Gruppe: GR-06-6
Michael Gassner

Softwaretechnik-Praktikum 2006
- Dokumentationskonzept -

1. Einleitung

Eine gute und vollständige Dokumentation ist bei einem Projekt an dem mehrere Programmierer mitwirken unablässlich. Der Quellcode wird so übersichtlicher und strukturierter, Klassen und Variablen werden „vor Ort“ erklärt. Ein Außenstehender kann sich viel einfacher und schneller in den gegebenen Quellcode einarbeiten, was die Wartung, Wiederverwendbarkeit und Weiterentwicklung durch andere erheblich vereinfacht.
Die Qualität der Dokumentation hat erheblichen Einfluss auf die Qualität des Gesamtprodukts, weshalb es notwendig ist, dass diese innerhalb einer Gruppe ausführlich und nach festgelegten Standards durchgeführt wird.

2. Anforderungen an den Quellcode

2.1. Lesbarkeit

Um eine gute Lesbarkeit zu garantieren müssen sich alle Programmierer an die „Regeln für guten Code“ halten. Einige Gedanken dazu sind im Balzert Abschnitt 4.2.1. einige Überlegungen formuliert, aber auch in den „Java Code Conventions“ von Sun Midrcosystems. (einzusehen unter http://java.sun.com/docs/codeconv/index.html).
Wichtig für gute Lesbarkeit ist auch die Namensgebung, auf die später noch einmal eingegangen wird.

2.2. Strukturierung

Gute Strukturierung wird durch eine vernünftige Modellierung der Klassen sichergestellt. Lange und unübersichtliche Methoden snd zu verhindern, indem man diese in kleinere Methoden aufteilt.

2.3. Fehlersuche

Es sind alle möglichen Fehler mit Hilfe von Java's Exception-Konzept abzufangen um diese schneller ausfindig zu machen. Es sollten keine Fehlercodes außer „null“ zurück gegeben werden.

2.4. Wartung/Weiterentwicklung

Um eine kurze Einarbeitungszeit sowie gute Wartbarkeit und Änderbarkeit zu gewährleisten, sollte der Zeck und die Funktionsweise einer Klasse möglichst selbsterklärend sein. Dies wird durch klare, plausible und realitätsnahe Modellierung der Programmklassen erreicht.

3. Interne Dokumentation – Dokumentation im Quellcode

3.1. Namensgebung

Von großer Wichtigkeit ist eine aussagekräftige Namensgebung, aus der Zweck und Verwendung der Klasse/Methode/Variable ersichtlich sind.
- Bei der Wahl des Namen ist auf Abkürzungen oder Eigenkreationen zu verzichten.
- Bei Methoden sollte der Name möglichst aus einem Verb und gegebenenfalls aus einem Substantiv bestehen, die in Kombination den Zweck der Methode darstellen.
- Es sind in den einzelnen Klassen set- und get-Methoden zu verwenden.

3.2. javadoc – Dokumentation

Mit javadoc hat man ein Software-Dokumentationswerkzeug, das mit Hilfe spezieller Kommentare im Quellcode automatisch eine Dokumentation im html-Format erstellt. Es können Klassen, Methoden, Felder oder Variablen über bestimmte Befehle definiert und kommentiert werden.
Die gängigsten Befehle im Überblick:
- Kommentare werden durch /** Kommentar */ gekennzeichnet
- mit @ können Parameter ausführlicher kommentiert werden
    - @param: Name und Bezeichnung von Parametern
    - @return: Beschreibung Ergebnisparameter
    - @exception: Name und Beschreibung der Ausnahme
    - @see: Verweis auf andere Klassen möglich

3.3. Inline – Dokumentation
Mit der Inline-Dokumentation werden Algorithmen näher beschrieben. Dies geschieht durch Kommentare innerhalb des Quellcodes. Dadurch wird die Verständlichkeit des Quellcodes erhöht, es werden die Gedanken hinter und die Funktion der einzelnen Programmteile näher erklärt.
Dies erleichtert die Einarbeitung in den Quelltext und fördert die Übersichtlichkeit.

3.4. Test – Dokumentation
Sinn und Funktionsweise des Testes sind zu erkären. Wozu dient er? wie wird der Test vollzogen? Welche Klasse wird getestet? Worauf lag der Fokus beim Test?

3.5. Verantwortlichkeit
Jeder Programmierer ist selbst dafür verantwortlich, dass sein Programmfragment den Vorgaben entsprechend dokumentiert ist. Der Quellcode ist sofort beim schreiben zu dokumentieren und nicht erst später, sonst könnten Sachen vergessen oder falsch dokumentiert werden. Selbstkontrolle innerhalb der Impelementierer ist zu begrüßen. Di endgültige Prüfung liegt beim Verantwortlichen für Qualitätssicherng und Dokumentation. Er weißt den Programmierer auf Fehler oder fehlende Dokumentation hin, worauf dies nachgearbeitet werden muss.

4. Externe Dokumentation

4.1. Design – Beschreibung
Die Design – Beschreibung dokumentiert das Design des Programms. Es beschreibt den Aufbau und die sich ergebende Paket- und Klassenstruktur. Dies gescieht in der Regel über UML-Diagramme.
Die Design – Beschreibung soll einen Überblick über das Programm geben und es anderen Programmierern ermöglichen sich in die Programmstruktur einzuarbeiten.

4.2. Benutzerhandbuch
Zu einem vollständigen Software – Produkt gehört eine vollständige und fehlerfreie Dokumentation. Eine Version dieser Dokumentation, die für den Anwender gedacht ist nennt man ein Benutzerhandbuch. Dieses entsteht parallel zur Entwicklung des Software – Pordukts und begründet sich auf dem Glossar und dem Lasten-/Pflichtenheft. Es gibt einen Überblick über das Programm, die Funktionalität, Systemvoraussetzungen, Installation und Handhabung des Programms.

4.3. Verantwortlichkeit
Die Designbeschreibung liegt in der Verantwortlichkeit des Verantwortlichen für Modellierung, das Benutzerhandbuch in der des technischen Assistenten in Zusammenarbeit mit der Gruppe.