Code Academy

Lecturer

Rokas Slaboševičius

# Exception handling

Data

# Today you will learn

01 What is exception handling?

02 Why do we need exception handling?

03 What are the types of exceptions?

# What is exception handling?

- In computing and computer programming, exception handling is the process of responding to the occurrence of exceptions - anomalous or exceptional conditions requiring special processing - during the execution of a program. In general, an exception breaks the normal flow of execution and executes a pre-registered exception handler; the details of how this is done depend on whether it is a hardware or software exception and how the software exception is implemented.

https://en.wikipedia.org/wiki/Exception_handling

# Exception handling keywords

Try - A code blog where you are expected to get a particular bug

Catch - A block of code that is run when an error is caught in the Try block

Finally - A block of code that is allowed at the end of a try/catch, whether or not the error was caught. E.g.: In the try block, we opened a file and need to close it whether an error occurred or not.

# Syntax

```
static void Main(string[] args)
{
    try
    {
        // statements causing exception
    }
    catch (AggregateException e1)
    {
        // error handling code
    }
    catch (ArgumentNullException e2)
    {
        // error handling code
    }
    catch (ArgumentException eN)
    {
        // error handling code
    }
    finally
    {
        // statements to be executed
    }

}
```

# Types of errors

Exceptions are defined as classes. The Exceptions class inherits directly or indirectly from the System.Exception class. Some of these are the System.ApplicationException and System.SystemException classes.

The System.ApplicationException class supports errors generated by the application, so exception classes created by the developer should inherit from this class.

System.SystemException is the parent class for all existing exception classes.

# Types of errors

| No. | Exception class and description |
|---|---|
| 1 | **System.IO.IOException:** handles errors related to input and output (input/output) |
| 2 | **System.IndexOutOfRangeException:** handles errors generated when a method tries to access an element in an array with an index too large for the array size. |
| 3 | **System.ArrayTypeMismatchException:** handles errors generated when an attempt is made to insert an element into an array whose data type is incorrect. |
| 4 | **System.NullReferenceException:** handles errors generated when a program tries to access a **null** element. |
| 5 | **System.DivideByZeroException:** handles errors when the program tries to divide by zero. |
| 6 | **System.InvalidCastException:** handles errors generated by a bad cast. |
| 7 | **System.OutOfMemoryException:** handles errors generated in case of memory failure. |
| 8 | **System.StackOverflowException:** handles errors when the stack is full |

# Error handling

Here we see a program that has a method Division with parameters int num1 and int num2 that it tries to divide and if it is caught trying to divide by zero then it goes to the catch(DivideByZeroException e) block and prints an error, then goes to finally and prints the result

```
internal class DivideNumbers
{
    int result = 0;

    public void Division(int num1, int num2)
    {
        try
        {
            result = num1 / num2;
        }
        catch (DivideByZeroException e)
        {
            Console.WriteLine($"Exception caught: {e}");
        }
        finally
        {
            Console.WriteLine($"Result: {result}");
        }
    }

    static void Main(string[] args)
    {
        var d = new DivideNumbers();
        d.Division(25, 0);
        Console.ReadKey();
    }
}
```

```
Microsoft Visual Studio Debug Console
Exception caught: System.DivideByZeroException: Attempted to divide by zero.
   at Sandbox.DivideNumbers.Division(Int32 num1, Int32 num2) in C:\Users\ITWORK\Documents\CodeAcademy\Sandbox\Sandbox\DivideNumbers.cs:line 17
Result: 0
```

# Creating your own exceptions

Here we see a class called TempIsZeroException which inherits from the Exception class. The other class is Temperature, which has an if clause that says if the temperature reaches 0 degrees our TempIsZeroException error is thrown.

```
public class TestTemperature
{
    static void Main(string[] args)
    {
        var temperature = new Temperature();
        try
        {
            temperature.ShowTemp();
        }
        catch (TempIsZeroException e)
        {
            Console.WriteLine($"TempIsZeroException: {e.Message}");
        }
        Console.ReadKey();
    }
}

public class TempIsZeroException : Exception
{
    public TempIsZeroException(string message) : base(message)
    {
    }
}
public class Temperature
{
    int temperature = 0;

    public void ShowTemp()
    {

        if (temperature == 0)
        {
            throw (new TempIsZeroException("Zero Temperature found"));
        }
        else
        {
            Console.WriteLine($"Temperature: {temperature}");
        }
    }
}
```

## Task 1

- Check the System.Convert.ToDouble documentation, what errors can occur when converting string to double?
- Write a program that will catch these errors and print to the console what the error was.

## Task 2

- Fix this code with the try/catch mechanism:

```
class GFG
{

    static void Main(string[] args)
    {
        // Declare an array of max index 4
        int[] arr = { 1, 2, 3, 4, 5 };

        // Display values of array elements
        for (int i = 0; i < arr.Length; i++)
        {
            Console.WriteLine(arr[i]);
        }

        // Try to access invalid index of array
        Console.WriteLine(arr[7]);
        // An exception is thrown upon executing
        // the above line
    }
}
```

## Task 3

- Anticipate possible errors in the code and print out the errors with try/catch:

```
static void Main(string[] args)
{
    int[] arr = { 19, 0, 75, 52 };
    // Try to generate an exception
    for (int i = 0; i < arr.Length; i++)
    {
        Console.WriteLine(arr[i] / arr[i + 1]);
    }
}
```

## Task 4

- Create a file reading class.
- What errors can occur when trying to read a file?
- Trying to find a file?
- Create mechanisms to protect against "breaking the programme"

# Useful information

**Exception handling**

https://www.tutorialspoint.com/csharp/csharp_exception_handling.htm

https://www.tutorialsteacher.com/csharp/csharp-exception-handling