Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

InIT Institut für Angewandte Informations-technologie

Lernaufgabe

V2 – GUI-Architekturen

Lernziel

Sie sind in der Lage, eine einfache GUI-Architektur nach dem MVC und nach dem MVP Patterns zu entwerfen und umzusetzen.

Einleitung

Es soll mit Hilfe des MVC Patterns eine Konsolenanwendung und mit Hilfe des MVP Patterns eine JavaFX-Anwendung erstellt werden.

Für beide Anwendungen steht ein Repository auf GitHub zur Verfügung:
https://github.zhaw.ch/SWEN1-LP2019/SWEN1_V2_Lernaufgabe_GUI-Architekturen_V2.1-V2.2

Aufgabe V2.1: Erstellen Sie eine MVC-Konsolenanwendung.

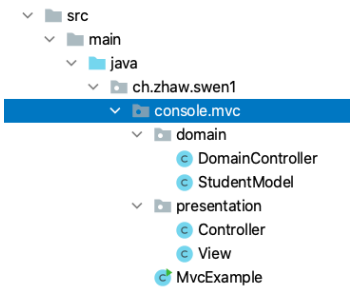
Die Anforderungen and die Anwendung sind:

1. Eingabe eines Namens eines Studierenden.
2. Speicherung des Namens im Domain Model via Domain Controller (nach Larman).
3. Ausgabe der Eingabe auf der Konsole, indem der View vom Domain Model über einen Change-Event informiert wird.
4. Es kann ein neuer Name eingegeben oder mittels Eingabe von 0 das Programm beendet werden.

```
enter new student name, 0 for exit

Felix
Student from Property Change Listener:
Name: Felix
enter new student name, 0 for exit
```

Vorgegebene Struktur:



Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

InIT Institut für Angewandte Informations-
technologie

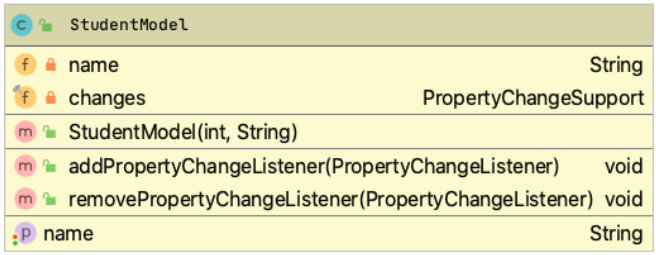
Vorgehen

Implementieren Sie folgende Klassen:

1. Domain Model: Student Model

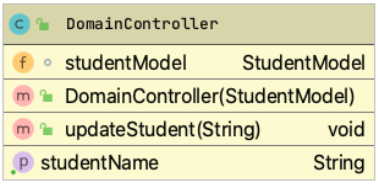
In der Property `name` soll der Name des Studierenden gespeichert werden. Der Zugriff soll über `getter` und `setter` erfolgen. Die `setter-Methode` soll den `PropertyChangeEvent` feuern.

Die Methode `addPropertyChangeListener()` erlaubt die Registrierung eines Listeners.



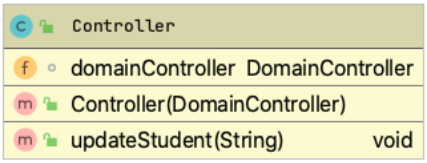
2. Domain Controller:

Der Domain Controller hat 2 Systemoperationen `updateStudent()` und `getStudent()`. Der Domain Controller ruft die `getter` und `setter-Methoden` des `StudentModels` auf.



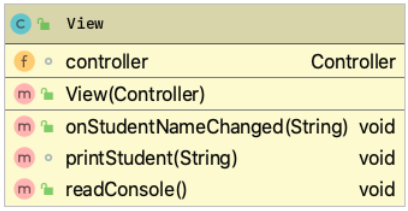
3. Presentation: Controller

Der MVC-Controller empfängt über die Methode `updateStudent()` die Eingabe der Konsole und leitet dies an den Domain Controller weiter.



4. Presentation: View

Die View ruft aus der Methode `readConsole()` die Methode `updateStudent()` im Controller auf. Das Model informiert die View über die Methode `onStudentNameChanged`. Die Methode `printStudent()` gibt die Information auf der Konsole aus.



