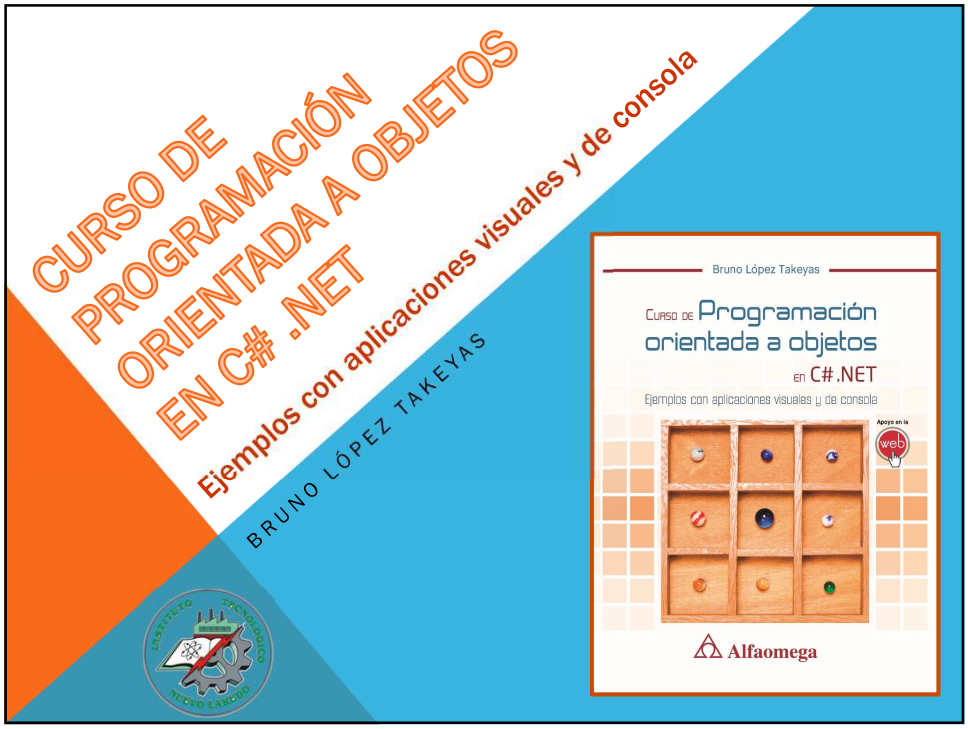

CURSO DE PROGRAMACIÓN ORIENTADA A OBJETOS EN C# .NET
Ejemplos con aplicaciones visuales y de consola
BRUNO LÓPEZ TAKEYAS
Bruno López Takeyas
Curso de Programación orientada a objetos en C#.NET
Ejemplos con aplicaciones visuales y de consola
Alfaomega


CAPÍTULO 9
SERIALIZACIÓN
(ALMACENAMIENTO DE OBJETOS EN ARCHIVOS)
Objeto
Bytes
Base de datos
Memoria
Archivo
2

Preguntas detonadoras

- ¿Qué es y para qué sirve la serialización?
- ¿Se pueden almacenar objetos en un archivo? ¿Cómo?

3

Serialización

- Es el proceso para almacenar un objeto en un dispositivo de almacenamiento secundario.
- Almacenar un objeto en un archivo en el disco duro, memoria USB, CD, etc.

4

¿Para qué sirve la serialización?

- Para transformar los datos y poder transferirlos por un canal de comunicación (internet, archivo, memoria, etc.)
- Para compartir información de un sistema en otra aplicación.

5

Tipos de serialización en el Framework .NET

Tipos de serialización
- Binario
- SOAP
- XML

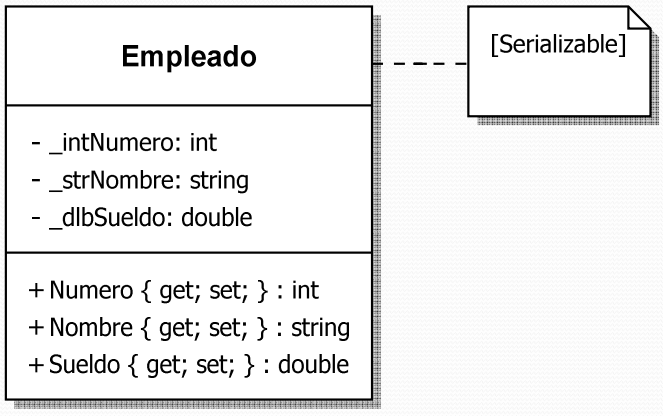
6

¿Cómo preparar una clase para serializar sus objetos?

