**Java Projekt   
3. Semester**

**Vokalbeltrainer**

|  |  |
| --- | --- |
| Version: | 1.0 |
| Druckdatum | Montag, 16. März 2015 |
| Projekt: | Erstellung eines Vokabeltrainers für die Firma speedspeak |
| Autor: | , Elias Schwarz, Daniel Hafner |
| Firma: | IT-B 13-16 |

**Typographische Vorlagen**

**Haupttitel**

Haben die Schriftfarbe grau und sind mit der Option „Fett“, 14 pt und „Unterstrichen“ hervorgehoben

**Übertitel**

Haben die Schriftfarbe dunkelorange und sind mit der Option „Fett“ mit der Grösse 12 pt hervorgehoben.

**Untertitel**

Haben die Schriftfarbe schwarz und sind mit der Option „Fett“ und der Grösse 10pt hervorgehoben.



Dieses Symbol steht für eine wichtige Information.



Dieses Symbol steht für einen Verweis zu einer Internetseite.



Dieses Symbol steht für einen Verweis innerhalb des Dokuments.

Zur Verbesserung der Handhabung sind die oben genannten typographischen Vorlagen für diesen Bericht definiert worden.

**Impressum**

**©2015, Ruel Holderegger / Elias Schwarz / Daniel Hafner - Alle Rechte vorbehalten**

**Titelbild**: Quellenangaben siehe Quellenverzeichnis

**Grafiken und Tabellen:** Erstellung mit Microsoft Office 2013**Inhaltsverzeichnis:**

1 Einleitung 4

1.1. Zweck 4

1.2. Ausgangslage 4

2 Analyse 5

2.1 Muss-Kriterien 5

2.1.1 Funktionalität 5

2.1.2 Benutzeroberfläche 5

2.1.3 Technische Aspekte 6

2.2 Abgrenzungskriterien 6

2.3 Geschäftsprozesse / Usecase 7

2.3.1 Übersicht 7

2.3.2 Geschäftsprozesse im Detail 7

3 Umsetzung 13

3.1 Erster GUI Entwurf 13

3.2 Anwendungsfalldiagramm 13

3.3 Klassendiagramm 15

3.4 Arbeit als Gruppe 16

3.4.1 Parallelität 16

3.4.2 Um ein mehrspuriges Programmieren zu ermöglichen, wurde das Projekt in mehrere Pakete aufgeteilt. Wie sich herausstellte haben sich Pakete überschnitten, weshalb wir die Aufgabenteilung anpassen mussten 16

3.4.3 Regelmässige Besprechungen 16

3.4.4 Regelmässige Besprechungen sollen sicherstellen, dass Problemstellungen so rasch wie möglich gelöst und allfällige konzeptionelle Änderungen vorgenommen werden können. 16

1.3. Testen der Anwendung 16

1.1.1. Testumgebung 16

1.1.2. Testprotokoll 16

3.5 Projekt-Tagebuch 17

3.6 Fazit 19

3.6.1 Ruel 19

3.6.2 Elias 19

3.6.3 Daniel 20

4 Literaturverzeichnis und Quellenangaben 21

4.1 Quellenangaben 21

4.2 Literaturverzeichnis 21

5 Anhang 22

5.1 API des Projekts 22

# Einleitung

## Zweck

Dieses Dokument beschreibt das Vorgehen und die Umsetzung des Java-Projekts „Vokabeltrainer“ für den Lehrgang Techniker HF Informatik am ZbW. Zielgruppe dieses Dokuments sind Benutzer des Programms, Entwickler die zukünftig an dem Projekt weiter arbeiten sowie die Lehrkräfte.

## Ausgangslage

Ausgangslage für das Projekt „Vokabeltrainer“, ist ein fiktiver im Informatik-Unterricht gestellter Kunden-Auftrag für die Entwicklung einer Sprach-Lernsoftware. Zudem galt es gewisse technische und organisatorische Vorgaben zu erfüllen.

Hauptziel dieses Projekts besteht darin, gelerntes Wissen zu vertiefen, sowie dieses als Prüfungsvorbereitung anzuwenden.

# Analyse

## Muss-Kriterien

### Funktionalität

|  |  |  |
| --- | --- | --- |
|  | **Kriterium** | **Beschreibung / Lösung** |
| 1 | Programmablauf | Der Programmablauf soll soweit der klassischen Lernkartei von Sebastian Leitner entsprechen. Bei Erfolg wird eine Karte ein Fach höher gesteckt, bei Misserfolg fällt die Karte zurück ins erste Fach. |
| 2 | Logik | Der Benutzer bestimmt selbst, ob er die Karte richtig oder falsch beantwortet hat. |
| 3 | Flexibilität | Es soll möglich sein neue Karten zu erfassen, editieren oder zu löschen. |
| 4 | Mehrsprachigkeit | Die Menüsprache kann gewechselt werden - ohne dass das Programm neu gestartet werden muss - zwischen Deutsch, Englisch, Französisch oder Italienisch. |
| 5 | Lern-Verlauf | Der Benutzer kann wählen, welches Fach er abarbeiten will, sofern sich Karten in einem Fach befinden. Somit wird auf den Wissensstand des Benutzers eingegangen. |
| 6 | Austausch zwischen Benutzern | Eine Kartei samt Lernstand soll in eine Datei exportiert werden können. Ebenfalls sollen exportierte Dateien ins Programm importiert werden können. Der Lernstand kann auch zurückgesetzt werden. |

