Christian Bunse & Tilo Zülske
Büro: Haus 4/306c
Tel.: 03831-45-7306

HOST Hochschule Stralsund

Aufgaben zur Lehrveranstaltung
Laborpraktikum Software

Änderungshistorie

Änderungs-datum	Änderung	Kommentar
	Initiale Version	

Die Vor-Aufgaben übernehmen die Rolle der Präsenzaufgaben. Wie bisher ist ihre Bearbeitung Voraussetzung für eine Benotung der Hausaufgaben. Sie müssen die Vor-Aufgaben jedoch nicht unbedingt im Labor bearbeiten und auch nicht persönlich vorab präsentieren. Bitte geben Sie auf jeden Fall die Vor-Aufgaben zusammen mit den Hausaufgaben dieses Blattes E ab.

Die Hausaufgaben dieses Aufgabenblattes sind **bis zum 31.12.2025 (23:50) abzugeben**. Verspätete Abgaben werden **nicht** berücksichtigt. **Die darauffolgende Abnahme der Hausaufgaben ist Pflicht (für alle Studierenden des Kurses) und erfolgt in den letzten beiden Vorlesungswochen im Januar 2025 (zu den regulären Laborzeiten)**.

Um Wartezeiten bei der Abnahme zu vermeiden, **müssen** Sie sich für einen Termin anmelden. Hierzu nutzen Sie das entsprechende Werkzeug auf der Moodle-Seite des Kurses.

Es wird keine weitere Prüfung im Fach LP Software geben. Die Noten werden umgehend in der ersten Februar-Hälfte vergeben und dann in das Online-Notensystem eingeben.

Aufgabenblatt #5

Voraufgaben

1) Eventbearbeitung (Event)

Ergänzen Sie das folgende Event-Beispielprogramm.

```
package application;

import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;


public class Main extends Application {
	@Override
	public void start(Stage primaryStage) {
		try {
			GridPane root = new GridPane();
			Scene scene = new Scene(root,400,400);

			TextField textField1 = new TextField();
			textField1.setPromptText("Schreibe hier");

			TextField textField2 = new TextField();
			textField2.setPromptText("Schreibe hier");


			final EventHandler<KeyEvent> keyEventHandler =
			  new EventHandler<KeyEvent>() {
			      public void handle(final KeyEvent keyEvent) {
			              //Hier fehlt Ihr Code
			              keyEvent.consume();
			      }

			};

			textField1.setOnKeyPressed(keyEventHandler);
			textField2.setOnKeyPressed(keyEventHandler);

			root.setConstraints(textField1, 0,0);
			root.setConstraints(textField2, 0,1);

		    root.getChildren().addAll(textField1,textField2);
			primaryStage.setScene(scene);
			primaryStage.show();
		} catch(Exception e) {
			e.printStackTrace();
		}
	}

	public static void main(String[] args) {
		launch(args);
	}
}
```

Das Programm soll die Tasten F1, F2 und F3 abfangen und eine entsprechende Meldung ausgeben. Nach Erkennen wird dazu ein entsprechender Text in das `textField2` geschrieben. Alle anderen Tasten werden ignoriert.

2) Tutorial Graphische Benutzeroberflächen

Bearbeiten Sie das JavaFX-Tutorial auf der ILIAS-Kursseite (Ordner „Literatur und Software“). Sie werden in diesem Tutorial eine kleine Anwendung bauen und sich mit der GUI-Entwicklung unter JavaFX vertraut machen. Wir verwenden für die Erstellung der grafischen Benutzerschnittstelle (GUI) den frei verfügbaren SceneBuilder. Zum Start des Tutorials werden Sie dieses Werkzeug installieren und Ihre IDE-Installation geeignet konfigurieren.

3) Meine erste GUI (Graphische Benutzeroberflächen, Listen Ansicht)

Erstellen Sie die folgende Anwendung:

- Textfeld zur Eingabe eines Begriffes,
- Darunter eine Liste, in der die eingegebenen Begriffe alle aufgeführt sind,
- Button „Hinzufügen“ zum Hinzufügen des Begriffs zur Liste,
- Button „Reset“ zum Löschen aller Elemente der Liste

Das kann z.B. so aussehen:

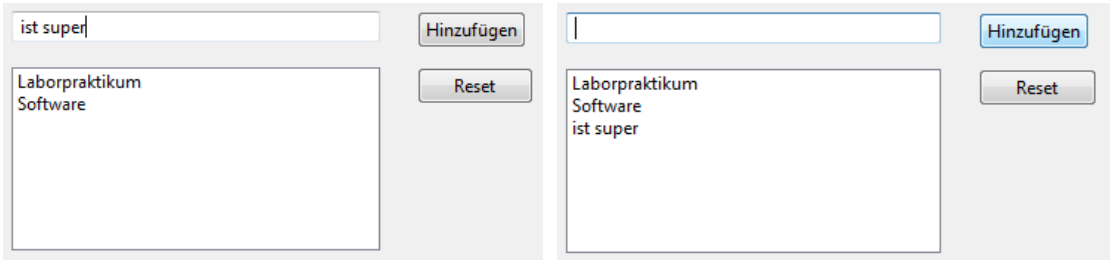

Abbildung 1: Musterlayout

Tipp: Nutzen Sie FXML (fxid, @FXML, …) und Ihre Controller Klasse zur Definition von Eventhandlern.

Hausaufgaben

Alle folgenden Teilaufgaben können Sie in Gestalt eines Programms zusammen abgeben.

4) Webbrowser und KeyPressed-Event

[2 Punkte]

Erstellen Sie eine JavaFX-Anwendung mit graphischer Benutzeroberfläche. Diese Anwendung soll ein Eingabefeld für einen Titel enthalten und einen Button „Suchen“. Sobald der Button Suchen oder die Taste ENTER gedrückt wird, erscheint im unteren Bereich der Anwendung die Übersicht zu einem Buch, so wie sie in Wikibooks enthalten ist. Hierzu verwenden Sie das JavaFX-Dialogelement „WebView“.

