**Programmierung II – Übung 5**

## Projekt Adressbuch

Erstellen Sie in *Netbeans* ein neues Projekt mit *File->New Project* im Hauptmenü. Wählen Sie im Fenster *New Project* unter *Categories JavFX* und unter *Projects JavaFX FXML Application* und betätigen Sie *Next*. Im nachfolgenden Fenster können Sie unter *Project Name* den Namen des Projekts angeben. Nennen Sie Ihr Projekt UBG_5. Achten Sie darauf, dass Sie den Pfad unter *Project Location* auf den Ordner setzten, in dem Ihr *Netbeans* Projekt gespeichert werden soll. Tragen Sie unter *FXML name* ein: *adressbuchView* und nennen Sie den Pfad unten unter *Create Application Class: adressbuch.AdressbuchStart*.

Das Projekt beinhaltet drei Dateien:

***AdressbuchStart***: Dies ist eine typische JavaFX Startklasse. Die Methode start wird von der Virtual Machine aufgerufen. Ihr wird das Hauptfenster übergeben (*Stage*) und es wird die in der Datei *adressbuchView.fxml* definierte GUI als *Scene* hineingesetzt und das Fenster wird angezeigt.

***adressbuchView.fxml***: Diese Datei enthält die Beschreibung der GUI. Die Beschreibung liegt in Form einer FXML-Datei vor, einem XML-Format, in dem grafische Benutzeroberflächen beschrieben werden können. Sie können sich die Datei ansehen, indem Sie die Datei im Projekt-Browser anwählen und im Kontextmenü *Edit* wählen. Hier wird offenkundig in einer sog. *AnchorPane* ein *Button* und ein *Label* gesetzt.

***AdressbuchViewController***: enthält die Controller-Klasse für die GUI. In der Klasse gibt es eine Methode *handleButtonAction(..)* die definiert, was gemacht werden soll, wenn der Button in der GUI geklickt wird. Die Methode *initialize*() wird von der Virtual Machine aufgerufen, wenn die GUI initialisiert wird. Hier kann man in der GUI weitere Elemente setzen oder Inhalte definieren.

Die GUI, die Sie hier aufbauen soll verwendet werden, um die Adressbuch-Anwendung zu bedienen. Kopieren Sie die Java-Dateien aus dem Ordner *Sourcen* und pasten Sie diese in das Paket *adressbuch*.

## Aufbau der Anzeige von Adressbucheinträgen

Wir werden zunächst in der Datei a*dressbuchView.fxml* die Inhalte des Hauptfensters definieren. Dies kann man direkt in der FXML-Datei editieren, aber zum Glück gibt es *JavaFX Scene Builder*, einen WYSIWYG-Editor für den Aufbau von grafischen Benutzeroberflächen.

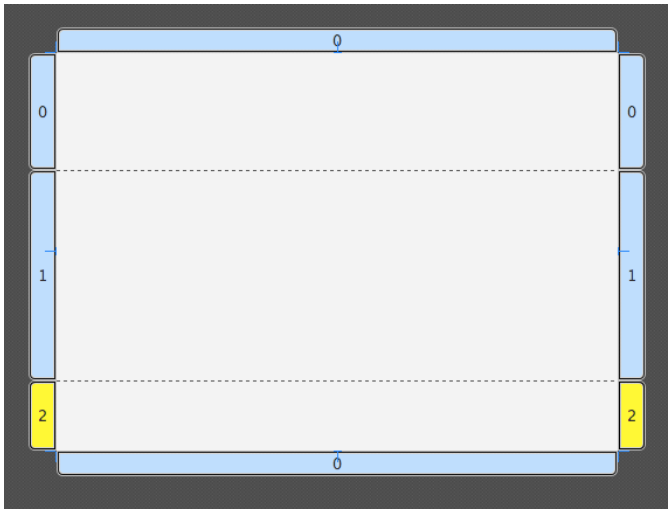
Doppelklicken Sie die Datei a*dressbuchView.fxml*, es öffnet sich der Scene Builder mit der Darstellung der in a*dressbuchView.fxml* definierten Benutzeroberfläche. Links unten im Fenster wird die Dokumenten-Hierarchie angezeigt, d.h. die Hierarchie des

Szenegrafen. Löschen Sie die beiden Elemente unter dem Knoten *AnchorPane* durch Auswahl von *Delete* aus dem Kontextmenü des selektierten Knotens.

