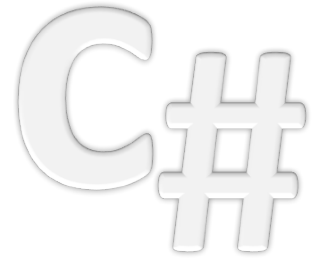




Exceptions

Hoofdstuk 17



In dit hoofdstuk ...

- Wat is een exception?
- Waarom zijn ze nuttig?
- De C# exception faciliteiten.

Overzicht

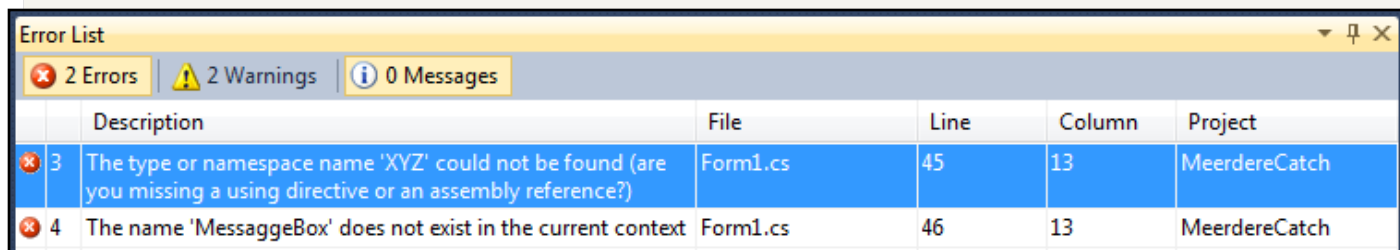
- Inleiding
- Try-Catch
- Combinatie Catch blokken
- Opgooien van exceptie
- Finally
- Zelf exceptie schrijven

Soorten fouten (1)

- Syntaxfouten

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

Bekijk je fouten in de “Error List”



The screenshot shows the 'Error List' window in Visual Studio. It has a yellow title bar and a toolbar with icons for expand, pin, and close. Below the toolbar, there are three tabs: '2 Errors' (selected), '2 Warnings', and '0 Messages'. The main area contains a table with two error entries.

	Description	File	Line	Column	Project
3	The type or namespace name 'XYZ' could not be found (are you missing a using directive or an assembly reference?)	Form1.cs	45	13	MeerdereCatch
4	The name 'MessageBox' does not exist in the current context	Form1.cs	46	13	MeerdereCatch

Soorten fouten (2)

- Logische fouten
 - Programma produceert foute resultaten
 - Niet altijd een crash tot gevolg!
 - Bv: oneindige lussen
 - Enkel op te lossen door uitvoerig te testen
- Run-time fouten
 - een onmogelijke operatie
 - Niet gedetecteerd door de compiler
 - Bv: deling door nul

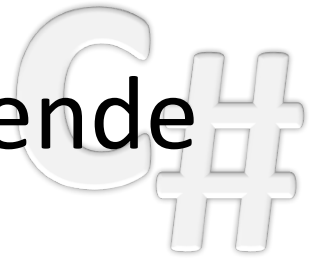
Exceptie

- Exception → er is een fout opgetreden
- Exception handling → afhandelen van de fout op een gecontroleerde manier
- Voorbeelden:
 - Ongeldige invoer (bv. een letter i.p.v. een getal)
 - Netwerkproblemen (bv. DNS fout)
 - Schijfproblemen (bv. bestand niet gevonden)
 - Hardwareproblemen (bv. geen papier in printer)

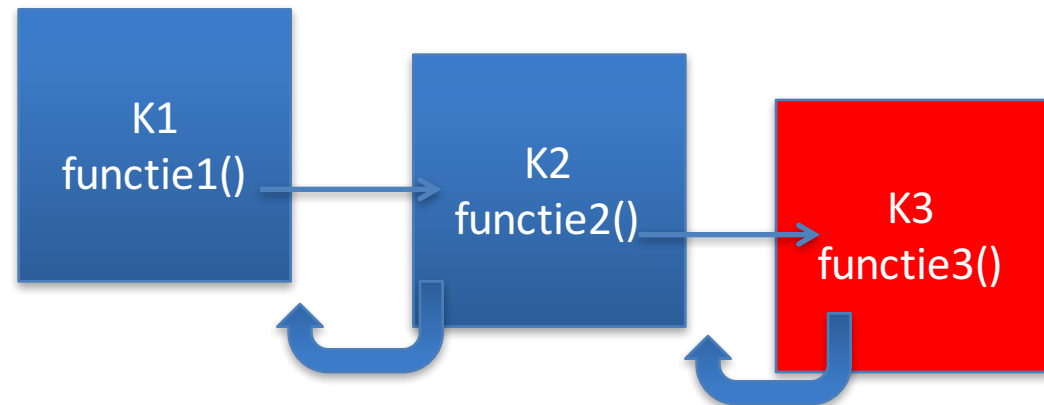
Problemen afhandelen

- Ongeldige invoer (bv. een letter i.p.v. getal)
- Netwerkproblemen (bv. DNS fout)
- Schijfproblemen (bv. bestand niet gevonden)
 - Stoppen van programma zou stom zijn
 - Gebruiker op hoogte brengen van probleem en hem nieuwe mogelijkheid bieden
- Hardwareproblemen (bv. geen papier in printer)
 - Gebruiker op hoogte brengen van probleem
 - Opties bieden om afdrukopdracht te verlaten of activiteit voor te zetten indien papier is aangevuld

Verschillende fouttypes en verschillende plaatsen voor afhandeling



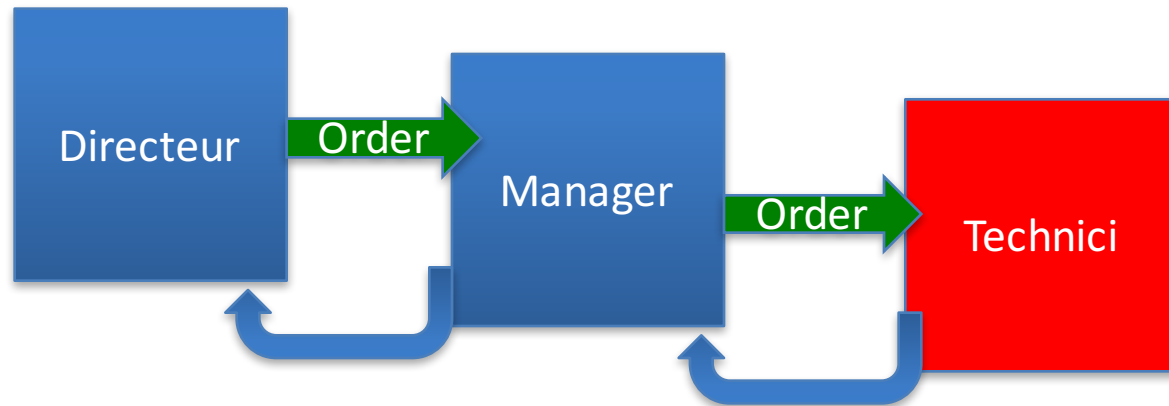
- Complexe systemen hebben hiërarchie van methoden (methoden roepen anderen aan)
- Sommige problemen kunnen lokaal (in methode zelf) afgehandeld worden, anderen misschien doorgegeven worden op hogere niveaus
 - Afh. van type fout



Verschillende fouttypes en afhandeling
op verschillende plaatsen

Analogie

- Organisatie Orders door hiërarchie.



- Bij fouten
 - Papier in printer is op. Technicus lost dit op. In zeldzame geval geen papier in organisatie, dan manager
 - Technicus struikelt over kabel en breekt been. Afhandeling door directeur door juridische kwesties

Ideale wereld

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

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

Werkelijkheid

```
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

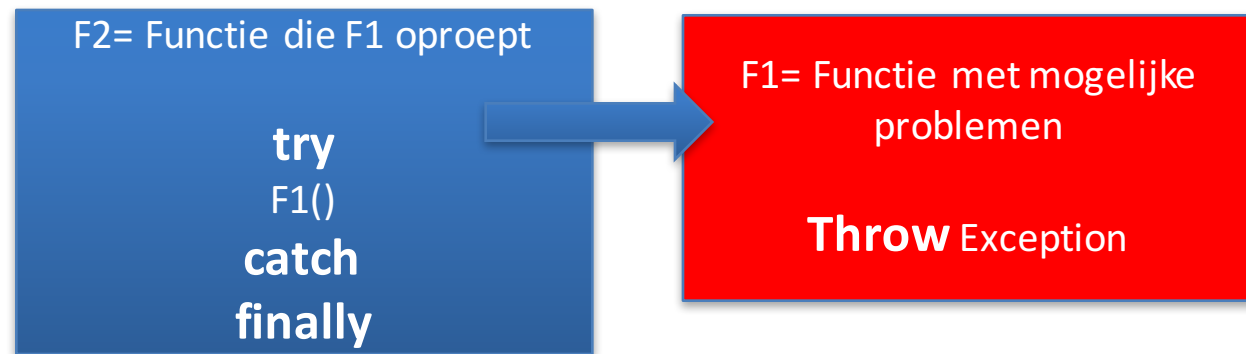
Niet overzichtelijk

17.2, p320

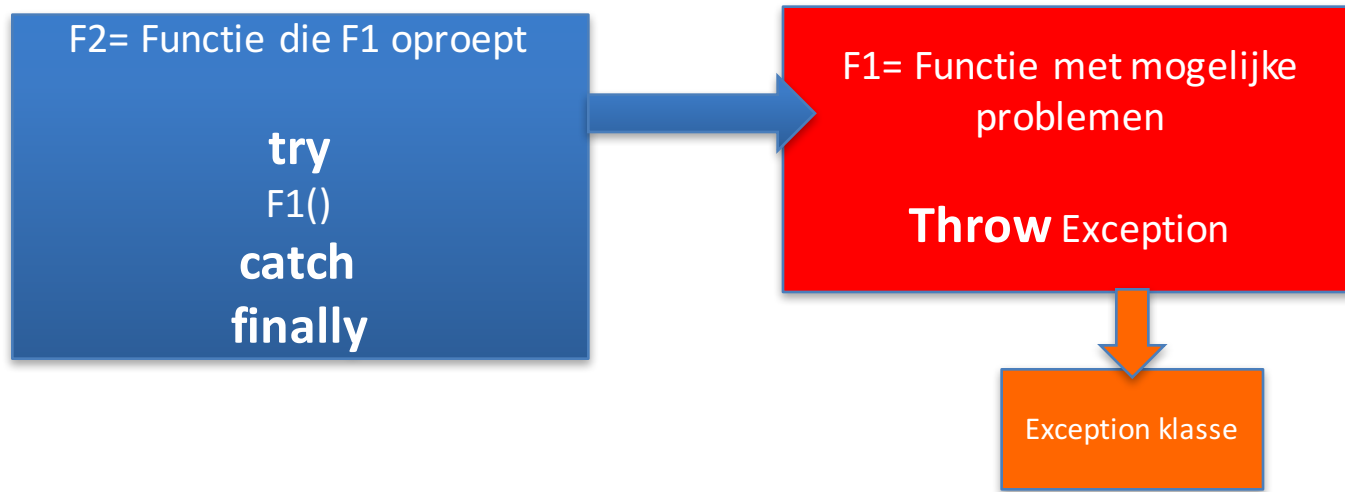
- Hoe kan methode retourneren dat het goed verlopen is of niet?
- Is dit altijd mogelijk?

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 (bij aanroep van methode die exception heeft opgegooid) kan men deze exception opvangen (dus niet altijd vlak erna met een *if*) (Engels: *to catch*)
- Sleutelwoorden:
throw,
try, catch,
finally



Terminologie visueel

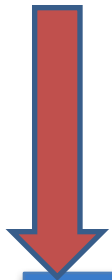


Overzicht

- Inleiding
- Try-Catch
- Combinatie Catch blokken
- Opgooien van exceptie
- Finally
- Zelf exceptie schrijven

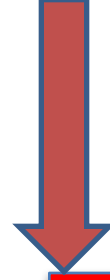
Typ in en wat krijg je?

- `double a = Double.Parse("blabla");`
- `double a = Convert.ToDouble("blabla");`



F2= Functie die F1
oproept

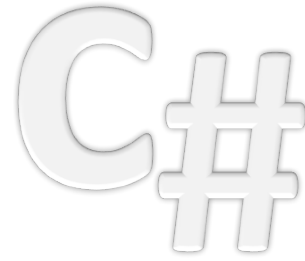
try
F1()
catch



F1= Functie met
mogelijke problemen

Throw Exception

Een try-catch voorbeeld



- Demo Exception Square

A screenshot of a Windows application window titled "Exception Square". The window has a light gray background and a blue border. It contains a label "Enter side:" followed by a text input field containing the value "2,5". Below the input field is a blue button labeled "Calculate". At the bottom of the window, the text "Area is 6,25 square units" is displayed.

A screenshot of the same "Exception Square" application window. The input field now contains the text "2XX5". The "Calculate" button is still present. At the bottom of the window, the text "Error in side, please re-enter." is displayed, indicating that an exception was caught and handled.

Een try-catch voorbeeld

```
private void button_Click(object sender, EventArgs e)
{
    double side;
    try
    {
        side = Double.Parse(textBox.Text);
        label.Content = String.Format("Area is {0} square units.",
                                     (side * side));
    }
    catch (FormatException exceptionObject)
    {
        label.Content = "Error in side, please re-enter.";
    }
}
```

Binnen het try blok kan zich een `FormatException` voordoen, nl: [Parse](#)

De exception wordt afgehandeld in het catch block

Alternatief voor
`Convert.ToDouble`

F2= Functie die F1 oproept

try
F1()
catch

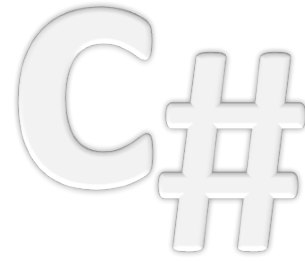
F1= Functie met
mogelijke
problemen

Throw Exception

try-catch: regels

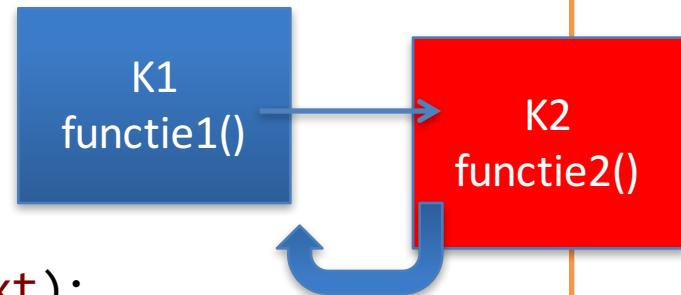
- Zet een try blok rond de code die je wil controleren op fouten
- Als in een statement een exception optreedt, stopt de uitvoering onmiddellijk
- Er wordt gesprongen naar het catch blok, waar de exception afgehandeld wordt
- Naam exceptionObject wordt niet gebruikt
 - Kan extra info over fout meegeven
- 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() procedure
→ stack trace

Een try-catch voorbeeld



```
private void calculateButton_Click(object sender, EventArgs e)
{
    try
    {
        DoCalc();
    }
    catch (FormatException exceptionObject)
    {
        label.Content = "Error in side, please re-enter.";
    }
}

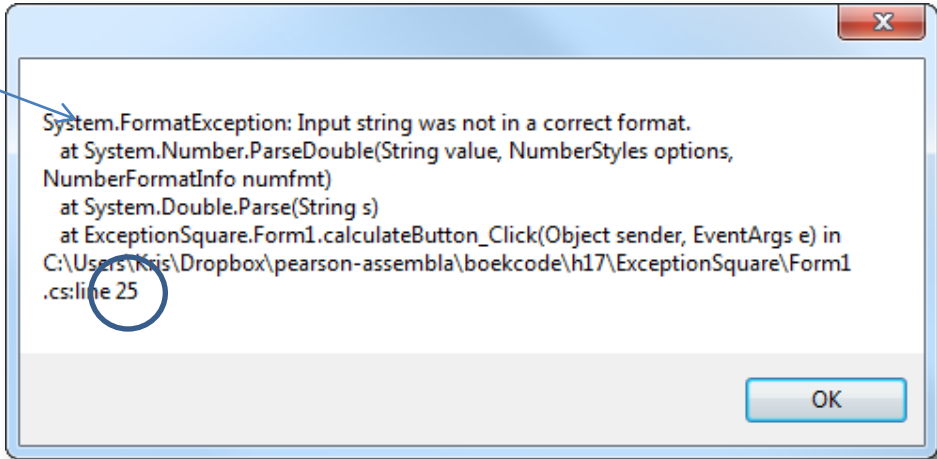
private void DoCalc()
{
    double side = Double.Parse(textBox.Text);
    label.Content = String.Format("Area is {0} square units.",
                                  (side * side));
}
```



Het exception object

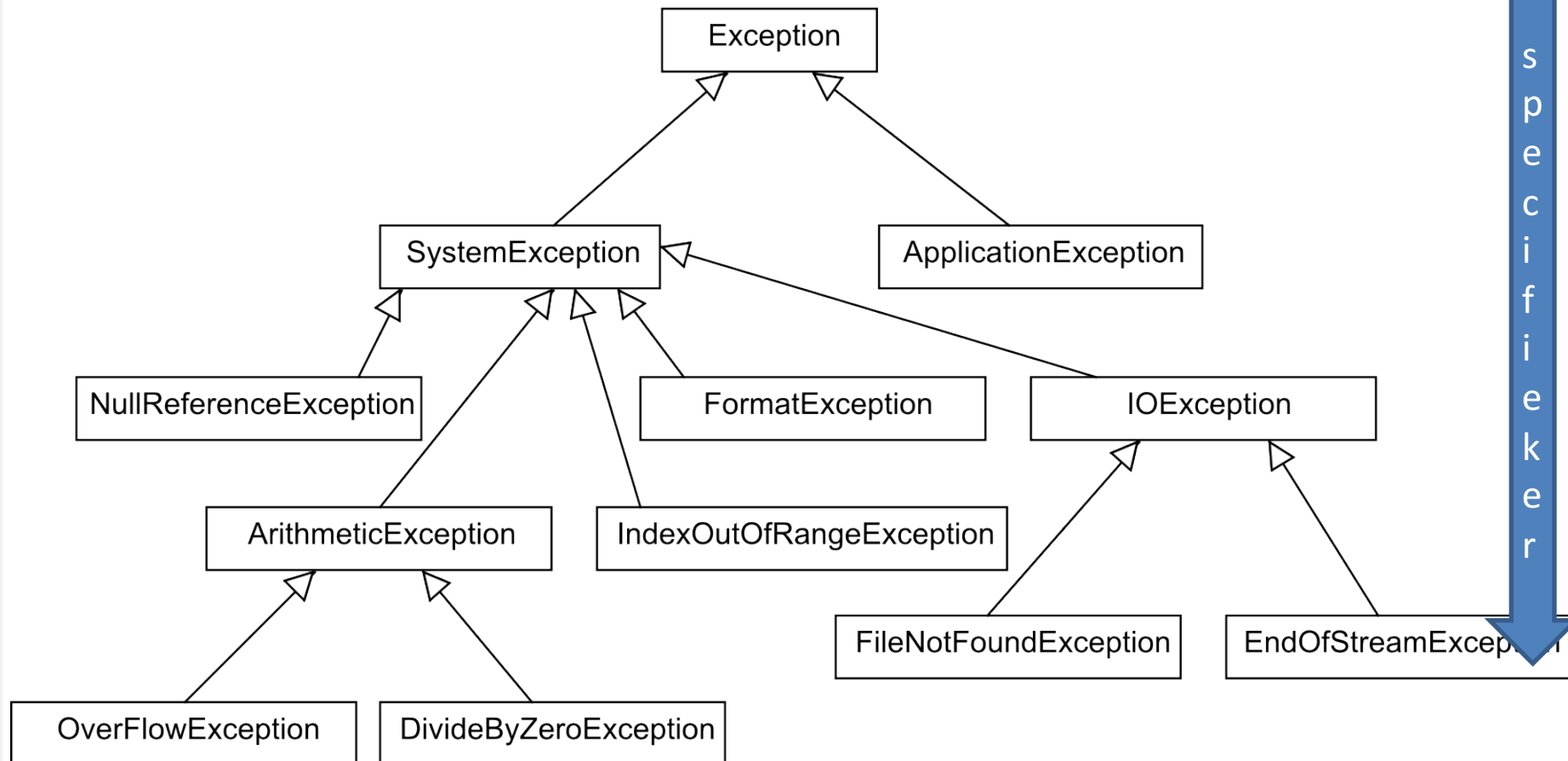
- Bevat nuttige informatie over de aard van de fout
- Tip: lees deze informatie, dit zal je helpen bij het debuggen!
- Properties
 - Message : kort bericht
 - StackTrace : hierarchie van methodes die geleid hebben tot de exception
 - Zie ook: Source, TargetSite, InnerException
- Methode
 - ToString(): string voorstelling van deze exception

```
MessageBox.Show(exceptionObject.Message);  
MessageBox.Show(exceptionObject.ToString());
```



System.FormatException: Input string was not in a correct format.
at System.Number.ParseDouble(String value, NumberStyles options,
NumberFormatInfo numfmt)
at System.Double.Parse(String s)
at ExceptionSquare.Form1.calculateButton_Click(Object sender, EventArgs e) in
C:\Users\Kris\Dropbox\pearson-assembla\boekcode\h17\ExceptionSquare\Form1
.cs:line 25

Classificatie



Opmerking

- Nog veel meer exceptie klassen
- Kennen van namen van excepties niet voldoende
 - Bron moet gekend zijn
- Voor elke methode die je gebruikt, documentatie bekijken
 - Bevat wat methode doet
 - Argumenten
 - Excepties die worden opgegooid

Overzicht

- Inleiding
- Try-Catch
- Combinatie Catch blokken
- Opgooien van exceptie
- Finally
- Zelf exceptie schrijven

Meerdere exceptions in 1 catch

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

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

Meerdere exceptions in 1 catch

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

```
...  
try  
{  
    SomeOperationWithIO();  
}  
catch (IOException ex)  
{  
    MessageBox.Show("IOException occurred.");  
}  
...
```

Combinatie van catch blokken

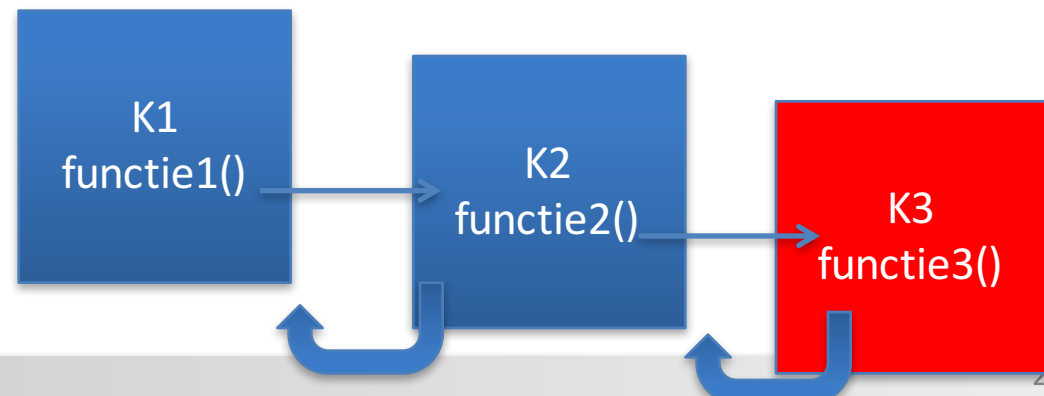
```
int bottom;
int top = 100;
try
{
    bottom = Int32.Parse(textBox.Text);
    label.Content = String.Format("Dividing into 100 gives {0}",
                                  (100 / bottom));
}
catch (DivideByZeroException exceptionObject)
{
    label.Content = "Error - zero: re-enter data.";
}
catch (FormatException exceptionObject)
{
    label.Content = "Error in number: re-enter.";
}
catch (SystemException exceptionObject)
{
    MessageBox.Show(exceptionObject.ToString());
}
```

Vang elk type van exception afzonderlijk op

Regel: hoe algemener de exception (SystemException), hoe later opvangen in het try-catch statement (waarom?)

Samenvatting

- Methode1()
 Methode2()
 Methode3() → Exception treedt op
 - Uitvoering van Methode3() wordt onmiddellijk gestopt
 - Als er een catch statement is, die deze Exception (of een superklasse ervan) opvangt, wordt deze uitgevoerd. De uitvoering gaat vervolgens verder na het try-catch blok.
 - Als er geen catch statement is, gaat de Exception naar de oproeper (Methode2()), indien deze ook geen catch statement heeft naar Methode1(), enz. Als er helemaal geen catch wordt gevonden, breekt het programma af met een foutmelding.
 - Exception propagation (voortplanting)



Overzicht

- Inleiding
- Try-Catch
- Combinatie Catch blokken
- Opgooien van exceptie
- Finally
- Zelf exceptie schrijven

Opgooien: een inleiding

```
private int WordToNumber(string word)
{
    int result = 0;
    if (word == "ten")
    {
        result = 10;
    }
    else if (word == "hundred")
    {
        result = 100;
    }
    else if (word == "thousand")
    {
        result = 1000;
    }
    else
    {
        throw new FormatException("Wrong input: " + word);
    }
    return result;
}
```

F2= Functie die F1
oproept

try
F1()
catch

F1= Functie met
mogelijke
problemen

Throw Exception

`throw new FormatException("Wrong input: " + word);`

Opgooien: een inleiding

```
private void convertButton_Click(object sender, EventArgs e)
{
    try
    {
        MessageBox.Show(Convert.ToString(WordToNumber("hXndred")));
    }
    catch (FormatException exceptionObject)
    {
        MessageBox.Show(exceptionObject.Message);
    }
}
```

F2= Functie die F1
oproept

try
F1()
catch

F1= Functie met
mogelijke
problemen

Throw Exception

Hoe afhandelen

- Zinnvolle foutmelding naar de gebruiker toe, eventueel vragen om nieuwe invoer
 - Opgelet, niet altijd gebruiker die invoer doet, invoer kan ook komen uit database
- Bij waarschijnlijke bugs, exceptions loggen naar bestanden of Event logs
- ***Nooit lege catch statements schrijven om exceptions te “verbergen”***

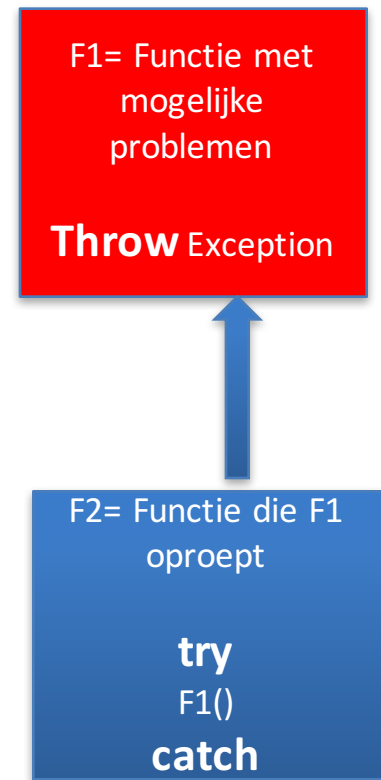
Samengevat

- Methode A die exceptie opgooit
throw

WordNumber(...)

- Methode B die Methode A aanroept
en die exceptie opvangt
try-catch

ConvertButton_Click(...)



Afhandelen

- Niet alle exceptions afhandelen, soms gewoon verkeerd programma en beter debuggen
- Vb `IndexOutOfRangeException`

```
int[] a = new int[10];  
for (int n=0; n<=10,n++) a[n]=25;
```

Overzicht

- Inleiding
- Try-Catch
- Combinatie Catch blokken
- Opgooien van exceptie
- Finally
- Zelf exceptie schrijven

finally

```
DatabaseConnection resource = ...;
try
{
    resource.Open();
    voer queries uit naar de database
}
catch (SQLException exceptionObject)
{
    toon foutmelding
}
finally
{
    // ruim de connectie op
    resource.Close();
}
```

Finally blok

- Wordt altijd uitgevoerd
 - Zelfs bij return in try
 - Zelfs bij niet opgevangen catch
- Achteraan try-catch blok

Vergelijk

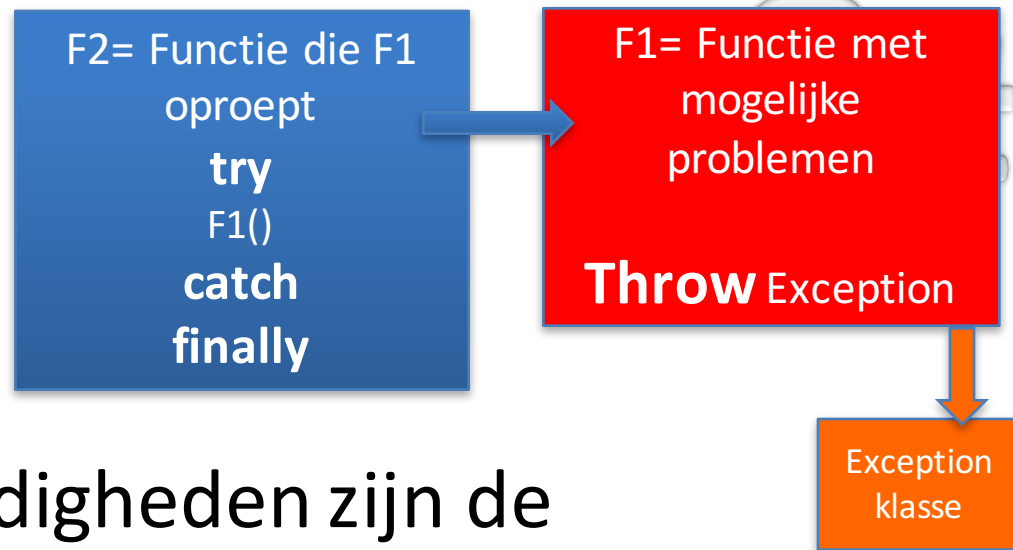
```
try {  
    Protect one or more  
    statements here.  
}  
catch (Exception e) {  
    Report and recover  
    from the exception  
    here.  
}  
finally {  
    Perform any actions  
    here common to  
    whether or not  
    an exception is  
    thrown.  
}
```

```
try {  
    Protect one or more  
    statements here.  
}  
catch (Exception e) {  
    Report and recover  
    from the exception  
    here.  
}  
  
Perform any actions  
here common to whether  
or not an exception is  
thrown.
```

Overzicht

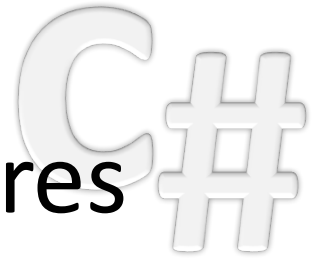
- Inleiding
- Try-Catch
- Combinatie Catch blokken
- Opgooien van exceptie
- Finally
- Zelf exceptie schrijven

Zelf exception schrijven



- In bepaalde omstandigheden zijn de ingebouwde exceptions van het .NET framework onvoldoende
- Bijvoorbeeld, je wil fouten die voor een bepaalde toepassing specifiek zijn, op dezelfde manier met exceptions afhandelen
- Hoe?
 - Schrijf zelf een klasse die overerft van `ApplicationException`

Exception voor ongeldig emailadres



- Bv ongeldig als er geen @ en geen . na @ komt
 - Kan met `FormatException`
 - Eigen klasse is echter duidelijker. Richtlijnen kunnen specifiek meegegeven worden.

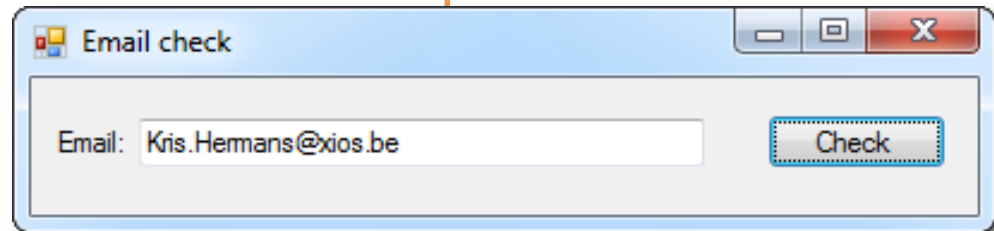
Zelf exceptions schrijven

```
public class InvalidEmailException
    : ApplicationException
{
    public InvalidEmailException(string message)
        : base(message)
    { }
}
```

Voorzie steeds een constructor die een message parameter doorgeeft. Dit is de eigenlijke foutmelding

Zelf exceptions schrijven

```
private void checkButton_Click(object sender, EventArgs e)
{
    try
    {
        CheckAddress(textBox.Text);
    }
    catch (InvalidEmailException ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```



```
private void CheckAddress(string email)
{
    if (!email.Contains('@'))
    {
        throw new InvalidEmailException(email +
                                         " does not contain @-sign!");
    }
    // other validation rules
    ...
}
```

F2= Functie die F1 oproept

try
F1()
catch

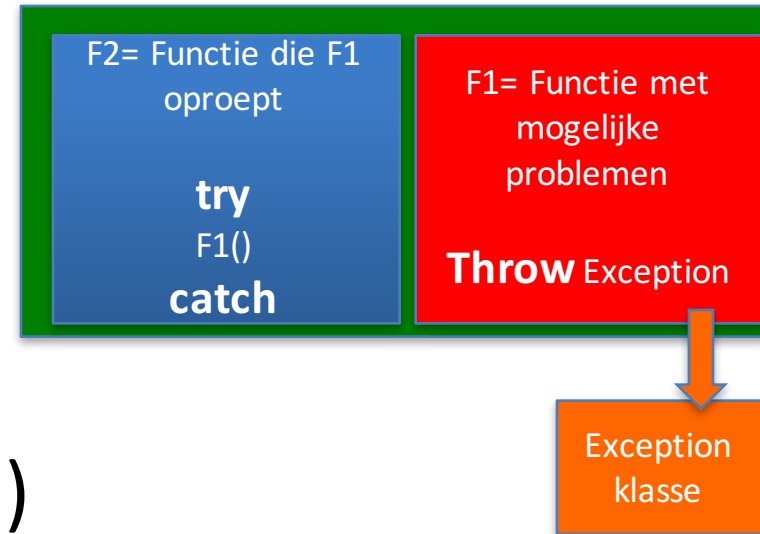
F1= Functie met
mogelijke
problemen

Throw Exception

Exception
klasse

Eigen exceptie schrijven

1. Nieuwe exceptie klasse aanmaken
InvalidEmailException
2. Methode F1 schrijven die nieuwe exceptie opgooit (Throw)
CheckAdress()
3. Methode F2 die methode F1 aanroept en die exceptie opvangt (Try-Catch)
CheckButtonClick()



Samenvatting

Overzicht

- Inleiding
- Try-Catch
- Combinatie Catch blokken
- Opgooien van exceptie
- Finally
- Zelf exceptie schrijven

Oefening 17.1

- Deling van 2 gehele getallen
Voorzie 2 tekstvakken voor invoer van 2 getallen. Laat de resultaten van de 2 delingen zien in 2 labels als je op knop duwt. Voeg de volgende exceptionhandelingen toe:
 - `FormatException`
 - `DivideByZeroException`

Oefening 17.2

- Methode om kwadratische vergelijking op te lossen
- `LosOp(a,b,c,opl1,opl2);`
Opl1 en Opl2 met referenties retourneren
Exception
 - Als Discriminant negatief is, dan `ArithmeticException` met gepaste foutmelding
 - Als $a=0$, dan `ArithmeticException` met gepaste foutmelding
 - Als a,b,c geen doubles zijn, dan `FormatException`

Oefening 17.7

- 1. BankException:ApplicationException
- BankRekening
 - Property saldo met get
 - Storting
 - 2. Gooit exception bij saldo boven 2500
 - Opname
 - 2. Gooit exception bij waarde onder 0
- GUI
 - Knop voor saldo
 - Textbox voor bedrag (+/-)
 - Label voor saldo
- Code achter GUI
 - Event bij knopklik
 - 3. Try-catch: FormatException, BankException