- Durante la serialización, los atributos, nombre de la clase y su ensamblado se convierten en una secuencia de bytes.
- Para serializar un objeto, su clase debe declararse como `[Serializable]`

7

Ejemplo en UML



8

Ejemplo codificado en C#

```
[Serializable]
   class Empleado
   {
       // Atributos privados
       …

       // Propiedades públicas
       …
   }
```

9

Espacios de nombres requeridos

```
using System.IO; // Para el uso
de archivos
using
System.Runtime.Serialization.For
matters.Binary;  // Para el uso
de la serialización
```

10

Crear un archivo

- Para grabar datos en un archivo, primero debe crearse mediante:

```
// Declaración del flujo del archivo
private System.IO.FileStream flujo;

// Creación del archivo
flujo = new FileStream(NombreArchivo, FileMode.Create);
```

11

Serializar el archivo creado

- Declaración del formateador para serializar el archivo

```
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter seriador;

seriador = new BinaryFormatter();
```

12

¿Cómo grabar un objeto en el archivo serializado?

- Almacena **miObjeto** en el archivo controlado por **flujo**

```
seriador.Serialize(flujo, miObjeto);
```

13

Cerrar el archivo

- Una vez usado el archivo, entonces debe cerrarse mediante:

```
if (flujo != null)
  flujo.Close();
```

14

Abrir un archivo en modo lectura

- Para leer datos de un archivo, primero debe abrirse mediante:

```
// Declaración del flujo del archivo
private System.IO.FileStream flujo;

// Apertura del archivo
flujo = new FileStream(NombreArchivo, FileMode.Open);
```

15

Serializar el archivo abierto en modo lectura

- Declaración del formateador para serializar el archivo

```
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter seriador;

seriador = new BinaryFormatter();
```

16

¿Cómo leer un objeto del archivo serializado?

- A este proceso se le conoce como deserialización.
- Lee **miObjeto** del archivo controlado por **flujo**

```
miObjeto =
seriador.Deserialize(flujo);
```

17

Prototipar al leer un objeto

- Al leer datos del archivo, debe indicarse el tipo de dato
- A este proceso se le conoce como prototipar

```
miEmpleado = (Empleado)
seriador.Deserialize(flujo);
```

18

Modos de apertura de archivos (FileMode)

FileMode	Uso
`CreateNew`	Crea un nuevo archivo. Si el archivo existe dispara una IOException
`Truncate`	Abrir un archivo existente. Una vez abierto, el archivo será truncado a cero bytes de longitud.
`Create`	Crea un nuevo archivo. Si el archivo existe será sobreescrito.
`Open`	Abrir un archivo existente. Si no existe dispara una FileNotFoundException.
`OpenOrCreate`	Abrir un archivo existente, si no existe, lo crea.
`Append`	Abrir un archivo para agregar datos al final en caso de existir; de lo contrario crea un archivo nuevo.

19

Modos de acceso de archivos (FileAccess)

FileAccess	Uso
`Read`	Acceso al archivo en modo de solo lectura
`ReadWrite`	Acceso al archivo en modo de lectura y escritura
`Write`	Acceso al archivo en modo de solo escritura

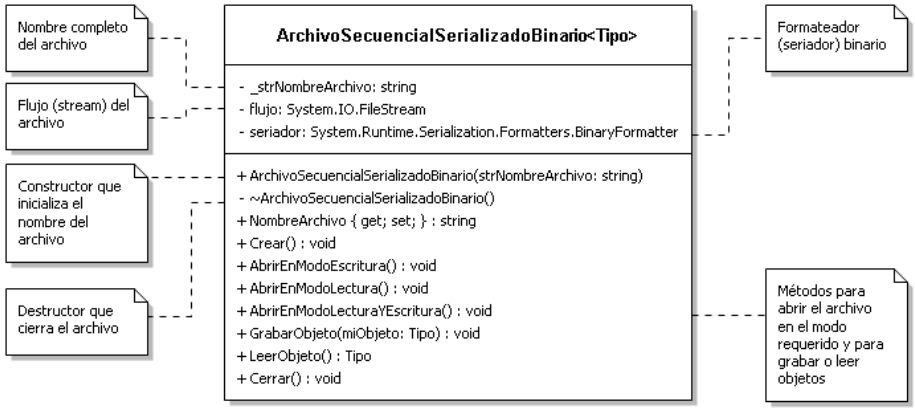
20

¿Cómo detectar si existe un archivo?

