

Logo  
Projektpartner



## Diplomarbeit

# Der Titel der Arbeit

**Untertitel der Arbeit**

Yasmin Musterfrau      Max Mustermann

Author Nachname

Imst, 4. April 2017

Betreut durch:

Alexander Scharmer

Claudio Landerer

Stefan Stolz

# **Eidesstattliche Erklärung**

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbst verfasst und keine anderen als die angeführten Behelfe verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht. Ich bin damit einverstanden, dass meine Arbeit öffentlich zugänglich gemacht wird.

---

Ort, Datum

---

David Wille

---

Patrick Wilhelm

# **Abnahmeerklärung**

Hiermit bestätigt der Auftraggeber, dass das übergebene Produkt dieser Diplomarbeit den dokumentierten Vorgaben entspricht. Des Weiteren verzichtet der Auftraggeber auf unentgeltliche Wartung und Weiterentwicklung des Produktes durch die Projektmitglieder bzw. die Schule.

---

Ort, Datum

---

Auftraggeber

# **Vorwort**

z. B. Hinweise, wie das bearbeitete Thema gefunden wurde oder Dank für die Betreuung (Kooperationspartner/in, Betreuer/innen, Sponsoren) etc.

# **Abstract (Deutsch)**

(ca. ½ bis max. 2 Seiten) Kurzbeschreibung von Aufgabenstellung und Problemlösung.

# **Abstract (Englisch)**

(ca. ½ bis max. 2 Seiten)

# Inhaltsverzeichnis

<b>1. Projektmanagement</b>	<b>12</b>
1.1. Metainformationen . . . . .	12
1.1.1. Team . . . . .	12
1.1.2. Betreuer . . . . .	12
1.1.3. Partner . . . . .	12
1.1.4. Ansprechpartner . . . . .	12
1.2. Vorerhebungen . . . . .	13
1.2.1. Projektzieleplan . . . . .	13
1.2.2. Projektumfeld . . . . .	13
1.2.3. Risikoanalyse . . . . .	14
1.3. Pflichtenheft . . . . .	14
1.3.1. Zielbestimmung . . . . .	14
1.3.2. Produkteinsatz und Umgebung . . . . .	15
1.3.3. Funktionalitäten . . . . .	15
1.3.4. Testszenarien und Testfälle . . . . .	15
1.3.5. Liefervereinbarung . . . . .	16
1.4. Planung . . . . .	16
1.4.1. Projektstrukturplan . . . . .	16
1.4.2. Meilensteine . . . . .	16
1.4.3. Gant-Chart . . . . .	16
1.4.4. Abnahmekriterien . . . . .	17
1.4.5. Pläne zur Evaluierung . . . . .	17
1.4.6. Ergänzungen und zu klärende Punkte . . . . .	17
<b>2. Vorstellung des Produktes</b>	<b>18</b>
<b>3. Eingesetzte Technologien</b>	<b>19</b>

<b>4. Problemanalyse</b>	<b>20</b>
4.1. USE-Case-Analyse . . . . .	20
4.2. Domain-Class-Modelling . . . . .	21
4.3. User-Interface-Design . . . . .	21
<b>5. Systementwurf</b>	<b>22</b>
5.1. Architektur . . . . .	22
5.1.1. Design der Komponenten . . . . .	22
5.1.2. Benutzerschnittstellen . . . . .	23
5.1.3. Datenhaltungskonzept . . . . .	23
5.1.4. Konzept für Ausnahmebehandlung . . . . .	23
5.1.5. Sicherheitskonzept . . . . .	23
5.1.6. Design der Testumgebung . . . . .	23
5.1.7. Design der Ausführungsumgebung . . . . .	24
5.2. Detailentwurf . . . . .	24
<b>6. Implementierung</b>	<b>26</b>
<b>7. Deployment</b>	<b>27</b>
<b>8. Tests</b>	<b>28</b>
8.1. Systemtests . . . . .	28
8.2. Akzeptanztests . . . . .	28
<b>9. Projektevaluation</b>	<b>29</b>
<b>10. Benutzerhandbuch</b>	<b>30</b>
<b>11. Betriebswirtschaftlicher Kontext</b>	<b>31</b>
<b>12. Zusammenfassung</b>	<b>32</b>
<b>13. Beispielkapitel</b>	<b>33</b>
13.1. Beispiele zitieren . . . . .	33
13.1.1. Beispiele Abbildungen . . . . .	34
13.2. Beispiele Listen . . . . .	35
13.3. Beispiel Codesequenz . . . . .	37
13.3.1. Quicksort in JAVA . . . . .	37

