

Java Coding Style Guide



Stand: 24.02.2015

Autor: Philipp Jenke

Dokumentation/Kommentare

- Jede Klasse, jede Methode und jede Objektvariable hat einen Beschreibungskommentar. Der Kommentar steht in der Zeile davor. Der Kommentar beginnt mit `/**` und endet mit `*/`.

Beispiel:

```
/**
 * Diese Klasse repräsentiert einen Punkt im dreidimensionalen
 * Raum. Jeder Punkt besteht aus drei Koordinaten.
 */
```

- Die Kommentare für Methoden müssen auch die JavaDoc-Tags `@param` für alle Methodenparameter und `@return` für den Rückgabewert beinhalten.
- Bei Standardmethoden genügt ein sehr kurzer Kommentar. Für Getter, Setter und Konstruktoren dürfen die Kommentare auch weggelassen werden.

Beispiel:

```
/**
 * Getter.
 */
```

oder

```
/**
 * Konstruktor.
 */
```

- Innerhalb von Blöcken werden immer dann Kommentare eingefügt, wenn besonders wichtige oder komplizierte Abschnitte folgen. Diese Kommentare stehen in der Zeile vor der beschriebenen Codezeile.

Beispiel:

```
// Wenn das Tamagotchi isst, wird es satt. Es wird aber auch müde.
hunger      = 0;
muedigkeit  = muedigkeit - 2;
```

Code-Einrückungen

- Jeder Block wird um zwei Leerzeichen (nicht Tabs) eingerückt.

Namenskonventionen

- Generell
 - Namen (Bezeichner) müssen sprechend sein. Das heißt, sie müssen selbst erklären, was sie bedeuten.

- Klassennamen
 - ... beginnen mit einem Substantiv.
 - ... beginnen mit einem Großbuchstaben, dann folgen Kleinbuchstaben.
 - ... beginnen jedes Teilwort mit einem Großbuchstaben, falls sie aus mehreren Wörtern zusammengesetzt sind ("UpperCamelCase").
 - ... beinhalten keine Sonderzeichen.

Beispiel:

```
public class Bruch{ ... }
```

- Methodennamen
 - ... beginnen mit einem Verb.
 - ... bestehen aus Kleinbuchstaben.
 - ... beginnen jedes Teilwort (außer das erste) mit einem Großbuchstaben, falls sie aus mehreren Wörtern zusammengesetzt sind ("lowerCamelCase").
 - ... beinhalten keine Sonderzeichen.

Beispiel:

```
public void testMultiplikation(){ ... }
```

- **Variablenamen**

- ... bestehen aus einem Substantiv oder sind aus mehreren Substantiven zusammengesetzt.
- ... bestehen aus Kleinbuchstaben.
- ... beginnen jedes Wort (außer das erste) mit einem Großbuchstaben, falls sie aus mehreren Wörtern zusammengesetzt sind ("lowerCamelCase").
- ... beinhalten keine Sonderzeichen.

Beispiel:

```
int anzahlDvds;  
int zaehler;  
int anzahlBananen;  
double ersterMesswert;
```

- Ausnahme: Konstanten

- ... bestehen aus Großbuchstaben.
- ... werden mit einem Unterstrich zusammengesetzt, falls sie aus mehreren Wörtern zusammengesetzt.

Beispiel:

```
final int MAX_ANZAHL_EINTRAEGE = 10;
```

Programmierstil

- Code darf sich nicht wiederholen (DRY-Prinzip, "Don't repeat yourself").
- Von zwei alternativen Lösungen, die das gleiche Ergebnis liefern, ist die einfachere/kürzere zu wählen (KISS-Prinzip, "Keep it simple and straight").
- Jede Klasse (ebenso wie jede Methode) hat genau eine eindeutige Aufgabe.
- Es dürfen keine Magic Numbers verwendet werden. Magic Numbers sind Zahlen, die direkt in einem Programmblock verwendet werden. Stattdessen müssen Sie für solche Zahlen Konstanten definieren, die einen verständlichen und selbsterklärenden Namen haben
- Analog zu der Regel zu "Magic Numbers" dürfen auch keine String-Literale in Code-Blöcken verwendet werden. Verwenden Sie auch hier immer Konstanten.

Blöcke

- Jeder Block wird in geschweifte Klammern eingefasst. Auch wenn der Block nur aus einer Zeile besteht (z.B. bei einer if-Anweisung).

Numerik

- Fließkommazahlen dürfen nicht mit dem == Operator verglichen werden (also auch nicht mit !=). Stattdessen muss ein Genauigkeitsintervall verwendet werden.

Beispiel:

```
double x = 1.0, y = 2.0  
if ( Math.abs( x - y ) < EPSILON ) {...}  
anstelle von  
if ( x == y ) {...}
```

Formatierung

- Eine Programmzeile beinhaltet nie mehr als 80 Zeichen.
- Zwischen Methoden und Blöcken werden Leerzeilen eingefügt. Innerhalb von Blöcken werden nur sparsam Leerzeilen verwendet, z.B. nur dann wenn ein neuer logischer Abschnitt beginnt.

Klassen

- Alle Objektvariablen haben den Sichtbarkeitsmodifizierer `private` oder `protected`.
- Ausnahme: Statische Konstanten dürfen auch als `public` deklariert werden.

- Jede Methode hat entweder `public` oder `private` oder `protected` als Sichtbarkeitsmodifizierer.
- Der Modifizierer `protected` wird nur verwendet, wenn es für eine Ableitungsimplementierung notwendig ist.
- Konstruktoren werden immer als `public` deklariert, außer es gibt zwingende, begründbare Ausnahmen (z.B. Verwendung des Singleton-Patterns).
- In einer Klasse wird nur `this` nur verwendet, wenn es notwendig ist.
- Methoden und Objektvariablen werden nur dann als `static` deklariert, wenn es notwendig ist (z.B. eine `main()`-Methode). Ausnahmen müssen im Einzelfall begründet werden.

Quellcode

- Jede Quellcode-Datei hat einen Header. Der Header beinhaltet mindestens folgende Informationen:
 - Name der Veranstaltung
 - Semester der Veranstaltung
 - Nummer der Praktikumsgruppe
 - Namen und E-Mailadressen aller Teilnehmer
 - Nummer des Aufgabenblatts
 - Verwendete Quellen (falls externe Quellen verwendet wurden)

Beispiel:

```
/**
 * Praktikum TIPR1, SS 2013
 * Gruppe: Max Muster (max.muster@haw-hamburg.de),
 *         Mia Meister (mia.meister@haw-hamburg.de)
 * Aufgabe: Aufgabenblatt 4, Aufgabe 1
 * Verwendete Quellen: Wikipedia (Begriff: Kugel)
 */
```