Medientechnik
Sommersemester 2016

Übung 03 (Bildfilter)

Terminübersicht

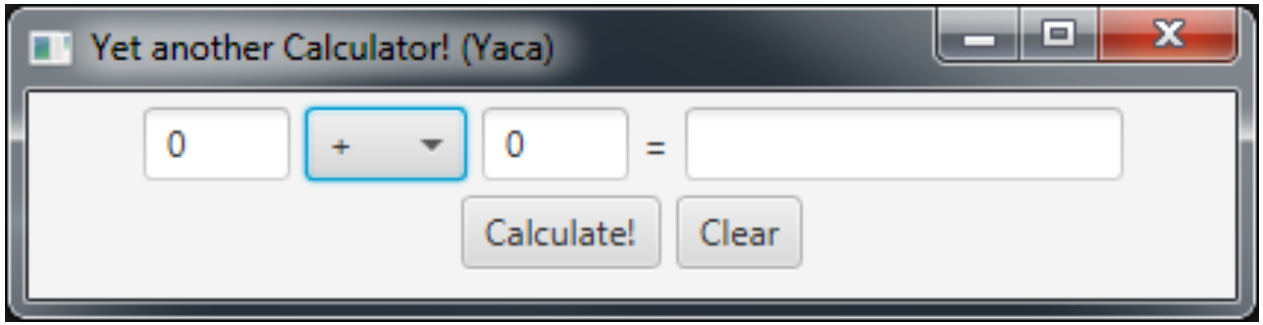
Nr	Zeitraum	Thema
1	18.04. - 21.04.	Organisatorisches, Bildbearbeitung
2	09.05. - 12.05.	JavaFX Einführung (GUIs, Szenengraph)
3	17.05. - 19.05.	Design Patterns: MVC, Observer
4	**23.05. - 25.05.**	**Bildfilter programmieren**
5	30.05. - 02.06.	Videobearbeitung
6	06.06. - 09.06.	Video Streams mit JavaFX
7	20.06. - 23.06.	Audiobearbeitung
8	27.06. - 30.06.	Multimedia mit JavaFX

Agenda

- Wiederholung Übung 2
- Dateien einlesen
- Bildfilter anwenden
- FXML
- Bildfilter Theorie
  - Konvolution
  - Farbfilter