*Der Titel der Arbeit*

13.4. Beispieltext . . . . .	39
<b>Abbildungsverzeichnis</b>	<b>41</b>
<b>Tabellenverzeichnis</b>	<b>42</b>
<b>Quelltexte</b>	<b>43</b>
<b>Literaturverzeichnis</b>	<b>44</b>
<b>A. Anhang-Kapitel</b>	<b>46</b>
A.1. Anhang-Section . . . . .	46

# **Einleitende Bemerkungen**

# **Notationen**

Beschreibung wie Code, Hinweise, Zitate etc. formatiert werden

# **1. Projektmanagement**

## **1.1. Metainformationen**

### **1.1.1. Team**

David Wille Patrick Wilhelm

### **1.1.2. Betreuer**

Dominik Neuner

### **1.1.3. Partner**

Hak Imst

### **1.1.4. Ansprechpartner**

Dominik Neuner

## 1.2. Vorerhebungen

### 1.2.1. Projektzieleplan

Projektziele-Hierarchie - SMART Ist Zustand: Aktueller Stand ist das es mehrere Webseiten existieren jedoch hat keine Seite die Funktionen die wir benötigen.

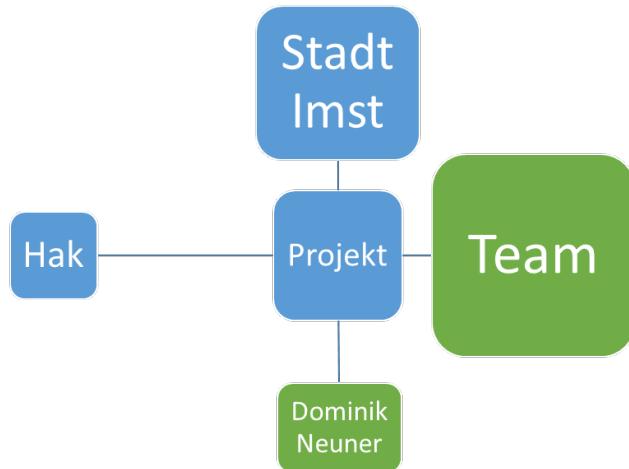
Es sollte eine Website erstellt werden wo die meisten Sehenswürdigkeiten angeführt sind und eventuell, wenn man eine Kooperation mit den Hotels eingeht das man sie in einer Map mit den Sehenswürdigkeiten einbindet. Innerhalb 2 Wochen einen Prototyp der Seite und die Organisation und Einteilungen der Programmierer erstellen. Nach einem Monat sollte die Datenbank Einbindung vorhanden sein. Ein Programmierer Kosten in der Stunde zirka 25€, wenn man 5 Programmierer Anstellt sollte das Projekt in zwei Monaten realisierbar sein. Kosten zirka: 60 000

### 1.2.2. Projektumfeld

- Identifikation der Stakeholder

STAKEHOLDER	EINFLUSS	NÄHE	EINSTELLUNGEN	BESCHREIBUNG
Team	groß	nahe	positiv	Arbeiter
Dominik Neuner	moderat	mittel	positiv	Koordinator
HAK	gering	fern	neutral	Raumeinteilung
Stadt Imst	groß	nah	neutral	Auftraggeber

- Charakterisierung der Stakeholder
- Maßnahmen
- Grafische Darstellung des Umfeldes



### 1.2.3. Risikoanalyse

- Risikomatrix

## 1.3. Pflichtenheft

### 1.3.1. Zielbestimmung

- Projektbeschreibung: Es soll eine Applikation entwickelt werden, bei der Fragen (zu einem Thema) und zugehörigen möglichen Antworten (mit einer richtigen) erstellt werden können. Jeder Frage muss ein Standort auf der Karte zugewiesen werden. Im Front-End können Fragen im Umkreis aufgelistet und gelöst werden. Für jede gelöste Frage erhöht sich der Punktestand des Benutzers.
- IST-Zustand
- SOLL-Zustand: Der SOLL-Zustand besteht daraus, dass die Projektbeschreibung erreicht wird.
- NICHT-Ziele (Abgrenzungskriterien): mehrere richtige Antworten, Punktestand steigt nicht, Standort ist nicht zugewiesen

