BINARIOS

[Serializable] -> para binario, va antes de la clase
[NonSerialized] -> para objetos que no se pueden serializar, se declara arriba del atributo objeto

Libreria:

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

SERIALIZAR:

```
public void SerializarBinario(string path)
{
        BinaryFormatter binario = new BinaryFormatter();
        FileStream escritor = new FileStream(path,FileMode.OpenOrCreate);
        binario.Serialize(escritor, this);
        escritor.Close();
}
```

DESERIALIZAR:

```
public void DeserializarBinario(string path)
{
        BinaryFormatter binario = new BinaryFormatter();
        FileStream escritor = new FileStream(path, FileMode.Open);
        Console.WriteLine(((Casteo a clase)(binario.Deserialize(escritor))).ToString());
         escritor.Close();
}
```

SQL

Configuración: click derecho en el proyecto, properties, config, conexión (nombre del string), tipo connection string, autocompleta el tercero, en el último se selecciona la db y el tipo es SQL Auth Windows

LIBRERIA:

```
using System.Data;
using System.Data.SqlClient;

    [NonSerialized]
      SqlConnection sqlcon = new SqlConnection(Properties.Settings.Default.Conexion); //Conexion es el nombre asignado en la primer columna
    [NonSerialized]
       SqlCommand sqlcomm;
```

SQL SELECCIONAR TABLA Y LEER:

```
Public NombreDelConstructor(){}

 public void ListarBD()
 {
    try
    {
       sqlconexion.Open();
       sqlcomander = new SqlCommand();
       sqlcomander.Connection = sqlconexion;
       sqlcomander.CommandType = System.Data.CommandType.Text;
       sqlcomander.CommandText = "SELECT TOP 1000
       [id],[marca],[precio],[color],[trazo],[soloLapiz],[tipo] FROM elementos";
       SqlDataReader reader = sqlcomander.ExecuteReader();
       while (reader.Read())
       {
          for (int i = 0; i < 7; i++)
          {
             Console.Write(reader[i].ToString() + " ");
          }
          Console.Write("\n");
       }
       reader.Close();
       sqlconexion.Close();
    }
    catch (Exception e)
    {
    }
 }
```

OBTENER ALGO DE LA TABLA Y FILTRAR POR ID:

```
public Persona ObtenerPersonaPorIDBD(int id)
    {
       sqlcomm = new SqlCommand();

       sqlcomm.Connection = sqlcon;
       sqlcomm.CommandType = CommandType.Text;
       sqlcomm.CommandText = "SELECT * FROM Personas WHERE Id=" + id;

       sqlcon.Open();
       SqlDataReader sqlread = sqlcomm.ExecuteReader();

       Persona per;
       if (sqlread.HasRows)
       {
          sqlread.Read();
          per = new Persona((int)sqlread[0], (string)sqlread[1], (string)sqlread[2], (int)sqlread[3]);
          sqlcon.Close();
       }
       else
       {
          sqlcon.Close();
             return null;
       }
       return per;
    }
```

INSERTAR OBJETO:

```
public bool AgregarPersonaBD(Persona per)
{
   sqlcomm = new SqlCommand();
   sqlcomm.Connection = sqlcon;
   sqlcomm.CommandType = CommandType.Text;
   sqlcomm.CommandText = "INSERT INTO Personas (nombre,apellido,edad) Values("+"'"+per.nombre+"'"+","+"'"+per.apellido+"'"+","+per.edad+")";
   sqlcon.Open();

   sqlcomm.ExecuteNonQuery();
   sqlcon.Close();
   return true;
}
```

MODIFICAR OBJETO

```
public bool ModificarPersonaBD(Persona per)
{
   sqlcomm = new SqlCommand();
   sqlcomm.Connection = sqlcon;
   sqlcomm.CommandType = CommandType.Text;
   sqlcomm.CommandText = "UPDATE Personas SET nombre = '"+per.nombre+"', apellido = '"+per.apellido+"', edad = '"+per.edad+"' WHERE
id="+per.id;
   sqlcon.Open();

   sqlcomm.ExecuteNonQuery();
   sqlcon.Close();
   return true;
}
```

