# **Universidad Cenfotec**



# **Investigación Técnica.**

Excepciones en POO (C#)

# **Profesor:**

# **Dennis Cordova**

# **Estudiantes:**

**José Alejandro Gómez Castro**

**José Andrés Ramírez Ocampo**

# **Fecha de entrega: 02/06/2016**

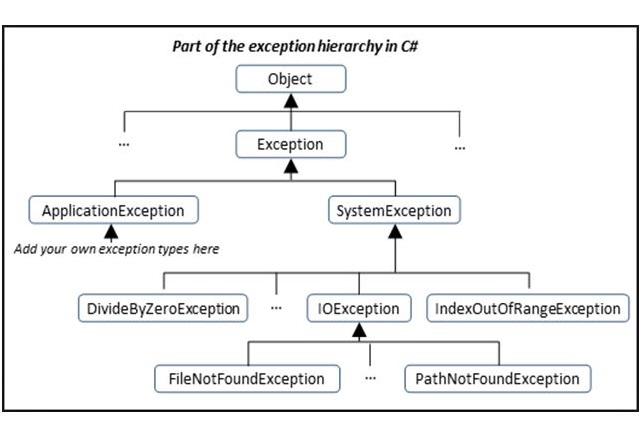
# **Segundo cuatrimestre 2016**

¿Qué es una excepción?

Las excepciones son un mecanismo para controlar el flujo de un programa. Con ellas se pueden realizar acciones especiales si eventualmente se da un error previsto en el sistema.

Mediante la gestión de excepciones se podría evitar repetir continuamente el código, en búsqueda de un posible error, con el objetivo de permitir que los métodos respondan frente a circunstancias inesperadas. Muchos lenguajes de programación proporcionan una forma alternativa de finalizar la ejecución de un método. Esto se logra mediante la sentencia throw. Cuando un método ejecuta una sentencia throw se dice que se lanza una excepción.

Jerarquía de clases de las excepciones



La sintaxis general es la siguiente:

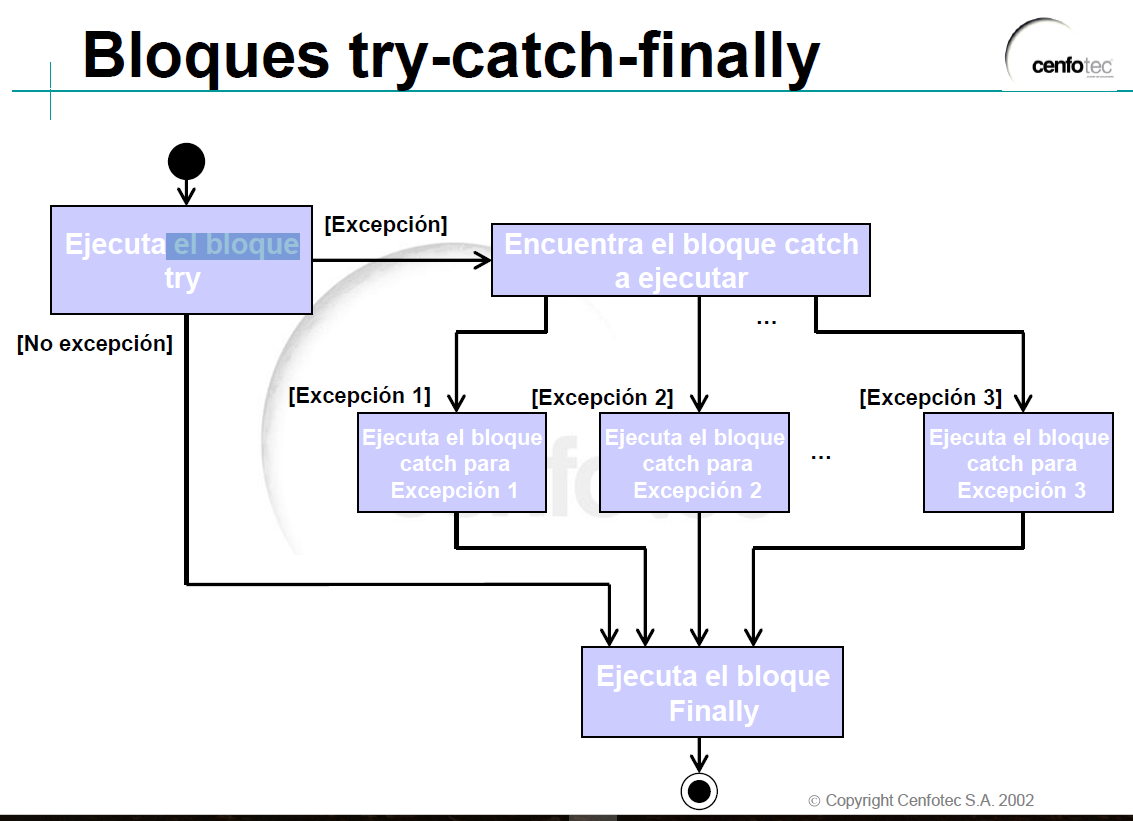
|  |
| --- |
| **RAISE\_APPLICATION\_ERROR**(<error\_num>,<mensaje>); |

