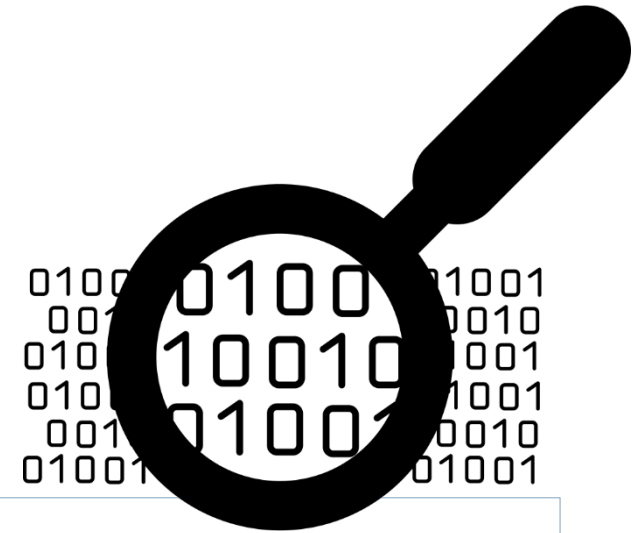




"La Ciencia sin Moral es Vana"

Universidad Católica de El Salvador
Facultad de Ingeniería y Arquitectura
Materia: Programación II
Docente: Master Giovanni Acosta

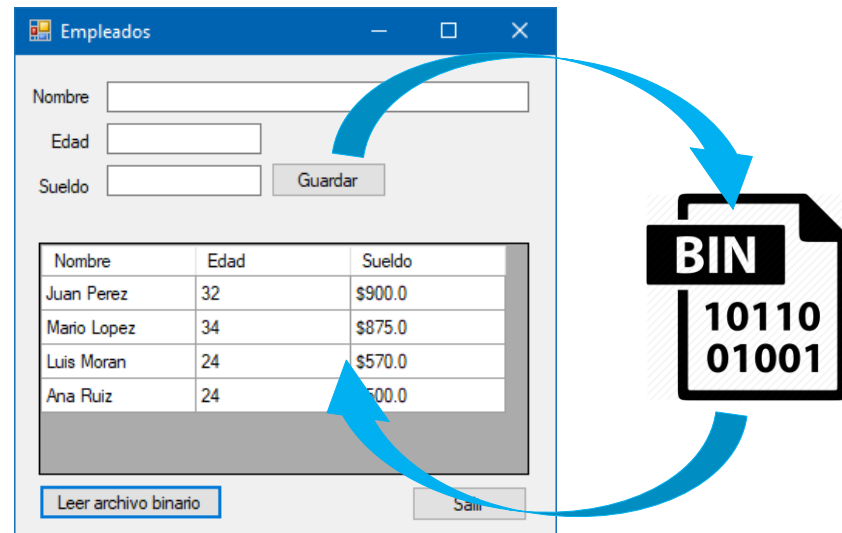


Tema 7: Manejo de archivos binarios y Serialización de objetos

Objetivos:

- ✓ Elaborar programas para escribir y leer archivos binarios
- ✓ Crear programas para escribir y leer archivos binarios con objetos serializables
- ✓ Practicar el paso de datos entre formularios