### **1.3.2. Produkteinsatz und Umgebung**

- Anwendungsgebiet: Ein Gebiet, das auf der Karte gut zu sehen ist, das interessant für die Zielgruppen ist und das auch einverstanden mit dieser Anwendung ist.
- Zielgruppen: Geologen, Geo-Begeisterte, Touristen, kleinere Gruppen
- Betriebsbedingungen
- Hard-/Softwareumgebung: Internet, Android-Gerät, GPS

### **1.3.3. Funktionalitäten**

- MUSS-Anforderungen
  - Funktional: GPS, Internet, Akku
  - Nicht-funktional: schöne Umgebung
- KANN-Anforderungen
  - Funktional: Bewertung, teilen, kommentieren
  - Nicht-funktional: Benutzerfreundlichkeit

### **1.3.4. Testszenarien und Testfälle**

- Beschreibung der Testmethodik: Testen ob alles angezeigt wird und ob der Punktestand aktualisiert wird.
- Testfall 1: Kleinere Details testen
- Testfall 2: Das ganze Programm testen
- ...

### **1.3.5. Liefervereinbarung**

- Lieferumfang: App
- Modus
- Verteilung(Deployment)

## **1.4. Planung**

### **1.4.1. Projektstrukturplan**

Einteilung der Arbeiter (Zeit, Gehalt), Ausführung, Kontrolle, Funktionalität und Qualität, Testdurchlauf, Behebung von Fehlern, Kunden präsentieren

### **1.4.2. Meilensteine**

Projektname, Projektnummer (Bei mehreren Projekten im Unternehmen), Meilensteinnummer, Projektstrukturplan-Code, Meilensteinname, Plantermine, Angepasste Plantermine (Name der Person die Termine geändert hat), Ist-Termin, Version, Datum, Ersteller

### **1.4.3. Gant-Chart**

Phase 1: 50 Tage

Phase 2: 30-40 Tage

Projektabschluss: maximal 5-10 Tage

#### **1.4.4. Abnahmekriterien**

sind Kriterien, die erfüllt sein müssen, damit der Auftraggeber zur Abnahme einer Projektleistung verpflichtet ist (Leistungsanforderungen und weitere wesentliche Bedingungen wurden erfüllt)

#### **1.4.5. Pläne zur Evaluierung**

#### **1.4.6. Ergänzungen und zu klärende Punkte**

## **2. Vorstellung des Produktes**

Vorstellung des fertigen Produktes anhand von Screenshots, Bildern, Erklärungen.

### **3. Eingesetzte Technologien**

- Kurzbeschreibung aller Technologien, die verwendet wurden.
- Technologien die aus dem Unterricht bekannt sind, nur nennen und deren Einsatzzweck im Projekt beschreiben, nicht die Technologien selbst.
- Technologien die aus dem Unterricht nicht bekannt sind, im Detail beschreiben incl. deren Einsatz im Projekt
- Fokus aus eingesetzten Frameworks

# **4. Problemanalyse**

## **4.1. USE-Case-Analyse**

- UseCases auf Basis von Benutzerzielen identifizieren:
  - Benutzer eines Systems identifizieren
  - Benutzerziele identifizieren (Interviews)
  - Use-Case-Liste pro Benutzer definieren
- UseCases auf Basis von Ereignissen identifizieren:
  - Externes Event triggert einen Prozess
  - zeitliches Event triggert einen Prozess (Zeitpunkt wird erreicht)
  - State-Event (Zustandsänderung im System triggert einen Prozess)
- Werkzeuge:
  - USE-Case-Beschreibungen (textuell, tabellarisch)
  - USE-Case-Diagramm
  - Aktivitätsdiagramm für den Use-Case (Interaktion zwischen Akteur und System abbilden)
  - System-Sequenzdiagramm (Spezialfall eines Sequenzdiagramms: Nur 1 Akteur und 1 Objekt, das Objekt ist das komplette System, es geht um die Input/Output Requirements, die abzubilden sind)

