

Praktikum GUI

Einleitung

Das folgende Praktikum ist in zwei Teile unterteilt.

Im ersten Teil ergänzen Sie das Resultat des Workshops um ein Model.

Im zweiten Teil wird eine Applikation erstellt. Diese Applikation stellt keine Ansprüche an ein in der Praxis taugliches Tool. Bestimmt gibt es bessere Tools, die in den entsprechenden Bereichen eingesetzt werden. Es geht darum, dass Sie die in der Vorlesung vermittelten Inhalte ausprobieren und umsetzen können.

Ziele

- Sie können ein einfaches JavaFX GUI nach Vorgabe erstellen.
- Sie setzen die zur Verfügung stehenden Container (Panels) korrekt ein.
- Sie finden die entsprechenden Einstellungen in der Doku.
- Sie testen verschiedene Möglichkeiten für die Darstellung und das Event-Handling aus.
- Sie wenden das Model-View-Control Pattern korrekt an.
- Sie können ein GUI mit dem [JavaFX Scene Builder](#) erstellen und FXML Dateien in die Anwendung integrieren.

Tooling

- Installiertes JDK 17+
- Gradle 7.6+
- JavaFX Scene Builder

Der [JavaFX Scene Builder](#) wird im zweiten Teil des Praktikums verwendet. Hinweise zur Installation und finden sie dort.

Struktur

Ein Praktikum kann verschiedene Arten von Aufgaben enthalten, die wie folgt gekennzeichnet sind:

[TU] – Theoretische Übung

Dient der Repetition bzw. Vertiefung des Stoffes aus der Vorlesung und als Vorbereitung für die nachfolgenden Übungen.

[PU] – Praktische Übung

Übungsaufgaben zur praktischen Vertiefung von Teilaspekten des behandelten Themas.

[PA] – Pflichtaufgabe

Übergreifende Aufgabe zum Abschluss. Das Lösen dieser Aufgaben ist Pflicht. Sie muss bis zum definierten Zeitpunkt abgegeben werden, wird bewertet und ist Teil der Vornote.

Zeit und Bewertung

Für dieses Praktikum stehen 2 Wochen in den Praktikumslektionen und im Selbststudium zur Verfügung.

Je nach Kenntniss- und Erfahrungsstufe benötigen Sie mehr oder weniger Zeit. Nutzen Sie die Gelegenheit den Stoff zu vertiefen, auszuprobieren, Fragen zu stellen und Lösungen zu diskutieren (Intensive-Track).

Falls Sie das Thema schon beherrschen, müssen Sie nur die Pflichtaufgaben lösen und bis zum angegebenen Zeitpunkt abgeben (Fast-Track).

Die Pflichtaufgabe wird mit 0 bis 2 Punkten bewertet (siehe *Leistungsnachweise* auf Moodle).

Referenzen

- Praktikumsverzeichnis – Quellcode, Projektstruktur
- OpenJFX Webseite
- OpenJFX Gradle Konfiguration
- OpenJFX Gradle Plugin
- JavaFX 17 Javadoc Dokumentation
- JavaFX Scene Builder Website

1. Erweitern des Workshops um ein Model [PU]

Setzen Sie aufbauend auf der Applikation aus dem Workshop eine Modellklasse (Model) ein und verbinden Sie dieses gemäss den Vorgaben von MVC mit der Applikation.

1.1. Ziel

Die Applikation WordCloud soll, beim Eingeben von Texten, diese einzelnen Wörter nicht mehr direkt in der History anzeigen.

- In der View wird ein Button geklickt.
 - Die Aktion am Controller setzt nicht mehr die View direkt, sondern ändert nur noch das Model.
- Gemäss MVC darf das Model nicht auf die View zugreifen:
 - Wir brauchen eine Möglichkeit, den Controller zu informieren, wenn er den Inhalt der View auf den neuesten Stand bringen soll.

1.2. Basis

- Verwenden Sie als Model die im [Praktikumsverzeichnis](#) zur Verfügung gestellten Klassen `WordModel`, `WordModelDecorator` sowie die beiden Interfaces `IsObservable` und `IsObserver` aus dem Projekt FXML-WordCloud.

