Exceptions

CONTENTS

- **Objectives**
  - To explain exception handling in C#
- **Contents**
  - Exception handling syntax
  - Throwing exceptions
  - Understanding execution flow with exceptions
- **Hands-on labs**

Simple example of exception being thrown

- Coding error

```
public static void Main(string[] args)
{
  int[] ages = new int[7];
  ages[7] = 34;
}
```

```
Exception in thread "main"
   IndexOutOfRangeException
```

A few unpredictable exceptions

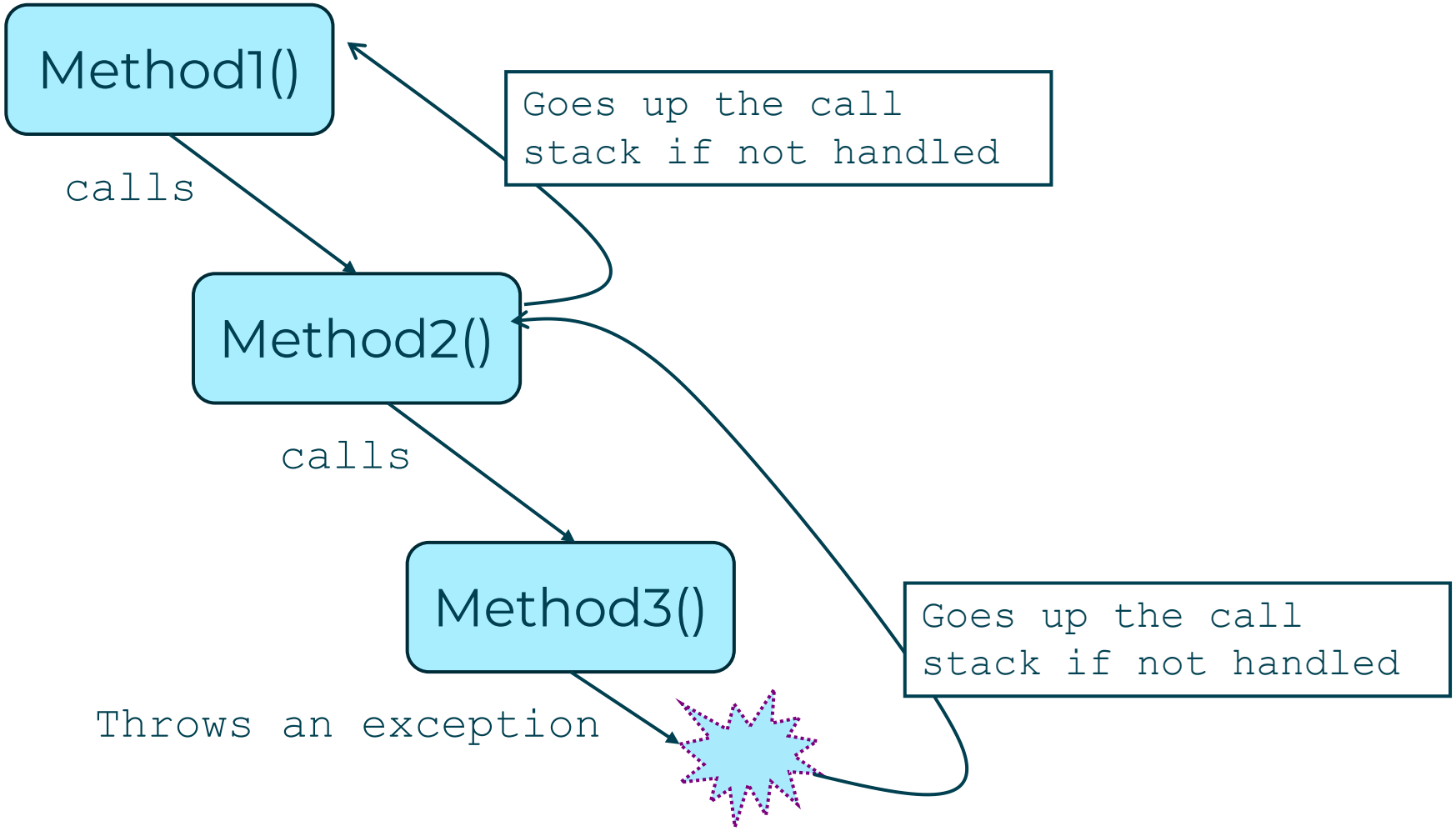
- **Accessing a file**
  - UnauthorizedAccessException
  - FileNotFoundException
- **Accessing objects**
  - NullReferenceException
- **Networking**
  - WebException
  - ProtocolViolationException
  - TimeoutException

TYPES OF EXCEPTIONS

- **Class Exception is the base class of all exceptions**
  - ArgumentException
  - DivideByZeroException
  - FileNotFoundException
  - And any custom exceptions

- **Unlike Java, C# does not have checked exceptions**
  - Checked exceptions are potentially recoverable exceptions (e.g. FileNotFoundException)
  - Handling/declaring checked exceptions is enforced by the compiler
  - Designers of C# chose not to include this feature in the language

Exceptions bubble up the call stack



Example of Exception bubbling up

Same coding error but exception thrown in called method

```
public static void Main(string[] args) {
        int[] ages = new int[7];
        ProcessAges(ages);
}

public static void ProcessAges(int[] ages) {
        Cȯŋṣȯľê Ẅsîʠêĺîŋê("Last age is" + ages[7]);
}
```

exception

Console output

```
Unhandled exception. IndexOutOfBoundsException
...
At Namespace.Program.ProcessAges
At Namespace.Program.Main
```

try/catch/finally syntax

```
try
{
  // guarded block
  execute_code();
}
catch( SomeSpecificExceptionType exn)
{
  Cȩŋşȩľê ẄsîŧȩĽîŋê(".....: " + exn.Message);

}
catch( Exception exn )
{  // catch everything else
  Cȩŋşȩľê ẄsîŧȩĽîŋê ("General error: " + exn.Message);

}
finally
{
  // closing files/connections
}
//remainder of containing method
```

Clean up and/or abort

Execute this whatever happens

Multiple exceptions example

```
try
{
        // some code
}
catch (IOException ex)
{
        Log(ex.ŞţǎçlŢsǎçê);
        throw new Exception("Cannot open file");
}
catch (SQLException ex)
{
        Log(ex.Message);
        throw new Exception("Database error!");
}
```

Inform the caller

Exception object has two properties to expose details
Message and **StackTrace**

Best practice guidance

Don't only catch class Exception

- Better to catch specific exceptions that you know how to handle

Don't need try { } catch { } in every method

- Makes code hard to read
- Again, only catch an exception if you have a meaningful way of handling it
  - Otherwise, just allow it to bubble up and let the caller deal with it

Only use exceptions for *exceptional* circumstances

- Don’t use for normal flow control

Be wary of just reporting back large error messages

- Might contain system information
- Might be better with “Sorry, we couldn’t find those credentials”

Use finally blocks to ensure resources are freed

- **C# uses exceptions to report errors**
  - Use `try` / `catch` / `finally` blocks to encapsulate such code
  - You can throw (or re-throw caught) exceptions
- **Unhandled exceptions will be caught by the CLR**

Review

LABS

Working with exceptions

Duration 30 minutes