Continuando con la prueba que has realizado en la tarea anterior, el uso de "File.Exists" nos permite saber si el fichero existe, pero ese no es el único problema que podemos tener al acceder a un fichero. Puede ocurrir, por ejemplo, que el archivo exista pero no tengamos permiso para acceder a él, o que intentemos escribir en un dispositivo que sea sólo de lectura, …

Una forma más eficaz de comprobar si ha existido algún tipo de error es utilizar el "**manejo de excepciones**".

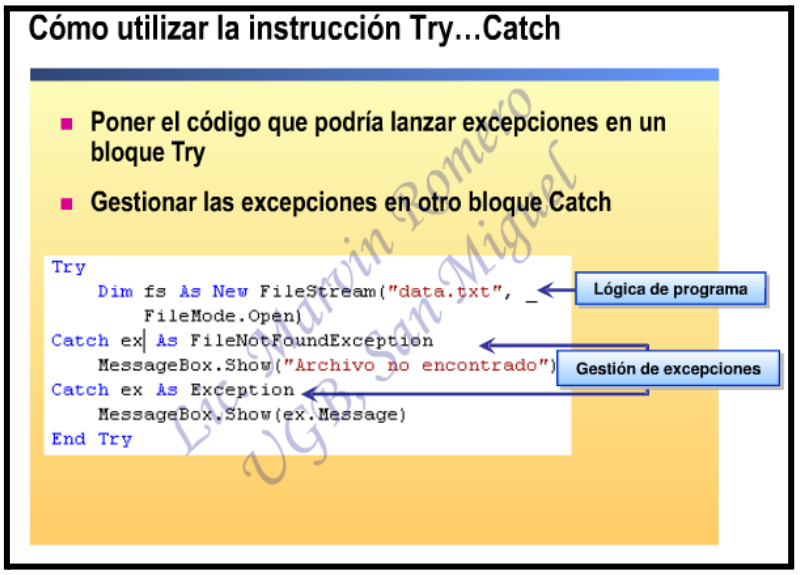
La generación de una excepción permite interrumpir el flujo normal de ejecución de la aplicación, proporcionando información acerca del error que representa. Sin intervención alguna por parte del desarrollador para gestionarla, se propaga automáticamente a través de la pila de llamadas hasta el punto de entrada de la aplicación e incluso puede ocasionar su cierre.

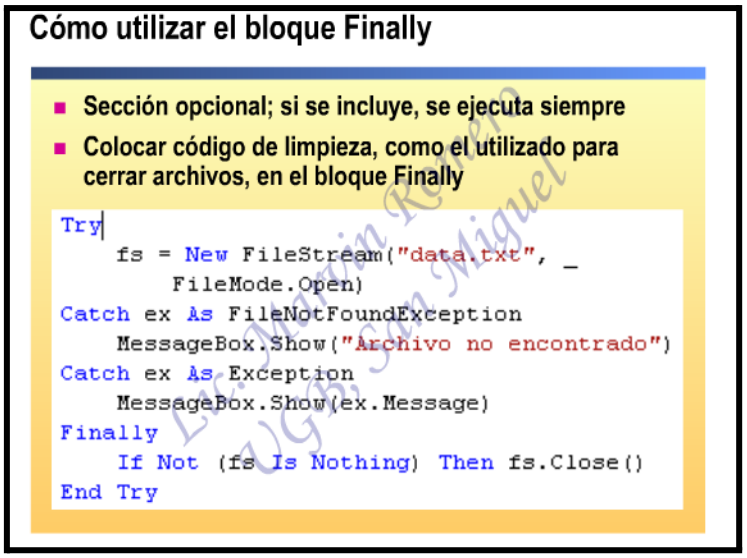
La idea es la siguiente: "intentar" dar una serie de pasos en nuestro código y al final de todos ellos, indicar qué pasos hay que dar en caso de que alguno de ellos no se consiga completar. Esto permite que el programa sea más legible que la alternativa convencional, que consistía en intentar un paso y comprobar errores; si todo era correcto intentar otro paso y volver a comprobar errores; si todo seguía siendo correcto, intentar otro nuevo paso, y así sucesivamente...

