

Java Projekt DHBW SoSe 2025

Elias Ciuman
Matrikel-Nr. 6265147

Abgabedatum: 27. August 2025

Inhaltsverzeichnis

1	Einleitung	2
2	Anforderungen	2
2.1	Funktionale Anforderungen	2
2.1.1	Aufgabe 1)	2
2.1.2	Aufgabe 2)	2
2.2	Nicht-funktionale Anforderungen	3
3	Projektarchitektur	3
3.1	Verwendete Technologien	3
3.2	Projektstruktur	4
4	Implementierung	4
4.1	Aufgabe 1)	4
4.2	Aufgabe 2)	5
4.3	Speicherung von Daten	6
5	Benutzung	7
5.1	Installation und Voraussetzung	7
5.2	Bedienungsanleitung	7
6	Fragen	8
6.1	Fragen zu Aufgabe 1	8
6.2	Fragen zu Aufgabe 2	9
7	Anhang	11

Projektdokumentation

1 Einleitung

Im Rahmen des Moduls “Programmieren”, das im ersten Studienjahr an der DHBW Mannheim im Studiengang Angewandte Informatik gelehrt wird, wurde innerhalb der Unit “Programmieren in Java” ein Programmentwurf gefordert.

Dabei soll mithilfe der Google API eine Applikation entwickelt werden. Der Nutzer ist mit dieser in der Lage Google Kalender als iCalendar-Datei zu exportieren, um diese wiederum in einer beliebigen Kalender-App zu importieren. Die genauen Anforderungen werden im nächsten Abschnitt genauer erläutert.

2 Anforderungen

Im Folgenden werden die Anforderungen, welche gestellt wurden genau erläutert und in funktionale Anforderungen, sprich die tatsächlichen Funktionen, welche die Anwendung enthalten muss, unterteilt. Des Weiteren wird ein kurzer Blick auf die nicht-funktionalen Anforderungen geworfen, welche angeben, wie die Anwendung arbeiten soll.

2.1 Funktionale Anforderungen

Die vorliegende Aufgabenstellung wurde in zwei Teilaufgaben unterteilt:

2.1.1 Aufgabe 1)

Ziel der Aufgabe ist es ein Programm zu entwickeln, welches über einen Befehl innerhalb einer Konsole ausgeführt werden soll. Dabei ist vorgegeben die folgenden Argumente des Befehls abzugreifen, um als Ergebnis die geforderte iCal-Datei zu erhalten:

- Klassenname
- Start-Datum
- End-Datum
- Dateiname

2.1.2 Aufgabe 2)

Im zweiten Teil soll eine Benutzeroberfläche zu der bestehenden Anwendung eingefügt werden. Die Startseite enthält die Tabelle mit den Namen der Ereignissen innerhalb des Kalenders, sowie den Start- und Endzeitpunkt. Der Nutzer soll außerdem die Möglichkeit besitzen aus drei Menüs mit jeweiligen Untermenüs auszuwählen:

- File
 - Save - Speichert die iCal-Datei
 - Exit - Beendet das Programm
- Application
 - Setting - Einstellungen, aus welchem Kalender von wann bis wann Events angezeigt werden
- Help
 - About - Zeigt Informationen über den Entwickler an

Initial soll im Hauptfenster nur die Spaltenüberschriften “Ereignis”, “von” und “bis” angezeigt werden. Nach dem die Einstellungen von dem Nutzer durchgeführt wurde, zeigt die Tabelle alle verfügbaren Ereignisse innerhalb des Zeitraums in der Tabelle an. Der Dialog “Setting” gibt dem Nutzer die Möglichkeit alle Einstellungen für den Export zu treffen. Dabei soll die Kalender-URL, sowie der Zeitraum von und bis eingetragen werden. Beim Klicken auf den Button “OK” werden die Einstellungen übernommen, bei “Cancel” wiederum sollen die Einstellungen verworfen werden, dennoch werden die vorherigen Einstellungen belassen.

2.2 Nicht-funktionale Anforderungen

Um die oben genannten Funktionen zu erreichen soll die Google API verwendet werden. Um dies zu erreichen sollen alle nötigen Bibliotheken sowie Gradle installiert werden. Gradle wird in dem dritten Abschnitt erläutert. Die Benutzeroberfläche soll mit einer Bibliothek bspw. Swing oder JavaFX erstellt werden. Es können optionale Features innerhalb des zweiten Aufgabenteils eingefügt werden.

