

Java 2: Übung AdressDB

Aufgabe 1

Anzeige aller Daten der Tabelle person in einer TableView

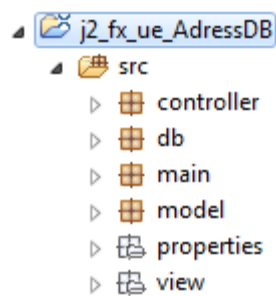


ID	Vorname	Nachname	PLZ	Ort	Straße	Telefon	Mobil	E-Mail
25	Hans	Schuster	13456	Berlin	Waldweg 5	030-4354353	0177-5435...	schuste@web.de
26	Ina	Schulz	33555	Hamburg	Teststraße 3	033-4455354	0153-2332...	schulz@ee.de
24	Anton	Meier	23434	Dresdene	Parkstraße 4	0224-345435	0163-3665...	werwer@se.de
27	Max	Muller	32444	Bonn	Dorfstraße 34	022-5435435	0133-4353...	134@gmx.de
28	Max	Müller	34324	Berlin	Hauptstraße 57	030-346566	0166-5464...	test@rrr.com
29	Anna	Müller	13445	Erkner	Platz 77	030-5345435	0177-4545...	aaa@a.de
30	Wasser	Werner	31333	München	Am Berg 6	033-8658678	0165-3543...	abx@we.de

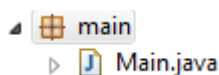
Erste Version der Adress-Tabelle

1a)

Erstelle ein neues JavaFX Projekt **j2_fx_ue_AdressDB** mit der folgenden Paketstruktur:



Das Paket main enthält die Klasse Main:





```

public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            BorderPane root = (BorderPane)FXMLLoader.load(getClass().getResource("/view/Person.fxml"));
            Scene scene = new Scene(root);
            scene.getStylesheets().add(getClass().getResource("/view/application.css").toExternalForm());
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        launch(args);
    }
}

```

1b)

 model
 Person.java





Schreibe eine Klasse **Person** als JavaFX-Bean (Properties) mit allen Attributen der Tabelle person:

id,vorname,nachname,plz, ort,strasse, telefon, mobil, email



Schreibe geeignete Konstruktoren.

1c)

Schreibe die Datenbankklassen:

 db
 DBConnect.java
 DBProp.java
 PersonDAO.java

Die Klasse DBConnect soll die Datenbankverbindung aufbauen und dabei die DBProp und eine Propertie-Datei verwenden:

 properties
 db.properties

Die Klasse PersonDAO soll die DBConnect „benutzen“ und zunächst nur die Methode **findAllPersons()** implementieren:

```

public List<Person> findAllPersons(){...}

```

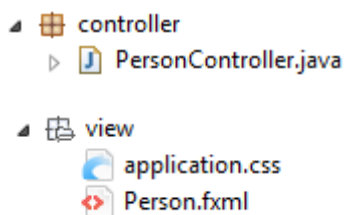
Diese Methode liefert eine Liste mit Person-Objekten und soll später für die TableView die Daten liefern.

1d)

Schreibe einen JUnit-Test für die Methode findAllPersons().

1e)

Umsetzung der TableView mit FXML und Controller:



Erzeuge eine *TableView* mit dem *SceneBuilder* in der *Person.fxml*.

Lade im *PersonController* die Tabellendaten, so das nach dem Start der Applikation die Tabelle angezeigt wird:

ID	Vorname	Nachname	PLZ	Ort	Straße	Telefon	Mobil	E-Mail
25	Hans	Schuster	13456	Berlin	Waldweg 5	030-4354353	0177-5435...	schuste@web.de
26	Ina	Schulz	33555	Hamburg	Teststraße 3	033-4455354	0153-2332...	schulz@ee.de
24	Anton	Meier	23434	Dresdene	Parkstraße 4	0224-345435	0163-3665...	werwer@se.de
27	Max	Muller	32444	Bonn	Dorfstraße 34	022-5435435	0133-4353...	134@gmx.de
28	Max	Müller	34324	Berlin	Hauptstraße 57	030-346566	0166-5464...	test@rrr.com
29	Anna	Müller	13445	Erkner	Platz 77	030-5345435	0177-4545...	aaa@a.de
30	Wasser	Werner	31333	München	Am Berg 6	033-8658678	0165-3543...	abx@we.de

Aufgabe 2

Neuen Datensatz einfügen

2a)

Die Programm soll jetzt um eine *TabView* ergänzt werden. Verwende dazu `TableView`→ „Wrap in“ des SceneBuilders.

[illegible]

Lege im zweiten Tab „Neuer Datensatz“ das folgende Formular zu, um neue Datensätze einzufügen. Gebe den Textfeldern auch eine `fx:id`.

The screenshot shows a JavaFX window with two tabs: 'Übersicht' and 'Neuer Datensatz'. The 'Neuer Datensatz' tab is active and contains a form with the following text fields, each with a label and a `fx:id` attribute: `Vorname`, `Nachname`, `PLZ`, `Ort`, `Straße/Nr`, `Telefon`, `Mobil`, and `E-Mail`. A 'Speichern' button is located at the bottom right of the form.

2b)

Schreibe in der Klasse *PersonDAO* eine neue Methode, um neue Datensätze einzufügen zu können:

```
public boolean savePerson(Person person){  
}
```

Verwende `PreparedStatement`!

Die Methode soll `true` zurückgeben, wenn das Insert erfolgreich war.