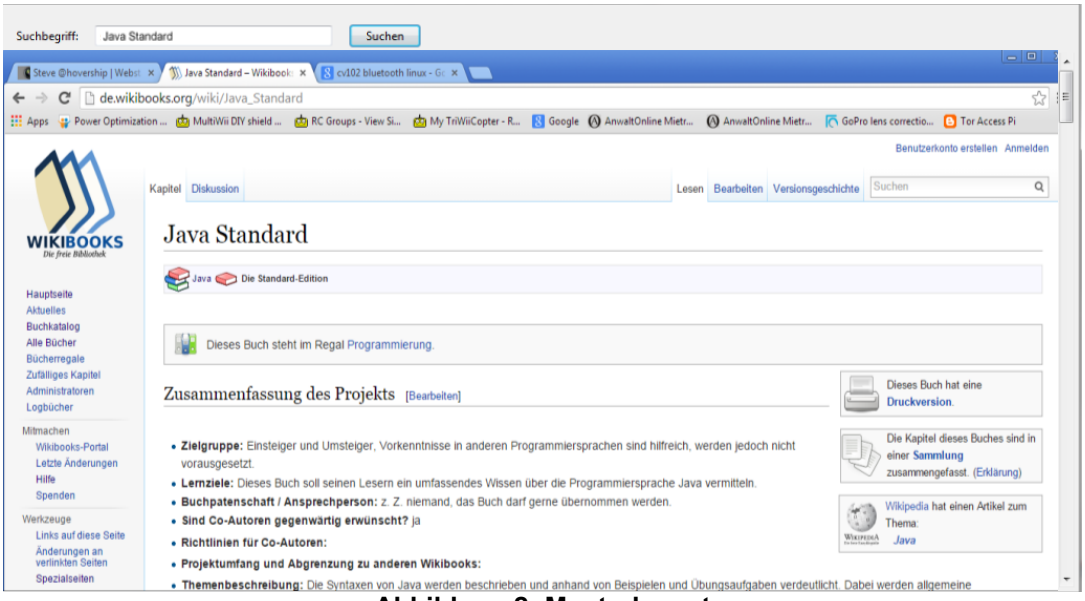

Abbildung 2: Musterlayout

Lesen Sie sich das Beispiel in der Online-Hilfe bzw. https://openjfx.io/javadoc/21/javafx.web/javafx/scene/web/WebView.html zum WebView durch und gehen Sie entsprechend vor. Die URL für Wikbooks ist `http://de.wikibooks.org/wiki/<Titel>`. Sie können initial das Browser-Dialogelement durch folgenden Aufruf in der `initialize()` Methode Ihrer FXML Datei ansprechen.

```
WebEngine engine = webView.getEngine();
engine.load("http://de.wikibooks.org");
```

Ihr Programm soll auf einen Mausklick und auf die ENTER-Taste (insb. dann, wenn sie im Eingabefeld gedrückt wird) reagieren. Sie müssen also zwei Events verarbeiten (TextField und Button). Beide EventHandler müssen dann auf gleiche Weise das Laden und Anzeigen der entsprechenden Wikibooks-Seite veranlassen.

Ihr Programm soll folgende Klassen enthalten:

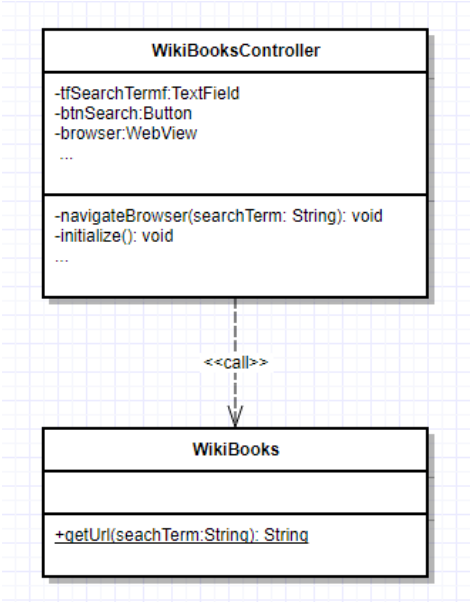

Abbildung 3

Die Klasse `WikiBooks` kapselt das Wissen über das URL-Format für Wikibooks-Titel. Wir werden im Verlaufe der folgenden Aufgaben noch weitere Methoden in dieser Klasse implementieren. Die unterstrichene Methode bedeutet, dass es sich um eine statische Methode handelt.

Das Klassendiagramm soll Ihnen Anregungen geben, wie Sie Ihr Programm vernünftig in einzelne Klassen und Methoden strukturieren können. Wenn Sie in Ihrem Programm eine andere Struktur verwenden, muss das kein Fehler sein. Verstehen Sie die hier abgebildeten Methoden bitte nur als Vorschlag. Allerdings: Sie dürfen nicht alles, was Ihr Programm leistet, in ein oder zwei „Riesen“-Methoden schreiben. Das wäre schlecht wartbar. Versuchen Sie so weit wie möglich doppelten Programmcode zu vermeiden.

Die Methode `navigateBrowser` ist ein Beispiel für eine Methode, die geschrieben wurde, um doppelten Programmcode zu vermeiden. Sie enthält in einer Musterlösung zwar nur zwei Zeilen. Dennoch macht es Sinn, diese zwei Zeilen in eine Extra- Methode zu schreiben, statt sie mehrfach (in EventHandlern) zu implementieren.

Tipp: Bitte nutzen Sie das Java FX Pendant zu der von Ihnen genutzten Java Version.

5) Information zur letzten Bearbeitung integrieren (Dynamisches Label, MessageBox)

[3 Punkte]

Teilaufgabe 1
Erweitern Sie die bereits erstellte Anwendung um die Information, wann und von wem das angezeigte Buch zuletzt bearbeitet wurde und in welchem Regal es zu finden ist. Nutzen Sie dazu Ihren Programmcode des Aufgabenblattes #4. Sollten Sie die Aufgaben von Blatt #4

nicht vollständig bearbeitet haben, besorgen Sie sich bitte eine Lösung bei Ihren Kommiliton(inn)en (**Referenzieren des/der ursprünglichen Autor*in nicht vergessen!!!)**.

Die Informationen zur letzten Bearbeitung, Urheber und Regal sollen ungefähr zeitgleich mit dem Inhalt des WebView aktualisiert werden und unterhalb des Artikels in Form von Labeln angezeigt werden (s. Abbildung 4). Die Kapitel werden nicht angezeigt. Optional können sie in einem weiteren Label die Anzahl der Kapitel anzeigen.

Tipp: Sollten Sie Netzwerkprobleme haben, empfiehlt es sich die Timeout-Property der `URLConnection`, welche Sie für den Zugriff auf die Wikibooks-Seite verwenden, auf einen anderen als den Standardwert zu setzen (zumindest während Sie das Programm entwickeln). Der Standardwert ist 0 *(„A timeout of zero is interpreted as an infinite timeout.“* [https://docs.oracle.com/en/java/javase/21/docs//api/java.base/java/net/URLConnection.html]).

Tipp: Der Response-Stream sollte von Ihrem Programm jedes Mal geschlossen werden, bevor Sie eine neue Anfrage erzeugen. Wenn Sie dies vergessen, könnte sich das darin bemerkbar machen, dass Ihr Programm nach ein paar eingegebenen Begriffen konsequent den Zugriff auf Wikibooks verwehrt.

Teilaufgabe 2
Fangen Sie Fehler beim Zugriff auf Wikibooks ab und melden eventuelle Fehler dem Benutzer in einer `Dialogbox (Warning, Error oder Exception Dialog)`.