- Input-Validierung
- Dateneingabe: JSpinner-Felder für Datumseingabe
- Speicherung von Einstellungen

3 Projektarchitektur

Dieser Abschnitt handelt von dem grundlegenden Aufbau des gesamten Projekts und die verwendeten Technologien und Bibliotheken.

3.1 Verwendete Technologien

Um die Schnittstelle zwischen den Google-Diensten und dieser Applikation zu erreichen, wurde die **Google-Calendar-API** verwendet. Das Build- und Abhängigkeitsmanagement übernimmt **Gradle**. Für die Erstellung und Darstellung der Benutzeroberfläche kommt **Java Swing** zum Einsatz.

3.2 Projektstruktur

Innerhalb des **src-Ordnern** befindet sich der für diesen Projektentwurf relevanter Quellcode. Dabei wurde darauf geachtet eine klare Struktur der Java-Klassen zu erreichen. So befinden sich innerhalb des **Swing-Ordnern** alle Klassen für Swing-Komponenten, welche verwendet wurden.

```
JavaProject_Ciuman_Elias/  
├── build/  
├── doc/  
├── gradle/  
├── src/  
│   ├── main/  
│   │   ├── java/  
│   │   │   ├── calendar/  
│   │   │   ├── main/  
│   │   │   ├── swing/  
│   │   │   └── util/  
│   │   └── resources/  
├── tokens/  
├── build.gradle  
├── gradlew  
├── gradlew.bat  
├── README.md  
└── settings.gradle
```

4 Implementierung

Dieser Abschnitt befasst sich mit der eigentlichen Bearbeitung und Implementierung der oben genannten Aufgaben.

4.1 Aufgabe 1)

Relevante Klassen: `clsGoogleCalendarService`, `clsCalenderExporter`, `clsMain`

clsGoogleCalendarService Zunächst werden die nötigen Konstanten erstellt, welche für die Initialisierung des Projekts verwendet werden. Darunter fallen Dinge wie der Name der Anwendung, welche zuvor innerhalb der Google-Projekte selbst erstellt wurde, sowie Konstanten zur Verarbeitung der JSON-Datei, welche für die Authentifizierung benötigt wird. Dem folgen drei Methoden:

getCredentials() Diese Methode ermöglicht es, dass dieses Programm auf den Google-Kalender zugreifen kann. Dafür liest diese die Daten aus der `credentials.json` aus um diese in der Autorisierung mit Hilfe von OAuth2 zu verwenden. Außerdem wird ein lokaler Speicher für die Tokens erstellt, um diese verwenden zu können. Danach wird ein lokaler Webserver gestartet, um den OAuth2-Callback von Google zu empfangen, welcher der User im Browser akzeptieren muss, um das Programm verwenden zu können.

getCalendarService() Die einzige Aufgabe dieser Methode ist es einen authentifizierten Google-Calendar-Service aufzubauen, um auf diesen zuzugreifen zu können. Dafür wird die **getCredentials()**-Methode verwendet, um die Authentifizierung durchzuführen.

getEvents() Diese Methode holt alle verfügbaren Events aus der übergebenen Kalender-ID, sowie dem Zeitraum. Dafür wird zunächst der Google-Calendar-Service aufgerufen und der Start- sowie Endzeitpunkt in einer Variable gespeichert. Danach wird eine Abfrage an die Google-Calendar-API gesendet, welche alle Events mit den jeweiligen übergebenen Parametern abrufen. Diese Events werden in einer Map, also in Schlüssel-Wert-Paaren geschrieben. Diese Liste wird zurückgegeben per **return**-Statement

clsCalendarExporter Diese Klasse beinhaltet lediglich die **export()**-Methode, welche mit einer Map von Events sowie dem Namen der zu erstellenden Datei aufgerufen wird. Diese iteriert durch jeden Eintrag dieser Liste und schreibt diese in eine Datei.

clsMain Innerhalb der Main-Klasse werden die oben genannten Klassen eingebunden, sowie die Argumente, welche über die Konsole den Nutzer eingegeben werden verarbeitet und in Variablen gespeichert. Demnach wird zunächst ein Google-Calendar-Service-Objekt erstellt und die **getEvents()**-Methode aufgerufen. Die Events werden in einer Liste gespeichert, welche, nachdem das CalendarExporter-Objekt erstellt wurde, der **export()**-Methode übergeben wird. Nach erfolgreicher Ausführung erhält der Nutzer seine Datei standardmäßig in das Projektverzeichnis abgelegt.