1.3. Aufgabenstellung

- Öffnen Sie das mit dem Workshop erstellte Projekt (oder arbeiten Sie den Workshop durch, um das Projekt zu erhalten).
- Ergänzen Sie das Projekt um die Model Klasse `WordModel`, den Decorator `WordModelDecorator` und die beiden Interfaces.
- Studieren Sie die zur Verfügung gestellten Klassen und Interfaces. **Sie sollen nicht geändert werden!**
 - `WordModel` enthält keine Hinweise auf das verwendete UI.
 - `WordModelDecorator` fügt die Möglichkeit zum Beobachten eines `WordModel` Objekts hinzu (Beobachter hinzufügen und entfernen).
 - Überlegen Sie sich, wie hier das Observer-Pattern umgesetzt wurde.
- Ändern Sie den Controller so ab, dass die Vorgaben im Ziel erreicht sind.



Um die einzelnen Worte ins `WordModel` zu schreiben, müssen Sie die Methode `addWord(String word)` für jedes eingegebene Wort einzeln aufrufen. Am einfachsten nehmen Sie die Worte in Kleinbuchstaben.

2. Die Übungsapplikation – ROI Calculator

Die Aufgaben in diesem Teil basieren auf der nachfolgend beschriebenen Übungsapplikation, die sie auf verschiedene Arten umsetzen.

2.1. Kurzbeschreibung

Der Capital Assets Calculator soll auf Basis von 4 Eingabewerten das voraussichtliche Kontovermögen für die kommenden Jahre berechnen. Die Eingabewerte sind:

- Kontovermögen beim Start (Initial amount)
- Jahreszins, der erwartet wird (Return in %)
- Kontoverwaltungskosten pro Jahr (Annual cost)
- Anzahl der Jahre, die berechnet werden sollen (Number of years)

Die daraus errechnete Vermögensentwicklung (pro Jahr eine Zeile) soll im Resultatbereich ausgegeben werden.

- Die Applikation soll die Eingabewerte prüfen und im Falle von ungültigen Werten eine Fehlermeldung im Resultatbereich ausgeben.
- Die Applikation soll ein Menü zur Verfügung stellen, welches ermöglicht:
 - die eingegebenen Werte zu löschen
 - den Resultatbereich zu löschen
 - einen Hilfetext im Resultatbereich anzuzeigen.

2.2. Anforderungen an die Applikation

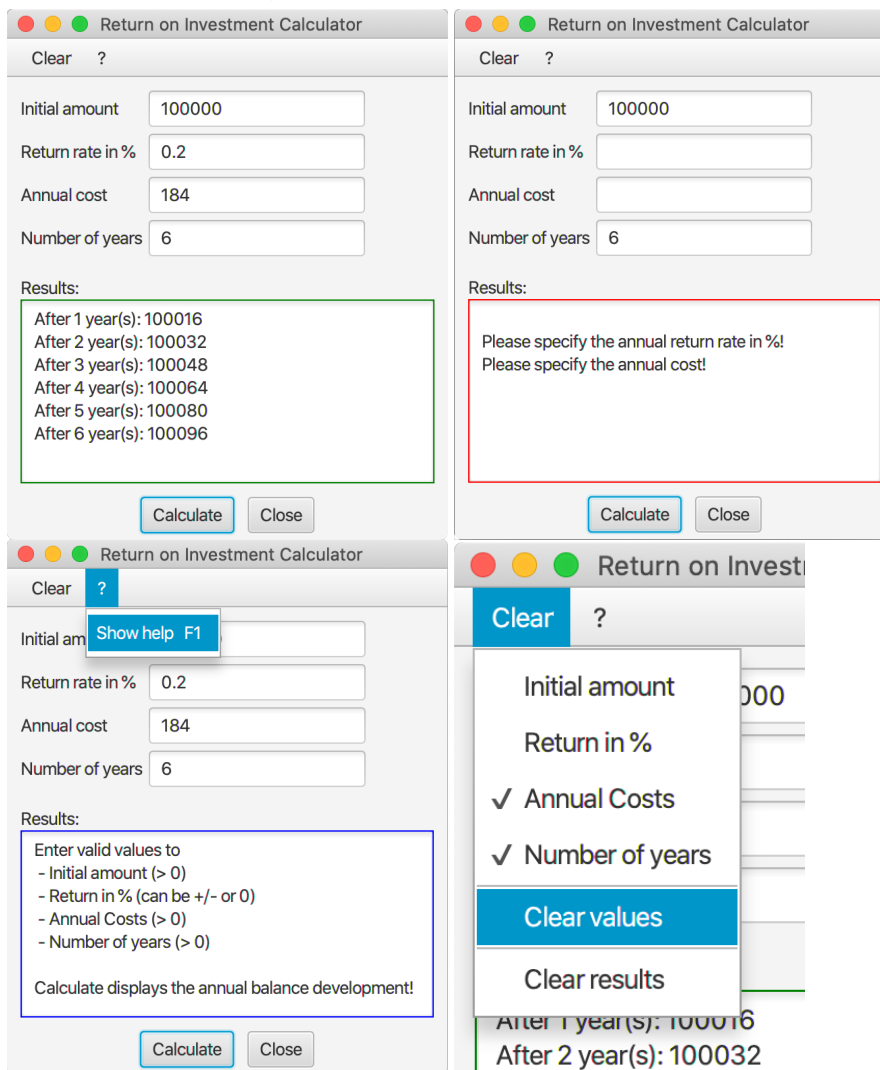
Bitte beachten Sie die weiter unten folgenden Bilder der Applikation in den verschiedenen Zuständen.