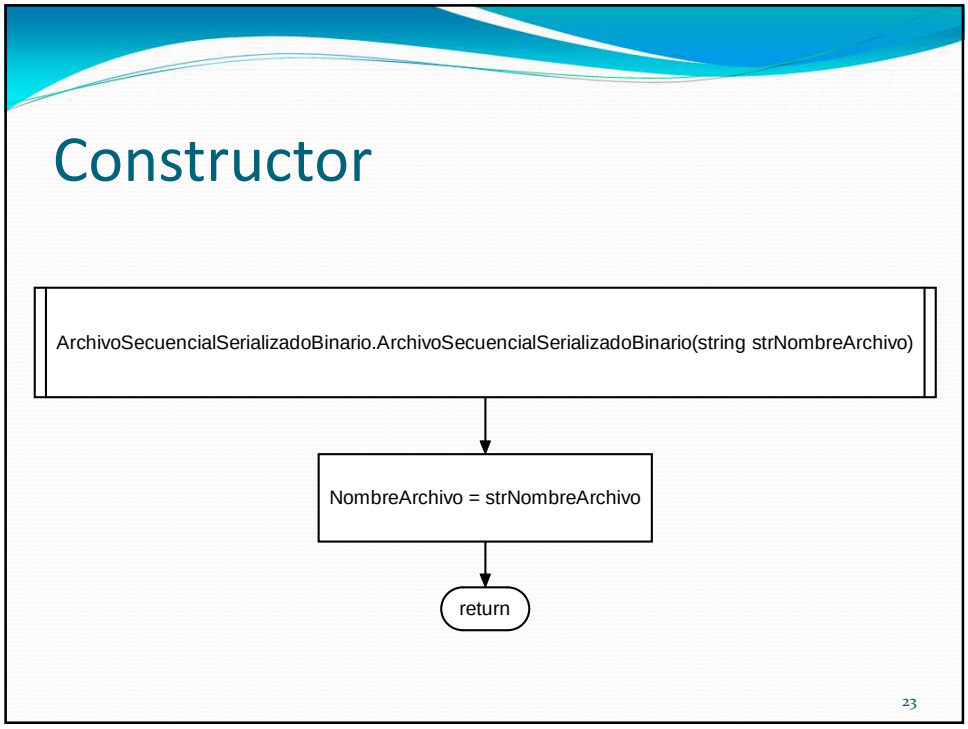
```
if (File.Exists(NombreArchivo))
{
    ……
}
```