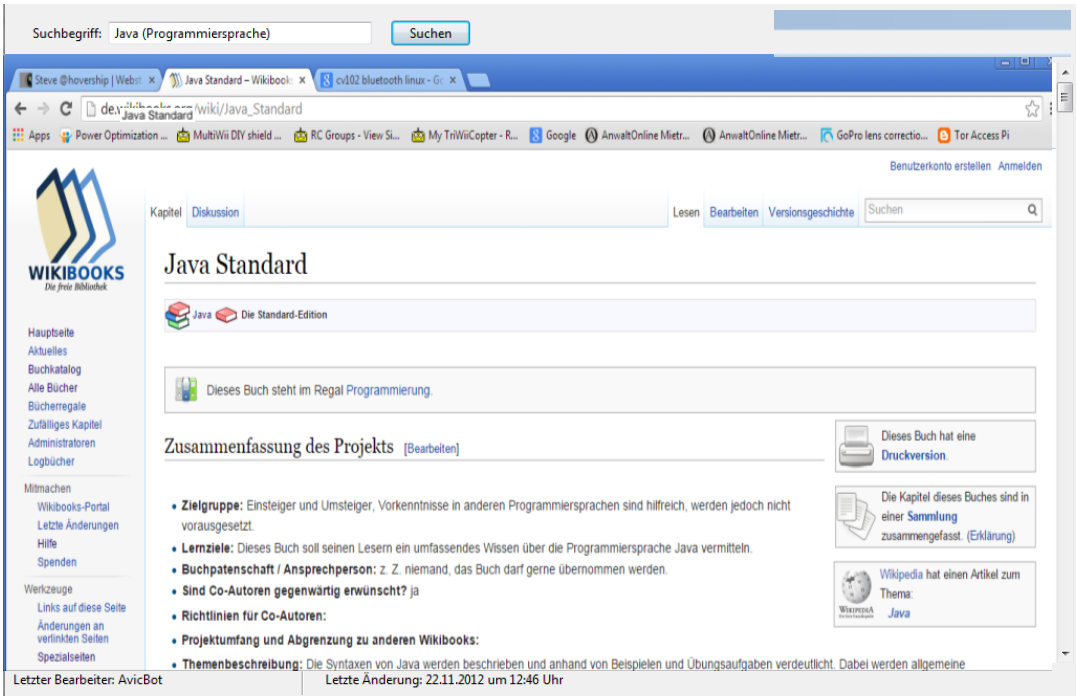

Abbildung 4: Musterlayout

Teilaufgabe 3
Bringen Sie alle Wikibooks-spezifische Funktionalität in der Klasse Wikibooks unter. Methoden zum Extrahieren der Meta-Informationen (letzter Bearbeiter, letzte Änderung, Regal) gehören nicht in die Klasse, welche Sie mit dem SceneBuilder erzeugt haben. Methoden, welche die extrahierten Daten des Artikels in die GUI übertragen, wären hier hingegen gut aufgehoben.

Teilaufgabe 4
Fügen sie die Button „Hinzufügen“, „Sortieren“, „Löschen“, „Speichern“, „Laden“, „Import“ und „Export“ zu Ihrer Applikation hinzu.

- Bei Wahl von *Hinzufügen* soll, so vorhanden, das angezeigte Wikibooks Element zu Ihrem Zettelkasten hinzugefügt werden. Hierzu müssen Sie evtl. weitere Informationen aus der XML-Repräsentation entnehmen.
- Der Button „*Sortieren*“ soll den Zettelkasten sortieren. Dabei soll die Sortierung („A-Z“ und „Z-A“) alternieren.
- Der Button „Delete“ soll alle Elemente mit dem Titel des angezeigten Wikibook-Elements aus der internen Repräsentation entfernen.
- Der Button „Speichern“ soll den vorhandenen Zettelkasten serialisieren XOR in eine Datenbank schreiben XOR als XML Datei ablegen. Der Button „Laden“ soll einen gespeicherten Zettelkasten mittels des zuvor gewählten Verfahrens laden.
- Ein weiterer Button soll den Import/Export von Daten in Pseudo-Bibtex Format (vgl. Aufgaben B.10 und C.6) aus/in einer Datei erlauben.
  Sollten Sie Aufgabe B.10 und/oder C.6 (Bonusaufgabe) nicht bearbeitet haben, können Sie den Code von Kommilitonen nutzen (Zitieren nicht vergessen) oder im Fall von C.6 eine *UnsupportedOperationException* nutzen und entsprechend visualisieren.

Teilaufgabe 5
Erzeugen Sie eine neue Exception-Klasse „`MyWebException`“, welche immer dann ausgelöst wird, wenn das Laden der Metadaten eines Artikels von der Export-Seite fehlschlägt.

Wenn Sie noch weitere Klassen und (insbesondere private) Methoden benötigen, um doppelt geschriebenen Code zu vermeiden, dann ist das völlig in Ordnung und vermutlich sogar eine gute Idee.

6) Synonyme und Zettelkasten (ListView, AlertDialog, Einsatz eines Web-Service)

[3 Punkte]

Hinweis: Sollten Sie keinen Zugriff auf den REST Service der Uni-Leipzig bekommen bearbeiten Sie alternativ Aufgabenteil B. Bitte beachten Sie, dass nur einer der Aufgabenteile (A XOR B) bearbeitet werden muß. Nutzen Sie dazu das Bsp.-Programm um den Corpora-Service aus Leipzig zu prüfen.

Aufgabenteil A:

Erweitern Sie die Anwendung um Dialog-Elemente, in denen die Liste aller Synonyme des gesuchten Begriffes und eine Liste aller im Zettelkasten enthaltenen Medien mit passendem Titel angezeigt werden. Diese Listen sollen ungefähr zeitgleich zum unteren Webbrowser-Element aktualisiert werden.

Vorbereitend müssen wir Code schreiben, der die Synonyme zu einem bestimmten Begriff ermittelt. Wir nutzen (wie bereits auf dem Aufgabenblatt D erwähnt) dazu den Wortschatz-Service der Universität Leipzig in Form eines REST-Webservice (vgl. http://api.corpora.uni-leipzig.de/ws/swagger-ui.html). Für die Bearbeitung dieser Aufgabe sollten Sie den Dienst „Similarity Service“ (vgl. http://api.corpora.uni-leipzig.de/ws/swagger-ui.html#!/similarity-service/getCoocSimilarWordsUsingGET) verwenden. Dazu benötigen Sie:

1. Installation einer Bibliothek die JSON-Daten verarbeiten kann. Hier bietet sich z.B. die JSON-Simple Variante an. Ein entsprechendes JAR-File finden Sie im Verzeichnis „Literatur und Software“ im ILIAS-Verzeichnis.

