Dokumentationskonzept

Version	Autoren	Datum	Kommentar
1.0	Ralf Kustolski, Robert Remus	2006-04-24	
1.5	Robert Remus, Norman Heino	2006-05-22	Revision

Inhaltsverzeichnis

1 Einleitung

The value of standardization is that it takes less brain cycles to parse the code, so that you can focus more on what the code means.[1]

Daher werden im folgenden Dokumentationskonzept Standards festgelegt und Richtlinien aufgezeigt, die einen einheitlichen und gut strukturierten Aufbau unseres Projekts garantieren sollen. Diese Grundsätze sind ständig selbstverantwortlich zu verfolgen.

2 Quelltextdokumentation

Der Quelltext[2] ist präzise und gewissensvoll zu dokumentieren um die Lesbarkeit zu erhöhen und eine gute Einarbeitung bzw. Wiedereinarbeitung zu gewährleisten.

2.1 „Public"-Methoden und „public"-Variablen

Zur Dokumentation von „public"-Methoden bzw. „public"-Variablen wird *Javadoc*[3] eingesetzt; Beispiel:

```
...
public class ExampleClass {
   /**
   * Eine beispielhafte Beispiel-Methode
   * @param exampleValueIn beispielhafter Übergabewert
   * @return beispielhafter Rückgabewert
   */
   public int exampleMethod(int exampleValueIn) {
      ...
      return exampleValueOut;
   }
...
}
```

Mindestens enthalten sein müssen ggf. die *Javadoc-Tags* `@param`, `@return` und `@throws` bzw. `@exception` sowie eine Kurzbeschreibung der Methode.

2.2 „Private"-Methoden und „private"-Variablen

Zur Dokumentation von „private"-Methoden bzw. „private"-Variablen wird die von Java zur Verfügung gestellte Notation zum Kommentieren genutzt; Beispiel:

[1] Zitat aus Bruce Eckel - Thinking In Java, 3^{rd} Edition, Revision 4.0
[2] Der Begriff Quelltext umfasst hier Klassen, Methoden und Variablen
[3] Javadoc ist ein Software-Dokumentationswerkzeug, das aus Java-Quelltexten automatisch dokumentierende HTML-Datien erstellt.

```
...
   /*
   * Eine beispielhafte Beispiel-Methode mit beispielhafter Übergabe-
   * und Rückgabevariable
   */
   private int exampleMethod(int exampleValueIn) {
      ...
      return exampleValueOut;
   }
...
```

2.3 Klassen-Köpfe

Zur Dokumentation einzelner Klassen wird *Javadoc* eingesetzt; Beispiel:

```
/**
* Klasse, die dem Beispiel dient.
* @author Robert Remus
* @version 1.5
*/
public class ExampleClass {
...
}
```

Nach Möglichkeit und Notwendigkeit ist mit *HTML-Tags* zu arbeiten. Mindestens enthalten sein müssen die *Javadoc-Tags* `@author` und `@version` sowie Datum und eine Kurzbeschreibung der Klasse.

2.4 Datei-Köpfe

Jede (Java-)Datei hat folgenden Kopf zu führen:

```
/*
* <Dateiname.dateinendung>
* Encoding: <Encodingtype>
*
* Copyright (c) 2006 <Autor(en)>, <email at adress dot domain>
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
```

```
* documentation and/or other materials provided with the distribution.
*
* 3. Neither the name of the University nor the names of its contributors
* may be used to endorse or promote products derived from this software
* without specific prior written permission.
*/
```

Beispiel

```
/*
* ExampleClass.java
* Encoding: UTF-8
*
* Copyright (c) 2006 Robert Remus, never@ev.er
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the distribution.
*
* 3. Neither the name of the University nor the names of its contributors
* may be used to endorse or promote products derived from this software
* without specific prior written permission.
*/
```

3 Quelltext

Einheitlich geschriebener Quelltext erhöht die Lesbarkeit und erleichtert das gemeinsame Arbeiten und Bearbeiten, bsp. durch Teams in Software-Projekten.

3.1 Konventionen zur Quelltexterstellung

3.1.1 Bezeichner

Bezeichner meint Package-, Klassen-, Methoden-, Objekt-, Variablen- sowie Konstantennamen.

- Bezeichner sind in *Englisch* zu wählen.
- Javadoc-Kommentare sind in *Englisch* zu verfassen; Deutsche Javadoc-Kommentare werden in Außnahmefällen akzeptiert.

- Bezeichner werden *zusammen* geschrieben, der jeweils erste Buchstabe eines jeden Wortes innerhalb eines Klassen-, Methoden- oder Variablennamens wird *groß* geschrieben.
- Packagenamen werden in *Kleinbuchstaben* geschrieben.
- Klassennamen beginnen mit einem *Großbuchstaben*.
- Methoden-, Variablen- und Objektnamen beginnen mit einem *Kleinbuchstaben*.
- Konstanten sind „magic numbers“ vorzuziehen, Konstantennamen werden in *Großbuchstaben* geschrieben.
- Methodennamen sollten in der Form *„Substantiv folgt Verb“* aufgebaut sein.
- Es sind die *JavaBeans* Bezeichnerkonventionen für `is-`, `get-` und `set`-Methoden zu verwenden.