- Nach dem Start der Anwendung wird das Hauptfenster angezeigt, in welchem der Benutzer folgende Aktionen ausführen kann:
 - Die 4 Werte eingeben.
 - Mit **[Calculate]** die Berechnung starten
 - Nach der Berechnung wird das Resultat im Resultatbereich ausgegeben
 - Der Rahmen des Resultatbereichs wird grün, wenn alles i.O. war
 - Ist ein Eingabewert falsch oder unvollständig, so wird eine Meldung ausgegeben (roter Rahmen)
 - Initial amount: > 0
 - Return in %: Darf nicht leer sein
 - Annual cost: > 0 und darf nicht leer sein
 - Number of years: >0 und < 99, nur ganze Zahlen
 - Das Menü **Clear** enthält 6 Einträge:
 - Je einen Eintrag pro Eingabewert (diese können gewählt oder abgewählt werden), z.B. **Clear > ✓ Initial amount**
 - Einen Eintrag **Clear > Clear values**, der alle Eingabewerte, die im Menu angewählt sind, wieder auf leer setzt
 - Einen Eintrag **Clear > Clear results**, der den Resultatbereich wieder zurücksetzt.
 - Das Menü **?** enthält einen Eintrag:
 - **? > Show help** zeigt einen Hilfetext im Resultatfenster an (blauer Rahmen).
 - Dies soll auch über die Taste **F1** ausgelöst werden können.
 - Mit **[Close]** wird die Applikation geschlossen.



Berechnen Sie den Wert des Vermögens für jedes Jahr mit: $\text{Amount} = \text{Amount} * (100\% + \text{ReturnIn}\%) - \text{AnnualCost}$ oder verwenden Sie die bereitgestellte Klasse `ValueHandler`.

Bilder als Vorlage

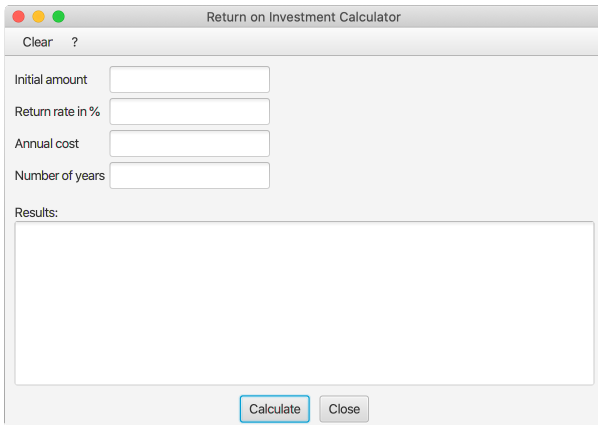


2.3. Überlegungen zum Einsatz von Containern (Panels)