## **4.2. Domain-Class-Modelling**

- "Dinge" (Rollen, Einheiten, Geräte, Events etc.) identifizieren, um die es im Projekt geht
- ER-Modellierung oder Klassendiagramme
- Zustandsdiagramme (zur Darstellung des Lebenszyklus von Domain-Klassen darstellen)

## **4.3. User-Interface-Design**

- Mockups
- Wireframes

# **5. Systementwurf**

## **5.1. Architektur**

### **5.1.1. Design der Komponenten**

Darstellung und Beschreibung der Systemarchitektur;

- statische Zerlegung des Systems in seine physischen Bestandteile (Komponenten, Komponentendiagramm)
- (textuelle) Beschreibung des dynamischen Zusammenwirkens aller Komponenten
- (textuelle) Beschreibung der Strategie für die Architektur, d. h. wie die Architektur in Statik und Dynamik funktionieren soll.
- Verwendung von Referenzarchitekturen bzw. Architekturmustern (als Schablonen, z.B. MVC, Plugin, Pipes and Filters)
  - MVC
  - Schichten
  - Pipes
  - Request Broker
  - Service-Oriented

### **5.1.2. Benutzerschnittstellen**

- Design des UIs
- Dialoge, Dialogsteuerung, Ergonomie, Gestaltung, Eingabeüberprüfungen

### **5.1.3. Datenhaltungskonzept**

- Design der Datenbank (ER-Modell)
- Design des Zugriffs auf diese Daten (Datenhaltungskonzept)
- Caching, Transaktionen

### **5.1.4. Konzept für Ausnahmebehandlung**

- Systemweite Festlegung, wie mit Exceptions umgegangen wird
- Exceptions sind primär aus den Bereichen UI, Persistenz, Workflow-Management

### **5.1.5. Sicherheitskonzept**

Beschreibung aller sicherheitsrelevanten Designentscheidungen

- Design der Security-Elemente
- Design von Safety-Elementen (Fehlertoleranz, Verfügbarkeit etc.)

### **5.1.6. Design der Testumgebung**

- wie wird getestet (Unit-Testing, Integrationstesting, Systemtests, Akzeptanztests)

- Testumgebung, Testprozess, Teststrategie, Testmethoden, Testfälle

### **5.1.7. Desing der Ausführungsumgebung**

- Deployment (DevOps)
- Betrieb (besonders Hoch- und Hertunerfahren der Anwendung)

## **5.2. Detailentwurf**

Design jedes einzelnen USE-Cases

- Design-Klassendiagramme vom Domain-Klassendiagramm ableiten (incl. detaillierter Darstellung und Verwendung von Vererbungshierarchien, abstrakten Klassen, Interfaces)
- Sequenzdiagramme vom System-Sequenz-Diagramm ableiten
- Aktivitätsdiagramme
- Detaillierte Zustandsdiagramme für wichtige Klassen

Verwendung von CRC-Cards (Class, Responsibilities, Collaboration) für die Klassen

- um Verantwortlichkeiten und Zusammenarbeit zwischen Klassen zu definieren und
- um auf den Entwurf der Geschäftslogik zu fokussieren

Design-Klassen für jeden einzelnen USE-Case können z.B. sein:

- UI-Klassen
- Data-Access-Klassen
- Entity-Klassen (Domain-Klassen)
- Controller-Klassen

## *Der Titel der Arbeit*

- Business-Logik-Klassen
- View-Klassen

Optimierung des Entwurfs (Modularisierung, Erweiterbarkeit, Lesbarkeit):

- Kopplung optimieren
- Kohäsion optimieren
- SOLID
- Entwurfsmuster einsetzen

# 6. Implementierung

Detaillierte Beschreibung der Implementierung aller Teilkomponenten der Software entlang der zentralsten Use-Cases:

- GUI-Implementierung
- Controllerlogik
- Geschäftslogik
- Datenbankzugriffe

Detaillierte Beschreibung der Teststrategie (Testdriven Development):

- UNIT-Tests (Funktional)
- Integrationstests

Zu Codesequenzen:

- kurze Codesequenzen direkt im Text (mit Zeilnummern auf die man in der Beschreibung verweisen kann)
- lange Codesequenzen in den Anhang (mit Zeilennummer) und darauf verweisen (wie z.B. hier ??)