5. MVC Example

Diese Klasse soll die main()-Methode enthalten und die MVC Anwendung initialisieren. Für die Initialisierung steht eine run()-Methode zur Verfügung. Dazu können Sie ggf. das folgende Code-Snippet verwenden:

```
StudentModel studentModel = new StudentModel(1, "Felix Muster");
DomainController domainController = new DomainController(studentModel);
Controller controller = new Controller(domainController);
View view = new View(controller);
studentModel.addPropertyChangeListener( e -> {
    view.onStudentNameChanged(e.getNewValue().toString());
});

view.readConsole();
```

Ergebnis

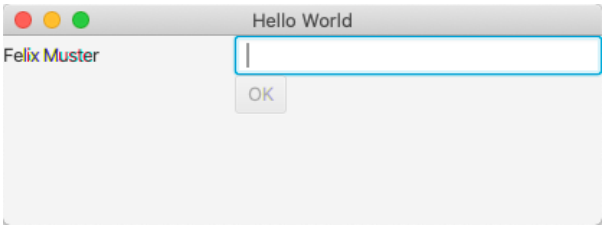
Ein funktionierendes Konsole-Programm nach dem MVC Pattern implementiert.

Zeit: 30’

Aufgabe V2.2: Erstellen Sie eine JavaFX-Anwendung nach dem MVP Pattern mit passive View (fxml).

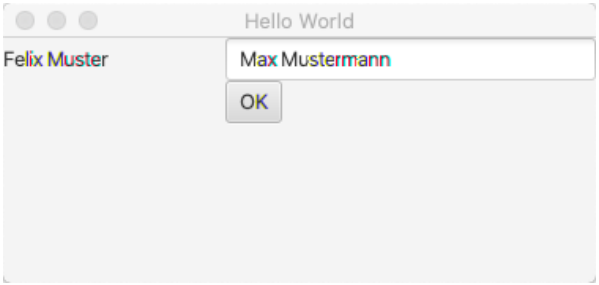
Die Anforderungen an die Anwendung sind:

1. Beim Starten der Anwendung wird links in einem Label der Initialwert aus dem Domain Model angezeigt.



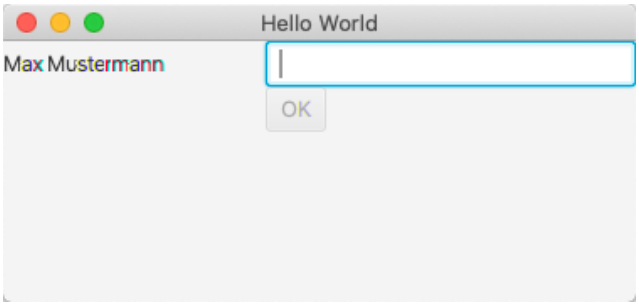
   Sofern kein Eingabewert im Textfeld steht ist der Button OK nicht aktiv.

2. Der Benutzer kann einen neuen Namen eingeben.

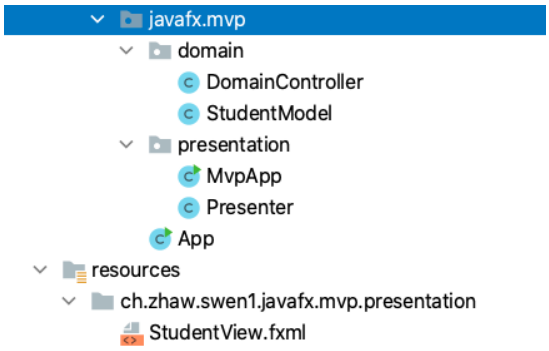


3. Die Eingabe wird mittels [Enter] oder mittels Klick auf [OK] im Domain Model gespeichert.

4. Das Domain Model informiert den Presenter über die Änderung. Der Presenter schreibt die Änderung in das Label links. Das Textfeld wird geleert.



Vorgegebene Struktur:



Vorgehen

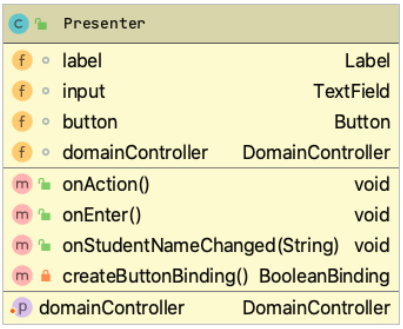
Kopieren Sie die beiden Klassen Domain Controller und Student Model aus der Aufgabe 11.1.

1. View

Die Passive-View mit Name `StudentView.fxml` ist bereits vorgeben. Sie ist mit dem Presenter über das FXML Binding verbunden. Insgesamt gibt es drei Steuerelemente: label, input und button. Über die Methoden `onAction` und `onEnter` informiert die View den Presenter.

2. Presenter

Der Presenter soll eine Methode `init` beinhalten, um den Domain Controller zu initialisieren. Die Methode `onStudentNameChanged` wird vom Model via den `PropertyChangeEvent` aufgerufen. Die Methode `createButtonBinding` kann verwendet werden, um die `disableProperty` des Buttons zu steuern.



3. MvpApp

Die Klasse MvpApp soll die JavaFX Anwendung hochfahren. Dazu wird noch folgendes Code-Snippet benötigt:

```
presenter.init(domainController);
```

Ergebnis

Ein funktionierendes JavaFX-Programm nach dem MVP Pattern implementiert.

Zeit: 45’