Yaca

- Letzte Übung: YaCa
  - MVC
  - Observer
  - Events

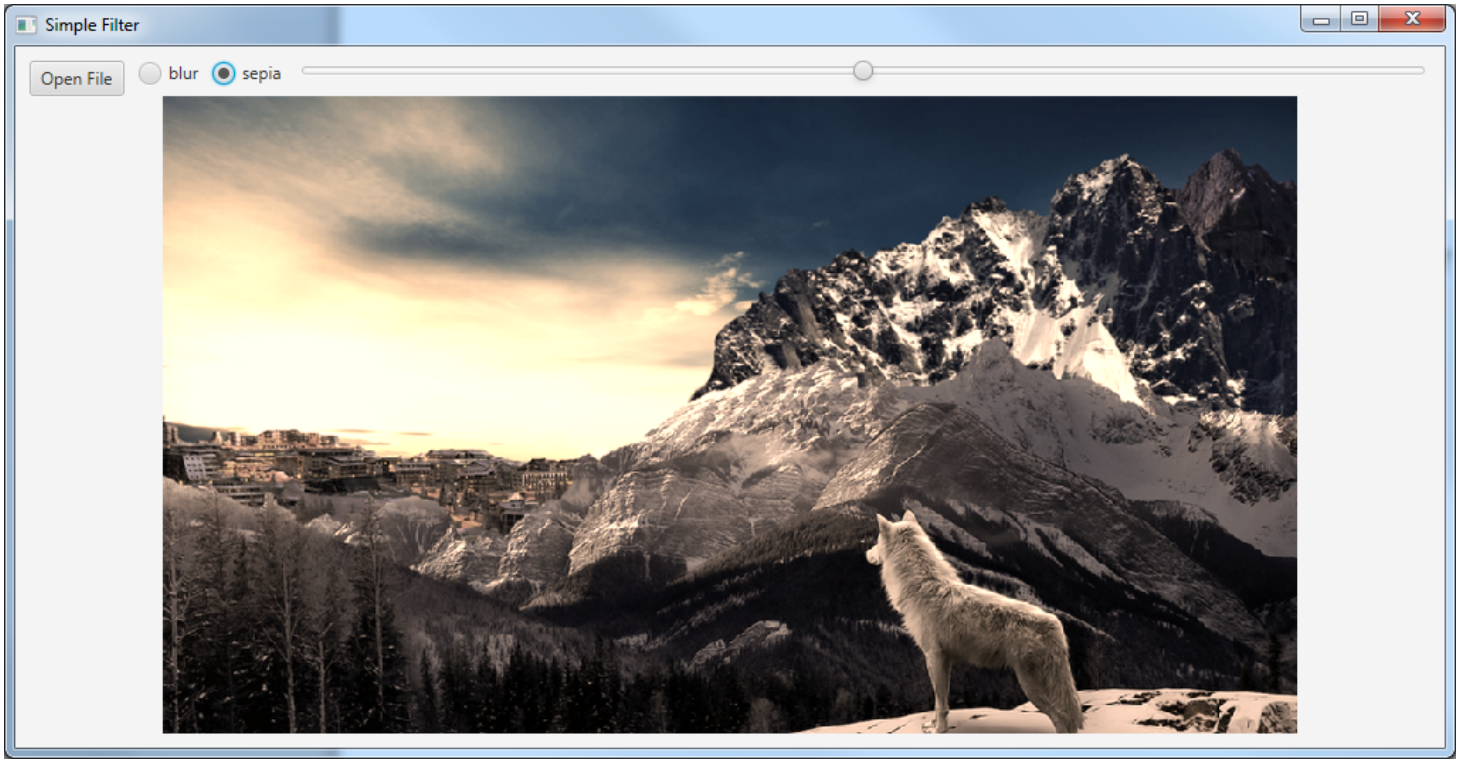


Warm-Up Quiz

1. Welche Methode der Schnittstelle EventHandler muss eine Klasse implementieren um sich für Events registrieren zu lassen?
2. Wie wird ein EventHandler einem Element hinzugefügt?
3. Ist Observable eine Klasse oder eine Schnittstelle?
4. Welche beiden Methoden müssen aufgerufen werden, wenn sich etwas am Modell geändert hat?

Heute: “Simple Filter”

- Wir erstellen ein einfaches Programm, welches beliebige Bilder öffnen kann und einen Filter anwendet. Ein Slider stellt die Effektstärke ein.



https://pixabay.com/de/wolf-natur-berg-1338354/ CC0

Setup

- Material von der Webseite herunterladen: http://www.medien.ifi.lmu.de/lehre/ss16/mt/uebung/ressourcen/mt_material03.zip