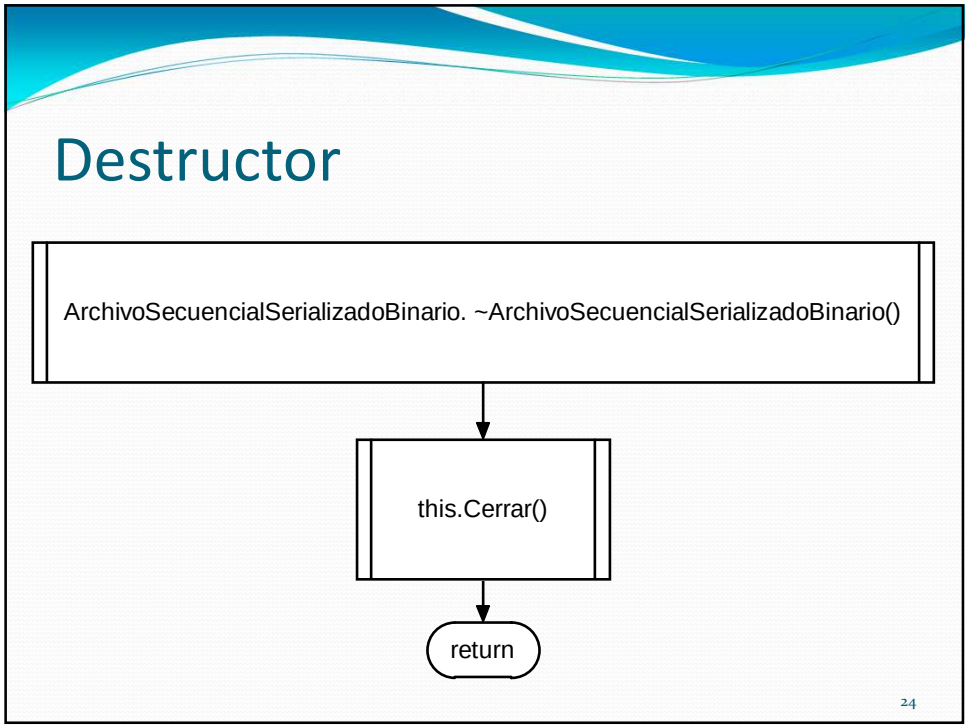
21

Diseño de una clase para el manejo de un archivo serializable

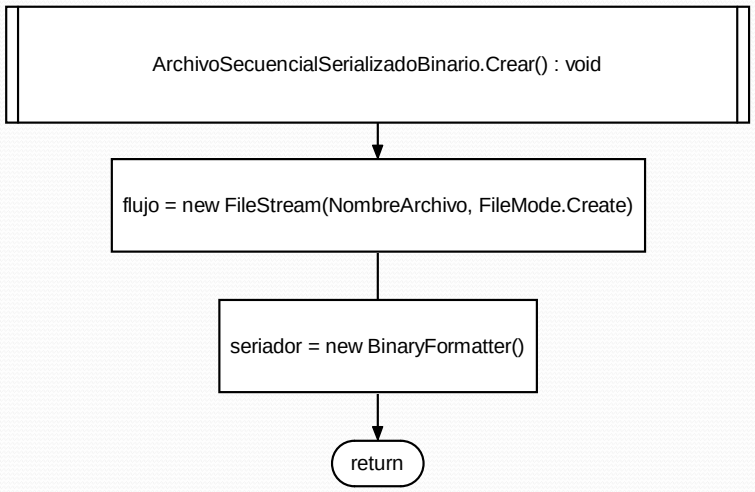


22


Constructor
ArchivoSecuencialSerializadoBinario.ArchivoSecuencialSerializadoBinario(string strNombreArchivo)
NombreArchivo = strNombreArchivo
return
23

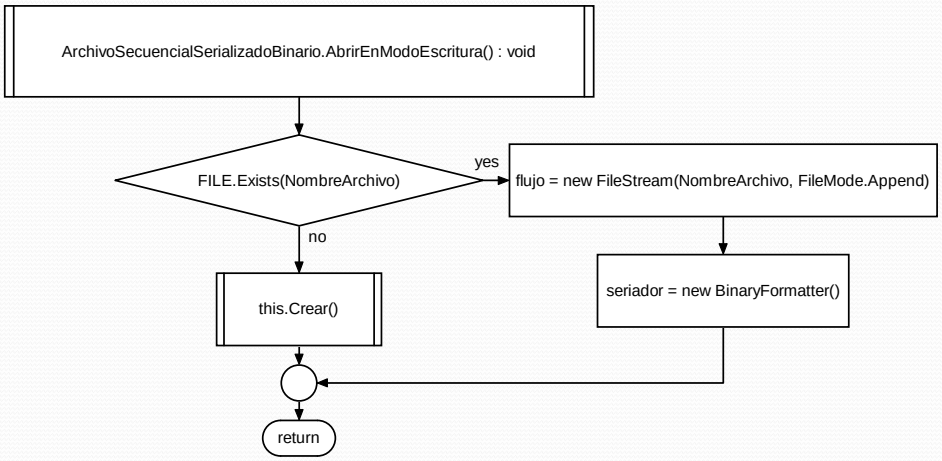

Destructor
ArchivoSecuencialSerializadoBinario. ~ArchivoSecuencialSerializadoBinario()
this.Cerrar()
return
24