## **7. Deployment**

- Umsetzung der Ausführungsumgebung
- Deployment
- DevOps-Thema

# **8. Tests**

## **8.1. Systemtests**

Systemtests aller implementierten Funktionalitäten lt. Pflichtenheft

- Beschreibung der Teststrategie
- Testfall 1
- Testfall 2
- Tesfall 3
- ...

## **8.2. Akzeptanztests**

# **9. Projektevaluation**

siehe Projektmanagement-Unterricht

# **10. Benutzerhandbuch**

falls im Projekt gefordert

# **11. Betriebswirtschaftlicher Kontext**

BW-Teil

## **12. Zusammenfassung**

- Etwas längere Form des Abstracts
- Detaillierte Beschreibung des Outputs der Arbeit

# **13. Beispielkapitel**

## **13.1. Beispiele zitieren**

Das ist ein Zitat mit Klammern, (Resnick, 1996), das ein Zitat ohne Klammern: Harel und Papert (1991). Hier das selbe Zitat mit einer Seitenangabe und Klammern (Resnick, 1996, S. 23).

Wird ein Absatz aus einer Quelle sinngemäß übernommen (nicht wörtlich), dann kann nach dem Absatz das entsprechende Zitat in Klammern angeführt werden. (Anastopoulou u.a., 2012, S. 33)

Wenn ein Zitat im Text angegeben wird, wie z.B. so Beer, Rudolf und Benischek, Isabella (2011), können die Klammern weggelassen werden.

Der folgende Absatz zeigt ein Blockzitat (wörtlich übernommene Textpassage aus einer Quelle):

Dr. Heinrich Faust ist ein angesehener Wissenschaftler und Akademiker, der trotz seiner wissenschaftlichen Studien und einer guten Bildung seinen Wissensdurst nicht stillen kann. Eines Nachts sitzt er in seinem Studierzimmer und grübelt über den Sinn des Lebens nach, findet jedoch keine Antworten. Daraufhin wendet er sich der Geisterwelt zu. Er beschwört einen Erdgeist, versucht sich den Geistern gleich zu stellen, was ihm jedoch nicht gelingt. Von Ohnmacht getrieben will er sich das Leben nehmen. Sein Selbstmordversuch wird jedoch von Glockenläuten zum Ostertag und seinen Kindheitserinnerungen gestört. (Ackermann,

2001, S. 21)

Hier wird ein wörtliches Zitat inline angegeben: „Das ist ein kleines direktes Zitat.“ Göhlich und Zirfas (2007), und danach geht es gleich wieder direkt weiter. Ob ein wörtliches Zitat inline oder als eigener Block angezeigt wird, entscheidet Latex auf Basis der Länge.

### **13.1.1. Beispiele Abbildungen**

Auf diese Weise kann man zum Beispiel in Latex auf die Abbildung 13.1 verweisen. Die Kennung für den Verweis vergibt man selbst mit dem „label“ Kommando bei der Abbildung.

Jede Abbildung muss nicht nur mindestens einen Verweis im Text haben. Es wird außerdem eine Bildunterschrift verlangt. Für diese ist festgesetzt, dass die Abbildungsunterschrift alleine ausreichend sein muss, um zu verstehen, was am Bild zu erkennen ist.

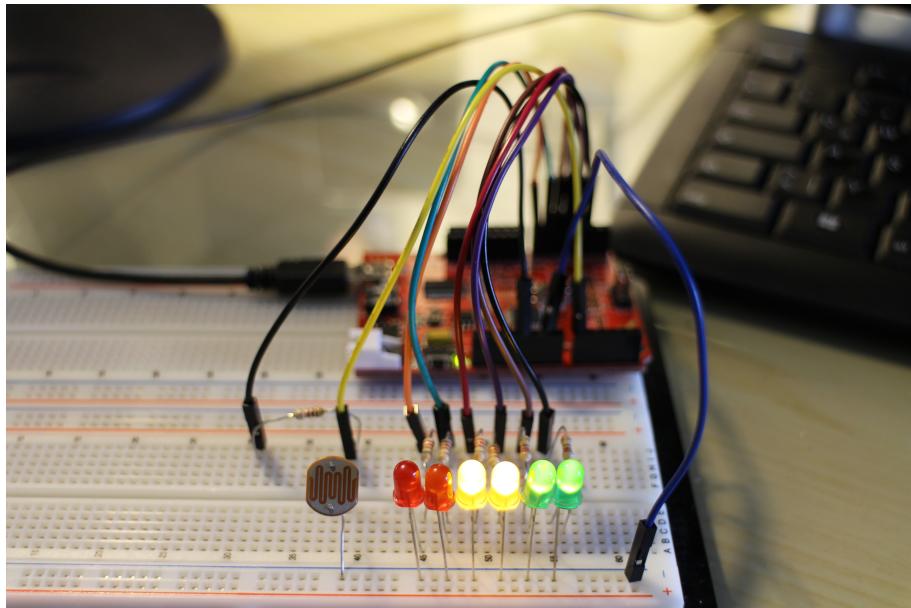
Der nächste wichtige Punkt sind die Quellenangaben bei Abbildungen. Der Author muss zu jeder Abbildung die notwendigen Rechte haben und idealer Weise gibt man diese bei der Abbildung mit an. In Abbildung 13.1 auf Seite 35 sieht man das.