### Benutzeroberfläche

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Kriterium** | **Beschreibung / Lösung** |
| 1 | Anlehnung an physische Karte | Damit der Lernprozess möglichst der klassischen Lernkartei entspricht, wird zu Beginn die Vorderseite einer Karte angezeigt. |
| 2 | Karte drehen | Mit Klick auf eine Schaltfläche oder auf die Karte, wird die Rückseite angezeigt. |
| 3 | Integration in OS | Grundsätzlich ist das Regelwerk des Betriebssystems zu beachten. |
| 4 | Bedienung | Die Bedienungsoberfläche ist auf Mausbedienung auszulegen.  Gewisse Funktionen wie das Wechseln von Vor- auf die Rückseite und umgekehrt (Karten verwalten) lassen sich mit der Taste TAB durchführen. |

### Technische Aspekte

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Kriterium** | **Beschreibung / Lösung** |
| 1 | Plattformunabhängigkeit | Das Programm soll plattformunabhängig betrieben werden können. |
| 2 | Datenträger | Das Programm soll direkt ab USB-Stick ausgeführt werden können. |

## Abgrenzungskriterien

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Kriterium** | **Beschreibung** |
| 1 | Anzahl Fächer | Die maximale Anzahl Fächer wird fix auf 7 Fächer beschränkt. |
| 2 | Inhalt | Einer Karteikarte können keine Bilder zugeordnet werden. |
| 3 | Sammlungs-Begrenzung | In der Software können nicht mehr als eine Kartei gleichzeitig geführt werden. Kartei muss exportiert und eine andere importiert werden. |
| 4 | Suchfunktion | Es wird keine Suchfunktionen innerhalb der Software bereitgestellt. |
| 5 | Software-Update | Es werden keine Update-Funktionen zur Verfügung gestellt. |
| 6 | Datenbank-Unabhängig | Für die Speicherung der Daten kann keine Datenbank verwendet werden. Die Daten müssen über die integrierte Export-Funktion in Form eines Exportfiles abgelegt werden. |
| 7 | Einzelplatz | Die Software stellt eine Einzelplatz-Anwendung dar. |
| 8 | Kopieren / Einfügen | Es wird keine „Copy/Paste“ Funktionalität zur Verfügung gestellt. |
| 9 | Schriftarten und Grössen | Jegliche Schriftarten und Schriftgrössen sind fix vorgegeben. |
| 10 | Hilfe-Funktion | Es werden keine Hilfe-Stellungen innerhalb der Applikation zur Verfügung gestellt. (zB Hilfe-F1) |

## Geschäftsprozesse / Usecase

### Übersicht

|  |  |
| --- | --- |
| **Nr.** | **Beschreibung** |
| P1 | Menu-Sprache ändern |
| P2 | Kartei bearbeiten |
| P3 | Karteikarte erstellen |
| P4 | Karteikarte auswählen |
| P5 | Karteikarte bearbeiten |
| P6 | Karteikarte löschen |
| P7 | Kartei Importieren (inkl Lernstand) |
| P8 | Kartei Exportieren (inkl Lernstand) |
| P9 | Lernen starten |
| P10 | Register auswählen |
| P11 | Rückseite anzeigen |
| P12 | Auswahl: Antwort korrekt |
| P13 | Auswahl: Antwort falsch |
| P14 | Beenden und speichern |
| P15 | Beenden ohne zu speichern |

### Geschäftsprozesse im Detail

|  |  |
| --- | --- |
| P1 – Student wählt Menüsprache | |
| Geschäftsprozess | Student wählt Menüsprache |
| Ziel | Menüsprache wird ohne Neustart gewechselt. |
| Vorbedingung | Programm wurde gestartet |
| Nachbedingung Erfolg | Sprache ist auf Zielsprache umgestellt. |
| Nachbedingung Fehlschlag | - |
| Akteur | Student |
| Auslösendes Ereignis | Sprache <<xy>> wurde angewählt |
| Beschreibung | Student soll aus einer Combobox eines der folgenden Sprachen auswählen können:  Deutsch  Französisch  Englisch  Italienisch |

|  |  |
| --- | --- |
| P2 – Karte bearbeiten | |
| Geschäftsprozess | Karte bearbeiten |
| Ziel | Karteninformationen (Fragewort oder Antwort) ist angepasst. |
| Vorbedingung | Karte vorhanden |
| Nachbedingung Erfolg | Fragewort oder Antwort wurde angepasst. |
| Nachbedingung Fehlschlag | Meldung an Benutzer, dass Karte schon vorhanden ist (Antwort oder Fragewort) |
| Akteur | Student |
| Auslösendes Ereignis | Button „Karte editieren“ wurde gedrückt. |
| Beschreibung | Bestehende Kartendaten können editiert werden (Fragewort und Antwort) |

|  |  |
| --- | --- |
| P3 – Neue Karte erstellen | |
| Geschäftsprozess | Neue Karte erstellen |
| Ziel | Karte mit Frage und Lösung ist erfasst. |
| Vorbedingung | Karte ist noch nicht vorhanden. |
| Nachbedingung Erfolg | Karte ist angelegt. |
| Nachbedingung Fehlschlag | Meldung an Benutzer, dass Karte bereits vorhanden ist, oder Lösungstext nicht angegeben worden ist. |
| Akteur | Student |
| Auslösendes Ereignis | Button „Neue Karte“ gedrückt. |
| Beschreibung | Eingeben Kartendaten (Fragewort, Antwort) |

