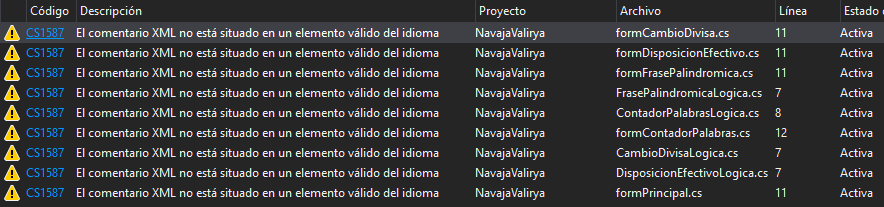
9-. Prueba las funcionalidades de análisis de código de Visual Studio para mejorar el código y resuelve, al menos, dos de las advertencias diferentes de las que te indica, documentando cómo lo has hecho.

**9.1)** Una vez ejecutado el análisis con las ***reglas mínimas administradas por Microsoft*** me da los siguientes errores (9 líneas):



La advertencia CS1587 significa que hay un comentario que no está situado correctamente.

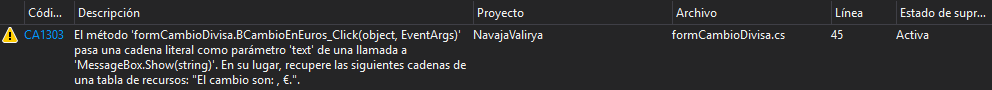
Corresponde a los comentarios de los namespace del proyecto que no se deben comentar tal y como indica el siguiente enlace.

<https://docs.microsoft.com/es-es/dotnet/articles/csharp/misc/cs1587>

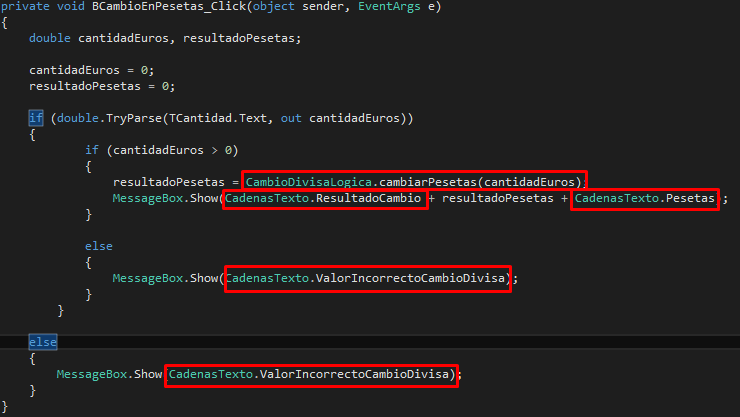
Al eliminarlos se corrige el error y como están comentados desde el SandCastle mediante el Namespace summaries, no es necesario comentarlos en el código.

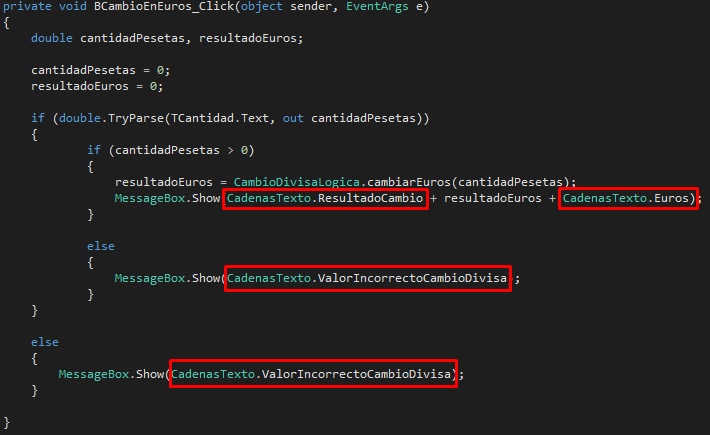
**Solucionados todos los errores aparecidos con las *reglas mínimas administradas por Microsoft.***