Es ist wichtig zu verstehen, dass Latex die Positionierung von Abbildungen übernimmt. Man definiert die Abbildung über begin-figure dort, wo man die Abbildung in etwa haben möchte, den Rest übernimmt Latex

### **Beispiele Tabellen**

Tabelle 13.1 ist ein Beispiel für eine aufwändigere Tabelle mit einer Abbildung und Überschrift.

Tabellen sind in Latex sehr kompliziert zu erzeugen. Alternativ kann man die Tabel-



© Stefan Stolz (CC BY-SA 3.0)

Abbildung 13.1.: Hintergrund: Arduino Board; Vordergrund: eine Lichterreihe und ein Lichtsensor (Fotowiderstand); In diesem Beispiel wird die Lichterreihe je nach Helligkeit des Umgebungslichtes gesteuert. Durch leichte Modifikationen kann man damit eine Lichtschranke oder auch eine Helligkeitssteuerung für das Smartphone simulieren.

len auch in einem anderen Programm gestalten und als Bild wieder einfügen. Dieses Bild kann dann innerhalb von begin-Table verwendet werden.

## 13.2. Beispiele Listen

Im Folgenden wird eine Liste gezeigt:

- Ich weiß, dass viele Geräte des täglichen Lebens durch Computer gesteuert werden und kann für mich relevante nennen und nutzen.
  1. Und jetzt eine Numerierung
  2. Und jetzt eine Numerierung
- Ich kann wichtige Bestandteile eines Computersystems (Eingabe-, Ausgabe-geräte und Zentraleinheit) benennen, kann ihre Funktionen beschreiben und diese bedienen.

# Der Titel der Arbeit

DW OR N PACKAGE (TOP VIEW)	
NC	1
V <sub>CC</sub>	2
SER IN	3
DRAIN0	4
DRAIN1	5
DRAIN2	6
DRAIN3	7
SRCLR	8
G	9
GND	10
	20
	19
	18
	17
	16
	15
	14
	13
	12
	11

NC – No internal connection

*V<sub>cc</sub>* Positive supply voltage  
GND Ground  
SER IN Daten Pin  
SRCK Clock Pin  
RCK Latch Pin  
*SRCLR* Wenn **shift-register clear** LOW ist, werden die input Register gelöscht  
*G* Wenn **output enable** HIGH ist, werden die Daten im Output Buffer LOW gehalten

Tabelle 13.1.: Aufwändige Tabelle mit Abbildung und Caption

Und jetzt eine Numerierung:

1. Aufzählungspunkt
  - a) Unteraufzählung
    - Und jetzt noch eine Ebene ohne Aufzählung
    - Und jetzt noch eine Ebene ohne Aufzählung
  - b) Unteraufzählung
2. Aufzählungspunkt
3. Aufzählungspunkt
4. Aufzählungspunkt
5. Aufzählungspunkt

### 13.3. Beispiel Codesequenz

In Listing 13.1 sieht man ein Quick-Sort-Listing in der Programmiersprache JAVA. Das Listings-Paket übernimmt die Formatierung von Codebausteinen und kann in der Präambel nach Belieben auf eine andere Sprache konfiguriert werden.

