File IO & Exceptions (Error Handling)

Software Engineering 1

Dr Swen E. Gaudl

University of Plymouth 2022

COMP10000

Mobile.Plymouth.ac.uk

Please Sign in using Code: _ _

Prep For Thursday:

- Work on Exercise3
- Look into the used concepts & come with questions!
- Do revisit the recordings if things are unclear!
- Experiment try things out yourself!

COMP1000 Agenda This Week:

- File Reading
- Loops Iteration (Worked into examples)
- Q&A

Data Input/Output

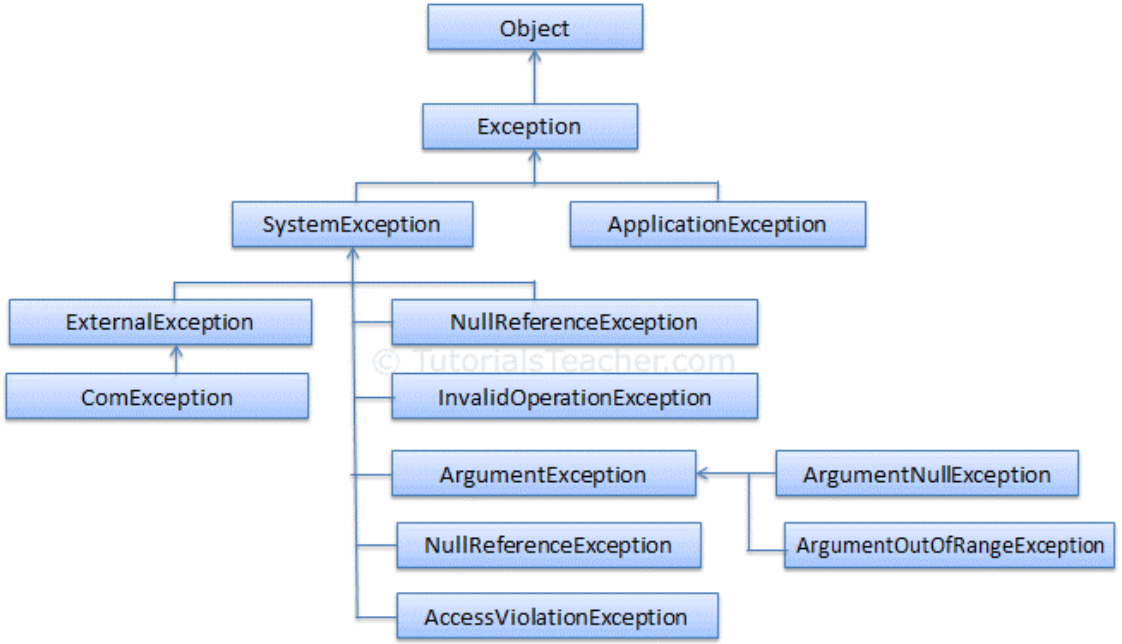
```
Console.WriteLine("Move a Move: ");

ConsoleKeyInfo name = Console.ReadKey();

Console.WriteLine("You pressed {0}", name.KeyChar);

string message = Console.ReadLine();
```

Error Handling

Try- Catch - Finally

csharp/programming-guide/exceptions/exception-handling

```
try
{
    number = float.Parse("1.0");
}
catch (FormatException ex)
{
    Console.Error.WriteLine(ex.Message);
    number = float.NaN;
}
```