|  |  |
| --- | --- |
| P4 – Karteikarte auswählen | |
| Geschäftsprozess | Karteikarte auswählen |
| Ziel | Es kann durch die Karten geblättert werden in dem Karten-Editor |
| Vorbedingung | Karten-Editor geöffnet und mehr als eine Karte vorhanden |
| Nachbedingung Erfolg | Karte gefunden |
| Nachbedingung Fehlschlag | Buttons für das Blättern sind ausgeblendet, da weniger als 2 Karten verfügbar |
| Akteur | Student |
| Auslösendes Ereignis | Button „<-“ oder „->“ gedrückt. |
| Beschreibung | Durch Karteikarten blättern |

|  |  |
| --- | --- |
| P5 – Karteikarte bearbeiten | |
| Geschäftsprozess | Karteikarte bearbeiten |
| Ziel | Frage und/oder Lösung einer Karteikarte wurde angepasst |
| Vorbedingung | Mindestens eine Karte ist vorhanden |
| Nachbedingung Erfolg | Karte wurde bearbeitet und gespeichert |
| Nachbedingung Fehlschlag | Keine Karte vorhanden die bearbeitet werden könnte |
| Akteur | Student |
| Auslösendes Ereignis | Im Karten-Editor in eines der Textfelder geklickt |
| Beschreibung | Bearbeiten der Kartendaten (Frage, Antwort) |

|  |  |
| --- | --- |
| P6 – Karte löschen | |
| Geschäftsprozess | Karte löschen |
| Ziel | Karte wird gelöscht |
| Vorbedingung | Karte vorhanden |
| Nachbedingung Erfolg | Karte ist nicht mehr vorhanden |
| Nachbedingung Fehlschlag | - |
| Akteur | Student |
| Auslösendes Ereignis | Button „Karte löschen“ wurde gedrückt. |
| Beschreibung | Karte samt Kartendaten (Fragewort und Antwort) werden aus der Kartei entfernt. |

|  |  |
| --- | --- |
| P7 – Karten importieren | |
| Geschäftsprozess | Kartei Importieren (inkl Lernstand) |
| Ziel | Eine Kartei inkl Lernstand aus einer csv-Datei ins Programm importiert |
| Vorbedingung | Korrekt formatierte Csv-Datei vorhanden |
| Nachbedingung Erfolg | Kartei im Programm vorhanden |
| Nachbedingung Fehlschlag | Fehlermeldung informiert über Fehlschlag des Imports |
| Akteur | Student |
| Auslösendes Ereignis | Button „Importieren“ im „Datei“-Menu wurde gedrückt |
| Beschreibung | Kartei aus Datei importieren |

|  |  |
| --- | --- |
| P8- Katei exportieren | |
| Geschäftsprozess | Kartei exportieren (inkl Lernstand) |
| Ziel | Eine Kartei inkl Lernstand in eine csv-Datei aus dem Programm exportiert |
| Vorbedingung | Gewählter Dateiname noch nicht vorhanden im selben Zielort |
| Nachbedingung Erfolg | Kartei in Datei gespeichert |
| Nachbedingung Fehlschlag | Fehlermeldung informiert über Fehlschlag des Exports |
| Akteur | Student |
| Auslösendes Ereignis | Button „Exportieren“ im „Datei“-Menu wurde gedrückt |
| Beschreibung | Kartei in Datei exportieren |

|  |  |
| --- | --- |
| P9 – Lernen starten | |
| Geschäftsprozess | Lernen starten |
| Ziel | Erste Karte wird angezeigt |
| Vorbedingung | Mindestens eine Karte vorhanden |
| Nachbedingung Erfolg | Erste Karte wird angezeigt |
| Nachbedingung Fehlschlag | Leere Karte wird angezeigt |
| Akteur | Student |
| Auslösendes Ereignis | Button „Lernen starten“ im „Datei“-Menu wurde gedrückt |
| Beschreibung | Lernen starten |

|  |  |
| --- | --- |
| P10 – Student wählt Fach | |
| Geschäftsprozess | Student wählt Fach |
| Ziel | Fach, in welchem gelernt werden soll, wird bestimmt (je weiter hinten das Wort, desto einfacher der Schwierigkeitsgrad) |
| Vorbedingung | mind. 2 und mehr Fächer sind mit Karten gefüllt. |
| Nachbedingung Erfolg | Schwierigkeit (Fach) ist festgelegt. |
| Nachbedingung Fehlschlag | Meldung an Benutzer, dass nur ein oder kein Fach mit Karten gefüllt ist. |
| Akteur | Student |
| Auslösendes Ereignis | Schwierigkeitsgrad wurde angewählt |
| Beschreibung | Student soll wählen können, ob er die schwierigen (Stufe 1 – 2), mittelschweren (Stufe 3 – 4) oder einfachen (Stufe 5 – 7) |

