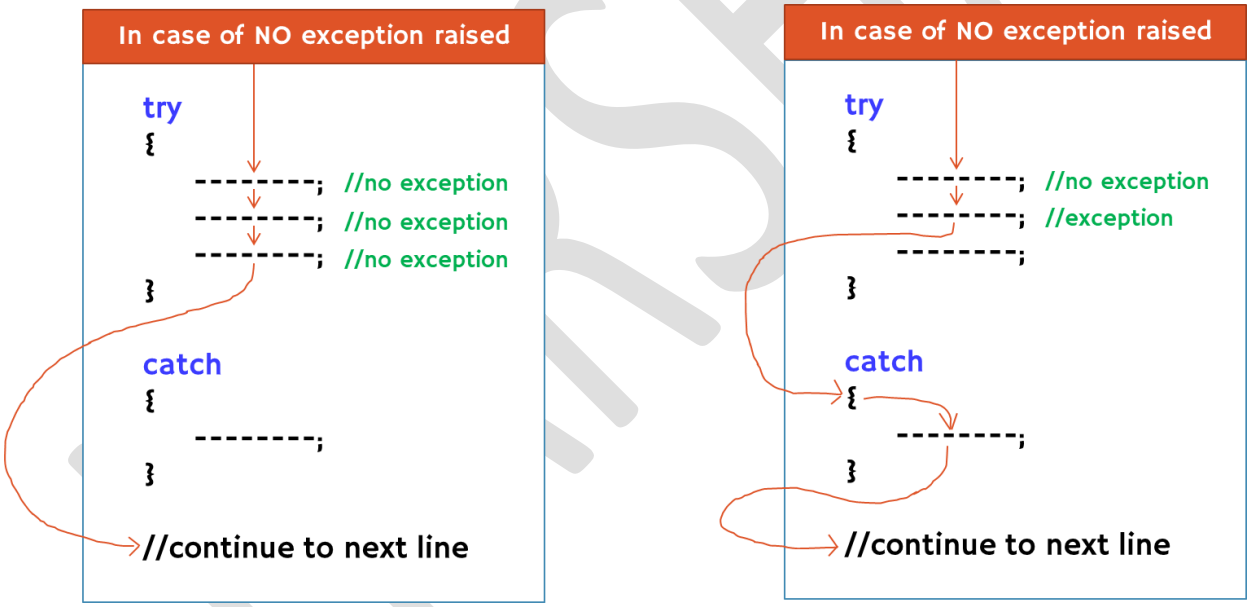
41

Exception Handling

Introducing 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.



- 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.

Exception Classes

1. **System.Exception**
    - Base class for all exception classes.
    - Properties: Message, StackTrace, InnerException
2. **System.IO.IOException**
    - Error during reading / writing some file in the disk.
3. **System.IndexOutOfRangeException**
    - Specified index is not found in the collection or array.
4. **System.NullReferenceException**
    - The reference variable contains "null"; but you have tried to access some member though it.
5. **System.InvalidOperationException**
    - The current state of the object is unable to execute a specific method.
6. **System.ArgumentException**
    - The argument supplied to the method is invalid.
7. **System.FormatException**
    - Unable to convert the given string into number (as the string contains other than digits).
8. **System.Data.SqlClient.SqlException**
    - Unable to read / write from the SQL Server database.

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".

Catch When

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

Custom Exception Class

- Inherited from System.ApplicationException or any other built-in exception class.
- Represents an exception related to specific entity. Ex: CustomerException
- It is bad idea to inherit from System.Exception or System.SystemException.

Custom Exception Class

```
class ClassName : System.ApplicationException
{
    //constructors
}
```

'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.

'nameof' operator

```
nameof ( FieldOrPropertyName )
```