Dokumentationsrichtlinien für das Programmierpraktikum (Java)

Inhaltsübersicht

I. Formale Anforderungen

1. Die Dokumentation muss als PDF mit im Repository abgelegt werden.
2. Es sind auf sinnvolle Seitenumbrüche zu achten (Ein neues Kapitel fängt auf einer neuen Seite an und nicht noch auf den letzten beiden Zeilen einer Seite).
3. Die Schriftgröße für den Fließtext ist 10 bis 12 Punkte, die Schriftgrößen (auch der Überschriften) sind einheitlich.
4. Sie ist frei von Rechtschreibfehlern und grammatikalischen Fehlern (sollte ein akzeptables Maß überschritten werden, führt dies zu Punktabzug).
5. Der Text ist streng formal und abstrakt formuliert (selbst wenn ich mich auf dieser Seite nicht daran halte). Als Orientierung für den Sprachstil könnt Ihr Euch an die Handbücher kommerzieller Softwareprodukte halten.
6. Es gibt zwei getrennte Einheiten: Das Benutzerhandbuch und das Programmierhandbuch.
   Das Programmierhandbuch darf Referenzen auf das Benutzerhandbuch haben (der Programmierer sollte immer beides zur Verfügung haben), nicht jedoch umgekehrt.
7. Die Seiten müssen auf jeden Fall fortlaufend nummeriert werden, entweder im Benutzerhandbuch beginnend und im Programmierhandbuch weiterzählend oder im Programmiererhandbuch neu beginnend.

II. Die Dokumentation

Die Dokumentation besteht immer aus folgenden Teilen:

1. Deckblatt

Das Deckblatt ist die allererste Seite der Doku und besteht aus:

1. dem Titel 'Programmierpraktikum'
2. dem Programmtitel
3. Name, Vorname, Fachrichtung, Matrikelnummer, Fachsemester und Verwaltungssemester.
4. Nur wenn E-Mail-Benachrichtigung nicht nur an die FH-E-Mail-Adresse (MInfxxxx@fh-wedel.de) gewünscht wird, muss die weitere E-Mail-Adresse angegeben werden.

2. Inhaltsverzeichnis

Es gibt entweder ein gemeinsames (bei durchgehender Seitennummerierung) oder jeweils für Benutzerhandbuch und Programmierhandbuch getrennte Inhaltsverzeichnisse.
Der Rest sollte klar sein (Das Inhaltsverzeichnis bezieht sich natürlich auf Seitenzahlen und die einzelnen Kapitel sind mit numerischen Zusätzen in eine vernünftige Struktur zu bringen.).
Beispiel:

2 Benutzerhandbuch 3

3. Allgemeine Problemstellung

Die Aufgabenstellung ist als allgemeine Problemstellung voranzustellen. Die Kopie von Abschnitten der Aufgabenstellung von der Webseite ist erlaubt, sofern dies kenntlich gemacht wird.

4. Benutzerhandbuch

Das Benutzerhandbuch besteht mindestens (ohne Ausnahme) aus folgenden Kapiteln (Kapitelüberschriften müssen genau so beibehalten werden):

1. Ablaufbedingungen: Eine saubere und vollständige tabellarische Auflistung der Hardware und Softwarekomponenten (inkl. Version), die für den Ablauf der kompilierten Version Eures Programms mindestens notwendig sind. In der Regel ist hier lediglich ein installiertes JRE in einer bestimmten Version vorzuschreiben.
2. Programminstallation/Programmstart: Welches Archiv muss ausgepackt, welche Datei mit welchen Parametern gestartet werden?
3. Bedienungsanleitung: Zuerst mal werden hier die Hintergrundinformationen, die für das Verständnis des Programms notwendig sind, aufgeführt. Die Spielregeln bzw. Programmanforderungen müssen hier nicht erklärt werden, sondern wie das Programm die Anforderungen erfüllt (Beispiel: "Ein Spielstand kann über den Menüpunkt Datei/Öffnen geladen werden. Wurde bereits ein Spiel gestartet, so erfolgt die Nachfrage ob und wohin der aktuelle Spielstand gespeichert werden soll.").
   Dann wird jede Funktionalität des Programms und jedes mögliche Ablaufstadium sorgfältig und so detailliert wie möglich erklärt.
   Dabei sollte von Screenshots ausführlich Gebrauch gemacht werden (diese sollen sinnvoll in den Fließtext eingebunden werden).
   Auch können z.B. Spielregeln aus der Aufgabenstellung kopiert in den Text einfließen.
   Eine weitere Untergliederung, inkl. Überschriften und Aufnahme in das Inhaltsverzeichnis, ist zwingend erforderlich.

5. Programmierhandbuch

Grundsätzlich soll dieser Teil dazu dienen, einen Software-Entwickler über Deine Gedankengänge bei der Planung und Umsetzung des Projektes zu informieren, damit er sich nach möglichst kurzer Einarbeitungszeit der Fortführung, Pflege etc. widmen kann. Aussagekräftige und gut strukturierte Darstellungen sind hier gefordert. Stell Dir selbst einmal die Frage, was Du alles brauchst, um Dich in Deinem Projekt nach einigen Jahren wieder zurechtzufinden.