|  |  |
| --- | --- |
| P11 – Rückseite anzeigen | |
| Geschäftsprozess | Rückseite anzeigen |
| Ziel | Rückseite der Karte anzeigen |
| Vorbedingung | Karte muss vorhanden sein |
| Nachbedingung Erfolg | Rückseite wird angezeigt |
| Nachbedingung Fehlschlag | - |
| Akteur | Student |
| Auslösendes Ereignis | Klick auf Karte |
| Beschreibung | Student dreht Karte. |

|  |  |
| --- | --- |
| P12 – Student beantworte Frage korrekt | |
| Geschäftsprozess | Student beantwortet Frage korrekt |
| Ziel | Student beantwortet Frage korrekt |
| Vorbedingung | Karte vorhanden |
| Nachbedingung Erfolg | Karte wird ins nachfolgende Fach verlegt |
| Nachbedingung Fehlschlag | - |
| Akteur | Student |
| Auslösendes Ereignis | Button „Richtig“ wurde angewählt |
| Beschreibung | Student hat die richtige Antwort im Kopf beantwortet und klickt den Button „Richtig“ an. |

|  |  |
| --- | --- |
| P13- Student beantworte Frage falsch | |
| Geschäftsprozess | Student beantwortet Frage falsch |
| Ziel | Student beantwortet Frage falsch |
| Vorbedingung | Karte vorhanden |
| Nachbedingung Erfolg | Karte ins 1. Fach gelegt. |
| Nachbedingung Fehlschlag | - |
| Akteur | Student |
| Auslösendes Ereignis | Button „Falsch“ wurde angewählt |
| Beschreibung | Student hat die falsche Antwort im Kopf beantwortet und klickt den Button „Falsch“ an. |

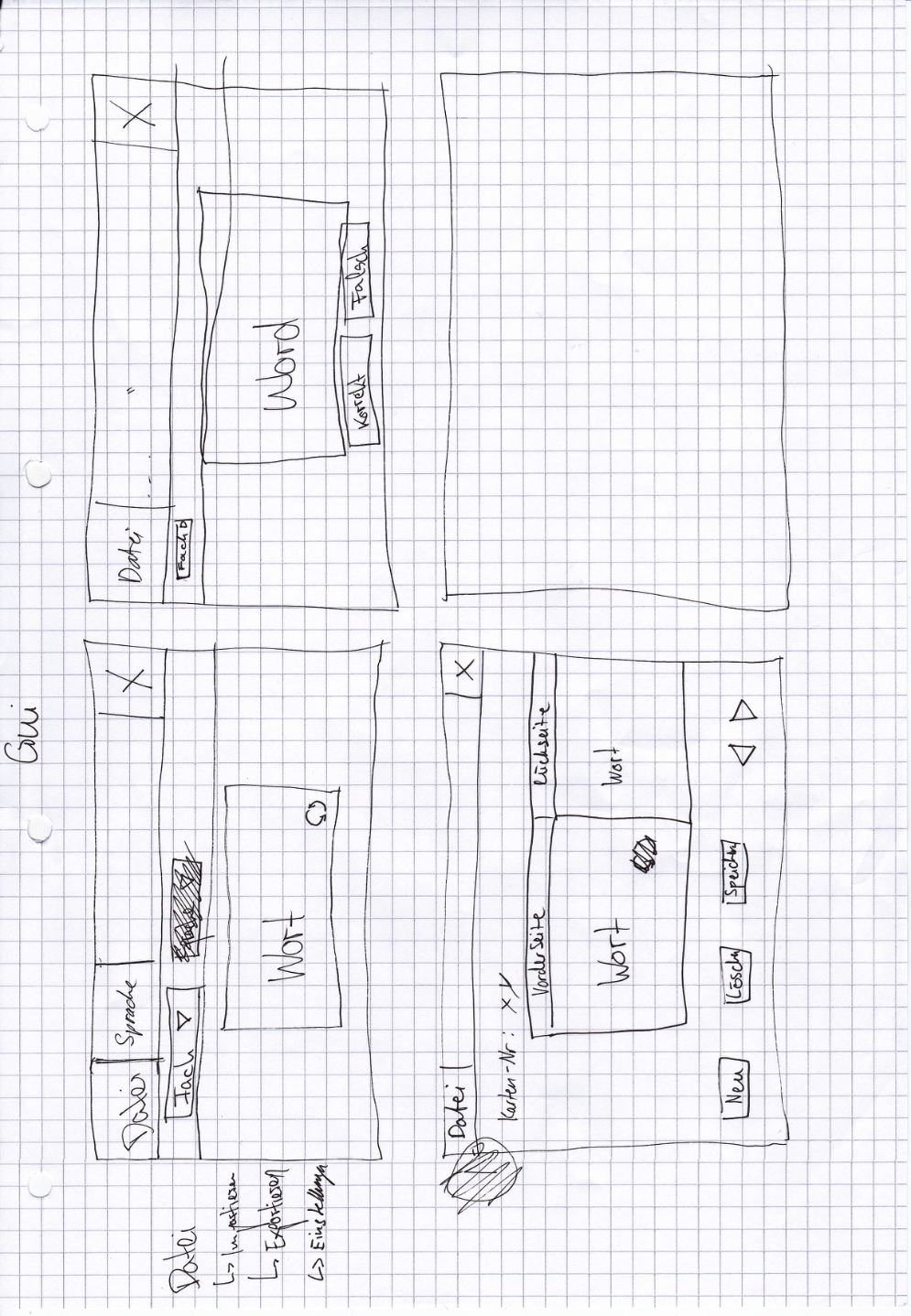
|  |  |
| --- | --- |
| P14 – Beenden und speichern | |
| Geschäftsprozess | *Beenden und speichern* |
| Ziel | Programm beendet und Karten werden in die interne csv-Datei gespeichert werden. |
| Vorbedingung | Programm gestartet. |
| Nachbedingung Erfolg | Programm beendet, Kartei gespeichert |
| Nachbedingung Fehlschlag | - |
| Akteur | Student |
| Auslösendes Ereignis | Programm beenden |
| Beschreibung | Programm wird beendet mit der Auswahl zu speichern. |

