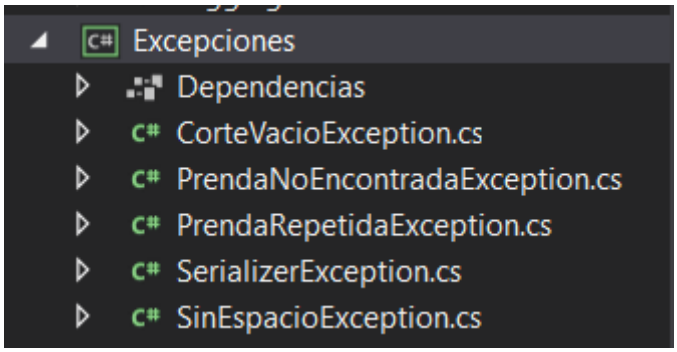
El programa creado se usa para la gestión de cortes de una fabrica textil, la cual puede fabricar solo las marcas implementadas.
(1 corte de ropa son muchas prendas, generalmente salen en curvas parejas)
En las funcionalidades que podemos encontrar tenemos el ABM y también exportar e importar en un formato serializado, el archivo es creado y tomado desde el escritorio.

La fabrica tiene una capacidad de 1000 prendas

Temas de clases implementados:

Clase 15 – Excepciones: Posee 5 excepciones las cuales se detallan a continuación, también atrapa excepciones no controladas por si ocurre algún error no esperado.

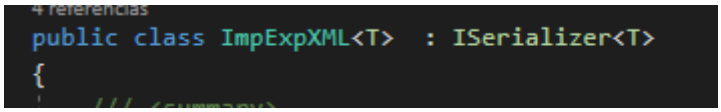


Clase 16 – Test Unitarios: Se realizaron 3 pruebas diferentes y corrieron con éxito
1) Que se instancien las listas de la fabrica

2) Al agregar un corte nuevo superar el limite y que lance una excepción

3) Que agregue bien un corte

Clase 17 – Tipos genéricos: En la clase para guardar el archivo se implemento ya que puede recibir un tipo de dato genérico para luego generalizar el que sea.



Clase 18 – Interfaces: En la clase indumentaria y en el proyecto de serilizer se implementó este tema, indicándole la estructura que debe usar

```
interface IPrenda
{
    13 referencias
    int Id { get; set; }
    9 referencias
    int S { get; set; }
    9 referencias
    int M { get; set; }
    9 referencias
    int L { get; set; }
    9 referencias
    int XL { get; set; }
    9 referencias
    int XXL { get; set; }

    3 referencias
    enumPrenda Tipo { get; set; }

    4 referencias
    enumMarca Marca { get; set; }

    4 referencias
    string Modelo { get; set; }

    8 referencias
    string Variante { get; set; }
}
```

```
1 referencia
interface ISerializer<T>
{
    2 referencias
    bool Guardar(string path, T datos);

    2 referencias
    bool Leer(string path, out T datos);
}
```

Clase 19 – Archivos y serialización: En el proyecto Serializer esta implementado todo en formato XML

```
2 referencias
public bool Guardar(string path, T data)
{
    XmlTextWriter writer = null;
    XmlSerializer serializer = null;
    try
    {
        writer = new XmlTextWriter(path, Encoding.UTF8);
        writer.Formatting = Formatting.Indented;
        serializer = new XmlSerializer(typeof(T));
        serializer.Serialize(writer, data);
        return true;
    }
    catch (Exception ex)
    {
        throw new SerializerException(ex);
    }
    finally
    {
        if (writer != null)
        {
            writer.Close();
        }
    }
}
```

```
2 referencias
public bool Leer(string path, out T data)
    {
        XmlTextReader reader = null;
        XmlSerializer serializer = null;
        try
        {
            reader = new XmlTextReader(path);
            serializer = new XmlSerializer(typeof(T));
            data = (T)serializer.Deserialize(reader);
            return true;
        }
        catch (Exception ex)
        {
            throw new SerializerException(ex);
        }
        finally
        {
            if (reader != null)
            {
                reader.Close();
            }
        }

    }
```

