Informatik Projekt – Planung

PDF-Datei in einem externen PDF-Reader anzeigen: mithilfe der Klasse Desktop

PDF-Datei in Java-Programm laden und anzeigen: externe Bibliothek einbinden, zum Beispiel ICEpdf

https://javabeginners.de/Ein-_und_Ausgabe/PDF_anzeigen.php

PDF-Datei in Java erstellen: mithilfe der externen Bibliothek iText

https://javabeginners.de/Ein-_und_Ausgabe/PDF_schreiben.php
https://kb.itextpdf.com/itext/extracting-objects-from-a-pdf

PDF-Datei in Java erstellen, Text auslesen und Dokument manipulieren: mithilfe der externen Bibliothek PDFBox von Apache

https://www.javaeinfacherklärt.de/?p=341
https://www-tutorialspoint-com.translate.goog/pdfbox/pdfbox_reading_text.htm?_x_tr_sl=en&_x_tr_tl=de&_x_tr_hl=de&_x_tr_pto=rq

auf PDF-Datei zugreifen: Ressourcen-Order in Projekt-Ordner anlegen, sodass auf Datei über einen
relativen Pfad zugegriffen werden kann

https://www.java-forum.org/thema/pdf-datei-in-java-einbinden.173987/

1.1) Vorlesungsfolien hochladen

Anforderungen
- PDF-Upload: Nutzer sollen PDFs in die Anwendung hochladen können

Technologien und Bibliotheken auswählen
- PDF-Verarbeitung: Apache PDFBox oder iText sind sehr beliebt für das Extrahieren von Text aus PDFs.

Mögliche Projektstruktur
- MainApp: Nutzeroberfläche (GUI)
  → start
- PDFProcessor: Text aus einer PDF-Datei extrahieren
  → extractText

4.1) Zusammenfassung der Vorlesungsfolien generieren

Anforderungen
- PDF-Verarbeitung: Extrahieren von Text aus PDF

• Zusammenfassung durch KI: Die Inhalte der Vorlesung sollen analysiert und zusammengefasst werden
• Nutzeroberfläche: CLI-Tool (Command-Line Interface) oder GUI

Technologien und Bibliotheken auswählen
• Zusammenfassung durch KI-Integration: REST-API verwenden, um eine KI für die Textzusammenfassung zu nutzen, z. B. Llama
• JSON-Verarbeitung: Für API-Kommunikation (falls benötigt) kann Gson oder Jackson verwendet werden.
• GUI: z.B. mit JavaFX

Mögliche Projektstruktur
• Summarizer: Zusammenfassung mithilfe von KI generieren
→ summarize
• Summary: Zusammenfassung bestehend aus Text

5.1) Anki Cards erstellen
Anforderungen
• PDF-Verarbeitung: Extrahieren von Text aus PDF
• Fragen und Antworten erstellen durch KI: Die Inhalte der Zusammenfassung sollen analysiert und in Fragen und Antworten zusammengefasst werden

Technologien und Bibliotheken auswählen
• Fragen und Antworten erstellen durch KI-Integration: REST-API verwenden, um eine KI für die Textzusammenfassung zu nutzen, z. B. Llama
• JSON-Verarbeitung: Für API-Kommunikation (falls benötigt) kann Gson oder Jackson verwendet werden.
• GUI: z.B. mit JavaFX

Mögliche Projektstruktur
• FlashcardGenerator: generiert Karteikarten mithilfe von KI
→ generateFlashcards
• FlashcardProcessor: verarbeitet KI-Antwort, z.B. im JSON Format
→ parseFlashcards
• Flashcard: Karteikarte bestehend aus Frage und Antwort
→ getQuestion
→ setQuestion
→ getAnswer
→ setAnswer

5.2) Anki Cards lernen
Anforderungen
• Anzeigebereich: Zeigt die Frage (und später die Antwort) an
• "Frage auflösen"-Button: Blendet die Antwort ein
• "Weiter"-Button: Wechselt zur nächsten Karteikarte
• Bewertungssystem: Nutzer kann angeben, ob die Frage richtig beantwortet wurde (ja, nein)
• Fortschrittsanzeige: Zeigt den Fortschritt der Lernsession an (z. B. "3/10 Fragen beantwortet").

Technologien und Bibliotheken auswählen

Mögliche Projektstruktur
- FlashcardSetLoader: Benutzer wählt ein Karteikartenset zum Lernen aus
  → loadFlashcardsFromFile
- FlashcardLearningSession: Steuerung der Lernsession, Verwaltung des Kartensets und Aktualisierung der GUI
  → startFlashcardLearningSession
  → showFlashcard
  → showNextFlashcard

x) Speicherung lokal als PDF

Anforderungen
- Speicherung: Sowohl die Originaldateien als auch die generierten Zusammenfassungen sollen lokal gespeichert werden.