Método para crear el archivo



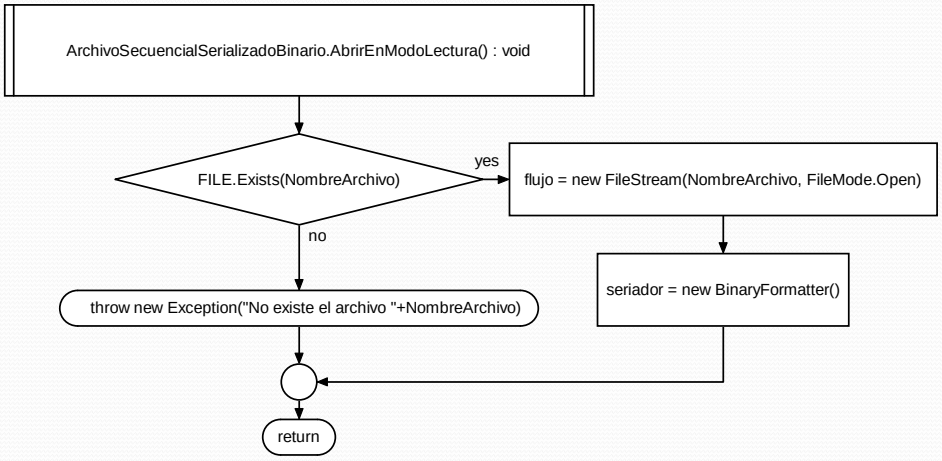
25

Método para abrir el archivo en modo escritura



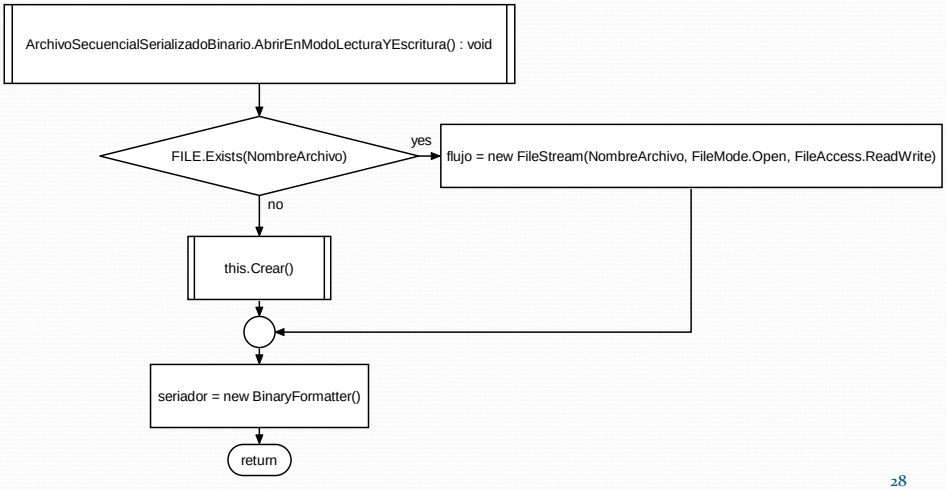
26

Método para abrir el archivo en modo lectura



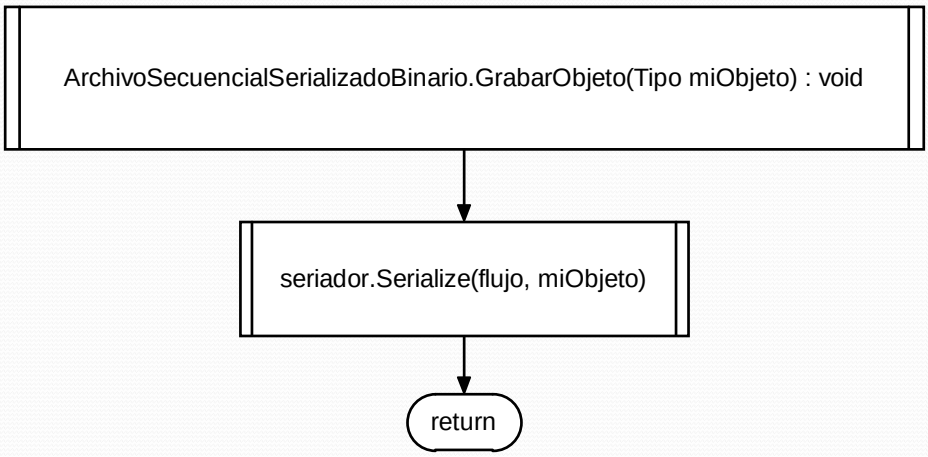
27

Método para abrir el archivo en modo lectura y escritura



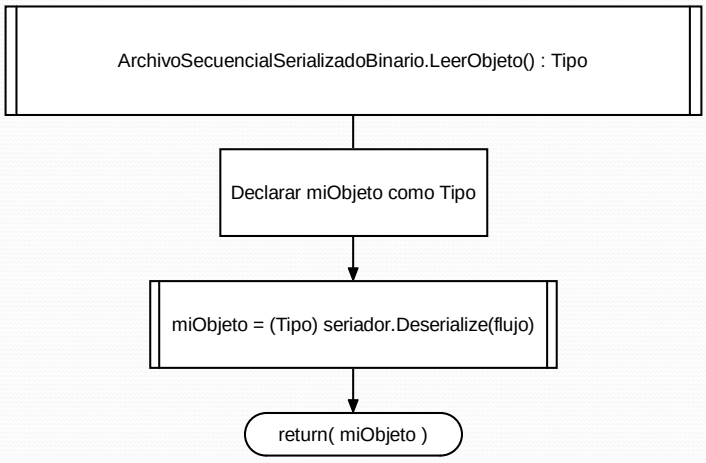
28

Método para grabar un objeto en el archivo



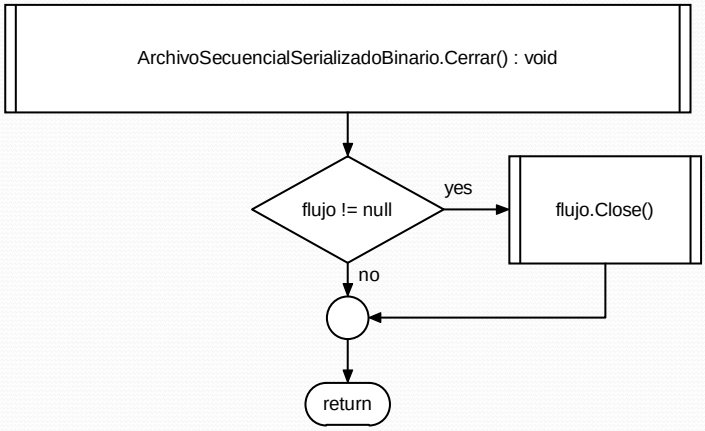
