Graphical User Interfac 2

1. Vorbereitung VSCode

Für die Verwendung von JavaFX nutzen wir in diesem Praktikum das Build-Management-Tool Gradle. Bis anhin haben Sie bei VSCode durch den Projektordner die interne Build-Prozesse verwendet. Somit konnten Sie den *lib*-Ordner nutzen und ihr Projekt ohne grosse Angaben starten. Dies ist für kleinere Projekte praktisch, jedoch bei grösseren Projekten sollte man auf durchdachte Build-Management-Tool zurückgreifen. Zudem ist die Einrichtung von JavaFX unter Gradle simpler.

Beachten Sie, dass Sie für JavaFX die **Java-Version 11** verwenden.

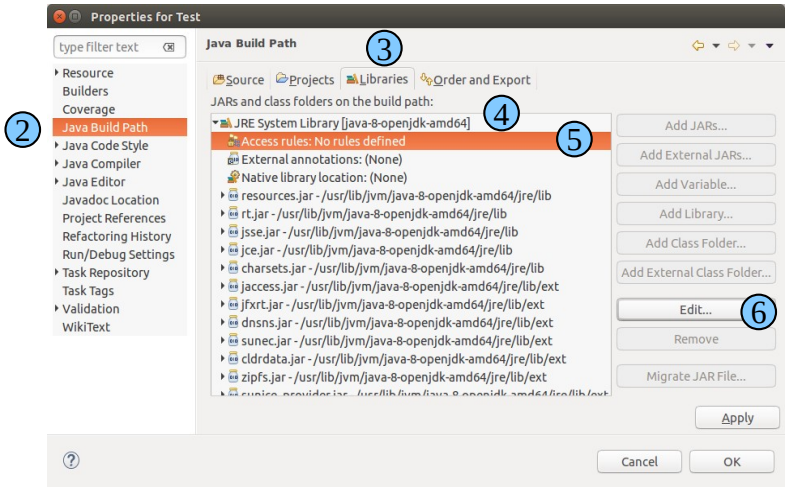
Im Dokument *Visual Studio Code für den Unterricht* finden Sie unter *JavaFX mit Gradle* eine Beschreibung zur Einrichtung des Projektes. Achten Sie, dass Sie anstelle der Installation von Gradle (Punkt 1), folgende Punkte beachten

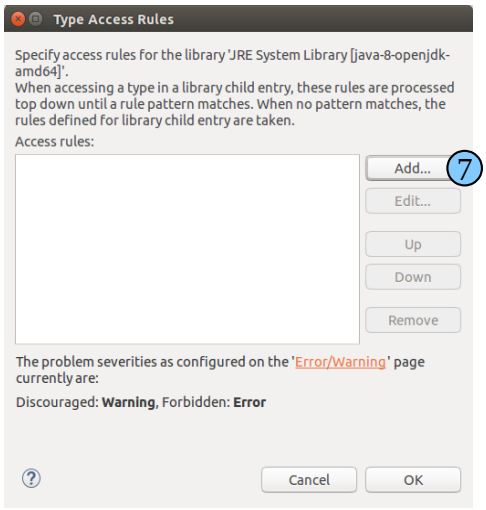
- Laden Sie sich die *gradle_project.zip* Datei aus dem Praktikumsverzeichnis herunter.
- In der Konsole von VSCode geben Sie für die Erstellung des Projektes folgender Befehl ein:
  - Linux / Mac: **`./gradlew init`**
  - Windows: **`gradlew.bat init`**
    Beachten Sie bei Windows, dass als Konsolenapplikation cmd ausgewählt ist (Dorpdown-Menü im VSCode-Terminal-Bereich)

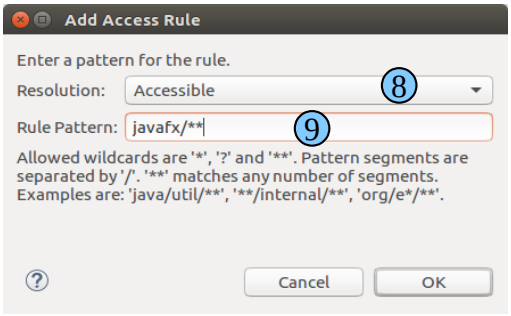
2. Vorbereitung Eclipse

Wenn Sie mit Eclipse arbeiten, werden Sie bei der Benutzung von JavaFX einen Fehler erhalten, da eine Zugriffsberechtigung für diese Programm-Bibiliothek fehlt. Um diese Berechtigung hinzuzufügen gehen Sie wie folgt vor:

1. Gehen Sie zu den Einstellungen des Projektes
   1. Rechtsklick auf den Projektordner im Package-Explorer
   2. *Properties* auswählen.
2. Wählen Sie im neuen Fenster *Java Build Path* aus.
3. Wählen Sie bei den Reitern den Punkt *Libraries* aus.
4. Klappen Sie den Punkt *JRE System Library* aus.
5. Wählen Sie den Punkt *Access rules* aus.
6. Drücken Sie auf den Knopf *Edit*.
7. Drücken Sie im neuen Fenster auf den Knopf *Add*.
8. Im neuen Fenster ändern Sie den Punkt *Resolution* auf *Accessible*.
9. Im Textfeld *Rule Pattern* geben Sie folgende Zeile ein: *javafx/***
10. Bestätigen Sie alle Fenster mit dem Knopf *OK*.

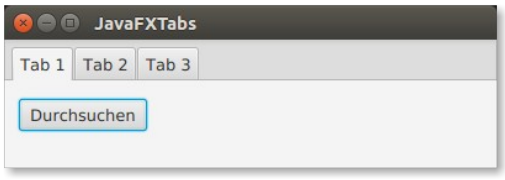






3. JavaFX Tabs

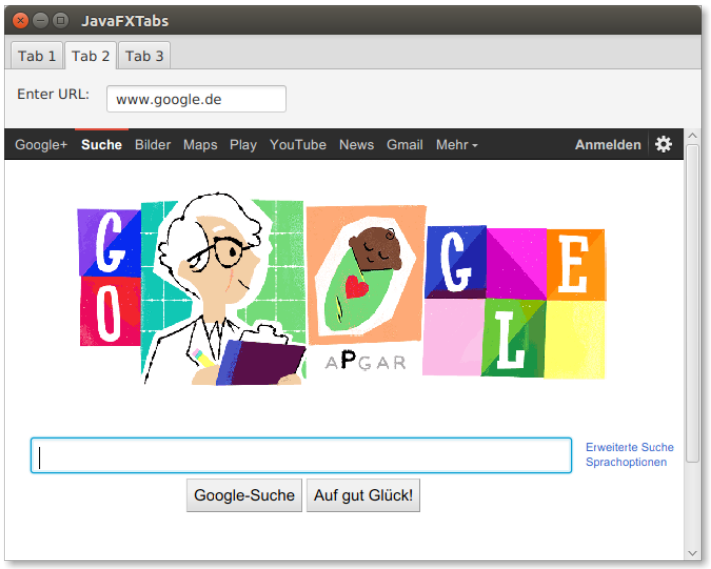
1. Erstellen Sie eine Applikation, die verschiedene Bedienelemente in Tabs präsentiert. Unter anderem einen *Button* zum Öffnen eines Dateiauswahl-Dialogs, ein Tab mit einem *Label* und *TextField* und einen noch leeren Tab. Sie sollten dann 3 Tabs haben:



   Nutzen Sie als Einstiegspunkt folgende Seite:
   http://www.java2s.com/Code/Java/JavaFX/AddTabtoTabPane.htm

2. Wenn der Benutzer im Dateiauswahl-Dialog ein Bild auswählt, soll dieses im Button-Tab dargestellt werden. Nutzen Sie dazu die Klasse *ImageView*:
   https://docs.oracle.com/javase/8/javafx/api/javafx/scene/image/ImageView.html
   Erweitern Sie Ihr Programm, so, dass im Dateiauswahl-Dialog nur Bilddateien ausgewählt werden können (*.png*, *.jpg*, *.bmp*).