Clase 21/22 – Bases de datos: En la clase acceso a datos.

```
public List<Indumentaria> LoadFromBD()
{
    List<Indumentaria> lista = new List<Indumentaria>();

    try
    {
        this.comando = new SqlCommand();

        this.comando.CommandType = CommandType.Text;

        this.comando.Connection = this.conexion;

        comando.CommandText = "SELECT * FROM Cortes ORDER BY Id";

        this.conexion.Open();

        SqlDataReader oDr = comando.ExecuteReader();

        while (oDr.Read())
        {
            if(oDr["Tipo"].ToString()=="Jogging")
            {
                int Id = oDr.GetInt32(0);
                int S = oDr.GetInt32(1);


                Joggings corteJ = new Joggings(oDr.GetInt32(0), oDr.GetInt32(1), oDr.GetInt32(2), oDr.GetInt32(3), o
                if (oDr["Variante"].ToString()=="Con punio")
```

Clase 23 – Hilos: En la clase del frmPrincipal se aplica un hilo para refrescar cada 500 ms los datos.

```
1 referencia
public void ActualizarGrilla()
{
    try
    {
        while (this.pauseThread == false)
        {
        Thread.Sleep(500);

            if (this.dtGdViewCortes.SelectedRows.Count > 0)
            {
                rowSelected = this.dtGdViewCortes.SelectedRows[0].Index;
            }

            if (this.dtGdViewCortes.InvokeRequired)
        {
            this.dtGdViewCortes.BeginInvoke((MethodInvoker)delegate ()
            {
                dtGdViewCortes.DataSource = null;
                this.dtInd = _acceso.LoadDTFromBD();

                dtGdViewCortes.DataSource = this.dtInd;
                configurarGrilla();

            });
        }
        }
    }
    catch(Exception)
    {
```

Clase 24 – Eventos y delegados: En el método ActualizarGrilla uso delegados y para direccionar el click del botón a otra función y cuando cierra el formulario.

```
    }
    /// <summary>
    /// Al cerrar el formulario cambia la bandera para que no se siga ejecutando el hilo
    /// </summary>
    1 referencia
    private void Cerrar(object sender, FormClosingEventArgs e)
    {
        pauseThread = true;
    }
}
```

Clase 25 – Métodos de extensión: En el proyecto extensor, se extiende la clase fabrica para verificar el espacio disponible .

```
    /// <summary>
    /// Extiende la fabrica y verifica el espacio disponible
    /// </summary>
    /// <param name="FB"></param>
    /// <returns>Espacio disponible</returns>
    2 referencias
    public static int Verificador(this Fabrica FB)
    {

        try
        {

            AccesoDatos _acceso = new AccesoDatos();
            int espacioOcupado = _acceso.EspacioOcupadoBD();
            if (espacioOcupado >= FB.EspacioTotal)
            {
                throw new SinEspacioException();
            }else
            {
                return FB.EspacioTotal-espacioOcupado;
            }

        }
        catch(Exception)
        {
            throw new SinEspacioException();
        }

    }
```

Para crear la BD se debe ejecutar las siguientes instrucciones:

```
USE [Santoliquido_TP4_BD]
GO

/** Object:  Table [dbo].[Cortes]    Script Date: 9/7/2021 18:29:56 **/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Cortes](
	[Id] [bigint] IDENTITY(1,1) NOT NULL,
	[S] [int] NULL,
	[M] [int] NULL,
	[L] [int] NULL,
	[XL] [int] NULL,
	[XXL] [int] NULL,
	[Tipo] [varchar](50) NULL,
	[Marca] [varchar](50) NULL,
	[Modelo] [varchar](50) NULL,
	[Variante] [varchar](50) NULL,
 CONSTRAINT [PK_Cortes] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```