Exceptions

Casey Peng

What is an exception?

Definition: An *exception* is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

Exception Object: contains error type and the state of the program when the error occurred.

Creating an exception object and passing it to the runtime system is called *throwing an exception*.

https://docs.oracle.com/javase/tutorial/essential/exceptions/definition.html

Use Try, catch and finally to handle exceptions…

1. Enclose the code that might throw an exception inside a try block

2. Provide one or more catch blocks after try block, each catch block is an exception handler that takes care of the type of exception. Only catch exceptions that you can handle.

3. Finally block: use to release resources, for example, to close files that were open in the try block. Finally block always executed, even when no exceptions occurs.

*Exceptions can be explicitly generated by a program by using the `throw` keyword.

.

Syntax of exception handling ...

1. `try {} catch{};`
2. `try {} catch {} catch{};`
3. `try {} finally{};`
4. `try {} catch {} finally{};`
5. Using throw to generate exception :

```
throw new IndexOutOfRangeException();
```

```
//try-catch
using System;
using System.IO;

0 references
public class ProcessFile
{
    0 references
    public static void Main()
    {
        try
        {
            using (StreamReader sr = File.OpenText("data.txt"))
            {
                Console.WriteLine($"The first line of this file is {sr.ReadLine()}");
            }
        }//open a file that does not exits
        catch (FileNotFoundException e)
        {
            Console.WriteLine($"The file was not found: '{e}'");
        }
        //The exception that is thrown when part of a file or directory cannot be found.
        catch (DirectoryNotFoundException e)
        {
            Console.WriteLine($"The directory was not found: '{e}'");
        }
        //IOException is the base class for exceptions thrown while accessing information using streams, files and directories.
        catch (IOException e)
        {
            Console.WriteLine($"The file could not be opened: '{e}'");
        }
    }
}
```

When handling an exception...

Try to use an existing exception type in the .NET Framework instead of a custom one.

You can use the existing exception when

- Exception that is caused by a usage error(error made by developer who is calling your method)
- You are handling an error that can be communicated to the caller with an existing .Net Framework exception. Try to throw a more specific exception , for example, throw an InvalidEnumArgumentException rather than an Argument Exception

You can also custom exceptions when ...

1. Program throw a unique exception that doesn't exit in an existing .NET Framework exception.

2. The exception required a handling that is different from the existing .NET Framework exception.

https://docs.microsoft.com/en-us/dotnet/api/system.exception?view=net-5.0

HOW TO CUSTOM YOUR OWN EXCEPTION CLASS...

1. Define a class that inherits from Exception(base class for all exceptions)
2. If needed, override any inherited members whose functionality you want to change or modify.
3. Determine whether your custom exception object is serializable:serialization enables you to save information about the exception and share the information
4. Define the constructors of your exception class.

https://docs.microsoft.com/en-us/dotnet/api/system.exception?view=net-5.0

Common exception types and condition you would throw them

Exception	Condition
ArgumentException	A non-null argument that is passed to a method is invalid.
ArgumentNullException	An argument that is passed to a method is `null`.
ArgumentOutOfRangeException	An argument is outside the range of valid values.
DirectoryNotFoundException	Part of a directory path is not valid.
DivideByZeroException	The denominator in an integer or Decimal division operation is zero.
DriveNotFoundException	A drive is unavailable or does not exist.
FileNotFoundException	A file does not exist.
FormatException	A value is not in an appropriate format to be converted from a string by a conversion method such as `Parse`.
IndexOutOfRangeException	An index is outside the bounds of an array or collection.
InvalidOperationException	A method call is invalid in an object's current state.
KeyNotFoundException	The specified key for accessing a member in a collection cannot be found.
NotImplementedException	A method or operation is not implemented.

Not all errors should be handled as exceptions...

1. Usage errors(an error in program logic) should be handled by modifying the error code
2. Program errors(run-time errors) that are not caused by code with bugs
3. System Failure:run-time error that cannot be handled programmatically

Exception Class (System) | Microsoft Docs

The END

Exception Hierarchy
in c#