   Das Einbinden von Bibliotheken geschieht in IntelliJ oder Eclipse in den Eigenschaften des Projektes.
   a. [IntelliJ] Rechtsklick auf das Projekt und „Open Module Settings“ wählen. Alternativ können Sie auch „F4“ nutzen; Im sich öffnenden Fenster auf „Libraries“ klicken und Jar (mittles „+“) hinzufügen.
2. Die Dokumentation der Parameter des Dienstes findet sich unter der o.g. URL.

3. Das folgende Programm demonstriert den allgemeinen Zugriff auf den Dienst:

```
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.util.Scanner;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

public class Syn2 {

    public static void main(String[] args) throws ParseException, IOException {
        String BasisUrl = "http://api.corpora.uni-leipzig.de/ws/similarity/";
        String Corpus = "deu_news_2012_1M";
        String Anfragetyp = "/coocsim/";
        String Suchbegriff = "sehen";
        String Parameter = "?minSim=0.1&limit=50";
        URL myURL;

        myURL = new URL(BasisUrl + Corpus + Anfragetyp + Suchbegriff + Parameter);

        JSONParser jsonParser = new JSONParser();
        String jsonResponse = streamToString(myURL.openStream());
        // den gelieferten String in ein Array parsen
        JSONArray jsonResponseArr = (JSONArray) jsonParser.parse(jsonResponse);
        // das erste Element in diesem Array
        JSONObject firstEntry = (JSONObject) jsonResponseArr.get(0);
        // aus dem Element den Container für das Synonym beschaffen
        JSONObject wordContainer = (JSONObject) firstEntry.get("word");
        // aus dem Container das Synonym beschaffen
        String synonym = (String) wordContainer.get("word");
        // ausgeben
        System.out.println(synonym);
        System.out.println(System.lineSeparator());

        // alle abgefragten Synonyme
        for (Object el : jsonResponseArr) {
            wordContainer = (JSONObject) ((JSONObject) el).get("word");
            synonym = (String) wordContainer.get("word");
            System.out.println(synonym);
        }
    }

    /**
     * Reads InputStream's contents into String
     *
     * @param is - the InputStream
```

```
         * @return String representing is' contents
         * @throws IOException
         */
        public static String streamToString(InputStream is) throws IOException {
            String result = "";
            // other options: https://stackoverflow.com/questions/309424/read-convert-
an-inputstream-to-a-string
            try (Scanner s = new Scanner(is)) {
                s.useDelimiter("\\A");
                result = s.hasNext() ? s.next() : "";
                is.close();
            }
            return result;
        }

    }
```

Ergänzen Sie in Ihrer GUI zwei `ListViews`. Diese sollen eine alphabetisch sortierte Liste der Synonyme und eine Liste der Titel des Zettelkastens (in Ihrer aktuellen Reihenfolge) enthalten.

Ermöglichen Sie weiterhin, dass der Benutzer durch Auswahl eines Synonyms der Liste eine erneute Suche anstoßen kann. Realisieren Sie dies durch einen weiteren Button „Suche Synonym“ neben der Liste der Synonyme. Der derart ausgewählte und gesuchte Begriff soll durch Ihr Programm automatisch in das Begriffs-Eingabefeld kopiert und eine neue Suche angestoßen werden. Achten Sie darauf, dass Ihre `ListView` nur die Einfach-Auswahl (`SINGLE`) zulässt.

Ihr Programm soll weiterhin auf einen Doppelklick auf den `ListView` genauso reagieren, als hätte der Benutzer den Button „Suche Synonym“ betätigt.

Gewünschtes Layout der Anwendung:

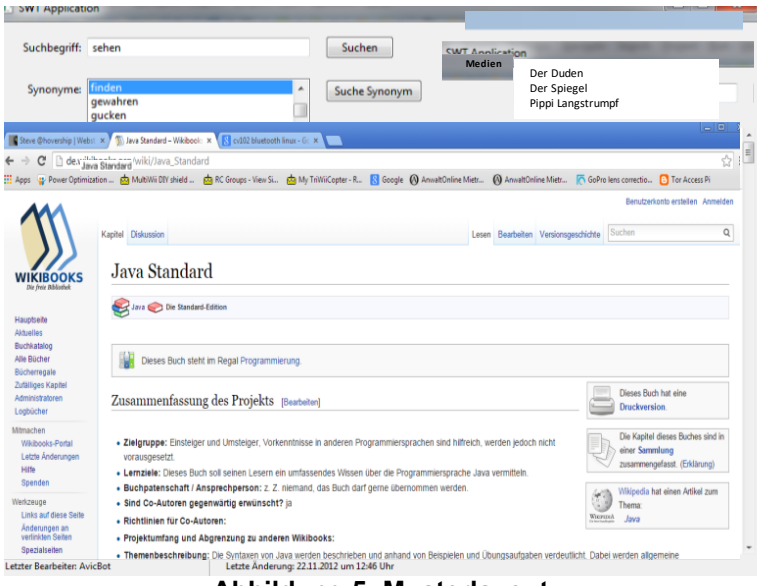

Abbildung 5: Musterlayout

Nach Auswahl des Synonyms „Spiegel“ und Betätigung von „Suche Synonym“ soll der Bildschirm etwa so aussehen:

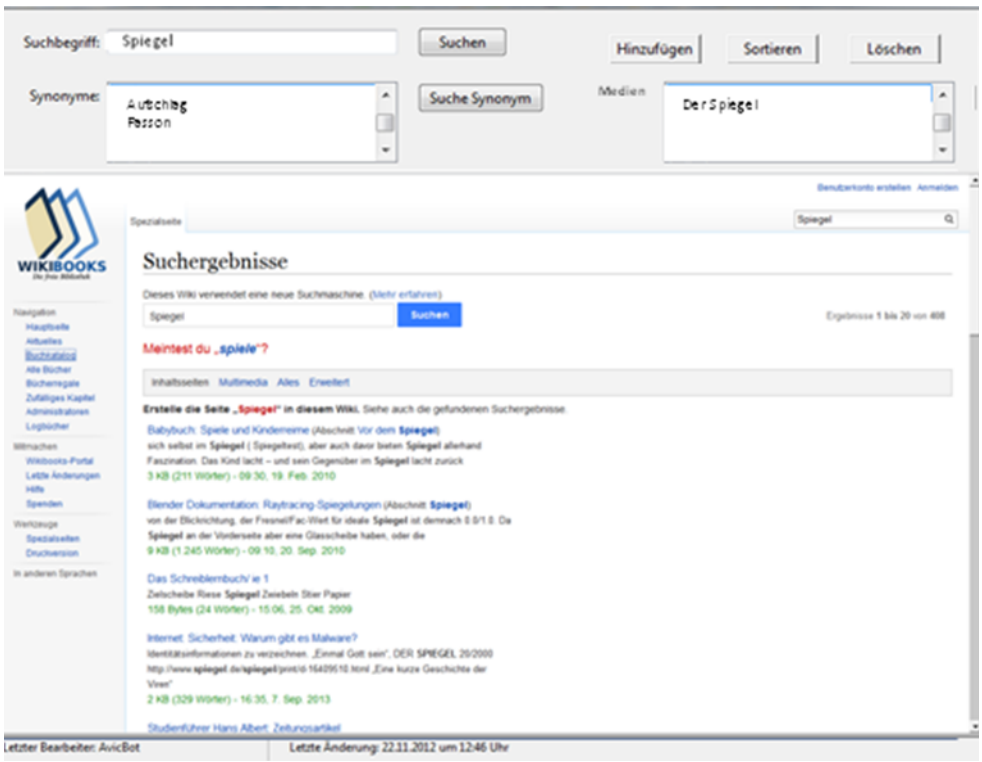

Abbildung 6

Statten Sie Ihr Programm mit einer Fehlermeldung wie in Abbildung 7 aus, wenn es Probleme beim Zugriff auf den Wortschatz-Dienst gibt. Hierzu können Sie die bereits erstellte „MyWebException“ nutzen.