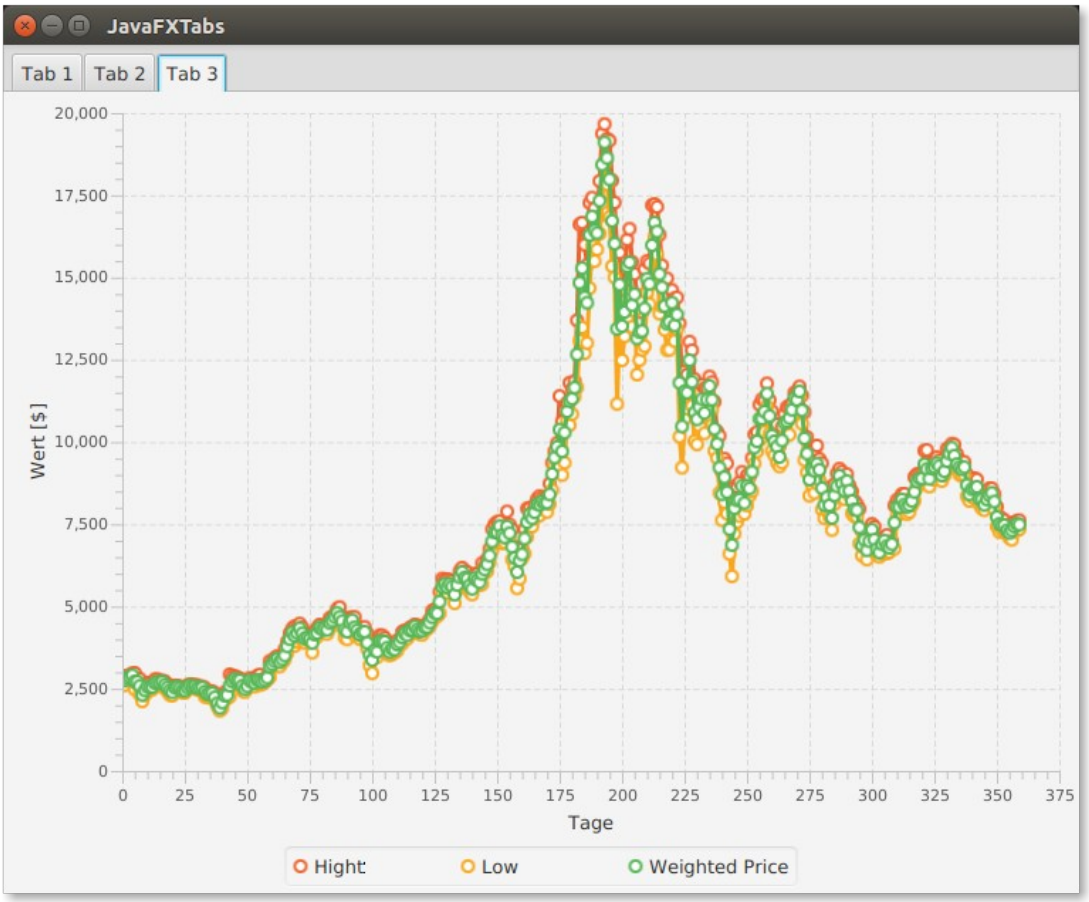
3. Versuchen Sie in einem weiteren Schritt eine Website zu öffnen, sobald man im Textfeld (Textfeld-Tab) eine Internet-Adresse eingegeben hat und auf Enter drückt. Dabei soll die Website unter dem Textfeld dargestellt werden. Lesen Sie sich dazu in die folgende Seite ein: https://docs.oracle.com/javafx/2/webview/jfxpub-webview.htm



4. Erstellen Sie im leeren Tab einen *LineChart*. Lesen Sie sich dazu auf folgende Seite ein: https://www.tutorialspoint.com/javafx/line_chart.htm
   Im Projektordner befindet sich die Datei *bitcoin.csv*. In dieser Datei ist der Bitcoin-Verlauf über dem Zeitraum eines Jahres aufgeführt. Die Daten sind im CSV-Muster (UTF8-Format) dargestellt und mit *;* voneinander getrennt. Jede Zeile entspricht einen Datenpunkt.

   Tipp: Öffnen Sie die csv-Datei in einem Texteditor (z.B.: notepad) und betrachten Sie die Struktur.