Archivos binarios



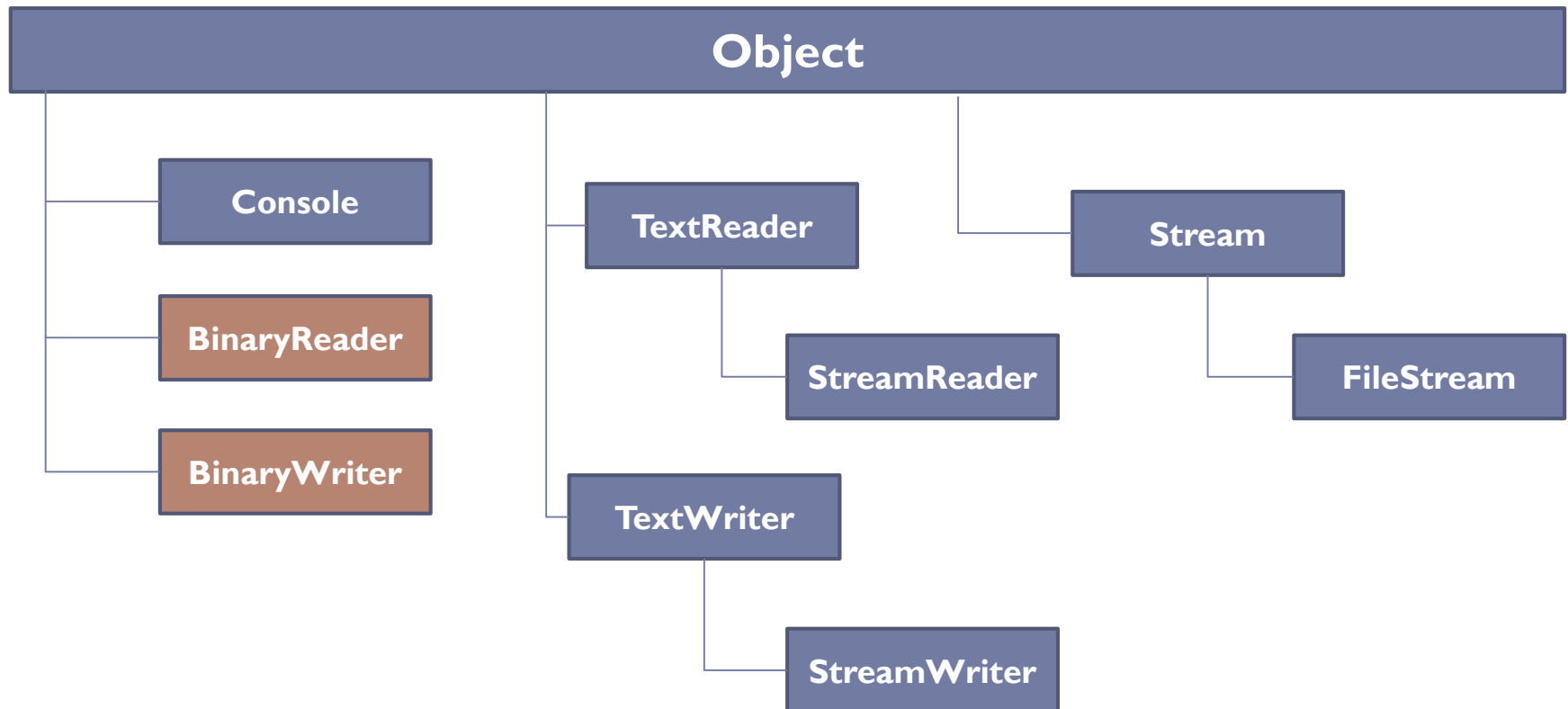
¿Qué es un archivo binario?

- ▶ Es un archivo o fichero que escribe tipos de datos primitivos (int, decimal, float, double, bool, string, etc.) no como cadenas de caracteres, sino en formato binario para posteriormente recuperarlos como tal, según su tipo de dato.
- ▶ Revisar los tipos de datos: <https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/built-in-types-table>
- ▶ Para estos casos, el espacio de nombre System.IO proporciona las clases **BinaryReader** y **BinaryWriter**, las cuales permiten leer y escribir, respectivamente, datos de cualquier tipo primitivo en formato binario y cadenas de caracteres en formato UTF-8.



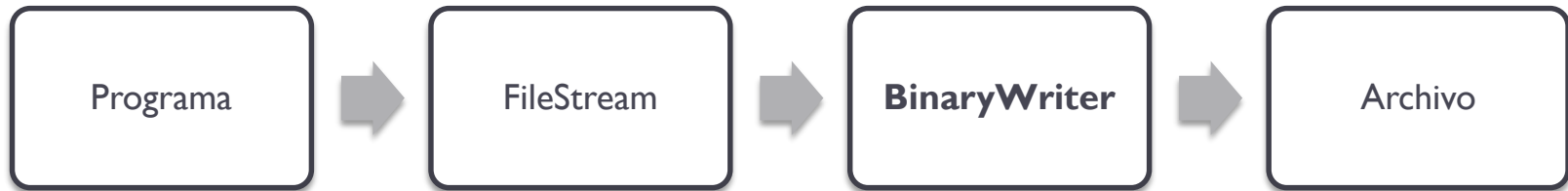
Espacio de nombres System.IO

- El espacio de nombres System.IO contiene clases que permiten leer y escribir en los archivos binarios.

