Karteikartenspiele

Fallstudie und die Technische Dokumentation

Inhalt

Organisationsteil

Projektorganisation

Zielgruppe

Diese Dokumentation gilt als technische Dokumentation und ist an Fachpersonen mit Kenntnissen in den für das Projekt verwendeten Technologien gerichtet. Als Voraussetzung für die Verständlichkeit des Inhaltes werden Kenntnisse in der objektorientierten Programmierung und das Verständnis für die Programmiersprache Java generell vorausgesetzt.

Am Projekt ist nur eine Person beteiligt , die alle Arbeitspakete inklusive Testing bearbeitet. Es besteht keine Rollenverteilung.

Dateiablage

Die Projektbezogenen Dateien werden in zwei Gruppen aufgeteilt.

- Planungs- und Dokumentationsrelevante Dateien
- Projektdateien

Datensicherung

Alle projektbezogene Dateien werden auf dem Rechner gespeichert weil die VM zu oft abstürzen und man es auf dem nicht virtuellen speichermedium hat.

Manuelles Backup auf einer externen Festplatte. Am Ende jeden Arbeitstages wird der gesamte Inhalt des Projektordners (Dokumentation und Workspace) auf eine externen Festplatte kopiert. Dabei wird das aktuelle Datum in den Ordername geschrieben, damit eine übersichtliche Versionierung entstanden.

Arbeitsjournal

Arbeitsjournal: Tag 1

Tabelle 1:Arbeitsjournal Tag 1

Tätigkeiten	**Aufwand geplant (Std)**	**Aufwand effektiv (Std)**
Miralem und ich haben zusammen gecodet, habe Miralem geholfen.	(3h)	(4.5h)
5 Klassen/Fenster erstellt = ExplenationWindow, ReadyWindow, FinaleWindow, MainWindow und QuesttionWindow.	3h	3h
Buttons hinzugefügt zum MainWindow, ExplenationWindow, ReadyWindow	1h	1.5h
Verknüpfung zwischen MainWindow und ExplenationWindow, zwischen MainWindow und ReadyWindow und ReadyWindow und QuesttionWindow gemacht.	0.5h	0.75h
Buttons = QUIT, START, HOW TO DO und BACK funktionsfähig gemacht	X	X
Total:	**4.5h**	**5.25h**
Abweichung SOLL / IST:	**0.75h**	
Probleme		
Es gab Probleme mit den Buttons manchmal haben die Imports gefehlt sowie die Privat Methoden.		
Hilfestellungen		
Hilfestellung war GPT-4		
Reflexion		
Die zweit wo ich mit Miralem verbracht habe hat mich viel weiter im Projekt gebracht und mir Motivation gegeben.		
Nächste Schritte		
Weiter an den anderen Klassen/Fenster arbeiten und Buttons und Verbindung fertigstellen. Timer und Score System erstellen. Abfrage System anfangen.		

Arbeitsjournal: Tag 2

Tabelle 2:Arbeitsjournal Tag 2

Tätigkeiten	**Aufwand geplant (Std)**	**Aufwand effektiv (Std)**
MainWindow ausgeweitert, ExplenationWindow fertig erstellt und Einsatzbereit, ReadyWindow fertig erstellt und Einsatzbereit.	(6h)	(7h)
Abfrage System implementier aber noch nicht ganz einsatzbereit.	3h	3h
Score und Timer System implementier aber noch nicht ganz ausgereift.	2h	2.25h
Verknüpfung für die letzten Klassen/Fenster gemacht.	0.5h	1.75h
Total:	**5.5h**	**7h**
Abweichung SOLL / IST:	**1.5h**	
Probleme		
Es gibt Probleme mit dem Abfrage System. Die Dateien können nicht aufgezeigt werden.		
Hilfestellungen		
Hilfestellung war GPT-4		
Reflexion		
Am zweiten Tag konnte ich mich sehr gut auf das Projekt fokussieren und habe vieles hinbekommen morgen mache ich weiter am Projekt weiter und hoffe das Abfrage System sowie den Score und Timer System komplett hinzubekommen.		
Nächste Schritte		
Abfrage System zum Laufen bringen und Die Dateien sichtbar machen.		

Arbeitsjournal: Tag 3

Tabelle 3:Arbeitsjournal Tag 3

Tätigkeiten	**Aufwand geplant (Std)**	**Aufwand effektiv (Std)**
Habe versucht code hinzufügen um Bilder einzufügen (nicht funktioniert)	1h	3h
Weiter am Score und Time System gearbeitet (am Ende entfernt).	1h	2h
Total:	**2h**	**5h**
Abweichung SOLL / IST:	**3h**	
Probleme		
Hatte Probleme mit Score und Time System hat nie richtig funktioniert.		
Hilfestellungen		
Hilfestellung war GPT-4		
Reflexion		
Am dritten Tag habe ich versucht den Score und Timer System hinzubekommen doch leider hat es für mich nie richtig gepasst darum habe ich sie aus dem Projekt rausgenommen. Trotz Hilfe von GPT-4 und dem Internet habe ich es nicht geschaft die Text Dateien auf den Fenster zu drucken.		
Nächste Schritte		
Ich frage den Herr Sollberger um hilfe		