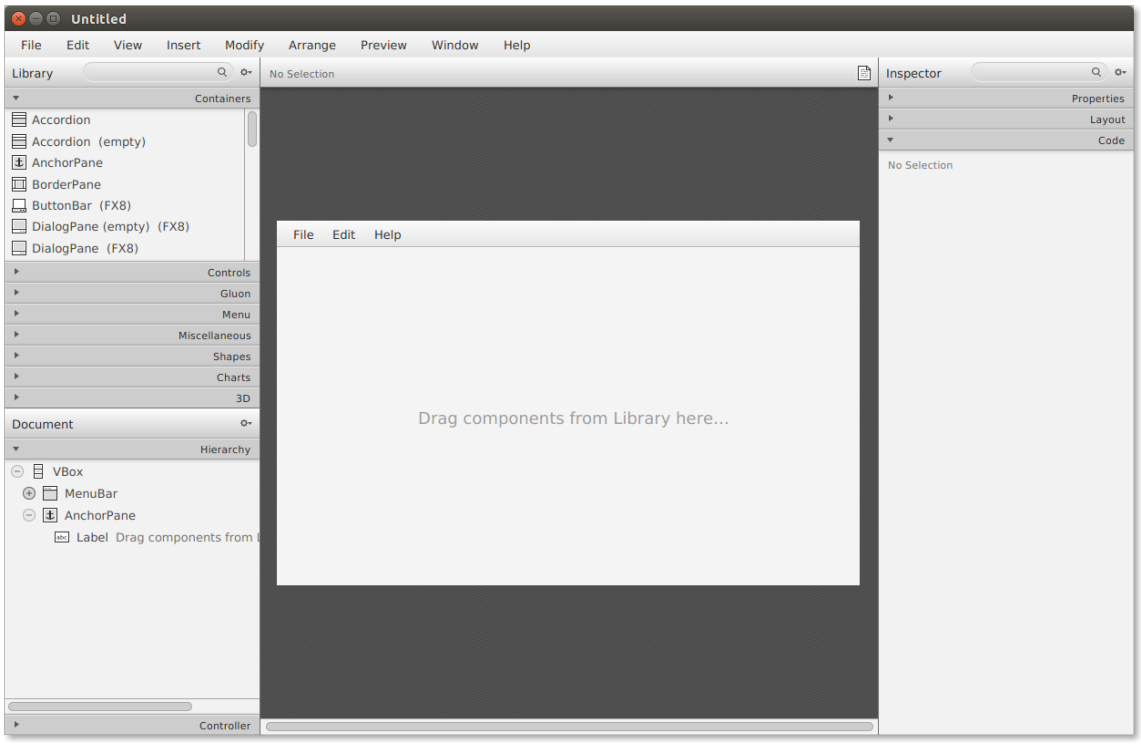
   Versuchen Sie die Werte *Hight*, *Low* und *Weighted Price* im Graph jeweils als Linie darzustellen. Die Werte sind in US-Dollar angegeben.



4. Der grosse Filter (SceneBuider)

Tool

Der JavaFX Scene Builder ist ein Design-Tool für die JavaFX Plattform. Der Scene Builder ermöglicht es durch einfaches "drag-and-drop" GUI-Komponenten zu einer Benutzeroberfläche zusammen zu bauen. Während man mit den grafischen Elementen die Benutzeroberfläche erstellt, wird im Hintergrund automatisch eine FXML-Datei generiert.



Die aktuellste Vesion des Scene-Builders kann unter folgendem Link bezogen werden:

http://gluonhq.com/products/scene-builder/

Beachten Sie dabei, dass Sie die Version aus der Kategorie *Download Scene Builder for Java 8* beziehen.

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.SeparatorMenuItem?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<VBox prefHeight="400.0" prefWidth="640.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1">
  <children>
    <MenuBar VBox.vgrow="NEVER">
      <menus>
        <Menu mnemonicParsing="false" text="File">
          <items>
            <MenuItem mnemonicParsing="false" text="New" />
            <MenuItem mnemonicParsing="false" text="Open…" />
            <Menu mnemonicParsing="false" text="Open Recent" />
            <SeparatorMenuItem mnemonicParsing="false" />
            <MenuItem mnemonicParsing="false" text="Close" />
            <MenuItem mnemonicParsing="false" text="Save" />
            <MenuItem mnemonicParsing="false" text="Save As…" />
            <MenuItem mnemonicParsing="false" text="Revert" />
            <SeparatorMenuItem mnemonicParsing="false" />
            <MenuItem mnemonicParsing="false" text="Preferences…" />
            <SeparatorMenuItem mnemonicParsing="false" />
            <MenuItem mnemonicParsing="false" text="Quit" />
          </items>
        </Menu>
        <Menu mnemonicParsing="false" text="Edit">
          <items>
            <MenuItem mnemonicParsing="false" text="Undo" />
            <MenuItem mnemonicParsing="false" text="Redo" />
            <SeparatorMenuItem mnemonicParsing="false" />
            <MenuItem mnemonicParsing="false" text="Cut" />
            <MenuItem mnemonicParsing="false" text="Copy" />
            <MenuItem mnemonicParsing="false" text="Paste" />
            <MenuItem mnemonicParsing="false" text="Delete" />
            <SeparatorMenuItem mnemonicParsing="false" />
            <MenuItem mnemonicParsing="false" text="Select All" />
            <MenuItem mnemonicParsing="false" text="Unselect All" />
          </items>
        </Menu>
        <Menu mnemonicParsing="false" text="Help">
          <items>
            <MenuItem mnemonicParsing="false" text="About MyHelloApp" />
          </items>
        </Menu>
      </menus>
    </MenuBar>
    <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0" prefWidth="-1.0" VBox.vgrow="ALWAYS">
      <children>
        <Label alignment="CENTER" contentDisplay="CENTER" layoutX="155.0" layoutY="177.0" style="&#10;" text="Drag components from Library here…" textAlignment="CENTER"
textFill="#9f9f9f" wrapText="false">
          <font>
            <Font size="18.0" />
          </font>
        </Label>
      </children>
    </AnchorPane>
  </children>
</VBox>
```

FXML

Sobald Sie ein Projekt aus dem Scene-Builder abspeichern, liegt Ihnen der Aufbau als FXML-Datei zur Verfügung. FXML ist ein Datei-Format welcher von der JavaFX-Bibliothek direkt eingelesen und zur Darstellung des GUIs verwendet werden kann. Dieses Format basiert auf der *Extensible Markup Language* (XML).

