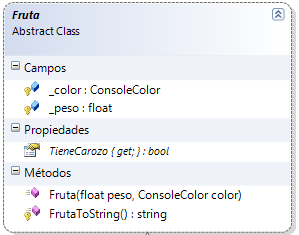
Codificar en C Sharp la siguiente jerarquía de clases en un proyecto de tipo ***Class Library***.



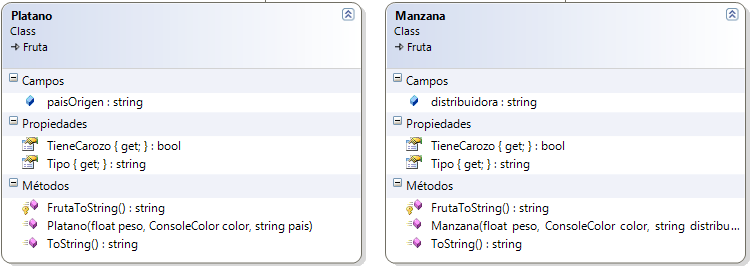
Clase Fruta

Propiedad

***TieneCarozo*** (abstracta; sólo lectura).

Métodos

***FrutaToString***:(de instancia; protegido; virtual) retorna la representación de la clase en formato string.



|  |  |
| --- | --- |
| Clase Manzana (hereda de Fruta)  Propiedades  ***TieneCarozo***:(sólo lectura) retornará **true**.  ***Tipo***: (sólo lectura) retornará “**Manzana**”.  Métodos  ***FrutaToString***: Se deberá reutilizar agregándole los campos propios.  ***ToString***: retorna la representación de la clase en formato string. | Clase Platano (hereda de Fruta)  Propiedades  ***TieneCarozo***:(sólo lectura) retornará **false**.  ***Tipo***: (sólo lectura) retornará “**Plátano**”.  Métodos  ***FrutaToString***: Se deberá reutilizar agregándole los campos propios.  ***ToString***: retorna la representación de la clase en formato string. |

Clase Cajon<T>

Propiedades

***Frutas***:(sólo lectura) expone al atributo de tipo List<T>.

***PrecioTotal***:(sólo lectura) retorna el precio total del cajón. Si el precio total supera los 25 pesos, se disparará el evento ***EventoPrecio***. Diseñarlo de acuerdo a las convenciones vistas. Crear el manejador necesario para que se imprima en un archivo de texto: la fecha (con hora, minutos y segundos) y el total del precio del cajón.

Constructores

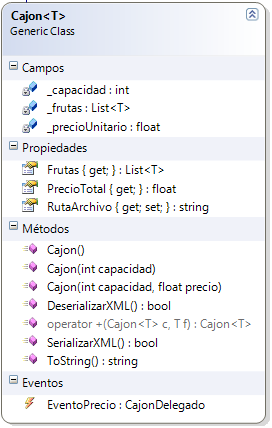
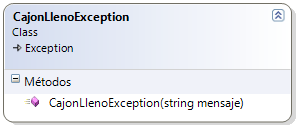
Por default: es el único que se encargará de inicializar la lista.

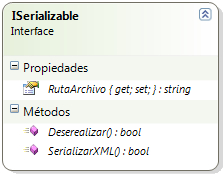
Métodos

***ToString***: Mostrará en formato de tipo string, la capacidad, la cantidad total de frutas, el precio total y el listado de todas las frutas contenidas en el cajón. Reutilizar código.

Sobrecarga de operador

***(+)*** Será el encargado de agregar frutas al cajón, siempre y cuando no supere la capacidad del mismo. Caso contrario se lanzará una ***CajonLlenoException***, cuyo mensaje explicará lo sucedido.





Interface ISerializable

Propiedad

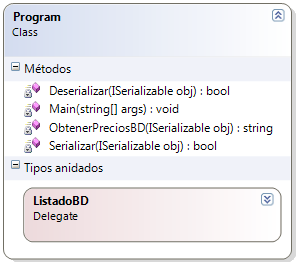
***RutaArchivo***: Establece/obtiene el path dónde se guardará el archivo XML.

Método

***SerializarXML***: serializará a XML la instancia que lo invoque, retornando true si lo pudo hacer; false, caso contrario.

***DeserializarXML***: deserializará de XML a la instancia que lo invoque, retornando true si lo pudo hacer; false, caso contrario.

