Lösungen zu den Übungen GUI

1. Erweitern des Workshops um ein Model [PU]

Eine Musterlösung für diese Aufgabe finden Sie im Lösungsverzeichnis unter `FXML-Wordcloud`.

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.

  Bisher wurden die eingegebenen Worte nur im GUI verwendet. Die Klasse `WordModel` soll die eingegebenen Worte halten und etwas damit tun (z.B. zählen der Worte).

- Studieren Sie die zur Verfügung gestellten Klassen und Interfaces. **Sie sollen nicht geändert werden!**
  - `WordModel` enthält keine Hinweise auf das verwendete UI.

    Dies ist eine wichtige Voraussetzung, damit verschiedene GUIs verwendet werden können.

  - `WordModelDecorator` fügt die Möglichkeit zum Beobachten eines `WordModel` Objekts hinzu (Beobachter hinzufügen und entfernen).
  - Überlegen Sie sich, wie hier das Observer-Pattern umgesetzt wurde.

    Die Klasse `WordModelDecorator` implementiert das Interface `IsObervable`. Dies beinhaltet die Methoden `addListener` und `removeListener`. Damit können Objekte vom Typ `IsObserver` hinzugefügt werden. Die Methode `informListener` (der Name spielt keine Rolle) ruft dann auf jedem "Zuhörer" die Methode `update` auf, welche durch das Interface `IsObserver` vorgegeben ist und vom "Zuhörer" implementiert werden muss.

- Ändern Sie den Controller so ab, dass die Vorgaben im Ziel erreicht sind.

  Als erstes muss der Controller das Model kennen. Dies ist gemäss MVC in Ordnung, nur umgekehrt

wäre falsch. Das Model selbst wird in der Hauptapplikation gehalten (`MainWindow`). Die Instanz wird dann an den Controller durchgereicht, nachdem dieser beim Öffnen des Hauptfensters (MainWindow.fxml) erstellt wurde.

Im Controller (`MainWindowController`) verwenden Sie nur den Typ `WordModelDecorator`. Dieser hat die gleichen Methoden wie `WordModel` und einige zusätzliche (siehe oben).

Die Methode `hinzufuegenText` fügt die Worte dem Model hinzu. Das Model macht dann irgend etwas mit den Worten (für das GUI uninteressant) und informiert das GUI über den neuen Zustand.

Beim Instantiieren des `wordModelDecorator` fügen Sie auch gleich den Listener als anonyme Klasse vom Typ `IsObserver` hinzu. Durch das Überschreiben der Methode `update` wird bestimmt, wie der geänderte Inhalt des Models angezeigt wird. Aktuell wird einfach Text angezeigt (in der Lösung). Dies könnte aber auch graphisch umgesetzt werden, dem `wordModel` Objekt ist das egal.

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

Kurzbeschrieb und Anforderungen und Überlegungen zu den JavaFX-Container/Panes), finden Sie in der Aufgabenstellung

3. Erstellen des ROI-Calculators mit Object-SceneGraph

3.1. Planung des Layouts [TU]

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

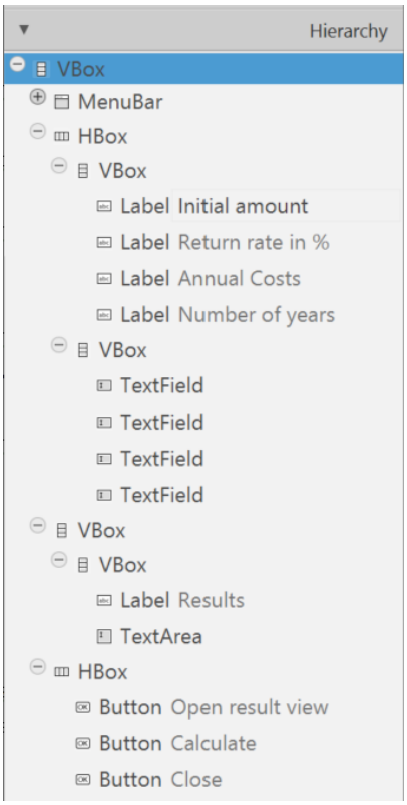
Beachten Sie vor allem die Positionierung, die Abstände und die Ausrichtung der Controls. Vergessen Sie auch das Menü nicht.

Verschiedene Kombinationen sind möglich. Teilen Sie das ganze Layout in Bereiche auf, die vom Verhalten oder der Darstellung her gleich sind. Z.B. könnten die vier Labels mit den Textfeldern des Eingabebereichs so separiert werden. Für diesen Bereich könnten Sie ein GridPane (2 Spalten mit 4 Zeilen) einsetzen. Oder 4 HBox Bereiche angeordnet in einer VBox. Oder 2 VBox Bereiche angeordnet in einer HBox.