ELIMINAR OBJETO POR ID (NO NECESITA EL OBJETO, SOLO ID)

```
public bool EliminarPersonaBD(int id)
{
   sqlcomm = new SqlCommand();
   sqlcomm.Connection = sqlcon;
   sqlcomm.CommandType = CommandType.Text;
   sqlcomm.CommandText = "DELETE FROM Personas WHERE ID="+id;
   sqlcon.Open();

   sqlcomm.ExecuteNonQuery();
   sqlcon.Close();
   return true;
}

public static List<Persona> ObtenerPersonaHC()
{

   Persona per1 = new Persona(1, "Lucas", "Massa", 20);
   Persona per2 = new Persona(2, "Pancho", "Di Marzo", 21);
   Persona per3 = new Persona(3, "Santiago", "Bonazi", 20);
   Persona per4 = new Persona(4, "Martin", "Alberio", 20);
   List<Persona> lista = new List<Persona>();
   lista.Add(per1);
   lista.Add(per2);
   lista.Add(per3);
   lista.Add(per4);

   return lista;
}
```

ENCONTRAR Y OBTENER OBJETO Y RETORNAR OBJETO

```
public static Persona ObtenerPersonaPorID(int id)
{
   List<Persona> lista = ObtenerPersonaHC();

   foreach (Persona i in lista)
   {
      if (i.id == id)
         return i;
   }
   return null;
}
```

//AGREGAR EL OBJETO A LA DB

```
public static bool AgregarPersona(Persona per)
{
   List<Persona> lista = ObtenerPersonaHC();
   lista.Add(per);
   return true; //uy

}
```

MODIFICAR OBJETO EN LA DB

```
public static bool ModificarPersona(Persona per)
{
   List<Persona> lista = ObtenerPersonaHC();

   for (int i = 0; i < lista.Count; i++)
   {
      if (per.id == lista[i].id)
      {
         lista[i] = per;
         return true;
      }
   }
   return false;
}
```

ELIMINAR OBJETO EN LA DB

```
public static bool EliminarPersona(Persona per)
{
    List<Persona> lista = ObtenerPersonaHC();
    Persona per2 = ObtenerPersonaPorID(per.id);
    if (lista.Remove(per2))
    {
        return true;
    }
    return false;
}
```

EVENTOS Y DELEGADOS

```
Evento e = new Evento();
//Se crea el evento y asigna el metodo estatico al evento
e.evento += OnEvento;
//hace lo mismo que la primera pero con el EventHandler, ambas funcionan igual
e.evento += new Evento.EventHandler(OnEvento);

e.OnEvento("HolaMundo");

public class Evento
 {
        public delegate void EventHandler(String s);

        public event EventHandler evento;

        public void OnEvento(String s)
        {
           if (evento != null)
           evento(s);
        }
      }

public class EventArgsMiClase:EventArgs
{
        Public var nombre;
        Public var loQueQuieras;

        Public EventArgsMiClase(var nombre, var loQueQuieras)
        {
                this.nombre=nombre;
                this.loQueQuieras=loQueQuieras;
        }
}
```

// se puede pasar como si fuera un eventargs al ejecutar el delegado/evento, se usa para pasar datos y atributos que no estén contenidos en el objeto sender y que puedan llegar a usarse en el método (por ejemplo, valor que provocó el evento y no se cargó), en caso de no necesitar pasar parámetros se usa EventArgs.Empty

XML

IMPORTANTE LAS CLASES QUE SE VAN A ESCRIBIR TIENEN QUE TENER ATRIBUTOS PUBLICOS Y UN CONSTRUCTOR POR DEFECTO

LIBRERIA:

using System.Xml.Serialization;
Using System.IO;

INCLUIR OTRAS CLASES:

En las clases que se van a serializar en el namespace de esa clase es recomendado usar:
[XmlInclude(typeof(ClaseHeredada))] *->Va en la clase principal, los typeof de las heredadas*
[Serializable]

Serializar XML:

```
XmlSerializer serializador = new XmlSerializer(this.GetType()); //TIPO QUE VA A GUARDAR

TextWriter escritor= new StreamWriter(path, false); //EL STREAMWRITER QUE SE USA PARA ESCRIBIR

//METODO PARA SERIALIZAR, EL PRIMER PARAMETRO ES EL SW Y EL SEGUNDO LA CLASE QUE SE GUARDA
serializador.Serialize(escritor, this);
escritor.Close();
```