Arbeitsjournal: Tag 4

Tabelle 4:Arbeitsjournal Tag 4

Tätigkeiten	**Aufwand geplant (Std)**	**Aufwand effektiv (Std)**
Mit der Hilfe von Herr Sollberger habe ich es geschaft Text Dateien auf dem Fenster zu drucken. Nun funktioniert mein Projekt	1h	1h
Button hinzugefügt, damit wenn man fertig mit allen Karteikarten ist man zur letzten Fenster kommt.	1h	1h
Total:	**2h**	**2h**
Abweichung SOLL / IST:	**0h**	
Probleme		
Das einzige Problem war das die Text Datei nicht in die Mitte platziert wird sondern auf der rechten Seite.		
Hilfestellungen		
Herr Sollberger		
Reflexion		
Am 4. Tag haben wir von Herr Sollberger gelernt wie man Unity Test durchführt.		
Nächste Schritte		
Am letzten Tag Unity Test machen.		

Arbeitsjournal: Tag 5

Tabelle 5:Arbeitsjournal Tag 5

Tätigkeiten	**Aufwand geplant (Std)**	**Aufwand effektiv (Std)**
Habe heute die Dokumentation fertig geschrieben.	6h	6h
Unit Test durchgeführt und dokumentiert.	1h	1h
Total:	**7h**	**7h**
Abweichung SOLL / IST:	**0h**	
Probleme		
Hatte am letzten Tag keine Probleme.		
Hilfestellungen		
Herr Sollbergers Doku.		
Reflexion		
Am Letzten Tag habe ich den Unit Test gemacht und habe die Doku fertig geschrieben.		
Nächste Schritte		
Am letzten Tag Unit Test machen.		

Reflektion

Fazit zum Projekt

Das Hauptziel des Projekts ist es, dem Benutzer Fragen in Form von Textdateiinhalten zu stellen und ihn aufzufordern, die richtige Antwort einzugeben, wo die Antwort der Dateiname (ohne Erweiterung) ist.

Persönliches Fazit

Während meiner Zeit, in der ich am Frage-Antwort-System gearbeitet habe, habe ich viele wertvolle Erfahrungen gesammelt und einige Herausforderungen gemeistert. Anfangs erschien mir das Projekt einfach, da die Grundidee klar war. Aber es hat sich als schwieriger erwiesen als ich gedacht habe.

Das Erkennen und Einlesen von Textdateien, das Anzeigen ihres Inhalts und die Implementierung einer effizienten Überprüfungsfunktion waren sicherlich technische Herausforderungen, die mich manchmal an meine Grenzen brachten. Es war jedoch sehr befriedigend, jedes dieser Probleme nacheinander zu lösen.

Abschließend kann ich sagen, dass dieses Projekt nicht nur meine technischen Fähigkeiten, sondern auch meine Soft Skills verbessert hat. Ich bin stolz auf das, was ich erreicht habe, und freue mich darauf, diese Erkenntnisse in zukünftige Projekte einfließen zu lassen.

Tabellenverzeichnis

Abbildungsverzeichnis

Projektteil

Analyse

Sie erhalten den Auftrag, eine Fallstudie zu einer Problemstellung durchzuführen und die daraus resultierende Applikation objektbasiert umzusetzen. In dieser Fallstudie müssen Sie das gegebene Problem analysieren und eine geeignete Lösung entwerfen und modellieren. Es müssen Überlegungen zur Art der Umsetzung und zur Gestaltung der Benutzerschnittstelle angestellt werden.

Ausgangslage (ist-Situation)

Die meisten gratis verfügbaren Karteikarten-Programme bieten nur eine eingeschränkte Funktionalität, um mit verschiedenen Medien (Bild, Text und Ton) Karteikarten zu erstellen. Mit diesen Programmen Vokabeln oder Fachbegriffe mit unterschiedlichen Medien in effektiver Art und Weise zu üben ist daher nur beschränkt möglich. Es gilt nun in dieser Fallstudie eine neue Lösung zu entwerfen, mit der der Benutzer Bilder, Texte oder Audiodateien zu den zu übenden Begriffen wild durchmischt als Karteikarten verwenden kann. Beispiel: Stellen Sie sich vor, der Benutzer möchte englisches Vokabular lernen. Das Programm sollte dem Benutzer dazu Karteikarten anzeigen können, die entweder ein Bild, ein deutsches Wort oder eine

Audiodatei enthalten. Aus diesem muss der Benutzer dann das Lösungswort erkennen und zum Beispiel in einem Textfeld zur Prüfung eingeben. Die Art der Prüfung kann selbstverständlich auch auf andere Weise geschehen. Genau solche Dinge müssen Sie in der Fallstudie erarbeiten, um ein Programm zu erschaffen, das den Benutzer aktiv beim Lernen unterstützt. Wie wäre es beispielsweise mit einem Scoring-System? ... Die "Hinweisdateien" (also eben die Bilder, Texte oder Audiodateien) soll der Benutzer selbst zum Programm hinzufügen können. So kann sich der Benutzer ein eigenes Karteikarten-Set zusammenstellen, aus dem das Programm dem Benutzer dann in zufälliger Reihenfolge Karteikarten anzeigen kann.

