# Code Conventies

C#

September 2016

## Revisiegeschiedenis

| Versie | Datum | Auteur | Omschrijving |
|---|---|---|---|
| **1** | 27/09/2016 | Kris Hermans | Eerste versie |
| **2** | 28/09/2016 | Kris Hermans | Extra voorbeeld CC-06 |

## Introductie

Programmeren volgens strikte richtlijnen heeft een aantal voordelen:

- de code is leesbaar voor elk teamlid;
- daardoor wordt de code ook beter onderhoudbaar;
- en geeft de volledige code een professionele indruk;

## Definities

**Pascal Casing**
De eerste letter van elk woord is met een hoofdletter geschreven en andere letters staan in kleine letters.

```
NewBachelorStudent
```

**Camel Casing**
De eerste letter van elk woord, behalve het eerste woord, is met een hoofdletter geschreven en andere letters staan in kleine letters.

```
newBachelorStudent
```

## Conventies

### CC-01 Gebruik Pascal Casing voor klassenamen

```
public class BachelorStudent
{
    ...
}
```

### CC-02 Gebruik Pascal Casing voor methodenamen

```
public void DrawLogo(int xPosition, int yPosition)
{
    ...
}
```

## CC-03 Gebruik Camel Casing voor variabelen en methode parameters

```
public void DrawLogo(int xPosition, int yPosition)
{
    Color randomColor = ...;
}
```

## CC-04 Gebruik de letter I als prefix voor interfaces, samen met Camel Casing

```
public interface IColoredBalloon
{
    ...
}
```

## CC-05 Gebruik geen Hongaarse notatie om variabelen te benoemen

Vroeger was het de gewoonte om een variabele benoemen met een prefix waaruit het type duidelijk wordt, bijvoorbeeld:

```
int nAge;     //VERBODEN
string sName; //VERBODEN
```

Soms gebruikt men ook een prefix m_ om aan te geven dat het om een membervariabele gaat. Ook dit raden we af.

```
int m_age;    //VERBODEN
```

Het is wel toegelaten, maar niet verplicht, om membervariabelen te prefixen met een underscore (_), zoals in:

```
int _age;     // OK
```

Wees dan wel consequent zodat alle membervariabelen dit patroon volgen.

## CC-06 Gebruik betekenisvolle namen voor klassen, methoden en variabelen. Geen afkortingen!

In orde:

```
string address;
int salary;

public class DeviceInformationDisplay
{
    public void UpdateSensors()
    {
        ...
    }
}
```

Niet in orde:

```
string addr;
int sal;

public class DevInfDisp
{
    public void UpdSens()
    {
        ...
    }
}
```

## CC-07 Gebruik geen variabelenamen die bestaan uit één karakter

Geen **i, n, s** enz. Maar wel **index**, **temp**, enz. Een uitzondering kan voor lusvariabelen:

```
for ( int i = 0; i < count; i++ )
{
      ...
}
```

## CC-08 Instanties van User Interface (UI) componenten benoem je door een betekenisvolle naam, gevolgd door de klassenaam van de betreffende component.

**C#**

```
Button cancelButton = new Button();
```

**XAML**

```
<Button x:Name="cancelButton" ...></Button>
```

Wanneer de door Visual Studio gegenereerde code in conflict is met de regels van methodenamen, hoef je dat niet te veranderen, bijvoorbeeld:

**XAML**

```
<Button x:Name="cancelButton" Click="cancelButton_Click"></Button>
```

**C#**

```
private void cancelButton_Click(object sender, RoutedEventArgs args)
{
    // Deze methode kan door Visual Studio gegenereerd worden
}
```

## CC-09 Zet maximaal één publieke klasse per bestand

Het bestand krijgt dezelfde naam als de klasse met extensie .cs

```
BachelorStudent.cs
```

### CC-10 Plaats accolades { } onder mekaar, netjes uitgelijnd

```
if (filled)
{
    rectangle.Fill = brush;
}
else
{
    rectangle.Fill = null;
}
```

en dus niet:

```
if (filled) {
    rectangle.Fill = brush;
} else {
    rectangle.Fill = null;
}
```

## Referenties


[1] C# Coding Conventions, https://msdn.microsoft.com/en-us/library/ff926074.aspx

[2] C# Coding Standards & Best Practices, http://www.dotnetspider.com/tutorials/BestPractices.aspx

[3] General naming conventions by Microsoft, https://msdn.microsoft.com/en-us/library/ms229045.aspx

[4] Regarding naming controls on Stackoverflow, http://stackoverflow.com/questions/440163/textboxemployeename-vs-employeenametextbox