

Handling Exceptions



Gill Cleeren

CTO Xebia Microsoft Services Belgium

@gillcleeren

Overview



Understanding exceptions in code

Using a try/catch block

Catching several types of exceptions

Using finally



Understanding Exceptions in Code





Errors will occur!

**Errors are problems that will occur
while our application is executing.**



Exceptions Will Occur



Divide by zero



File not accessible



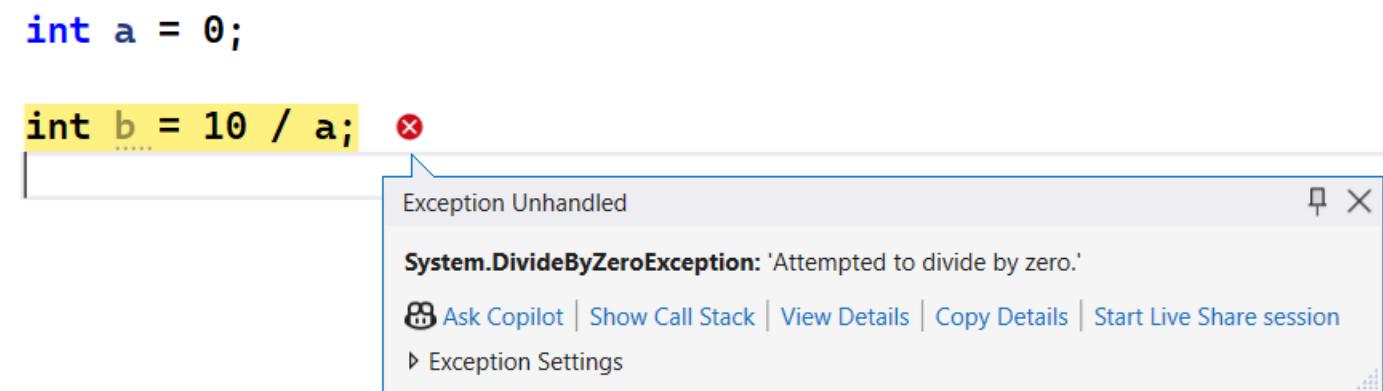
Incorrect permissions



Database unreachable



The Default Handling of the Exception



```
Microsoft Visual Studio Debug X + ▾
```

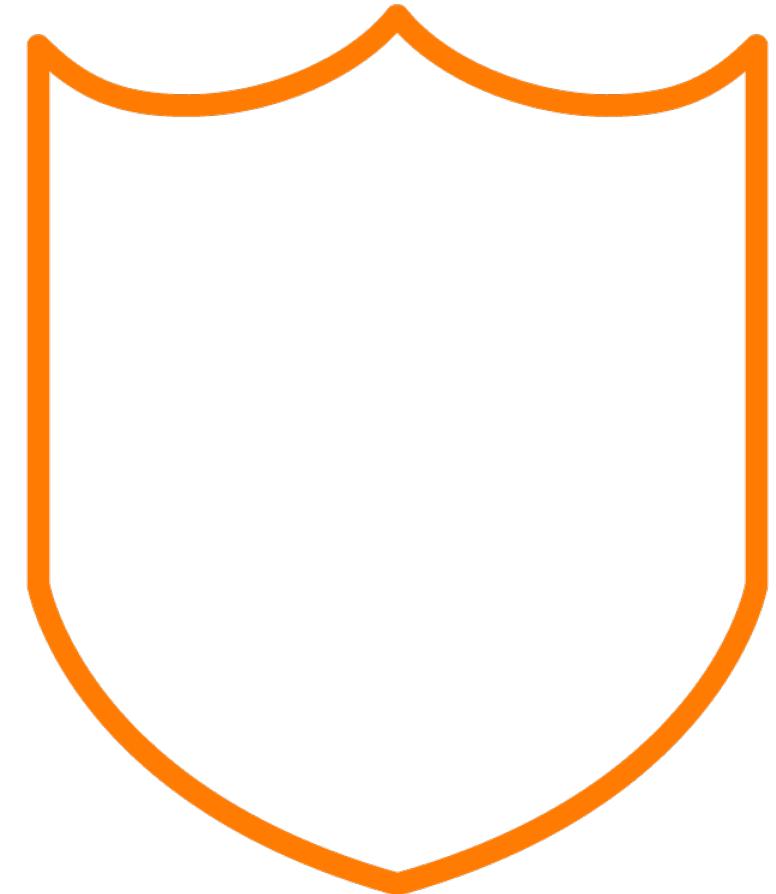
```
Unhandled exception. System.DivideByZeroException: Attempted to divide by zero.
at Program.<Main>$(<Main>$<Main> args) in D:\code\CA\HelloWorld\HelloWorld\Program.cs:line 3

D:\code\CA\HelloWorld\HelloWorld\bin\Debug\net8.0\HelloWorld.exe (process 31704) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```



Using a try/catch Block





In C#, error handling code can be written separately

- Application code executes
- Covered by exception handling code
- Exceptions that occur can be handled

Based on

- try
- catch



Using a try/catch block