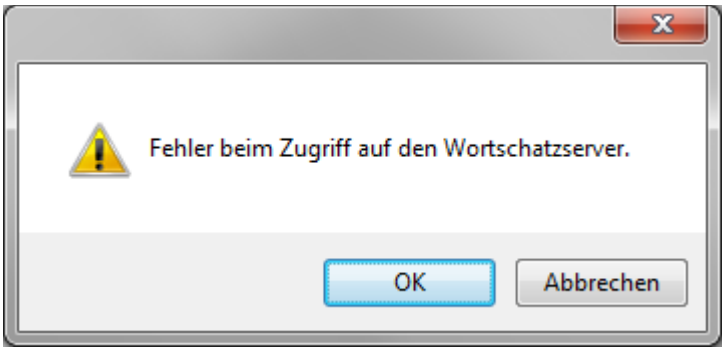

Abbildung 7

Aufgabenteil B (nur wenn Teil A nicht bearbeitet werden kann!**):**

Erweitern Sie die Anwendung um Dialog-Elemente, in denen die Liste aller Wikipedia-Artikel mit Bezug zum gesuchten Begriff und eine Liste aller im Zettelkasten enthaltenen Medien mit passendem Titel angezeigt werden. Diese Listen sollen ungefähr zeitgleich zum unteren Webbrowser-Element aktualisiert werden.

Vorbereitend müssen wir Code schreiben, der es erlaubt Informationen von Wikipedia zu erhalten. Wir nutzen dazu die Wikimedia API (https://www.mediawiki.org/wiki/API%3aMain_page) in Form eines Webservice. Zugriff auf Wikipedia Daten zum Suchbegriff „Java“ erhalten Sie bspw. mittels folgender URL:

https://de.wikipedia.org/w/api.php?action=query&origin=*&format=json&generator=search&gsrnamespace=0&gsrlimit=5&gsrsearch='Java'
Nutzt man die URL in einem Browser erhält man folgendes Ergebnis:

```
{
  "batchcomplete": "",
  "continue": {
    "gsroffset": 5,
    "continue": "gsroffset||"
  },
  "query": {
    "pages": {
      "2512": {
        "pageid": 2512,
        "ns": 0,
        "title": "JavaScript",
        "index": 4
      },
      "2521": {
        "pageid": 2521,
        "ns": 0,
        "title": "Java",
        "index": 3
      },
      "3912": {
        "pageid": 3912,
        "ns": 0,
        "title": "Java (Programmiersprache)",
        "index": 1
      },
      "26349": {
        "pageid": 26349,
        "ns": 0,
        "title": "Java (Insel)",
        "index": 2
      },
      "812354": {
        "pageid": 812354,
        "ns": 0,
        "title": "Java-Technologie",
        "index": 5
      }
    }
  }
}
```


Uns interessieren hier nur die in Wikipedia vorhandenen Titel (hier „Java, Java Script, Java (Progrmmiersprache“ etc.). Zum Auslesen verwenden wir wiederum JSON-Simple. Ein Beispielprogramm zum Zugriff auf Wikipedia mittle JSON-Simple findet sich auf Moodle.

Ergänzen Sie in Ihrer GUI zwei `ListViews`. Diese sollen eine alphabetisch sortierte Liste der Wikipedia-Titel und eine Liste der Titel des Zettelkastens (in Ihrer aktuellen Reihenfolge) enthalten.

Ermöglichen Sie weiterhin, dass der Benutzer durch Auswahl eines Artikels der Liste eine erneute Suche anstoßen kann. Realisieren Sie dies durch einen weiteren Button „Suche Wikipedia“ neben der Liste der Synonyme. Der derart ausgewählte und gesuchte Begriff soll durch Ihr Programm automatisch in das Begriffs-Eingabefeld kopiert und eine neue Suche angestoßen werden. Achten Sie darauf, dass Ihre `ListView` nur die Einfach-Auswahl (`SINGLE`) zulässt.

Ihr Programm soll weiterhin auf einen Doppelklick auf den `ListView` genauso reagieren, als hätte der Benutzer den Button „Suche Wikipedia“ betätigt. Das gewünschte Layout folgt dabei dem aus Aufgabenteil A. Statten Sie Ihr Programm zusätzlich mit einer Fehlermeldung, wenn es Probleme beim Zugriff auf den Wikipedia-Dienst gibt. Hierzu können Sie die bereits erstellte „MyWebException“ nutzen.

7) Nicht vorhandene Synonyme/Artikel behandeln („Enabled“)

[1 Punkt]

Bei Auswahl eines Begriffs ohne Synonym (bzw. Wikipedia_Artikel) soll Ihre Anwendung etwa wie in Abbildung 8 aussehen. In diesem Fall sollen Sie also explizit eine Zeichenkette „`<keine>`“ zum `ListView` hinzufügen.

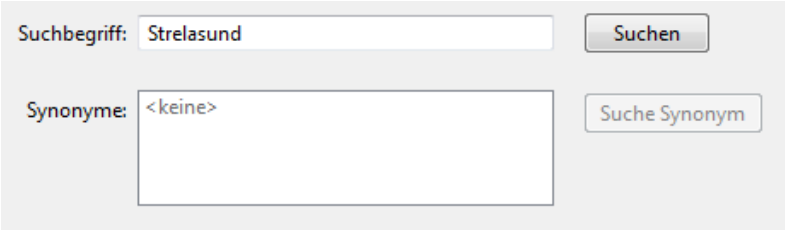

Abbildung 8

Ergänzen Sie Ihr Programm derart, dass, wenn keine Synonyme (Artikel) gefunden wurden, die Schaltfläche „Suche Synonym“ („Suche Wikipedia) und die Liste der Synonyme (Artikel) deaktiviert (ausgegraut) wird. Verwenden Sie dazu die Methode *setDisable(true)* des Steuerelemente. Vergessen Sie nicht, die Dialogelemente wieder zu aktivieren, wenn der Benutzer einen neuen Begriff eingibt, für den Synonyme gefunden wurden.