Technologien und Bibliotheken auswählen
- Dateisystem-Speicherung: mit Java-IO-Bibliothek

Mögliche Projektstruktur
- SummarySaver: Zusammenfassung in eine neue PDF-Datei speichern
  → saveSummary
- FlashcardSaver: schreibt die generierten Fragen und Antworten in eine PDF-Datei
  → saveFlashcards
- FileManager: Dateien lokal in einem festgelegten Verzeichnis speichern
  → getFile

Bibliothek PDFBox von Apache

Funktionen von PDFBox:

PDF-Datei in Java erstellen, Text extrahieren und Dokument manipulieren

Im Folgenden sind die vier Hauptkomponenten von PDFBox aufgeführt:

PDFBox – Dies ist der Hauptteil der PDFBox. Er enthält die Klassen und Schnittstellen zur Inhaltsextraktion und -bearbeitung.

FontBox – Diese enthält die mit der Schriftart verbundenen Klassen und Schnittstellen. Mit diesen Klassen können wir die Schriftart des Textes im PDF-Dokument ändern.

XmpBox – Diese enthält die Klassen und Schnittstellen, die XMP-Metadaten verarbeiten.

Preflight – Diese Komponente wird verwendet, um die PDF-Dateien anhand des PDF/A-1b-Standards zu überprüfen.

Tutorials:

https://www.javaeinfacherklärt.de/?p=341

PDFBox in Projekt integrieren:

Maven Projekt auf IntelliJ erstellen
dependencies von folgender Website kopieren:
https://www-tutorialspoint-com.translate.goog/pdfbox/pdfbox_environment.htm?_x_tr_sl=en&_x_tr_tl=de&_x_tr_hl=de&_x_tr_pto=rq
dependencies in pom.xml Datei einfügen
pom.xml Datei neu laden (lädt die dependencies herunter)

Öffnen eines vorhandenen PDFs:

https://www-tutorialspoint-com.translate.goog/pdfbox/pdfbox_loading_a_document.htm?_x_tr_sl=en&_x_tr_tl=de&_x_tr_hl=de&_x_tr_pto=rq

Extrahieren von Text aus PDF zu String:

https://www-tutorialspoint-com.translate.goog/pdfbox/pdfbox_reading_text.htm?_x_tr_sl=en&_x_tr_tl=de&_x_tr_hl=de&_x_tr_pto=rq

Pfad zu PDF finden:

PDF öffnen, oben den Pfad kopieren
file:/// und %20 für Leerzeichen herauslöschen

Dateien speichern:

je nach Dateiart verschiedene Möglichkeiten:

1. AppData:

Sinnvoll, wenn die Daten nicht direkt vom Benutzer bearbeitet werden sollen (z. B. interne Anwendungsdaten, Logs oder Cache).
Ideal für technische Daten oder Konfigurationsdateien, weniger für generierte Inhalte wie PDFs.

2. user.home:

Sinnvoll, wenn die Dateien dem Benutzer leicht zugänglich sein sollen und die Anwendung plattformübergreifend ist.
Empfohlen für deine Lernanwendung, da die Benutzer ihre generierten PDFs und Zusammenfassungen leicht wiederfinden können.

3. Documents:

Sinnvoll, wenn es sich um Endbenutzerdaten handelt, die regelmäßig überprüft oder verwendet werden sollen, wie in deinem Fall die Zusammenfassungen und Karteikarten.

4. Benutzerdefinierte Speicherorte:

Optional sinnvoll, wenn die Benutzer erfahren genug sind, ihre Speicherorte selbst zu wählen.

1. AppData und Äquivalente Speicherorte der anderen Betriebssysteme:

Windows:
Roaming-Verzeichnis (C:\Users\<Benutzername>\AppData\Roaming

Appdata = versteckter Ordner, in dem Programme Dateien ablegen können
erreichbar z.B. über...
… Ausführen-Dialog:
- 'Windows-Taste' + 'r'; Suchbegriff "%appdata%' eingeben
- führt direkt in Roaming-Unterordner des AppData-Verzeichnisses

… Datei-Explorer
- Datei-Explorer öffnen
- in Adressleiste des Datei-Explorers %appdata% eingeben und 'Enter' drücken
- führt direkt in Roaming-Unterordner des AppData-Verzeichnisses