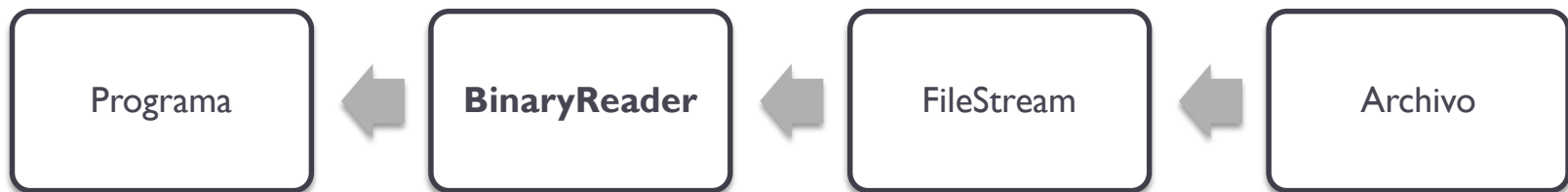


BinaryWriter y BinaryReader

- ▶ Un flujo de la clase **BinaryWriter** permite a una aplicación escribir datos de cualquier tipo primitivo.



- ▶ Un flujo de la clase **BinaryReader** permite a una aplicación leer datos de cualquier tipo primitivo escritos por un flujo de la clase **BinaryWriter**.



FileStream (flujo de datos)

- ▶ Los datos de un archivo pueden ser escritos o leídos byte a byte utilizando un flujo de la clase FileStream.
- ▶ Opciones de **FileMode**:

CreateNew	Crea un nuevo archivo
Create	Crea un nuevo archivo y si el archivo existe será sobrescrito
Open	Abre un archivo existente
OpenOrCreate	Abre un archivo si existe o crea un nuevo archivo sino existe
Truncate	Abre un archivo existente. Una vez abierto, el archivo será truncado a cero bytes de longitud
Append	Abre un archivo si existe para añadir datos al final del mismo, o crea un nuevo archivo si no existe



Modos de acceso al archivo binario

► Opciones de **FileAccess**:

Read	Permite acceder al archivo para realizar operaciones de lectura
ReadWrite	Permite acceder al archivo para realizar operaciones de lectura y escritura
Write	Permite acceder al archivo para realizar operaciones de escritura



Demo: escritura (archivo binario)



```
private void btnGuardar_Click(object sender, EventArgs e)
{
    string nombre;
    int edad;
    decimal sueldo;
    FileStream fs = null;
    BinaryWriter bw = null;
    try
    {
        fs = new FileStream("empleados.dat", FileMode.Append, FileAccess.Write);
        bw = new BinaryWriter(fs);
        nombre = txtNombre.Text;
        edad = Convert.ToInt32(txtEdad.Text);
        sueldo = Convert.ToDecimal(txtSueldo.Text);
        bw.Write(nombre);
        bw.Write(edad);
        bw.Write(sueldo);
        MessageBox.Show("Los datos fueron almacenados");
        txtNombre.Clear();
        txtEdad.Clear();
        txtSueldo.Clear();
        txtNombre.Focus();
    }
    catch (Exception)
    {
        MessageBox.Show("Ingrese los datos correctamente");
    }
    finally
    {
        if (bw != null) bw.Close();
    }
}
```

Empleados	
Nombre	<input type="text" value="Juan Perez"/>
Edad	<input type="text" value="32"/>
Sueldo	<input type="text" value="900"/> <input type="button" value="Guardar"/>

Escribe campo por campo en el archivo binario,
según el tipo de dato especificado para cada
campo

Demo: lectura (archivo binario)



```
private void btnLeer_Click(object sender, EventArgs e)
{
    string nombre;
    int edad;
    decimal sueldo;
    FileStream fs = null;
    BinaryReader br = null;
    try
    {
        fs = new FileStream("empleados.dat", FileMode.Open, FileAccess.Read);
        br = new BinaryReader(fs);
        dgvEmpleados.Rows.Clear();
        while (true)
        {
            nombre = br.ReadString();
            edad = br.ReadInt32();
            sueldo = br.ReadDecimal();
            dgvEmpleados.Rows.Add(nombre, edad, sueldo);
        }
    }
    catch (Exception)
    {
    }
    finally
    {
        if (br != null) br.Close();
        dgvEmpleados.ClearSelection();
    }
}
```

