UNIVERSIDAD CATÓLICA DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
INGENIERÍA EN SISTEMAS INFORMÁTICOS
CICLO: I - 2020
MATERIA: PROGRAMACIÓN II
DOCENTE: ING. GIOVANNI ACOSTA HENRIQUEZ (giovanni.acosta@catolica.edu.sv)

PRÁCTICA 8: INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS

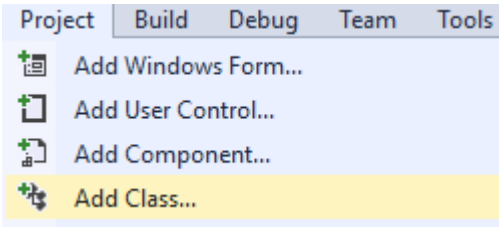
Objetivos:

- ✓ Crear clases con campos y métodos (constructores, acceso y modificación)
- ✓ Programar aplicaciones para implementar las clases desarrolladas
- ✓ Serializar colecciones de objetos

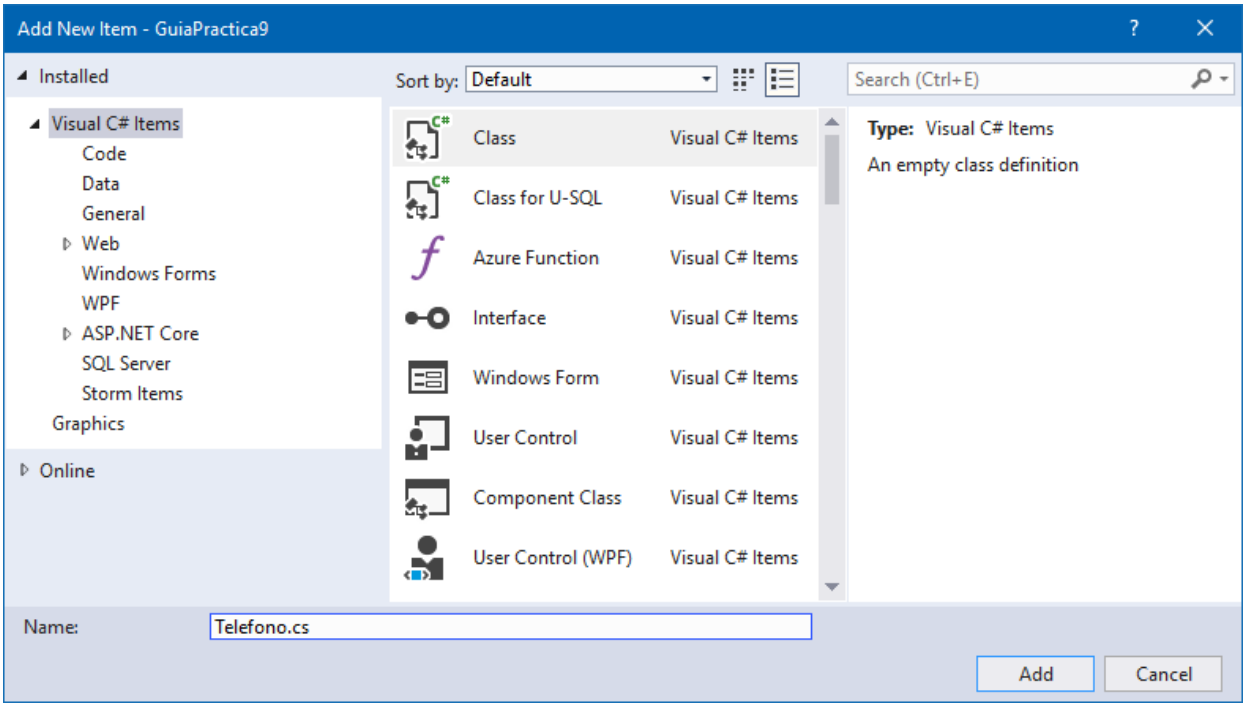
1. Ejecutar Visual Studio .NET
2. Crear un nuevo proyecto de tipo Aplicación de Windows Forms (Windows Form App)
3. Crear un formulario para cada uno de los siguientes ejemplos.

Ejemplo 1: creación e implementación de la clase **Telefono**

1. Crear la clase Teléfono:
   a. Menú Project (Proyecto)
   b. Opción Add Class… (Agregar Clase…)



   c. Asignar a la clase el nombre de **Telefono.cs**



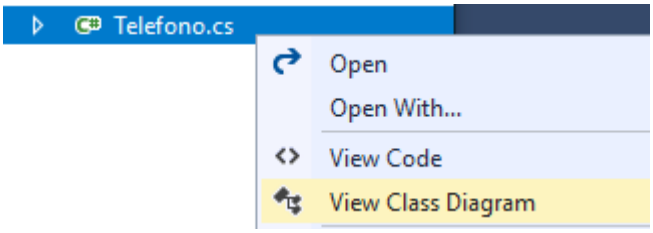
   d. Clic en el botón Add (Agregar)

e. Agregar el siguiente código a la clase **Telefono**

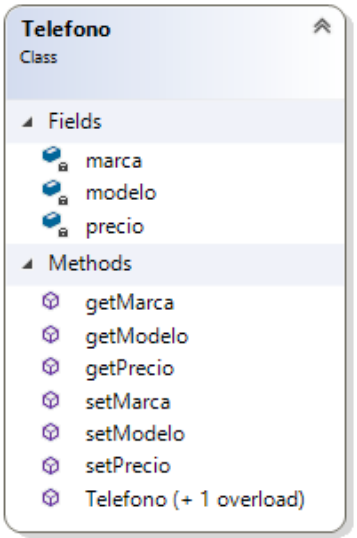
```
class Telefono
{
    // Campos marca, modelo y precio del teléfono
    private string marca;
    private string modelo;
    private decimal precio;

    // Método constructor sin parámetros
    0 references
    public Telefono()
    {
        marca = "";
        modelo = "";
        precio = 0;
    }

    // Método constructor con parámetros
    1 reference
    public Telefono(string marca, string modelo, decimal precio)
    {
        this.marca = marca;
        this.modelo = modelo;
        this.precio = precio;
    }

    // Método para retornar la marca
    1 reference
    public string getMarca()
    {
        return marca;
    }

    // Método para establecer la marca
    0 references
    public void setMarca(string marca)
    {
        this.marca = marca;
    }

    // Método para retornar el modelo
    1 reference
    public string getModelo()
    {
        return modelo;
    }
```

```
    // Método para establecer el modelo
    0 references
    public void setModelo(string modelo)
    {
        this.modelo = modelo;
    }

    // Método para retornar el precio
    1 reference
    public decimal getPrecio()
    {
        return precio;
    }

    // Método para establecer el precio
    0 references
    public void setPrecio(decimal precio)
    {
        if (precio >= 0)
        {
            this.precio = precio;
        }
    }
}
```