… Sichtbarkeit von versteckten Ordnern aktivieren
- Datei-Explorer öffnen
- Menü „Ansicht“, „Optionen“ oder „Ansichtseinstellungen ändern“
- „Ordneroptionen“ → Tab „Ansicht“ → erweiterten Einstellungen → „Versteckte Dateien, Ordner und Laufwerke anzeigen“ → „Übernehmen“ und „OK“
- Gehe zu Benutzerordner, der in der Regel den Benutzernamen trägt, und AppData-Ordner sollte nun sichtbar sein

… Direkte Pfadangabe
- Pfad zum AppData-Ordner in Adressleiste des Datei-Explorers eingeben
- Pfad sieht i.d.R. so aus: „C:\Users\[IhrBenutzername]\AppData“ bzw. „C:\Benutzer[IhrBenutzername]AppData„.

MacOS:
Verzeichnis ~/Library/Application Support wird als Standard für Anwendungsdaten verwendet

Linux/Unix:
Verzeichnis ~/.config wird häufig als Speicherort für Anwendungsdaten verwendet.

Links:

appdata Ordner Windows:
https://www.windows-faq.de/2024/01/19/appdata-ordner-aufgabe-und-funktion/

„AppData„-Ordner besteht i.d.R. aus den Unterordnern „Local„, „LocalLow“ und „Roaming„.

1. Appdata/Roaming
- Ordner enthält Daten, die über ein Netzwerk mit anderen Computern geteilt werden können, sofern Sie sich in einer Umgebung mit einem Server befinden
- die in diesem Ordner gespeicherten Daten sind benutzerspezifisch, aber nicht gerätespezifisch

• das bedeutet, dass Einstellungen und Dateien, die hier gespeichert werden, beim Einloggen auf einem anderen Computer im selben Netzwerk verfügbar sind.
Typische Beispiele sind:
Browser-Lesezeichen
E-Mail-Einstellungen
Dokumentenvorlagen

2. Appdata/Local
• Ordner speichert Informationen, die ausschließlich auf Ihrem lokalen Computer bleiben
• diese Daten sind benutzer- und gerätespezifisch
• Anwendungen nutzen diesen Ordner, um Daten zu speichern, die zu groß sind oder aus Sicherheitsgründen nicht über ein Netzwerk geteilt werden sollten.
Typische Beispiele sind:
Zwischengespeicherte Dateien
Lokale Anwendungsdaten
Große Datensätze

3. Appdata/LocalLow

• Ordner ist eine Variante des „Local„-Ordners, aber mit niedrigeren Sicherheitsberechtigungen
• wird hauptsächlich von Anwendungen mit geringerem Sicherheitszugriff, wie Internet-Browsern im geschützten Modus, verwendet
• die Daten sind weniger privilegiert und dienen hauptsächlich dazu, die Sicherheit zu erhöhen, indem sie Anwendungen mit geringerer Berechtigung zugewiesen werden.
Typische Beispiele sind:
Daten aus dem geschützten Modus des Browsers
Add-Ons oder Plug-Ins mit niedrigeren Berechtigungen

Verzeichnisse durch Java Programm anlegen
https://openbook.rheinwerk-verlag.de/java8/06_003.html#u6.3

Speicherorte für Dateien je nach Betriebssystem für Java Applikation:
https://backsite.wordpress.com/2012/07/13/wo-kann-eine-java-applikation-die-benutzer-spezifischen-daten-speichern/

Bei Windows werden Einstellungen meist in APPDATA/Progname gespeichert. Bei Unix-Systemen werden die Daten im /home-Ordner mit einem vorangestellten „.“ unsichtbar hinterlegt. Zum Beispiel so: /home/.Progname
Dieser einfache Code gibt den Pfad zum Einstellungs-Ordner:

```
// Für Windows

String settingsPath = System.getenv("APPDATA");

// Für Unix

if ( settingsPath == null ) {

   settingsPath = System.getProperty("user.home");
```

```
   settingsPath += ".";

}

settingsPath += "Progname";
```

jar in Projekt integrieren:

pdfbox jar googlen
maven.org öffnen
Files View all auswählen
javadoc, sources und jar downloaden
in IntelliJ auf Striche oben gehen
Projekt Structure öffnen
Libraries
auf + drücken
in downloads alle drei jars auswählen
sichern

Adobe PDF-Bibliothek

Diese Bibliothek bietet API in Sprachen wie C++, .NET und Java und damit können wir Text aus PDF-Dokumenten bearbeiten, anzeigen, drucken und extrahieren.