Dokumentationskonzept

1. Grundregeln für übersichtlichen Javacode [nach Balzert]

   Nach Balzert existieren im Wesentlichen vier Kriterien für übersichtlichen, gut geschriebenen Quellcode:

   - Verbalisierung
     - aussagekräftige und selbsterklärende Namensgebung
     - geeignete Kommentare
     - geeignete Bezeichnerwahl (lieber längere Bezeichner, kürzere sind meist nicht aussagekräftig)

     Durch eine gute Verbalisierung ist eine leichte Einarbeitung in unseren Quellcode möglich und sie erleichtert zusätzlich die Modifikation und Wartung des Produktes.
   - Verwendung problemadäquater Datentypen

     Können Daten durch Basistypen beschrieben werden, ist der geeignete Basistyp zu wählen. Der Wertebereich sollte so festgelegt werden, dass er möglichst genau das Problem widerspiegelt. Durch die Verwendung problemadäquater Datentypen werden die Daten des Projekts 1:1 in Datentypen des Programms abgebildet, d.h. Wertebereiche werden weder unter- noch überspezifiziert.
   - Verfeinerung

     Der Code wird durch Abstraktionsebenen strukturiert, wodurch eine Art Hierarchie entsteht. Der Entwicklungsprozess wird im Quellprogramm dokumentiert und so können Entwicklungsentscheidungen besser nachvollzogen werden.
   - Integrierte Dokumentation

     Die Dokumentation ist integraler Bestandteil jeder Programmieraufgabe. Sie wird eine Kurzbeschreibung des Programms (empfehlenswert ist auch ein Programmvorspann, welcher den Programmnamen, die jeweilige Aufgabe, Zeit-und Speicherkomplexität, Name des Programmautors sowie Versionsnummer und Datum enthält), Verwaltungsinformationen und Kommentierung des Quellcodes enthalten. Dabei ist eine Nachdokumentation am Ende der Codeerstellung nicht von Vorteil, da Informationen, die während der Entwicklung angefallen sind, oft nicht mehr vorhanden sind. Zudem sollten Entwicklungsentscheidungen dokumentiert werden, um bei etwaigen Modifikationen und Neuentwicklungen bereits gemachte Erfahrungen auswerten zu können.

   Im Sinne dieser vier Kriterien haben wir fi¿½r unser Projekt folgende Standards festgelegt:

   - Es wird eine einheitliche, gut lesbare Formatierung des Codes angestrebt - Einrückung, konsistente Umbrüche, maximal 80 Zeichen pro Codezeile, wobei eine Codezeile nicht mehr als eine Anweisung beinhaltet.
   - Wir bemühen uns, um sinnvolle Funktionsgrößen. D.h. eine Funktion wird nicht mehr als 30 Zeilen haben.
   - Um das Einfügen von Kommentaren vor/hinter Programmblöcken zu erleichtern, werden Blockklammern jeweils allein in einer Quellcodezeile stehen.
   - Import-Statements werden nicht durch Sternchen abgekürzt, und bei einer Anzahl von 10 alphabetisch und nach Produkten sortiert.
   - Beim Aufruf einer Methode aus einer anderen Klasse, wird der Pfad zu dieser Methode bei den Import-Statements mit aufgeführt.

- Iteratoren, welche im Kopf einer Schleifenanweisung genutzt werden, werden auch dort initialisiert.
- Selbstverständlich wird jede Klasse in die automatische Dokumentation JavaDoc eingebunden werden. (mehr dazu in Punkt 2)
- JavaDoc-Kommentare für Variablen werden nicht mehr als eine Zeile umfassen.

Am Ende steht als Ergebnis ein verständliches, leicht lesbares, selbstdokumentierendes und gut wartbares Software-Projekt.

2. Javadoc - automatisierte Dokumentation in JAVA

   Funktionsweise: Javadoc ist ein Software-Dokumentationswerkzeug, welches mittels geeigneter Kommentare im Java-Quellcode automatisch eine HTML-Dokumentationsdatei (es sind auch andere Formate, wie pdf und xml möglich) generiert. Dadurch können Beschreibungen für Interfaces, Klassen, Methoden und Felder über spezielle Tags definiert werden. Dabei sind Tags die einzelnen Themen von Javadoc(z.B.: Der Tag @see [Klassenname] erzeugt einen Verweis zur Klasse Klassenname). Javadoc gibt die Themen samt Informationen normalerweise in HTML und angebrachter Form heraus. In den Javadoc-Kommentaren werden Tags mit einem @ angegeben. Dann folgt der Name des Themas und dann die Informationen dazu. Hier eine kurze übersicht der wichtigsten zu nutzenden Tags:

   - @see [Klassenname] - Dieses Tag erzeugt einen Verweis zur Klasse Klassenname.
   - @see [Klassenname]#[Methodenname] - Dieses Tag erzeugt einen Verweis zu der Methode Methodenname der Klasse Klassenname.
   - @param [Parametername] [Beschreibung] - Dieses Tag wird zur Dokumentation von übergabeparametern verwendet. Dabei ist Beschreibung eine wörtliche Beschreibung des Parameters.
   - @version [Text] - Hiermit kann die Version des Progammcodes dokumentiert werden.
   - @author [Name] - Hier sollte der Name des Autors eingefügt werden.
   - @return [Beschreibung] - Hiermit kann ein Return-Wert beschrieben werden.
   - @exception [Klassenname] - Mit diesem Tag wird ein Verweis auf die Exception Klassenname erstellt.

   ... Die Liste der Themen ist beliebig erweiterbar.

3. Besonderheiten bei "Programmierung in Paaren" (PP)

   Da die Implementierung nach dem Grundsatz "Programmieren in Paaren" erfolgt, liegt auf den "Richtlinien für guten Quellcode" und dem "Prinzip der integrierten Dokumentation" noch größere Bedeutung, da sich Mitglieder anderer PP's ebenfalls in kürzester Zeit in den, von einer PP erstellten, Quellcode einarbeiten müssen.

   - Da das Projekt in JAVA implementiert wird, wird sich an die "Code Connvention for the JAVA Programming Language" gehalten werden. Einige Bestandteile sind unter Punkt 1 bereits formuliert. Im Internet befinden sich unter der URL http://java.sun.com/docs/codeconv/html/Code weitere Grundsätze. Vor Beginn der Implementierung wird sich Jede PP dieses Dokument durcharbeiten, und sich an die dort formulierten Grundsätze halten.
   - Der Quellcode selbst ist natürlich von der jeweiligen PP zu dokumentieren. Dabei werden die Blockkommentare und die Inlinekommentare von den PP's überall dort verfasst, wo sie für das Verständnis des Quellcodes von Nöten sind. Auch die javadoc-Kommentare werden natürlich von den jeweiligen PP's verfasst.
   - Nach Abschluss der Implementierung einer Storrie, ist der Code den anderen Gruppenmitgliedern im CVS zugänglich zu machen.

- Zudem ist von jedem Zweierteam eine kleine Entwurfsbeschreibung (in Stichpunkten) für ihre Storrie zu verfassen und im Gruppen-Wiki publik zu machen. Bei Abweichungen von der Gesamt-Entwurfbeschreibung, werden diese dann von den anderen Gruppenmitgliedern im Wiki diskutiert, und erst dann wird die Entwurfsbeschreibung des Gesamtsystems gegebenenfalls vom Verantwortlichen für Dokumentation bzw. Modellierung im CVS aktualisiert.