4.2 Aufgabe 2)

Relevante Klassen: DateLabelFormatter, dlgAbout, dlgSettings, MainFrame, clsGoogleCalendarUtil, clsMain

Dieser Aufgabenteil handelt von der Implementierung einer Benutzeroberfläche. Dafür wurde Swing verwendet. Die folgende Auflistung erklärt die jeweiligen Klassen und deren Funktion:

MainFrame Hierbei handelt es sich um das Hauptfenster der Anwendung, welches der Nutzer bei Starten der Anwendung zuerst sieht. Dieses erbt von dem JFrame und nutzt als Layoutmanager das BorderLayout. Initial ist eine leere JTable mit den Spaltenüberschriften zusehen. Genauer dazu findet sich in den Fragen zu Aufgabe 2) im sechsten Abschnitt dieser Dokumentation. Nachdem der User die Einstellungen getroffen hat, wird über eine for-Schleife die Zeilen dynamisch in die Tabelle eingefügt. Außerdem findet sich in dem Hauptfenster die Menüleiste, welche mit einer JMenuBar realisiert wurde. Die einzelnen Menüpunkte werden mit JMenuItem eingebaut. Des Weiteren beinhaltet diese Klasse die Realisierung der Menüpunkte durch ActionListener, welche auf einen Klick des Nutzers reagieren und die entsprechende Aktion ausführen. Beispielsweise für den Export wird dafür auf die zuvor definierten Klassen **clsGoogleCalendarService** und **clsCalendarExporter** sowie deren Methoden zurückgegriffen.

dlgAbout Dieser Dialog wird angezeigt, wenn der Nutzer im Menü auf Help/About klickt. Dieser enthält Informationen zum Entwickler sowie die optionalen Features, welche diese Anwendung enthält. Diese Klasse erbt von `JDialog` und ist ein modaler Dialog, welcher über dem Hauptfenster angezeigt wird und verwendet ein `BorderLayout` um den Inhalt zentriert anzuzeigen. Der Inhalt wird durch ein `JLabel` mit HTML-formatiertem Text angezeigt. Die Schriftgröße passt sich automatisch an die Fenstergröße an.

dlgSettings Hier kann der Nutzer die Einstellungen treffen, um die Events im Hauptfenster anzeigen zu können. Diese Klasse erbt von `JDialog` und ist ein Modal-Dialog, welcher über den Hauptfenster angezeigt wird. Für das Layout wurde ein `BorderLayout`, für das Formular selbst ein `GridLayout` verwendet. Diese Klasse enthält zwei Textfelder für die Kalender-ID und den Namen der Datei, nachdem der Export durchgeführt wurde. Die Dateiendung `.ics` muss nicht vom Nutzer eingegeben werden, diese wird automatisch von der Anwendung angehängt. Für die Realisierung der Datumsfelder wurde ein externes Repository verwendet (`JDatePickerImpl`) um dem Nutzer eine benutzerfreundlichen Datepicker bereitzustellen sowie eine damit einhergehende Input-Validierung. Außerdem befinden sich in diesem Dialog zwei `JButtons` "OK" und "Cancel". Diese Buttons besitzen jeweils einen `ActionListener`. Der OK-Button führt eine Input-Validierung aus. Dafür wird zunächst überprüft, ob der Nutzer jedes Feld ausgefüllt hat. Ansonsten wird eine Fehlermeldung angezeigt. Außerdem wird überprüft, ob das Von-Datum vor dem Bis-Datum liegt. Danach wird die Kalender-ID asynchron überprüft, ob diese gültig ist. Dafür wird ein Loading-Dialog angezeigt, während die Anwendung die ID überprüft. Dafür wird im Hintergrund eine Util-Klasse aufgerufen `clsGoogleCalendarUtils`. Diese überprüft mit der Methode `calendarIdExists(Calendar calendarService, String calendarId)` ob bei einem Aufruf der Kalender-ID ein Fehler mit dem Code 404, sprich der Kalender wurde nicht gefunden, auftritt. Wenn dies passiert, bekommt der Nutzer eine Fehlermeldung, ansonsten wird das Programm ausgeführt und der Nutzer erhält die Tabelle mit den Events.