Projektziel (soll-Situation)

Dies ist eine Übersicht der Anforderungen an die Lösung. Die Bewertung erfolgt anhand des Bewertungsrasters weiter unten.

Programm

1. Der Benutzer kann Karteikarten mit Bildern, Texten oder Audiodateien erstellen

2. Das Programm unterstützt den Benutzer aktiv beim Lernen Unterstützung

3. Der Benutzer kann die Hinweisdateien und Lösungswörter selbst zum Programm hinzufügen

4. Das Programm prüft die Lösung des Benutzers und gibt diesem Feedback über die Korrektheit der Eingabe

Dokumentation

1. Es muss eine saubere Dokumentation zum gesamten Projekt (inklusive Fallstudie) geführt werden 2. Das Problem muss in der Dokumentation vor der Umsetzung analysiert werden

3. Auf Basis der Analyse wird ein Pflichtenheft mit allen Anforderungen an das Endprodukt erstellt

4. Die Dokumentation muss den gewählten Lösungsansatz in allen Bereichen vollständig beschreiben und begründen

5. Alle Aspekte des Programms müssen in der Dokumentation genauestens beschrieben werden (Was nicht in der Dokumentation steht, existiert nicht und wird nicht bewertet!)

6. Die Klassenstruktur muss anhand eines Klassendiagramms im Voraus geplant werden

7. Die Dokumentation muss alle für das Programm zutreffenden User Stories enthalten

8. Es müssen vor der Umsetzung Mockups für die Benutzerschnittstelle erstellt werden, in der alle Ansichten und Aktionen eindeutig beschrieben werden

9. Sie müssen an jedem Arbeitstag ein ausführliches Arbeitsjournal führen

9.1. Im Arbeitsjournal werden für jeden Arbeitstag alle Tätigkeiten am Projekt und die benötigte Arbeitszeit aufgelistet

9.2. Zudem wird am Ende jedes Arbeitstages eine Reflexion geschrieben, in der die Ereignisse des Tages, Stellen an denen Sie Probleme hatten und Dinge die Sie gelernt haben in ein paar Sätzen zusammengefasst werden

Anforderungen

Dieser Abschnitt behandelt die konkreten Anforderungen an die zu erstellten Lösung.

Tabelle 6: Funktionale Anforderungen

	Anforderungen
1	Man kann Karteikarten mit einem vordefinierten Lösungswort erfassen.
2	Jede Karteikarte hat ein Medium oder mehrere Medien als Hinweis.
3	Als Medium können Text, Bild oder Ton gewählt werden.
4	Die Karteikarten können in zufällige Reihenfolge durchgearbeitet werden.
5	Beim Durcharbeiten wird im Falle von mehreren Medien ein zufälliges Medium für die Karte ausgewählt
6	Die Benutzer kann mithilfe eines Textfeldes seine Antwort gegen die korrekte Lösung prüfen
7	Beim Eingeben der korrekten Antwort innerhalb kürzester Zeit erhält der Benutzer einen Punktebonus
8	Beim Eingeben der falschen Antwort wird ein Punkteabzug vergeben.

Tabelle 7: Nichtfunktionale Anforderungen

	Anforderungen
1	Das Programm muss dem Benutzer beim Lernen unterstützen.
2	Das Programm soll ein einfaches GUI erhalten.
3	Das Programm soll auf macOS und Windows laufen.
4	Die Benutzeroberfläche wird auf Englisch gehalten.

User Stories

Rollen

Das Programm wird nur durch Rollen bedient. Diese Rolle wird schlicht «Benutzer» genannt. Der Benutzer ist es, der die Karteikarten erstellt und anschliessend auch durcharbeitet.

Stories

Tabelle 8: User Story

	User Story
1	Als Benutzer kann ich Karteikarten erstellen, um mit diesen etwas auswendig zu lernen.
2	Als Benutzer kann ich die Karteikarten durchgeben, um diese auswendig zu lernen.
3	Als Benutzer kann ich meine Antwort vom Programm kontrollieren lassen, um zu sehen, ob ich den Begriff richtig gelernt habe.
4	Als Benutzer kann ich am Ende des Durchgehen eine Auswertung sehen, um einen Überblick über meinen Lernfortschritt zu erhalten.
5	Als Benutzer kann ich Punkte erhalten und verlieren, um Feedback zu einer Leistung zu erhalten
6	Als Benutzet kann ich den Prozess des Durchgehen am Ende erneut starten um meine Resultate zu verbessern.

Planung

Funktionen

In diesem Abschnitt werden die Kernfunktionen des Programmes erläutert.

1. Karteikarten erstellen:
Die Karteikarten in Ihrem Projekt entsprechen Textdateien. Jede Datei repräsentiert eine Karteikarte.

Vorgehensweise:

1. Öffnen Sie einen Texteditor Ihrer Wahl (z.B. Notepad unter Windows, TextEdit unter macOS usw.).

2. Schreiben Sie den Inhalt der Karteikarte in die Datei.

3. Speichern Sie die Datei in einem bestimmten Ordner auf Ihrem Computer. Der Dateiname (ohne die .txt-Erweiterung) repräsentiert die "Antwort" oder das Thema der Karteikarte.