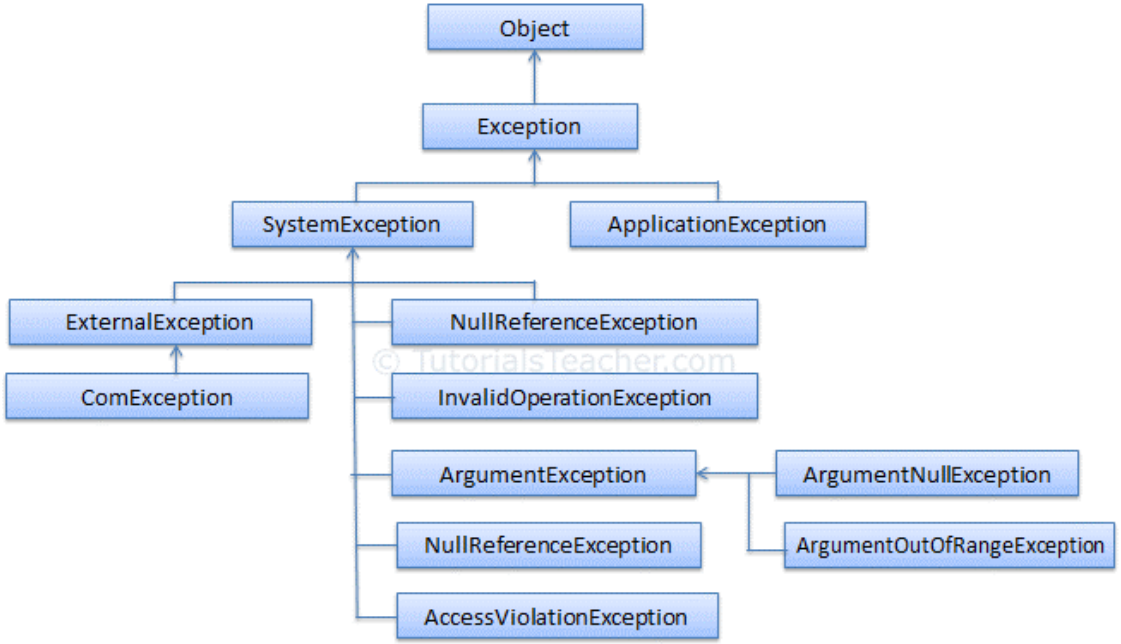


Error Handling

Try- Catch - Finally

csharp/programming-guide/exceptions/exception-handling

```
float [] numbers = new float[5];
try
{
    numbers[6] = 1.1f;
}
catch (ArgumentoutofRangeException ex)
{
    Console.Error.WriteLine(ex.Message);
    // how to deal with this now?
}
```



File Input/Output

Simple Full File Reading

https://docs.microsoft.com/en-us/dotnet/api/system.io.file.readalltext

```
try
{
    // Open the file to read from.
    text = File.ReadAllText(filePath, Encoding.UTF8);
}
catch (FileNotFoundException ex)
{
    Console.Error.WriteLine(ex.Message);
    text = "";
}
finally
{
    text = text.Trim();
}
```

File Input/Output

Simple Full File Reading

https://docs.microsoft.com/en-us/dotnet/api/system.io.file.readalltext

```
try {
    if (!File.Exists(filePath))
    {
        // Create a file to write to.
        string createText = "Beginning Log:" + Environment.NewLine;
        File.WriteAllText(filePath, createText, Encoding.UTF8);
     }
     foreach (string line in text)
     {
         File.AppendAllText(filePath, line + Environment.NewLine, Encoding.UTF8);
     }
}
catch (IOException ex)
{
    Console.Error.WriteLine(ex);
}
```

File Input/Output

https://www.csharp-examples.net/read-text-file/
https://www.tutorialspoint.com/csharp/csharp_text_files.htm

```
string[] lines;
var list = new List<string>();
var fileStream = new FileStream(filePath, FileMode.Open, FileAccess.Read);

using (var streamReader = new StreamReader(fileStream, Encoding.UTF8))
{
    string line;
    while ((line = streamReader.ReadLine()) != null)
    {
       list.Add(line);
    }
}
lines = list.ToArray();
```

File Input/Output

https://www.csharp-examples.net/read-text-file/
https://www.tutorialspoint.com/csharp/csharp_text_files.htm

```
string[] lines;
var list = new List<string>();
var fileStream = new FileStream(filePath, FileMode.Open, FileAccess.Read);

using (var streamReader = new StreamReader(fileStream, Encoding.UTF8))
{
    string line;
    while ((line = streamReader.ReadLine()) != null)
    {
       list.Add(line);
    }
}
lines = list.ToArray();
```

File Input/Output Advanced

- FileStream – for reading and writing to a file.
- IsolatedStorageFileStream – for reading and writing to a file in isolated storage.
- MemoryStream – for reading and writing to memory as the backing store.
- BufferedStream – for improving performance of read and write operations.
- NetworkStream – for reading and writing over network sockets.
- PipeStream – for reading and writing over anonymous and named pipes.
- CryptoStream – for linking data streams to cryptographic transformations.

File Input/Output Advanced

More Sophisticated:

- StreamWriter/Stream Reader vanilla solution
- Buffered approach for large/slow resources Bufferedstream

Careful Advanced!