- Neues Projekt oder neues Modul in IDEA erstellen

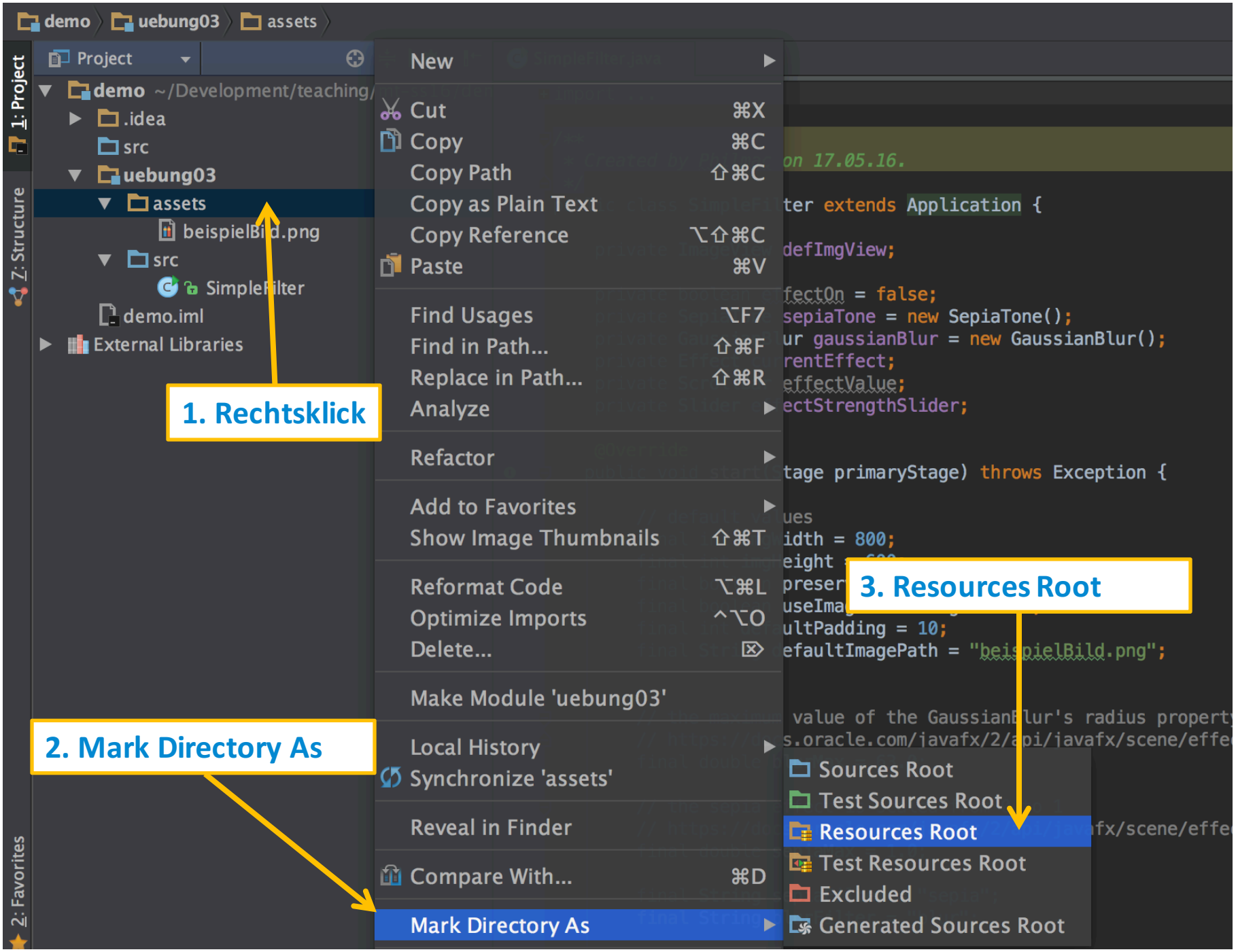
- Code Skelett enthält bereits lauffähige GUI in der Klasse **`SimpleFilter.java`**, die wir mit ein paar Funktionen erweitern werden.

Bildpfad

- Die Variable `defaultImagePath` enthält den Pfad zur Bilddatei.

- Eventuell muss der Pfad angepasst werden (je nach System bzw. Projektkonfiguration), zum Beispiel:
  - `"beispielBild.png"`
  - `"resources/images/beispielBild.png"`
  - `"file:beispielBild.png"`
  - `"C:/beispielBild.png"`

- Man kann ein Flag für bestimmte Ordner setzen, um dem Compiler mitzuteilen, wo man die Ressourcen abgelegt hat. Siehe Darstellung auf nächster Folie (IDEA)


demo
uebung03
assets
Project
demo ~/Development/teaching/
.idea
src
uebung03
assets
beispielBild.png
src
SimpleFilter
demo.iml
External Libraries
1. Rechtsklick
New
Cut ⌘X
Copy ⌘C
Copy Path ⇧⌘C
Copy as Plain Text
Copy Reference ⌥⇧⌘C
Paste ⌘V
Find Usages ⌥F7
Find in Path... ⇧⌘F
Replace in Path... ⇧⌘R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails ⇧⌘T
Reformat Code ⌥⌘L
Optimize Imports ^⌥O
Delete...
Make Module 'uebung03'
Local History
Synchronize 'assets'
Reveal in Finder
Compare With... ⌘D
Mark Directory As
2. Mark Directory As
Sources Root
Test Sources Root
Resources Root
Test Resources Root
Excluded
Generated Sources Root
3. Resources Root
1: Project
7: Structure
2: Favorites

TODOs 1-3

```
openFile.setOnAction(e -> {
    try{
        FileChooser fileChooser = new FileChooser();
        // TODO step 1

        // we need to convert the file into a URL, as String.
        // TODO step 2

        // immediately apply the effect, if we have one.
        // TODO step 3

    }catch (Exception exception){
        System.out.println(exception.getMessage());
    }
});
```

