Java-Leitfaden

Version 1.2

17/06/2003

Ralf Engel, Patrick Gebhard, Michael Kipp, Martin Klesen
{rengel, gebhard, kipp, klesen}@dfki.de

Erklärtes Ziel dieses Leitfadens ist es, Java-Programmierer so anzuleiten, dass ihr Code auch von anderen gut gelesen und somit gewartet/wiederverwendet werden kann. Die oberste Regel lautet: KLARHEIT GEHT VOR EFFIZIENZ

1) **Dokumentiere mit javadoc**
Die Dokumentation sollte nach javadoc-Richtlinien erfolgen. Dokumentiert werden sollte jede Methode und jede Klasse, die public ist. Folgendes Beispiel zeigt die Angaben, die unbedingt enthalten sein sollten (die javadoc-Keywords sind rot, die Variablen-/Klassennamen blau gefärbt):

```
/**
 * Computes the alpha of omega by using lots of deltas.
 * @param omega The parameter for a single omega entity.
 * @return The alpha of omega.
 * @throws NoSuchOmegaException Thrown when the omega is invalid
 */
```

Auch selbst-definierte Runtime-Exceptions müssen dokumentiert werden.

2) **Benutze als Oberpackage: de.dfki**
Alle Packages sollten Unterpackages von de.dfki sein

3) **Lasse niemals einen Catch-Block leer**
Beim Exception-Handling darf der catch-Block NIEMALS leer bleiben. Stattdessen sollte entweder eine Fehlermeldung ausgegeben oder die Methode „printStackTrace“ aufgerufen werden.

4) **Halte dich an die Java-Namenskonventionen**
Variablen-, Klassen- und Methodennamen sollten den Java-Konventionen entsprechen (Unterstriche nur bei Konstanten benutzen!):

   a. Konstanten groß schreiben, z.B. GLOBAL_VARIABLE
   b. Normale Variablen klein beginnen, z.B. normalVariable
   c. Klassennamen groß beginnen, z.B. MyClass
   d. Methodennamen klein beginnen, z.B. normalMethod

5) **Verwende sprechende Variablennamen**
Für Variablen mit tragender Funktion unbedingt „sprechende“, d.h. selbst-erklärende, Variablennamen benutzen. Gute Beispiele: addressCounter, returnValue, filename. Schlechte Beispiele: adcn, rv, fn.

6) **Begründe statische Konstrukte**
Bei statischen Klassen/Methoden (außer main) in der Dokumentation begründen, warum diese Klasse/Methode statisch sein muss. Wenn keine gute Begründung vorliegt: einfach nicht-statisch machen!

7) **Halte deine Methoden kurz**
Keine Methode sollte länger als zwei Bildschirmseiten sein. Stattdessen sollten funktional abgeschlossene Blöcke in separate (private) Methoden ausgelagert werden.

8) **Dokumentiere auf Englisch**
Die Dokumentation sollte nach Möglichkeit auf Englisch erfolgen. Wenn der Programmierer aber sehr unsicher im Englischen ist, dann bitte, bitte auf Deutsch.

9) **Schreibe nur eine Anweisung pro Zeile**
Jede Zeile sollte nur eine einzige Anweisung enthalten. Also nicht so:

```
int i; i=j+1; System.out.println(i);
```

10) **Nutze geschweifte Klammern wie folgt...**
Die Klammerung mit geschweiften Klammern sollte so sein, dass nach einer öffnenden Klammer kein Code kommt und eine schließende auf einer neuen Zeile steht. Beispiel:

```
if (quality.equals(„good“)) {
  System.out.println(„OK“);
} else {
  System.out.println(„no good“);
}
```

11) **Deklariere Variablen/Methoden möglichst als „private“**
Der Zugriff auf Variablen und Methoden sollte möglichst stark eingegrenzt werden, damit klar ist, auf was „von außen“ zugegriffen werden kann und soll. Alle Methoden und Variablen, die nur innerhalb einer Klasse benutzt werden, sind als „private“ zu deklarieren.

12) **Benutze vorhandene Java Klassen und Methoden**
Zur Datenrepräsentation und Input-/Output und sonstwas sollte man immer erst checken, ob das nicht schon von Java abgedeckt ist (z.B. Listen, Logging, Sortieren, XML-Zeug und vieles mehr) und dann die entsprechenden Java-Konstrukte benutzen. Selbst programmierte Klone der vorhandenen Java-Konstrukte sind meistens fehleranfällig, unverständlich und müssen früher oder später eh rausoperiert werden.

13) **Entsorge toten Code**
Code, der nicht mehr genutzt wird, sollte möglichst frühzeitig entfernt (gelöscht, auskommentiert, in eine andere Datei ausgelagert) werden, damit ein Fremdbetrachter die wirklich verwendeten Konstrukte schnell identifizieren kann. Unter toten Code fallen überflüssige Konstruktoren, Variablen und Methoden. Apropos: bitte schreibt keine Methoden „auf Halde“, d.h. weil ihr vermutet, dass sie irgendwann mal nützlich sein könnten.