8) Begriffshistorie

[3 Punkte]

Ergänzen Sie Ihr Programm durch einen „Zurück“- und einen „Vorwärts“-Button. Dazu müssen Sie intern eine Liste der bisher eingegebenen Begriffe verwalten. Wenn der Benutzer auf „Zurück“ klickt, wird der zuletzt eingegebene Begriff angezeigt. So lange, bis der Benutzer beim ersten eingegebenen Begriff angelangt ist. In diesem Moment wird der Zurück-Button von Ihrem Programm deaktiviert. Analog soll der Vorwärts-Button funktionieren. Bitte beachten Sie das jede Navigation innerhalb der Historie **alle** Felder (Webview, Synonyme/Wikipedia, Eingabefeld, etc.) aktualisieren soll.

Vorbild für diese Funktionalität ist der Vor- und Zurück-Button im Firefox.

Ihr Programm soll etwa so aussehen:

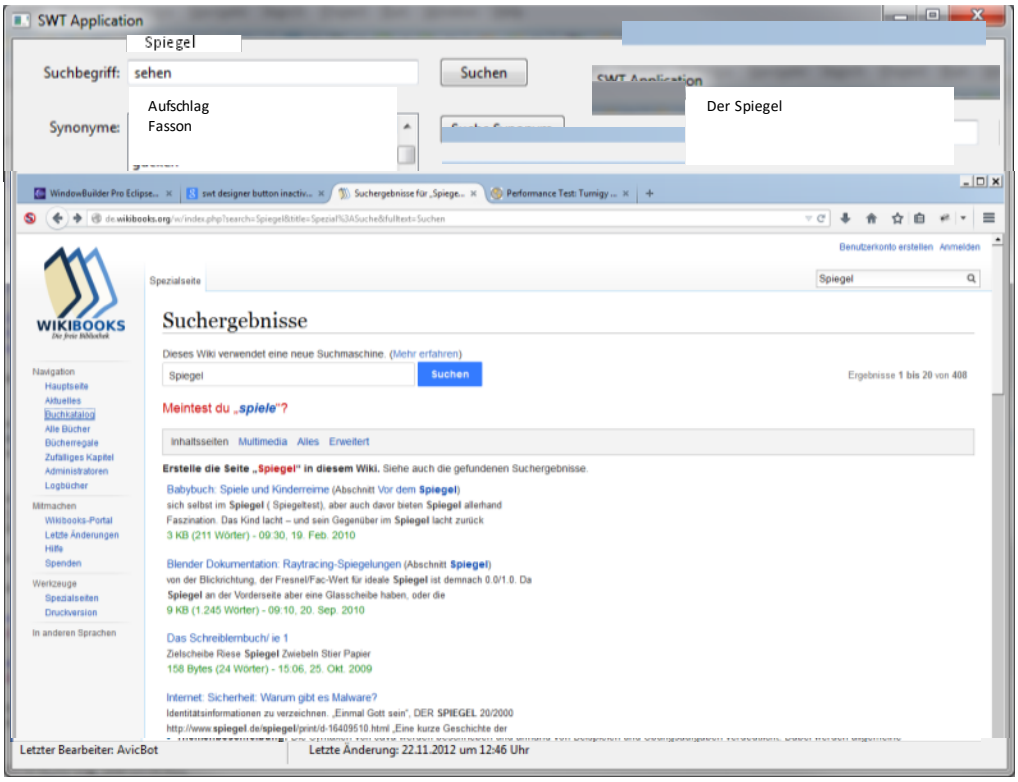

Abbildung 9

Sie sollten in Ihrem Programm neben der Liste, die die Begriffshistorie enthält, auch noch ein Attribut realisieren, das den Index des derzeit angezeigten Begriffs enthält. Zu Beginn ist dieser Index gleich -1 und die Länge der Begriffshistorie ist 0. Immer wenn der Benutzer einen neuen Begriff eingibt, wächst die Begriffshistorie um ein Listenelement und der Index enthält den Index des letzten Begriffs der Begriffshistorie. Wenn der Benutzer zurückblättert, verringert Ihr Programm den Index jeweils um 1. Wenn nun der Benutzer mittendrin einen neuen Begriff eingibt, so muss Ihr Programm die Begriffshistorie „abschneiden“. Alle älteren Begriffe müssen in der Historie erhalten bleiben. Alle jüngeren Begriffe müssen Sie löschen.

Probieren Sie bitte folgenden Testfall durch:

- Eingabe des Begriffs A, „Suchen“ betätigen
- Eingabe des Begriffs B, „Suchen“ betätigen
- Eingabe des Begriffs C, „Suchen“ betätigen
- „Zurück“ betätigen → es erscheint der Artikel B.
- Eingabe des Begriffs D, „Suchen“ betätigen
- „Zurück“ betätigen → es erscheint der Artikel B.
- „Zurück“ betätigen → es erscheint der Artikel A. Der „Zurück“-Button ist disabled
- „Vor“ betätigen → es erscheint der Artikel B. „Zurück“- und „Vor“-Button sind enabled
- „Vor“ betätigen → es erscheint der Artikel D. Der „Vor“-Button ist disabled

9) Combobox aller bisher eingegebenen Begriffe

[2 Punkte]

Realisieren Sie eine JavaFX `ComboBox` in der alle bisher eingegebenen Begriffe verzeichnet sind und dem Nutzer zur Direkt-Auswahl zur Verfügung stehen.

Die Begriffe sollen umgekehrt chronologisch sortiert sein. Ganz oben steht also immer der letzte Begriff, ganz unten der älteste.

Wenn links ein neuer Suchbegriff eingegeben und bestätigt oder ein Synonym selektiert und gesucht wird, dann soll der Inhalt der `ComboBox` aktualisiert werden. Der dann in der `ComboBox` angezeigte (selektierte) Begriff soll dem links als Suchbegriff angezeigten Text entsprechen.

Wenn der Vor- oder Zurück-Button betätigt wird, soll sich Ihr Programm automatisch den richtigen Begriff in der SWT-`Combo` vorselektieren. Dies gilt auch wenn der Begriff per Maus ausgewählt wird.

