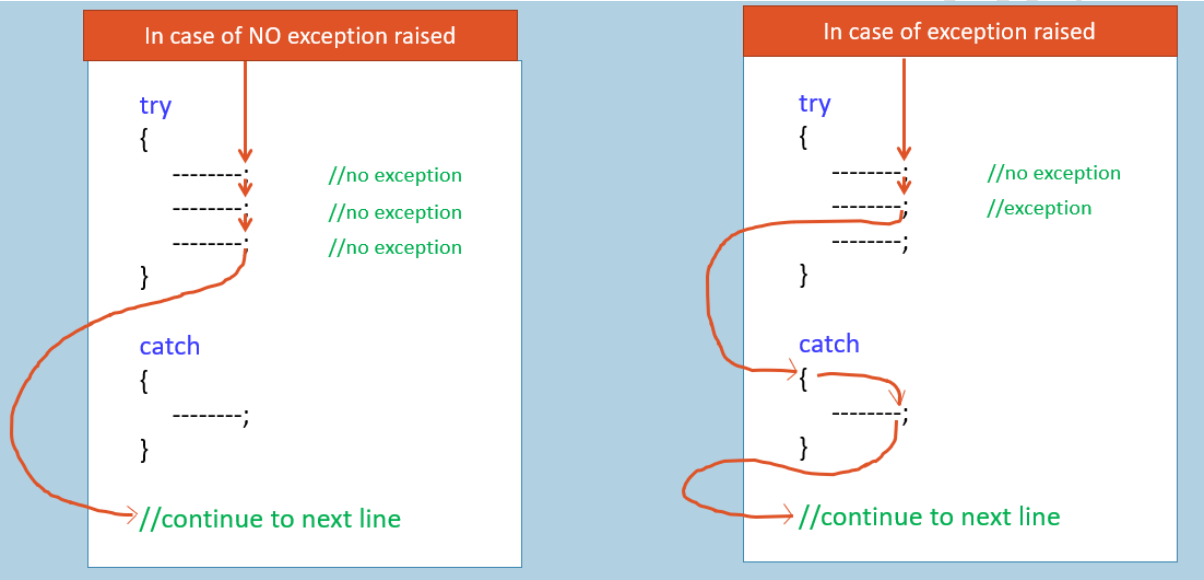
C# - Ultimate Guide - Beginner to Advanced | Master class

Section 28 – Exception Handling

Exception Handling

- Exception is a run time error occurs while executing the application.
- When exception occurs, the current application terminates abruptly.
- Exception Handling avoids abrupt termination of the application, in case of exception.



```
try
{
  statement1;
  statement2;
  statement3;
  …
}
catch (ExceptionClassName variable)
{
  //do something with exception variable
  //or show error message
}
finally
{
  //do some cleaning process
}
```

- When CLR is unable to execute a statement, it is treated as exception.
- ‘try’ and ‘catch’ blocks are mandatory.
- ‘finally’ block and multiple ‘catch’ blocks are optional.
- "try" block contains all the actual code, where exceptions may occurs.
- Multiple "try" blocks for one catch block is not allowed. Nested "try" blocks is allowed
- "catch" block contains error handling code; it executes only when a particular type of exception is raised during the execution of "try" block. Multiple "catch" blocks is allowed.
- "finally" block executes after successful completion of "try" block; or after any catch block. It is optional.
- "throw" keyword is used to throw built-in or custom exceptions, in case of invalid values found.

FormatException

FormatException represents an error when it is unable to convert a string to a number, as the string contains characters other than digits (such as alphabets, spaces etc.).

Occurs when calling methods of 'Convert' class or 'Parse' method of numeric types with incorrect string value.

It's a bad practice to throw FormatException implicitly / explicitly.

throw new FormatException(…)

Constructors of 'FormatException'

1. FormatException()

It initializes nothing.

2. FormatException(string message)

It initializes 'Message' property.

3. FormatException(string message, Exception innerException)

It initializes 'Message' and 'InnerException' properties.

IndexOutOfRangeException

IndexOutOfRangeException represents an error when an index was supplied outside the available range to an array.

Occurs when accessing an array with a wrong index value, which is less than 0 or greater than or equal to size of the array.

It's a bad practice to throw IndexOutOfRangeException implicitly / explicitly.

throw new IndexOutOfRangeException(…)

Constructor of 'IndexOutOfRangeException'

1. IndexOutOfRangeException()

It initializes nothing.

2. IndexOutOfRangeException(string message)

It initializes 'Message' property.

3. IndexOutOfRangeException(string message, Exception innerException)

It initializes 'Message' and 'InnerException' properties.

NullReferenceException

NullReferenceExeption represents an error when you try to access a property, indexer or method through a reference variable when its value is null.

It's a bad practice to throw NullReferenceException implicitly / explicitly.

throw new NullReferenceException(…)

Constructors of 'NullReferenceException'

1. NullReferenceException()

It initializes nothing.