|  |  |
| --- | --- |
| P15: Beenden ohne zu speichern | |
| Geschäftsprozess | *Beenden ohne zu speichern* |
| Ziel | Programm beendet |
| Vorbedingung | Programm gestartet |
| Nachbedingung Erfolg | Programm beendet |
| Nachbedingung Fehlschlag | - |
| Akteur | Student |
| Auslösendes Ereignis | Programm beenden |
| Beschreibung | Programm wird beendet. |

# Umsetzung

## Erster GUI Entwurf

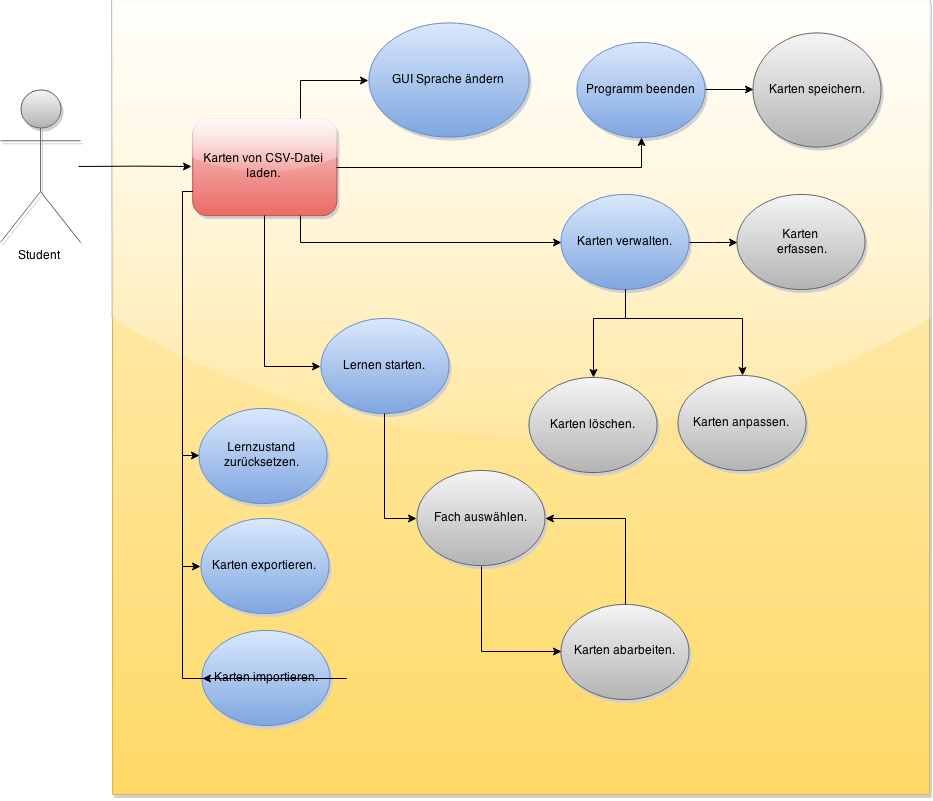
Nachdem wir uns auf den Programmablauf geeinigt hatten, erstellten wir einen ersten GUI-Entwurf.