DateLabelFormatter Diese "Hilfsklasse" wird verwendet für die Anzeige und Eingabe von Datumswerten in `JDatePicker`. Hier werden Strings in Values sowie Values in Strings umgewandelt. Values werden für die Verarbeitung benötigt, Strings für die Anzeige

4.3 Speicherung von Daten

Relevante Klassen: `dlgSettings`, `clsSettingsManager`

Um die Speicherung von eingegebenen Daten auch nach dem Schließen der Anwendung zu realisieren wurde die Klasse `clsSettingsManager` erstellt. Dabei werden die Daten in der `settings.properties` innerhalb des Projektverzeichnisses gespeichert. Diese Klasse enthält zwei Methoden. `loadSettings()` versucht bei der Erstellung des `clsSettingsManager`-Objekts diese Datei zu laden. `saveSettings()` wird von der `dlgSettings`-Klasse aufgerufen, nachdem der Nutzer seine Daten eingegeben hat und auf OK klickt. Die Daten werden in der

zuvor genannten Datei gespeichert, sodass diese erhalten bleiben, auch wenn die Anwendung geschlossen wird.

5 Benutzung

5.1 Installation und Voraussetzung

Da dieses Projekt in eine .jar-Datei gebaut wurde, muss für die Verwendung der Anwendung lediglich Java (mindestens JRE) auf dem Computer installiert sein. Zum Beispiel:

- <https://openjdk.org/>
- <https://www.oracle.com/de/java/>
- ...

Außerdem muss das Projektverzeichnis (über Moodle hochgeladen) heruntergeladen und an einen beliebigen Ort entpackt werden. Es wird die credentials.json mitgeliefert. Dort ist ein Test-Google-Account angelegt, welcher einen Primary-Kalender enthält (für den Konsolen-Aufruf) sowie ein Kalender, welcher mit einer ID aufgerufen werden kann.

5.2 Bedienungsanleitung

Beide Aufgabenteile können mit diesem Projekt ausgeführt werden

Export per Argumenten in der Konsole

1. Konsole (cmd) öffnen
2. Zum Projektverzeichnis mit `cd` navigieren
3. Anwendung mit dem Befehl `java -cp build/libs/JavaProject_Ciuman_Elias-1.0.jar main.clsMain <Start-Datum> <End-Datum> <Dateiname>.ics` aufrufen
4. Dabei das folge Datums-Format verwenden: YYYY-MM-DD
5. Beispiel: `java -cp build/libs/JavaProject_Ciuman_Elias-1.0.jar main.clsMain 2025-07-01 2025-12-31 calendar.ics`
6. Die Datei wird in das Projektverzeichnis abgelegt

Export per GUI

1. Konsole (cmd) öffnen
2. Zum Projektverzeichnis mit `cd` navigieren
3. Die GUI öffnet sich mit dem Befehl `java -cp build/libs/JavaProject_Ciuman_Elias-1.0.jar main.clsMain`
4. Es erscheint das Hauptfenster mit leerer Tabelle

5. Um Einstellungen zu treffen in der Menüleiste auf **Appliaction** dann **Settings**
6. Es öffnet sich ein Modalfenster mit den Einstellungen
7. Dort in das Feld **Kalender-ID** folgende ID eintragen:
`aaed8773fd1b77d61ddbc771aa99a3da048c9ee90107`
`9dd8afb3dfa389417c5c@group.calendar.google.com`
 Die anderen Felder nach Belieben ausfüllen. Es sind Ereignisse im Kalender vom 1.7.25 bis 31.12.25 eingetragen
8. Es wird überprüft, ob der Kalender existiert. Hier kann in der Kalender-ID auf **primary** eingegeben werden, sollte es zu Fehlern kommen
9. Mit OK die Einstellungen bestätigen
10. Die Tabelle ist jetzt mit den Events je nach Eingabe befüllt
11. Über das Menü **File** dann **Save** wird die Datei in das Download-Verzeichnis des Nutzers gespeichert

6 Fragen

In diesem Abschnitt werden die gestellten Fragen zu den jeweiligen Aufgaben beantwortet.

6.1 Fragen zu Aufgabe 1

Frage: Welche Funktion bietet das Tool Gradle? Wofür kann es eingesetzt werden?