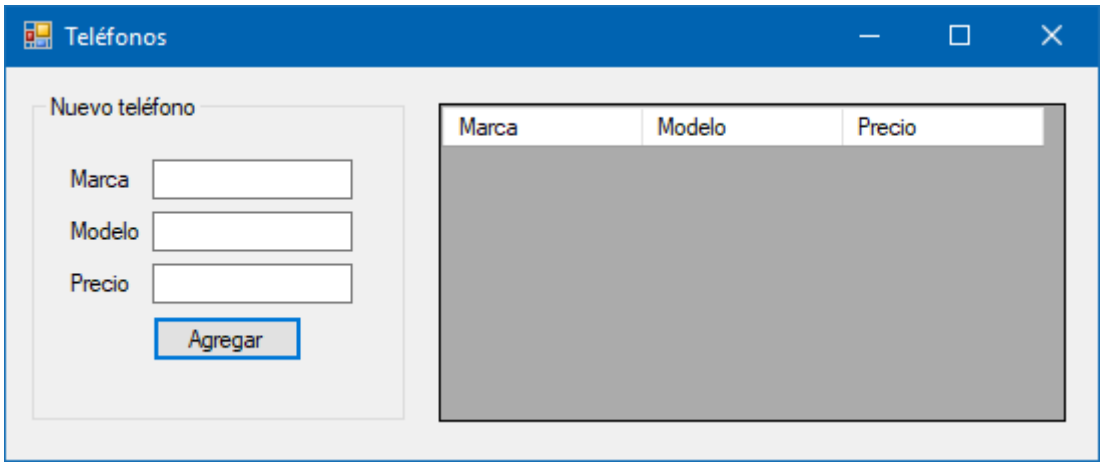
f. Para visualizar el diagrama de clase, en el Explorador de Soluciones, hacer clic derecho sobre la clase **Telefono** y luego hacer clic en la opción **View Class Diagram** (Ver Diagrama de Clases)



g. Diagrama de clase **Telefono**



2. Crear el formulario Teléfonos, con la siguiente estructura:



Control	Name	Text
GroupBox1		Nuevo teléfono
Label1		Marca
Label2		Modelo
Label3		Precio
TextBox1	txtMarca	Marca
TextBox2	txtModelo	Modelo
TextBox3	txtPrecio	Precio
Button1	btnAgregar	Agregar
DataGridView1	dgvTelefonos	

Agregar el espacio de nombres: `using System.Collections;`

En las declaraciones generales del formulario crear el List<Telefono>:

```
List<Telefono> listaTelefonos = new List<Telefono>();
```

Agregar el siguiente código al evento **click** del botón Agregar:

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    string marca;
    string modelo;
    decimal precio;

    if (txtMarca.Text == String.Empty)
    {
        MessageBox.Show("Ingrese la marca", "Teléfonos",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
        txtMarca.Focus();
    }
    else if (txtModelo.Text == String.Empty)
    {
        MessageBox.Show("Ingrese el modelo", "Teléfonos",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
        txtModelo.Focus();
    }
```

```
    else if (!decimal.TryParse(txtPrecio.Text, out precio) || precio<0)
    {
        MessageBox.Show("Ingrese el precio correctamente", "Teléfonos",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
        txtPrecio.Focus();
        txtPrecio.SelectionStart = 0;
        txtPrecio.SelectionLength = txtPrecio.TextLength;
    }
    else
    {
        marca = txtMarca.Text;
        modelo = txtModelo.Text;
        precio = Convert.ToDecimal(txtPrecio.Text);
        Telefono tel = new Telefono(marca, modelo, precio);
        listaTelefonos.Add(tel);
        actualizarGrid(ref dgvTelefonos);
        txtMarca.Clear();
        txtModelo.Clear();
        txtPrecio.Clear();
        txtMarca.Focus();
        MessageBox.Show("El teléfono fue creado correctamente", "Teléfonos",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

Crear el método personalizado actualizarGrid:

```
public void actualizarGrid(ref DataGridView grid)
{
    grid.Rows.Clear();
    foreach (Telefono tel in listaTelefonos)
    {
        grid.Rows.Add(tel.getMarca(), tel.getModelo(), tel.getPrecio());
    }
    grid.ClearSelection();
}
```

Probar el funcionamiento del formulario. (F5)

Ejemplo 2: creación e implementación de la clase **Trabajador** para ser serializada en un **archivo binario**

1. Crear la clase **Trabajador**:

```
[Serializable]
class Trabajador
{
    private string nombre;
    private DateTime fechaNac;
    private string sexo;
    private string estadoCivil;
    private string profesion;
    private decimal sueldo;

    public Trabajador(string nombre, DateTime fechaNac, string sexo,
        string estadoCivil, string profesion, decimal sueldo)
    {
        this.nombre = nombre;
        this.fechaNac = fechaNac;
        this.sexo = sexo;
        this.estadoCivil = estadoCivil;
        this.profesion = profesion;
        this.sueldo = sueldo;
    }

    public string Nombre { get => nombre; set => nombre = value; }
    public DateTime FechaNac { get => fechaNac; set => fechaNac = value; }
    public string Sexo { get => sexo; set => sexo = value; }
    public string EstadoCivil { get => estadoCivil; set => estadoCivil = value; }
    public string Profesion { get => profesion; set => profesion = value; }
    public decimal Sueldo { get => sueldo; set => sueldo = value; }
}
```

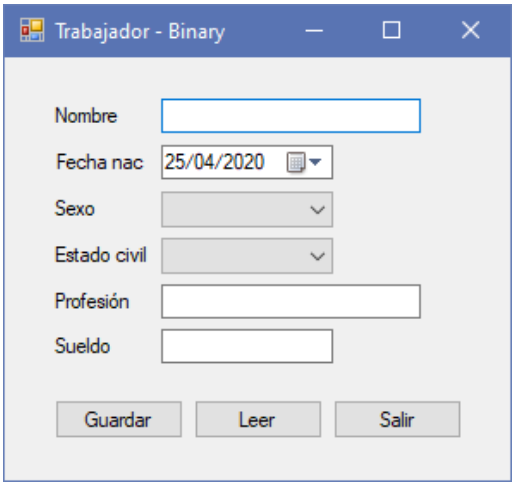
2. Crear la clase **Archivo**: (*servirá para serializar y deserializar objetos*)

   Agregar las siguientes clases:

```
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;


class Archivo
{
    private string rutaArchivo;

    public Archivo(string rutaArchivo)
    {
        this.rutaArchivo = rutaArchivo;
    }
```

```
    public bool GuardarArchivo(Object objeto)
    {
        try
        {
            BinaryFormatter formatter = new BinaryFormatter();
            FileStream fs = new FileStream(rutaArchivo, FileMode.Create, FileAccess.Write);
            formatter.Serialize(fs, objeto);
            fs.Close();
            return true;
        }
        catch (Exception)
        {
            return false;
        }
    }

    public Object LeerArchivo()
    {
        try
        {
            BinaryFormatter formatter = new BinaryFormatter();
            FileStream fs = new FileStream(rutaArchivo, FileMode.Open, FileAccess.Read);
            Object objeto = (Object)formatter.Deserialize(fs);
            fs.Close();
            return objeto;
        }
        catch (Exception)
        {
            return null;
        }
    }
}
```