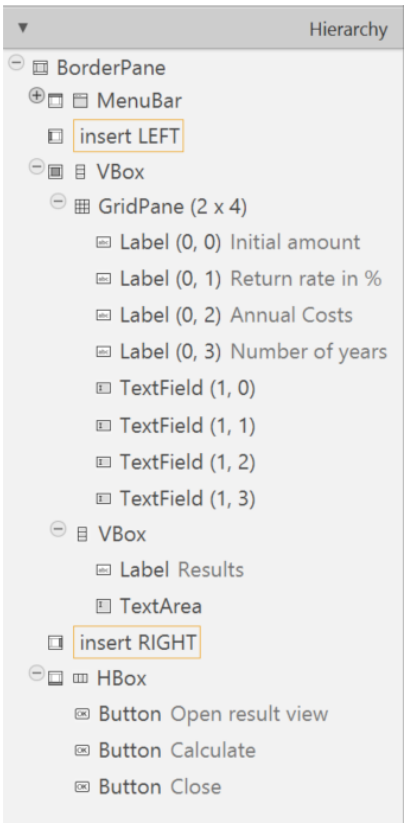
Jeder Ansatz hat Vor- und Nachteile. Bei verschachtelten Containern ist es sinnvoll, eine Einstellung möglichst weit aussen einzustellen.

Gemäss obigem Beispiel: Wenn alle Textfelder sich gleich verhalten sollen, was die Grösse betrifft und den Abstand zwischen den Zeilen, so sollte das in der umgebenden VBox eingestellt werden (Padding für den Abstand zwischen den Textfeldern und die Breite der VBox an die Textfelder vererben). Erst wenn die einzelnen Textfelder ein unterschiedliches Verhalten bei der Grösse aufweisen sollen, nehmen Sie Einstellungen dort vor.

Es gibt unzählige mögliche Kombinationen. Hier 2 davon:



Es wurden zur Organisation der Controls nur VBox und HBox eingesetzt.



Im BoderPane bleiben der Bereich LEFT und der Bereich RIGHT leer. Die Bereiche TOP, CENTER und BOTTOM wurden belegt.
Die leeren Bereiche werden nicht angezeigt und belegen keinen Platz.

Ob diese Layoutkombinationen im Code oder mit dem SceneBuilder erstellt werden, spielt keine Rolle. Die Verschachtelung bleibt die Gleiche.

3.2. Erstellen der Applikation [PU]

Eine Musterlösung für diese Aufgabe finden Sie im Lösungsverzeichnis unter `Calculator`.

a. Ergänzen Sie die Projektkonfiguration (Gradle) für den Einsatz von JavaFX

Die Projektkonfiguration finden sie in der `build.gradle` Datei. Wichtig ist, dass JavaFX ein Plugin benötigt, welche die Konfiguration unterstützt und insbesondere die platformspezifischen (Windows, macOS, Linux) Abhängigkeiten löst. Es muss im Plugin-Teil der Konfiguration integriert werden:

```
plugins {
    // ...
    // Adding JavaFX support and dependencies
    id 'org.openjfx.javafxplugin' version '0.0.13'
}
```

Die Konfiguration des Plugins erfolgt im entsprechenden Block:

```
// Configuration for JavaFX plugin
javafx {
    version = '17.0.6'
    modules = [ 'javafx.controls', 'javafx.fxml' ]
}
```

Zum einen können Sie die spezifische JavaFX-Version angeben, die verwendet werden soll. Diese sollte natürlich zur verwendeten Java-Version kompatibel sein.

Zum anderen können sie spezifizieren, welche JavaFX-Module eingebunden werden sollen. Zum Beispiel wird das `javafx.fxml` Modul nur benötigt, wenn sie auch mit FXML arbeiten, also erst bei Aufgabe 4.

b. Erstellen Sie das Layout für die Applikation

Das Zusammenbauen des Scene-Graph kann aufwändig werden. Wenn Sie das Layout (Aufgabe 1) seriös gemacht haben, ist es jedoch reine Fleissarbeit die Objekte zu erstellen und richtig zusammenzubauen. Sinnvollerweise unterteilen Sie diese Aufgabe in mehrere Methoden, zum Beispiel fürs Menu, das Hauptfenster, ggf. auch Teile des Hauptfensters (Eingabebereich, Resultat, Buttons).

Für immer wieder verwendete Angaben (Abstand zwischen Elementen, Farben, etc.), ergibt es Sinn Konstanten zu definieren oder Werte zu berechnen, damit bei einer Änderung nur an einer Stelle korrigiert werden muss.

c. Fügen Sie die Handler für die benötigten Events hinzu.

In der Musterlösung finden sie verschiedene Varianten von Handlern:

- `ChangeListener` um bei einer Änderung der Fenstergrösse automatisch die Höhe des Resultatfensters anzupassen.
- Action-`EventHandler` welche auf Menu oder Button Betätigung reagieren.
- Mouse-`EventHandler` welche auf Maus-Clicked, Moved, … über bestimmten Elementen reagieren.

Bei allen Event-Handlern sollte die Logik möglichst in separate Methoden ausgelagert werden, welche aus der (anonymen) inneren Klasse (oder einem Lambda-Ausdruck) aufgerufen werden kann. Damit wird der Event-Handler übersichtlichern und sie können die Menge an dupliziertem Code reduzieren.

d. Probieren Sie verschiedene Lösungsansätze und hinterfragen Sie die Vor- und Nachteile der gewählten Lösung.

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

Die Lösungen zu den bewerteten Pflichtaufgaben erhalten Sie nach der Abgabe und Bewertung aller Klassen.