## Aufbau der grafischen Benutzeroberfläche

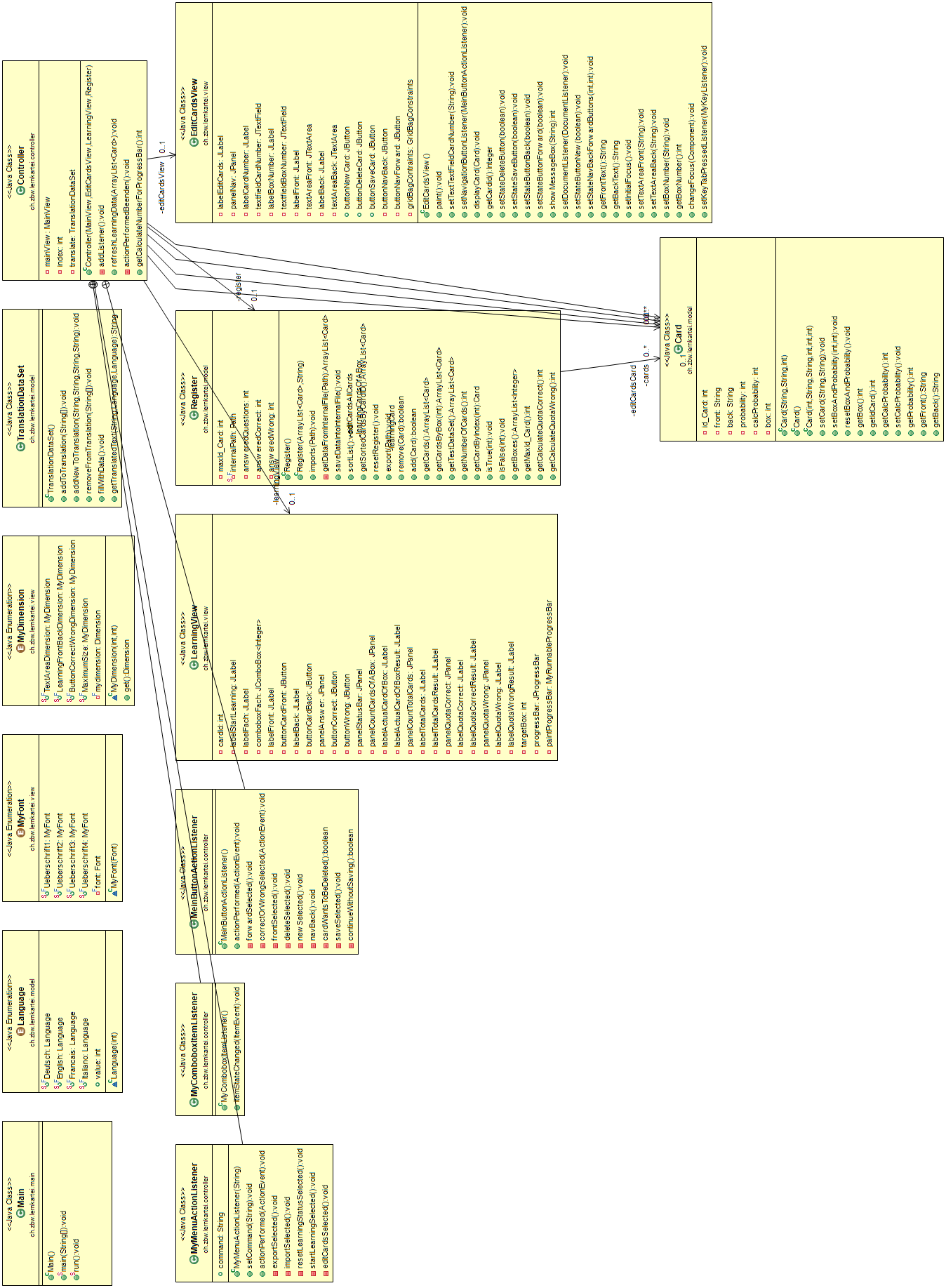
## Anwendungsfalldiagramm

Das Anwendungsfalldiagramm zeigt die Funktionalitäten der Anwendung.



## Klassendiagramm

Nach der Fertigstellung des Projektes zeigen sich die implementierten Klassen wie folgt:



## Arbeit als Gruppe

### Parallelität

### Um ein mehrspuriges Programmieren zu ermöglichen, wurde das Projekt in mehrere Pakete aufgeteilt. Wie sich herausstellte haben sich Pakete überschnitten, weshalb wir die Aufgabenteilung anpassen mussten

### Regelmässige Besprechungen

### Regelmässige Besprechungen sollen sicherstellen, dass Problemstellungen so rasch wie möglich gelöst und allfällige konzeptionelle Änderungen vorgenommen werden können.

## Testen der Anwendung

### Testumgebung

Die nachfolgenden Testfälle wurden ab einem USB-Stick durchgeführt.

Die Voraussetzungen für den Start der Anwendung:

* JRE 1.7 muss installiert sein
* Ordern src muss vorhanden sein
* Datei data.csv muss vorhanden sein
* Die Anwendung lernkartei.jar muss im gleichen Verzeichnis liegen wie der Ordner src

### Testprotokoll

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr.** | **Zweck des Tests** | **Vorgehen** | **Resultat** |
| 1 | Export der Karten soll funktionieren | Karten exportieren:  Datei – Export | Export funktioniert |
| 2 | Import der Karten soll funktionieren | Karten importieren: Datei – Import | Import funktioniert |
| 3 | Karte erstellen und speichern soll funktionieren | Neue Karte erstellen und speichern. Nummer prüfen, Text prüfen. | Wenn eine leere Karte erstellt wird ohne zu speichern wird sie trotzdem gespeichert. |
| 4 | Karte soll gelöscht werden können. | Erste/Letzte Karte löschen. | Lücken in der Kartennummer werden nicht gefüllt. Letzte Karte kann gelöscht werden |
| 5 | In Karte verwalten navigieren. | Navigieren mit löschen, Karte erstellen prüfen. Buttons prüfen | Funktioniert. |
| 6 | Fach soll ausgewählt und gewechselt werden können | Fach auswählen nachdem Fragen beantwortet wurden. | Wenn eine Frage falsch beantwortet wird, sollte es möglich sein das erste/das nächste Fach wählen zu können. |
| 7 | Fragen richtig beantworten. | Karte kommt ins nächste Fach (max. 7). | Funktioniert |
| 8 | Frage soll als falsch beantwortet werden können | Frage mit falsch beantworten  Karte kommt ins Fach 1 | Funktioniert |
| 9 | Reihenfolge der Karte soll geprüft werden | Reihenfolge der Karte prüfen (Wahrscheinlichkeit)  Die Fragen die öfters falsch beantwortet wurden sollten eine höhere Wahrscheinlichkeit erhalten. | Funktioniert |
| 10 | Anwendung soll ab einem USB-Stick ausgeführt werden können | Jar-File inkl. Ordner src und data.csv auf einen USB-Stick laden und Programm starten | Funktioniert |
| 11 | Sprache soll gewechselt werden können | Sprache im Menü und auf den unterschiedlichen Fenstern prüfen | Buttons (Richtig und Falsch) und Buttons in den Einstellungen werden nicht übersetzt |
| 12 | Änderungen speichern | Programm beenden und Frage mit JA beantworten  Karten werden ins interne csv gespeichert. | Funktioniert |
| 13 | Änderungen nicht speichern | Programm beenden und Frage mit NEIN beantworten  csv-File wird nicht verändert. | Funktioniert |