3. Crear el formulario Empleados, con la siguiente estructura:



Control	Name	Text
textBox1	txtNombre	
dateTimePicker	dtpFechaNac	
comboBox1	cboSexo	
comboBox2	cboEstadoCivil	
textBox2	txtProfesion	
textBox3	txtSueldo	
button1	btnGuardar	Guardar
button2	btnLeer	Leer
button3	btnSalir	Salir

Modificar las siguientes propiedades del control **errorProvider**:

Propiedad	Valor
Name	errMensaje
BlinkStyle	NeverBlink

Agregar los siguientes espacios de nombres:

Agregar el siguiente código en las declaraciones generales del formulario:

```
Archivo archivo = new Archivo("trabajador.bin");
```

Agregar el siguiente código al evento **click** del botón Guardar:

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    bool completado = true;
    if (txtNombre.Text == string.Empty)
    {
        errMensaje.SetError(txtNombre, "Ingrese el nombre");
        completado = false;
    }
    else
        errMensaje.SetError(txtNombre, null);
    if (dtpFechaNac.Value >= DateTime.Now.Date || dtpFechaNac.Value.Date.Year<1990)
    {
        errMensaje.SetError(dtpFechaNac, "Ingrese una fecha válida");
        completado = false;
    }
    else
        errMensaje.SetError(dtpFechaNac, null);
    if (cboSexo.SelectedIndex < 0)
    {
        errMensaje.SetError(cboSexo, "Seleccione el sexo");
        completado = false;
    }
    else
        errMensaje.SetError(cboSexo, null);
    if (cboEstadoCivil.SelectedIndex < 0)
    {
        errMensaje.SetError(cboEstadoCivil, "Seleccione el estado civil");
        completado = false;
    }
    else
        errMensaje.SetError(cboEstadoCivil, null);
    if (txtProfesion.Text == string.Empty)
    {
        errMensaje.SetError(txtProfesion, "Ingrese la profesión");
        completado = false;
    }
    else
        errMensaje.SetError(txtProfesion, null);
    decimal sueldo;
    if (!decimal.TryParse(txtSueldo.Text, out sueldo))
    {
        errMensaje.SetError(txtSueldo, "Ingrese un sueldo válido");
        completado = false;
    }
```

```
    else
        errMensaje.SetError(txtSueldo, null);

    if (completado)
    {
        string nombre = txtNombre.Text;
        DateTime fechaNac = dtpFechaNac.Value;
        string sexo = cboSexo.Text;
        string estadoCivil = cboEstadoCivil.Text;
        string profesion = txtProfesion.Text;
        Trabajador emp = new Trabajador(nombre, fechaNac, sexo, estadoCivil, profesion, sueldo);
        if (archivo.GuardarArchivo(emp))
        {
            txtNombre.Clear();
            dtpFechaNac.ResetText();
            cboSexo.SelectedIndex = -1;
            cboEstadoCivil.SelectedIndex = -1;
            txtProfesion.Clear();
            txtSueldo.Clear();
            MessageBox.Show("Los datos fueron almacenados");
            txtNombre.Focus();
        }
    }
}
```

Agregar el siguiente código al evento **click** del botón Leer:

```
private void btnLeer_Click(object sender, EventArgs e)
{
    Trabajador emp = (Trabajador)archivo.LeerArchivo();
    if (emp != null)
    {
        txtNombre.Text = emp.Nombre;
        dtpFechaNac.Value = emp.FechaNac;
        cboSexo.Text = emp.Sexo;
        cboEstadoCivil.Text = emp.EstadoCivil;
        txtProfesion.Text = emp.Profesion;
        txtSueldo.Text = string.Format("{0:C2}", emp.Sueldo);
    }
    else
    {
        MessageBox.Show("No es posible leer el archivo");
    }
}
```

Establecer el **Form2** en Program.cs como programa de inicio.

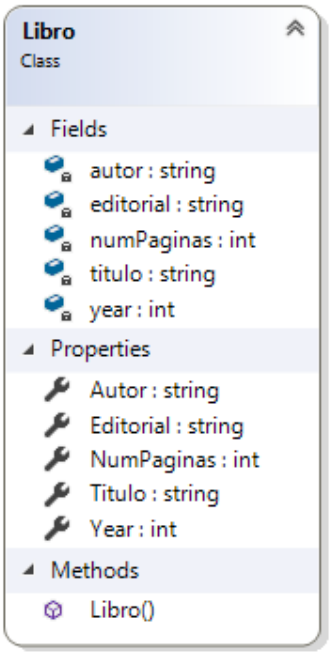
Probar el funcionamiento del formulario. (F5)

Ejemplo 3: serialización binaria de una colección List de objetos Libro

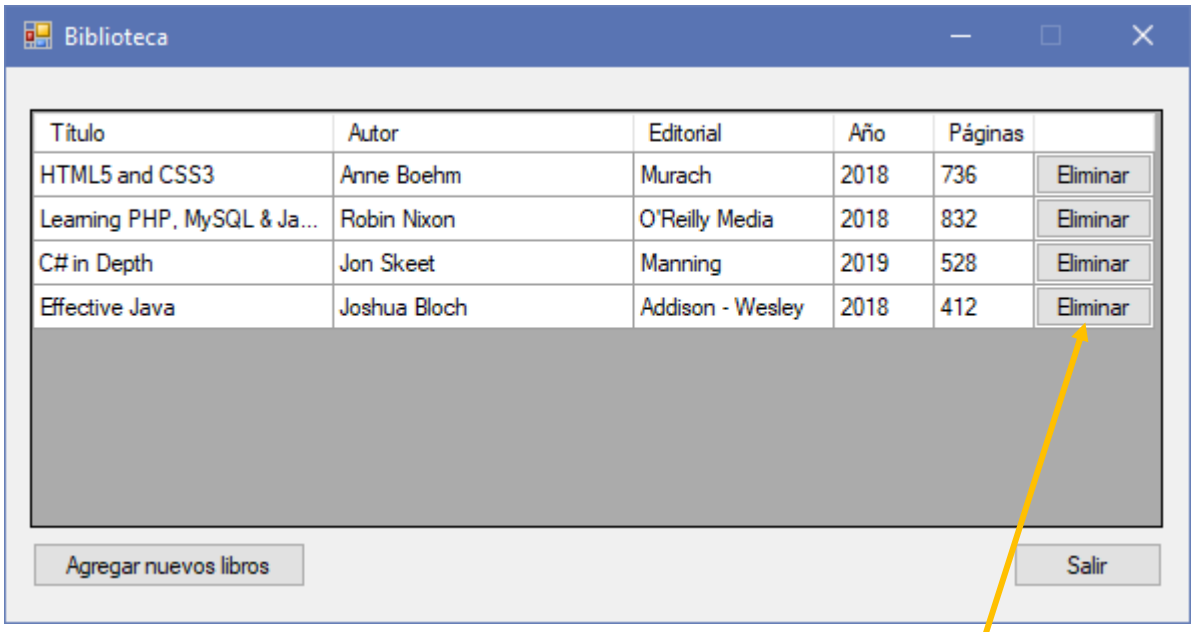
1. Crear la clase **Libro**

