HOCHSCHULE
ESSLINGEN

Nah an Mensch und Technik.

Fakultät Informatik und Informationstechnik

Studiengang Technische Informatik

Arbeit zur Erlangung des akademischen Grades

Bachelor of Engineering

Setzen einer wissenschaftlichen Arbeit mit LaTeX

Max Mustermann

Sommerssemester 2017
Firma: Große Firma GmbH

Betreuer: Dipl. Ing. (FH) Max Betreuer

Erstprüfer: Prof. Dr. Hans Wissenschaftler
Zweitprüfer: Prof. Dr. Walter Forscher

Gewidmet den Besuchern des LaTeX Ferienkurses. Auf dass diese Vorlage zu einer guten Note beiträgt.

„Premature optimization is the root of all evil.“

Donald Knuth

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Esslingen, 2. August 2023	
Ort, Datum	Max Mustermann

Sperrvermerk

Das vorliegende Dokument enthält vertrauliche Daten der Firma „FIRMENNAME und GESCHÄFTSFORM“. Veröffentlichungen oder Vervielfältigungen des vorliegen Dokuments, auch nur auszugsweise, sind ohne ausdrückliche Genehmigung der Firma „FIRMENNAME und GESCHÄFTSFORM“ nicht gestattet. Das Dokument ist lediglich den betreuenden Professoren zugänglich zu machen. Ohne schriftliche Genehmigung der Firma darf dieses Dokument nicht in der Bibliothek der Hochschule ausgelegt werden.

Danksagung

Ich danke allen Besuchern des LaTeX Ferienkurses, sowie Professor Dausmann für die Möglichkeit diesen zu betreuen. Weiterhin danke ich Donald Knuth für die Erfindung von TeX, sowie Leslie Lamport für seine tollen TeX Makros (a.k.a. LaTeX).

Inhaltsverzeichnis

1. Einleitung

In diesem Kapitel wird auf die Motivation der Arbeit eingegangen und die Aufgabenstellung erklärt.

1.1. Motivation

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

1.2. Aufgabenstellung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss

keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2. Grundlagen

In diesem Kapitel werden die wichtigen Grundlagen für diese Aufgabenstellung erläutert.

2.1. Die Programmiersprache Java

Da das Projekt in Java realisiert wird, wird hier ein kurzer Überblick gegeben. Java kann auch Rechnen, somit ist die Lösung für Gleichung 2.1 auf Seite 3 nicht weit entfernt [3] [1].

$$x = 1 + y \ : \ y = 7 \tag{2.1}$$

2.1.1. Klassen

In Listing 2.1 auf Seite 3 ist eine einfache Java Klasse zu sehen. In den letzten Jahren hat sich die Bezeichnung **POJO** für einfache Klassen eingebürgert. Eine ausführliche Einführung in Java und die **Objekt Orientierte Programmierung (OOP)** ist in *Java als erste Programmiersprache* von Joachim Goll, Cornelia Heinisch und Frank Müller enthalten [2].

```
1 class Simple
2 {
3         private String text;
4
5         public Simple(String text) {
6                 this.text = text;
7         }
8
9         public void printText() {
```

```
10                  System.out.println(text);
11          }
12  }
```

Listing 2.1: Eine einfache Java Klasse

2.1.2. Java Logo

In Abbildung 2.1 auf Seite 4 ist das Java Logo abgebildet. Die **JRE**, sowie der **JDK**, verwenden dieses Logo an vielen Stellen. Es ist auch sonst in vielen Programmen und Internet Seiten zu sehen.

Abbildung 2.1.: Das Java Logo

Es gibt aber auch andere Logos, die mit Java in Zusammenhang stehen. Ein Beispiel ist das Java Maskottchen Duke, das in Abbildung 2.2 auf Seite 6 gezeigt wird.

2.1.3. Zugriffsschutz

Um Teile einer Klasse gegen unbefugten Zugriff zu schützen besitzt Java die in Tabelle 2.1 auf Seite 5 enthaltenen **Modifizierer**.

Das Prinzip des **Information Hiding** schreibt vor dass man den Zugriff soweit wie Möglich einschränkt.

2.1.4. Die Klasse `Object`

In Java erben alle Klassen automatisch von der Klasse `Object`. Einige wichtige Methoden der Klasse `Object` sind in Tabelle 2.2 auf Seite 6 aufgelistet.