Methodisches Testverfahren

Stand: 14.03.2015

## Projekt-Tagebuch

Hier wird der Projekt-Fortschritt und die Kommunikation innerhalb der Gruppe chronologisch aufgezeigt.

|  |  |
| --- | --- |
| **Datum** | **Tätigkeiten** |
| 26.01.2015 | Als erstes haben wir anhand der Anfrage und der Aufgabenstellung die grundsätzlichen Funktionen der Applikation besprochen und diese Schritt für Schritt genauer definiert. Die Interpretation und vor allem die Team-Interne Einigung einzelner Punkte hat einige Zeit in Anspruch genommen.  Einer dieser Diskussions-Punkte war zum Beispiel folgender Satz:  „Gerade in Computerprogrammen lassen sich auch Verfahren anwenden, bei denen jede Karte mit einem Wortpaar einen bestimmten Wert n zugewiesen bekommt, der sich bei Erfolg um eins erhöht, bei Nichterfolg um eins erniedrigt (der Wert ist jedoch minimal 2 und kann die Anzahl der Karten nicht überschreiten).“  Ergänzend zur Funktions-Definition in einer ersten Version unseres Pflichtenheftes haben wir das GUI skizziert.  Entwicklungsumgebung mithilfe von GitHub-Repository eingerichtet. |
| 01.02.2015 | Heute haben wir das Pflichtenheft erstellt. Das etwas trockene dokumentieren von Punkten, wie den Abgrenzungskriterien hat wieder reichlich Stoff für angeregte Diskussionen geboten. Wir konnten uns aber bei allem gegenseitig überzeugen.  Unsere Entwicklungsumgebung hat uns zuerst reichlich Zeit gekostet, weil es ständig Versionskonflikte gab. Nachdem die Repository direkt ins Eclipse importiert wurde, ist das Problem nicht mehr aufgetreten. |
| 02.02.2015 | Zu Beginn der Lektion haben wir das Pflichtenheft gemäss dem Feedback von Herrn Palmer angepasst. Anschliessend haben wir die Klassen und deren Methoden in einem UML-Diagramm erfasst.  Ebenfalls haben wir entschieden unsere Aufgaben in kleine Pakete einzuteilen, um mögliche Probleme effizienter abarbeiten zu können, bevor es möglicherweise zu spät ist. Die Einteilung wurde wie folgt gemacht: Ruel → GUI, Elias → Logik, Daniel → Logik und Dokumentation. |
| 09.02.2015 | In einer Video-Unterhaltung mit Bildschirmfreigabe haben wir offene Fragen geklärt. Wir haben die Import-Funktion vorerst durch einen Test-Konstruktor ersetzt, damit wir die bestehenden Funktionen bereits testen können.  Ruel will bis zum nächsten mal das GUI für das Anzeigen von Karten, den Start Bildschirm und die Karten-Bearbeitungsseite schreiben. Elias wird am Karten-Handling und der Sortierung nach Wahrscheinlichkeit arbeiten. Daniel wird beginnen den Code mit JavaDoc zu kommentieren und schon bald mit der Dokumentation beginnen. |
| 22.02.2015 | Aktueller Stand GUI besprochen. Code review von einzelnen Methoden. Projekt wurde auf den unterschiedlichen Betriebssystemen nicht korrekt abgebildet.  Bei den Mac-Usern wurden die Umlaute nicht korrekt dargestellt. Das Problem konnte behoben werden, indem das Projekt auf den UTF-8-Zeichensatz eingestellt wurde. Dazu kam das Problem, dass eine implementierte Funktion von Ruel bei den Mac-Usern Kompilier-Fehler hervorbrachte. Nach längerer Nachforschung haben wir festgestellt das OSX eine veraltete java-Version installiert hatte. Die Lösung war eine aktuelle Java Runtime Environment (Version 1.7) zu installieren. Die neue Libary musste nur noch dem Projekt angehängt werden. |
| 02.03.2015 | Ruel hat das GUI schon fast fertig implementiert, jedoch haben wir noch keine Funktionen für Richtig und Falsch Buttons implementiert. Elias arbeitet derzeit an der Lern-Logik und Daniel ist an der Import-Funktion dran. |
| 08.03.2015 | Das GUI steht, Ruel hat sogar die Ästhetik noch ausgearbeitet. Auch das Lernen funktioniert - jetzt gehts ans Bugs fixen und ans ‘finetuning’. Register werden zum Beispiel noch mehrfach angezeigt im Dropdown-Menu. Elias hat die Logik der Karten bei Falsch-/ oder Richtig-Beantwortung implementiert. Dazu hat er das Programm getestet, ob es via einem USB-Stick ausführbar ist. Feststellung davon war, dass das eingebettete Bild nicht angezeigt wurde. Nach längere Analyse hat er festgestellt das der Pfad nicht dynamisch implementiert war. Daniel hat die Import und Export-Funktion implementiert, jedoch müssen er und Ruel noch eine Methode implementieren, die zwischen Windows und Unix-Systemen differenzieren kann (Back- und Forward-Slashes im Pfad). |
| 11.03.2015 | Elias und Ruel haben das Zusammenspiel der Logik mit dem GUI noch einmal überdacht. Diese Absprache war sehr wichtig da festgelegt wurde wer von welcher Methode was erwartet. Somit konnten die grössten Bugs der Software gefixt werden. Daniel hat die Dokumentation ergänzt und mit dem Layout der Präsentation begonnen. |
| 13.03.2015 | Ruel hat begonnen die bis jetzt bekannten Bugs zu fixen. Elias hat die Sortierung noch einmal überarbeitet. Dazu hat er das Anwendungsfalldiagramm erstellt. Daniel hat die Differenzierung zwischen Windows- und Unix-Pfad implementiert. |
| 14.03.2015 | Elias hat eine Testfall-Liste erstellt und mithilfe dieser das Programm auf einem USB-Stick getestet. Die Fehler wurden nach Schweregrad eingestuft. Die schwerwiegendsten Mängel wurden gefixt. Ruel hat das Programm mit einer Fortschrittsanzeige erweitert. Daniel hat die Präsentation und die Dokumentation schon nahezu fertig – es fehlen nur noch die Diagramme. |
| 15.03.2015 | Heute haben wir uns bei Ruel getroffen um die Dokumentation zusammen durchzugehen und wo nötig Änderungen vorzunehmen. Wir haben auch die Präsentation noch weiter bearbeitet und die weiteren Schritte für die Präsentation morgen besprochen. |