Eine Beispiel-Sequenz:

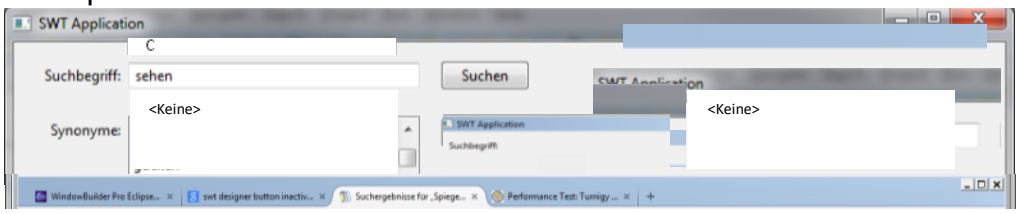

Abbildung 10: Anzeige nach der Eingabe von drei Suchbegriffen und Aufklappen der Combobox

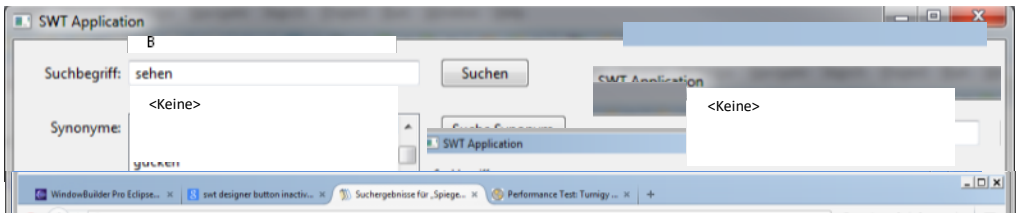

Abbildung 11: Anzeige nach Betätigung des Zurück-Buttons

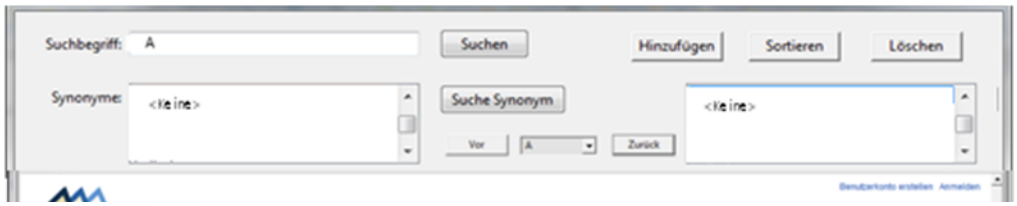

Abbildung 12: Anzeige nach Selektion des ersten Begriffs in der Combobox

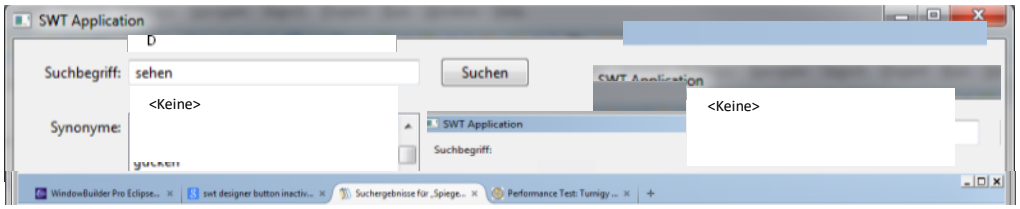

Abbildung 13: Anzeige nach Eingabe und Suche des vierten Begriffs und Aufklappen der Combobox

10) Menü und DialogBox mit Informationen über das Programm

[1 Punkt]

Ergänzen Sie in Ihrem Programm ein Menü. Das Menü soll einen Menüpunkt „?“ mit einem Untermenüpunkt enthalten, so wie es die folgende Abbildung zeigt:

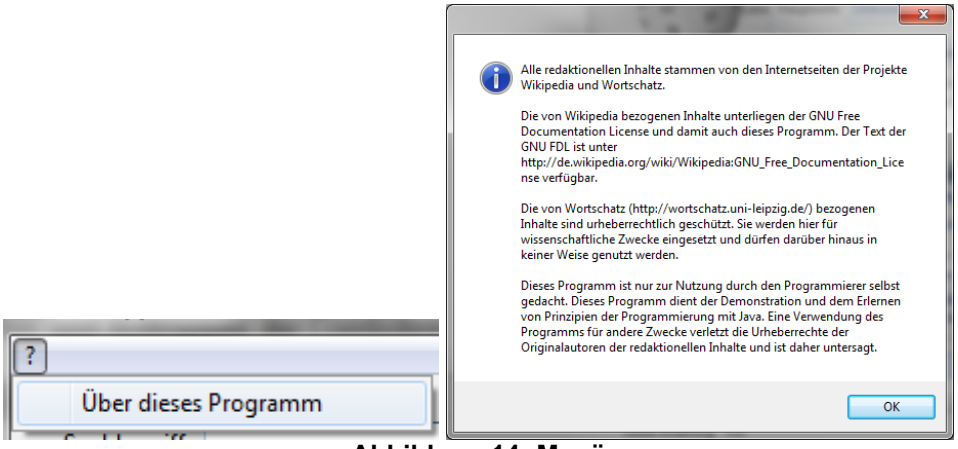

Abbildung 14: Menü

Bei Betätigung des Untermenüpunktes „Über dieses Programm“ soll ein `Information Dialog (Alert mit AlertType.INFORMATION)` mit folgendem Text erscheinen:

> *Alle redaktionellen Inhalte stammen von den Internetseiten der Projekte Wikibooks und Wortschatz.*
>
> *Die von Wikibooks bezogenen Inhalte unterliegen seit dem 22. Juni 2009 unter der Lizenz CC-BY-SA 3.0 Unported zur Verfügung. Eine deutschsprachige Dokumentation für Weiternutzer findet man in den Nutzungsbedingungen der Wikimedia Foundation. Für alle Inhalte von Wikibooks galt bis zum 22. Juni 2009 standardmäßig die GNU FDL (GNU Free Documentation License, engl. für GNU-Lizenz für freie Dokumentation). Der Text der GNU FDL ist unter http://de.wikipedia.org/wiki/Wikipedia:GNU_Free_Documentation_License verfügbar.*
>
> *Die von Wortschatz (http://wortschatz.uni-leipzig.de/) oder Wikipedia (www.wikipedia.de) bezogenen Inhalte sind urheberrechtlich geschützt. Sie werden hier für wissenschaftliche Zwecke eingesetzt und dürfen darüber hinaus in keiner Weise genutzt werden.*
>
> *Dieses Programm ist nur zur Nutzung durch den Programmierer selbst gedacht. Dieses Programm dient der Demonstration und dem Erlernen von Prinzipien der Programmierung mit Java. Eine Verwendung des Programms für andere Zwecke verletzt möglicherweise die Urheberrechte der Originalautoren der redaktionellen Inhalte und ist daher untersagt.*

11) Layout

[1,5 Punkte]

Wenn der Benutzer Ihrer Anwendung das Anwendungsfenster vergrößert oder verkleinert, dann hat dies zurzeit noch keine Auswirkung auf die Größe und Position der Steuerelemente. Wir wollen Ihre Anwendung dafür nun fit machen.

Machen Sie sich dazu mit den Stärken der in JavaFx angebotenen Layouts (Container bzw. Pane) vertraut. Abbildung 15 zeigt, wie das Layout Ihrer Anwendung aussehen könnte.