```
try
{
    //code goes here
}
catch (Exception ex)
{
    //Here we can handle the exception
    throw;
}
```



The try Block

```
try
{
    //code goes here
}
catch (Exception ex)
{
    throw;
}
```

Code in try block is attempted
If all goes well, will execute regularly
If execution fails, jumps to catch block



The catch Block

```
try
{
    //code goes here
}
catch (Exception ex)
{
    throw;
}
```

Exceptions will be caught here

Handles specific type of exception

- Here we have a base exception
- Multiple types of exceptions can be handled differently



Using a try/catch Block

```
try
{
    string input = Console.ReadLine();
    int a = int.Parse(input);
}
catch (FormatException ex)
{
    //Here we can handle the exception
}
```





Should all code go in a try/catch block?



Exceptions in the Documentation

Exceptions

[FileNotFoundException](#)

The file is not found.

[UnauthorizedAccessException](#)

The file is read-only or is a directory.

[DirectoryNotFoundException](#)

The specified path is invalid, such as being on an unmapped drive.

[IOException](#)

The file is already open.



Demo



Adding exception handling



Inspecting the Exception Details

Message

InnerException

StackTrace

HelpLink



Inspecting the Exception Details

```
try
{
    string input = Console.ReadLine();
    int a = int.Parse(input);
}
catch (FormatException ex)
{
    Console.WriteLine(ex.Message);
    Console.WriteLine(ex.StackTrace);
}

}
```



Demo



Using the exception details



Catching Several Types of Exceptions

Using Multiple Exception Types