Beispiel:
Sie möchten eine Karteikarte zum Thema "Hauptstadt von Deutschland" erstellen. Der Inhalt könnte "Was ist die Hauptstadt von Deutschland?" sein, und Sie könnten die Datei als Berlin.txt speichern.

2. Karteikarten im Programm laden:

Basierend auf dem vorherigen Code, müssten Sie einen Ordner mit den Textdateien (Karteikarten) auswählen. Das Programm würde dann die Textdateien aus diesem Ordner lesen und den Inhalt als Frage anzeigen. Ihre Aufgabe wäre es, die richtige Antwort (d.h. den Dateinamen ohne .txt-Erweiterung) in das bereitgestellte Textfeld einzugeben.

Vorgehensweise:

1. Starten Sie das Programm.

2. Verwenden Sie die Funktion zum Auswählen eines Ordners (falls implementiert) oder stellen Sie sicher, dass der Ordnerpfad im Code korrekt gesetzt ist.

3. Das Programm zeigt den Inhalt der ersten Textdatei (Karteikarte) an.

4. Geben Sie Ihre Antwort in das Textfeld ein und klicken Sie auf den "Check"-Button.

5. Das Programm gibt Feedback, ob die Antwort korrekt oder falsch war, und lädt die nächste Karteikarte.

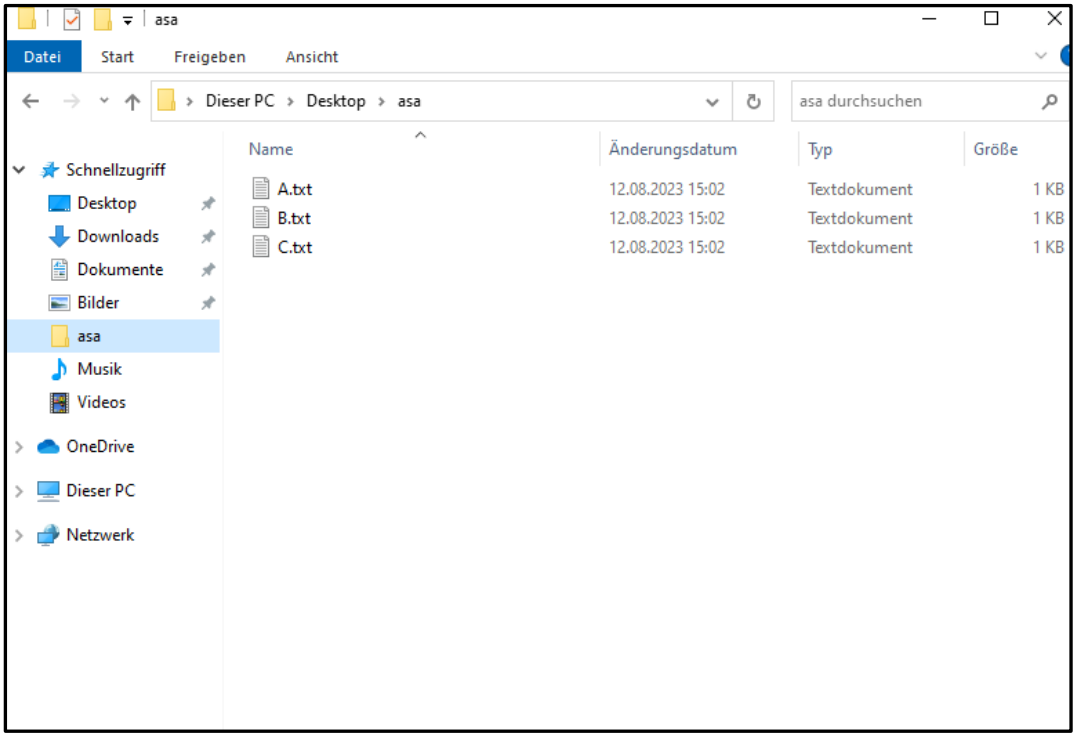

Abbildung 1:Ordner Struktur

Anforderungen an die Struktur :

- Im Kartei-Ordner befindet sich mindestens eine Ordner.
- Im Kartei-Ordner befindet sich kein Unter-Ordner.
- Im jedem Karteikarten-Ordner befindet sich mindesten ein Bild-, Text- oder Audiodatei.

Wenn die Struktur richtig ist, werden die Karteikarten in eine Liste geladen und dem Benutzer ein Startbildschirm angezeigt, auf die Anzahl der geladenen Karteikarten steht und der Lernprozess gestartet werden kann.

Storyboard

In diesem Abschnitt werden alle Fenster des Programms als Mockup dargestellt. Die effektive optische Gestaltung hängt vom auszuführenden System ab.

Alle Fenster des GUIs (mit Ausnahme der System-Dialoge) werden mit einer einheitlichen Grösse gestaltet. Der Benutzer kann die Fenstergrösse allerding Belieben anpassen.