2. NullReferenceException(string message)

It initializes 'Message' property.

3. NullReferenceException(string message, Exception innerException)

It initializes 'Message' and 'InnerException' properties.

ArgumentNullException

ArgumentNullException represents an error when a null value is passed as argument to a method; and that method doesn't accept null's into that parameter.

It's a good practice to throw ArgumentNullException implicitly / explicitly.

throw new ArgumentNullException(…)

Constructors of 'ArgumentNullException'

1. ArgumentNullException()

It initializes nothing.

2. ArgumentNullException(string paramName)

It initializes 'ParamName' property.

3. ArgumentNullException(string message, Exception innerException)

It initializes 'Message' and 'InnerException' properties.

4. ArgumentNullException(string paramName, string message)

It initializes 'ParamName' and 'Message' properties.

ArgumentOutOfRangeException

ArgumentOutOfRangeException represents an error when a numeric value is passed as argument to a method; and it outside the acceptable range of values accepted by that method.

It's a good practice to throw ArgumentOutOfRangeException implicitly / explicitly.

throw new ArgumentOutOfRangeException(…)

Constructors of 'ArgumentOutOfRangeException'

1. ArgumentOutOfRangeException()

It initializes nothing.

2. ArgumentOutOfRangeException(string paramName)

It initializes 'ParamName' property.

3. ArgumentOutOfRangeException(string message, Exception innerException)

It initializes 'Message' and 'InnerException' properties.

4. ArgumentOutOfRangeException(string paramName, string message)

It initializes 'ParamName' and 'Message' properties.

5. ArgumentOutOfRangeException(string paramName, object actualValue, string message)

It initializes 'ParamName', 'ActualValue' and 'Message' properties.

ArgumentException

ArgumentException represents an error when the argument value of a parameter is invalid as per any validation rules / requirements.

It's a good practice to throw ArgumentException implicitly / explicitly.

throw new ArgumentException(…)

Constructors of 'ArgumentException'

1. ArgumentException()

It initializes nothing.

2. ArgumentException(string message)

It initializes 'Message' property.

4. ArgumentException(string message, Exception innerException)

It initializes 'Message' and 'InnerException' properties.

5. ArgumentException(string message, string paramName)

It initializes 'Message' and 'ParamName' properties.

6. ArgumentException(string message, string paramName, string innerException)

It initializes 'Message', 'ParamName' and 'InnerException' properties.

InvalidOperationException

InvalidOperationException represents an error when you call a method; and it is invalid to call it a per current state of the object.

It's a good practice to throw InvalidOperationException implicitly / explicitly.

throw new InvalidOperationException(…)

Constructors of 'InvalidOperationException'

1. InvalidOperationException()

It initializes nothing.

2. InvalidOperationException(string message)

It initializes 'Message' property.

3. InvalidOperationException(string message, Exception innerException)

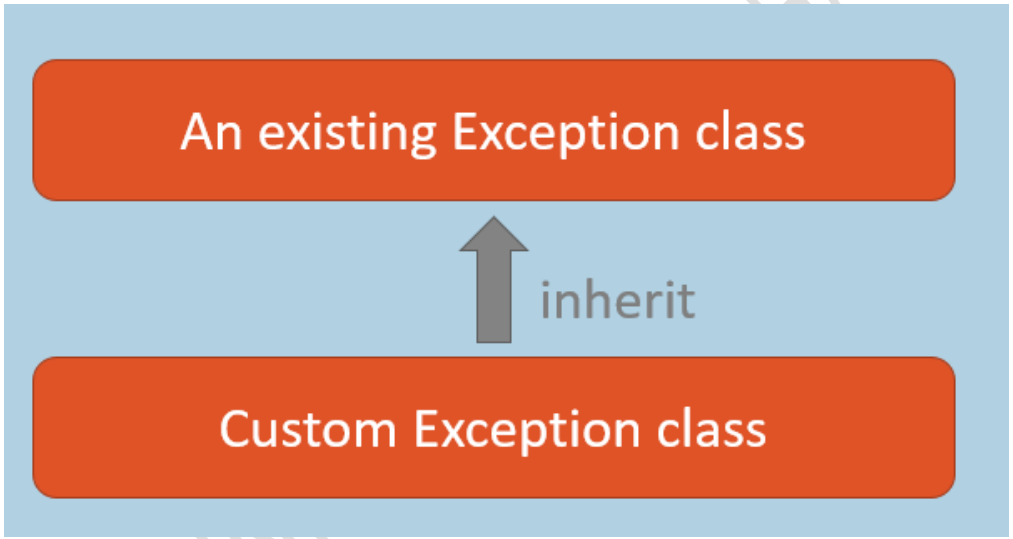
It initializes 'Message' and 'InnerException' properties.

