rkCodingStandards

C++ Java PHP

René Knipschild
Custom Software Development
www.rkcsd.com
Postfach 1468
D-34484 Korbach

08. Januar 2015

1 Allgemeines

Die **rkCodingStandards** sind Richtlinien, nach denen René Knipschild Applikationen in C++, Java und PHP entwickelt. Sie gelten für alle Anwendungen, die nicht zu Testzwecken und nicht für Dritte, die andere Standards wünschen, geschrieben werden.

Für alle Sprachen und Standards gilt: Eingerückt wird per Tabulator. Korrekte Einrückung ist Pflicht. Abgesehen von C++, wo Neuzeilen am Dateiende eingefügt werden müssen, dürfen bei Java und PHP keine Neuzeilen am Dateiende eingefügt werden. Sämtliche Leerzeichen am Zeilenende sind zu entfernen. Alle Dateien werden mit der Kodierung UTF-8 gespeichert. Zeilenumbrüche werden, wie unter Linux-Systemen üblich, ausschließlich mit einem „Line Feed“ (LF), und nicht etwa mit einem „Carriage Return“ (CR), wie unter Macintosh üblich, und auch nicht mit einer Kombination aus beidem (CRLF), wie unter Windows üblich, vorgenommen. Die Verwendung von Kommentaren ist nicht erwünscht, bis auf einen verpflichtenden mehrzeiligen Kommentar am Dateianfang, der Aufschluss über das Projekt, den Autor, die **Aufgabennummer des Bugtrackers** (FS#XXX) und die Version der Datei gibt. Innerhalb dieses Kommentars können auch Notizen hinzugefügt werden. Kommentare werden grundsätzlich in **Englisch** verfasst.

```
/*
* ################################################
* #                                              #
* # ProjectName                                  #
* # A short project description                  #
* #                                              #
* # Copyright (C) by Ren\'{e} Knipschild         #
* #                                              #
* # rk@reneknipschild.net                        #
* # www.reneknipschild.net                       #
* #                                              #
* ################################################
*
* File: FileName - PackageName
* Version: 1.0.0
* Last modified: 1970/01/01 00:00 GMT
* Author: DeeDee0815
*
* ===Notes=======================================
* There are currently no notes.
* ================================================
*/
```

Ansonsten werden Kommentare nur verwendet, wenn keine andere Mög-

lichkeit besteht, den Code später noch nachvollziehen zu können, wie im Falle von beispielsweise TODO- Kommentaren. Pakete, Klassen, Methoden, Variablen, Dateien und alle anderen Dinge, die sonst noch benannt werden müssten, werden genau wie Kommentare grundsätzlich auf Englisch eindeutig benannt. Konstantenbenamungen bestehen in jeder Sprache ausschließlich aus Großbuchstaben und Unterstrichen.

Die Ordnerstruktur eines Projekts ist wie folgt aufgebaut:

```
1 <PFAD_ZUM_PROJEKT>/<PROJEKT>/trunk
2 <PFAD_ZUM_PROJEKT>/<PROJEKT>/tags
3 <PFAD_ZUM_PROJEKT>/<PROJEKT>/branches
```

Die Bezeichnung trunk steht für den Hauptast des Projektes. Das Verzeichnis branches steht für eventuelle Entwicklungszweige, das Verzeichnis tags dient der Speicherung von Entwicklungsständen.

2 C++

Am Anfang einer **C++-Datei** finden alphabetisch sortiert die Includes statt. Danach erfolgt die Namespace-Initialisierung: ebenfalls in alphabetischer Reihenfolge. Grundsätzlich werden einzelne Leerzeilen zur Übersichtsverbesserung eingefügt. Mit derartigen Einfügungen sollte aber nicht all zu großzügig umgegangen werden. Auch ist mehr als eine Leerzeile hintereinander nicht erlaubt.

```
#include <iostream>

using namespace std;

int main(int argc, char** argv)
{
  cout << "output";
  return 0;
}
```

Die öffnende (geschweifte) Klammer einer **Klasse** steht in jedem Fall in einer neuen Zeile. Klassennamen beginnen mit einem Großbuchstaben, gegebenenfalls befindet sich davor noch ein klein geschriebenes Kürzel wie zum Beispiel „rk“, und enthaltenen den enthaltenen Begriffen entsprechend Unterstriche und gegebenenfalls weitere Großbuchstaben.

```
class rkSample_MainClass
{
  int rkSample_DefaultMethod(void)
  {
    return 0;
  }
};
```