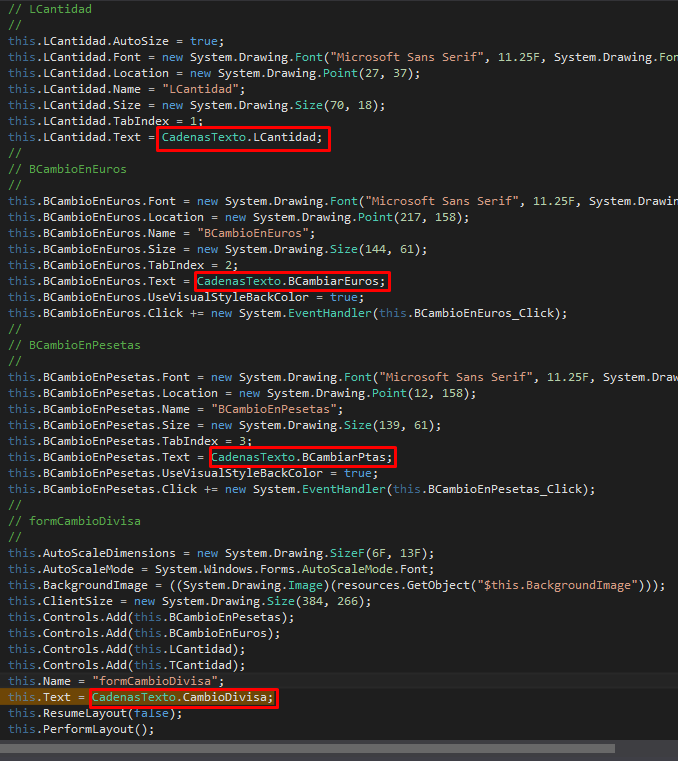
**9.2)** Una vez ejecutado el análisis con las ***reglas de corrección básicas de Microsoft*** me da el siguiente error en diferentes partes del código (36 líneas):



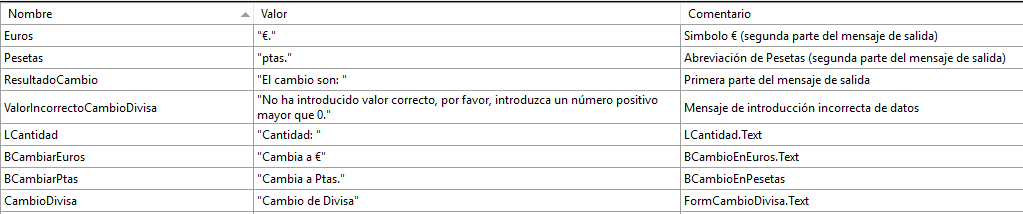
La advertencia CA1303 significa que se deben usar cadenas de texto para facilitar la traducción.

He corregido los de los botones del formulario FormCambioDivisa y los de los “.Text” del FormCambioDivisa.designer del siguiente modo:



Textos del FormCambioDivisa.designer:

Aplicando las Cadenas de Texto expuestas a continuación:



**Solucionados los errores aparecidos con las *reglas mínimas administradas por Microsoft en el formulario FormCambioDivisa y formCambioDivisa.designer.***

Navaja Suiza II

Los proyectos serán entregados dentro del plazo establecido. En caso de entregarse fuera de este, se

considerarán no aptos.

Control de versiones:

2. Todos los cambios serán subidos a Github, incluidos documentos y soluciones. Solo se corregirá lo

subido a GitHub.

Refactorización:

3. Usa los estilos de codificación definidos para cambiar los nombres de clases, controles gráficos y

variables, con el fin de aumentar el nivel de comprensión.

4. Quita los inputbox usados, modificando los formularios para que la información se introduzca mediante

cajas de texto.

a. Limita la longitud de las cajas de texto.

Propiedad TextBox.MaxLength

https://msdn.microsoft.com/es-es/library/system.windows.controls.textbox.maxlength(v=vs.110).a

spx

b. Valida la entrada de datos antes de llamar a los métodos de la lógica de negocio. Si un dato

introducido no cumple las reglas establecidas por los requisitos de la aplicación, se rechaza la

acción, indicando al usuario dónde está el error y cómo resolverlo.