Es sollten im Normalfall mindestens die Methoden `toString()` und `equals()` überschrieben werden. Das Überschreiben von `clone()` kann aber – je nach Anwendung – auch Sinn machen. Beim Überschreiben von Methoden sollte man die Annotation `@Over- ride` verwenden, da sie einen davon abhält die Methode versehentlich nur zu überladen.

Modifier	**Wirkung**
`private`	Privat
`public`	Öffentlich
`protected`	Eingeschränkt
`final`	Nicht änderbar
...	...

Tabelle 2.1.: Modifizierer für den Zugriffsschutz

Methode	Funktion
`toString()`	Darstellung als `String`
`equals()`	Vergleiche mit anderen Objekten
`hashCode()`	Hashcode erzeugen (Siehe `Hashtable`)
`clone()`	Exakte Kopie erzeugen
...	...

Tabelle 2.2.: Wichtige Methoden der Klasse `Object`

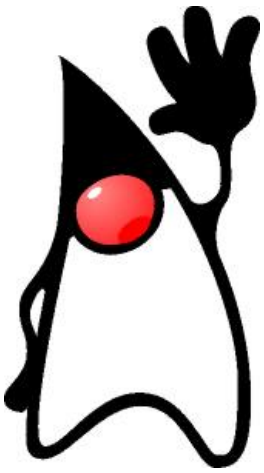

Abbildung 2.2.: Duke, das Java Maskottchen

3. Anforderungen

3.1. Zweck des Systems

Name des Systems: Hier die Bezeichnung des Systems einfügen.

3.2. Funktionales Top-Level Requirement

Name des Systems: Hier die Top-Level-Anforderung einfügen.

3.3. Die geforderten Anwendungsfälle

- Anforderung #1
- Anforderung #2
- ...

3.4. Funktionale Anforderungen

3.4.1. Anforderungen zum Anwendungsfall #1

Anforderung 100: ...

3.5. Nicht funktionalen Anforderungen

3.5.1. Forderungen an die Standardsoftware

Anforderung 200: ...

3.5.2. Forderungen an Zugriffssicherheit

Anforderung 210: ...

3.5.3. Forderungen an die Architektur

Anforderung 220: ...

3.5.4. Forderungen an die Bedienbarkeit

Anforderung 230: ...

3.5.5. Forderungen an die Performance

Anforderung 240: ...

4. Systemanalyse

Abbildung 4.1 zeigt den prinzipiell Ablauf der Systemanalyse.

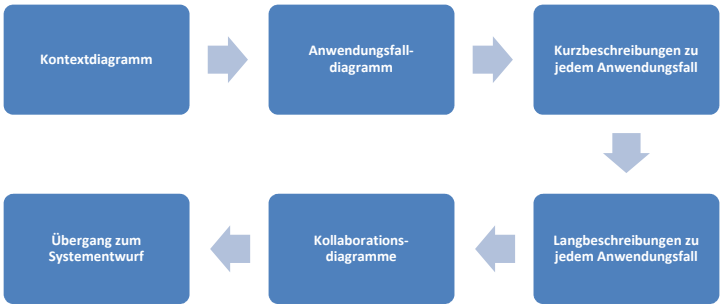

Abbildung 4.1.: Ablauf der Systemanalyse

4.1. Kontextdiagramm

...

4.2. Anwendungsfalldiagramm

...

4.3. Kurzbeschreibungen zu jeden Anwendungsfall

...

5. Systementwurf

5.1. Klassendiagramm

5.2. Logisches Datenmodell

5.3. Physikalisches Datenmodell

5.4. Entwurf der grafischen Oberfläche

6. Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Literaturverzeichnis

[1] Ekbert Hering, Rolf Martin und Martin Stohrer. *Physik für Ingenieure*. Springer Verlag, 2004.

[2] Joachim Goll, Cornelia Heinisch und Frank Müller. *Java als erste Programmiersprache*. Teubner Verlag, 2005.

[3] Lothar Papula. *Mathematische Formelsammlung für Wissenschaftler und Ingenieure*. Vieweg Verlag, 2003.

[4] Manfred Dausmann, Ulrich Bröckl und Joachim Goll. *C als erste Programmiersprache*. Teubner Verlag, 2005.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

A. Anhang zum Systementwurf

Allgemeine Beschreibung des Anhangs

A.1. Diagramme

Hier werden Diagramme platziert, die in den Textkapitel zuviel Platz beanspruchen.

A.2. Tabellen

Hier werden Tabellen platziert, die in den Textkapitel zuviel Platz beanspruchen.

A.3. Quellcodelistings

Hier werden Tabllen platziert, die in den Textkapitel zuviel Platz beanspruchen.