Gradle ist ein sogenanntes "Build-Automatisierungs-Tool" unter anderem für Java-Projekte wie dieses. Die Aufgaben, welche dieses Tool übernimmt sind die Kompilierung, Testdurchführungen, Archivierung sowie Deployments auf Basis einer programmierbaren DSL (Groovy oder Kotlin). Gradle wird verwendet um größere Projekte effizient zu bauen. Ein weiterer Vorteil ist die Unterstützung Plugins für Java zu integrieren.

Frage: Was bedeuten die Schritte "API aktivieren" und "OAuth-Zustimmungsbildschirm konfigurieren"?

API aktivieren bedeutet, dass das vorhandene Projekt Dienste der sogenannten Schnittstelle verwenden kann. In diesem Beispiel wird die Google-Calendar-API verwendet. Es muss sichergestellt werden, dass die API aktiviert wurde, um die vorhandenen Dienste und Services dieser API in diesem Projekt verwenden zu können.

OAuth-Zustimmungsbildschirm konfigurieren wird benötigt, damit der Nutzer diesen Bildschirm zu sehen bekommt, wenn er die Anwendung aufruft, um seine Zustimmung zu geben, dass die Google-Calendar-API gewisse Daten zum Ausführen der Anwendung verwenden darf. Dafür muss der Ersteller dieser Anwendung zuvor diesen Zustimmungsbildschirm konfigurieren mit unter anderem dem Anwendungsname oder der Support-E-Mail sowie die angeforderten Zugriffsrechte (Scopes).

Frage: Wie kann man in einer IDE die Google Calendar API verfügbar machen, so dass diese für ein Projekt verwendet werden kann?

Im ersten Schritt muss innerhalb der Google-Cloud ein Projekt erstellt werden und die Google-Calendar-API aktiviert werden. Des Weiteren ist es nötig die OAuth zu aktivieren und die credentials.json herunterzuladen und im Projektverzeichnis unter dem resources-Ordner im src-Ordner zu speichern. Danach muss man die Google Calendar-Client Bibliothek in das Projekt einbinden. In diesem Fall wird Gradle verwendet. Folgender Code-Abschnitt wird dafür in die gradle.build eingefügt werden.

```
implementation 'com.google.apis:google-api-services-calendar
:v3-rev20230808-2.0.0'
implementation 'com.google.api-client:google-api-client
:2.2.0'
implementation 'com.google.oauth-client:google-oauth-client-
jetty:1.34.1'
```

Frage: Erläutern Sie die Code-Zeilen final NetHttpTransport HTTP TRANSPORT = GoogleNetHttpTransport.newTrustedTransport(); Calendar service = new Calendar.Builder(HTTP TRANSPORT, JSON FACTORY, getCredentials(HTTP TRANSPORT)) .setApplicationName(APPLICATION NAME) .build();

Diese Code-Zeilen erstellen einen Calendar-Service um auf den Google-Kalender in der Anwendung zugreifen zu können. Dafür wird zunächst die Transport-Schicht für HTTP-Anfragen erstellt. Dafür wird ein Objekt erzeugt, mit SSL-Zertifikaten, sodass HTTPS verwendet werden kann. Das Schlüsselwort final bedeutet, dass HTTP_TRANSPORT im weiteren Verlauf des Programms nicht mehr verändert werden kann. Danach wird ein Calendar-Objekt erstellt. Calendar.Builder() erstellt einen Builder für den Google Calendar Service. Dabei erhält dieser die folgenden Parameter:

- HTTP_TRANSPORT - zuvor erstellte HTTPS-Verbindung
- JSON_FACTORY - JSON-Verarbeitungsobjekt, konvertiert Daten zwischen Java-Objekten und JSON.
- getCredentials(HTTP_TRANSPORT) - Authentifizierungsdaten (OAuth 2.0 Credentials), damit API auf gegebenen Google-Account zugreifen darf.

.setApplicationName(APPLICATION_NAME) setzt den Namen der Anwendung aus der Variable APPLICATION_NAME und .build() baut final das Calendar-Service-Objekt, um diesen dann für die weitere Logik der Anwendung zu verwenden.

6.2 Fragen zu Aufgabe 2

Frage: Wie definiert man Zeilen/Spalten für ein JTable-Objekt?

Dafür definiert man zunächst ein TableModel. Die Spalten werden durch ein String-Array beim Erstellen dieses TableModels angegeben

```
String[] columns = {"Ereignis", "von", "bis"};
```

Dieses Array wird dem TableModel übergeben. Zeilen werden dem Model mit `addRow()` hinzugefügt. Dabei werden innerhalb der runden Klammern die Daten der jeweiligen Spalten in der Zeile angegeben.