Int32.TryParse Method

https://msdn.microsoft.com/en-us/library/f02979c7(v=vs.110).aspx

Double.TryParse Method

https://msdn.microsoft.com/en-us/library/system.double.tryparse(v=vs.110).aspx

String.IsNullOrEmpty Method (String)

https://msdn.microsoft.com/en-us/library/system.string.isnullorempty(v=vs.110).aspx

String.Length Property

https://msdn.microsoft.com/library/system.string.length(v=vs.110).aspx

Int32.MaxValue Field

https://msdn.microsoft.com/en-us/library/system.int32.maxvalue(v=vs.110).aspx

Int32.MinValue Field

https://msdn.microsoft.com/en-us/library/system.int32.minvalue(v=vs.110).aspx

Igual en double, float, datetime, etc..

5. Crea clases con la lógica de negocio de cada subsistema.

a. Las clases estarán en el mismo namespace del subsistema.

b. El nombre de la clase incluirá la palabra Logica.

c. Los métodos serán static en la mitad de los casos. La otra mitad será necesario instanciar la

clase para que se puede acceder a sus métodos.

static (Referencia de C#)

https://msdn.microsoft.com/es-es/library/98f28cdx.aspx

d. Usa excepciones para comunicar errores de la lógica al interfaz y try/catch para recuperarlos en

los formularios.

Excepciones y control de excepciones (Guía de programación de C#)

https://msdn.microsoft.com/es-es/library/ms173160.aspx

e. Parámetros tipo out o salidas de métodos para devolver resultados.

out (Referencia de C#)

https://msdn.microsoft.com/es-es/library/t3c3bfhx.aspx

f. Diseña nuevas excepciones personalizadas para facilitar el control de errores o usa alternativas

como la vista en clase, en la última parte del tema de test unitarios. Extra: +1 punto máximo.

Excepciones en C# - Parte 3: Diseño de Excepciones Personalizadas

https://ortizol.blogspot.com.es/2014/07/excepciones-en-csharp-parte-3-diseno-de-excepciones-p

ersonalizadas.html

Redacción de casos de prueba II:

6. Documentación de las pruebas con sus correspondientes casos de prueba y pruebas de caja

negra(explicación). Debes crear los casos de prueba pruebas sobre la aplicación.

● Debe ser el guión usado para codificar las pruebas.

● Estará incluido dentro de la solución.

Cómo: Agregar elementos existentes a un proyecto

https://msdn.microsoft.com/es-es/library/9f4t9t92(v=vs.100).aspx

Pruebas unitarias:

7. Crea pruebas unitarias en las clases de la lógica de negocio, teniendo en cuenta el documento creado

para pruebas. Relaciona los métodos de prueba con los casos de prueba.

a. Lanza las pruebas y completa las tablas con los errores encontrados.

· Debes indicar si has encontrado un error en la tabla de casos de uso.

b. Resuélvelos y modifica la tabla de casos de pruebas.

8. Lanza las pruebas de nuevo y comprueba que todo está correcto.

a. Haz una copia de la tabla indicando si te ha funcionado al volver a ejecutarlo.

Extras +2 puntos:

Refactorización:

9. Prueba las funcionalidades de análisis de código de Visual Studio para mejorar el código.

a. Ve este vídeo y aplícalo en tu proyecto:

Análisis de Código en Visual Studio 2010.mp4

https://www.youtube.com/watch?v=D1MLuWEna5o

Tutorial: Analizar código administrado en previsión de defectos de código

https://msdn.microsoft.com/es-es/library/ms182066.aspx

b. Ejecuta el análisis de código de forma manual. Si no encuentras advertencias, cambia el

conjunto de reglas.

c. Resuelve, al menos, dos de las advertencias diferentes de las que te indica, documentando

cómo lo has hecho.

Contenidos relacionados para ampliación (Visual Studio .net Versiones de prueba):

Probar la interfaz de usuario con pruebas de IU automatizadas

https://msdn.microsoft.com/es-es/library/dd286726(v=vs.100).aspx

Usar cobertura de código para determinar la cantidad de código que se está probando

https://msdn.microsoft.com/es-es/library/dd537628.aspx

Generate unit tests for your code with IntelliTest

https://msdn.microsoft.com/en-us/library/dn823749.aspx