```
[Serializable]
public class Libro
{
    private string titulo;
    private string autor;
    private string editorial;
    private int year;
    private int numPaginas;

    public Libro()
    {

    }

    public string Titulo { get => titulo; set => titulo = value; }
    public string Autor { get => autor; set => autor = value; }
    public string Editorial { get => editorial; set => editorial = value; }
    public int Year { get => year; set => year = value; }
    public int NumPaginas { get => numPaginas; set => numPaginas = value; }
}
```

Diagrama de clase: Libro

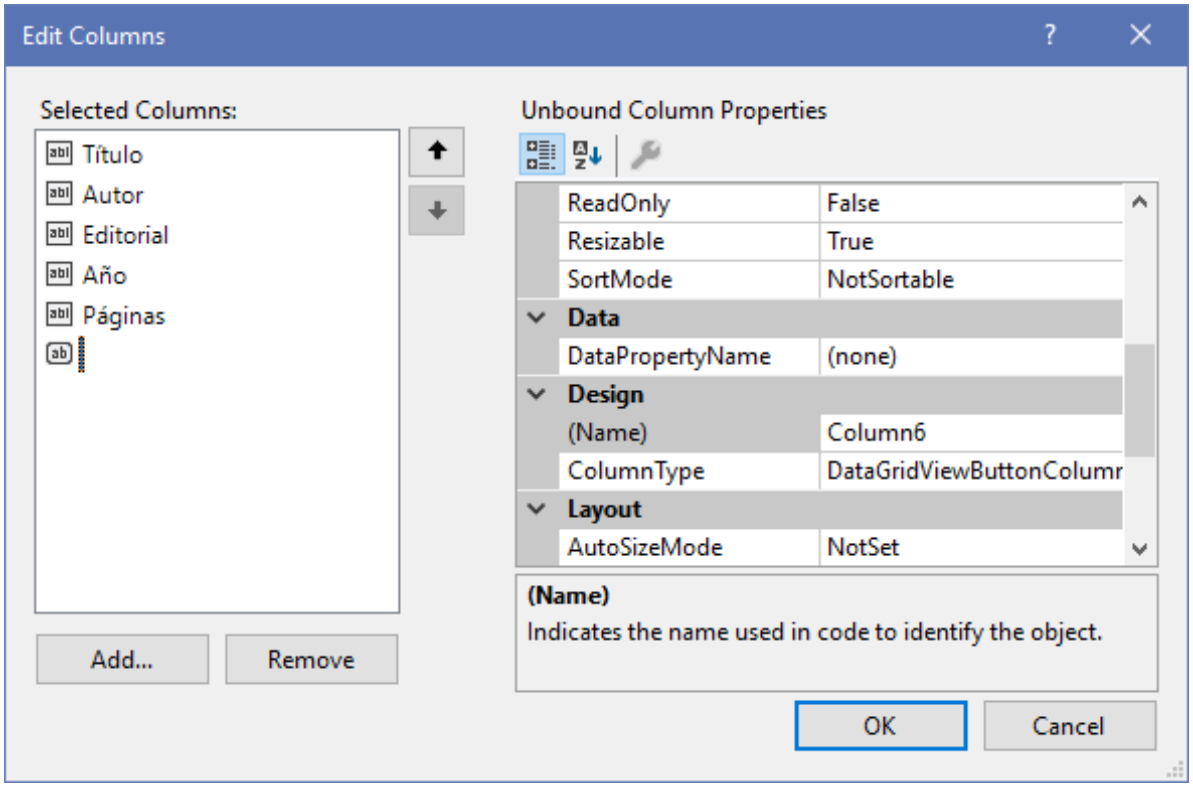


2. Crear el formulario Biblioteca, con la siguiente estructura:



Control	Name	Text
DataGridView1	dgvLibros	
button1	btnAgregar	Agregar nuevos libros
button2	btnSalir	Salir

Agregar 6 columnas al DataGridView: (establecer la columna6 como ButtonColumn)



Control	Columna	Propiedad	Valor
DataGridView1	Column6	Text	Eliminar
DataGridView1	Column6	UseColumnTextForButtonValue	True
DataGridView1	Column6	Name	Eliminar

Agregar la siguiente referencia: `using System.Collections.Generic;`

En las declaraciones generales del formulario instanciar la clase Archivo y crear la lista Libros:

```
Archivo archivo = new Archivo("libros.bin");
private List<Libro> listaLibros;
```

Crear la función **ActualizarGrid** que servirá para poblar el DataGridView con la lista de Libros:

```
private void ActualizarGrid(ref DataGridView grid)
{
        grid.Rows.Clear();
        foreach (Libro libro in listaLibros)
        {
            grid.Rows.Add(libro.Titulo, libro.Autor, libro.Editorial, libro.Year, libro.NumPaginas);
        }
        grid.ClearSelection();
}
```

Agregar el siguiente código al evento **Load** del formulario Biblioteca:

```
private void Form3_Load(object sender, EventArgs e)
{
    listaLibros = (List<Libro>)archivo.LeerArchivo();
    if (listaLibros != null)
    {
        ActualizarGrid(ref dgvLibros);
    }
    else
    {
        listaLibros = new List<Libro>();
    }
}
```

Agregar el siguiente código al evento **Clic** del botón Agregar nuevos libros:

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    Form4 formLibro = new Form4();
    if (formLibro.ShowDialog(this) == DialogResult.OK)
    {
        Libro libro = formLibro.getLibro();
        if (libro != null)
        {
            listaLibros.Add(libro);
            archivo.GuardarArchivo(listaLibros);
            listaLibros = (List<Libro>)archivo.LeerArchivo();
            ActualizarGrid(ref dgvLibros);
            MessageBox.Show("El nuevo libro fue almacenado", "Biblioteca",
                            MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
    formLibro.Dispose();
}
```

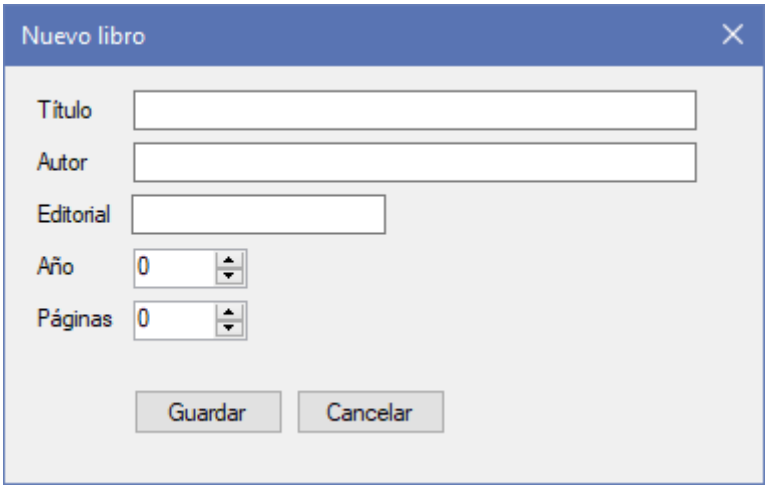
Agregar el siguiente código al evento **CellContentClick** del DataGridView:

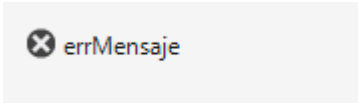
```
private void dgvLibros_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    if (dgvLibros.Rows.Count > 0 && dgvLibros.Columns[e.ColumnIndex].Name == "Eliminar")
    {
        string titulo = dgvLibros.Rows[e.RowIndex].Cells[0].Value.ToString();
        DialogResult result = MessageBox.Show("Esta seguro de eliminar el libro: " +
                                              titulo, "Biblioteca",
                                              MessageBoxButtons.YesNo,
                                              MessageBoxIcon.Warning,
                                              MessageBoxDefaultButton.Button2);
        if (result == DialogResult.Yes)
        {
            listaLibros.RemoveAt(e.RowIndex);
            archivo.GuardarArchivo(listaLibros);
            listaLibros = (List<Libro>)archivo.LeerArchivo();
            ActualizarGrid(ref dgvLibros);
        }
    }
}
```

Agregar el siguiente código al evento **Click** del botón Salir:

```
private void btnSalir_Click(object sender, EventArgs e)
{
    this.Close();
}
```

2. Crear el formulario Nuevo libro, con la siguiente estructura:





Control	Name	Text
textBox1	txtTitulo	
textBox2	txtAutor	
textBox3	txtEditorial	
numericUpDown1	nudYear	
numericUpDown2	nudPaginas	
button1	btnGuardar	Guardar
button2	btnCancelar	Cancelar

Modificar las propiedades del formulario:

Propiedad	Valor
MaximizeBox	False
MinimizeBox	False
FormBorderStyle	FixedDialog
StartPosition	CenterScreen

Modificar las propiedades del errorProvider:

Propiedad	Valor
Name	errMensaje
BlinkStyle	NeverBlink

En las declaraciones general del formulario crear un objeto Libro y el método getLibro:

```
private Libro libro = new Libro();

public Libro getLibro()
{
    return libro;
}
```

Agregar el siguiente código al evento **Click** del botón Guardar:

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    bool completado = true;
    if (txtTitulo.Text == string.Empty)
    {
        errMensaje.SetError(txtTitulo, "Ingrese el título");
        completado = false;
    }
    else
        errMensaje.SetError(txtTitulo, null);
    if (txtAutor.Text == string.Empty)
    {
        errMensaje.SetError(txtAutor, "Ingrese el autor");
        completado = false;
    }
    else
        errMensaje.SetError(txtAutor, null);

    if (txtEditorial.Text == string.Empty)
    {
        errMensaje.SetError(txtEditorial, "Ingrese la editorial");
        completado = false;
    }
    else
        errMensaje.SetError(txtEditorial, null);

    if (nudYear.Value < 2000)
    {
        errMensaje.SetError(nudYear, "Ingrese un año igual o superior a 2000");
        completado = false;
    }
```

```
        else
            errMensaje.SetError(nudYear, null);
        if (nudPaginas.Value <= 0)
        {
            errMensaje.SetError(nudPaginas, "Ingrese un número de páginas válido");
            completado = false;
        }
        else
            errMensaje.SetError(nudPaginas, null);

        if (completado)
        {
            this.DialogResult = DialogResult.OK;
            libro.Titulo = txtTitulo.Text;
            libro.Autor = txtAutor.Text;
            libro.Editorial = txtEditorial.Text;
            libro.Year = Convert.ToInt32(nudYear.Value);
            libro.NumPaginas = Convert.ToInt32(nudPaginas.Value);
            this.Close();
        }
    }

    private void btnCancelar_Click(object sender, EventArgs e)
    {
        this.DialogResult = DialogResult.Cancel;
        libro = null;
        this.Close();
    }
}
```

Agregar el siguiente código al evento **Click** del botón Cancelar:

```
private void btnCancelar_Click(object sender, EventArgs e)
{
    this.DialogResult = DialogResult.Cancel;
    libro = null;
    this.Close();
}
```

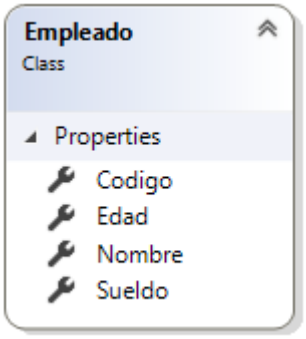
Establecer el **Form3** en Program.cs como programa de inicio.

Probar el funcionamiento del formulario. (F5)

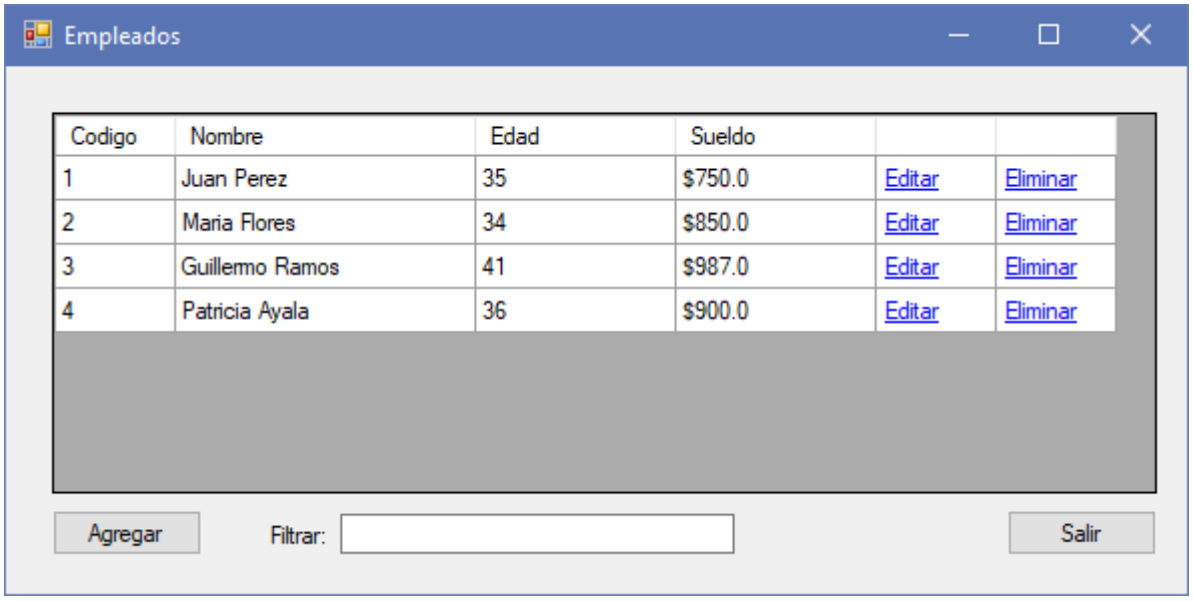
2. Crear la clase **Empleado**