## Implementar la interface ISerializable en Manzana y Cajón.



Clase Program

Métodos

***Deserializar***:(privado; de clase) recibe como parámetro un objeto de tipo ISerializable y retornará true si pudo deserializar; false, caso contrario. Reutilizar código (sólo una línea de código.)

***Main***: punto de inicio de la aplicación.

***ObtenerPreciosBD***:(privado; de instancia) recibe como parámetro un objeto de tipo ISerializable y retornará un listado (en formato string) de precios obtenidos de la base de datos ***Precios***, la cual posee una única tabla ***PreciosFruta*** { ***id*** (autoincremental - numérico) - ***descripcion*** (cadena) - ***precio*** (numérico) }. Dicho método será invocado por el ***delegado*** ***ListadoBD***.

***Serializar***:(privado; de clase) recibe como parámetro un objeto de tipo ISerializable y retornará true si pudo serializar; false, caso contrario. Reutilizar código (sólo una línea de código.)

Main

class Program

{

static void Main(string[] args)

{

Cajon<Fruta> cajon = new Cajon<Fruta>(3, 15.36f);

Cajon<Platano> cajonplatanos = new Cajon<Platano>(4, 13.5f);

Manzana m1 = new Manzana(15.0f, ConsoleColor.Red, "Moño Azul");

Manzana m2 = new Manzana(22.06f, ConsoleColor.Magenta, "Parajes pampeanos");

Manzana m3 = new Manzana(5.5f, ConsoleColor.DarkYellow, "El abuelo");

Platano p1 = new Platano(4.8f, ConsoleColor.Yellow, "Ecuador");

Platano p2 = new Platano(21.7f, ConsoleColor.DarkYellow, "Colombia");

Platano p3 = new Platano(16.99f, ConsoleColor.Yellow, "Brasil");

try

{

cajon += m1;

cajon += m2;

cajon += p1;

cajon += p2;

cajon += m3;

}

catch (Exception exc)

{

Console.WriteLine(exc.Message);

}

Console.WriteLine(cajon);

//EVENTOS

Console.WriteLine(cajon.PrecioTotal);

try

{

cajonplatanos += p1;

cajonplatanos += p2;

cajonplatanos += p3;

}

catch (Exception e)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(e.Message.ToString());

Console.ForegroundColor = ConsoleColor.Green;

}

//SERIALIZAR CAJON

cajon.RutaArchivo = "cajon.xml";

if (Program.Serializar(cajon))

Console.WriteLine("Serializado correctamente.");

else

Console.WriteLine("Error al serializar.");

//SERIALIZAR MANZANA

m2.RutaArchivo = "manzana.xml";

if (Program.Serializar(m2))

{

Console.WriteLine("Manzana serializada correctamente.");

}

else

{

Console.WriteLine("Error al serializar Manzana.");

}

//DESERIALIZAR CAJON

if (Program.Deserializar(cajon))

Console.WriteLine(" > Cajon deserializado correctamente.");

else

Console.WriteLine(" > Error al deserializar Cajon.");

//DESERIALIZAR MANZANA

if (Program.Deserializar(m2))

Console.WriteLine(" > manzana deserializada correctamente.");

else

Console.WriteLine(" > Error al deserializar manzana.");

//BASE DE DATOS

ListadoBD dele = new ListadoBD(Program.ObtenerPreciosBD);

Console.WriteLine(dele.Invoke(cajon));

Console.ReadLine();

}

public delegate string ListadoBD(ISerializable obj);

private static bool Serializar(ISerializable obj)

{

return obj.SerializarXML();

}

private static bool Deserializar(ISerializable obj)

{

return obj.DeserializarXML();

}

private static string ObtenerPreciosBD(ISerializable obj)

{

SqlConnection sqlcon = new SqlConnection(Properties.Settings.Default.Conexion);

SqlCommand sqlcomm;

sqlcon.Open();

sqlcomm = new SqlCommand();

sqlcomm.Connection = sqlcon;

sqlcomm.CommandType = System.Data.CommandType.Text;

sqlcomm.CommandText = "SELECT \* FROM PreciosFrutas";

SqlDataReader lector = sqlcomm.ExecuteReader();

while (lector.Read())

{

Console.WriteLine(lector.ToString()+"\n");

}

lector.Close();

sqlcon.Close();

return lector.ToString();

}

}