Hauptmenü

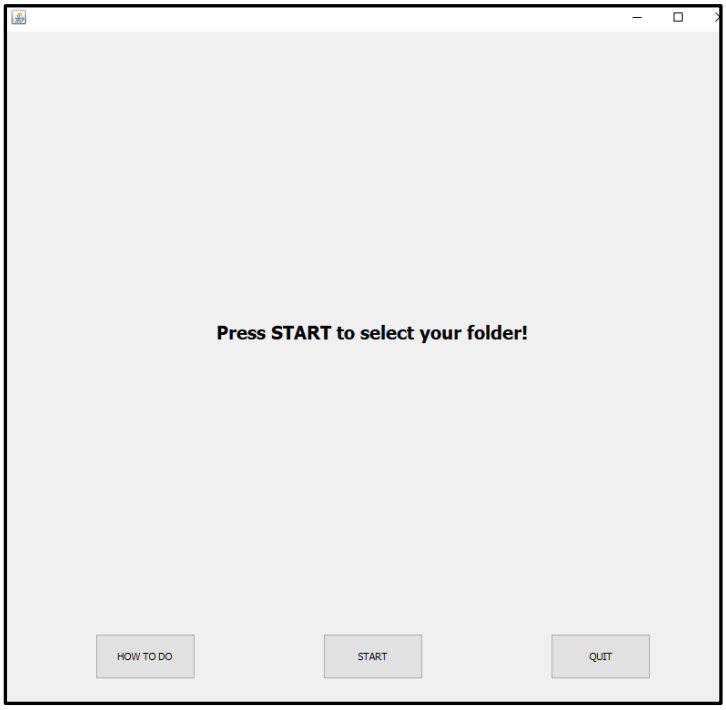

Abbildung 2: Hauptmenü

Im Hauptmenü kann man den **HOW TO DO** Button drücken um eine ausführliche Erklärung zu bekommen wie man alles vorbereitet und das Programm startet.

Der **START** Button macht das was er sagt, das Karteikartenspiel starten. Wenn man den **START** Button drückt wird man zum Auswahl-Fenster gebracht wo man den Ordner auswählen kann.

Mit **QUIT** kann man das Programm schliessen.

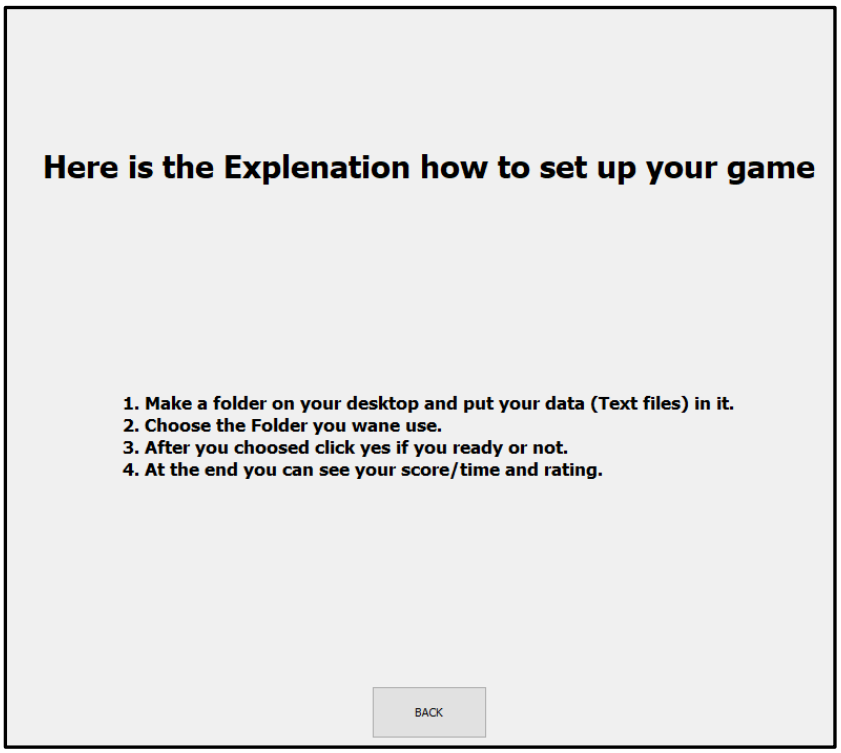

Abbildung 3: Erklärungs-Fenster

Im Erklärungs-Fenster wird kurz erklärt wie man die Ordner und Dateien vorbereiten muss und wie es funktioniert.

Mit **BACK** Button kommt man wieder zum Hauptmenü.

