Exceptions

Agenda

Exceptions

Exception handling

Catch exceptions

Throw exceptions

Multiple exception catches

Finally block in an exception catch

Exceptions in C#

An exception = exceptional error condition that needs to be processed differently from your program logic

Exception

Represents an execution error

Occur at runtime

More robust than other error handling methods

An exception is thrown = An error occurred, and an Exception class is instantiated by the runtime and is returned back to the caller

Exception class

It’s a class that is instantiated by the runtime or when you throw an exception

It contains useful information about the error

.NET provides many built-in exceptions

Custom exceptions can be created

Exception class members

`Message`

`StackTrace`

`InnerException`

`Source`

Exception class

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

Built-in exceptions

2 families of exceptions :

- `SystemException`
- `ApplicationException`

Common system exceptions

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

Exception handling : try/catch/finally

Handle Exception

Use a try... catch... finally block

The code is placed in the try

The catch handles an exception when it occurs

The optional finally block is executed whatever

Try catch statement

```
try
{
    int[] array = {1,2,3};
    int unexistingElement = array[10];
}
catch (IndexOutOfRangeException ex)
{
    // Exception handling logic
}
```

Try catch finally statement

```
try
{
    int[] array = {1,2,3};
    int unexistingElement = array[10];
}
catch (IndexOutOfRangeException ex)
{
    // Exception handling logic
}
finally
{
    // Executed whatever, exception thrown or not
}
```

Multiple catch blocks

```
try
{
    // code
}
catch (IndexOutOfRangeException ex)
{
    // Exception handling logic if IndexOutOfRangeException occurs
}
catch (Exception ex)
{
    // Exception handling logic if other type of exception occurs
}
finally
{
    // optional
}
```

Condition filter

```
try
{
    // code
}
catch (ArgumentException ex) when (ex.ParamName == "some-value")
{
    // Exception handling logic if IndexOutOfRangeException occurs
    // and ex.ParamName == "some-value"
}
catch (ArgumentException ex)
{
    // Exception handling logic if other type of exception occurs
    // and ex.ParamName != "some-value"
}
```

Demo

How to use try/catch block

Add try/catch blocks to the console app

Demo

Add try/catch blocks to the api

Exception handling : throw exceptions

Throw an exception

An exception can be instantiated and thrown programmatically

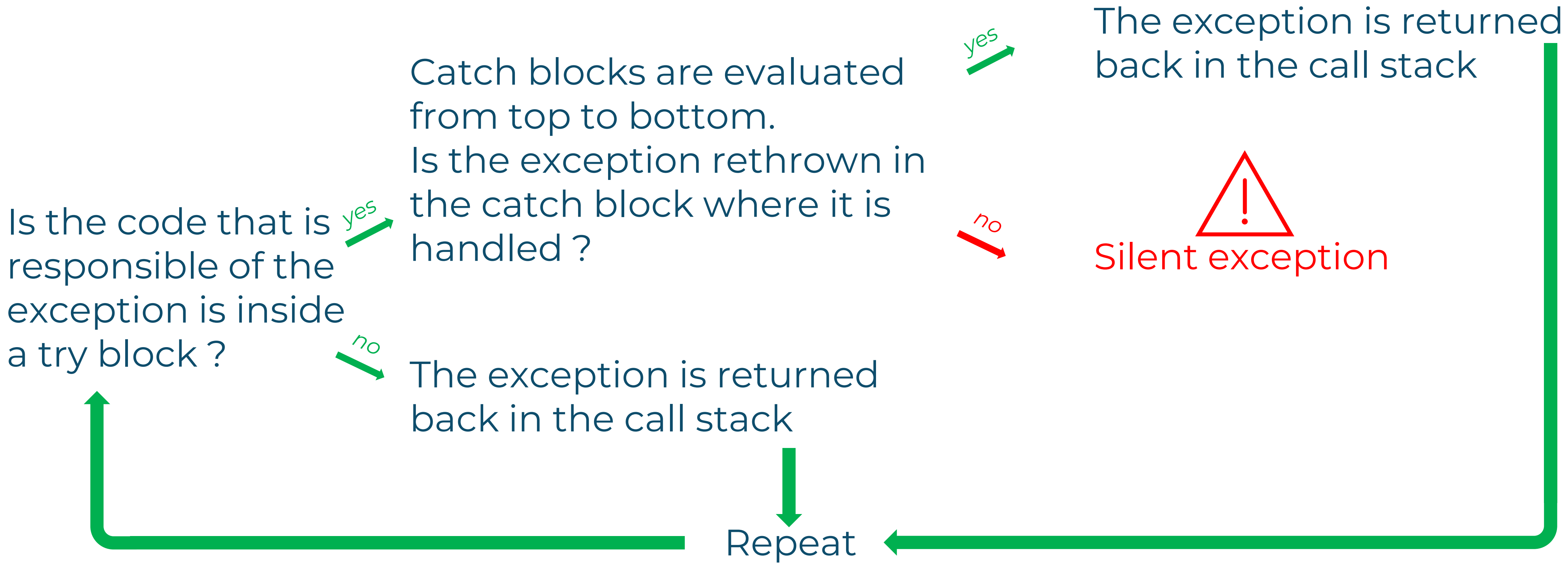
Throw an exception

```
if (string.IsNullOrEmpty(parameter))
{
    throw new ArgumentNullException("Provided argument is empty");
}
```

Rethrow an exception

```
try
{
    int[] array = {1,2,3};
    int unexistingElement = array[10];
}
catch (IndexOutOfRangeException ex)
{
    // Exception handling logic
    throw;
}
```

What happens when an exception is thrown ?



Is the code that is responsible of the exception is inside a try block ?

Catch blocks are evaluated from top to bottom.
Is the exception rethrown in the catch block where it is handled ?

The exception is returned back in the call stack

Silent exception

The exception is returned back in the call stack

Repeat

Demo

Throw exceptions programmatically in api

What happens when an exception reaches the first calling method ?

When an exception reaches the calling program, the error will be displayed on a console application, a web page or a dialog box depending on the type of your application

Demo

Throw exceptions programmatically in the library

Demo

Create try with multiple catch blocks

Demo

Create try with finally block

Summary

An exception is an error condition during execution

The Exception class represents an exception is thrown by the runtime

Exception are handled with a try…catch…finally statement

You can throw exceptions programmatically