Custom Exception class

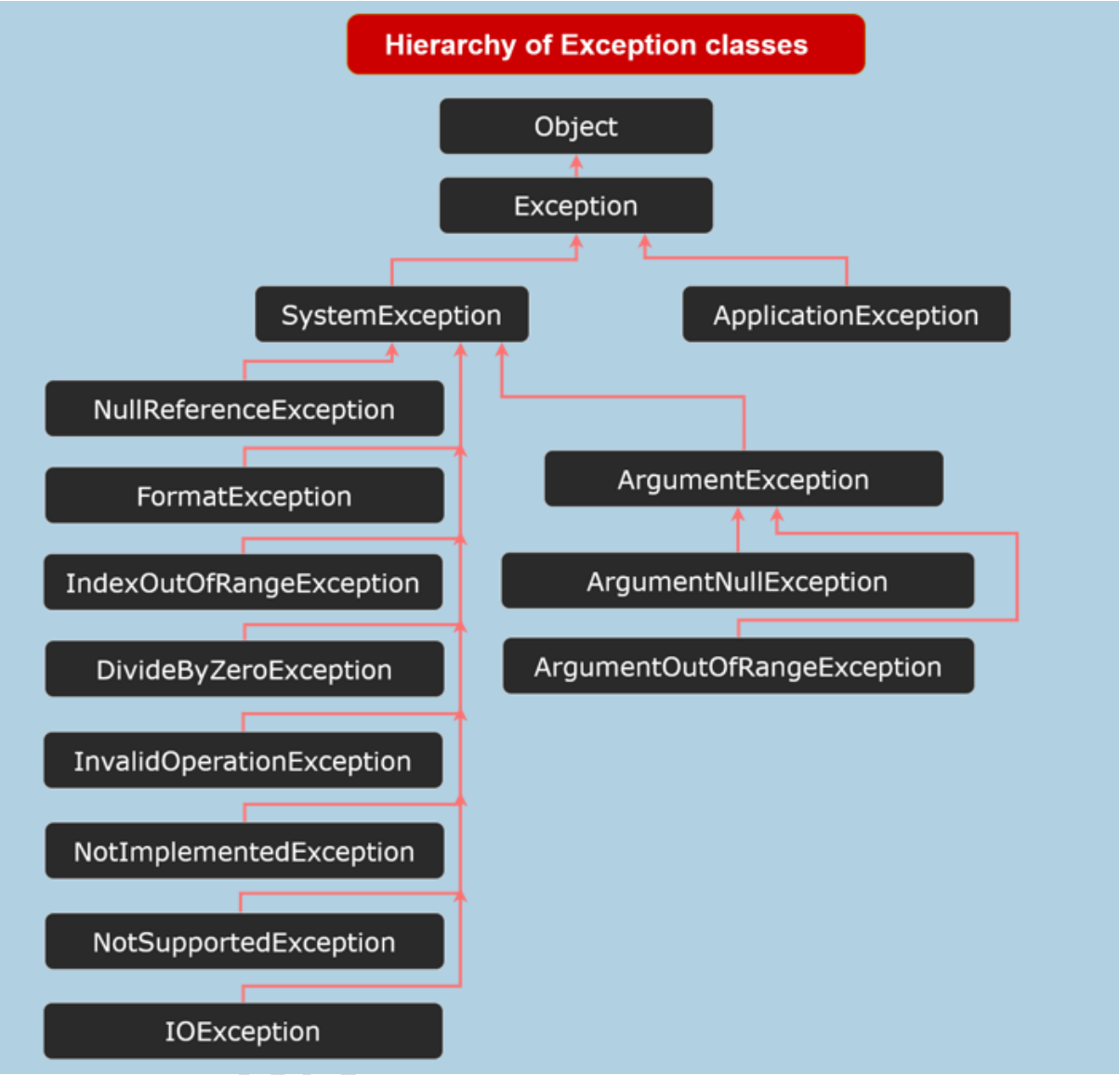
Custom Exception class is a class that is derived from any one of the exception classes (such as System.Exception or any other exception class).

If none of the pre-defined exception class meets your requirement, you will create an user-defined exception class (custom exception class).

throw new CustomExceptionClassName(…)



Hierarchy of Exception Classes



All exception class in .net are derived from System.SystemException class.

System.SystemException is derived from System.Exception class.

Custom Exception classes should NOT be inherited from ApplicationException.

Catch-When

New feature introduced in C# 7.1.

The "catch" block catches the exception, only when the given condition "true".

"Catch-when" is also known as "Exception Filters".

Eg:

```
try
{
  //statements
}
catch (ExceptionType referenceVariable) when (condition)
{
  //error handling
}
```

'nameof' operator

Introduced in C# 6.0.

Returns actual name of the specified field / property.

Useful when you are writing same code for multiple properties.

Eg:

```
nameof ( FieldOrPropertyName )
```