3.1.2 Spacing

Spacing meint den Einsatz von Leerzeichen, Tabulatoren und Zeilenumbrüchen.

- Öffnende geschweifte Klammern werden *nicht* in eine neue Zeile gesetzt.
- Die Einrücktiefe entspricht *vier* Leerzeichen bei normaler Einrückung, *acht* Leerzeichen bei Einrückung zum Fortsetzen einer Zeile.
- Falls eine Zeile umgebrochen werden muss, geschieht dies nach einem Operator oder nach einer öffnenden Klammer.
- Vor einer öffnenden Klammer befindet sich bei Methoden *kein* Leerzeichen.
- Nach einer öffnenden und vor einer schließenden Klammer befinden sich *keine* Leerzeichen.
- Zur deutlichen Abgrenzung zu Methoden, wird nach `if-`, `switch-`, `catch-`, `while-`, `for-` und `return`-Statements die öffnende Klammer mit einem Leerzeichen abgetrennt.
- Das `else`-Statement steht in der selben Zeile wie die vorangehende schließende geschweifte Klammer, abgetrennt durch ein Leerzeichen.
- Binäre und ternäre Operatoren werden durch Leerzeichen abgetrennt; Außnahmen bilden das Inkrementieren und Dekrementieren um eine Konstante.
- Nach Kommata und Semikola ist jeweils ein Leerzeichen einzufügen.
- Längerer Quellcode ist in mehrere logisch zusammenhängende Blöcke zu unterteilen, welche jeweils durch eine Leerzeile voneinander getrennt sind.
- Nach dem Casting-Operator wird der Operand direkt ohne Leerzeichen angeschlossen.

Diese Konventionen folgen den Regeln zur Quellcodeerstellung aus „Achut Reddy, Java™ Coding Style Guide, Sun Microsystems, Inc., 1998“.

3.2 Beispiel

```
public class DerivedClass extends BaseClass implements InterfaceOne,
        InterfaceTwo, InterfaceThree {

    int memberOne = 1;
    double memberTwo = 3.14159d;

    public static int staticMethod(int intParam) {

        for (int i = 0; i < intParam+1; ++i) {
            if (intParam < 10) {
                memberOne *= memberTwo;
            } else {
                memberOne *= (memberTwo / 2.0d);
            }
        }
        return (memberOne + memberTwo);
    }

    public int memberOne() {

        return memberOne;
    }
}
```

4 Richtlinien

Neben den bekannten Prinzipien der Verbalisierung, der problemadäquaten Datentypen, der Verfeinerung und der integrierten Dokumentation sollten u. A. die Richtlinien aus „Bruce Eckel – Thinking In Java, 3^{rd} Edition, Revision 4.0; Appendix B – Java Programming Guidelines: Design“ befolgt werden.

5 Dokumentsatz

Alle das Projekt begleitende, zu veröffentlichende Dokumente sind deutschsprachig und werden mit LaTeX in der Schriftart „Computer Modern“ im Schriftgrad 11 pt gesetzt. Kopf- und Fußzeilen sind diesem Dokumentationskonzept nachzuempfinden, das sonstige Layout ist problemadäquat zu wählen.

6 Benutzerhandbuch

Das Benutzerhandbuch[4] wird ein Mittelweg zwischen Benutzer-Leitfaden und Referenz-Handbuch, also Aufgaben- und Produktorientierung sein. Unser Benutzerhandbuch soll dem

[4]Ein Benutzerhandbuch soll etwas beschreiben und/oder zur Benutzung eines Produkts anleiten

Leser einerseits einen umfangreichen Einblick in die Funktionalität von teamInstance verschaffen, ihn andererseits aber auch bei der Installation und Konfiguration von teamInstance unterstützen.

7 Online-Hilfesystem

Es ist ein begleitendes Online-Hilfesystem nach JSR-168-Standard durch den Portlet-Helpmode vorgesehen.

8 Verantwortlichkeiten

Sowohl die interne und externe Dokumentation des Quelltextes als auch die Ausformulierung deutscher Sätze zur jeweiligen Funktionalität für Benutzerhandbuch und Online-Hilfesystem obliegt dem jeweiligen Implementierer.
Die Zusammenführung des Benutzerhandbuchs erfolgt durch den Verantwortlichen für Dokumentation.
Die Überwachung der Einhaltung der vereinbarten Standards zur Quelltexterstellung erfolgt durch den Verantwortlichen für die Implementierung und den Verantwortlichen für Qualitätssicherung, die Überwachung der Einhaltung der Grundsätze zur internen und externen Dokumentations erfolgt durch den Verantwortlichen für Dokumentation und den Verantwortlichen für Qualitätssicherung.