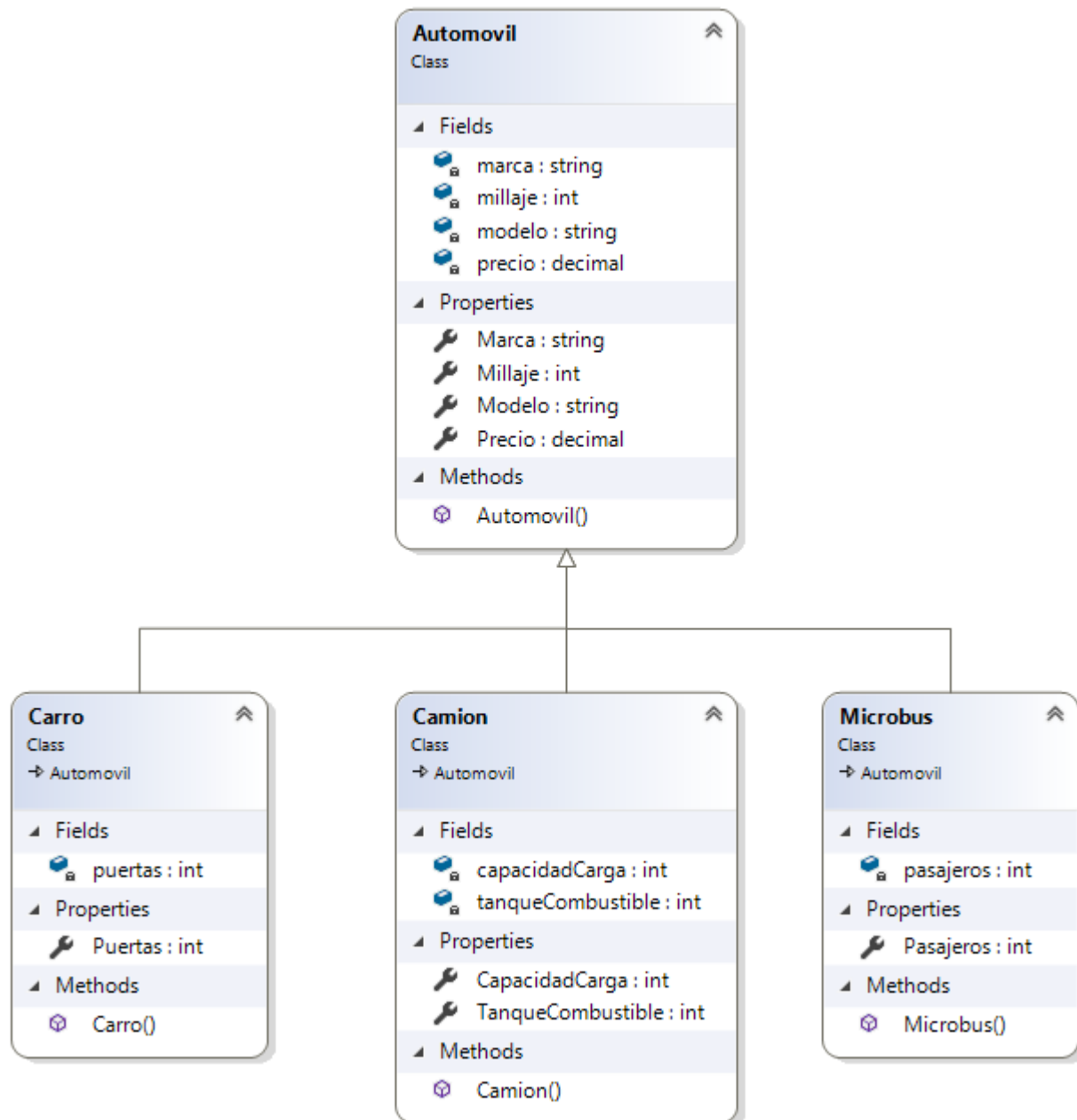
Probar el funcionamiento del formulario. (F5)



Agregar un nuevo proyecto de nombre **práctica10** a la solución del portafolio en Visual Studio .NET

Indicaciones: agregar a la solución un proyecto de C# de tipo Aplicación Windows Forms y programar las siguientes soluciones.

