Dokumentationskonzept

1 Grundregeln für guten Quellcode und gute quellcodenahe Dokumentation

Ziel einer Quelltextdokumentation ist es, später leichter Fehler im Code finden und Änderungen vornehmen zu können. Es ist essentiell, dass alle beteiligten Programmierer einen möglichst einheitlichen Programmierstil verwenden und den Inhalt ihres Codes kommentieren. Jeder Programmierer ist selbst für die Dokumentation des von ihm erstellten Quellcodes verantwortlich. Darum gilt es im Vorfeld einige Regeln und Prinzipien zum Programmieren festzuhalten und sich beim Entwerfen des Quellcodes streng nach diesen zu richten. Man kann folgende grundsätzliche Kriterien für guten Javacode unterscheiden.

1.1 Prinzip der Verbalisierung

Gute Verbalisierung ermöglicht eine leichte Einarbeitung in den Quellcode und vereinfacht Modifikationen und Wartungsarbeiten am Projekt. Dazu müssen an jeder Stelle aussagekräftige und selbsterklärende Bezeichner gewählt werden, Kommentare müssen kurz und prägnant sein. In Folge dessen sind eher längere Bezeichnernamen zu wählen, da kürzere häufig nicht gut genug den Sachverhalt widerspiegeln.

1.2 Prinzip der integrierten Dokumentation

Die Dokumentation als integraler Bestandteil jeglicher Programmieraufgabe wird optimalerweise quellcodenah entwickelt. Dabei werden Erklärung, Entwicklungsstadien, Entscheidungen, sowie Modifikationen und Neuentwicklungen in natürlicher Sprache hinterlegt. Bei Nachdokumentationen fehlt es oft an wichtigen Informationen die während der Codeerstellung noch vorhanden waren.

1.3 Problemadäquate Datentypen

Datentypen sollten Probleme weder unter- noch überspezifizieren, weshalb der Wertebereich exakt definiert werden muss. Können Basistypen sinnvoll eingesetzt werden, so ist dies zu tun, während komplexe Daten objektorientiert als eigene Klassen realisiert werden.

1.4 Verfeinerung

Abstraktionsebenen bilden eine Hierachie im Code, der Entwicklungsprozess inklusive aller Entscheidungen muss dokumentiert werden.
Unter diesen Gesichtspunkten gelten für die Gruppe GR-07-3 folgende verbindliche syntaktische und semantische Regeln:

- Es wird an jeder Stelle englische Namensgebung verwendet.
- Variablenname beginnen mit kleinen Buchstaben.
- Besteht ein Variablenname aus mehreren Wörtern, dann beginnt jedes Wort mit einem Großbuchstaben, z.B. timeLeft, Unterstriche werden nicht zur Trennung eingesetzt.
- Paketnamen enthalten nur Kleinbuchstaben und keine Unterstriche.
- Klassennamen beginnen immer mit einem Großbuchstaben.