Lee campo por campo del archivo binario, según el tipo de dato con el que fue almacenado, y repite la operación hasta que llega al final del archivo, donde finaliza con una excepción en tiempo de ejecución

Empleado

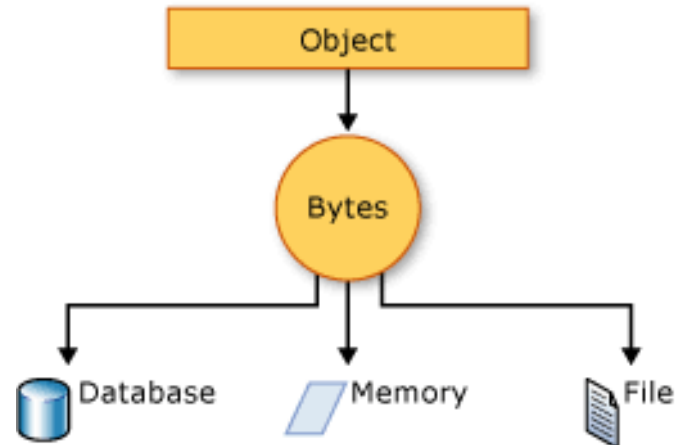
Nombre

Edad

Sueldo

Nombre	Edad	Sueldo
Juan Perez	32	\$900.0
Mario Lopez	34	\$875.0
Luis Moran	24	\$570.0
Ana Ruiz	24	\$500.0

Serialización y deserialización de objetos en C#



¿Qué es la serialización?

- ▶ Es el proceso para almacenar un objeto en un dispositivo de almacenamiento secundario (disco duro, memoria, etc.)
- ▶ Sirve para transformar los datos y poder transferirlos por un canal de comunicación (internet, archivo, memoria, etc.)
- ▶ Tipos de serialización en el Framework .NET: **Binario**, SOAP, XML

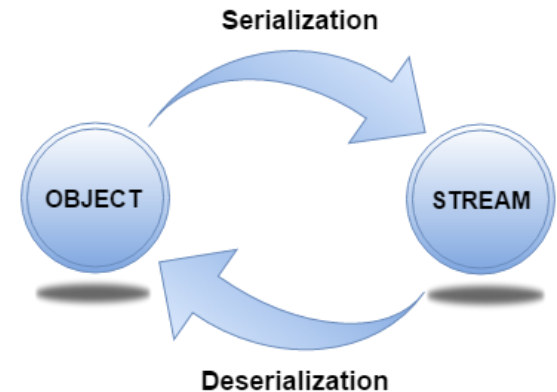
- ▶ Para serializar un objeto, la clase o estructura de datos debe declararse como:
[Serializable]



¿Cómo escribir y leer un objeto en el archivo serializado?

- ▶ Sintaxis para crear el formateador de archivo binario:

```
using System.Runtime.Serialization.Formatters.Binary;  
BinaryFormatter formatter = new BinaryFormatter();
```



- ▶ Sintaxis para serializar un objeto:

```
formatter.Serialize(FileStreamDeEscritura, Objeto);
```

- ▶ Sintaxis para deserializar un objeto:

```
Objeto = (tipoObjeto) formatter.Deserialize(flujoStreamDeLectura);
```