In der Vorlesung wurden verschiedene Container vorgestellt, die Controls oder weitere Container enthalten können. Die Kombination dieser Container trägt maßgeblich zum Verhalten und zur Gestaltung einer Applikation bei. Vorgestellt wurden z.B.:

GridPane	Ordnet die Inhalte in Zeilen und Spalten an
HBox	Ordnet die Inhalte in einer Zeile
VBox	Ordnet die Inhalte in einer Spalte
BorderPane	Ordnet die Inhalte in 5 Bereichen: Left, Right, Top, Bottom und Center
...	weitere können Sie im Manual oder im Unterrichtsstoff nachschlagen...

Für die Benutzeroberfläche der zu erstellende Applikation sind die folgenden Anforderungen bekannt:



Return on Investment Calculator

Clear ?

Initial amount

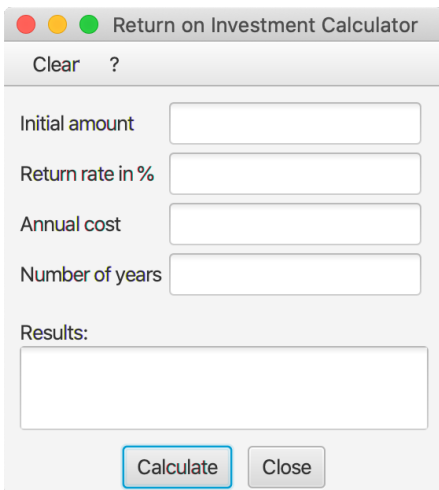
Return rate in %

Annual cost

Number of years

Results:

Der Resultatbereich füllt den Platz des Fensters aus.
Die Eingabefelder und die Beschriftungen sind links oben angeordnet und bleiben in der Grösse konstant.
Das Menü ist oben-links angeordnet, die Buttons bleiben zentriert.



Return on Investment Calculator

Clear ?

Initial amount

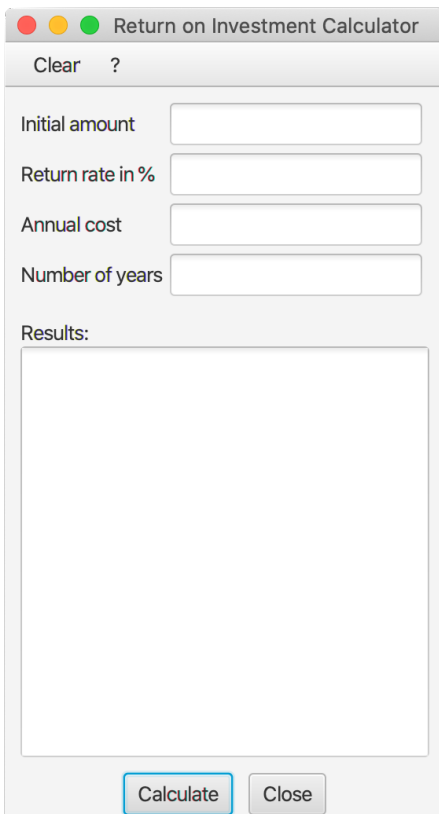
Return rate in %

Annual cost

Number of years

Results:

Beim Verkleinern des Fensters verkleinert sich der Resultatbereich.
Die Breite des Fensters soll limitiert werden, so dass die Beschriftungen lesbar bleiben.



Return on Investment Calculator

Clear ?

Initial amount

Return rate in %

Annual cost

Number of years

Results:

Wird die Höhe des Fensters vergrößert, wächst der Resultatbereich mit.

3. Erstellen des ROI-Calculators mit Object-SceneGraph

3.1. Planung des Layouts [TU]

- Überlegen Sie sich, welche Container sie verwenden möchten, um die Anforderungen zu erfüllen
- Zeichnen Sie mindestens zwei Möglichkeiten auf
- Beschreiben Sie die Vor- und Nachteile, die sich aus Ihren Überlegungen ergeben könnten



Beachten Sie vor allem die Positionierung, die Abstände und die Ausrichtung der Controls. Vergessen Sie auch das Menü nicht.