Die öffnende (geschweifte) Klammer einer **Methodendeklaration** steht in jedem Fall in einer neuen Zeile. Methodennamen beginnen mit einem Großbuchstaben, gegebenenfalls befindet sich davor noch ein klein geschriebenes Kürzel wie zum Beispiel „rk“, und enthaltenen den enthaltenen Begriffen entsprechend Unterstriche und gegebenenfalls weitere Großbuchstaben. Methoden müssen ohne Leerzeichen zwischen dem Methodennamen und der öffnenden (runden) Klammer deklariert werden.
Methoden müssen ohne Leerzeichen zwischen dem Methodennamen und der öffnenden (runden) Klammer aufgerufen werden. Hinter dem Trennungskomma zwischen zwei Methodenparametern muss ein Leerzeichen stehen. Zwischen der abschließenden (runden) Klammer und dem Semikolon darf kein Leerzeichen stehen.

```
int Sample_MainMethod(arg, arg)
```

```
{
    return 0;
}

Sample_MainMethod(arg, arg);
```

Bei der **Wertzuweisung einer Variable** steht grundsätzlich ein Leerzeichen zwischen Variable und Gleichheitszeichen sowie ein Leerzeichen zwischen Gleichheitszeichen und Wert. Zwischen dem Wert und dem Semikolon darf kein Leerzeichen stehen. Variablennamen beginnen mit Kleinbuchstaben und enthalten gegebenenfalls, den enthaltenen Begriffen entsprechend, Unterstriche und werden gegebenenfalls durch Großbuchstaben und/oder Zahlen ergänzt. Es gilt immer angemessene Variablentypen zu wählen.

```
int number = 5;
bool yes = false;
char letter = 'c';
string text = "text";
```

Bei der Verwendung von einem **Kontrollausdruck** muss ein Leerzeichen zwischen dem Schlüsselwort und der öffnenden (runden) Klammer stehen. Die verpflichtend zu verwendende (geschweifte) Klammer steht zwangsläufig in einer neuen Zeile. Bei der Verwendung einer logischen Verknüpfung muss zwischen einer Bedingung1 und einer logischen Verknüpfung beziehungsweise einer logischen Verknüpfung und einer Bedingung2 ein Leerzeichen stehen. Bei der Verwendung von Vergleichsoperatoren steht ein Leerzeichen zwischen dem Wert1 und dem Operator beziehungsweise dem Operator und dem Wert2. Bei der Verwendung von Rechenzeichen steht jeweils ein Leerzeichen zwischen Operator und Wert.

```
if (val == val && val != val)
{
  act;
}
else if (val < val || val > val)
{
  act;
}
else if (!cond)
{
  act;
}
```

3 Java

Am Anfang einer **Java-Datei** finden alphabetisch sortiert die Imports statt. Danach erfolgt die Klassendefinition entsprechend dem Dateinamen. Grundsätzlich werden einzelne Leerzeilen zur Übersichtsverbesserung eingefügt. Mit derartigen Einfügungen sollte aber nicht all zu großzügig umgegangen werden. Auch ist mehr als eine Leerzeile hintereinander nicht erlaubt.

```
1 import java.*
2 import javax.*
3 class Main {
4   public static void main(String[] arguments) {
5     System.out.println("output");
6   }
7 }
```

Die öffnende (geschweifte) **Klammer** einer Klasse steht keinesfalls in einer neuen Zeile. Klassennamen beginnen mit einem Großbuchstaben und enthaltenen den enthaltenen Begriffen entsprechend weitere.

```
1 class MainClass {
2   public static void main(String[] arguments) {
3     System.out.println("output");
4   }
5 }
```

Die öffnende (geschweifte) Klammer einer **Methodendeklaration** steht keinesfalls in einer neuen Zeile, wobei ein Leerzeichen zwischen der abschließenden, runden Klammer und der öffnenden, geschweiften Klammer steht. Methodennamen beginnen mit einem Kleinbuchstaben und enthalten gegebenenfalls, den enthaltenen Begriffen entsprechend, Großbuchstaben. Methoden müssen ohne Leerzeichen zwischen dem Methodennamen und der öffnenden (runden) Klammer deklariert werden.
Methoden müssen ohne Leerzeichen zwischen dem Methodennamen und der öffnenden (runden) Klammer aufgerufen werden. Hinter dem Trennungskomma zwischen zwei Methodenparametern muss ein Leerzeichen stehen. Zwischen der abschließenden (runden) Klammer und dem Semikolon darf kein Leerzeichen stehen.

```
1 public method(argument, argument) {
2   actions;
3 }
4
5 method(argument, argument);
```

Bei der **Wertzuweisung einer Variable** steht grundsätzlich ein Leerzeichen zwischen Variable und Gleichheitszeichen sowie ein Leerzeichen zwischen Gleichheitszeichen und Wert. Zwischen dem Wert und dem Semikolon

darf kein Leerzeichen stehen. Variablennamen beginnen mit Kleinbuchstaben und enthalten gegebenenfalls, den enthaltenen Begriffen entsprechend, Großbuchstaben und werden gegebenenfalls durch Zahlen ergänzt. Es gilt immer angemessene Variablentypen zu wählen.

```
int number = 5;
boolean yes = false;
char character = 'c';
String text = "text";
```

