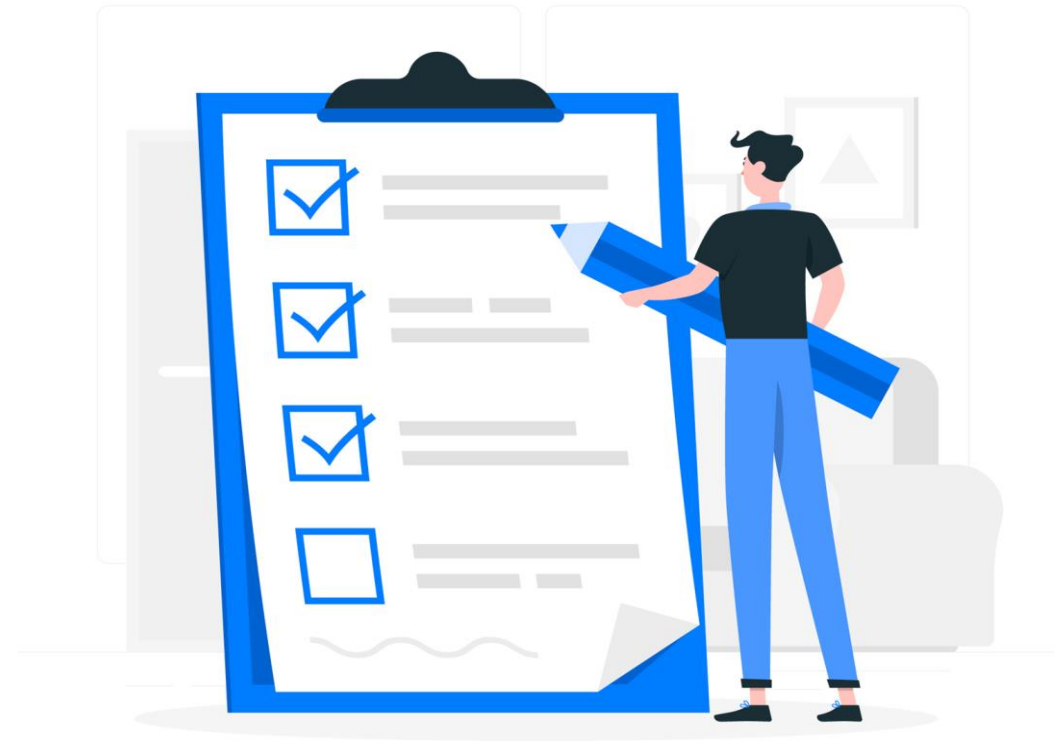


EXCEPTION HANDLING



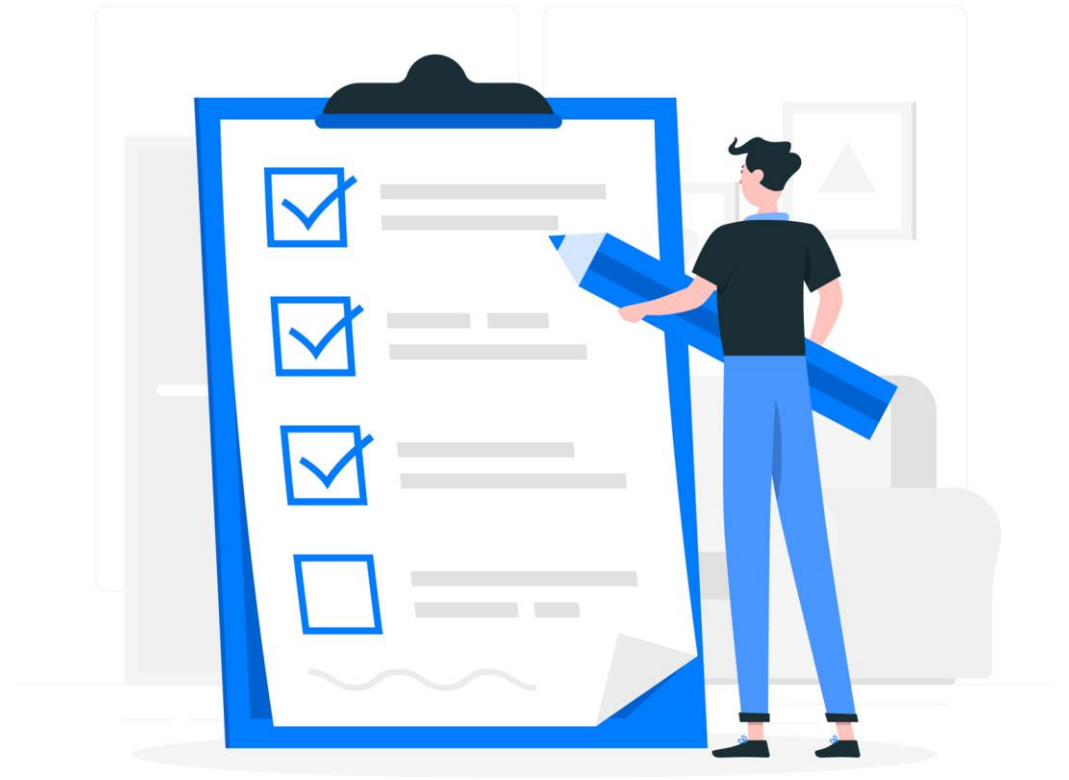
CONTENTS

- What is Exception?
- Why should we catch exceptions ?
- 'Exception' and 'Error' ?
- The different kinds of exceptions
- Namespace
- Some System/Runtime Exceptions
- What is Exception Handling?
- Structured Exception Handling



CONTENTS

- **try/catch Blocks**
- **Multiple Catch Blocks**
- **Finally Blocks**
- **Application Exception**
- **Defensive Programming**
- **Best Practices for Handling Exceptions**



What Is Exception???

Exception is an event during the execution of the program

It disrupts the normal flow of instructions during the execution of a program.

An exception is an unexpected state during the execution of code



Why should we catch exceptions