Wählen Sie den Knoten *AnchorPane* aus. Ganz rechts im *Scene Builder* lassen sich die Eigenschaften des Elements inspizieren und editieren. Öffnen Sie die Eigenschaften unter Layout und tragen Sie in die Felder *Pref Width* und *Pref Height* die Werte 480 und 340 ein.

Ziehen Sie nun aus dem Containers Menü oben links die *GridPane* auf den Knoten *AnchorPane* in der Hierarchieansicht unten links. Der Knoten wird unter die *GridPane* gehängt und erscheint im Editor. Wählen Sie aus dem Hauptmenü *Modify->Fit To Parent*, um das Element in der Größe an den Elternknoten anzupassen.

Aus der *GridPane(2x3)* soll eine *GridPane* mit drei Zeilen und einer Spalte gemacht werden. Selektieren Sie dazu die Spalte mit der Nummer 0 (wird gelb angezeigt) und wählen Sie im Kontextmenü *Delete*.



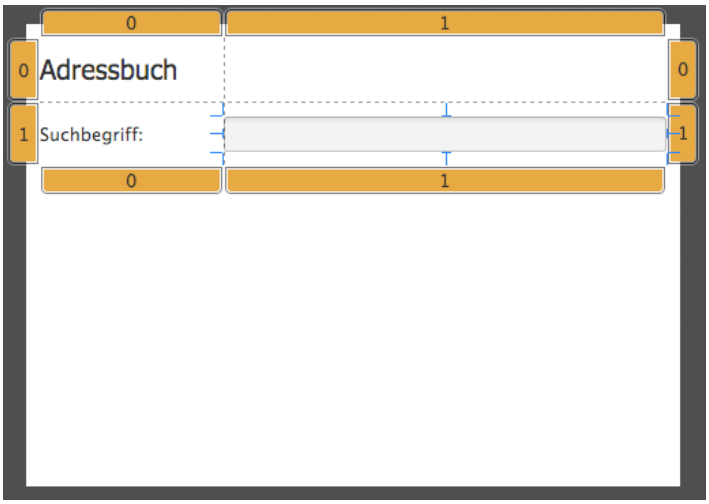

Wählen Sie nun den Zeilenmarker mit der Zahl 0 aus (wird gelb angezeigt) und tragen Sie rechts im Eigenschaftenbrowser als *Pref Height* den Wert 80 ein.

Tragen Sie für die zweite Zeile den *Pref Height* Wert 160 und für die dritte Zeile den Wert 40 ein.

Ihre GUI sollte jetzt ungefähr so aussehen.

Um einen kleinen Rand um Ihre GUI-Inhalte zu behalten, stellen Sie für die GridPane im Eigenschaftenbrowser unter Padding in den vier Textfeldern dern Wert 10 ein.

In die oberste Zeile des *GridPane* sollte eine weitere *GridPane* eingesetzt werden. Ziehen Sie aus *Containers* eine *GridPane* auf den Knoten *GridPane* in der Hierarchie. Löschen Sie hier die unterste Zeile und stellen Sie bei Spalte 0 die *Pref Width auf 120 und bei Spalte 1 auf 310.*



