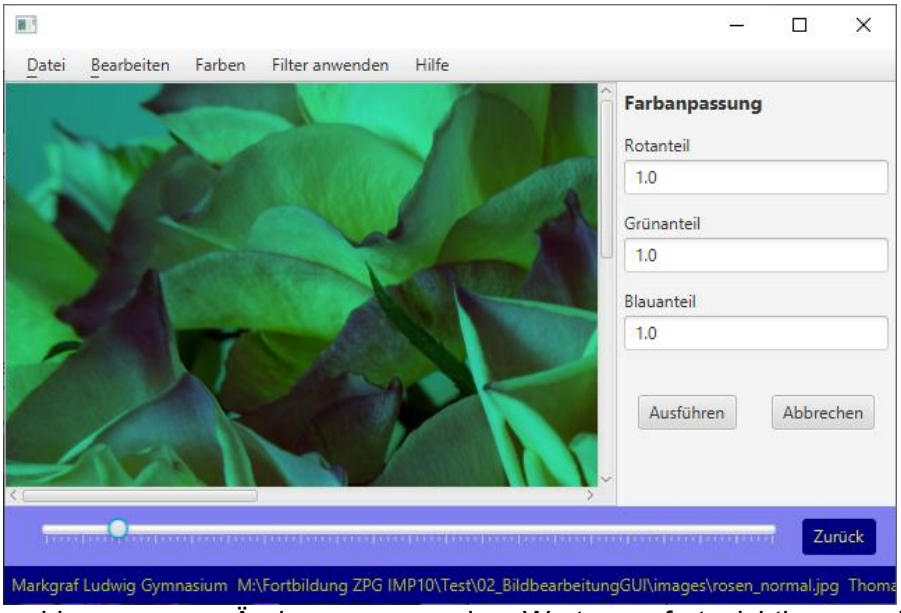
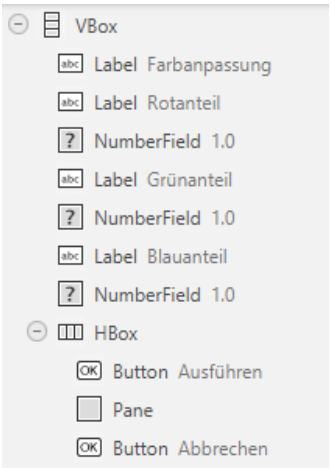
Bisher wurden für die Ausführung der Action-Methoden vom Benutzer keine Eingaben benötigt. Dies wird sich nun ändern. Benötigt eine Methode zusätzliche Informationen, dann soll im Hauptbereich neben dem Bild ein Eingabebereich erscheinen, in dem die Eingaben vorgenommen werden können.



Damit die Auswirkungen von Änderungen an den Werten sofort sichtbar werden, soll jede Änderung an den Werten eine Neuberechnung des Bildes auslösen. Der Button "Ausführen" bestätigt das aktuelle Bild, der Button "Abbrechen" kehrt zum vorherigen Bild zurück.

Gestaltung der Oberfläche



Die Eingabebereiche werden jeweils als eigene FXML-Dateien mit dem SceneBuilder gestaltet. Man startet mit einem leeren Dokument und fügt als Hauptelement zunächst eine VBox ein. Diese soll als PrefSize eine Breite von 200 Pixeln haben, die Höhe soll der berechneten Höhe entsprechen. Sie hat einen Innenrand (Padding) von fünf Pixeln.

Der VBox fügt man die Kontrollelemente hinzu und passt die Darstellung an (Schriftart, Abstände = Padding/Margin), so dass die Hierarchie der rechs zu sehenden entspricht und die Darstellung wie oben gezeigt aussieht. Danach werden die Eingabefelder benannt (fxid: nfRotanteil, nfGruenanteil, nfBlauanteil) und den Buttons eine Action-Methode hinzugefügt (bAusfuehren und bAbbrechen).

1. *Erstelle mit dem Scene Builder das Formular und speichere es in dem view-Ordner deines Projektes. Füge im Menü "Farben" einen neuen Menüeintrag "Farben anpassen" hinzu und trage eine Action-Methode ein.*

Einbinden des Eingabebereichs in die GUI

Genauso wie im Hauptprogramm wird die FXML-Datei geladen und automatisch interpretiert. Als Ergebnis erhält man eine VBox (das Hauptelement). Diese soll dem Hauptbereich (der HBox in der GUI) hinzugefügt werden. Wie jedes Containerelement hat die HBox eine Liste von Unterelementen. Diese Liste bekommt man mit getChildren(). Sollte schon ein Eingabebereich angezeigt werden, muss dieser zunächst entfernt werden.