- To make your program robust and reliable.
- And it is vital for ensuring the quality of your code .
- A good programmer is someone who never introduce bad code in his projects
- If an exception is not 'handled' in code



The application will crash and user will see an ugly message.

Instead, you can catch the exception, log the errors and show a friendly message to the user.

Exception and Error?

An error is the event; an exception is the object that event creates.

When an error interrupts the flow, the program tries to find an exception handler

A block of code that tells it how to react, Which will help it resume the flow.



TYPES OF EXCEPTION

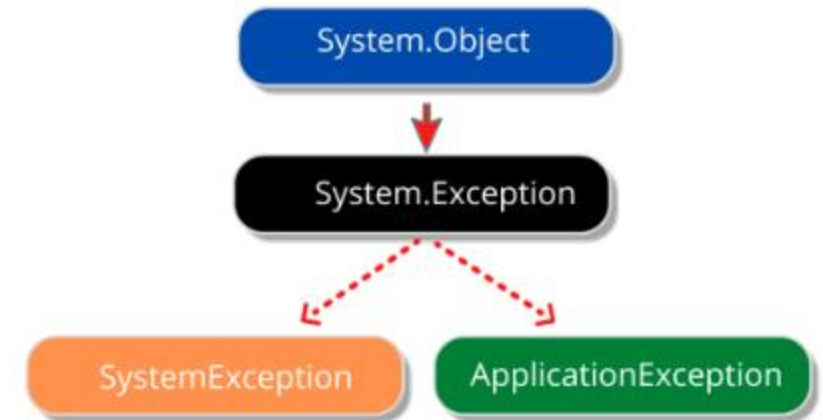
SYSTEM EXCEPTION

Thrown by the CLR at run time

APPLICATION EXCEPTION

Thrown By Application,using “throw”