Frage: Was muss beachtet werden, wenn bei einem JTable die Spaltenüberschriften angezeigt werden sollen

Dafür ist es nötig, die JTable innerhalb eines JScrollPane einzufügen. Das JScrollPane sorgt dafür, dass die Spaltenüberschriften sichtbar werden (JTable-Header).

Frage: Welche Layoutmanager haben Sie für welchen Dialog/welches Fenster verwendet? Begründen Sie.

Dialog/Fenster	Layoutmanager	Begründung
Mainframe/headerPanel	BorderLayout/BoxLayout	BorderLayout ist gut, um Hauptbereiche klar zu strukturieren / BoxLayout sorgt für saubere vertikale Ausrichtung der Überschriften
dlgSettings	GridLayout	GridLayout ist ideal für gleichmäßig verteilte Formularelemente
dlgAbout	BorderLayout	BorderLayout ermöglicht eine einfache Zentrierung des Inhalts im Dialog

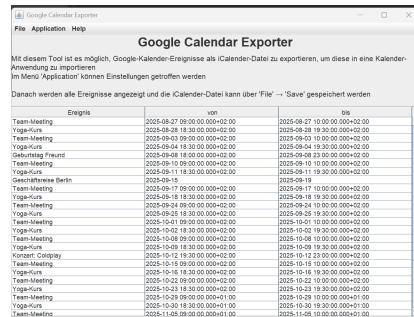
Frage: Wie definiert man, welche Aktion bei der Auswahl eines Menüs ausgeführt werden soll?

Um die richtigen Aktionen bei der Auswahl eines Menüpunkts auszuführen werden **ActionListener** verwendet. Dem Listener wird die gewünschte Aktion übergeben und mit `addActionListener` mit dem dazugehörigen JMenuItem verbunden.

```
JMenuItem saveItem = new JMenuItem("Save");
saveItem.addActionListener(e -> {
    // Hier steht die Aktion, die beim Klicken auf Save
    // ausgeführt werden soll
});
```

7 Anhang

Screenshots aus der Anwendung



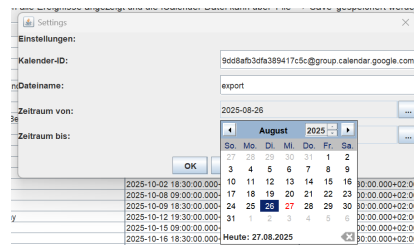
Google Calendar Exporter

Mit diesem Tool ist es möglich, Google-Kalender-Ereignisse als iCalendar-Datei zu exportieren, um diese in eine Kalender-Anwendung zu importieren.
Im Menü 'Application' können Einstellungen getroffen werden.

Danach werden alle Ereignisse angezeigt und die iCalendar-Datei kann über 'File' -> 'Save' gespeichert werden.

Ereignis	von	bis
Team-Meeting	2025-08-27 09:00:00.000+02:00	2025-08-27 10:00:00.000+02:00
Yoga-Kurs	2025-08-28 18:30:00.000+02:00	2025-08-28 19:30:00.000+02:00
Team-Meeting	2025-08-03 09:00:00.000+02:00	2025-08-03 10:00:00.000+02:00
Yoga-Kurs	2025-08-04 18:30:00.000+02:00	2025-08-04 19:30:00.000+02:00
Geburtstag Freund	2025-08-08 18:00:00.000+02:00	2025-08-08 23:00:00.000+02:00
Team-Meeting	2025-08-10 09:00:00.000+02:00	2025-08-10 10:00:00.000+02:00
Yoga-Kurs	2025-08-11 18:30:00.000+02:00	2025-08-11 19:30:00.000+02:00
Stresskurse Berlin	2025-08-18	2025-08-18
Team-Meeting	2025-08-17 09:00:00.000+02:00	2025-08-17 10:00:00.000+02:00
Yoga-Kurs	2025-08-18 18:30:00.000+02:00	2025-08-18 19:30:00.000+02:00
Team-Meeting	2025-08-24 09:00:00.000+02:00	2025-08-24 10:00:00.000+02:00
Yoga-Kurs	2025-08-25 18:30:00.000+02:00	2025-08-25 19:30:00.000+02:00
Team-Meeting	2025-10-01 09:00:00.000+02:00	2025-10-01 10:00:00.000+02:00
Yoga-Kurs	2025-10-02 18:30:00.000+02:00	2025-10-02 19:30:00.000+02:00
Team-Meeting	2025-10-08 09:00:00.000+02:00	2025-10-08 10:00:00.000+02:00
Yoga-Kurs	2025-10-09 18:30:00.000+02:00	2025-10-09 19:30:00.000+02:00
Koncert: Cuijman	2025-10-12 18:30:00.000+02:00	2025-10-12 23:00:00.000+02:00
Team-Meeting	2025-10-15 09:00:00.000+02:00	2025-10-15 10:00:00.000+02:00
Yoga-Kurs	2025-10-16 18:30:00.000+02:00	2025-10-16 19:30:00.000+02:00
Team-Meeting	2025-10-22 09:00:00.000+02:00	2025-10-22 10:00:00.000+02:00
Yoga-Kurs	2025-10-23 18:30:00.000+02:00	2025-10-23 19:30:00.000+02:00
Team-Meeting	2025-10-29 09:00:00.000+01:00	2025-10-29 10:00:00.000+01:00
Yoga-Kurs	2025-10-30 18:30:00.000+01:00	2025-10-30 19:30:00.000+01:00
Team-Meeting	2025-11-05 09:00:00.000+01:00	2025-11-05 10:00:00.000+01:00