```
[Serializable]
public class Empleado
{
    public int Codigo { get; set; }
    public string Nombre { get; set; }
    public int Edad { get; set; }
    public decimal Sueldo { get; set; }
}
```

Diagrama de clase: Empleado

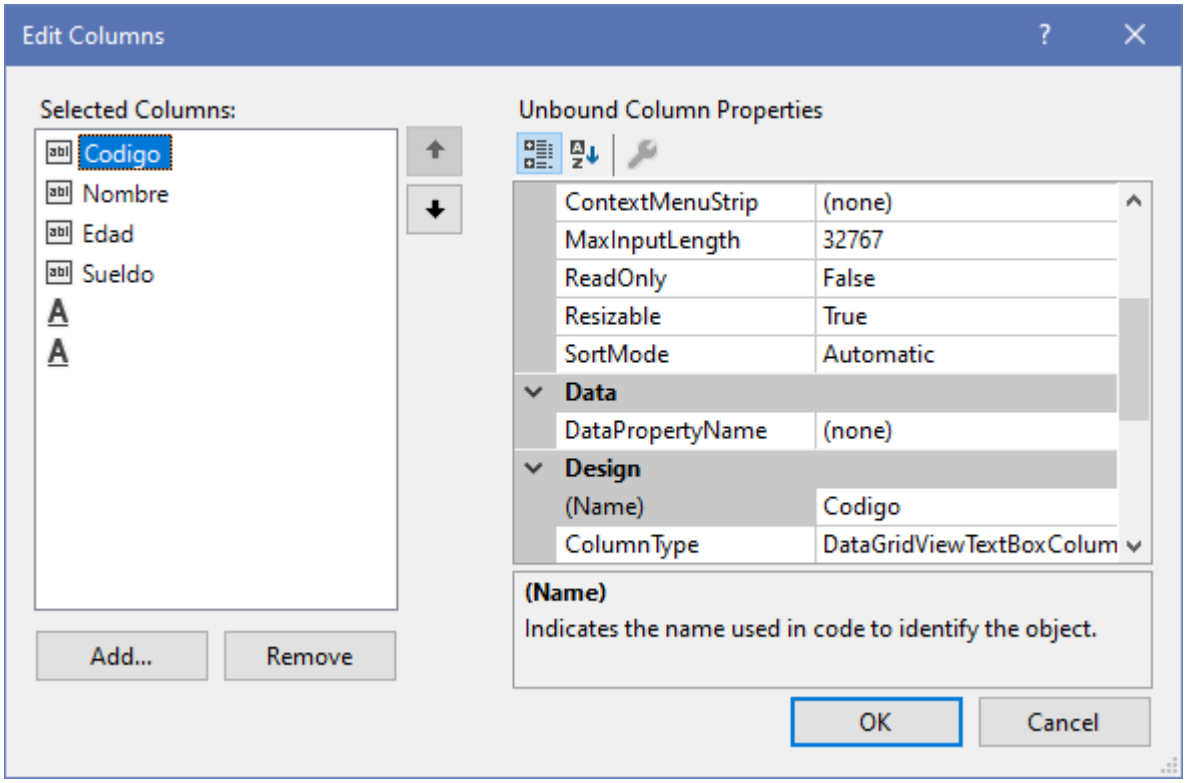


3. Crear el formulario Empleados, con la siguiente estructura:



Control	Name	Text
DataGridView1	dgvEmpleados	
button1	btnAgregar	Agregar nuevos libros
button2	btnSalir	Salir

Agregar 6 columnas al DataGridView: (establecer las columnas 5 y 6 como DataGridViewLinkColumn)



Control	Columna	Propiedad	Valor
DataGridView1	Column5	ColumnType	DataGridViewLinkColumn
DataGridView1	Column5	Name	Editar
DataGridView1	Column5	TrackVisitedState	False
DataGridView1	Column6	ColumnType	DataGridViewLinkColumn
DataGridView1	Column6	Name	Editar
DataGridView1	Column6	TrackVisitedState	False

Agregar las siguientes referencias:

```
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Linq;
```

Agregar las siguientes referencias:

```
Empleado empleado;
Dictionary<int, Empleado> empleadosDictionary;
List<int> listaCodigos = new List<int>();
Archivo archivo = new Archivo("empleados.bin");
```

Crear la función **ActualizarGrid** que servirá para poblar el DataGridView con el diccionario Empleados:

```
private void ActualizarGrid(ref DataGridView grid)
{
    grid.Rows.Clear();
    foreach (Empleado empleado in empleadosDictionary.Values)
    {
        grid.Rows.Add(empleado.Codigo, empleado.Nombre, empleado.Edad,
            empleado.Sueldo, "Editar", "Eliminar");
    }
    grid.ClearSelection();
}
```

Agregar el siguiente código al evento **Load** del Formulario:

```
private void Form3_Load(object sender, EventArgs e)
{
    empleadosDictionary = (Dictionary<int, Empleado>)archivo.LeerArchivo();
    if (empleadosDictionary != null)
    {
        ActualizarGrid(ref dgvEmpleados);
    }
    else
    {
        empleadosDictionary = new Dictionary<int, Empleado>();
    }
}
```

Agregar el siguiente código al evento **Click** del botón Agregar:

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    empleado = null;
    listaCodigos = empleadosDictionary.Select(Codigos => Codigos.Key).ToList();
    Form6 dialog = new Form6(null, listaCodigos);
    if (dialog.ShowDialog() == DialogResult.OK)
    {
        empleado = dialog.getEmpleado();
        if (empleado != null)
        {
            empleadosDictionary.Add(empleado.Codigo, empleado);
            archivo.GuardarArchivo(empleadosDictionary);
            ActualizarGrid(ref dgvEmpleados);
            MessageBox.Show("El nuevo empleado fue almacenado", "Empleados",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
}
```