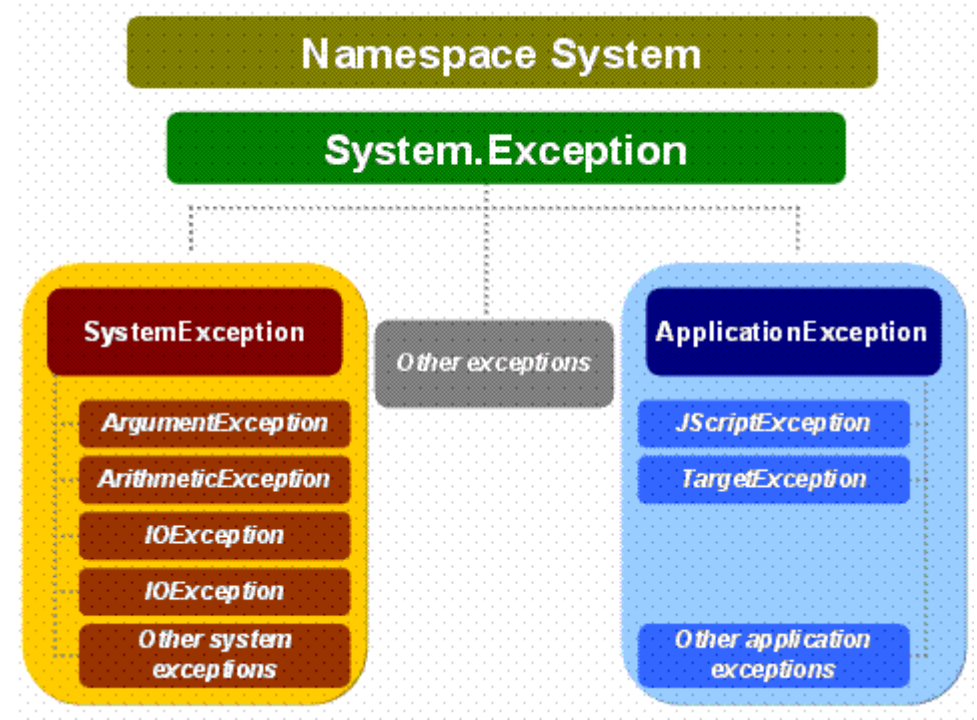
C#
EXCEPTION



Clause

Namespace

System.Exception is the base class for all exceptions in C#.



System/Runtime Exceptions

OutOfMemoryException

NullReferenceException

InvalidCastException

ArrayTypeMismatchException

IndexOutOfRangeException

Arithmetic Exception

OverflowException

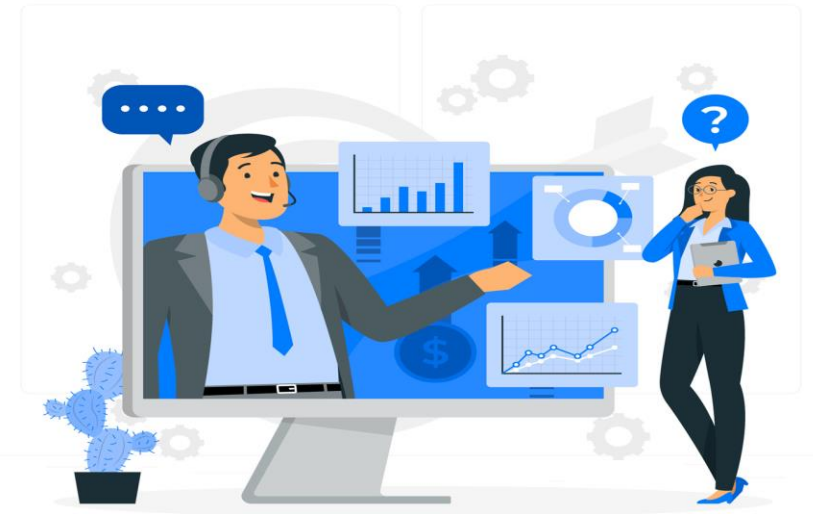
DivideByZeroException

What Is Exception Handling??

“Exception handling” means interpreting and reacting to the exceptions created by errors.

Built mechanism in .NET framework to detect and handle run time errors.

Achieved using the Try - Catch - Finally block



Structure:

Code Block for which we want to catch some Exception

Each catch deals with class of Exceptions, determined by the run-time system based on the type of the argument

The code in finally is executed always after leaving the try-block

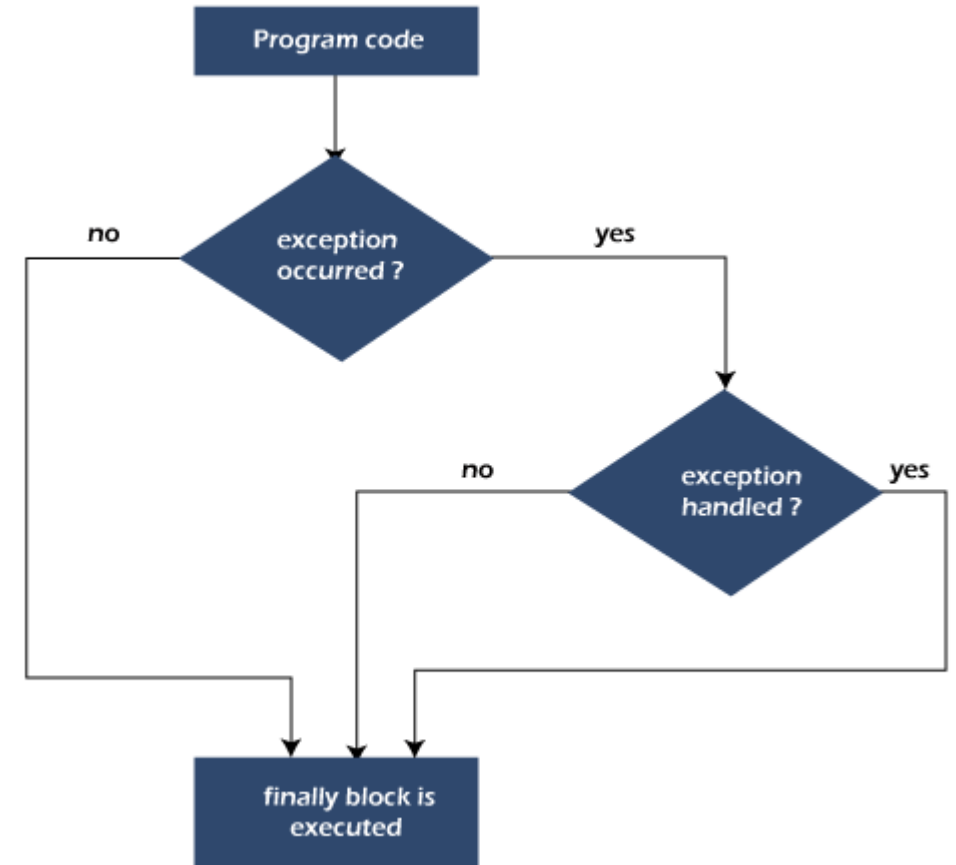
```
try{  
    .....  
}  
catch(SomeException e1){  
    .....  
}  
catch(AnotherException e2){  
    .....  
}  
finally{  
    .....  
}
```

