5. Exception handling

H2. Geheugenmanagement, uitzonderingen en namespaces

Inleiding

- Exception → er is een uitzondering tijdens uitvoer opgetreden
- Exception handling → afhandelen van de uitzondering op een gecontroleerde manier tijdens uitvoer

- Voorbeelden:
  - Ongeldige invoer (bv. een letter i.p.v. een getal)
  - Netwerkproblemen (bv. Geen IP adres)
  - Schijfproblemen (bv. bestand niet gevonden)
  - Hardwareproblemen (bv. geen papier in printer)

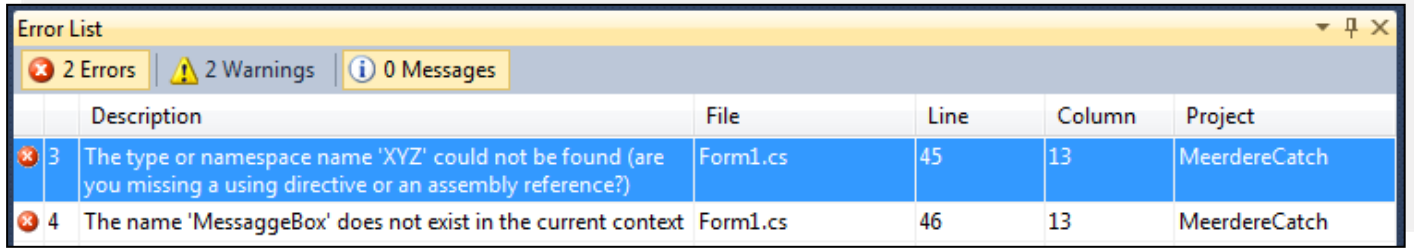
Inleiding

- Syntaxfouten

Heeft niets met exception handling te maken!

```
private void button1_Click(object sender, EventArgs e)
{
    XYZ newValue;
    MessaggeBox.Show("Op knop 1 gedrukt");
}
```

Bekijk je fouten in de “Error List”



Oldschool error handling

- Voorstelling programma met een “normale” werking. Als er nooit fouten zouden optreden is dit correct:

```
MethodeA();
MethodeB();
MethodeC();
```

Oldschool error handling

```
MethodeA();
if (MethodeA misliep)
{
// handel het methodeA-probleem af
}
else
{
    MethodeB();
    if (methodeB misliep)
    {
        // handel het methodeB-probleem af
    }
    else
    {
        MethodeC();
        if (methodeC misliep)
        {
            // handel het methodeC-probleem af
        }
    }
}
```

- Oude manier
- Ingewikkeld
- Niet overzichtelijk

Jargon

- Als een fout zich voordoet in het programma:
  - Wordt er door de runtime omgeving of door de methode zelf een speciaal object aangemaakt
  - Men zegt dat een exception opgegooid wordt
    (Engels: ***to throw***)
- Hoe afhandelen:
  - Op de gepaste locatie kan men deze exception opvangen (dus niet altijd vlak erna met een `if`)
    (Engels: ***to catch***)
- Sleutelwoorden: `throw`, `try`, `catch`, `finally`

`try-catch`: regels

- **Zet een `try` blok rond de code die je wil controleren op fouten**
- Als in een statement een exception optreedt, stopt de uitvoering onmiddellijk
- Er wordt gesprongen naar het `catch` blok, waar de exception afgehandeld wordt
- Als de exception niet wordt opgevangen, wordt deze doorgegooid naar de oproepende methode
  - Als ook deze ze niet kan opvangen → weer doorgooien
  - Uiteindelijk kom je uit bij de `Main()`

Een `try-catch` voorbeeld

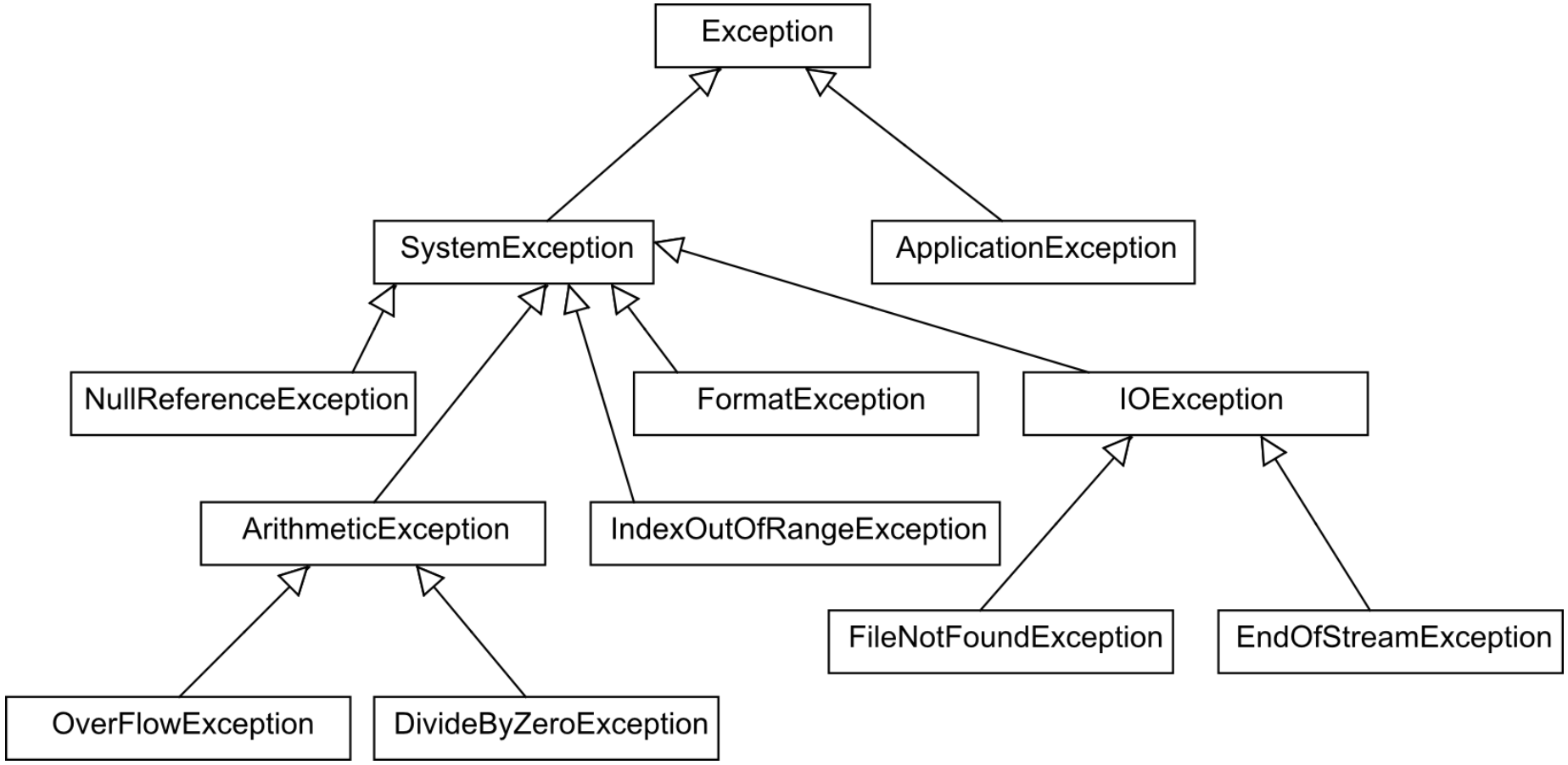
```
try
{
    string input = Console.ReadLine();
    int converted = Convert.ToInt32(input);
}
catch ()
{
    Console.WriteLine("Verkeerde invoer!");
}
```

Het Exception object

- Bevat nuttige informatie over de aard van de fout
- *Tip: lees deze informatie, dit zal je helpen bij het debuggen!*

- Properties
  - **`Message`** : kort bericht
  - **`StackTrace`** : hierarchie van methodes die geleid hebben tot de exception
  - Zie ook: **`Source, TargetSite, InnerException`**
- Methode
  - **`ToString():`** `string` voorstelling van deze exception

Classificatie (zie H6)



Meerdere exceptions in 1 `catch`

```
...
try
{
    SomeOperationWithIO();
}
catch (FileNotFoundException ex)
{
    Console.WriteLine("File not found, choose other file");
}
catch (EndOfStreamException ex)
{
     Console.WriteLine("End of stream: file corrupt");
}
...
```

Ofwel alle specifieke gevallen opvangen, zodat je een foutafhandeling hebt per geval

Meerdere exceptions in 1 `catch`

```
...
try
{
    SomeOperationWithIO();
}
catch (IOException ex)
{
    Console.WriteLine("IOException occurred.");
}
...
```

Ofwel 1 `catch` die alle subklassen van `IOException` behandelt. Dit is naar de gebruiker toe minder duidelijk.
(Bestand niet gevonden of corrupt?)