Das Programmierhandbuch ist ein sehr wichtiger Teil der Dokumentation und besteht immer und ohne Ausnahme mindestens aus folgenden Kapiteln (Kapitelüberschriften sollten genauso beibehalten werden):

1. **Entwicklungskonfiguration:**
   Eine saubere und vollständige tabellarische Auflistung der Softwarekomponenten mit Hilfe derer Du das Programm entwickelt hast. Relevant sind hier Betriebssystem und Compiler mit Angabe der jeweiligen Version. Wurde das Programm auf mehreren unterschiedlichen Rechnern entwickelt, sind die unterschiedlichen Konfigurationen anzugeben.
2. **Problemanalyse und Realisation:**
   Folgendermaßen können die erwarteten Informationen übersichtlich strukturiert werden (Grundsätzlich gelten diese Ausführungen auch für andere (ähnliche) technische Dokumentationen, in denen dem Leser Hintergrundwissen zur Lösung einer Aufgabenstellung vermittelt werden soll):
   - Sinnvollerweise startet man mit einer Auflistung der zu erfüllenden Aufgaben, wobei auf einen ausreichenden Detaillierungsgrad geachtet werden sollte. Die einzelnen Punkte ergeben sich aus der Aufgabenstellung (und werden ggf. in Diskussion mit dem Auftraggeber weiter verfeinert).
   - Die Aufstellung wird nach Sachgebieten, Modulen, übergeordneten Teilaufgaben o.ä. gruppiert und ggf. Detailpunkte zusammengefasst. Hiermit erhält man eine Art „Checkliste der zu erfüllenden Aufgaben“, die während der Projektphase durchaus noch modifiziert werden kann.
   - Damit verfügt man über ein Dokument (evtl. Teil eines Lasten-/Pflichtenheftes), das die wesentlichen Problemstellungen darstellt --> Problemanalyse.
   - In der nun folgenden Realisationsanalyse werden für jeden einzelnen Teilaspekt ein oder mehrere Lösungswege aufgezeigt und diskutiert (Pro und Contra). Das passiert häufig ohne Festlegung auf eine bestimmte Programmiersprache (hier im Programmierpraktikum gilt natürlich Java!).
   - Aus dieser Diskussion ergibt sich (hoffentlich) die beste aller Lösungen für diesen Problempunkt und man kann den nächsten anpacken… Sind nun alle einzelnen Punkte bearbeitet und steht damit ein Lösungsweg für jeden Teilaspekt fest, so ist das Kapitel „Realisationsanalyse“ (in der ersten Version) komplettiert.
   - Im dritten Unterkapitel steht für jede soeben gefundene (lt. Diskussion beste) Teillösung, die Darstellung der Implementierung im Vordergrund. Charts, Diagramme, Verweise auf andere Kapitel (Programmorganisationsplan, Datenstrukturen etc.) helfen dem Softwareentwickler, der sich in Dein Projekt einarbeiten muss, Deine Gedankengänge nachzuvollziehen. Und schon ist die Realisationsbeschreibung fertig!
   - Die Präsentationsform des Gesamtkapitels obliegt natürlich Dir, sollte aber möglichst übersichtlich und wohlstrukturiert sein. Also beispielsweise: Zu lösendes Teilproblem – pro/contra zu den möglichen Lösungsansätzen – Beschreibung der Realisation. Und dann das gleiche Spiel für die nächsten Punkte. Wählt man diese Darstellungsform, so entfällt für den Leser das leidige Springen zwischen den drei Hauptkapiteln. Zusätzlich kann dann auch die Überschriftenfolge „Problemanalyse“, „Realisationsanalyse“ und „Realisationsbeschreibung“ zu jedem Detailpunkt entfallen.
   - Unabhängig von der gewählten Präsentationsform sollte jeder Teilaspekt schnell zu finden sein: gute Struktur, keine übermäßig langen Texte, Konzentration auf die wesentlichen Fakten, ggf. Aufnahme in das Inhaltsverzeichnis, sinnvolle und eindeutige Überschriften. Eine erste Version dieses Kapitels ist tunlichst vor der eigentlichen Programmierung zu erstellen und als Planungshilfe für das Projekt zu benutzen. Natürlich können nach der Planung beim Programmieren Änderungen und Erweiterungen der ursprünglichen Problemanalyse und Realisation entstehen, die dann in die Endversion des Kapitels mit übernommen werden.