Work Flow:

Structured exception handling tests specific pieces of the code when exception occurs.

The **Try...Catch...Finally** control structure is fundamental to structured exception handling.

It Reacts differently based on the type of thrown exception.



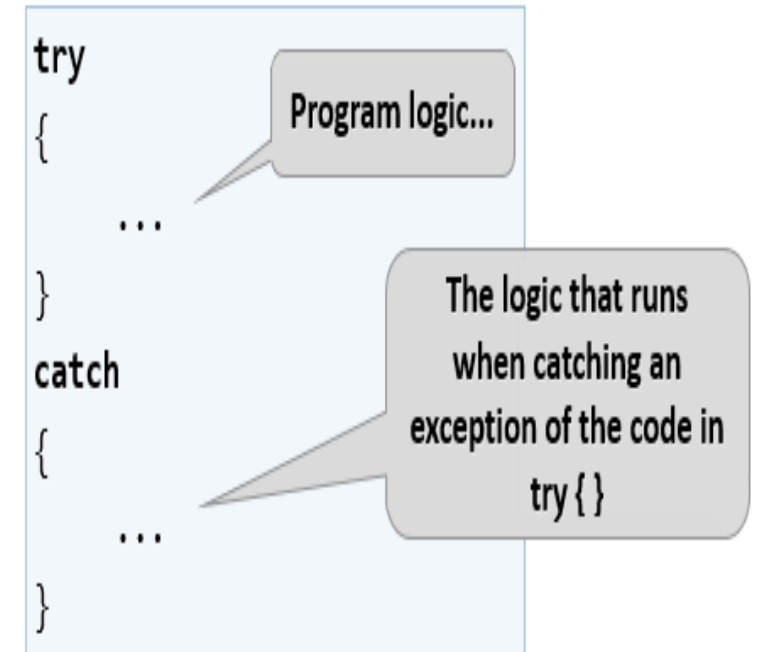
try/catch Blocks

When exceptions are thrown, It must be handled

Done by using try-catch block

Code that could throw an exception is put in the try block

Exception handling code goes in the catch block



Multiple Catch Blocks

A try block can throw multiple exceptions, which can handle by using multiple catch blocks.

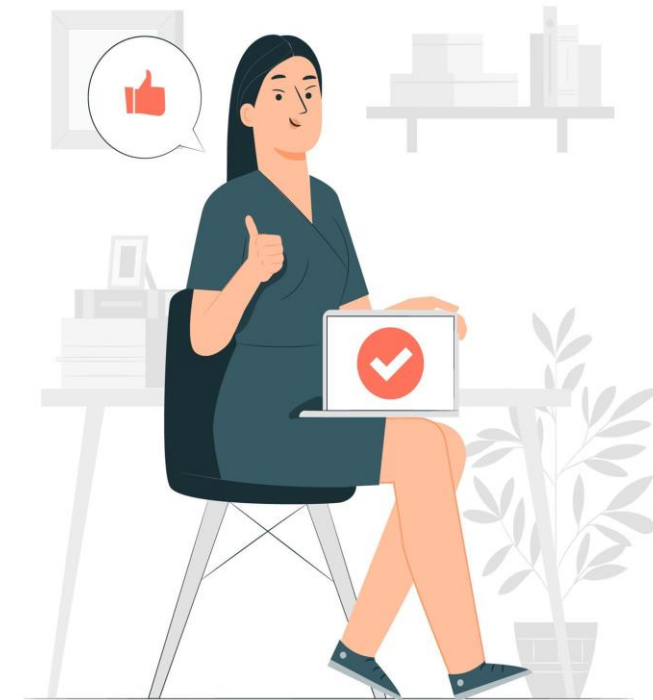


Finally Blocks

Finally block must appear after all the catch block.