Step 1: File Chooser Dialog

```
openFile.setOnAction(e -> {
    try{
        FileChooser fileChooser = new FileChooser();
        fileChooser.getExtensionFilters().add(
                new FileChooser.ExtensionFilter("JPG", "*.jpg"));

        // this actually shows the dialog.
        File loadedFile = fileChooser.showOpenDialog(null);

        // we need to convert the file into a URL, as String.
        // TODO step 2

        // immediately apply the effect, if we have one.
        // TODO step 3

    }catch (Exception exception){
        System.out.println(exception.getMessage());
    }
});
```

Step 2: Image Objekt instanziieren

```
openFile.setOnAction(e -> {
    try{
        FileChooser fileChooser = new FileChooser();
        fileChooser.getExtensionFilters().add(
             new FileChooser.ExtensionFilter("JPG", "*.jpg"));

        File loadedFile = fileChooser.showOpenDialog(null);

        // we need to convert the file into a URL, as String.
        String imgURL = loadedFile.toURI().toURL().toString();

        // construct the new Image
        Image loadedImg = new Image(imgURL,
                  imgWidth, imgHeight,
                  preserveImageRatio, useImageSmoothing);
        // now update the image view
        defImgView.setImage(loadedImg);

        // immediately apply the effect, if we have one.
        // TODO step 3

    }catch (Exception exception){
        System.out.println(exception.getMessage());
    }
});
```

Bildfilter in JavaFX

- JavaFX hat bereits sehr viele Bildfilter integriert, z.B.
  - SepiaTone: Farbfilter
  - GaussianBlur: Weichzeichner
  - Glow: “Farbglühen” Effekt
- Wichtigstes Package: javafx.scene.effect
  https://docs.oracle.com/javafx/2/api/javafx/scene/effect/package-summary.html

Package javafx.scene.effect

Provides the set of classes for attaching graphical filter effects to JavaFX Scene Graph Nodes.

See: Description

Class Summary

Class	Description
Blend	An effect that blends the two inputs together using one of the pre-defined `BlendModes`.
BlendBuilder<B extends BlendBuilder<B>>	Builder class for javafx.scene.effect.Blend
Bloom	A high-level effect that makes brighter portions of the input image appear to glow, based on a configurable threshold.
BloomBuilder<B extends BloomBuilder<B>>	Builder class for javafx.scene.effect.Bloom
BoxBlur	A blur effect using a simple box filter kernel, with separately configurable sizes in both dimensions, and an iteration parameter that controls the quality of the resulting blur.
BoxBlurBuilder<B extends BoxBlurBuilder<B>>	Builder class for javafx.scene.effect.BoxBlur
ColorAdjust	An effect that allows for per-pixel adjustments of hue, saturation, brightness, and contrast.
ColorAdjustBuilder<B extends ColorAdjustBuilder<B>>	Builder class for javafx.scene.effect.ColorAdjust
ColorInput	An effect that renders a rectangular region that is filled ("flooded") with the given `Paint`.
ColorInputBuilder<B extends ColorInputBuilder<B>>	Builder class for javafx.scene.effect.ColorInput
DisplacementMap	An effect that shifts each pixel by a distance specified by the first two bands of of the specified `FloatMap`.
DisplacementMapBuilder<B extends DisplacementMapBuilder<B>>	Builder class for javafx.scene.effect.DisplacementMap
DropShadow	A high-level effect that renders a shadow of the given content behind the content with the specified color, radius, and offset.
DropShadowBuilder<B extends DropShadowBuilder<B>>	Builder class for javafx.scene.effect.DropShadow
Effect	The abstract base class for all effect implementations.
FloatMap	A buffer that contains floating point data, intended for use as a parameter to effects such as `DisplacementMap`.
FloatMapBuilder<B extends FloatMapBuilder<B>>	Builder class for javafx.scene.effect.FloatMap

Unsere `currentEffect` Variable

- `private Effect currentEffect;`
  Wird verändert, wenn sich der aktuell ausgewählte RadioButton ändert:

```
filterToggleGroup
    .selectedToggleProperty()
    .addListener(
      (observable, oldValue, newValue)
       -> { });
```

