Fabrica de indumentaria. Laredo Agustin 2C

Este programa sirve para poder manufacturar Indumentarias a partir de modelos disponibles. Nuestra fabrica puede generar “Modelos” de camisetas y zapatillas para ponerlas a disposición, y luego ser manufacturarlas creando un documento de manufactura.

Para lograr este funcionamiento se optó por que la fabrica será una clase estática, solo tendremos una en cada ejecucion de la aplicación. Esta clase estática tiene dos listas, una de “disponibles” en los que se cargan los modelos que podrán ser producidos, estos modelos no deben ser repetidos. Y otra lista de manufacturados que tienen una cantidad incremental de objetos si ya se encuentran en la lista.

Los mismo se pensó para la generación y lectura de documentos con la clase DocumentosFabrica. Esta clase nos permite guardar documentos con objetos serializados de cualquier clase hija de Indumentaria, gracias a que este método de generación de archivos recibe parámetro de tipo. Para leer se aplico la misma lógica.

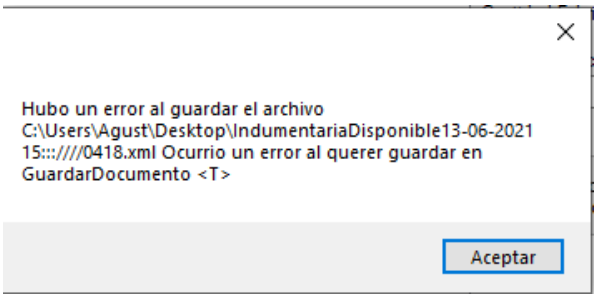
Por la parte de objetos, se decidió usar una clase abstracta Indumentaria que implementa la interface IFabricable que obliga a tener las propiedades CostoProduccion, CantidadManufacturada y el método Fabricar() que es el responsable de agregar la indumentaria disponible a la lista de producción

Conceptos Aplicados

1.Excepciones.

Se lanzaron excepciones desde la lógica que se controlaron y mostraron en la lógica.

Al cerrar la aplicación se guarda en XML la lista de indumentarias disponibles para que se lean la próxima vez a ser lanzada la aplicación. Este es un error en el caso de que falle el guardado



Este mensaje se lanza cuando al querer guardar el documento de disponibles triggereado por el evento FormClosing, haciendo un catch y facilitándonos el path del archivo que no se logró guardar, pero también nos muestra el mensaje de la innerException, que en este caso es la excepción lanzada desde la logica

```
private void FrmPrincipal_FormClosing(object sender, FormClosingEventArgs e)
{
    string nombreArchivo = "\\IndumentariaDisponible" + DateTime.Now.ToString("dd-MM-yyyy HH:::////mmss") + ".xml";

    string pathArchivo = Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + nombreArchivo;

    try
    {
        DocumentosFabrica.GuardarDocumento<Indumentaria>(pathArchivo);
    }
    catch (Exception error)
    {
        MessageBox.Show("Hubo un error al guardar el archivo " + pathArchivo + error.Message);
    }

}
```

Mientras que este error que catcheamos se produjo en el método DocumentosFabrica.GuardarDocumento<T>

```
public static void GuardarDocumento<T>(string pathArchivo) where T : Indumentaria
{


    try
    {
        XmlSerializer ser = new XmlSerializer(typeof(List<T>));
        Stream myStream = new FileStream(pathArchivo, FileMode.Create, FileAccess.Write);
        if(pathArchivo.Contains("Produccion"))
        {
            ser.Serialize(myStream, Fabrica.ListaIndumentariaProduccion<T>());
        }
        else if (pathArchivo.Contains("Disponible"))
        {
            ser.Serialize(myStream, Fabrica.ListaIndumentariaDisponible<T>());
        }

        myStream.Close();
    }
    catch (Exception ex)
    {
        throw new IndumentariasExceptionsErrorAlGenerarArchivo(" Ocurrio un error al querer guardar en GuardarDocumento <T>" , ex);
    }
}
```

2.Test Unitarios

Se probaron los principales métodos de la lógica.