Contains the code that always executes, whether or not any exception occurs.

If you use transfer control statement in finally block, you will receive compile time error.



When to use finally block ?

- A code fragment that must be executed regardless of exception.
- Cleaning up resources such as closing file objects, releasing database connections etc.
- Only One finally block is associated with try block.

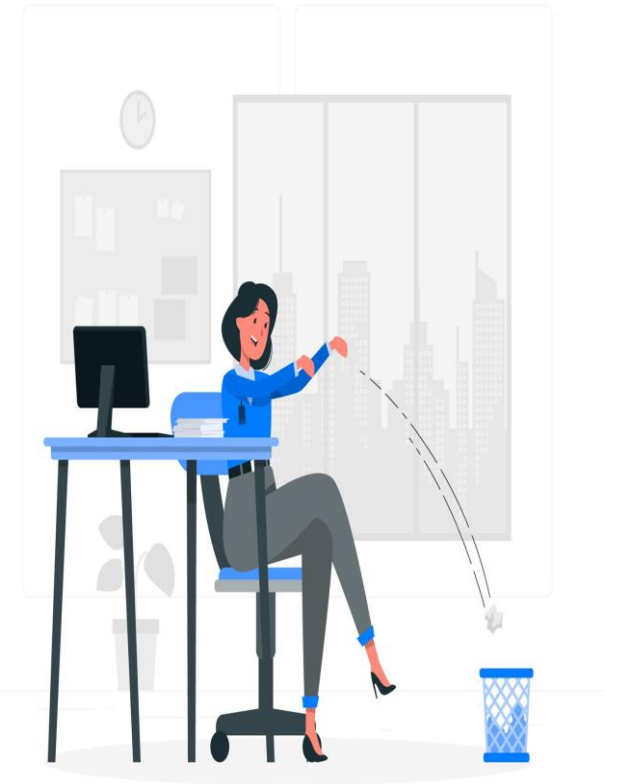


Application Exception

Thrown by a user program, not by the common language runtime

Create a custom exception class by deriving the `ApplicationException` class.

The **throw** statement is used to signal the occurrence of an anomalous situation (exception) during the program execution.

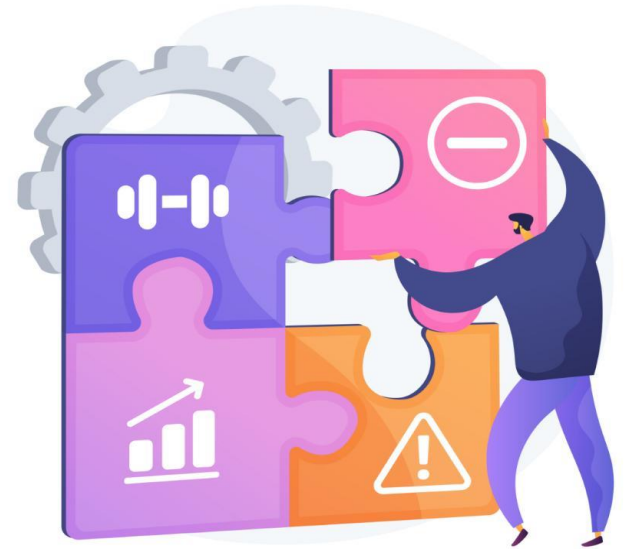


Diffensive Programming

Defensive programming is a technique that makes programs more robust to unexpected events .

Less program "crashes" at run time ,It increases the quality of software.

Defensive Programming is the concept of avoiding the exceptional situations to occur, even before it can occur, and exception handling is about handling an exception after it has happened,



Defensive Programming Strategies

Check For Nulls

Be Careful When Working with
Arrays

Check Object State

Use Safe Parsing

Check Your Arguments

Use Safe Parsing



Best Practices for Handling Exceptions

A well-designed set of error handling code blocks can make a program more robust and less prone to crashing the application

Know when to use a try/catch block

Use try/finally blocks around code that can potentially generate an exception and centralize your catch statements in one location.



End exception class names with the word "Exception".

In C# and the Managed Extensions for C++, use at least the three common constructors when creating your own exception classes.

Use grammatically correct error messages, including ending punctuation



