Dokumentationskonzept

1. Zielbestimmung

Die interne Dokumentation ist maßgebend für die Qualität des Quellcodes und somit auch für die Qualität der entsprechenden Software. Eine gute Dokumentation erleichtert eine Einarbeitung in den Quellcode für Projektfremde oder nach längerem Projektstillstand, aufgrund besserer Lesbarkeit. Aus diesen Gründen sollte man bei der quelltextnahen Dokumentation bestimmte Standards einhalten.

2. Qualitätsstandards der integrierten Dokumentation

2.1 Kommentierung

In Java wird die integrierte Dokumentation durch Kommentare verwirklicht, die mittels `javadoc` extrahiert und automatisch in ein HTML-Dokument umgewandelt werden. Dokumentationskommentare in Java sind wie folgt aufgebaut: `/** Kommentar */` oder `//Kommentar`
Kommentare gehören nicht zum eigentlichen Quelltext und werden vom Compiler überlesen. In ihnen können außerdem verschiedene Befehle vorkommen, wie z.B. HTML-Befehle oder Befehle mit „@“ am Anfang.

Die wichtigsten Merkmale des Programms werden direkt am Anfang eines neuen Quellcodes im Programmvorspann gegeben. Dieser beinhaltet eine Kurzbeschreibung des Programms und diverse Verwaltungsinformationen:

- Selbsterklärender Programmname
- Kurze Beschreibung der Programmaufgabe
- Zeit- und Speicherkomplexität
- Programmautor
- Version und Datum

Die Versionsnummer besteht in der Regel aus 2 Teilen, der Release- und der Level-Nummer. Beide Ziffern sind durch einen Punkt voneinander getrennt. Am Anfang der Entwicklung beginnt man in der Regel bei 0.1 zu zählen. Die Level-Nummer wird erhöht, wenn kleine Änderungen vorgenommen werden, die Release-Nummer erhöht sich bei größeren Erweiterungen bzw. Änderungen. Die erste fertige Version eines Programms trägt somit die Nummer 1.0.
Bsp. für einen Programmvorspann:

```
/**Programmname: Kundenverwaltung
 * Aufgabe: Es werden Firmenkunden verwaltet
 * Die Anzahl der erfassten Kunden wird gezählt.
 * @author Anne Nitsche
 * @version 1.0 / 22.4.05
 */
```

Des Weiteren sollten alle Programmkomponenten (Methode/Operation, Konstruktor, Attributdeklarationen usw.) kurz vorher gekennzeichnet oder im Falle von Operationen auch näher beschrieben werden. Bei der Kommentierung von Operationen können spezielle Befehle genutzt werden, die genauere Auskunft z.B. über verwendete Parameter geben. Solche Befehle sind:

- @param: Name und Bezeichnung von Parametern
- @return: Beschreibung eines Rückgabeparameters
- @exception: Name und Beschreibung einer Ausnahme
- @see: Verweis auf eine Klasse

Ferner können auch Schleifen mit einem kleinen Kommentar am Rande versehen werden, in dem kurz die Aufgabe genannt wird, damit später keine Unklarheiten entstehen, wenn mehrere Schleifen ineinander vorkommen.

2.2 Formatierung

Um die Übersichtlichkeit und Lesbarkeit zu verbessern, sollten auch bei der Formatierung bestimmte Regeln beachtet werden.

Der *Aufbau einer Klasse* sollte immer derselben Reihenfolge obliegen:

1. Attribute deklarieren
2. Konstruktoren
3. Operationen
4. Setter (schreibende Operationen)
5. Getter (lesende Operationen).

Eine geregelte Benutzung von *Leerzeichen* bringt ebenfalls einen deutlichen Aufbau:

1. Operanden und Operator werden durch ein Leerzeichen getrennt
2. keine Leerzeichen für Punktnotation
3. kein Leerzeichen zwischen Operationsname und Klammer, sowie nach öffnender und vor schließender Klammer

Durch *Einrückungen von Klammern* bestimmter Strukturen wird der Quellcode überschaubar:

1. Zusammengehörende geschweifte Klammern stehen untereinander, d.h. in derselben „Spalte“
2. eine geschweifte Klammer steht allein in einer Zeile
3. Zeilen innerhalb einer geschweiften Klammer sind gleichmäßig nach rechts eingerückt (mind. 4 Leerzeichen)

2.3 Verbalisierung

Unter Verbalisierung versteht man, bezogen auf die Entwicklung eines Programms, eine geeignete Namensgebung. Klassen, Operationen, Attribute und Konstruktoren haben alle eine festgelegte Syntax für den Aufbau ihrer Namen (Bezeichner). Trotzdem sollten sie auch problemnah sein, d.h. der Name sollte die Funktion verraten. Bsp.: `aendernAdresse`
Das erläuterte Konzept erhöht die Verständlichkeit eines Programms.
(Auf dessen festgeschriebene Syntax soll jetzt nicht näher eingegangen werden.)

3. Verantwortlichkeiten

Die Hauptverantwortung für die interne Dokumentation liegt in den Händen der Implementierer. Zeitgleich mit dem Schreiben des Quellcodes für das entstehende Programm, müssen sie ihr Werk nach oben genannten Regeln dokumentieren.
Die Entwurfs-Dokumentation kann von allen geschrieben werden, die mit sämtlichen Merkmalen (Struktur- und Entwurfs-Prinzipien und -Entscheidungen) der entstehenden Software vertraut sind.

4. Externe Dokumentation – Entwurfsdokumentation

4.1 Bestandteile

- Entwurfs-Beschreibung
- Handbuch / Online-Hilfe

4.2. Design-Beschreibung

Ziel der Design-Beschreibung ist es, Fremdprogrammieren die Klassenstruktur des Programmes darzulegen, um eine schnelle Einarbeitung zu ermöglichen
Sie dokumentiert und begründet die getroffenen Designentscheidungen in Bezug auf OOD-Konzepte und die Klassenstruktur mit Hilfe von UML-Diagrammen.

4.3 Benutzerhandbuch bzw. Online-Hilfe

Diese Dokumentation ist für den Endbenutzer bzw. Anwender des Software-Produktes vorgesehen. Das Benutzerhandbuch bzw. eine Online Hilfe sollte parallel zur Software-Entwicklung entstehen.
Der Anwender, wird darin in das Produkt eingeführt. Weiterhin müssen alle Funktionen der Anwendung ausreichend beschrieben und erklärt sein. Außerdem sollte es detaillierte Informationen über Systemvoraussetzungen, Installation und Handhabung des Programms beinhalten.
Ebenfalls ist das Vorhandensein einer Begriffsübersicht mit zugehörigen Erläuterungen zu beachten, für die beispielsweise alle Begriffe des Glossars in Betracht kommen.