29

Método para leer un objeto del archivo



30

Método para cerrar el archivo



31

Declaración y creación del objeto global para controlar el archivo

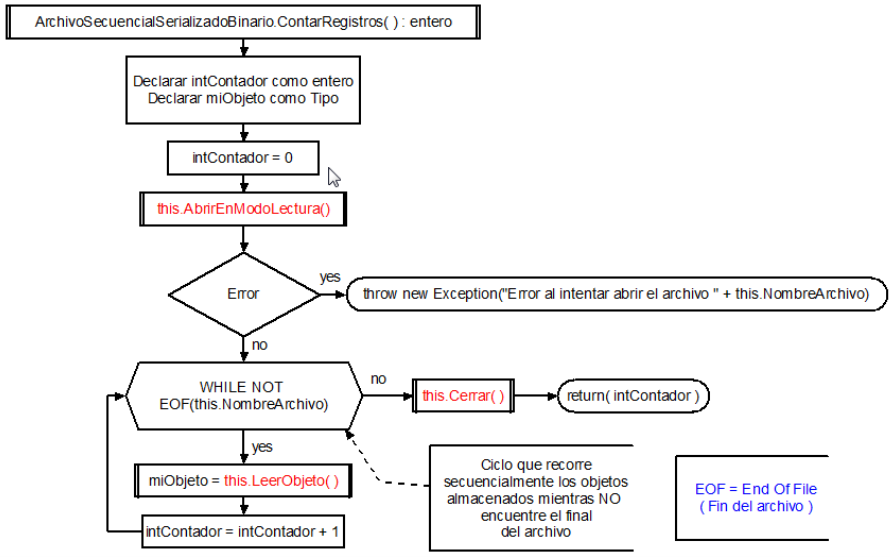
```
// Declaración y creación global de un objeto para el archivo

      ArchivoSecuencialSerializadoBinario<Empleado> miArchivo
    = new ArchivoSecuencialSerializadoBinario<Empleado>
              ("c:\\Datos\\Empleados.dat");
```