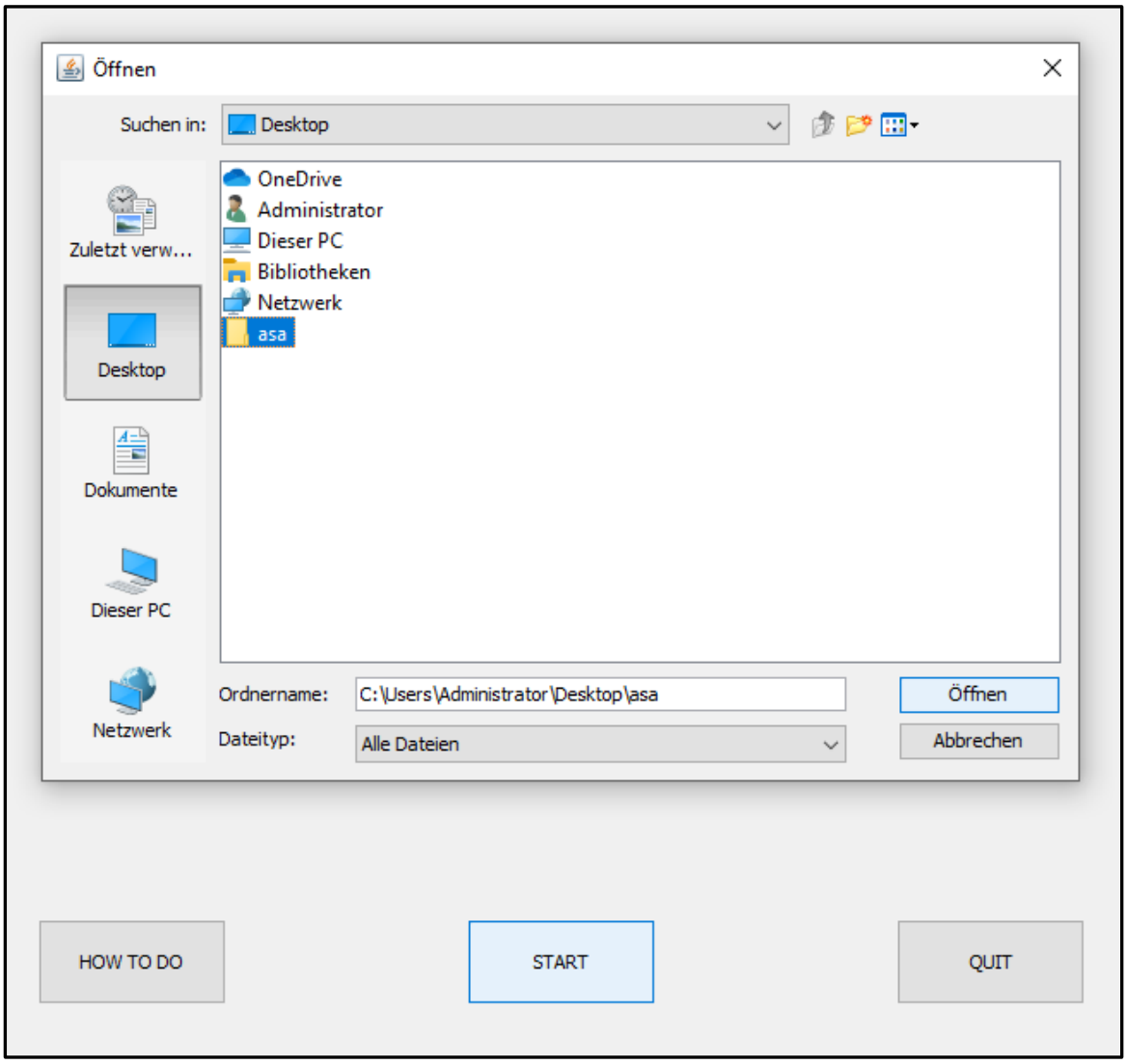

Abbildung 4: Ordner Auswahl

Nachdem man den **START** Button gedrückt hat muss man dann den richtigen Ordner auswählen .

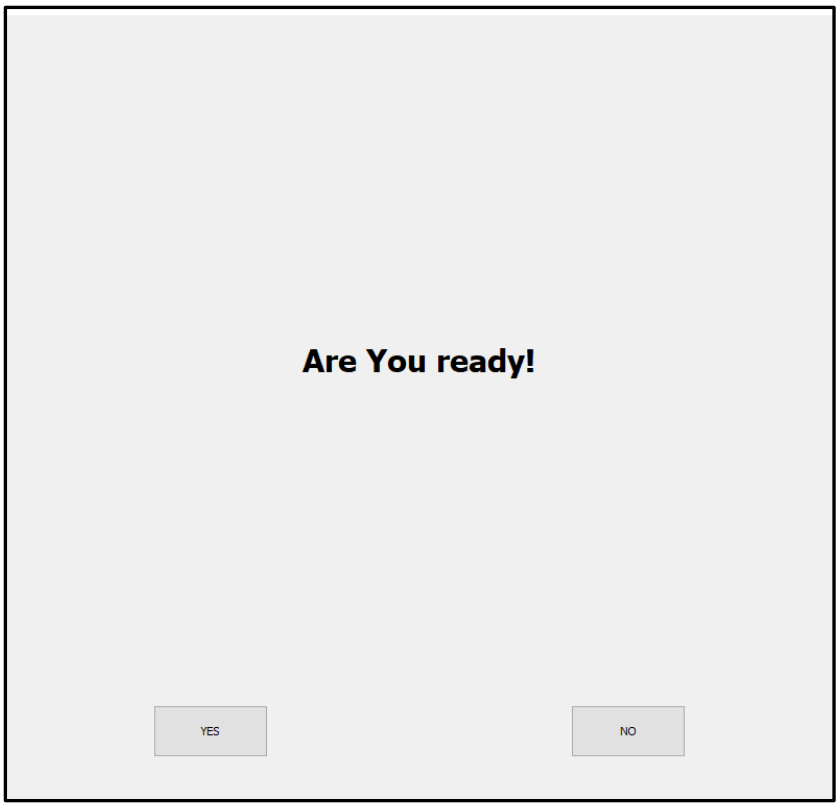

Abbildung 5: Ready Fenster

Im Ready Fenster frag es nochmals nach ob man parat ist mit dem Spiel anzufangen

YES Button startet es und der **NO** Button bringt dich wider zum Hauptmenü zurück.

