Exception Handling

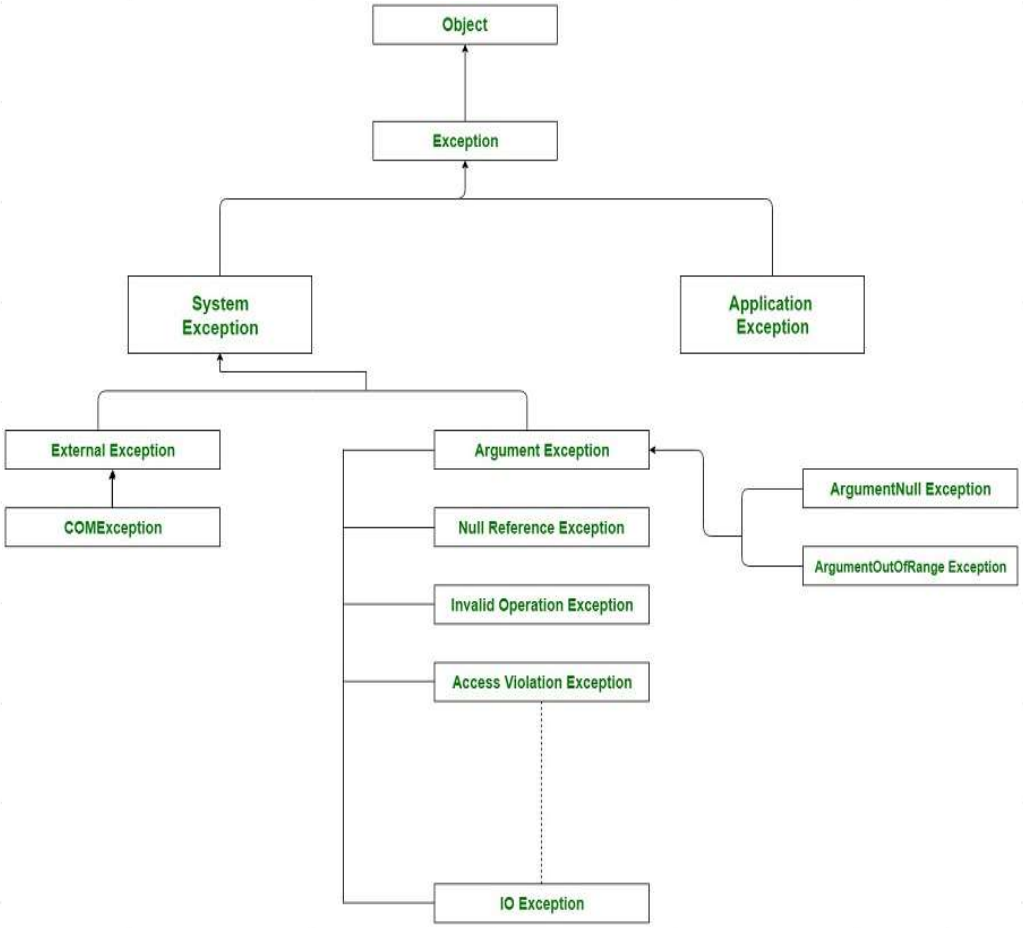
Arctech Info Private Limited

What are exceptions

- An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at runtime, that disrupts the normal flow of the program's instructions.
- Sometimes during the execution of the program, the user may face the possibility that the program may crash or show an unexpected event during its runtime execution
- This unwanted event is known as Exception and it generally gives the indication regarding something wrong within the code.

Exception Hierarchy

- In C#, all the exceptions are derived from the base class Exception which gets further divided into two branches as ApplicationException and SystemException.
- SystemException is a base class for all CLR or program code generated errors.
- ApplicationException is a base class for all application related exceptions.
- All the exception classes are directly or indirectly derived from the Exception class. In case of ApplicationException, the user may create its own exception types and classes. But SystemException contains all the known exception types such as DivideByZeroException or NullReferenceException etc.



Different Exception Classes

- There are different kinds of exceptions which can be generated in C# program:
- **Divide By Zero exception**: It occurs when the user attempts to divide by zero
- **Out of Memory exceptions**: It occurs when then the program tries to use excessive memory
- **Index out of bound Exception**: Accessing the array element or index which is not present in it.
- **Stackoverflow Exception:** Mainly caused due to infinite recursion process
- **Null Reference Exception** : Occurs when the user attempts to reference an object which is of NULL type.

.....and many more.

Properties of the Exception Class

- The Exception class has many properties which help the user to get information about the exception during the exception.
  - **Data**: This property helps to get the information about the arbitrary data which is held by the property in the key-value pairs.
  - **TargetSite**: This property helps to get the name of the method where the exception will throw.
  - **Message**: This property helps to provide the details about the main cause of the exception occurrence.
  - **HelpLink**: This property helps to hold the URL for a particular exception.
  - **StackTrace**: This property helps to provide the information about where the error occurred.
  - **InnerException**: This property helps to provide the information about the series of exceptions that might have occurred.

What is exception handling

- The execution of an exception handler so that the program code does not crash is called exception handling. Exception handling is important because it gracefully handles an unwanted event, an exception so that the program code still makes sense to the user.

Keyword	Definition
try	Used to define a try block. This block holds the code that may throw an exception.
catch	Used to define a catch block. This block catches the exception thrown by the try block.
finally	Used to define the finally block. This block holds the default code.
throw	Used to throw an exception manually.

Syntax

```
Try{
  // statements that may cause an exception
}
catch(Specific_Exception_type obj){
  // handler code
}
catch(Specific_Exception_type obj){
  // handler code
}
finally{
  //default code
}
```

- There can be multiple catch blocks for one try block, to process different types of exceptions
- Finally block can be only one
- Try catch blocks can be nested within each other