   Um die Überlegungen auch nachvollziehbar zu machen ist eine gute Doku-Struktur mit passenden Überschriften unabdingbar (niemand liest 3-4 Seiten Fließtext).
3. **Beschreibung grundlegender Klassen:**
   Hier werden alle selbst erstellten Klassen, die Du im Programm tatsächlich benutzt, aufgelistet, jeweils mit einer abstrakten Beschreibung der Struktur (bei dynamischen Datenstrukturen wird zwecks besserer Anschaulichkeit zusätzlich eine Grafik verlangt), dem Zweck, dem diese Struktur in dem Programm dient, sowie ihr java-spezifischer Aufbau (= die explizite Typdefinition aus dem Programm). Hier kann gerne eine UML-Notation verwendet werden, ein automatisch generiertes UML-Diagramm ist jedoch nicht

sinnvoll, außer es wurde auf die wesentlichsten Informationen reduziert.
Außerdem wird der Aufbau aller im Programm verwendeten Dateien aufgeführt.
Im Gegensatz zum Kapitel Problemanalyse und Realisation, in dem die Auswahl und deren Begründung einer oder mehrerer Datenstrukturen stattfindet, soll hier der tatsächliche Einsatz und die genaue Zusammensetzung der Datenstrukturen erläutert werden.

4. **Programmorganisationsplan:**
   Stellt grafisch dar, wie sich die Klassen, die ihr verwendet, untereinander aufrufen. Verwendet hier gerne ein UML-Klassendiagramm, welches die im vorigen Punkt umfassend vorgestellten Klassen nur mit ihren wichtigsten Eigenschaften und Methoden enthält. Es ist sinnvoll, in der Grafik die Klassen z.B. durch farbliche Hervorhebung zu gruppieren. Auch können mehrere Teilbereiche jeweils unter Weglassung verwendeter Klassen dargestellt werden.
   Wurde ein Tool zur Erstellung des Klassendiagramms verwendet, so ist dieses anzugeben. Die Ausgabe eines Tools muss auf jeden Fall so bearbeitet werden, dass die Programmorganisation einfach zu erfassen ist.
   Zusätzlich zur Grafik sollte ein Fließtext oder eine tabellarische Darstellung die Grafik und ihre Aufteilung beschreiben.
5. **Programmtests:**
   Während und nach der Programmerstellung muss das Programm auf Herz und Nieren getestet werden! Dieser Vorgang soll nicht willkürlich geschehen. Dieser Testvorgang soll systematisch durchgeführt und dokumentiert werden.
   Es müssen Testfälle für alle Funktionalitäten, Bedienmöglichkeiten und -fehler und alle Eventualitäten erdacht werden. Die Tests verlaufen also systematisch nach einem Vorgehensplan (Für jedes Eingabefeld z.B. werden also alle möglichen Konstellationen inkl. Falscheingaben getestet).
   Da das Programm von Euch selbst erstellt wurde, ist es Euch möglich Testfälle abzuleiten, bei denen besondere Probleme erwartet werden. Es soll genügen, wenn nur diese speziellen Fälle dokumentiert werden (Programm kann blockieren, mögliche Dateninkonsistenzen, Verletzung von weiteren Bedingungen durch fehlerhafte Eingabe z.B. 'Div by zero'). Dazu ist es aber notwendig, diese Fälle zu erkennen (am Besten gleich beim Programmieren).
   Diese Testfälle stellt Ihr in diesem Kapitel in einer Tabelle, bestehend aus den Spalten 'Testfall', 'erwartetes Ergebnis' und 'erzieltes Ergebnis' dar. In der Spalte 'Testfall' steht jeweils eine detaillierte Beschreibung des Testfalls (so gut beschreiben, dass ich ihn leicht nachstellen kann). In der Spalte 'erwartetes Ergebnis' steht jeweils die Reaktion des Programms auf den Testfall, die Ihr erwartet habt (bei mathematischen Berechnungen also das Ergebnis), bevor Ihr den Testfall getestet habt. In der Spalte 'erzieltes Ergebnis' steht das tatsächliche Ergebnis. Der Inhalt der beiden letzten Spalten wird, wenn alles gut geht, der Gleiche sein. (Bei mathematischen Rechnungen können sich aufgrund der begrenzten numerischen Genauigkeiten Abweichungen ergeben, die nicht auf einen inkorrekten Algorithmus oder fehlerhafte Algorithmusumsetzung zurückzuführen sind. Diese sind dann per Fehlerabschätzung zu begründen.)
   Werden zum Testen externe Dateien (schreibgschützte Datensätze, absichtlich veränderte Dateien, spez. Spielstände etc.) benötigt, so sind diese in der Beschreibung des Testfalles klar zu benennen (und logischerweise mitzuliefern). Ist das Programm durch das Laden von Dateien in der Lage, bestimmte Zustände einzunehmen (z.B. Laden von Spielständen), so muss mit verschiedenen Dateien die Funktionalität wichtiger Algorithmen demonstriert werden.
   Dieses Kapitel hat also zwei Aufgaben, erstens soll es protokollieren, wie der Test organisiert und durchgeführt wurde, zweitens sollen die Testergebnisse dargestellt werden.

Klk - last update 27.08.2015

Kontakt FH Wedel

Staatlich anerkannte
Fachhochschule Wedel
Gemeinnützige Schul-
gesellschaft mbH
Feldstraße 143
D - 22880 Wedel
Tel.: +49 (0)4103 - 8048-0
Fax: +49 (0)4103 - 8048-39
E-Mail: sekretariat@fh-wedel.de