Agregar el siguiente código al evento **CellContentClick** del DataGridView:

```
private void dgvEmpleados_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    if (dgvEmpleados.Rows.Count > 0 && dgvEmpleados.Columns[e.ColumnIndex].Name == "Editar")
    {
        empleado = new Empleado
        {
            Codigo = Convert.ToInt32(dgvEmpleados.CurrentRow.Cells[0].Value),
            Nombre = dgvEmpleados.CurrentRow.Cells[1].Value.ToString(),
            Edad = Convert.ToInt32(dgvEmpleados.CurrentRow.Cells[2].Value),
            Sueldo = Convert.ToDecimal(dgvEmpleados.CurrentRow.Cells[3].Value)
        };
        listaCodigos = empleadosDictionary.Select(Codigos => Codigos.Key).ToList();
        Form6 dialog = new Form6(empleado, listaCodigos);
        if (dialog.ShowDialog() == DialogResult.OK)
        {
            empleado = dialog.getEmpleado();
            if (empleado != null)
            {
                int codigo = Convert.ToInt32(empleado.Codigo);
                if (empleadosDictionary.ContainsKey(codigo))
                {
                    empleadosDictionary[codigo].Nombre = empleado.Nombre;
                    empleadosDictionary[codigo].Edad = empleado.Edad;
                    empleadosDictionary[codigo].Sueldo = empleado.Sueldo;
                }
                archivo.GuardarArchivo(empleadosDictionary);
                dgvEmpleados.Rows[e.RowIndex].Cells[0].Value = empleado.Codigo;
                dgvEmpleados.Rows[e.RowIndex].Cells[1].Value = empleado.Nombre;
                dgvEmpleados.Rows[e.RowIndex].Cells[2].Value = empleado.Edad;
                dgvEmpleados.Rows[e.RowIndex].Cells[3].Value = empleado.Sueldo;
               MessageBox.Show("El empleado fue actualizado", "Empleados",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
        dgvEmpleados.ClearSelection();
    }
    else if (dgvEmpleados.Rows.Count > 0 && dgvEmpleados.Columns[e.ColumnIndex].Name == "Eliminar")
    {
        int cod = Convert.ToInt32(dgvEmpleados.CurrentRow.Cells[0].Value);
        DialogResult result = MessageBox.Show("Esta seguro de eliminar el empleado de código "
                                              + cod, "Empleados",
                                              MessageBoxButtons.YesNo, MessageBoxIcon.Warning,
                                              MessageBoxDefaultButton.Button2);
        if (result == DialogResult.Yes)
        {
            empleadosDictionary.Remove(Convert.ToInt32(dgvEmpleados.CurrentRow.Cells[0].Value));
            archivo.GuardarArchivo(empleadosDictionary);
            dgvEmpleados.Rows.Remove(dgvEmpleados.CurrentRow);
            dgvEmpleados.ClearSelection();
        }
    }
}
```

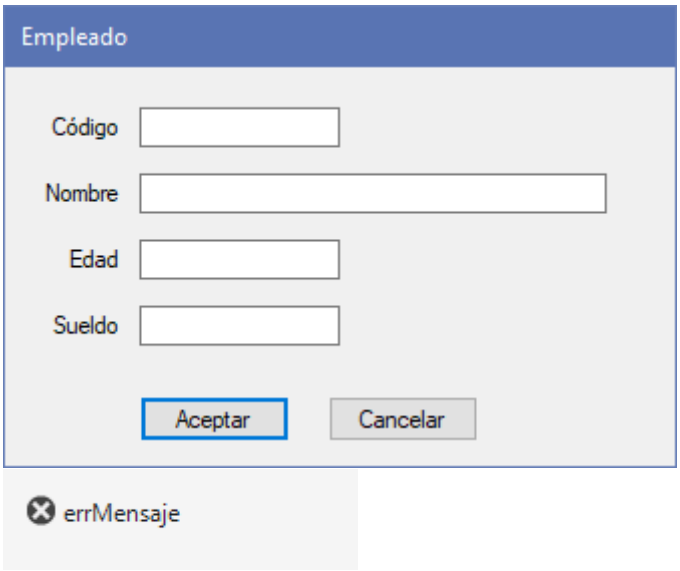
Agregar el código a los siguientes eventos del DataGridView:

```
private void dgvEmpleados_CellMouseEnter(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0)
        dgvEmpleados.Rows[e.RowIndex].DefaultCellStyle.BackColor = Color.LightCyan;
}

private void dgvEmpleados_CellMouseLeave(object sender, DataGridViewCellEventArgs e)
{
    dgvEmpleados.Cursor = Cursors.Default;
    if (e.RowIndex >= 0)
        dgvEmpleados.Rows[e.RowIndex].DefaultCellStyle.BackColor = Color.White;
    dgvEmpleados.ClearSelection();
}

private void dgvEmpleados_CellMouseMove(object sender, DataGridViewCellMouseEventArgs e)
{
    if (e.RowIndex >= 0 && e.ColumnIndex == dgvEmpleados.Columns["Editar"].Index)
        dgvEmpleados.Cursor = Cursors.Hand;
}

private void txtNombreFiltrar_TextChanged(object sender, EventArgs e)
{
    if (txtNombreFiltrar.Text != string.Empty)
    {
        dgvEmpleados.Rows.OfType<DataGridViewRow>().ToList().ForEach(row => { row.Visible = false; });
        dgvEmpleados.Rows.OfType<DataGridViewRow>().Where(
            r => r.Cells["Nombre"].Value.ToString().Trim() == txtNombreFiltrar.Text.Trim())
            .ToList().ForEach(row => row.Visible = true);
    }
    else
        dgvEmpleados.Rows.OfType<DataGridViewRow>().ToList().ForEach(row => { row.Visible = true; });
}
```

Agregar el siguiente código al evento **Click** del botón Salir:

```
private void btnSalir_Click(object sender, EventArgs e)
{
    this.Close();
}
```

3. Crear el formulario Nuevo empleado, con la siguiente estructura:



Control	Name	Text
textBox1	txtCodigo	
textBox2	txtNombre	
textBox3	txtEdad	
textBox4	txtSueldo	
button1	btnGuardar	Guardar
button2	btnCancelar	Cancelar

Modificar las propiedades del formulario:

Propiedad	Valor
MaximizeBox	False
MinimizeBox	False
FormBorderStyle	FixedDialog
StartPosition	CenterScreen

Modificar las propiedades del errorProvider:

Propiedad	Valor
Name	errMensaje
BlinkStyle	NeverBlink

Agregar el siguiente código al método constructor del formulario:

```
public partial class Form6 : Form
{
    public Form6(Empleado empleado, List<int> Codigos)
    {
        InitializeComponent();
        if (empleado != null)
        {
            txtCodigo.Text = empleado.Codigo.ToString();
            txtCodigo.Enabled = false;
            txtNombre.Text = empleado.Nombre;
            txtEdad.Text = empleado.Edad.ToString();
            txtSueldo.Text = empleado.Sueldo.ToString();
        }
        listaCodigos = Codigos;
    }
```

En las declaraciones del formulario, agregar el siguiente código:

```
private List<int> listaCodigos = new List<int>();
private Empleado empleadoActual = new Empleado();

public Empleado getEmpleado()
{
    return empleadoActual;
}
```

Agregar el siguiente código al evento **Clic** del botón Aceptar:

```
private void btnAceptar_Click(object sender, EventArgs e)
{
    bool validado = true;
    error.Clear();
    if (txtCodigo.Enabled)
    {
        if (int.TryParse(txtCodigo.Text, out int codigo) && codigo > 0)
        {
            if (listaCodigos.Contains(codigo))
            {
                MessageBox.Show("El código " + codigo + " ya existe");
                error.SetError(txtCodigo, "Ingrese otro código");
                validado = false;
            }
        }
        else
        {
            error.SetError(txtCodigo, "Ingrese un código entero entre 1 y 9999");
            validado = false;
        }
    }
    if (txtNombre.TextLength <= 0)
    {
        error.SetError(txtNombre, "Ingrese el nombre");
        validado = false;
    }
    if (!int.TryParse(txtEdad.Text, out int edad) || (edad < 18 || edad > 65))
    {
        error.SetError(txtEdad, "Ingrese una edad entre 18 y 65");
        validado = false;
    }
    if (!decimal.TryParse(txtSueldo.Text, out decimal sueldo) || sueldo < 0)
    {
        error.SetError(txtSueldo, "Ingrese un sueldo válido");
        validado = false;
    }
    if (validado)
    {
        empleadoActual.Codigo = Convert.ToInt32(txtCodigo.Text);
        empleadoActual.Nombre = txtNombre.Text;
        empleadoActual.Edad = Convert.ToInt32(txtEdad.Text);
        empleadoActual.Sueldo = Convert.ToDecimal(txtSueldo.Text);
        DialogResult = DialogResult.OK;
    }
}
```