Siendo:

* error\_num es un entero negativo comprendido entre -20001 y -20999
* mensaje la descripción del error

|  |
| --- |
| **DECLARE**  v\_div **NUMBER**;  **BEGIN**  **SELECT** 1/0 **INTO** v\_div **FROM** DUAL;  **EXCEPTION**  **WHEN** **OTHERS** **THEN**  **RAISE\_APPLICATION\_ERROR**(-20001,'No se puede dividir por cero');  **END**; |

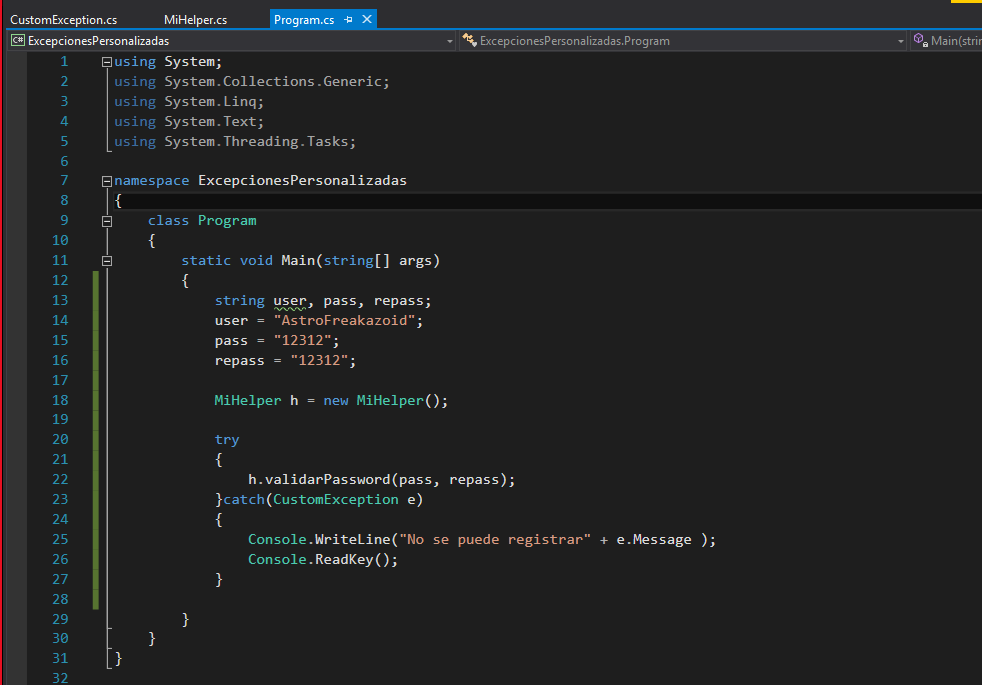
Diagrama de actividad de la ejecución del bloque try-catch-finally.



