C# Exception handling

HO
GENT

Exception: wat?

- Omschrijving
  - Een “onregelmatigheid” bij het normale verloop
  - Vereist een gepaste reactie
  - Is geen “event”!
- Methods geven een fout aan door een uitzondering “op te werpen” (*raise*)
- .NET runtime implementeert exception handling
  - Een exception
    - is een object van *class Exception*
    - Wordt opgeworpen in een specifieke thread
    - Passeert door de stack van de thread tot deze afgehandeld wordt … of de thread stopt (=> in main thread: programma stopt)
    - Blijft niet beperkt tot grens *process* of machine
    - Passeert door *managed* en *unmanaged code*

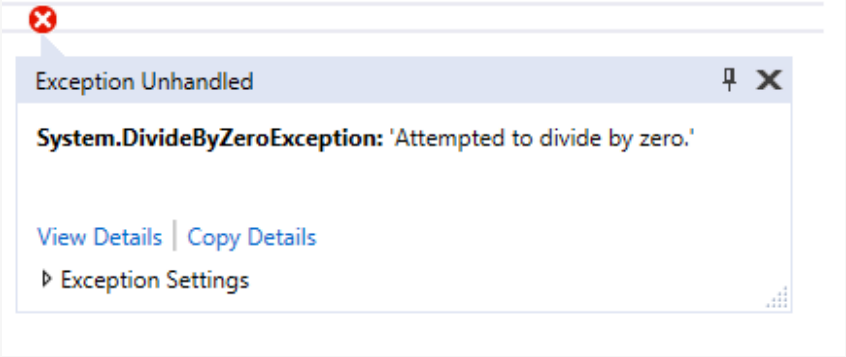
HO GENT

Exception: geen handler

- Call stack
- Geen handler

```
class ExceptionhandlingExample
{
            public int Div(int number, int divisor)
            {
                        //thrown an exception if divisor is 0
                        return number / divisor;
            }
}
```

```
private static void ExceptionExample()
{
            WriteLine("Exception handling example.");
            ExceptionhandlingExample example = new ExceptionhandlingExample();
            Write("Enter numerator:");
            var num = ReadLine();
            Write("Enter denumerator:");
            var denumerator = ReadLine();
            var quotient = example.Div(Convert.ToInt32(num), Convert.ToInt32(denumerator));
            WriteLine($"Quotient of numerator:{num}, denumerator:{denumerator} is {quotient}");
}
```



Exception: handler, geen filter

```
public int Div(int nominator, int denominator)
{
            int quotient = 0;
            try
            {
                        quotient = nominator / denominator;
            }
            catch
            {
                        Console.WriteLine($"Exception occured");
            }
            return quotient;
}
```

HO GENT

Exception: handler, filter

```
public int Div(int nominator, int denominator)
{
		int quotient = 0;
		try
		{
				quotient = dividend / divisor;
		}
		catch (Exception exception)
		{
				Console.WriteLine($"Exception occered '{exception.Message}'");
		}
		return quotient;
}
```

HO GENT

Multiple catch blocks

```
using System;
class Mn
{
        static void Main()
        {
                int[] number = { 8, 17, 24, 5, 25 };
                int[] divisor = { 2, 0, 0, 5 };

                for (int j = 0; j < number.Length; j++)
                        try
                        {
                                Console.WriteLine("Number: " + number[j]);
                                Console.WriteLine("Divisor: " + divisor[j]);
                                Console.WriteLine("Quotient: " + number[j] / divisor[j]);
                        }
                        catch (DivideByZeroException)
                        {
                                Console.WriteLine("Not possible to divide by zero");
                        }
                        catch (IndexOutOfRangeException)
                        {
                                Console.WriteLine("Index is out of range");
                        }
        }
}
```

HO GENT

Multiple catch blocks: respecteer hierarchie

A previous catch clause already catches all exceptions of this or a super type ('System.Exception')

```
using System;
class Test
{
	static void Main()
	{
		try
		{
			int a=10,b=0,c=0;
			c=a/b ;
			Console.WriteLine(c);
		}
		catch(System.Exception e)
		{
			Console.WriteLine(e.Message);
		}
		catch(System.DivideByZeroException ex)
		{
			Console.WriteLine(ex.Message);
		}
	}
}
```

HO GENT

Exception: finally

- Wordt altijd uitgevoerd
- Geef resources vrij, bijvoorbeeld *file handle*
- Niet toegelaten: *return, break, goto, continue, exit, try*

```
public int Div(int nominator, int denominator)
{
		int quotient = 0;
		try
		{
			quotient = nominator / denominator;
		}
		catch
		{
			Console.WriteLine($"Exception occured");
		}
		finally
		{
			Console.WriteLine("Exception occured and cleaned.");
		}
		return quotient;
}
```

HO GENT

Exception class

- Base class voor alle *exception* classes
- Properties
  - *StackTrace*: identificeert plaats in code
  - *InnerException*: gebruikt om een serie van exceptions te linken aan elkaar
  - *Message*: details, “*localized*”
- Methods
  - *GetBaseException*(): geeft originele binnenste exception terug

HO
GENT

User-defined exception

- Afleiden van class Exception
  - Tip: naam eindigt op “Exception”

- 3 constructoren voorzien:
  - public MyException() {}
  - public MyException(string message) : base (message) {}
  - public MyException(string message, Exception inner) : base (message, inner) {}

