# Code Conventies

## Java

September 2016

## Revisiegeschiedenis

| Versie | Datum | Auteur | Omschrijving |
|---|---|---|---|
| **1** | 28/09/2016 | Tom Schuyten | Eerste versie |
| **1.1** | 30/09/2016 | Tom Schuyten<br>Bart Clijsner | Correcties |

## 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 Camel 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 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
```

Wees dan wel consequent zodat alle membervariabelen dit patroon volgen.

## CC-05 Gebruik betekenisvolle namen voor klassen, methoden en variabelen. Geen afkortingen!

In orde:

```
String address;
int salary;
```

Niet in orde:

```
String addr;
int sal;
```

## CC-06 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-07 Instanties van User Interface (UI) componenten benoem je door een betekenisvolle naam, gevolgd door de klassenaam van de betreffende component.

**Java**

```
JButton cancelButton = new JButton();
```

## CC-08 Gebruik hoofdletters en _ als scheidingsteken tussen woorden voor constante variabelen:

```
static final int MIN_WIDTH = 4;
static final int MAX_WIDTH = 8;
```

### CC-09 Plaats accolades { } na en onder het statement, netjes uitgelijnd

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

en dus niet:

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

### CC-10 Gebruik een aparte lijn per declaratie voor variabelen van hetzelfde type

```
int level;
int size;
```

en dus niet:

```
int level, size;
```

### CC-11 Verkies maximaal één publieke klasse per bestand

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

```
BachelorStudent.java
```

## Referenties


[1] Java Coding Conventions, http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

[2] Clean Code: A Handbook of Agile Software Craftsmanship, https://github.com/apoterenko/software-development-ebooks/blob/master/%5BClean%20Code%20A%20Handbook%20of%20Agile%20Software%20Craftsmanship%20Kindle%20Edition%20by%20Robert%20C.%20Martin%20-%202009%5D.pdf

[3] Github repo: https://github.com/wgroeneveld/cleancode-course

[4] Slides: http://www.prato-services.eu/cleancode/index.html#/