DESERIALIZAR:

```
XmlSerializer serializador = new XmlSerializer(this.GetType());
TextReader lector = new StreamReader(path);
Console.WriteLine(((Cartuchera<T>)serializador.Deserialize(lector)).ToString());
lector.Close();
```

//ESCRIBIR COMÚN Y CORRIENTE EN TXT UN STRING

```
public static void Escribir(object o, EventArgs sender)
{
    TextWriter sw = new StreamWriter("Escritura.txt", true);
    sw.WriteLine(DateTime.Now.ToString() + " | " + ((Cartuchera<Utiles>)o).ultimoAgregado.ToString());
    sw.Close();
}
```

FORMS

```
Public constructor()
{
        this.dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
        this.dataGridView1.MultiSelect = false;
}

        frmABM formABM = new
        frmABM(prov.ObtenerPersonaPorIDDB(int.Parse(dataGridView1.SelectedCells[0].Value.ToString())),0);
        //obtiene el id (primer columna) y utiliza método que devuelve la persona de la db en base a ese id



frmABM formABM = new frmABM(this.lista[listBox1.SelectedIndex], 1);
```

//Si hay una lista en vez de una db y se usa un datagridview, se obtiene el elemento en el indice de la lista, la idea es removerlo

OPENFILEDIALOG:

INTERFAZ

Metodos sin cuerpos que se implementan en otros lados.

```
        public interface IArchivo<T>
        {
            bool Guardar(string archivo, T datos);
            bool Leer(string archivo, out T datos);
        }
    }
```

THREADING

Librerias: System.Threading;

Métodos: Thread.Start(); | Thread.Start(xxx); Si es parametrizado **| Thread.Sleep(time);**

```
class Mensajes
  {
     public void Mostrar1()
     {
        for (int i = 0; i < 10; i++)
        {
           Console.WriteLine("Escribiendo desde ==> 1");
           Thread.Sleep(1000);
        }
        Console.WriteLine("Fin Hilo 1");
     }

     public void Mostrar2()
     {
        for (int i = 0; i < 10; i++)
        {
           Console.WriteLine("Escribiendo desde ==> 2");
           Thread.Sleep(1200);
        }
        Console.WriteLine("Fin Hilo 2");
     }

     public void MostrarConParametros(object p)
     {
        for (int i = 0; i < 10; i++)
        {
           Console.WriteLine("Escribiendo desde ==> {0}",p);
           Thread.Sleep(1500);
        }
        Console.WriteLine("Fin Hilo con parámetros");
     }
  }

   class Program
  {
     static void Main(string[] args)
     {
        Mensajes mensaje = new Mensajes();
        //INVOCO EXPLICITAMENTE AL DELEGADO
        Thread hilo1 = new Thread(new ThreadStart(mensaje.Mostrar1));
        //INVOCO IMPLICITAMENTE AL DELEGADO
        Thread hilo2 = new Thread(mensaje.Mostrar2);
        //INVOCO A DELEGADO QUE ADMITE UN PARAMETRO (OBJECT)
        Thread hilo3 = new Thread(new ParameterizedThreadStart(mensaje.MostrarConParametros));
        //INICIO LOS HILOS (INVOCANDO AL DELEGADO)
        hilo1.Start();
        hilo2.Start();
        hilo3.Start("con parametros");


        Console.ReadLine();

     }
  }
```

OPEN SAVE DIALOG

```
private void ConfigurarOpenSaveDialog(object sender, EventArgs e)
    {
       this.openFileDialog1.Multiselect = false;
       this.openFileDialog1.Filter = "Archivo  de imagen (.jpg)|.jpg|All Files (.)|.";
       this.openFileDialog1.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);
       this.openFileDialog1.FileName = "";
       this.openFileDialog1.Title = "Seleccione una foto...";
       this.openFileDialog1.ShowDialog();
       if (!this.openFileDialog1.CheckPathExists)
       {
          MessageBox.Show("Error: El directorio no existe");
       }
       else if (!this.openFileDialog1.CheckFileExists)
       {
          MessageBox.Show("Error: El archivo no existe");
       }
       this.txtFoto.Text = openFileDialog1.FileName;
    }
```