Contexto:

Es un programa para atención al cliente de un club de vinos, se asocian clientes pagando una cuota mensual y a cambio se le envían una caja distinta de vino cada mes.

Funcionalidad:

Agregar Cliente: Se deberá rellenar los campos pedidos y presionar el botón AGREGAR. Se deberá ser mayor de edad para agregar al cliente.

Dar de baja Cliente: Se deberá poner el DNI del cliente a eliminar en el txt en pantalla, si esta todo bien se mostrara en el rich a partir de ahí se podrá eliminar el cliente presionando en el botón especifico.

Modificar Cliente: Se deberá poner el DNI y buscar igual que en el apartado de dar de baja, si el cliente existe se mostrarán todos sus datos, desde ahí mismo se podrá modificar el cliente, por último, presionar el guardar.

Stock: Se debe seleccionar el tipo y la cantidad. Luego presionar en agregar o eliminar dependiendo que se quiera hacer.

Temas:

Excepciones: Este tema esta por todo el código, básicamente en cada form hay una excepción. Luego en la clase genérica “listado” en los métodos ADD y REMOVE, lanzo una excepción.

```
2 referencias
public bool add(T obj)
{
    bool retorno = false;
    if (obj is not null)
    {
        if (!EstaEnLaLista(this.lista, obj))
        {
            this.lista.Add(obj);
            retorno = true;
        }
        else
        {
            throw new EstaOnoEnlalista("Ya se encuentra en la lista!");
        }

    }
    return retorno;
```

```
1 referencia
public bool Remove(T obj)
{
    bool retorno = false;
    if (obj is not null)
    {
        if (this.EstaEnLaLista(this.lista, obj))
        {
            this.lista.Remove(obj);
            return true;
        }
        else
        {
            throw new EstaOnoEnlalista("No se encuentra en la lista");

        }
    }
    return retorno;
```

Genéricos: En la biblioteca de clases entidades esta la clase LISTADO que la utilizo tanto para los clientes como para el stock de las cajas de vino. Además, utilizo genéricos en la interfaz Iarchivos para usarla en la clase de archivos txt y la serializadora. Por último en la clase serializadora para serializar tanto el stock de cajas de vino como los clientes.

Interfaces: Utilizo la interfaz Iarchivos en la clase Archivotxt y claseSerializadoraXml.

```
public class ClaseSerializadoraXml<T> : IArchivos<T>
{

    protected string path;
```

```
3 referencias
public class ArchivoTxt : IArchivos<string>
{
    public string path;
    public string nombreArchivo;
```

```
public interface IArchivos<T>
{
    5 referencias
    T Leer(string nombreArchivo);
    5 referencias
    void Escribir(T elemento, string nombreArchivo);
}
```

Archivos: Para el apartado de archivos lo guardo dentro del debug de la misma aplicación.