3.2. Erstellen der Applikation [PU]

Basis

Als Basis finden Sie im [Praktikumsverzeichnis](#) das Projekt **Calculator** mit zwei bereitgestellten Klassen. Sie sind nicht verpflichtet, diese Klassen zu verwenden. Es steht Ihnen frei, eine eigene Struktur aufzubauen oder die Struktur zu erweitern.

Main

Diese Klasse enthält das Grundgerüst der Hauptanwendung und muss erweitert werden.

ValueHandler

Diese Klasse bietet Ihnen Hilfsfunktionen für die Prüfung der Eingabewerte und die Berechnung der Resultate.

Umsetzung

- Ergänzen Sie die Projektkonfiguration (Gradle) für den Einsatz von [JavaFX](#)



Seit Java 11 ist JavaFX nicht mehr Teil der Java Standard Edition und wird unabhängig davon im [OpenJFX-Projekt](#) weiterentwickelt.

Für Projekte die Java 11 und neuer verwenden, muss es deshalb explizit zum Projekt hinzugefügt werden.

Da JavaFX neben generischen Libraries (API, Controls, FXML) auch Plattform-/Betriebssystemabhängige Komponenten benötigt, wird für Gradle (und Maven) ein PlugIn zur Verfügung gestellt, welches die notwendigen Komponenten (Dependencies) lädt und in den Prozess einbindet.

- Deklarieren und konfigurieren Sie das JavaFX-Plugin gemäss der [OpenJFX Anleitung für Gradle](#).
 - Verwenden Sie die JavaFX-Version '17' (oder neuer)
- Erstellen Sie das Layout für die Applikation
 - Bauen Sie den SceneGraph für die Anwendung gemäss Ihren Überlegungen aus der theoretischen Aufgabe zusammen und binden Sie diesen ins Hauptfenster ein.
 - Fügen Sie die Handler für die benötigten Events hinzu.
 - Entscheiden Sie, ob Sie innere Klassen verwenden wollen oder ob Sie mit anonymen Klassen arbeiten.
 - Auch eine Kombination ist möglich (und eventuell sinnvoll).
 - Probieren Sie verschiedene Lösungsansätze und hinterfragen Sie die Vor- und Nachteile der gewählten Lösung.

4. Erstellen des ROI-Calculators mit FXML [PA]

Nachdem Sie sich bis hierher einige Gedanken zur Verwendung der Container (Panels) gemacht haben und praktische Erfahrung mit dem Aufbau von JavaFX User Interfaces gesammelt haben, sind Sie nun bereit für die Umsetzung der Benutzeroberfläche mit FXML unter Verwendung des [Scene Builder](#).



FXML unterstützt die Trennung von View und Controller. Die Behandlung von Aktionen sollen deshalb konsequent in Methoden des Controllers ausgelagert werden und die Aktualisierung der GUI-Komponenten mittels JavaFX-Properties erfolgen.

SceneBuilder

SceneBuilder ist ein Werkzeug zur Bearbeitung von FXML-Dateien und ermöglicht somit die Erstellung und Bearbeitung eines SceneGraphs und die Verknüpfung mit einer zugehörigen Controller-Klasse.

SceneBuilder muss als eigenständiges Werkzeug [heruntergeladen und installiert](#) werden.



IDE's können den SceneBuilder einbinden (FXML-Code-Ansicht & Scene Builder-Ansicht), bzw. liefern bereits eine Version von SceneBuilder mit (IntelliJ). Diese bietet jedoch meist nur einen eingeschränkten Funktionsumfang, weshalb es bei intensiver Arbeit trotzdem Sinn macht, die unabhängige Version zu installieren und aus der IDE aufzurufen (Dateipfad in Einstellungen konfigurieren und mit "Open In SceneBuilder" öffnen)

4.1. Basis

Als Basis finden Sie im [Praktikumsverzeichnis](#) das Projekt **FXML-Calculator** mit zwei bereitgestellten Klassen und einer leeren FXML-Datei

`src/main/resources/ch/zhaw/prog2/fxmlcalculator/MainWindow.fxml`.



