**EXCEPCIONES**

En el trascurso de la ejecución de un programa pueden ocurrir errores, los cuales pueden provocar la terminación del programa o el bloqueo del mismo. Para evitar esto se han definido las Excepciones, que tratan los errores de ejecución para que al ocurrir, pase a ejecutarse un determinado conjunto de instrucciones que la resuelvan.

La técnica es poner las instrucciones que sabemos que pueden dar un error en una cláusula específica llamada “try”. O sea intenta ejecutar esas instrucciones, y si dan error ejecutamos otras.

Es un modo estructurado para el manejo de situaciones de error.

Las excepciones son instancias de tipos de clase derivador de la clase System.Exception

try

{

Intrucciones que pueden dar el error

}

catch( Exception e)

{

Instrucciones que solucionan el error

}

**Clase System. Exception**

Propiedades

* Message. Descripción del error
* InnerException. Código interno de la excepción.

Lanzamiento de excepciones

* Debido a que ocurra un error y la ejecute
* El programador puede invocar a la excepción con una instrucción throw

try

{

Sentencias que pueden dar error

}

catch(Excepción más específica e)

{

Tratamiento de la excepción más específica

}

catch(Excepción siguiente específica e)

{

Tratamiento de la excepción siguiente específica

}

……………….

catch(Excepción más general e)

{

Tratamiento de la excepción más general

}

finally

{

Intrucciones que siempre se ejecutan

}

“e” es una variable de tipo Exception, que apuntará al objeto de tipo Exception que crea con la información de la excepción ocurrida.

“finally” es opcional, y si se pone se ejecuta siempre.

Puede haber 1, 2 ó más “catch” como tipos de excepciones se quieran capturar.

Orden de ejecución

1. Catch que case
2. Finally
3. Si no casa ninguno o no hay, ejecutamos el finally. Después vamos al método que llamó a este y comprobamos allí. Así sucesivamente hacia atrás. Si no hay ningún tratamiento de excepciones, el programa termina.

Excepciones más usuales

Excepción Causa

|  |  |
| --- | --- |
| System.OutOfMemoryException | Se lanza cuando falla un intento de reservar memoria mediante el operador new. |
| System.NullReferenceException | Se lanza cuando se pretende acceder mediante una referencia null a un supuesto objeto. |
| System.ArrayTypeMismatchException | Se lanza cuando se intenta almacenar un elemento de un tipo no adecuado en un array. |
| System.IndexOutOfRangeException | Se lanza cuando se intenta acceder a un elemento de un array que no existe. Si índice es menor a 0 o mayor que el número de elementos del array |
| System.ArithmeticException | Se da con errores matemáticos |
| System.DivideByZeroException | Cuando se divide por cero |
| System.OverflowException | Cuando hay overflow, o sea el resultado no entra en el tipo de dato |
| System.ArgumentNullException | Argumento nulo en la llamada a una función. |
| System.Exception | Excepción genérica |

using System;

public class pruebamedia

{

int suma=5;

int cuenta=0;

float media;

public void hazmedia()

{

try

{

media = suma / cuenta;

Console.WriteLine(“Media: {0}, media); // No se ejecuta nunca

}

catch(DivideByZeroException ex)

{

Console.WiteLine(“ Cuenta es cero, ponemos resultado a cero”);

media = 0;

}

catch(Exception ex)

{

}

}

}

public class test

{

public static void Main()

{

pruebamedia objetomedia = new pruebamedia();

objetomedia.hazmedia();

Console.WriteLine(“Media: {0} “, objetomedia.media); // Visualiza cero

}

}