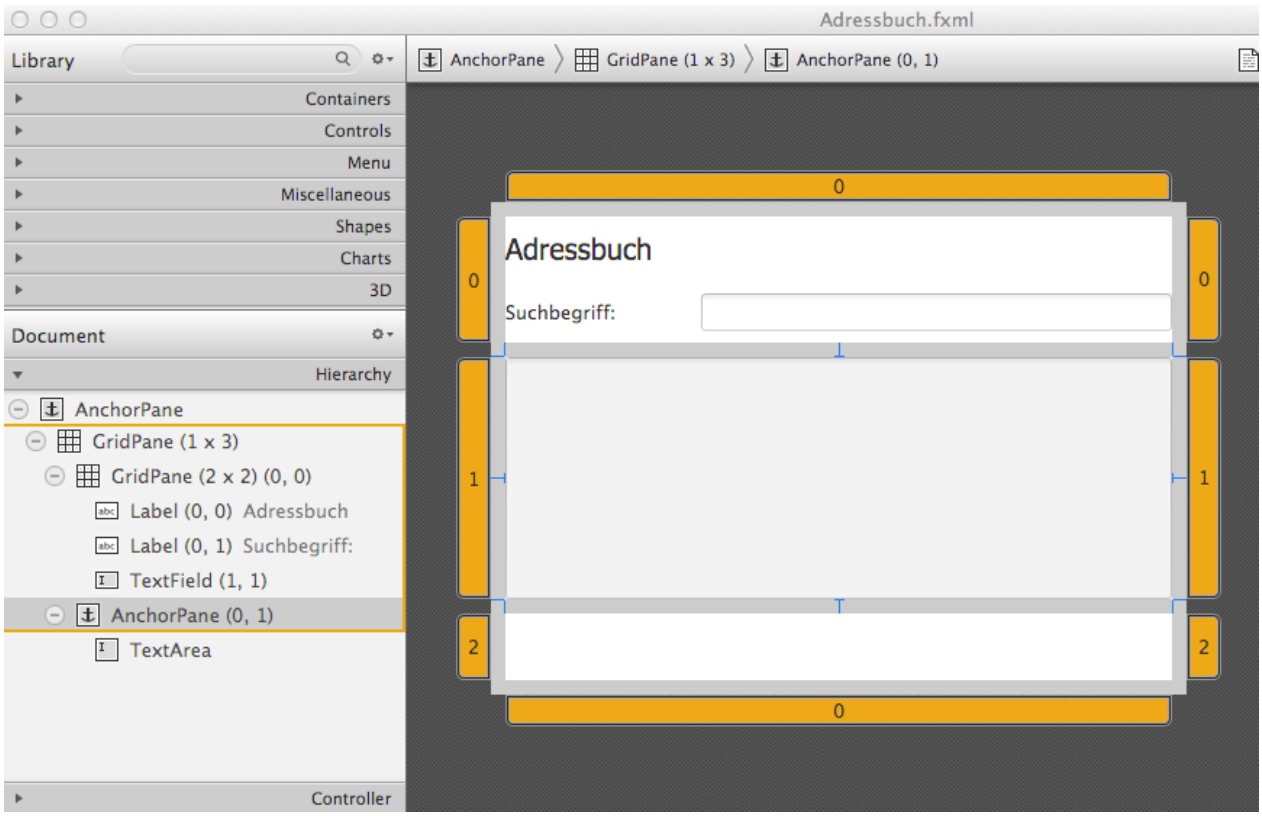

Ziehen Sie nun aus dem Menü *Controls* je ein Label in die Zelle (0,0) und (0, 1). Wählen Sie das oberste Label aus und ändern Sie den Text im Eigenschaftenbrowser unter

Properties zu *Adressbuch*. Stellen Sie als Font *Tahoma 20px* ein. Ändern Sie den Text des zweiten Labels auf *Suchbegriff:* und stellen Sie als Font *Tahoma 13px* ein.

Ziehen Sie aus dem Menü *Controls* nun ein *TextField* direkt auf die Zelle (1,1) im Editor. Dieses sollte automatisch so eingepasst werden, dass es die Zelle in der Breite einnimmt.

Stellen Sie bei dem Knoten *GridPane(1x3)* unter *Layout VGap* auf 10. Dadurch werden zwischen den Bereichen 10 Pixel Abstand eingehalten.

Ziehen Sie jetzt aus dem Menü *Controls* eine *TextArea* in die mittlere Zeile des *GridPane (1x3)* im Editor. Ihre GUI sollte nun so aussehen:




**Koppeln der Anzeige mit dem Controller**

In das Textfeld soll der Nutzer einen Suchbegriff eingeben (Teil eines Namens oder einer Telefonnummer), in der *TextArea* sollen daraufhin die Einträge aus dem Adressbuch angezeigt werden, die auf den Suchbegriff passen. D.h. aus dem Controller muss man sowohl auf das Textfeld, als auch auf die *TextArea* zugreifen können.