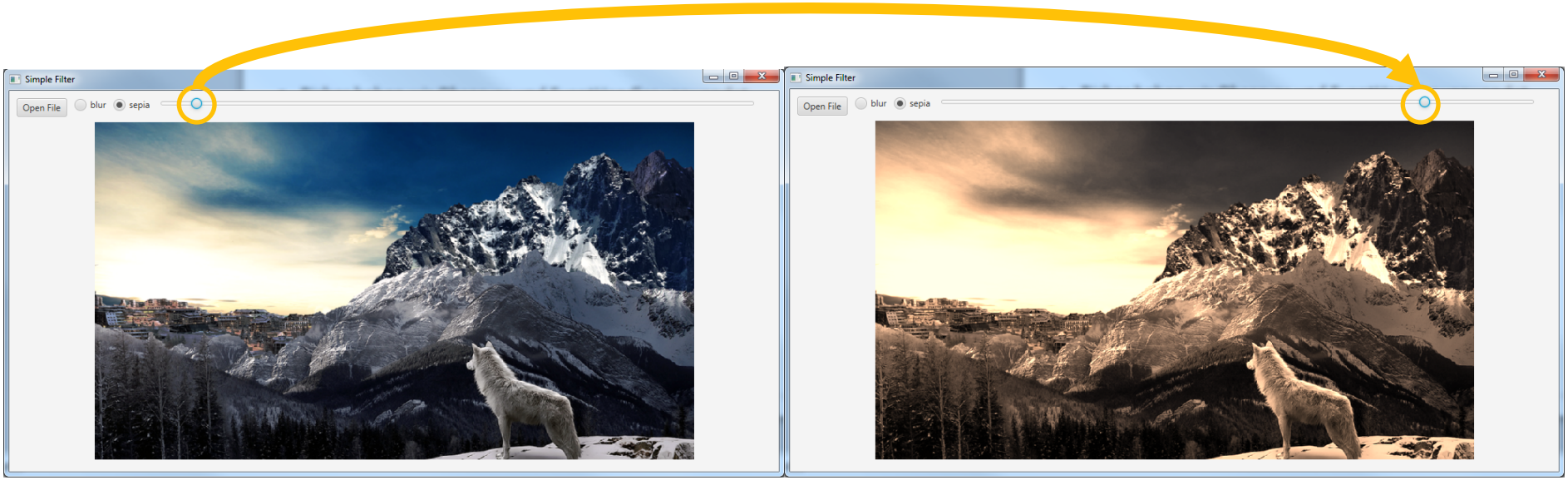
- Funktionsweise: je nachdem welcher RadioButton ausgewählt ist, wird currentEffect entweder zu einem SepiaTone oder GaussianBlur. Das funktioniert weil die vorgefertigten Effekte von javafx.scene.effect.Effect erben.

Step 3: Filter anwenden

```
openFile.setOnAction(e -> {
    try{
        FileChooser fileChooser = new FileChooser();
        fileChooser.getExtensionFilters().add(
                new FileChooser.ExtensionFilter("JPG", "*.jpg"));

        File loadedFile = fileChooser.showOpenDialog(null);

        String imgURL = loadedFile.toURI().toURL().toString();

        Image loadedImg = new Image(imgURL,
                imgWidth, imgHeight,
                preserveImageRatio, useImageSmoothing);
        defImgView.setImage(loadedImg);

        // immediately apply the effect, if we have one.
        if(currentEffect != null) {
            defImgView.setEffect(currentEffect);
        }

    }catch (Exception exception){
        System.out.println(exception.getMessage());
    }
});
```

DataBinding: Effektstärke anpassen

- Bisher haben wir Observer und EventHandler verwendet.
- Ziele für die Effektstärke:
  - Der Wert des Sliders soll die Effektstärke widerspiegeln
  - Sobald sich der Slider-Wert ändert, soll sich die Effektstärke ändern



Databinding: Details

- Ziel: Daten / Werte austauschen, z.B. zwischen View und Model ohne einen Event Handler implementieren zu müssen.