#### 13.3.1. Quicksort in JAVA

Quelltext 13.1: QuickSort in Java

```
1 public class QuickSort
2 {
3     public static void main(String[] args)
4     {
5         int [] x =
6         {
7             9, 2, 4, 7, 3, 7, 10
8         }
9         ;
10        System.out.println(Arrays.toString(x));
11
12        int low = 0;
13        int high = x.length - 1;
14
15        quickSort(x, low, high);
16        System.out.println(Arrays.toString(x));
17    }
18
19    public static void quickSort(int[] arr, int low,
20                                int high)
21    {
22        if (arr == null || arr.length == 0)
23            return;
```

```
24     if (low >= high)
25         return;
26
27     // pick the pivot
28     int middle = low + (high - low) / 2;
29     int pivot = arr[middle];
30
31     // make left < pivot and right > pivot
32     int i = low, j = high;
33     while (i <= j)
34     {
35         while (arr[i] < pivot)
36         {
37             i++;
38         }
39
40         while (arr[j] > pivot)
41         {
42             j--;
43         }
44
45         if (i <= j)
46         {
47             int temp = arr[i];
48             arr[i] = arr[j];
49             arr[j] = temp;
50             i++;
51             j--;
52         }
53     }
54
55     // recursively sort two sub parts
56     if (low < j)
57         quickSort(arr, low, j);
58 }
```

```
59     if (high > i)
60         quickSort(arr, i, high);
61     }
62 }
```

## 13.4. Beispieltext

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.

Das hier ist der zweite Absatz. 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.

Und nun folgt – ob man es glaubt oder nicht – der dritte Absatz. 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 gleich-

## *Der Titel der Arbeit*

gültig, 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.

Nach diesem vierten Absatz beginnen wir eine neue Zählung. 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.

# **Abbildungsverzeichnis**

13.1. Arduino mit Lichtsensor und Lichterkette . . . . .	35
--	----

# **Tabellenverzeichnis**

13.1. Aufwändige Tabelle mit Abbildung und Caption . . . . .	36
--	----

# Quelltexte

# Literaturverzeichnis

- [Ackermann 2001] ACKERMANN, Edith: Piaget's constructivism, Papert's constructionism: What's the difference. In: *Future of learning group publication* 5 (2001), Nr. 3, S. 438. – URL [http://lovettresourcenetwork.wiki.lovett.org/file/view/EA.Piaget+\\_+Papert.pdf](http://lovettresourcenetwork.wiki.lovett.org/file/view/EA.Piaget+_+Papert.pdf). – Zugriffssdatum: 2014-07-09
- [Anastopoulou u. a. 2012] ANASTOPOULOU, Stamatina ; BERLAND, Matthew ; FRANT, Janete B. ; BOYTCHEV, Pavel ; BRENNAN, Karen ; CHRONAKI, Anna ; CLAYSON, James ; CORREIA, Secundino ; DAGIENE, Valentina ; DEKOLI, Margarita: Constructionism 2012 Theory Practice and Impact. (2012), August. – URL [http://users.uoa.gr/~zsmyrnaiou/conferences\\_after2008/constructionism%201\\_2012.pdf](http://users.uoa.gr/~zsmyrnaiou/conferences_after2008/constructionism%201_2012.pdf). – Zugriffssdatum: 2014-03-26
- [Beer, Rudolf und Benischek, Isabella 2011] BEER, RUDOLF ; BENISCHEK, ISABELLA: Aspekte kompetenzorientierten Lernens und Lehrens. In: BIFIE (Hrsg.): *Kompetenzorientierter Unterricht in Theorie und Praxis*. Graz : Leykam, 2011
- [Göhlich und Zirfas 2007] GÖHLICH, Michael ; ZIRFAS, Jörg: *Lernen: Ein pädagogischer Grundbegriff*. Stuttgart : Kohlhammer, April 2007. – ISBN 9783170188693
- [Harel und Papert 1991] HAREL, Idit ; PAPERT, Seymour: *Situating Constructionism*. Norwood, N.J : Ablex Publishing Corporation, U.S., 1991. – ISBN 9780893917869
- [Resnick 1996] RESNICK, Mitchel: Distributed constructionism. In: *Proceedings of the 1996 international conference on Learning sciences*, International Society of

*Der Titel der Arbeit*

the Learning Sciences, 1996, S. 280–284. – URL <http://dl.acm.org/citation.cfm?id=1161173>. – Zugriffsdatum: 2015-04-20

# **A. Anhang-Kapitel**

## **A.1. Anhang-Section**

Testtext