2. *Öffne die Datei "optionenAnzeigen.java" und füge die darin enthaltene Methode deinem Controller hinzu. Kommentiere an den vorgesehenen Stellen den Quelltext.*

Damit kein neuer Controller für den Eingabebereich erstellt wird, wird mit setController(this) das aktuelle Controller-Objekt als Controller gesetzt werden. Nach dem Laden der FXML-Datei wird normalerweise die initialize-Methode() des Controllers aufgerufen. Es muss verhindert werden, dass Befehle aus der initialize-Methode erneut ausgeführt werden, da diese sich auf den Hauptteil der GUI beziehen. Daher wird ein neues Attribut in der Klasse definiert, das speichert, ob die GUI-Elemente schon initialisiert wurden:

```
private boolean schonInitialisiert;
```

Dieses wird im Konstruktor der Klasse, der beim Erzeugen eines Objekts ausgeführt wird, zunächst auf false gesetzt:

```
public Controller() {
    schonInitialisiert = false;
}
```

In der initalize-Methode wird dann kontrolliert, ob schon initialisiert wurde:

```
public void initialize() {
    if (!schonInitialisiert) { // noch nicht initialisiert?
       …
       schonInitialisiert = true;
    }
}
```

In der Action-Methode des entsprechenden Menüeintrages muss nun nur noch die Methode erzeugeOptionen mit dem Namen der FXML-Datei als Parameter aufgerufen werden:

```
optionenAnzeigen("farbanpassung.fxml");
```

3. *Video: gui_erstellen_9.mp4.*

Füge das Attribut schonInitialisiert in deinen Controller ein. Erstelle den Konstruktor und passe die initialize()-Methode an.
Füge eine Action-Methode für den Menüpunkt "Farben anpassen" in deinen Controller ein. Rufe dort die Methode erzeugeOptionen auf.
Erzeuge zwei zunächst leere Action-Methoden für die beiden Buttons.

Reaktion auf Änderung der Eingaben

Egal, ob man NumberTextField, TextField oder Slider zum Eingeben der Werte benutzt, kann das Programm darauf reagieren, dass die Eingabe sich geändert hat. Die dafür notwendige Methode kann leider nicht im Scene Builder eingetragen werden. Daher muss sie nach dem Erzeugen der Kontrollelemente dem Element zugewiesen werden.

TextField / NumberTextField: Setzen des Listeners für die TextProperty

```
nfRotanteil.textProperty().addListener(
  (observable, oldValue, newValue) -> farbanpassung(aktuellesBild)
);
```

Slider: Setzen des Listeners für die ValueProperty

```
sRotanteil.valueProperty().addListener(
  (observable, oldValue, newValue) -> farbanpassung(aktuellesBild)
);
```

Hier wird die Methode farbanpassung(...) als Listener zugeordnet und außerdem das aktuelle Bild übergeben. Innerhalb der Methode farbanpassung soll das übergebene Bild angepasst und im Viewer neu gesetzt werden. In der Viewer-Historie wird es aber nicht bei jeder Änderung eines Wertes gespeichert (saveOldImage auf false), da die endgültigen Einstellungen vom Benutzer erst ausprobiert werden. Statt dessen wird beim Erzeugen des Eingabebereichs das Bild ein einziges Mal mit viewer.pushImage() gespeichert. Beim Drücken des Abbrechen Buttons wird dieses Bild mit viewer.back() wieder hergestellt. Beim Speichern-Button bleibt das neu berechnete Bild erhalten. In beiden Fällen wird der Eingabebereich aus dem Hauptbereich entfernt (hauptbereich.getChildren().remove(1);).

Die Methode farbanpassung() selbst muss wie alle Methoden, die direkt auf eine Menüauswahl reagieren implementiert werden: Es werden also die Eingaben aus den Kontrollelementen ausgelesen, die Berechnungen durchgeführt und dann die Kontroll-Elemente angepasst. Die Kontrollelemente müssen - wie bei der Haupt-GUI auch - als Attribute deklariert werden (Copy-Paste aus dem Scene Builder ist auch hier möglich).

4. *Video: gui_erstellen_10.mp4*

Implementiere die Methode mFarbanpassung() so, dass eine Listener-Methode farbanpassung(...) für alle Eingabefelder gesetzt wird (für alle die gleiche). Außerdem muss das aktuelle Bild gesichert und an die Listener-Methode übergeben werden. Implementiere die Listener-Methode farbanpassung(Picture altesBild), so dass das alte Bild mit den eingetragenen Faktoren manipuliert und im viewer neu angezeigt wird. Implementiere die Button-Action-Methoden für "Abbrechen" und "Ausführen".

5. *Expertenaufgabe: Ersetze die NumberTextField-Eingabefelder durch Slider für den Bereich zwischen 0 und 2. Passe dein Programm entsprechend an.*