32

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    // Declaración y creación de un objeto local para almacenar los datos del empleado
    Empleado miEmpleado = new Empleado();

            try
            {
                miEmpleado.Numero=int.Parse(txtNumero.Text);
                miEmpleado.Nombre = txtNombre.Text;
                miEmpleado.Sueldo = double.Parse(txtSueldo.Text);
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message);
                return;
            }

            try
            {
                // Abrir el archivo para escribir en él
                miArchivo.AbrirEnModoEscritura();

                // Grabar el objeto
                miArchivo.GrabarObjeto(miEmpleado);

                MessageBox.Show("Datos almacenados correctamente !!!");
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                miArchivo.Cerrar();
            }

            // Mostrar los datos de los autos en el dataGridView
            MostrarDatos();
}
```

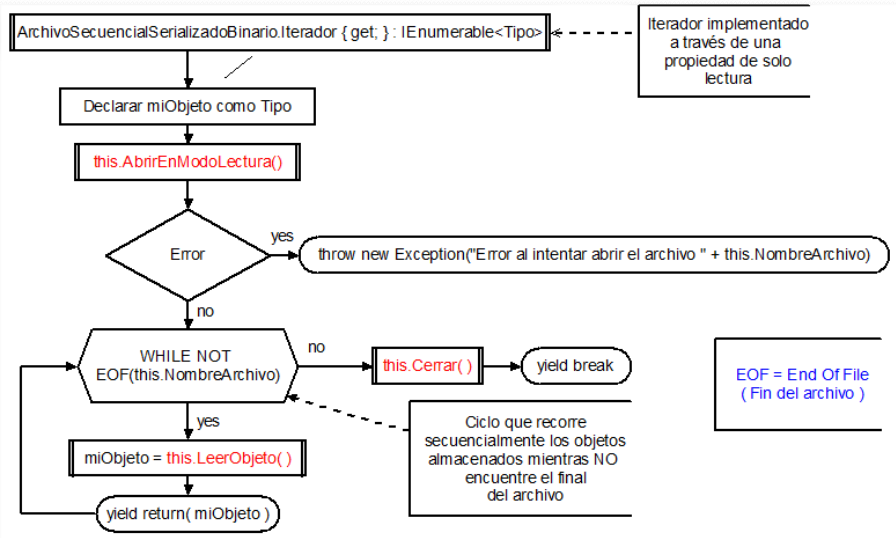
```
void MostrarDatos()
{
    // Declaración y creación de un objeto local
    Empleado miEmpleado = new Empleado();

    // Limpia el dataGridView
    dgEmpleados.Rows.Clear();

    try
    {
      // Abrir el archivo para leer
      miArchivo.AbrirEnModoLectura();

      while(true)
      {
          // Lee un objeto del archivo
          miEmpleado = miArchivo.LeerObjeto();

          // Muestra los datos del objeto leído en el dataGridView
          dgEmpleados.Rows.Add(miEmpleado.Numero.ToString(), miEmpleado.Nombre,
miEmpleado.Sueldo.ToString("C"));
      }
    }
    catch (Exception ex)
    {
    }
    finally
    {
        miArchivo.Cerrar();
    }
}
```

Método para contar los objetos almacenados en el archivo



35

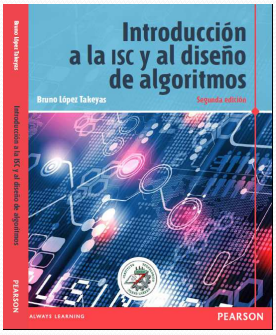
Iterador que devuelve los objetos almacenados en el archivo

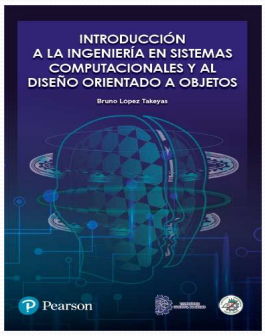


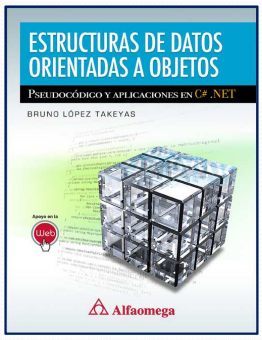
36

Otros títulos del autor

http://www.itnuevolaredo.edu.mx/Takeyas/Libro







bruno.lt@nlaredo.tecnm.mx

Bruno López Takeyas