Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Programación Genérica y Eventos: Informe – Tarea 8 – Parte 2**

* **Asignatura:** Programación Genérica y Eventos.
* **Profesora:** Mónica Nano.
* **Alumnos:** Tomas Molina y Edgar Karpowicz.
* **Tema:** Tarea 8 – Informe – Parte 2.
* **Fecha:** 20/10/24

**Informe: Tarea 8 – Parte 2**

* **Manejo de Excepciones Básico:**
* **Introducción:**

En el Presente Informe, para la Materia de Programación Genérica y Eventos de la Carrera Ingeniería Informática en la Universidad Blas Pascal, se desarrollará sobre las Excepciones y su manejo en el Lenguaje de Programación C#. Esto con el objetivo de profundizar y aprender sobre este Lenguaje y sus Conceptos Claves, los cuales son aplicables para otros Lenguajes con diferentes Sintaxis.

* **Excepciones y su Manejo:**

Antes de comenzar a enunciar los tipos de Excepciones, como es su manejo, la mejor forma de utilizarlas, entre otros, es clave entender que son estas.

Las Excepciones en C# son un tipo de Error que ocurre durante la Ejecución de una Aplicación. Típicamente, son problemas que al no ser anticipados en el Código escrito por el Programador, conllevan a errores que deben ser corregidos. Para lidiar con estas sin tener que anticipar cada posible escenario y/o problema, en las Aplicaciones Modernas en C#, se usa o implementa una Lógica de Manejo de Excepciones para manejar explícitamente cuando ocurren las Excepciones y que debe suceder si llegase a ocurrir alguna de estas.

En el caso de C#, este Lenguaje permite implementar un Manejo de Excepciones mediante las Palabras Claves: try, catch, finally, y throw.

* **Try:** bloque que engloba un pedazo de código. El código dentro se intenta ejecutar y si genera una excepción el catch que le sigue lidia con la misma.
* **Catch:** cuando una excepción ocurre, el Código dentro del Bloque catch se ejecuta. Aquí es donde uno puede manejar la Excepción. Así mismo, en este Bloque definimos que tipo de Excepción esperamos que ocurra.
* **Finally:** el Bloque finally permite la ejecución de un Código especifico sin importar la Ejecución. Por ejemplo, sirve para desechar un objeto que requiera ser desechado.
* **Throw:** la Palabra Clave throw crea una Excepción nueva. “Tiramos” o hacemos Throw a una Excepción.
* **Tipos de Excepciones:**

Al momento de trabajar con las Excepciones, más allá de lo básico anteriormente enunciado, uno debe tener en cuenta que hay una gran cantidad de estas predefinidas en C#, de las cuales, deberemos elegir cual de estas pueden llegar a ocurrir en nuestro Código, y utilizarlas en el Bloque catch. Entre los Tipos de Excepciones que podemos mencionar se encuentran:

* **Exception:** excepción genérica.
* **AccessViolationException:** se tira cuando se intenta leer o escribir en memoria protegida.
* **FileNotFoundException:** se da cuando se intenta acceder a un archivo que no existe en el disco.
* **IOException:** se tira cuando un error I/O ocurre. System.IO permite leer y escribir en Archivos.
* **UnauthorizedAccessException:** se da cuando el Sistema Operativo niega el acceso por un Error I/O, o un error especifico de seguridad.
* **EndOfStreamException:** se tira cuando hay un intento de leer más allá de cuando termina el Stream y/o Texto.
* **FileFormatException:** cuando el formato del Archivo ingresado que se supone que se conforme a un tipo de especifico se encuentra malformado.
* **DirectoryNotFoundException:** cuando el directorio o parte del archivo no pueden ser encontrados.
* **ReadOnlyException:** se tira si se intenta cambiar el valor de una columna en estado Read-Only.
* **OperationAbortedException:** se da si una Operación en proceso es abortada por el Usuario.
* **TimeoutException:** cuando la cantidad de tiempo dado para un proceso y/u Operación ha expirado.
* **StackOverflowException:** se tira cuando el stack de ejecución tiene demasiadas llamadas a métodos en formato nested.
* **OverflowException:** cuando una operación ya sea aritmética, de conversión, entre otros, resulta en un overflow.
* **Entre Otras.**

**Para ver más Excepciones visitar:**

[**https://www.completecsharptutorial.com/basic/complete-system-exception.php**](https://www.completecsharptutorial.com/basic/complete-system-exception.php)

* **¿Cuándo Capturar Excepciones Generales y Especificas?**

Al momento de elegir cuales Tipos de Excepciones debemos implementar en nuestro Código en términos de Captura, uno se puede encontrar en el debate de cuando es apropiado capturar excepciones especificas versus excepciones generales.

En la mayoría de los casos, es mejor que intentemos predecir de la manera más especifica posible cuales Excepciones tienen altas chances de ocurrir en la ejecución de nuestro Código, para así promover una Implementación Robusta del Programa. Dado que si conocemos la Excepción que esta ocurriendo en ese momento podemos planear una forma de recuperar / resolver el Error que se esta dando en ese momento. Algo que no puede suceder si se están utilizando Excepciones Generales, dado que las mismas no nos permitirán resolver la Situación que se este dando.

Un ejemplo de esto sería si se solicita la dirección de un Archivo, y el Usuario da una que no existe. Con Excepciones Especificas, es decir, FileNotFoundException, uno puede planear sobre como logear este Error e intentar Resolverlo, algo que no puede hacerse con las Excepciones Generales.

* **Bibliografía:**
* <https://stackify.com/csharp-exception-handling-best-practices/>
* <https://www.w3schools.com/cs/cs_exceptions.php>
* <https://www.completecsharptutorial.com/basic/complete-system-exception.php>
* <https://learn.microsoft.com/en-us/dotnet/api/system.io?view=net-8.0>