Im Kopf der GUI finden wir das Menü und die Navigationsleiste. Darunter folgt der Browser (rechts) und der Bereich für die Synonyme bzw. Medien (Titel). Diese Bereiche sind durch einen verschiebbaren Steg voneinander abgegrenzt. Mit dessen Hilfe lässt sich der Bereich horizontal zugunsten des Browsers oder des Synonymbereichs vergrößern (JavaFX: `splitPane`). Im Fuß des Layouts befinden sich die Labels für die Metainformationen des Wikibooks-Eintrags. Für das Anwendungsfenster ist eine Mindestgröße definiert, welche stets alle Bedienelemente sichtbar hält. Vergrößert oder maximiert man das Anwendungsfenster, so wachsen das Eingabefeld für den Suchbegriff (horizontal) sowie der mittlere Bereich für Synonyme und Browser (horizontal und vertikal), so dass die zur Verfügung stehende Größe des Fensters stets sinnvoll ausgenutzt wird.

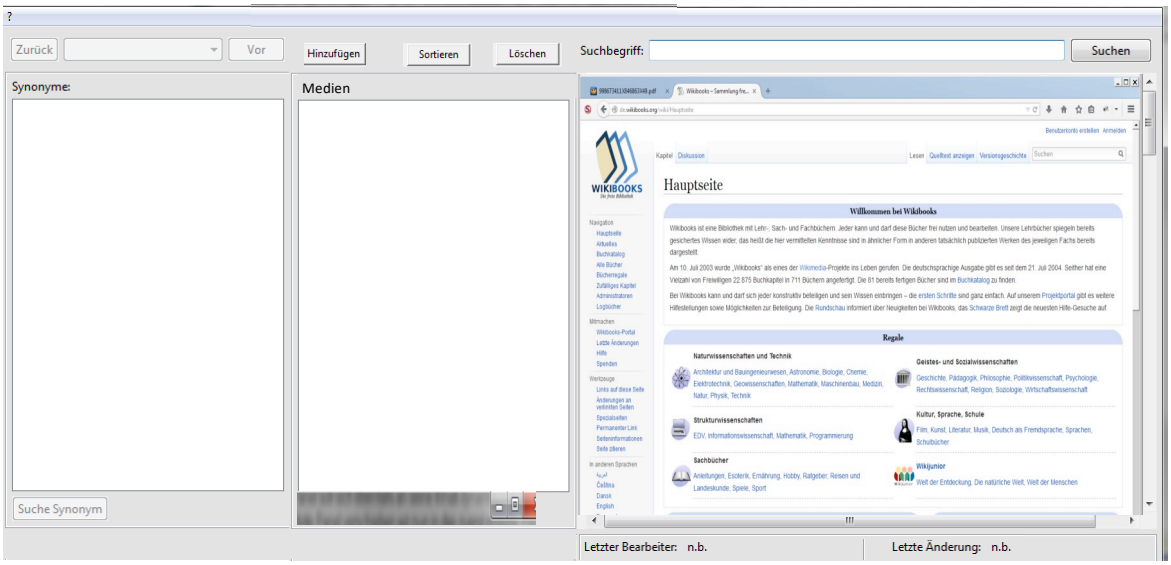

Abbildung 15: Layout in der definierten Mindestgröße

Die Anwendung soll in der Titelleiste einen eigenen Anwendungs-Titel (z.B. „Mein Wikibooks-Browser“) und ein Anwendungs-Icon (z.B. Wikibooks Logo) erhalten.

Ihre Anwendung muss nicht 1:1 gleich aussehen, sollte aber mindestens die „Features“ dieses Layout-Vorschlags erfüllen (sicht- und lesbare Bedienelemente, sinnvolle Nutzung der wachsenden Größe des Anwendungsfensters, Synonyme und Browser teilen sich dynamisch einen Bereich, Mindestgröße des Fensters, Titel und Icon).

12) Tabulator-Reihenfolge und Short-Cut

[1 Punkt]

Nun wollen wir dafür sorgen, dass sich die Anwendung auch nur über die Tastatur bedienen lässt. Bei Drücken der Tab-Taste sollen die folgenden Dialogelemente in einer sinnvollen Reihenfolge abgelaufen werden:

1. Suchbegriff-Textbox
2. Suchen-Button
3. Synonyme-Listbox
4. Suche-Synonym-Button
5. Titel-Listbox
6. Hinzufügen-Button
7. Sortieren-Button
8. Löschen-Button
9. Laden-Button
10. Speichern-Button
11. Menüpunkt „?“
12. Zurück-Button
13. Combobox im Navigationsbereich
14. Vor-Button

Alle anderen Dialogelemente sollen nicht als Tabstopp fungieren. Bitte beachten Sie, dass in JavaFX sich die sog. „TraversalEngine“ darum kümmert, die Elemente eines Layouts zu fokussieren. Normalerweise werden die Elemente dabei in ihrer internen Reihenfolge durchlaufen (Reihenfolge, in der die Container hinzugefügt wurden). Wenn Sie die Reihenfolge ändern möchten können Sie die gewünschte Reihenfolge in Ihrer fxml-Datei herstellen, einen Event-Filter (Abfangen der Tab-Taste und manuelles setzen des Focus) nutzen oder eine eigene Traversal-Engine implementieren (Nutzen der internen API durch Ableiten von com.sun.javafx.scene.traversal.TraversalEngine).

Beim Testen merken wir, dass man ein selektiertes Synonym noch nicht mit ENTER-Taste als neuen Suchbegriff übernehmen kann. Implementieren Sie den zugehörigen Event-Handler.

Geben Sie dem Benutzer noch die Möglichkeit, das Infofenster auch auf kürzerem Wege zu erhalten. Über die Taste F1 soll überall aus der Anwendung heraus der Info-Dialog („Über diese Anwendung…“) erscheinen.

13) BONUS-Aufgabe Browser-Events

[2 Punkte]

Bitte beachten Sie, dass es sich bei dieser Aufgabe um eine Bonusaufgabe handelt. D.h. mit dieser Aufgabe erworbene Punkte können zur Aufwertung des Punktestands genutzt werden. Ein Nicht-bearbeiten, wirkt sich dagegen nicht aus.

Der WebView (mit der zugehörigen WebEngine) in Ihrer Applikation (Anzeige Wikibooks) beinhaltet einen vollwertigen Webbrowser. Dieser erlaubt die Anwahl von Links und das Navigieren im WWW. Ihre Aufgabe ist es

1) Die Anwahl externer Links (Ziele außerhalb von WikiBooks) zu verhindern.
2) Interne Links auszuwerten. Sollte es sich bei der ausgewählten Seite um ein Buch handeln so ist der Titel des Buches in das „Suchfeld“ Ihrer Applikation zu übertragen. Zusätzlich sollen evtl. vorhandene Synonyme angezeigt werden.

Tipp: Hier könnten Ihnen die Bibliothek LibFX oder die Methode locationProperty() der webEngine von Nutzen sein.