1. Elaborar la siguiente jerarquía de clases.

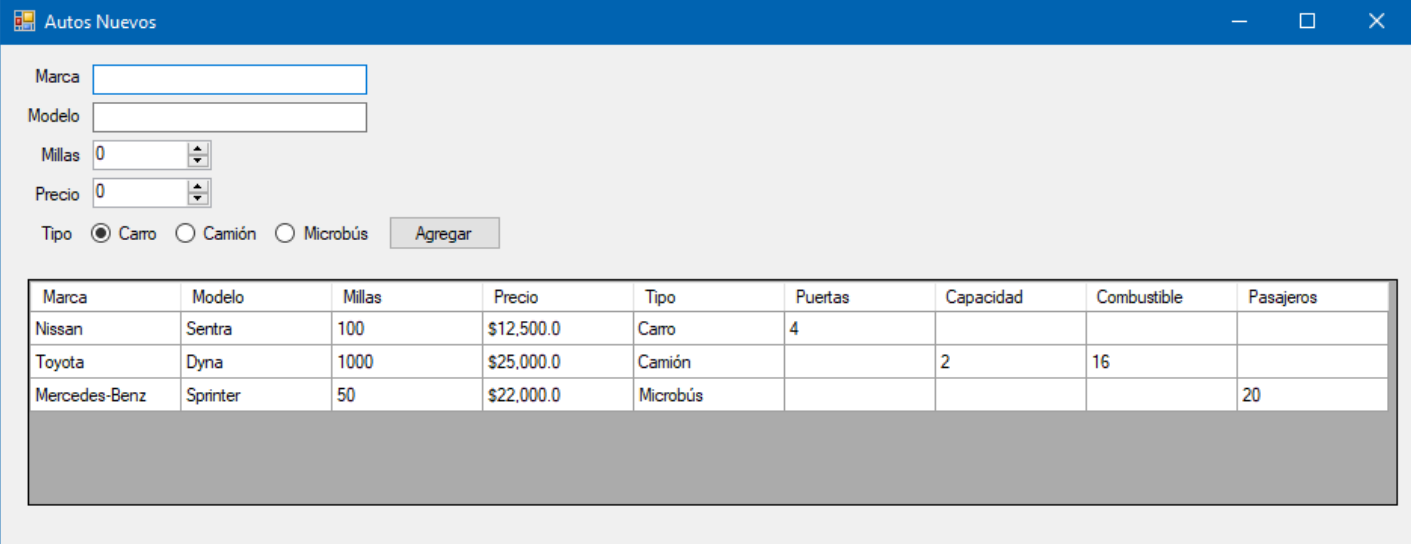


Requerimientos:

La clase **Automovil**, tendrá un método constructor con parámetros (marca, modelo, millaje y precio)

Las clases derivadas (**Carro**, **Camion** y **Microbus**) tendrán un método constructor con los parámetros de la clase base más sus propios parámetros.

2. Programar la siguiente aplicación, implementando la jerarquía de clases de Automovil



Marca	Modelo	Millas	Precio	Tipo	Puertas	Capacidad	Combustible	Pasajeros
Nissan	Sentra	100	\$12,500.0	Carro	4			
Toyota	Dyna	1000	\$25,000.0	Camión		2	16	
Mercedes-Benz	Sprinter	50	\$22,000.0	Microbús				20

Requerimientos:

Validar la captura de datos generales: marca, modelo, millas y precio

Utilizar InputBox o TextBox (ocultos, que se harán visibles cuando sean necesarios) para la captura de datos particulares de cada tipo de vehículo, validando la captura de datos