## Fazit

### Ruel

Rückblickend auf die ganze Arbeit konnte ich viel dabei lernen, wie zukünftig ein Projekt korrekt angegangen werden soll. Im Nachhinein würde ich einige Punkte anders gestalten. Insbesondere für die Erstellung der Dokumentation sollte zukünftig mehr Beachtung geschenkt werden: Es ist wichtig, dass die durchgeführten Tätigkeiten jeweils unmittelbar am Ende eines Arbeitstages aufgeschrieben werden. Nachträgliches Dokumentieren erwies sich als sehr aufwändig und zeitraubend.

### Elias

Zu Beginn des Projekts dachte ich, dass ich den Überblick über das Ganze habe, und hätte nicht gedacht, dass das Projekt so umfangreich werden würde. Als wir dann aber etwas detaillierter über die Einzelheiten diskutiert haben und wir mit den ersten grösseren Problemstellungen konfrontiert wurden wurde mir klar, dass das Ganze komplexer ist als ich zuerst annahm.  Daher war es umso wichtiger, dass wir uns gemeinsam auf ein weiteres Vorgehen einigen konnten.

Über die Aufteilung der Hauptklassen in „GUI“, „Logik“, und „Logik/Dokumentation“ waren wir uns schnell einig. Auch über die Aufteilung der Aufgaben im Team waren wir uns nach einer Diskussion einig.  Als ich mich mit meinem Teil (die Logik) des Vokabeltrainers auseinandersetzte wurde mir schnell klar, dass die Implementierung der Schnittstellen für die anderen Klassen schwieriger sein würde als zuerst angenommen. Im speziellen die Logikklasse war sehr abhängig von anderen Klassen und umgekehrt, was die Zusammenarbeit sehr forderte.

Daraus resultierten viele spannende Diskussionen.  Dass wir das Projekt als Gruppenarbeit angehen durften hatte Vor- und Nachteile. Auf der einen Seite konnten Aufgaben aufgeteilt werden und so Zeit eingespart werden aber auch unsere konstruktiven Ideen zusammengetragen werden, auf der anderen Seite musste für die Planung und Diskussion mehr Zeit eingerechnet werden. Der Aspekt der Zusammenarbeit war eine interessante und bereichernde Erfahrung.  Das Projekt hatte auf mich einmal mehr einen grossen Lerneffekt. Durch das Angehen und Lösen der Probleme konnte ich viel Neues dazulernen, und es war immer wieder ein Erfolgserlebnis wenn ich ein komplexeres Problem lösen konnte.“

### Daniel

Rückblickend auf die ganze Arbeit konnte ich viel dabei lernen, wie zukünftig ein Projekt korrekt angegangen werden soll. Im Nachhinein würde ich einige Punkte anders gestalten. Insbesondere für die Erstellung der Dokumentation sollte zukünftig mehr Beachtung geschenkt werden.

# Literaturverzeichnis und Quellenangaben

## Quellenangaben

|  |  |
| --- | --- |
| **Beschreibung** | **Link** |
| Hilfestellungen Oracle | <http://www.docs.oracle.com> |
| Java ist auch eine Insel | <http://openbook.rheinwerk-verlag.de/javainsel/> |
|  |  |

## Literaturverzeichnis

|  |  |
| --- | --- |
| **Beschreibung** | **Link (falls vorhanden)** |
|  |  |