Bei der Verwendung von einem **Kontrollausdruck** muss ein Leerzeichen zwischen dem Schlüsselwort und der öffnenden (runden) Klammer stehen, sowie ein Leerzeichen zwischen der schließenden (runden) Klammer und der verpflichtend zu verwendenden (geschweiften) Klammer, welche wiederum keinesfalls in einer neuen Zeile steht. Bei der Verwendung einer logischen Verknüpfung muss zwischen einer Bedingung1 und einer logischen Verknüpfung beziehungsweise einer logischen Verknüpfung und einer Bedingung2 ein Leerzeichen stehen. Bei der Verwendung von Vergleichsoperatoren steht ein Leerzeichen zwischen dem Wert1 und dem Operator beziehungsweise dem Operator und dem Wert2. Bei der Verwendung von Rechenzeichen steht jeweils ein Leerzeichen zwischen Operator und Wert.

```
if (value == value && value != value) {
  actions;
} else if (value < value || value > value) {
  actions;
} else if (!condition) {
  actions;
}
```

4 PHP

Am **Anfang und Ende einer PHP-Datei** werden die Tags <?php und ?> verwendet. Nach <? php und vor ?> erfolgt grundsätzlich ein Zeilenumbruch. Zwischen <?php und ?> wird nicht zusätzlich eingerückt.

```
<?php
echo("output");
?>
```

Grundsätzlich werden **doppelte Anführungszeichen** benutzt. Bei Arrayindexen werden jedoch einfache benutzt. Zahlen stehen generell nicht in Anführungszeichen. Einfache Anführungszeichen sind ansonsten nur erlaubt, wenn diese unbedingt erforderlich sind: Zum Beispiel bei der Verwendung von preg_replace().

Bei der **Wertzuweisung einer Variable** steht grundsätzlich ein Leerzeichen zwischen Variable und Gleichheitszeichen sowie ein Leerzeichen zwischen Gleichheitszeichen und Wert. Zwischen dem Wert und dem Semikolon darf kein Leerzeichen stehen. Variablennamen bestehen grundsätzlich ausschließlich aus Kleinbuchstaben und gegebenenfalls ergänzend aus Unterstrichen oder auch Zahlen.

```
$number = 5;
$yes = false;
$text = "text";
$array['index'] = "";
```

Bei der Verwendung von einem **Kontrollausdruck** muss ein Leerzeichen zwischen dem Schlüsselwort und der öffnenden (runden) Klammer stehen, sowie ein Leerzeichen zwischen der schließenden (runden) Klammer und der verpflichtend zu verwendenden (geschweiften) Klammer. Bei der Verwendung einer logischen Verknüpfung muss zwischen einer Bedingung1 und einer logischen Verknüpfung beziehungsweise einer logischen Verknüpfung und einer Bedingung2 ein Leerzeichen stehen. Bei der Verwendung von Vergleichsoperatoren steht ein Leerzeichen zwischen dem Wert1 und dem Operator beziehungsweise dem Operator und dem Wert2.

```
if (value == value && value != value) {
  actions;
} elseif (value < value || value > value) {
  actions;
} elseif (!condition) {
  actions;
}
```

Die öffnende (geschweifte) Klammer einer **Funktionsdeklaration** steht in jedem Fall in einer neuen Zeile. Funktionsnamen bestehen aus Kleinbuchstaben und gegebenenfalls aus Zahlen. Funktionen müssen ohne Leerzeichen

zwischen dem Funktionsnamen und der öffnenden (runden) Klammer deklariert werden.
Funktionen müssen ohne Leerzeichen zwischen dem Funktionsnamen und der öffnenden (runden) Klammer aufgerufen werden. Hinter dem Trennungskomma zwischen zwei Methodenparametern muss ein Leerzeichen stehen. Zwischen der abschließenden (runden) Klammer und dem Semikolon darf kein Leerzeichen stehen.

```
function function(argument, argument)
{
  return 0;
}

function(argument, argument);
```

Die öffnende (geschweifte) Klammer einer **Klasse** steht in jedem Fall in einer neuen Zeile. Klassennamen bestehen aus Kleinbuchstaben, keinesfalls jedoch aus Zahlen. Bei der **Instanzierung einer Klasse** wird die Klassenvariable entsprechend der Klasse benannt. Bei der Verwendung von Rechenzeichen steht jeweils ein Leerzeichen zwischen Operator und Wert.

```
class class
{
  function function(argument)
  {
    return 0;
  }
}

$class = new class();
```