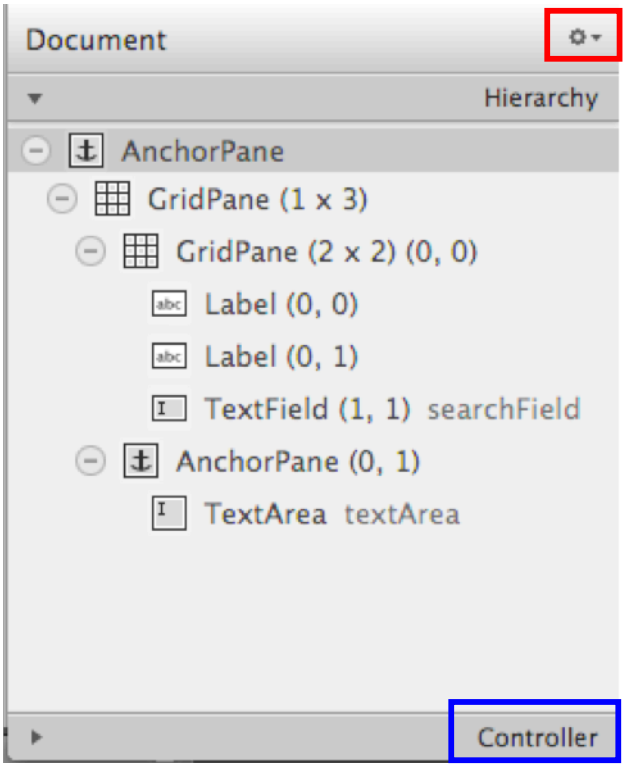
Um die einzelnen interaktiven bzw. veränderlichen GUI-Elemente mit dem Controller zu koppeln, müssen diese zunächst mit sog. *fx:ids* versehen werden. Durch *fx:ids* bekommen die GUI-Elemente, auf die man im Controller zugreifen möchte, Bezeichner

die sich auf Attribute des Controllers beziehen. Den Attributen des Controllers werden dann hinter den Kulissen beim Start des Programms die GUI-Objekte injiziert.

Ein GUI-Element kann im Eigenschaften-Browser auf der rechten Seite im Bereich *Code* mit einer *fx:id* versehen werden. Wählen Sie den Knoten *TextField* in der Hierarchieansicht aus (Sie müssen dazu unter Umständen ein paar Knoten aufklappen) und klappen Sie im Eigenschaften-Browser den Bereich *Code* auf. Tragen Sie in das Textfeld *fx:id searchField* ein.

Geben Sie der *TextArea* die *fx:id textArea*.

In der Hierarchieansicht kann man sich die *fx:ids* der einzelnen Knoten anzeigen lassen. Wählen Sie dazu auf der linken Seite in der Zeile *Document* im Menü mit dem Zahnrad (siehe rote Umrandung) *Hierarchy Displays->fx:id*.




Um das GUI-Element mit dem zugehörigen Controller zu koppeln, müssen Sie diesen zuweisen. Klappen Sie unten links den Bereich Controller auf (siehe blaue Umrandung). Im Textfeld Controller Class sollte der *AdressbuchViewController* eingetragen sein. Falls dies nicht so sein sollte, können Sie im Menü neben dem Textfeld den richtigen Controllernamen auswählen.

Speichern Sie Ihre GUI mit *File->Save* und schließen Sie den *Scene Builder*.

Als nächstes müssen sie dafür sorgen, dass in der zugehörigen Controller-Klasse die entsprechend annotierten Attribute mit dem Namen der *fx:ids* angelegt werden. Das lässt sich mit *Netbeans* leicht erledigen. Wählen Sie im Project-Browser die .fxml-Datei aus und wählen Sie im Kontextmenü *Make Controller*. Öffnen Sie die Klasse *AdressbuchViewController* im Editor. Wie Sie sehen, hat *Netbeans* für jede *fx:id* der GUI ein Attribut vom Typ des entsprechenden GUI-Elements angelegt und dieses mit der Annotation *@FXML* versehen. Die Annotation sorgt dafür, dass den Attributen beim Start der Anwendung automatisch die GUI-Objekte injiziert werden.

Die Klasse enthält noch das Attribut *label* und die Methode *handleButtonAction(..)*, löschen Sie beide.

Der *AdressbuchViewController* soll verwendet werden, um ein Objekt vom Typ Adressbuch zu bedienen. Fügen Sie der Klasse daher ein normales privates Attribut *adressbuch* vom Typ *Adressbuch* hinzu.

Der Controller wird von der JavaFX-Anwendung initialisiert, wenn die Anwendung gestartet wird. Dafür wird von der Virtual Machine die Methode *initialize()* aufgerufen.