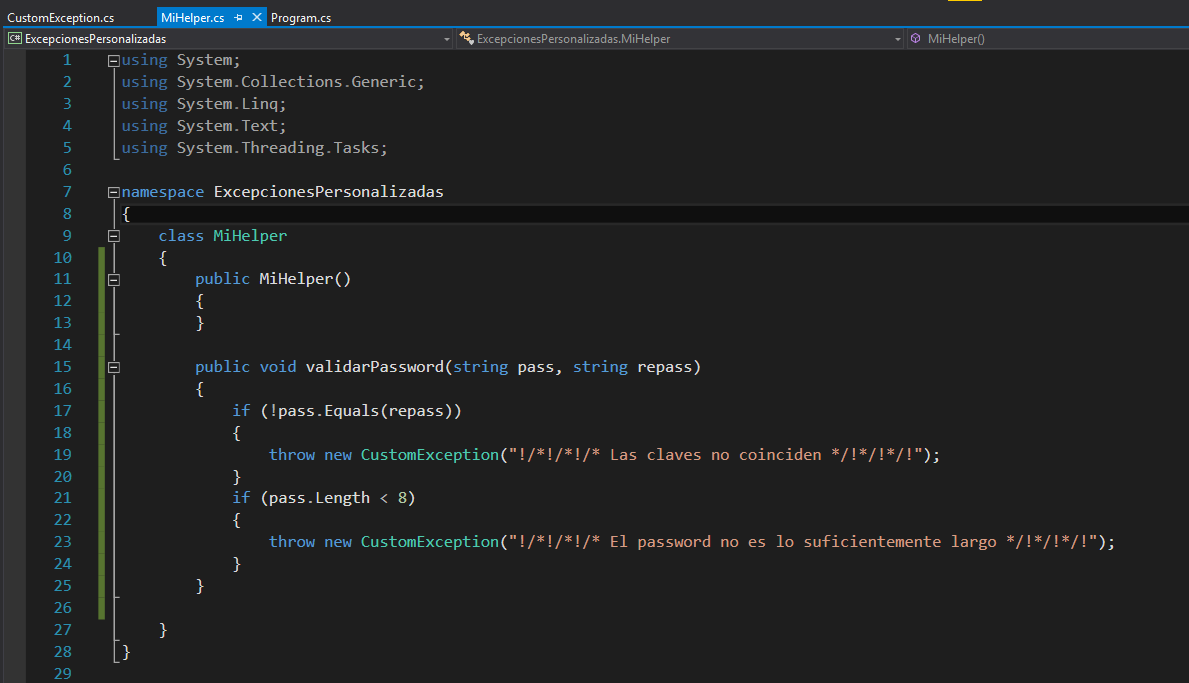
Excepciones personalizadas

La excepciones personalizadas son aquellas diseñadas por el desarrollador estas excepciones deben heredar de la clase padre “Exception”.

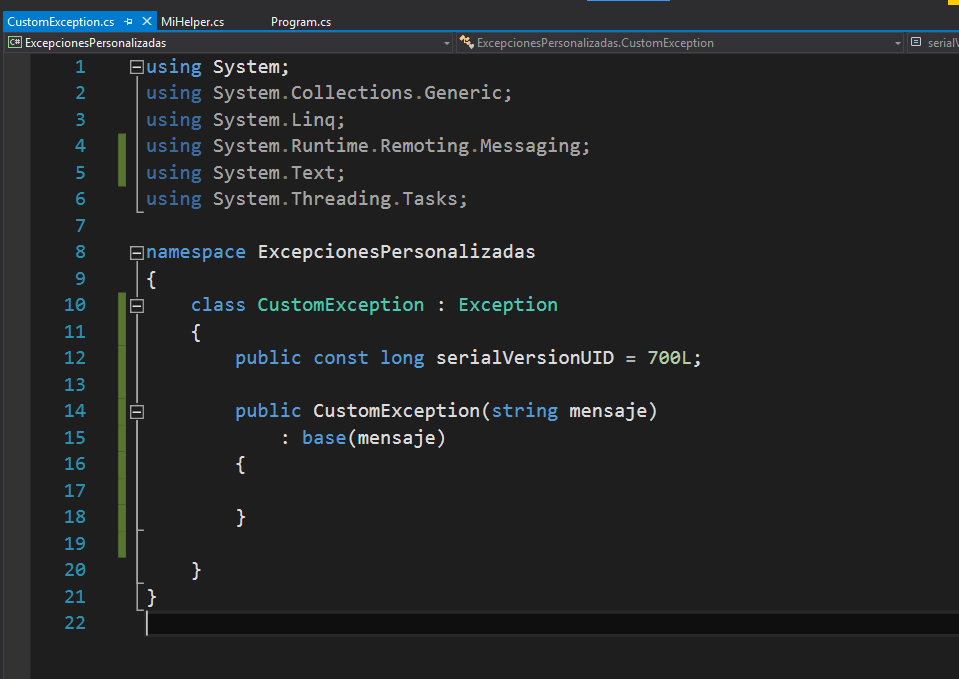
A continuación se muestra un ejemplo de una excepción personalizada para validar que los campos de contraseña sean iguales y lo suficientemente largos.