FXML-Dateien werden nicht kompiliert, müssen jedoch zur Laufzeit im Klassenpfad zur Verfügung stehen. Gradle (und auch Maven) erwartet deshalb, dass Sie im Ressourcen-Ordner (default: `src/main/resources/`) abgelegt werden (am besten im gleichen Package bzw. Unterverzeichnis wie die zugehörigen Klassen `ch/zhaw/prog2/fxmlcalculator/`). Beim Erstellen der Anwendung werden Sie automatisch kopiert und beim Start in den Klassenpfad integriert.

Sie sind nicht verpflichtet, diese Klassen zu verwenden. Es steht Ihnen frei, eine eigene Struktur aufzubauen oder die Struktur zu erweitern.

Main

Diese Klasse enthält das Grundgerüst der Hauptanwendung und muss erweitert werden.

MainWindowController

Diese Klasse ist fast leer und muss erweitert werden. Sie ist insbesondere noch nicht mit dem Layout (`MainWindow.fxml`) verbunden.

ValueHandler

Diese Klasse enthält Hilfsfunktionen für die Prüfung der Eingabewerte und die Berechnung der Resultate.

4.2. Umsetzung

- Verifizieren Sie die Projektkonfiguration in `build.gradle` für FXML.
Da für FXML ein zusätzliches JavaFX-Modul (`javafx.fxml`) benötigt wird, muss dieses in der Konfiguration des Plugins hinzugefügt werden.
- Erstellen Sie den SceneGraph mithilfe des SceneBuilder als FXML-Spezifikation, laden diesen und

binden ihn im Hauptfenster ein.

- Als Vorlage können Sie die vorhandene `MainWindow.fxml` im Ressourcen-Ordner und die Hauptanwendungsklasse verwenden.
- c. Erstellen Sie die Controllerklasse und verknüpfen Sie die Controls und Actions mit dem FXML-SceneGraph
 - Als Vorlage finden Sie im Praktikumsverzeichnis bereits eine Klasse für den Controller.
- d. Überlegen Sie sich, wie das vorhandene Datenfeld `result` der Hilfsklasse `ValueHandler` verwendet werden kann, um die View (z.B. Textfeld im `MainWindow`) mittels Observer-Pattern zu aktualisieren, damit `ValueHandler` selbst keine Referenz auf die View oder den Controller benötigt.
 - Welche Hilfsmittel bietet JavaFX dazu an?
 - Passen Sie ihre Anwendung entsprechend an.
- e. Fügen Sie im `MainWindow` einen weiteren Button und zugehörige Aktion (e.g. `openResultWindow`) ein. Diese soll ein zusätzliches einfaches Resultatfenster öffnen.
 - Als Vorlage finden Sie im Praktikumsverzeichnis bereits eine FXML-Datei (`ResultWindow.fxml`) und einen leeren Controller.
 - Erweitern Sie den Controller von `ResultWindow` so, dass auch hier das Resultat angezeigt wird, sobald es in `ValueHandler` neu gerechnet wird.
 - Verwenden Sie in beiden Views das gleiche `ValueHandler` Objekt.

4.3. Hinweise

Die Layout-Einstellungen für die verschiedenen Panes sind grundsätzlich so vorkonfiguriert, dass sich die Grösse der Panes nach dem vorhandenen und nach dem verwendeten Platz richtet. Für die meisten Fälle ist das ok und Sie müssen nur an wenigen Stellen eingreifen. Zum Beispiel um die Mindestbreite festzulegen.



Die Anzeige des Previews stimmt nicht genau mit der echten Anzeige überein. Starten Sie zwischendurch das Programm neu, um die echte Ansicht zu sehen.

Abschluss

Stellen Sie sicher, dass die Pflichtaufgabe mittels `gradle run` gestartet werden kann und pushen Sie die Lösung vor der Deadline in Ihr Abgaberepository.