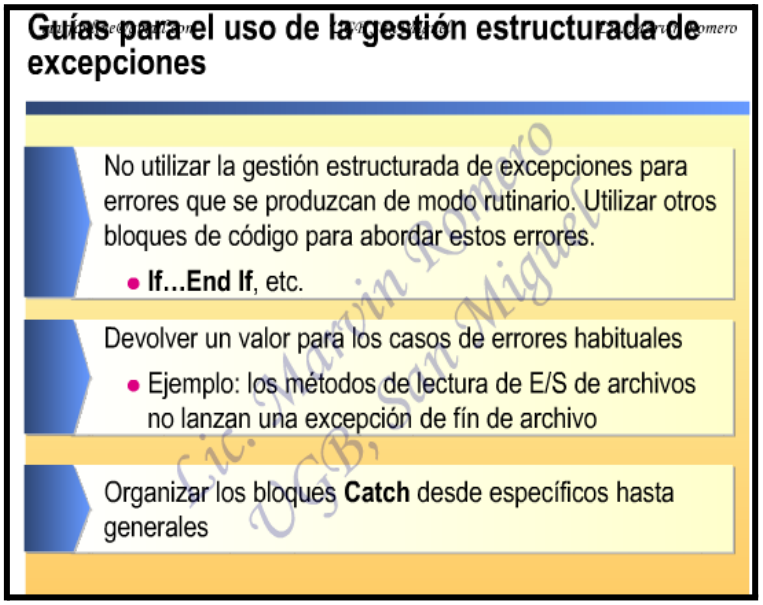
La forma de conseguirlo es dividir en **dos bloques** las partes del código que puedan dar lugar a error:

- en un primer bloque ***(Try)***, indicaremos los pasos que queremos "intentar"
- a continuación, detallaremos las posibles excepciones que queremos "interceptar" ***(Catch)***, y lo que se debe hacer en ese caso

Enlaces a consultar:
Instrucción Try...Catch...Finally (Visual Basic)
Control de errores de E/S en .NET



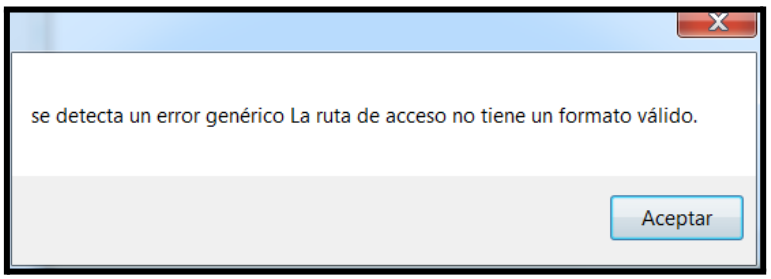




Puesta en práctica:

Tarea4a (un único bloque *Catch ex As Exception*):

Aplicación que muestre el contenido de un archivo de texto cuyo nombre y ruta teclea el usuario. Debes capturar y tratar un posible error genérico (***Exception)*** mostrando un mensaje indicativo. Es decir, si el usuario deja en blanco el nombre del archivo o indica un nombre de archivo erróneo el resultado que hemos de ver en pantalla será algo parecido a la imagen.



```
Try
                ….
        Catch ex As Exception
            MessageBox.Show("se detecta un error genérico " & ex.Message)
        End Try
```

Por supuesto, el código anterior puede generar errores por muchas y diversas razones, tal y como habrás leído aquí. Por ejemplo, se puede generar una excepción si:

- el archivo no se encuentra,
- la ruta no existe,
- la unidad en la que se ubica el archivo no está lista (tal vez se ha solicitado la lectura de un archivo en una unidad de DVD sin medio...),
- no dispone de permiso para tener acceso al archivo o a la carpeta solicitada,
- se ha especificado un nombre de archivo no válido,
- ...

Esta lista podría seguir de forma indefinida. Pues bien, lo razonable no es interceptar todas las excepciones a la vez, sino crear un análisis para cada caso que permita recuperarse del error y seguir adelante, para lo que se suelen crear **varios bloques Catch**. Así, dentro de cada bloque Catch podríamos indicar una excepción más concreta, de forma que el mensaje de aviso sea más concreto y dar pasos más adecuados para solucionar el problema.

.NET Framework ofrece un gran número de clases de excepciones específicas. Todas estas clases heredan de **la clase base Exception**. En la documentación de .NET se incluye una serie de tablas en las que encontrarás todas las excepciones que se pueden generar al llamar a un método determinado.

Tarea4b: mejora la aplicación anterior incluyendo en la misma **más de un bloque Catch,** de manera que tu código obtenga un mensaje más explícito que haga saber al usuario cuál ha sido el problema. Utiliza varias excepciones específicas. Por ejemplo, realiza pruebas de ejecución cambiando el nombre del archivo de modo que se encuentre:

- en una ruta válida pero selecciona un archivo que no existe,
- en una unidad que no existe,
- en una ruta que no existe,
- en una unidad que no está lista,
- …

Estas son algunas de las clases que puedes probar:

- FileNotFoundException
- DirectoryNotFoundException
- ArgumentException
- UnauthorizedAccessException
- IOException
- Exception
- …

Sube a Moodle el proyecto comprimido con la tarea propuesta.