Img 01. En esta imagen se muestra el Main que contiene el bloque try-catch, además cuenta con un objeto de tipo MiHelper que se encarga de hacer el throw de la excepción



Img 02. Esta imágen muestra el código de la clase MiHelper la cual cuenta con el constructor vacío además de el método valiadarPassword() que recibe dos cadenas de texto (la contraseña y la confirmación de la misma). Dentro del método hay un if que valida que la contraseña y su confirmación sean iguales, también que sea mayor o igual a 8 caracteres.



Img 03. Aquí se muestra la excepción personalizada que hereda de Exception y tiene un método constructor que recibe un mensaje de tipo string.

Métodos para el control de excepciones

C# y muchos otros lenguajes de programación cuentan con la estructura "try – catch – finally" para el manejo adecuado de las excepciones.

try

{

// Instrucciones cuando no hay una excepción

} catch (TypeException ex)

{

// Instrucciones cuando se produce una excepcion

} finally

{

// Instruciones que se ejecutan, tanto si hay como sino hay excepciones

}

**Bloque try**

Todo el código que vaya dentro de esta sentencia será el código sobre el que se intentará capturar el error si se produce y una vez capturado hacer algo con él. Lo ideal es que no ocurra un error, pero en caso de que ocurra un bloque try nos permite estar preparados para capturarlo y tratarlo.

**Bloque catch**

En este bloque definimos el conjunto de instrucciones necesarias o de tratamiento del problema capturado con el bloque try anterior. Es decir, cuando se produce un error o excepción en el código que se encuentra dentro de un bloque try, pasamos directamente a ejecutar el conjunto de sentencias que tengamos en el bloque catch.

**Bloque finally**

El bloque finally que es un bloque donde podremos definir un conjunto de instrucciones necesarias tanto si se produce error o excepción como si no y que por tanto, se ejecuta siempre.

**Ejemplo**

El siguiente ejemplo muestra el manejo de la excepción que puede ocurrir en una división al dividir cualquier número entre 0.

A continuación se muestra el siguiente código donde se tiene un método SafeDivision() que recibe como parámetros dos variables de tipo double, dentro del método tenemos una estructura condicional simple que evalúa si el divisor es igual a 0; si el resultado es cierto lanza la excepción esto significa que crea una instancia de la clase DivideByZeroException.

La excepción es atrapada en el catch que se encuentra en el Main, se imprime en pantalla el mensaje: "Attempted divide by zero." y finaliza la ejecución del programa.

|  |
| --- |
| class ExceptionTest {  static double SafeDivision(double x, double y)  {  if (y == 0)  throw new System.DivideByZeroException();  return x / y;  }  static void Main()  {  // Input for test purposes. Change the values to see  // exception handling behavior.  double a = 98, b = 0;  double result = 0;   try  {  result = SafeDivision(a, b);  Console.WriteLine("{0} divided by {1} = {2}", a, b, result);  }  catch (DivideByZeroException e)  {  Console.WriteLine("Attempted divide by zero.");  }  } } |

# Bibliografía

* Universidad Cenfotec (2012). Tema08-1 Excepciones.
* <http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=678:gestion-de-excepciones-en-java-captura-con-bloques-try-catch-finally-ejemplos-resueltos-sencillos-cu00927c&catid=58:curso-lenguaje-programacion-java-nivel-avanzado-i&Itemid=180>
* <https://www.youtube.com/watch?v=LvkxCgs5Lhw>
* <https://msdn.microsoft.com/es-es/library/ms173160.aspx>
* http://jarroba.com/excepciones-exception-en-java-con-ejemplos/