Demo: serializar objetos

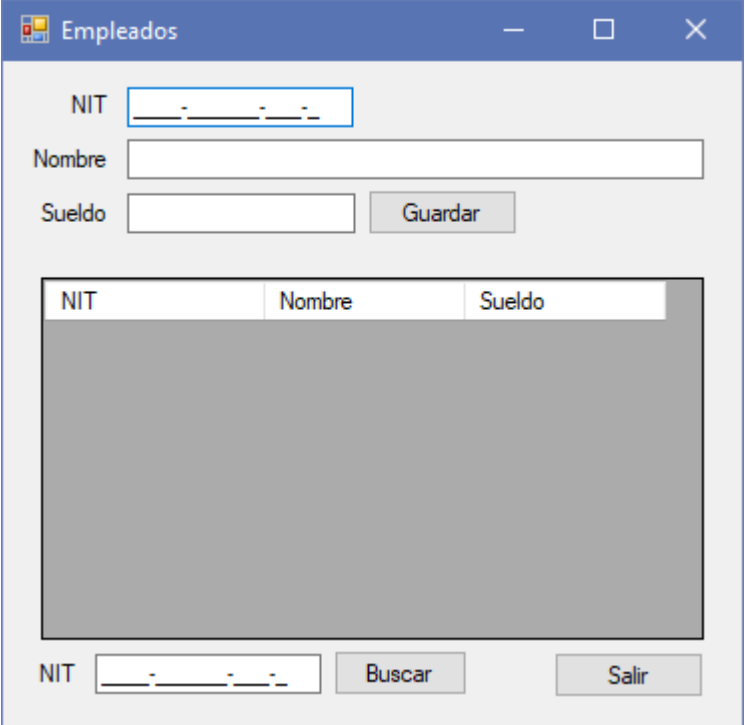


- Serializar la estructura de datos Empleado:

[Serializable]

6 references

```
public struct Empleado
{
    public string nit;
    public string nombre;
    public decimal sueldo;
}
```



The screenshot shows a Windows application window titled "Empleados". It contains three input fields labeled "NIT", "Nombre", and "Sueldo". To the right of the "Sueldo" field is a button labeled "Guardar". Below these fields is a table with three columns: "NIT", "Nombre", and "Sueldo". The table body is currently empty. At the bottom of the window, there is another "NIT" input field, a "Buscar" button, and a "Salir" button.

- Creación de diccionario y formateador:

```
private static Dictionary<string, Empleado> empleadosDictionary
    = new Dictionary<string, Empleado>();
private static BinaryFormatter formatter = new BinaryFormatter();
private const string NOMBRE_ARCHIVO = "datos.dat";
```

Demo: serializar objetos (cont..)



- Agregar la estructura de datos al diccionario:

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    try
    {
        Empleado empleado = new Empleado();
        empleado.nit = mktNIT.Text;
        empleado.nombre = txtNombre.Text;
        empleado.sueldo = Convert.ToDecimal(txtSueldo.Text);
        if (empleadosDictionary.ContainsKey(empleado.nit))
        {
            MessageBox.Show("Ya fue agregado antes un empleado con NIT " + empleado.nit);
        }
        else
        {
            empleadosDictionary.Add(empleado.nit, empleado);
            guardarArchivo();
            leerArchivo();
            actualizarGrid(ref dgvEmpleados);
            mktNIT.Clear();
            txtNombre.Clear();
            txtSueldo.Clear();
            mktNIT.Focus();
        }
    }
    catch (Exception)
    {
        MessageBox.Show("Ingrese los datos corretamente");
    }
}
```

Agrega pareja Clave – Valor (NIT – Empleado) al diccionario, para guardarlo en un archivo binario

Demo: serializar objetos (cont..)



- ▶ Guardar el diccionario serializado en el archivo binario:

```
private static void guardarArchivo()
{
    try
    {
        FileStream writerFS = new FileStream(NOMBRE_ARCHIVO, FileMode.Create, FileAccess.Write);
        formatter.Serialize(writerFS, empleadosDictionary);
        writerFS.Close();
        MessageBox.Show("Los datos fueron guardados");
    }
    catch (Exception)
    {
        Console.WriteLine("No fue posible almacenar los datos de empleados");
    }
}
```

Demo: serializar objetos (cont..)



- ▶ Leer el archivo binario y recuperar el diccionario serializado:

```
private static void leerArchivo()  
{  
    if (File.Exists(NOMBRE_ARCHIVO))  
    {  
        try  
        {  
            FileStream readerFS = new FileStream(NOMBRE_ARCHIVO, FileMode.Open, FileAccess.Read);  
            empleadosDictionary = (Dictionary<String, Empleado>) formatter.Deserialize(readerFS);  
            readerFS.Close();  
        }  
        catch (Exception)  
        {  
            Console.WriteLine("El archivo no esta disponible o no fue posible leerlo");  
        }  
    }  
}
```


Demo: serializar objetos (cont..)



- Actualizar el DataGridView con los datos del diccionario:

```
public static void actualizarGrid(ref DataGridView grid)
{
    if (empleadosDictionary.Count > 0)
    {
        grid.Rows.Clear();
        foreach (Empleado empleado in empleadosDictionary.Values)
        {
            grid.Rows.Add(empleado.nit, empleado.nombre, empleado.sueldo);
        }
        grid.ClearSelection();
    }
    else
    {
        MessageBox.Show("No existe información almacenada sobre empleados");
    }
}
```



Demo: serializar objetos (cont..)



- Buscar datos dentro del Diccionario de empleados:

```
private void btnBuscar_Click(object sender, EventArgs e)
{
    //Buscar empleado por NIT
    if (dgvEmpleados.Rows.Count > 0 && mktNitBuscar.MaskFull)
    {
        if (empleadosDictionary.ContainsKey(mktNitBuscar.Text))
        {
            Empleado empleadoEncontrado = empleadosDictionary[mktNitBuscar.Text];
            MessageBox.Show("Datos del empleado: \n\n" +
                "NIT: " + empleadoEncontrado.nit + "\n" +
                "Nombre: " + empleadoEncontrado.nombre + "\n" +
                "Sueldo: " + empleadoEncontrado.sueldo.ToString("C1"));
        }
        else
        {
            MessageBox.Show("No existe un empleado con el NIT: " + mktNitBuscar.Text);
        }
    }
}
```

NIT	<input type="text" value="____-____-____"/>	<input type="button" value="Buscar"/>	<input type="button" value="Salir"/>
-----	---	---------------------------------------	--------------------------------------

Paso de datos entre Formularios

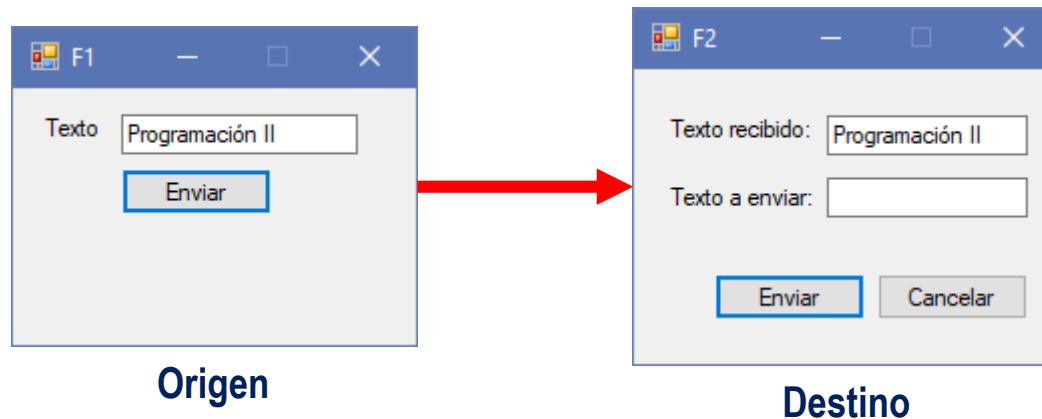
The diagram illustrates the flow of data between a table and a form. A window titled "Empleados" contains a table with the following data:

Nombre	Edad	Sueldo		
Juan Perez	35	\$950.0	Editar	Eliminar

Below the table are buttons for "Agregar" and "Filtrar:". A modal window titled "Empleado" is open, showing input fields for "Nombre" (containing "Juan Perez"), "Edad" (containing "35"), and "Sueldo" (containing "950"). At the bottom of the modal are "Aceptar" and "Cancelar" buttons. Blue arrows indicate the data flow: one arrow points from the "Editar" link in the table to the "Empleado" modal, and another arrow points from the "Aceptar" button in the modal back to the table.

¿Cómo pasar datos del formulario origen al destino?

- ▶ Para pasar datos entre formularios se puede realizar lo siguiente:
 - ▶ Caso I: pasar datos del formulario origen al formulario destino.



- ▶ Evento clic del botón **Enviar** del formulario **origen** (F1), pasar datos a través del método constructor del formulario **destino** (F2)