Programar los siguientes eventos de validación de los TextBox:

```
private void txtNombre_Validated(object sender, EventArgs e)
{
    if (txtNombre.TextLength <= 0)
    {
        error.SetError(txtNombre, "Ingrese el nombre");
    }
    else
    {
        error.SetError(txtNombre, "");
    }
}

private void txtEdad_Validating(object sender, System.ComponentModel.CancelEventArgs e)
{
    if (!int.TryParse(txtEdad.Text, out int edad) || (edad < 18 || edad > 65))
    {
        error.SetError(txtEdad, "Ingrese una edad entre 18 y 65");
    }
    else
    {
        error.SetError(txtEdad, "");
    }
}

private void txtSueldo_Validating(object sender, System.ComponentModel.CancelEventArgs e)
{
    if (!decimal.TryParse(txtSueldo.Text, out decimal sueldo) || sueldo < 0)
    {
        error.SetError(txtSueldo, "Ingrese un sueldo válido");
    }
    else
    {
        error.SetError(txtSueldo, "");
    }
}

private void txtCodigo_Validated(object sender, EventArgs e)
{
    if (!int.TryParse(txtCodigo.Text,out int codigo) || (codigo<=0))
    {
        error.SetError(txtCodigo, "Ingrese un número entre 1 y 9999");
    }
    else
    {
        error.SetError(txtCodigo, "");
    }
}
```

Agregar el siguiente código al evento **Clic** del botón Cancelar:

```
private void btnCancelar_Click(object sender, EventArgs e)
{
    DialogResult = DialogResult.Cancel;
}
```

Establecer el **Form5** en Program.cs como programa de inicio.

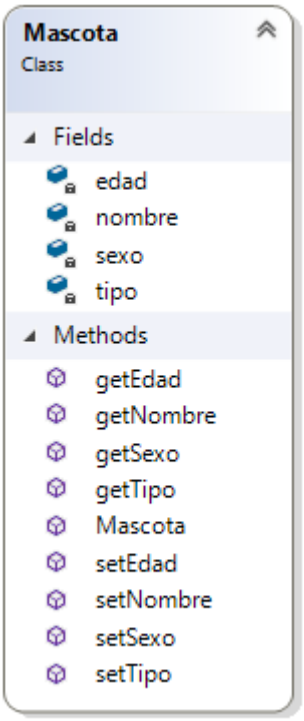
Probar el funcionamiento del formulario. (F5)

	Agregar un nuevo proyecto de nombre **práctica8** 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 una aplicación para implementar la clase Mascota.**

   Clase:



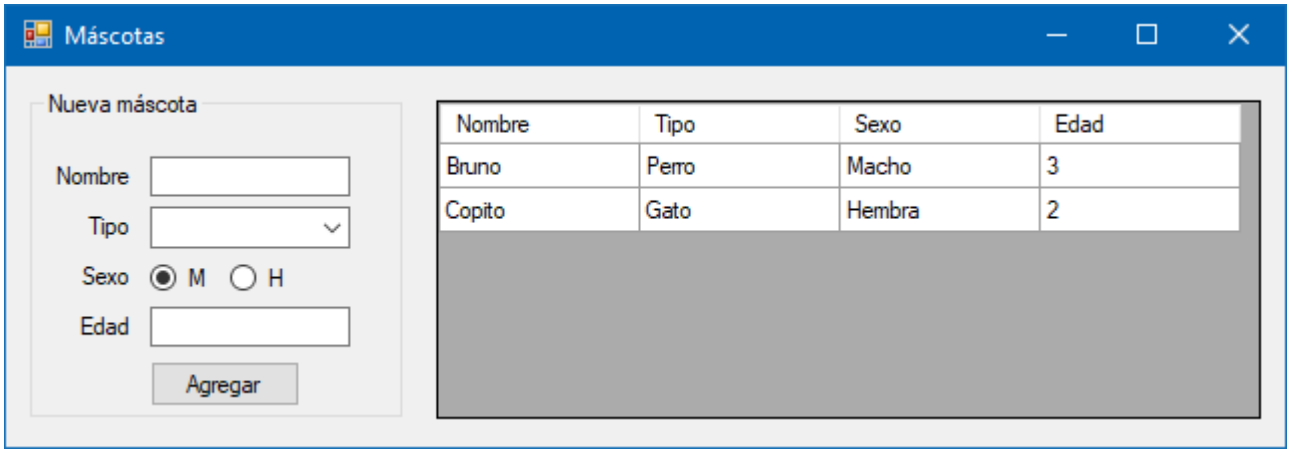
   Clase: **Mascota**

   Campos privados: nombre (string), tipo (string), sexo (string) y edad (int)

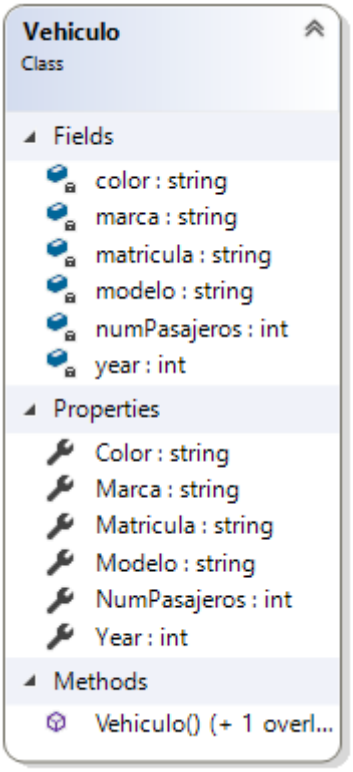
   Método constructor: recibirá como parámetros el nombre, tipo, sexo y edad. Validar que el sexo sea M o H y que la edad sea mayor a cero.

   Métodos de acceso y modificación: para cada uno de los campos privados. Validar que el sexo sea M o H y que la edad sea mayor a cero.

   Aplicación:



2. **Elaborar una aplicación para serializar en binario una colección de objetos Vehiculo.**



Clase: **Vehiculo**

Campos privados: matricula, marca, modelo, color, year y numPasajeros

Método constructor sin parámetros.

Método constructor con parámetros: recibirá los valores para inicializar cada campo.

Propiedades para cada campo de la clase.

Desarrollar la siguiente aplicación: