

# C# Essentials

Naamconventies en documentatie

Lector: Tom Quareme

# Naamconventies in C#

- **HOUD HIER REKENING MEE OP EXAMEN!!!**
- We volgen de C# coderingsstandaarden.
  - Naamgeving van variabelen, methods, classes,...
  - **camelCase:**
    - 1ste letter van het 1ste woord krijgt kleine letter.
    - 1ste letter van elk volgend woord is een hoofdletter.
    - Enkel gebruikt voor lokale variabelen/constanten, private variabelen/constanten en parameters van methods.
  - **PascalCase:**
    - 1ste letter van elk woord start met hoofdletter.
    - Wordt voor alle overige dingen gebruikt.

# Naamconventies in C#

- **HOUD HIER REKENING MEE OP EXAMEN!!!**

<i>Identifier</i>	<i>Casing</i>	<i>Voorbeeld</i>
<b>Public variabele</b>	PascalCase	public string <b>MijnNaam</b> ;
<b>Private variabele</b>	camelCase	private string <b>mijnNaam</b> ;
<b>Public const</b>	PascalCase	public const int <b>Minimum</b> = 0;
<b>Private const</b>	camelCase	private const int <b>minimum</b> = 0;
<b>Lokale variabele/constante</b>	camelCase	int <b>mijnVariabele</b> = 10;
<b>Enum waarde</b>	PascalCase	public enum FileMode { <b>Append</b> , ...}
<b>Parameter van method</b>	camelCase	private int ZetOm(string <b>waarde</b> ) { ... }

# Naamconventies in C#

- **HOUD HIER REKENING MEE OP EXAMEN!!!**

<i>Identifier</i>	<i>Casing</i>	<i>Voorbeeld</i>
<b>Namespace</b>	PascalCase	namespace <b>System.Text</b> { ... }
<b>Class</b>	PascalCase	public class <b>GameEngine</b> { ... }
<b>Interface</b>	PascalCase	public interface <b>IEnumerable</b> { ... }
<b>Method</b>	PascalCase	private int <b>BerekenLoon()</b> { ... }
<b>Property</b>	PascalCase	public string <b>EersteNaam</b> { get; set; }
<b>Event</b>	PascalCase	public event EventHandler <b>Click</b> ;

# Documentatie/Commentaar

- Code verduidelijken.
  - Complexe of verwarrende code duidelijker uitleggen.
  - XML tags in C# commentaar om nadien documentatie te genereren.
- Tips:
  - Zet geen onnodige nutteloze commentaar als de code al genoeg zegt!
  - Beter **duidelijke (langere) naam** dan korte naam en veel commentaar.
  - Commentaar moet korter zijn dan de code zelf!
  - Vraag je af wat andere programmeur moeilijk zou vinden bij lezen van je code.

# XML in commentaar

- In C#: XML brengt structuur in commentaar.
  - XML comments worden getoond bij hoveren over code in Visual Studio.
  - Niet mogelijk voor namespaces.
- Soorten XML tags:
  - <summary>
  - <para>
  - <returns>
  - <param>

# <summary> </summary>

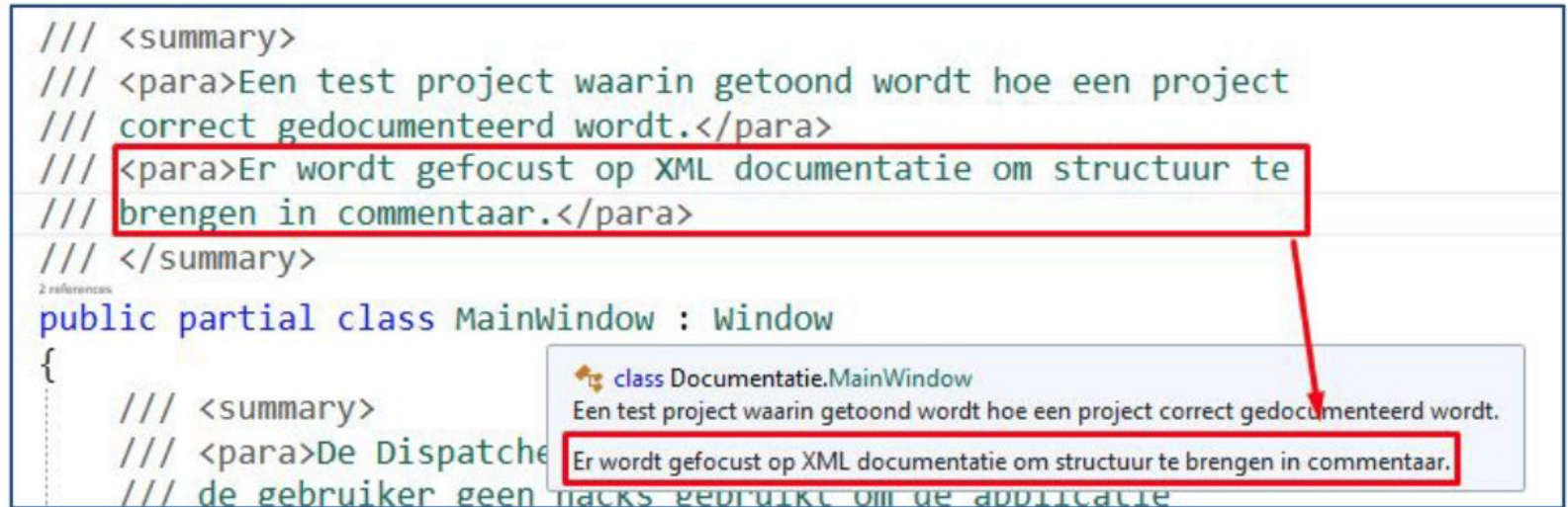
- Om uitleg toe te voegen aan een class, method,...
- Gebruik eerst ook 3x "/" (///)!

```
/// -----  
/// Author:      Sander De Puydt  
/// Create Date: 20/02/2002  
/// Description: Documentatie demo project  
/// -----  
/// <summary>  
/// Een test project waarin getoond wordt hoe een project  
/// correct gedocumenteerd wordt.  
/// </summary>  
2 references  
public partial class MainWindow : Window  
{  
    0 references  
    public MainWindow(  
    {  
        InitializeComponent();  
    }  
}
```

# <para> </para>

- Om paragrafen in commentaar te schrijven.
- Komt binnen <summary> </summary>.

```
/// <summary>
/// <para>Een test project waarin getoond wordt hoe een project
/// correct gedocumenteerd wordt.</para>
/// <para>Er wordt gefocust op XML documentatie om structuur te
/// brengen in commentaar.</para>
/// </summary>
2 references
public partial class MainWindow : Window
{
    /// <summary>
    /// <para>De Dispatcher
    /// de gebruiker geen hacks gebruikt om de applicatie
```



class Documentatie.MainWindow  
Een test project waarin getoond wordt hoe een project correct gedocumenteerd wordt.  
Er wordt gefocust op XML documentatie om structuur te brengen in commentaar.



## <returns> </returns>

- Komt onder de <summary></summary>.
- Geeft aan wat voor waarde een methode teruggeeft.

```
/// <summary>
/// Geeft een proces een threat score. Hoe hoger
/// de score is, hoe gevaarlijker het process is
/// voor de applicatie.
/// </summary>
/// <returns>Een threat score tussen 0 (ongevaarlijk)
/// en 10 (gevaarlijk).</returns>
1 reference
private double GenerateThreatScore(string processNaam)
{
    if (processNaam == null)
    {
        return 1;
    }
    Random rand = new Random();
    return rand.Next(0,11);
}
```



# <param> </param>

- Geeft uitleg over parameters van methode.
- Zorg dat de parameter name overeenkomt met die van de method!
- <param **name="parameterNaam"**>Uitleg over parameter. </param>

```
/// <summary>
/// Geeft een proces een threat score. Hoe hoger
/// de score is, hoe gevaarlijker het process is
/// voor de applicatie.
/// </summary>
/// <param name="procesNaam">Naam van een actief
/// proces op hetzelfde systeem als de applicatie</param>
/// <returns>Een threat score tussen 0 (ongevaarlijk)
/// en 10 (gevaarlijk).</returns>
1 reference
private double GenerateThreatScore(string procesNaam)
{
    if (procesNaam.Equals("Hack"))
    {
        return 10;
    }
}
```

(parameter) string procesNaam  
Naam van een actief proces op hetzelfde systeem als de applicatie