- Bestehen Klassennamen aus mehreren Worten werden diese nicht durch Unterstriche getrennt sondern fortlaufend geschrieben, wobei alle Worte mit Großbuchstaben beginnen.
- Objektnamen beginnen immer mit einem Kleinbuchstaben, enden in der Regel mit dem Klassennamen, z.B. examEditor.
- Attributnamen beginnen immer mit einem Kleinbuchstaben.
- Konstanten bestehen nur aus Großbuchstaben, bestehen sie aus mehreren Worten, werden diese durch Unterstriche getrennt.
- Methodennamen beginnen immer mit einem Kleinbuchstaben, folgende Worte beginnen jeweils mit Großbuchstaben und es kommen keine Unterstriche vor.
- Methodennamen heißen getAttributname(), wenn nur ein Attributwert eines Objektes gelesen wird, setAttributname(attr:Attributname), wenn nur ein Attributwert eines Objektes gespeichert wird und isAttributname(), wenn das Ergebnis nur wahr (true) oder falsch (false) sein kann, z.B. isMarried(), isClosed().
- In Klassen werden zuerst alle Attributdeklarationen, die für die gesamte Klasse gelten, notiert. Anschließend die Konstruktoren, gefolgt von Operationen, z.B. Zuweisung, Vergleich. Am Schluss stehen die Operationen mit schreibendem bzw. lesendem Zugriff (setAttributname() bzw. getAttributname()). Die Reihenfolge ist verbindlich.
- Jede Klasse wird vollständig mit Javadoc nah an der Entwicklung dokumentiert. Die Dokumentation umfasst dabei mindest ein Programmvorspann, vollständige Methodendokumentation sowie Dokumentation der relevanten Attribute. Der Vorspann besteht dabei mindestend aus Name, Aufgabenstellung, Komplexitätskenngrößen (insofern relevant), Name des Autors, Versionsnummer, Datum. Methodendokumentationen umfassen mindestens Beschreibung der Methode und ihrere Parameter, sowie Rückgabewerte in Javadoc-Syntax. Attributsbeschreibungen im Javadocformat umfassen eine einzeilige Beschreibung.
- Quellcodedokumentation erfolgt nach dem Befehl mittels '//'. Sollte sich ein Quellcodekommentar über mehrere Zeilen erstrecken, sind die '//' der nächsten Zeilen bündig untereinander zu schreiben. Nach '//' folgt ein Leerzeichen.
- Bei binären Operatoren werden Operanden und Operator durch jeweils ein Leerzeichen getrennt.
- Bei binären Operatoren mit Konstanten werden diese als erster Operant notiert und die Variable als zweiter.
- Es werden keine Leerzeichen bei der Punktnotation (Objekt.Operation) verwendet.
- Zwischen Operationsname und Klammer steht kein Leerzeichen. Nach der öffnenden und vor der schließenden Klammer steht ebenfalls kein Leerzeichen, z.B. setColor(Color.blue).
- Nach Schlüsselwörtern steht grundsätzlich ein Leerzeichen.
- Geschweifte Klammern zählen als Befehl, also folgt in der Zeile, in der eine geschweifte Klammer vorkommt, nichts mehr.
- Grundsätzlich steht in einer Zeile nur ein Befehl.
- Alle Zeilen innerhalb eines Klammerpaars sind jeweils um 4 Leerzeichen nach rechts eingerückt.

- Import-Anweisung werden alphabetisch sortiert, es sollten nur ganze Pakete importiert werden (also per paket.*), wenn mehr als 2 Klassen aus diesem Paket verwendet werden.
- Codezeilen sollten mit Kommentaren nie über eine Bildschirmbreite hinausgehen.
- Methoden sollten nie mehr als eine Bildschirmseite umfassen.

2 Javadoc

Javadoc ist ein Software-Dokumentationswerkzeug das aus Java-Quelltexten automatisch HTML-Dokumentationsdateien erstellt. Die Dokumentation kann somit durch spezielle Kommentare im Quelltext erstellt und gesteuert werden. Dadurch können Beschreibungen für Interfaces, Klassen, Methoden und Felder über spezielle Doclet-Tags definiert werden.
Javadoc-Kommentare beginnen mit '/**' und enden mit '*/'. Diese Kommentare können nun Beschreibungen enthalten oder auch spezielle tags, die durch @ eingeleitet werden. Das Standard-Doclet erzeugt eine Ausgabe in HTML, es wären aber auch andere Formate wie .pdf möglich.

- Sollten im Javadoc-Kommentar Zeilenumbrüche auftreten beginnen die neuen Zeilen mit '*' gefolgt von einem Leerzeichen. Dabei sind alle Sternchen jeweils bündig unter dem ersten Sternchen des '/**'.
- Zu benutzen sind die folgenden tags:

@see [Klasse]#[Methode]	erbt die Klasse Methoden von Oberklassen oder überschreibt/überlädt diese, so ist ein Verweis zur Oberklassenmethode anzubringen
@param [Parameter] [Beschreibung]	Muss für alle Parameter einer Methode erstellt werden um die Bedeutung der Variablen zu erklären
@return [Beschreibung]	erklärt den Rückgabewert
@exception [Klasse]	verweist auf die Beschreibung der Exception 'Klassenname'
@version [Text]	Versionbeschreibung
@author [Name]	Autor dieser Klasse

Weitere tags sollten vermieden werden.