HO GENT

Throw user defined exception (1/4)

```
internal class NumberIsExceededException : Exception
{
    public NumberIsExceededException(string message) : base(message)
    {
    }

    public NumberIsExceededException(string message, Exception innerException):base(message,innerException)
    {
    }

    protected NumberIsExceededException(SerializationInfo serializationInfo, StreamingContext streamingContext) : base(serializationInfo, streamingContext)
    {}
}
```

HO GENT

Throw user defined exception (2/4)

```
internal class StringCalculatorUpdated
{
    public int Add(string numbers)
    {
        var result = 0;
        try
        {
            return IsNullOrEmpty(numbers) ? result : AddStringNumbers(numbers);
        }
        catch (NumberIsExceededException numberIsExceededException)
        {
            Console.WriteLine($"Exception occurred:'{numberIsExceededException.Message}'");
        }

        return result;
    }
        // … StringToInt32()
}
```

HO GENT

Throw user defined exception (3/4)

```
    private int StringToInt32(string n)
    {
        var number = Convert.ToInt32(string.IsNullOrEmpty(n) ? "0" : n);
        if(number>1000)
            throw new NumberIsExceededException($"Number :{number} exceeds the limit of
1000.");
        return number;
    }
```

HO GENT

Throw user defined exception (4/4)

```
private static void CallStringCalculatorUpdated()
{
    WriteLine("o Add operation can take 0, 1, or 2 comma - separated numbers, and will return their sum
        for example \"1\" or \"1, 2\"\n" +
        "o Add operation should accept empty string but for an empty string it will return 0.\n"
        "o Throw an exception if number > 1000\n");
        StringCalculatorUpdated calculator = new StringCalculatorUpdated();
    Write("Enter numbers comma separated:");
    var num = ReadLine();
    Write($"Sum of {num} is {calculator.Add(num)}");
}
```

HO
GENT

Rethrow (1/2)

Als een exception opnieuw wordt opgeworpen door het exception object op te werpen met throw, wordt de stack trace herstart en gaat de stack trace tot op die plaats verloren.

```
void CatchAndRethrowExplicitly()
{
  try
  {
    ThrowException();
  }
  catch(ArithmeticException e)
  {
    // Violates the rule.
    throw e;
  }
}

void CatchAndRethrowImplicitly()
{
  try
  {
    ThrowException();
  }
  catch(ArithmeticException e)
  {
    // Satisfies the rule.
    throw;
  }
}
```

Rethrow (2/2)

```
using System;
namespace UsageLibrary
{
  class TestsRethrow
  {
    static void Main()
    {
      TestsRethrow testRethrow = new TestsRethrow();
      testRethrow.CatchException();
    }
    void CatchException()
    {
      try
      {
        CatchAndRethrowExplicitly();
      }
      catch(ArithmeticException e)
      {
        Console.WriteLine("Explicitly specified:{0}{1}", Environment.NewLine, e.StackTrace);
      }
      try
      {
        CatchAndRethrowImplicitly();
      }
      catch(ArithmeticException e)
      {
        Console.WriteLine("{0}Implicitly specified:{0}{1}", Environment.NewLine, e.StackTrace);
      }
    }
    void ThrowException()
    {
      throw new ArithmeticException(“Illegal expression");
    }
  }
}
```

HO GENT

Demo debugging exceptions

- Verander de manier waarop de debugger bepaalde exceptions behandelt

HO
GENT

Exception of return code?

- Return code: wanneer je de uitkomst kent en verwacht
- Exception: wanneer onverwachte uitzonderingen optreden
- Return code: nogal arbitrair en afhandeling verspreid over code ;-(

HO
GENT

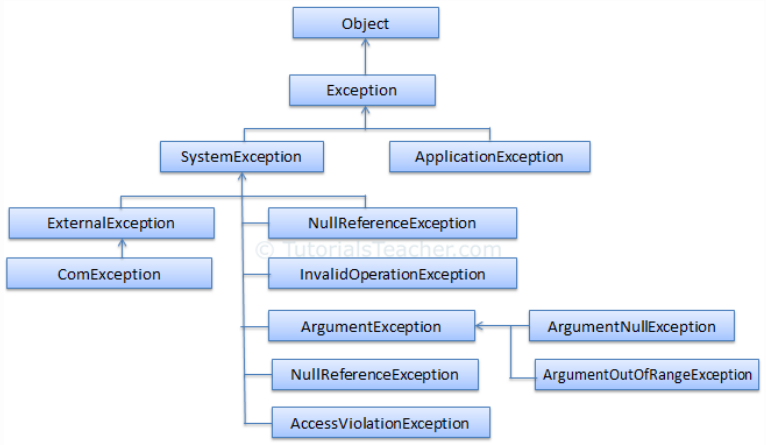
Exception: wanneer opvangen?

- Zo weinig mogelijk
- Zo hoog mogelijk

HO
GENT

Compiler-generated exceptions

- https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/exceptions/compiler-generated-exceptions
- Exception hierarchy
  - SystemException
    - ArithmeticException
    - StackOverflowException
    - NullReferenceException
    - IndexOutOfRangeException
    - InvalidCastException
    - ExternalException
      - Win32Exception
  - IOException
  - YourOwnException



HO GENT