(a) Tabelle mit Events



Settings

Einstellungen:

Kalender-ID: 9dd8afb3dfa389417c5c@group.calendar.google.com

Dateiname: export

Zeitraum von: 2025-08-26

Zeitraum bis: 2025-08-26

OK

2025-10-02 18:30:00.000+02:00 10 11 12 13 14 15 16 30.00.000+02:00

2025-10-08 09:00:00.000+02:00 17 18 19 20 21 22 23 30.00.000+02:00

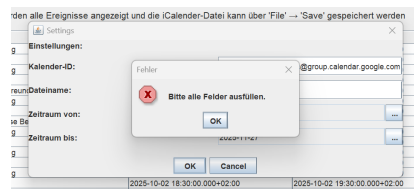
2025-10-09 18:30:00.000+02:00 24 25 26 27 28 29 30 30.00.000+02:00

2025-10-12 19:30:00.000+02:00 31 1 2 3 4 5 6 30.00.000+02:00

2025-10-15 09:00:00.000+02:00 30.00.000+02:00

2025-10-16 18:30:00.000+02:00 Heute: 27.08.2025 30.00.000+02:00

(b) Einstellungen



Settings

Einstellungen:

Kalender-ID: @group.calendar.google.com

Dateiname: export

Zeitraum von: 2025-08-26

Zeitraum bis: 2025-08-26

OK

2025-10-02 18:30:00.000+02:00 10 11 12 13 14 15 16 30.00.000+02:00

2025-10-08 09:00:00.000+02:00 17 18 19 20 21 22 23 30.00.000+02:00

2025-10-09 18:30:00.000+02:00 24 25 26 27 28 29 30 30.00.000+02:00

2025-10-12 19:30:00.000+02:00 31 1 2 3 4 5 6 30.00.000+02:00

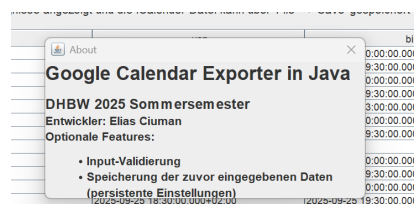
2025-10-15 09:00:00.000+02:00 30.00.000+02:00

2025-10-16 18:30:00.000+02:00 Heute: 27.08.2025 30.00.000+02:00

Fehler: Bitte alle Felder ausfüllen.