Acá se testea la capacidad de generar listas con tipos especificados (Camisetas o Zapatillas en este caso) en el parámetro de tipo, basándose en la lista general de la fabrica de tipo Indumentaria

```
[TestMethod]
0 referencias
public void FiltrarTipoIndumentaria()
{
    //Testeo que se suman 4 zapatillas ninguna camiseta
    this.CuatroIndumentarias("Disponible-Zapatillas");

                } else if (parametros.Contains("Zapatilla"))
                {
                    Fabrica.agregarADisponible(indumentariaBienZapa1);
                    Fabrica.agregarADisponible(indumentariaBienZapa2);
                    Fabrica.agregarADisponible(indumentariaBienZapa3);
                    Fabrica.agregarADisponible(indumentariaBienZapa4);
                } else if (parametros.Contains("Camiseta"))

    Assert.IsTrue(Fabrica.ListaIndumentariaDisponible<Camiseta>().Count == 0);
    Assert.IsTrue(Fabrica.ListaIndumentariaDisponible<Zapatilla>().Count == 4);
    Assert.IsTrue(Fabrica.ListaIndumentariaDisponible<Indumentaria>().Count == 4);
    Fabrica.indumentariaDisponible.Clear();
    //Testeo que se suman 4 camisetas ninguna camiseta
    this.CuatroIndumentarias("Disponible-Camiseta");
    Assert.IsTrue(Fabrica.ListaIndumentariaDisponible<Camiseta>().Count == 4);
    Assert.IsTrue(Fabrica.ListaIndumentariaDisponible<Zapatilla>().Count == 0);
    Assert.IsTrue(Fabrica.ListaIndumentariaDisponible<Indumentaria>().Count == 4);
    Fabrica.indumentariaDisponible.Clear();
}
```

3. **Generics.**

Estas pruebas chequean que esta función logre su cometido, solo con su parámetro de tipo es capaz de devolver una lista genérica de ese mismo tipo filtrando la lista completa de Indumentaria

```
18 referencias | 0/4 pasando
public static List<T> ListaIndumentariaDisponible<T>() where T : Indumentaria
{
    try
    {
        List<T> listaRetorno = Fabrica.indumentariaDisponible.OfType<T>().ToList();
        return listaRetorno;
    }
    catch (Exception error)
    {
        throw new Exception("Error al generar informe de indumentaria disponible", error);
    }

}
```

4. **Interfaces**

Mi indumentaria debe ser Fabricable, es decir que todo objeto Indumentaria debe implementar la interface IFabricable, que obliga a implementar las propiedades CostoProduccion, CantidadManufacturada y el método Fabricar().

Declaración:

```
namespace TP3
{
    1 referencia
    public interface IFabricable<T> where T : Indumentaria
    {
        7 referencias
        int CantidadManufacturada { get; set; }

        5 referencias
        float CostoProduccion { get; set; }

        7 referencias
        void Fabricar();
    }
}
```

Implementación

```
public int CantidadManufacturada
{
    get { return this.cantidadManufacturada; }
    set
    {
        if (value >= 0)
        {
            this.cantidadManufacturada = value;
        }
        else
        {
            throw new Exception("No se puede manufacturar menos de una prenda");

        }
    }
}

5 referencias
public virtual float CostoProduccion
{
    set
    {
        if (value <= 0)
        {
            throw new IndumentariasExceptionsPrecioErroneo("El precio no es valido!");
        }
        else
        {
            this.costoProduccion = value;
        }
    }
    get
    {
        return this.costoProduccion;
    }
}

7 referencias
public void Fabricar()
{
    Fabrica.agregarATandaDeProduccion(this);
    this.cantidadManufacturada++;

}
```

5. Archivos

Como mencionamos antes al momento de cargar nuestra fabrica leeremos el ultimo documento de indumentariaDisponible.

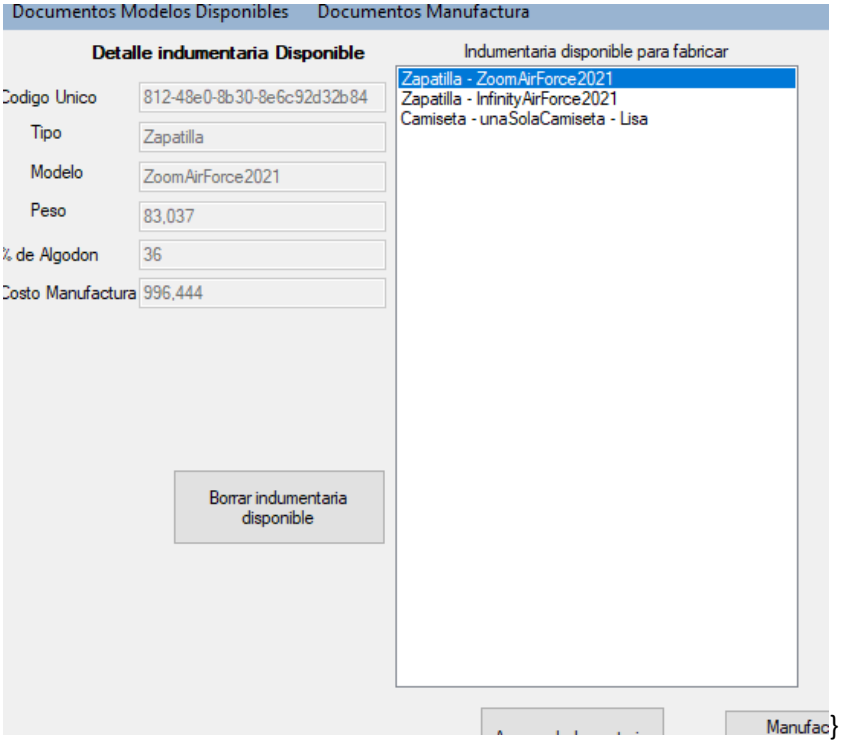
```
private void FrmPrincipal_Load(object sender, EventArgs e)
{

    string[] files = Directory.GetFiles((Environment.GetFolderPath(Environment.SpecialFolder.Desktop)), "IndumentariaDisponible???????????????.xml");

    if (files.Length == 0)
    {
        this.lstBoxInduManufacturada.DataSource = Fabrica.ListaIndumentariaProduccion<Indumentaria>();
    }
    else
    {
        DocumentosFabrica.LeerDocumento(files[files.Length - 1], out Fabrica.indumentariaDisponible);
        this.lstBoxInduDisponible.DataSource = Fabrica.ListaIndumentariaDisponible<Indumentaria>();
        this.lstBoxInduManufacturada.DataSource = Fabrica.ListaIndumentariaProduccion<Indumentaria>();
    }

}
```