```xml
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.SeparatorMenuItem?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<VBox prefHeight="400.0" prefWidth="640.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1">
  <children>
    <MenuBar VBox.vgrow="NEVER">
      <menus>
        <Menu mnemonicParsing="false" text="File">
          <items>
            <MenuItem mnemonicParsing="false" text="New" />
            <MenuItem mnemonicParsing="false" text="Open…" />
            <Menu mnemonicParsing="false" text="Open Recent" />
            <SeparatorMenuItem mnemonicParsing="false" />
            <MenuItem mnemonicParsing="false" text="Close" />
            <MenuItem mnemonicParsing="false" text="Save" />
            <MenuItem mnemonicParsing="false" text="Save As…" />
```

Die FXML-Datei kann wie folgt in einem JavaFX-Projekt eingebaut werden:

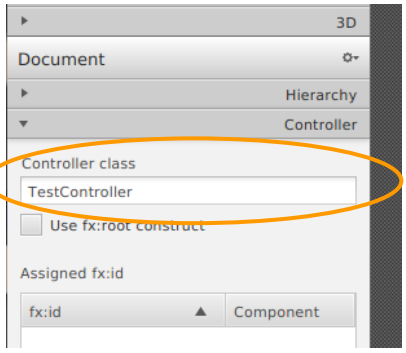
```java
public class Test2 extends Application{
	public static void main(String[] args) {
		launch(args);
	}

	public void start(Stage primaryStage){
		primaryStage.setTitle("JavaFX FXML Test");
		try {
			File file = new File("test.fxml");
			Parent root = XMLLoader.load(file.toURI().toURL());
			Scene scene = new Scene(root);
			primaryStage.setScene(scene);
			primaryStage.show();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```

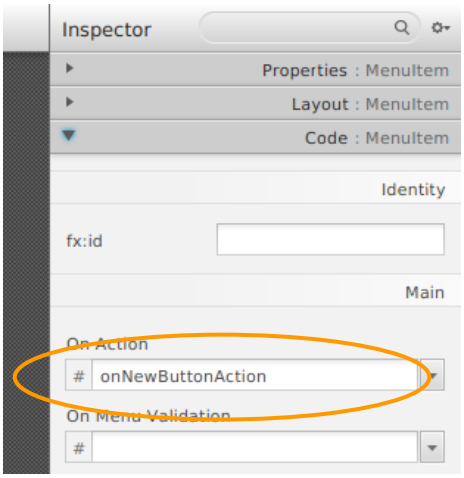
Controller

Um Ereignisse (z.B. Knopf wird gedrückt) abzufangen, wird eine Controller-Klasse benötigt. Dazu legen Sie einfach eine Klasse an. In diesem Beispiel wird die Klasse *TestController* angelegt.

Nun können Sie im Scene-Builder den Controller hinterlegen. Dazu wählen Sie auf der linken Seite die Kategorie *Controller* aus und geben im Textfeld *Controller class* den vollständigen Namen der Klasse ein. Wenn Ihre Klasse in einem Package-Bereich liegt, **muss der ganze Namensbereich** angegeben werden.



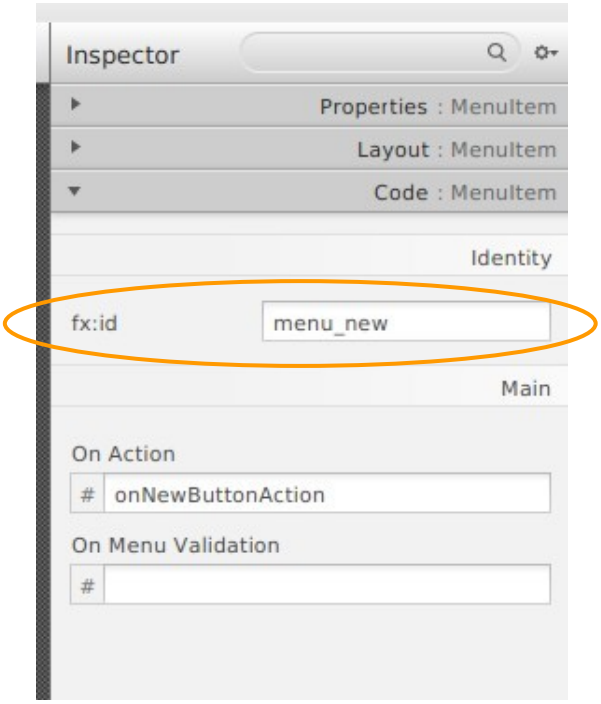
Nun können Sie bei den Elementen im Bereich Code die Methodennamen für die entsprechenden Events hinterlegen:



Die *Controller*-Klasse können Sie dann wie folgt aufbauen:

```
public class TestController {
     @FXML protected void onNewButtonAction(ActionEvent event) {
          System.out.println("Hallo");
     }
}
```

Um Zugriff auf ein Element zu erhalten, wählen Sie das Element im Scene-Builder aus und geben im *Code*-Bereich im Textfeld *fx:id* eine eindeutige Identität ein (Attributname):



Die Controller-Klasse kann dann wie folgt erweitert werden:

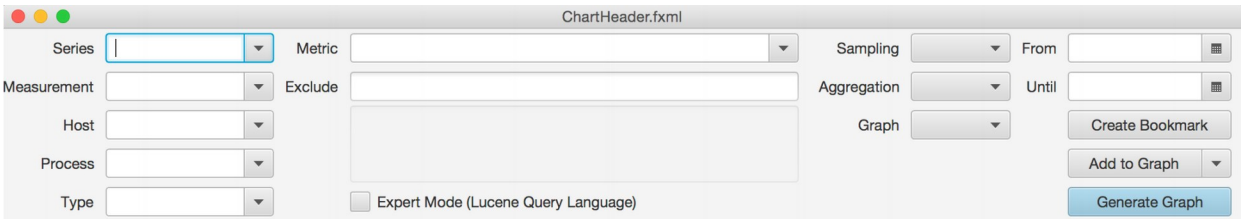
```
public class TestController {
	@FXML private MenuItem menu_new;

	@FXML protected void onNewButtonAction(ActionEvent event) {
		System.out.println("Hallo");
		menu_new.setText("Clicked!");
	}
}
```

Hinweis: Die Annotation @FXML benötigt man nur bei privaten Feldern (private, protected, default).

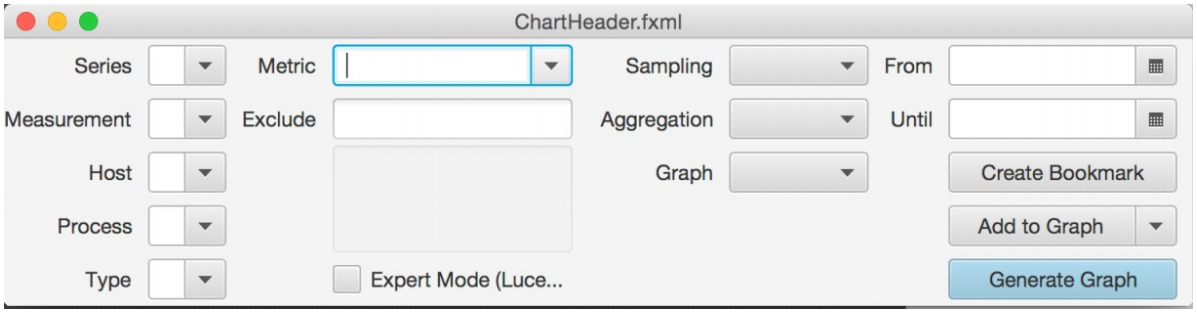
Aufgaben

1. Erstellen Sie eine JavaFX Oberfläche mit dem folgenden Inhalt:



Die Oberfläche soll in horizontaler Richtung sich optimal an den zu Verfügung stehenden Platz dynamisch anpassen. Folgende Anforderungen sind umzusetzen:

1. Die Comboboxen links verkleinern sich bis auf ein Minimum so, dass noch ein Buchstabe sichtbar bleibt.
2. Der Bereich in der Mitte (z.B. Combobox Metric) nutzen den restlichen zur Verfügung stehenden Platz.