OK

(c) Fehlermeldung wenn nicht alle Felder ausgefüllt



About

Google Calendar Exporter in Java

DHBW 2025 Sommersemester

Entwickler: Elias Ciunan

Optionale Features:

- Input-Validierung
- Speicherung der zuvor eingegebenen Daten (persistente Einstellungen)

2025-10-02 18:30:00.000+02:00 10 11 12 13 14 15 16 30.00.000+02:00

2025-10-08 09:00:00.000+02:00 17 18 19 20 21 22 23 30.00.000+02:00

2025-10-09 18:30:00.000+02:00 24 25 26 27 28 29 30 30.00.000+02:00

2025-10-12 19:30:00.000+02:00 31 1 2 3 4 5 6 30.00.000+02:00

2025-10-15 09:00:00.000+02:00 30.00.000+02:00

2025-10-16 18:30:00.000+02:00 Heute: 27.08.2025 30.00.000+02:00

(d) About-Dialog

Abbildung 1: Screenshots aus der Anwendung

Klassendiagramm für Aufgabe 1

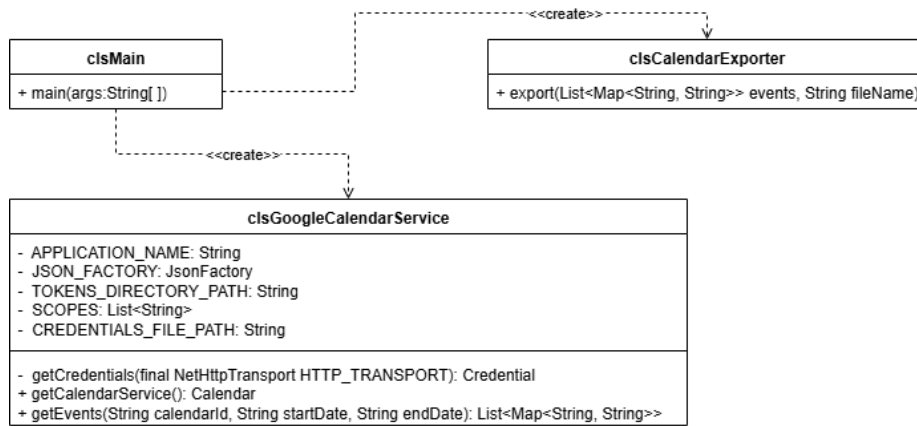


Abbildung 2: Klassendiagramm Aufgabe 1

Aktivitätsdiagramm Export

Export mit Hilfe der Benutzeroberfläche aus Sicht des Nutzers

Nutzer starten Programm über Konsole

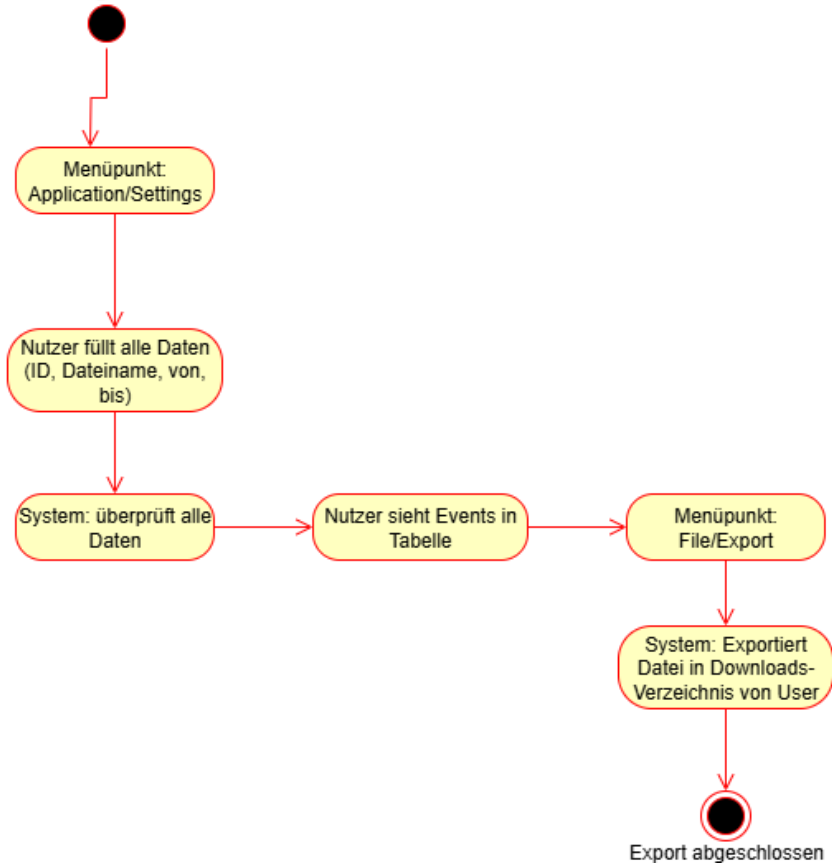


Abbildung 3: Aktivitätsdiagramm Export