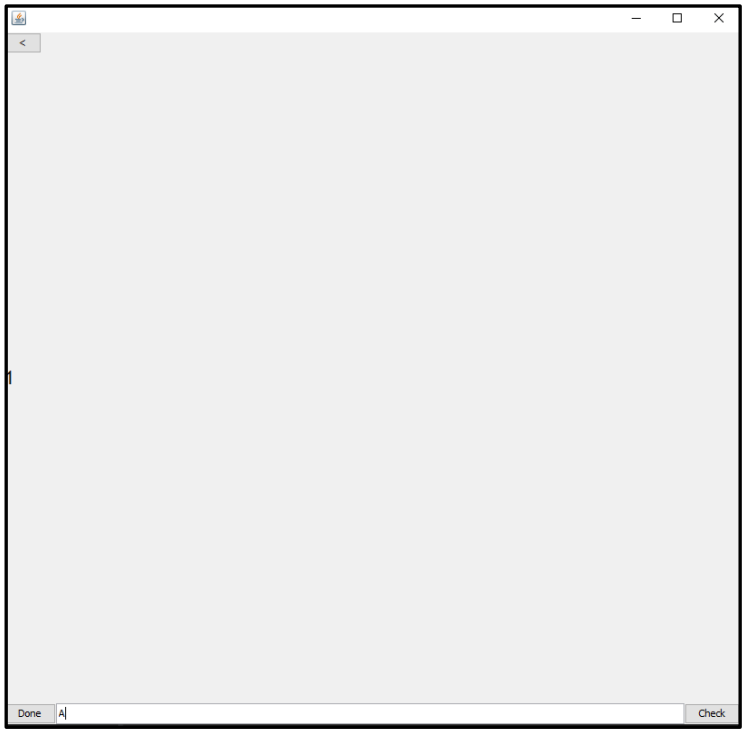

Abbildung 6: Frage Fenster

Hier werden die Text Datei angezeigt und unten kann man die Fragen beantworten.

Wenn irgendetwas nicht stimmen würde mit den Dateien kann man noch immer den **Done** Button drücken um zum Letzen Fenster zu kommen.

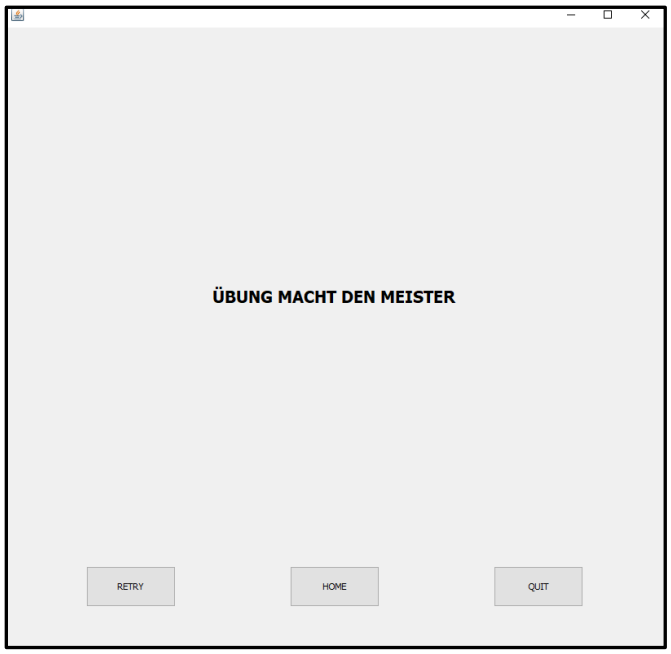

Abbildung 7:Letzter Fenster

Nachdem man fertig ist mit all den Dateienkommt man zum Letzten Fenster wo man:

- **RETRY** also wiederholen kann.
- **HOME** wieder zum Hauptmenü kommt.
- **QUIT** um das Spiel zu beenden.

Klassenstruktur

Die zu erstellten Klassen werden folgendermassen in drei verschiedene Package zu liegen kommen:

ch.csbe.guidemonstartion: Haupt-Package mit der Hauptklasse

ch.csbe.guidemonstartion.ui: Package mit allen UI-relevante Klassen sowie alle Kateikarten-Logik.

Umsetzung

Struktur

Karteikarten

Selbstverständlich! Das Konzept von digitalen Karteikarten kann in mehreren Schichten aufgebaut werden, die sich in Modell (Daten), Ansicht (Benutzeroberfläche) und Steuerung (Logik) unterteilen lassen. Dies ist ein klassisches Beispiel für das MVC-Modell (Model-View-Controller). Lassen Sie uns diese Struktur für Ihr Karteikarten-Projekt durchgehen:

1. Karteikarten-Modell (Model):

Das Karteikarten-Modell repräsentiert die Datenstruktur und die Logik hinter einer Karteikarte. Eine grundlegende Karteikarte hat:

Frage: Der Text oder Inhalt, den der Benutzer beantworten muss.

Antwort: Die korrekte Antwort auf die Frage.