```
private void btnEnviar_Click(object sender, EventArgs e)
{
    F2 f2 = new F2(txtOrigen.Text);
    f2.Show();
}
```

¿Cómo pasar datos del formulario origen al destino?

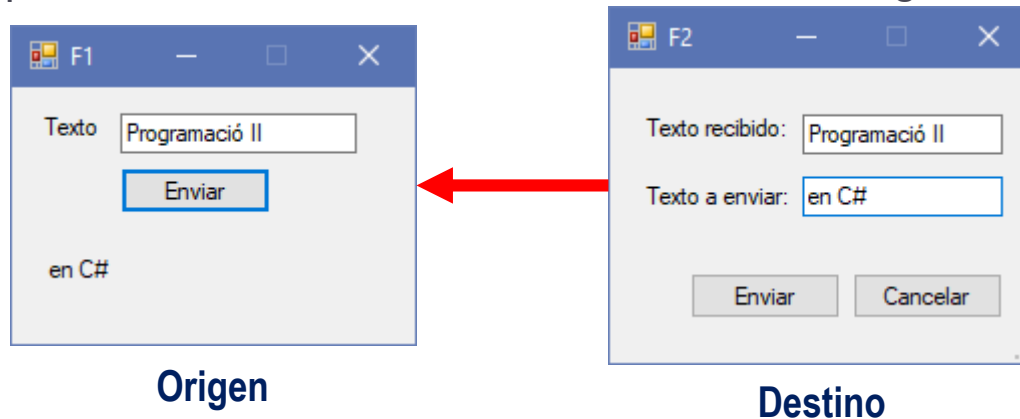
- Evento el **método constructor** del formulario **destino** (F2), agregar los parámetros necesarios para recibir los datos enviados por el formulario **origen** (F1).

```
namespace GuiaPractica7
{
    4 references
    public partial class F2 : Form
    {
        1 reference
        public F2(string texto)
        {
            InitializeComponent();
            txtRecibido.Text = texto;
        }
    }
}
```



¿Cómo pasar datos del formulario destino al origen?

- ▶ Caso 2: pasar datos del formulario destino al formulario origen.



- ▶ Modificar el código del evento clic del botón **Enviar** del formulario **origen** (F1), pasar datos a través del método constructor del formulario **destino** (F2) y abrirlo en modo modal.

```
private void btnEnviar_Click(object sender, EventArgs e)
{
    F2 f2 = new F2(txtOrigen.Text);
    if (f2.ShowDialog() == DialogResult.OK)
    {
        lblMensaje.Text = f2.Mensaje;
    }
}
```

1. Pasa datos del origen en el constructor del destino (F2)
2. Abre el formulario destino en modo modal, ShowDialog()
3. Recibe datos del destino a través de la propiedad Mensaje del formulario destino.

¿Cómo pasar datos del formulario destino al origen?

- ▶ En el formulario destino (F2), crear **propiedades publicas** para pasar datos al destino, en este ejemplo se ha creado sólo la propiedad Mensaje de tipo String.

```
//Propiedad
```

```
2 references
```

```
public string Mensaje { get; set; }
```

- ▶ Programar el evento clic del botón Enviar del formulario destino:

```
private void btnEnviar_Click(object sender, EventArgs e)
{
    Mensaje = txtEnviar.Text;
    DialogResult = DialogResult.OK;
    this.Close();
}
```

1. Pasa el valor del textbox a la propiedad Mensaje
2. Regresará el valor de OK al formulario origen (F1)
3. Cierra el formulario destino (F2)

Demo: pasar datos entre formularios



- Nota: el código fuente de esto demo estará disponible en la plataforma.

Formulario Origen

Nombre	Edad	Sueldo		
Juan Perez	35	\$950.0	Editar	Eliminar

Formulario Modal (Empleado)

Nombre:

Edad:

Sueldo:

Destino (Formulario Modal)

Bibliografía

1. Asad, A. (2017). The C# Programmer's Study Guide (MCSD) Exam: 70-483. Pakistan: Apress.
2. Ceballos, F. (2013). Microsoft C# Curso de programación. Segunda edición. México: Alfaomega.
3. Putier, S. (2015). C# 6 y Visual Studio 2015 Los fundamentos del lenguaje. España: ENI.

► Clase FileStream:

[https://msdn.microsoft.com/es-es/library/system.io.filestream\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/system.io.filestream(v=vs.110).aspx)

► Clase BinaryReader:

[https://msdn.microsoft.com/es-es/library/system.io.binaryreader\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/system.io.binaryreader(v=vs.110).aspx)

► Clase BinaryWriter:

[https://msdn.microsoft.com/en-us/library/system.io.binarywriter\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.binarywriter(v=vs.110).aspx)