Sorgen Sie dafür, dass das Attribut *adressbuch* als erstes in der Methode *initialize()* eine neue Instanz der Klasse *Adressbuch* zugewiesen bekommt.

### Einstellen der *TextArea*

Als nächstes muss die *TextArea* eingestellt werden.

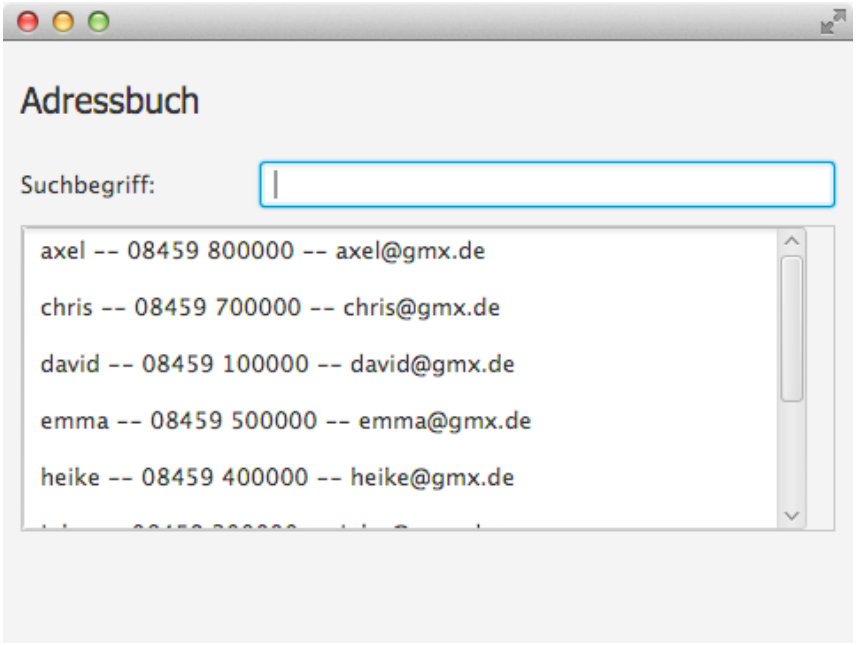
Die *TextArea* soll am Anfang eine Liste aller Kontakte anzeigen, sobald der Nutzer in das Textfeld einen Suchbegriff einträgt, soll das Ergebnis einer Suchanfrage in der TextArea angezeigt werden. Das Anzeigen der zu dem Suchbegriff eingegebenen Kontakte soll in eine Methode *showKontakte(..)* ausgelagert werden. Fügen Sie ihrer Klasse daher die folgende Methode hinzu:

```
private void showKontakte(Kontakt[] kontakte)
```

Diese soll die *TextArea* zunächst löschen und dann alle Kontakte des Arrays in der TextArea anzeigen. Benutzen Sie dafür die Methode *appendText(..)* der Klasse *TextArea*.

Sorgen Sie dafür, dass Methode in der Methode *initialize* aufgerufen wird, um nach dem Start des Programms alle Kontakte des Adressbuchs in der *TextArea* anzuzeigen.

Probieren Sie es aus, indem Sie *AdressbuchStart* auswählen und im Kontextmenü *Run File* aufrufen. Jetzt müssten alle Kontakte aus dem Adressbuch in der *TextArea* angezeigt werden.



**EventHandling für das Textfeld**

Abschließend muss noch das Textfeld interaktiv gemacht werden. Es soll ein Event auslösen, sobald der Nutzer darin editiert hat. D.h. sobald ein neuer Buchstabe eingegeben wird oder einer gelöscht wird, soll die Suche neu angestoßen werden und alle Kontakte, die auf den Suchbegriff passen, werden in der Tabelle angezeigt.

Schreiben Sie zunächst eine private Methode *filterList()*, die aus dem Textfeld *searchField* den Text ausliest, beim Adressbuch alle Kontakte zu diesem Präfix anfragt und das resultierende Array in der *TextArea* anzeigt. Nutzen Sie hierfür möglichst bereits vorhandene Methoden.

Melden Sie in der Methode *initialize()* die Methode *filterList()* als EventHandler an. Schauen Sie im Skript nach wie man einen EventHandler für das Editieren des Textfeldes anmelden kann.

Probieren Sie aus, ob es funktioniert. Die *TextArea* müsste nun aktualisiert werden, sobald eine Interaktion im Textfeld stattfindet.