De similar manera y con el uso de Generics se puede guardar una lista de disponibles con solo elementos de un tipo pasado por parámetro de tipo, mediante las opciones del menuToolstrip.

```
1 referencia
private void documentosDisponiblesSoloCAMISETASToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (Fabrica.ListaIndumentariaDisponible<Camiseta>().Count == 0)
    {
        MessageBox.Show("No se encuentran camisetas poner a disponibilidad");
        return;
    }

    string nombreArchivo = "\\IndumentariaDisponibleCamisetas " + DateTime.Now.ToString("dd-MM-yyyy HHmmss") + ".xml";


    string pathArchivo = Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + nombreArchivo;

    try
    {
        DocumentosFabrica.GuardarDocumento<Camiseta>(pathArchivo);
    }
    catch (IndumentariasExceptionsErrorAlGenerarArchivo error)
    {
        MessageBox.Show("Hubo un error al generar el archivo " + nombreArchivo + error.Message);
    }
}
```

Nótese que hacemos una verificación, para que no se ejecute la operación sea el caso que no se encuentren Camisetas disponibles, y haciendo un catch y mostrando si el problema llegara a darse al momento de guardar la lista

UI:



A la izquierda es el resultado de guardar solo camisetas y a la izquierda se muestra el guardado que se produce al salir de la aplicación que contempla todas las indumentarias

```
<?xml version="1.0"?>
- <ArrayOfCamiseta xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  - <Camiseta>
      <CantidadManufacturada>0</CantidadManufacturada>
      <CostoProduccion>1000</CostoProduccion>
      <CodigoUnico>8e2-4477-8b3a-203b8889dbc8</CodigoUnico>
      <Peso>25.5</Peso>
      <PorcentajeAlgodon>0</PorcentajeAlgodon>
      <Modelo>unaSolaCamiseta</Modelo>
      <Estampado>false</Estampado>
    </Camiseta>
</ArrayOfCamiseta>
```

```
<?xml version="1.0"?>
- <ArrayOfIndumentaria xmlns:xsd="http://www.w3.org/2001/XMLSc
  - <Indumentaria xsi:type="Zapatilla">
      <CantidadManufacturada>1</CantidadManufacturada>
      <CostoProduccion>996.444031</CostoProduccion>
      <CodigoUnico>29d-47d6-8656-eaf8a51ddf19</CodigoUnico>
      <Peso>83.037</Peso>
      <PorcentajeAlgodon>36</PorcentajeAlgodon>
      <Modelo>ZoomAirForce2021</Modelo>
      <Capellada>Zoom</Capellada>
      <Suela>AirForce</Suela>
    </Indumentaria>
  - <Indumentaria xsi:type="Zapatilla">
      <CantidadManufacturada>0</CantidadManufacturada>
      <CostoProduccion>1516.155</CostoProduccion>
      <CodigoUnico>79f-49e1-ae54-f5d7a6ca8132</CodigoUnico>
      <Peso>74.565</Peso>
      <PorcentajeAlgodon>61</PorcentajeAlgodon>
      <Modelo>InfinityAirForce2021</Modelo>
      <Capellada>Infinity</Capellada>
      <Suela>AirForce</Suela>
    </Indumentaria>
  - <Indumentaria xsi:type="Camiseta">
      <CantidadManufacturada>0</CantidadManufacturada>
      <CostoProduccion>1000</CostoProduccion>
      <CodigoUnico>8e2-4477-8b3a-203b8889dbc8</CodigoUnico>
      <Peso>25.5</Peso>
      <PorcentajeAlgodon>0</PorcentajeAlgodon>
      <Modelo>unaSolaCamiseta</Modelo>
      <Estampado>false</Estampado>
    </Indumentaria>
</ArrayOfIndumentaria>
```