2. Beim Druck auf den Button „Generate Graph“ sollen alle Inhalte strukturiert ausgegeben werden.
    1. Erstellen Sie dazu eine Klasse HeaderController. Diese Klasse wird im Scene-Builder als Controller registriert.
    2. Erstellen Sie im Controller die Methode void onGenerateGraph(ActionEvent e) und registrieren Sie diese im Scenebuilder.
    3. Testen Sie, ob beim Druck auf den Knopf die Methode aufgerufen wird.
    4. Schreiben Sie eine Klasse *HeaderSettings* die alle Werte des Dialogs in einem Objekt speichern kann. Alle Attribute sollen dabei auf *private* gesetzt werden und nur durch Methoden (Getter and Setter) veränderbar und aufrufbar sein:

```
public String getSeries() {
   return series;
}

public void setSeries(String series) {
   this.series = series;
}
```

Implementieren Sie eine *toSettings()*-Methode im Controller um ein *HeaderSettings*-Objekt zu erzeugen (Rückgabewert *HeaderSettings*-Objekt).

5. Beim Druck auf den Knopf sollen alle Eingabeparameter in der Konsole ausgegeben werden. Nutzen Sie dazu die *toString()*-Methode aus der Objekt-Klasse. Dabei sollen ihre Eingabeelemente folgende Parametern zulassen:

- ***Series***
  - *old, new*
- ***Measurement***
  - *fast, intensive*
- ***Host***
  - *fcc, ada, jfc*
- ***Process***
  - *0, 1*
- ***Type***
  - *max, average*
- ***Metric***
  - *0, 1*
- ***Sampling***
  - *0, 1, 2*
- ***Aggregation***
  - *0, 1, 2*
- ***From***
  - Datum in Unixzeit-Form
- ***Until***
  - Datum in Unixzeit-Form

3. Im Praktikumsordner finden Sie die Datei data.db. Hierbei handelt es sich um eine SQLite-Datenbank. In dieser Datenbank befindet sich eine Tabelle mit dem Namen *data*. Diese Tabelle enthält 100000 Einträge die mit der Struktur Ihrer Oberfläche übereinstimmt.

   Das bedeutet, dass diese Tabelle folgender Aufbau vorweist:

series	measurement	host	process	type	metric	sampling	aggregation	date	value
old	fast	jfs	0	max	1	2	0	1462765466	57.9869901522896
old	intensive	ffc	0	average	0	0	1	1134286925	23.0638976417373
new	fast	jfs	1	max	0	1	1	1041636569	23.4217682231306
new	intensive	ada	1	max	1	2	1	1171811722	0.246336768387057
old	fast	ffc	0	average	1	2	0	1317007124	23.3893453217775
old	intensive	jfs	0	average	0	0	1	1213227036	52.8462808066304
new	fast	ffc	1	max	1	2	1	1227617383	3.37589035603501
new	fast	ada	0	average	0	2	1	1376185288	50.6488462370366
old	intensive	ffc	1	max	0	1	2	1319953107	37.9748935807161
old	fast	jfs	1	average	0	0	1	1328445257	15.3800428015118
old	intensive	ada	1	average	1	1	2	1055924170	30.6361236783403
old	fast	ada	1	max	0	1	2	1279742664	93.003671281661
old	intensive	ada	1	average	0	1	2	1339651813	17.1643810691765

   Erweitern Sie nun Ihr Programm, dass dieses beim Drücken von *Generate Graph* die *value*-Werte (Y-Werte) in Abhängigkeit des Datums *date* (X-Werte) auf einen *LineChart* unterhalb der Eingabe- Elemente darstellt. Dabei sollen **nur** die Werte angezeigt werden, die den anderen, vom Benutzer angegebenen Werten (*series*, *measurement*, *host* ...), übereinstimmen, inklusive *From* und *Until*, die sich auf den Wert *date* beziehen.

   Die Werte *date* und *value* sind dabei wie folgt formatiert:

   - **date**
     - Datum in Unixzeit-Form
   - **value**
     - Werte von 0 bis 100 (Gleitkommazahl)

Um auf die Datenbank zugreifen zu können, können Sie die Programmbibliothek sqlite-jdbc verwenden. Lesen Sie sich dazu auf folgende Seite ein:

https://github.com/xerial/sqlite-jdbc#usage

Nutzen Sie SQL-Queries um die Daten aus der Datei gefiltert zu lesen. Eine kurze Einführung finden Sie auf folgender Seite:

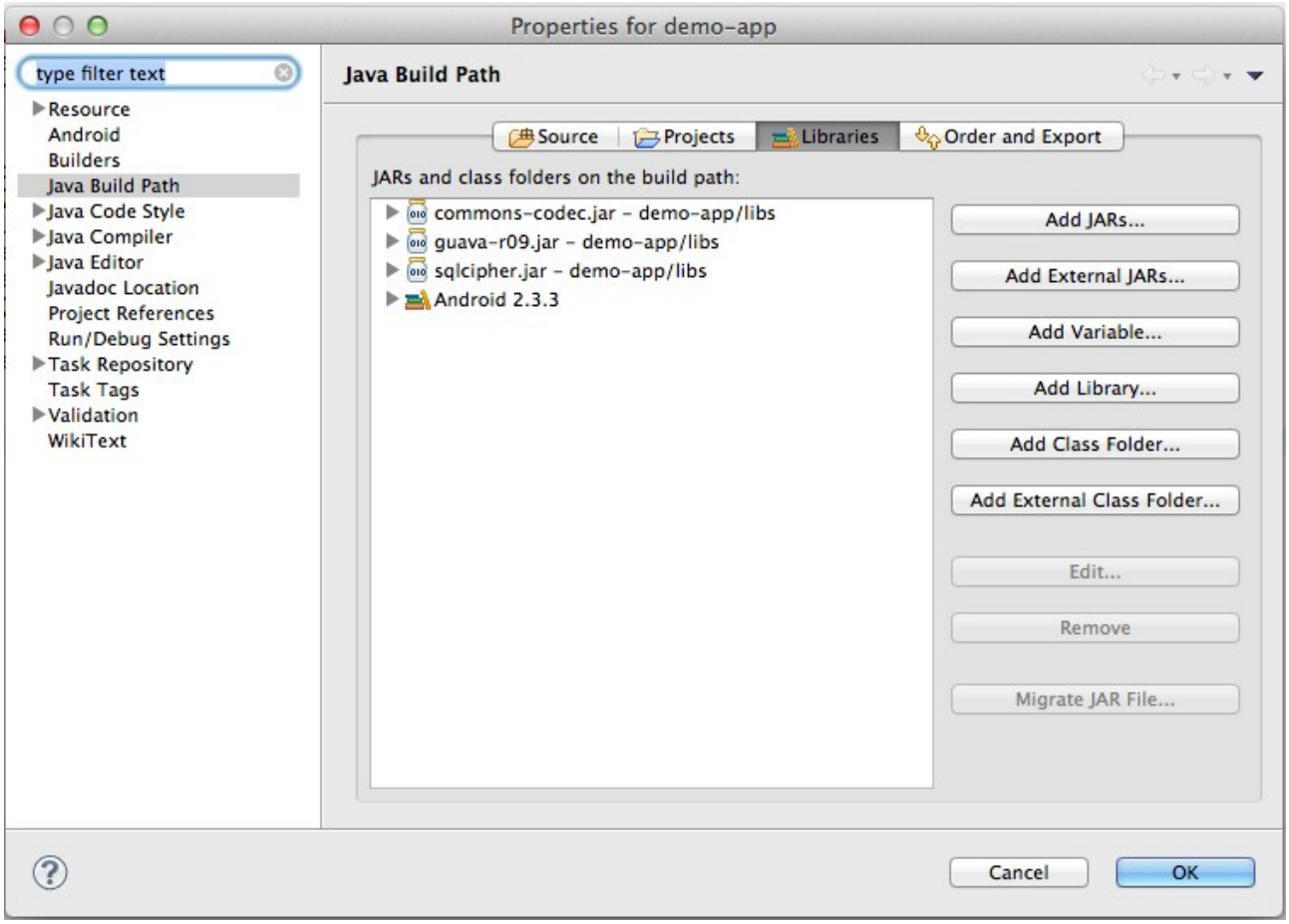
https://www.tutorialspoint.com/sql/index.htm

Wie Sie die Programmbibliothek in Eclipse einbinden, finden Sie auf der nächsten Seite.

Jar-Dateien Eclipse

sqlite-jdbc ist nicht in der allgemeinen Java-Bibliothek vorhanden und muss zusätzlich bezogen werden. Für Java existieren dazu verschiedene Möglichkeiten. Eine Variante ist der Bezug einer JAR-Datei (**J**ava **A**rchive). Diese Datei besitzt in sich die gesamte Programmstruktur einer Bibliothek und kann daher kompakt dem Projekt hinzugefügt werden. Die JAR-Datei von sqlite-jdbc finden Sie im Praktikum-Ordner. Alternativ können Sie auch die JAR-Datei von der Projektwebsei te beziehen.

Um eine JAR-Datei in Eclipse einzufügen können Sie auf *Project → Build Path → Configure Build Path → Libraries* gehen:



Dort können Sie über den Knopf *Add External JARs* die JAR-Datei in Ihrem Projekt implementieren. (Video-Anleitung)