Die Klasse `PreparedStatement` besitzt eine Methode `getUpdateCount()`, die die Anzahl der eingefügten Datensätze zurückgibt.

Diese Methode soll im *PersonController* aufgerufen werden:

```
@FXML  
public void save(ActionEvent e){  
    boolean ok = dao.savePerson(new Person(...));  
    ...  
}
```

Das Person-Objekt muss natürlich die neuen Daten bekommen.

Durch das Drücken des Speichern-Buttons wird die Methode `save()` aufgerufen.

Wenn der Datensatz erfolgreich gespeichert wurde, soll im oberen Teil der Applikation eine Meldung erscheinen. Verwende dazu ein Label (`infoLabel`)

The screenshot shows a web application window titled "Datensatz X". At the top, a message "Datensatz gespeichert!" is displayed in a red-bordered box. Below the title bar, there are eight text input fields arranged vertically, each containing a specific value. At the bottom right of the form, there is a button labeled "Speichern".

Field Index	Value
1	Ina
2	Meier
3	34243
4	Berlin
5	Dorfstraße 2
6	32423434
7	0166-2342342
8	asd@werwe.de

Füge in den Oberen Teil der Applikation einen Refresh-Button ein:



ID	Vorname	Nachname	PLZ	Ort	Straße	Telefon	Mobil	E-Mail
25	Hans	Schuster	13456	Berlin	Waldweg 5	030-4354353	0177-5435...	schuste@web.de
26	Ina	Schulz	33555	Hamburg	Teststraße 3	033-4455354	0153-2332...	schulz@ee.de
24	Anton	Meier	23434	Dresdene	Parkstraße 4	0224-345435	0163-3665...	werwer@se.de
27	Max	Muller	32444	Bonn	Dorfstraße 34	022-5435435	0133-4353...	134@gmx.de
28	Max	Müller	34324	Berlin	Hauptstraße 57	030-346566	0166-5464...	test@rrr.com

Wenn Refresh-Button gedrückt wird, soll die Tabelle mit den neuen aktuellen Daten (nach hinzufügen eines neuen Datensatzes) gezeigt werden.

Aufgabe 3

Datensatz löschen

3a)

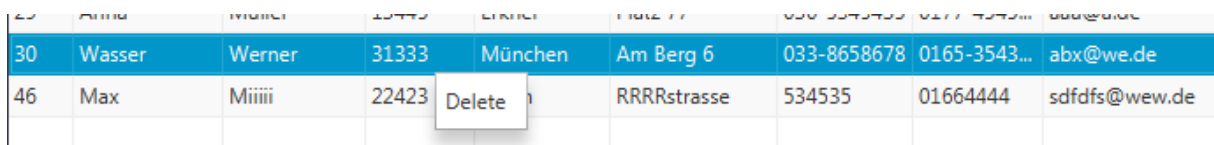
Schreibe in der Klasse PersonDAO eine Methode zum löschen eines Datensatzes:

```
public boolean deletePerson(int id){  
    ...  
}
```

Auch hier soll true zurückgegeben werden, wenn das löschen erfolgreich war.

3b)

Diese Methode soll aus dem Controller mit Hilfe eines Kontextmenüs aufgerufen werden:



30	Wasser	Werner	31333	München	Am Berg 6	033-8658678	0165-3543...	abx@we.de
46	Max	Miiiiii	22423	Delete	RRRRstrasse	534535	01664444	sdfdfs@wew.de

Wenn das Löschen erfolgreich war, soll der Eintrag aus der ObservableList ebenfalls gelöscht werden, so dass die TableView aktualisiert wird.

Aufgabe 4

Update

4a)

Schalte die Spalten auf editierbar.

Nur die Felder Telefon, Mobil und E-Mail sollen editierbar und updatebar sein!

Um Spalten editieren zu können, muss die setCellFactory-Methode aufgerufen werden:

```
emailColumn.setCellFactory(TextFieldTableCell.forTableColumn());
```

Außerdem muss die TableView auf editable="true" eingestellt werden:

```
<TableView fx:id="table" editable="true" ...
```

Straße	Telefon	Mobil	E-Mail
/aldweg 5	4354353	0177-5435...	schuste@web.de
eststraße 3	<input type="text" value="7777"/>	01882332...	schulz@ee.de
erkstraße 4	0224-345400	055-36656	wenner@ee.de

4b)

Schreibe in der DAO-Klasse die Methode update():

```
public boolean updatePerson(int id, String fieldName, String newValue) {
```

```
...
```

Diese Methode soll ebenfalls true zurückgeben, wenn das Update erfolgreich war.

Die Methode soll in der PersonController-Klasse von der Methode editCommit aufgerufen werden:

```
@FXML
public void editCommit(CellEditEvent<Person, String> c){

    String newValue = c.getNewValue();

    boolean updateOk =dao.updatePerson(p.getId(), fieldName, newValue);
```

Wenn das Update erfolgreich war, soll im oberen Info-Feld wieder eine Meldung erscheinen.

In der FXML muss die Methode als Eventhandler angegeben werden:

```
<TableColumn fx:id="telefonColumn" prefWidth="77.0" text="Telefon"
onEditCommit="#editCommit" >
```


Hinweis:

Die Überschriften der Spalten in der TableView unterscheiden sich von denen der Datenbank-Tabelle (E-Mail→email).

Damit die update-Methode funktioniert, müssen diese Spaltennamen „gemapt“ werden.