```
try
{
    string input = Console.ReadLine();
    int a = int.Parse(input);
    int b = 10 / a;
}
catch (FormatException fex)
{
    //Here we can handle the exception
}
catch (DivideByZeroException dbzex)
{
    //Here we can handle the exception
}
```





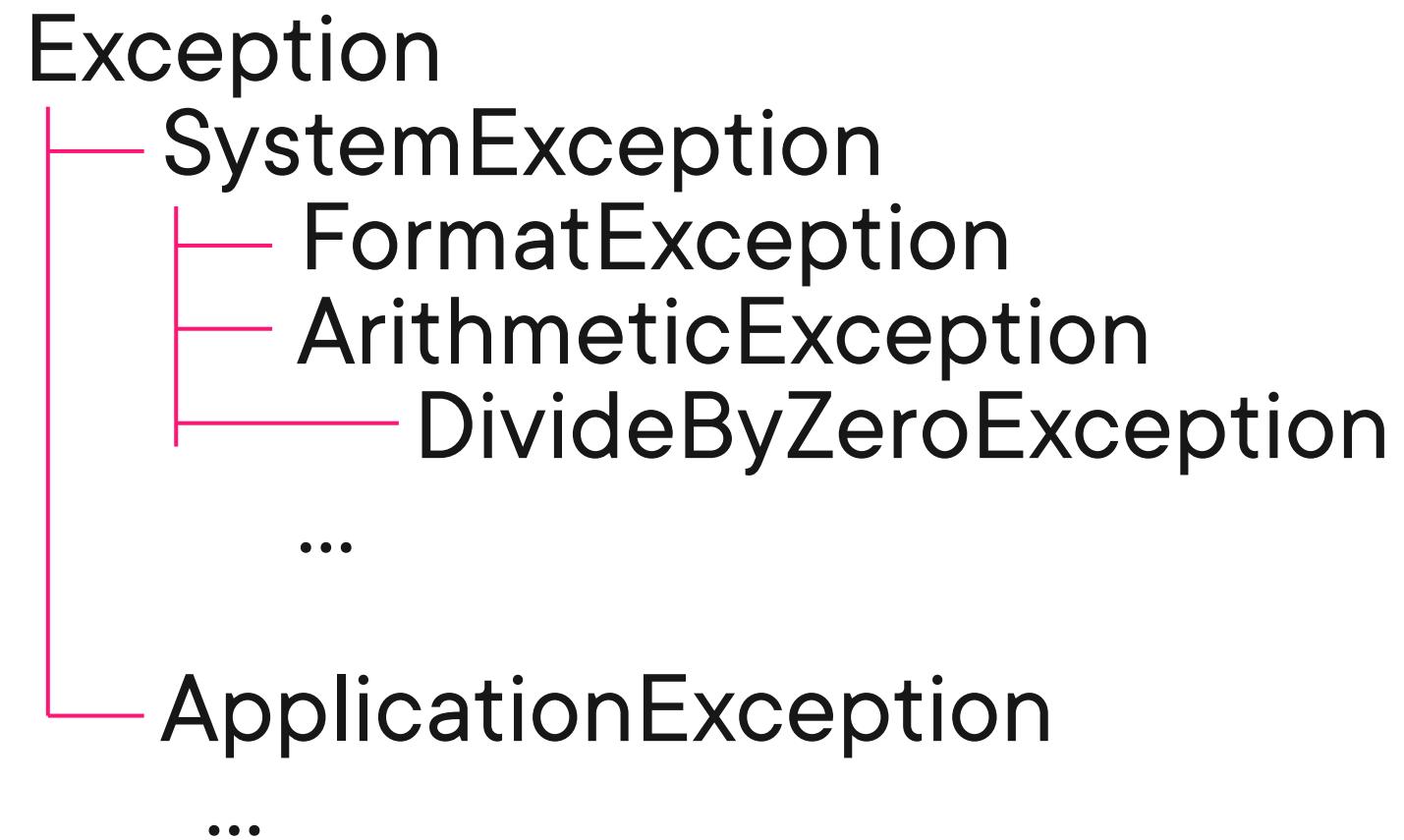
Handling different exception types

.NET comes with many exception types

Exceptions are classes and are part of a hierarchy



The Exception Hierarchy



Catching All Possible Exceptions

```
try
{
    string input = Console.ReadLine();
    int a = int.Parse(input);
    int b = 10 / a;
}
catch (FormatException fex)
{
    //Here we can handle the exception
}
catch (DivideByZeroException dbzex)
{
    //Here we can handle the exception
}
catch(Exception ex)
{
    //This will be invoked if another type of exception occurs
}
```



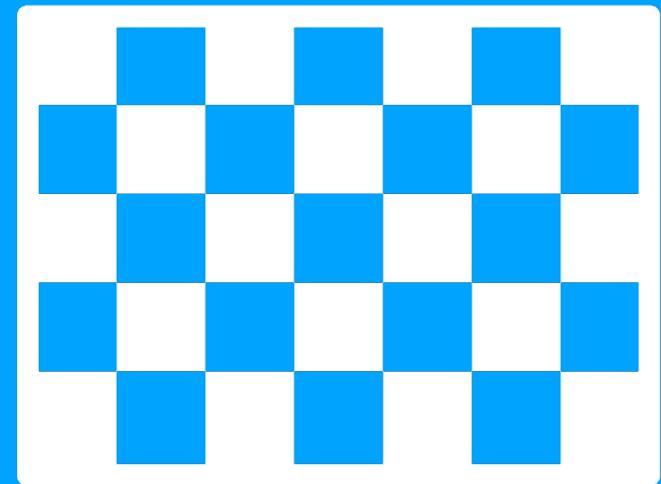
Demo



Catching multiple types of exceptions



Using finally



Finally!

If we have code that needs to run regardless if all went fine or not, we can add a finally block.



Using a finally Block

```
try
{
    //code goes here
}
catch (Exception ex)
{
    //Here we can handle the exception
    throw;
}
finally
{
    //This will always execute
}
```



Demo



Adding a finally block



Summary



A try/catch block makes our code more robust

The finally block will always execute



Up Next:

Your next steps with C#