ID oder Dateiname: Ein eindeutiger Identifikator für die Karteikarte, der oft aus dem Dateinamen abgeleitet wird.

In Ihrem speziellen Fall wird jede Karteikarte durch eine Textdatei repräsentiert. Der Inhalt der Datei ist die Frage und der Dateiname (ohne die .txt-Erweiterung) ist die Antwort.

2. Benutzeroberfläche (View):

Die Benutzeroberfläche ist das, was der Benutzer sieht und mit dem er interagiert. In Ihrem Projekt könnte die Benutzeroberfläche Folgendes umfassen:

Textfeld oder Bereich für die Frage: Zeigt den Inhalt der aktuellen Karteikarte (Textdatei) an.

Eingabefeld: Hier gibt der Benutzer seine Antwort ein.

Check-Button: Der Benutzer klickt darauf, um seine Antwort zu überprüfen.

Feedback-Bereich: Zeigt dem Benutzer, ob seine Antwort richtig oder falsch war.

Navigationsbuttons: Zum Durchblättern der Karteikarten oder zum Zurückkehren zum Hauptmenü.

3. Steuerung (Controller):

Dieser Teil des Systems steuert die Interaktionen zwischen dem Modell und der Ansicht. Es handelt sich um die Logik, die bestimmt, wie die Anwendung auf Benutzereingaben reagiert. In Ihrem Projekt könnte die Steuerung:

Das Laden der Karteikarten (Textdateien) aus einem ausgewählten Ordner steuern.

Die richtige Antwort aus dem Dateinamen extrahieren und sie mit der Benutzereingabe vergleichen.

Feedback geben, ob die Antwort des Benutzers richtig oder falsch war.

Die Anzeige der nächsten Frage steuern.

Zusammenfassung:

Modell: Repräsentiert die Datenstruktur der Karteikarten.

Ansicht: Was der Benutzer sieht und wie er mit der Anwendung interagiert.

Steuerung: Die Logik, die bestimmt, wie die Anwendung auf Benutzereingaben reagiert.

Testing

Testkonzept

Sobald alle Features der Applikation umgesetzt wurden, wird erstmals die Testphase eingeleitet. Nach jeder Änderung, die nach der Testphase ausgeführt wird, wird eine neue Testphase gestartet und das Testprotokoll wird erneut durchgearbeitet.$

Ein Reales einer Version, für die das Testprotokoll nicht durchgearbeitet wurde, ist nicht gestattet.

Die Testphase läuft in zwei schritten ab: Zunächst werden die automatischen Unit-Tests durchgeführt und anschliessend wird auf allen System der funktionale Test gemäss den unterstehenden Testfälle durchgeführt.

Testfälle

Dieser Abschnitt behandelt alle Testfälle, die im zwei Schritten der Testphase durchzuführen sind.

Als Vorbedingung für alle Testfälle gilt, dass das Programm gestartet wurde und das Hauptfenster angezeigt wird.

Tabelle 9: Korrektes Karteikarten Ordern Auswahl:

Testfall 1: Korrektes Karteikarten Ordner auswählen.
Ablauf 1. Knopf START drücken 2. Ordner auswählen mit Inhalt. 3. Auswahl bestätigen.

Tabelle 10: Korrektes Ergebnis

Testfall 1: Korrektes Karteikarten Ordner auswählen.	
Erwartetes Ergebnis • Das Programm erkennt das es ein Ordner ist und das Dateien enthält sind. • Das es den Ordner einlesen kann.	Tatsächliches Ergebnis • Wie erwartet erkennt es den Ordner und die darin enthaltenen Dateien.

Tabelle 11: Fehlerhafte Karteikarten Ordner Auswahl

Testfall 1: Fehlerhafte Karteikarten Ordner auswählen.
Ablauf 1. Knopf START drücken 2. Ordner auswählen ohne Inhalt. 3. Auswahl bestätigen.

Tabelle 12: Fehlerhaftes Ergebnis

Testfall 1: Fehlerhafte Karteikarten Ordner auswählen.	
Erwartetes Ergebnis • Das Programm sollt nicht starte und soll eine Fehlermeldung geben wo sagt das man ein Ordner auswählen soll der Inhalt hat.	Tatsächliches Ergebnis • Leider Startet das Programm auch ohne Inhalt.

Unit-Test

Für die Applikation wurden drei Unit-Test umgesetzt, die vor dem Durcharbeiten der Testfälle erfolgreich durchlaufen werden müssen. Andernfalls ist es sinnlos, die Testfälle durchzuführen, wenn diese Funktionalitäten gar nicht richtig funktionieren.

Test 1:

Ich habe versucht einen Unit Test zu machen der Schauen soll ob er alle Dateien gelesen hat und die richtige Anzahl an Dateien einliest.

Leider gab es eine Fehlermeldung die ich mir nicht erklären kann.

Referenzen

GPT-4: Unit Test, fileChoser, @Override für das beenden des Programmes mit dem esc button, loadNextMedia, setSelectedFolder.

Manuel Sollberger: Der Rest wo wir im Unterricht gelernt haben.

https://www.java-tutorial.org/swing.html den Rest wo ich mit dem Wissen vom Herr Sollberger erlernt habe.