try/catch

In this lesson, you'll learn how the try/catch block is used via examples.

WE'LL COVER THE FOLLOWING

Introduction

The `try/catch` performs an operation and should an error occur, will transfer control to the `catch` block. Try keyword is used to monitor the exceptions which are there in our code. Followed by the `try` block `catch` block should be there.

```
using System;

class ExceptionTest
{
  public static void Main(string[] args)
  {
    try
    {
      Console.WriteLine(args[0]);
      Console.WriteLine(args[1]);
      Console.WriteLine(args[2]);
      Console.WriteLine(args[3]);
      Console.WriteLine(args[4]);
    }
    catch (IndexOutOfRangeException e)
    {
      Console.WriteLine(e.Message);
    }
  }
}
```

Multiple Catches

Here is an example with multiple `catches`:

main.cs

log2.txt

```
using System;
using System.IO;

class ExceptionTest
{
  public static void Main()
  {
    try
    {
      string fileContents = new StreamReader(@"log.txt").ReadToEnd();
      Console.Write(fileContents);
    }
    catch (UnauthorizedAccessException e) // Access problems
    {
      Console.WriteLine(e.Message);
    }
    catch (FileNotFoundException e)       // File does not exist
    {
      Console.WriteLine(e.Message);
    }
    catch (IOException e)                 // Some other IO problem.
    {
      Console.WriteLine(e.Message);
    }
  }
}
```

In all `catch` statements you may omit the type of exception and the exception variable name:

```
using System;

class ExceptionTest
{
  public static void Main(string[] args)
  {
    try
    {
      int number = 1/0;
    }
    catch (DivideByZeroException)
    {
      // DivideByZeroException
    }
    catch
```

```
    catch
    {
      // some other exception
    }

  }
}
```

Interesting, right? Now let's move onto the `try/catch/finally` block!