- Varianten:
  - Unidirektional:
    Änderung an Wert in A bewirkt Änderung in B, aber nicht umgekehrt
  - Bidirektional:
    Änderung an Wert in A bewirkt Änderung in B, **und umgekehrt**

- In JavaFX wird Databinding durch **Properties** möglich gemacht. Mehr Infos:
  https://docs.oracle.com/javafx/2/binding/jfxpub-binding.htm

Step 4: Databinding Slider ↔ Effekt

```
gaussianBlur.radiusProperty()
            .bindBidirectional(
                effectStrengthSlider.valueProperty());


sepiaTone.levelProperty()
         .bindBidirectional(
                effectStrengthSlider.valueProperty());
```

Vorkehrungen bei Effektwechsel…

```
String newEffectName = (String)((RadioButton)
newValue).getUserData();
double relativeSliderValue =
                effectStrengthSlider.getValue() /
                effectStrengthSlider.getMax();
switch (newEffectName){
    case sepiaFilter:
        effectStrengthSlider.setMax(sepiaMax);
        currentEffect = sepiaTone;
        break;
    case blurFilter:
        effectStrengthSlider.setMax(blurMax);
        currentEffect = gaussianBlur;
        break;
}
effectStrengthSlider.setValue(
    relativeSliderValue * effectStrengthSlider.getMax());
defImgView.setEffect(currentEffect);
```

Hausaufgaben

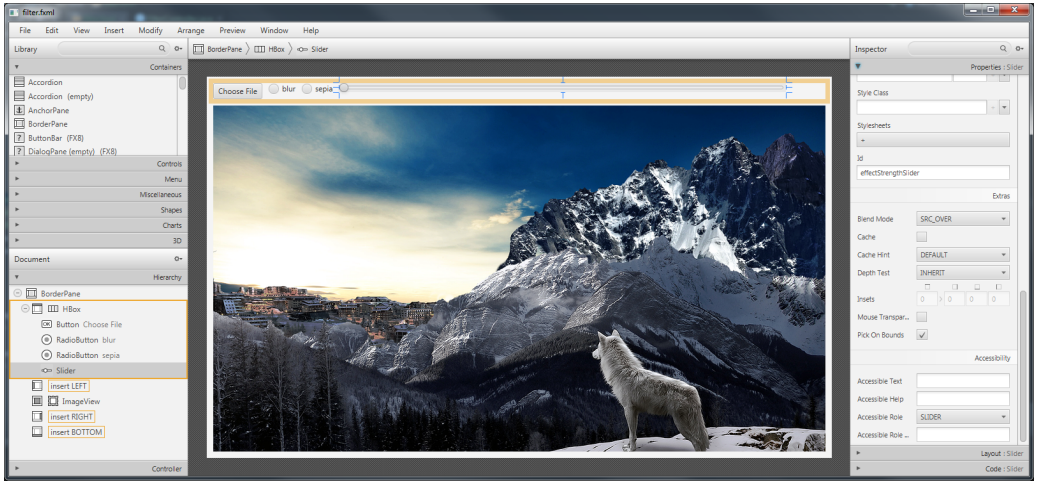
- Ändere den Code so ab, dass er einem MVC Pattern folgt.
- Füge weitere Effekte hinzu und lasse sie über Radio Buttons aktivieren

Franz Xaver Max Ludwig?

FXML

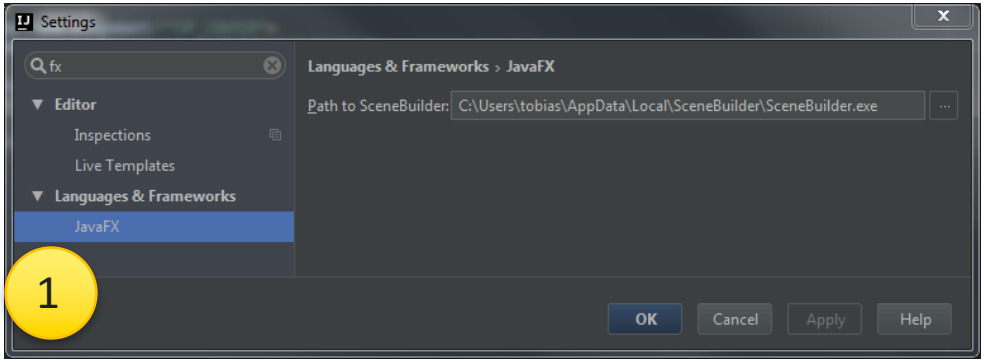
FXML

- JavaFX-spezifische Erweiterung von XML
- Erlaubt das einfachere gestalten von Layouts für GUIs
- Weniger imperativer Code = weniger Overhead
- SceneBuilder:
  - Drag & Drop Erstellung von Layouts
  - Download: http://gluonhq.com/open-source/scene-builder/

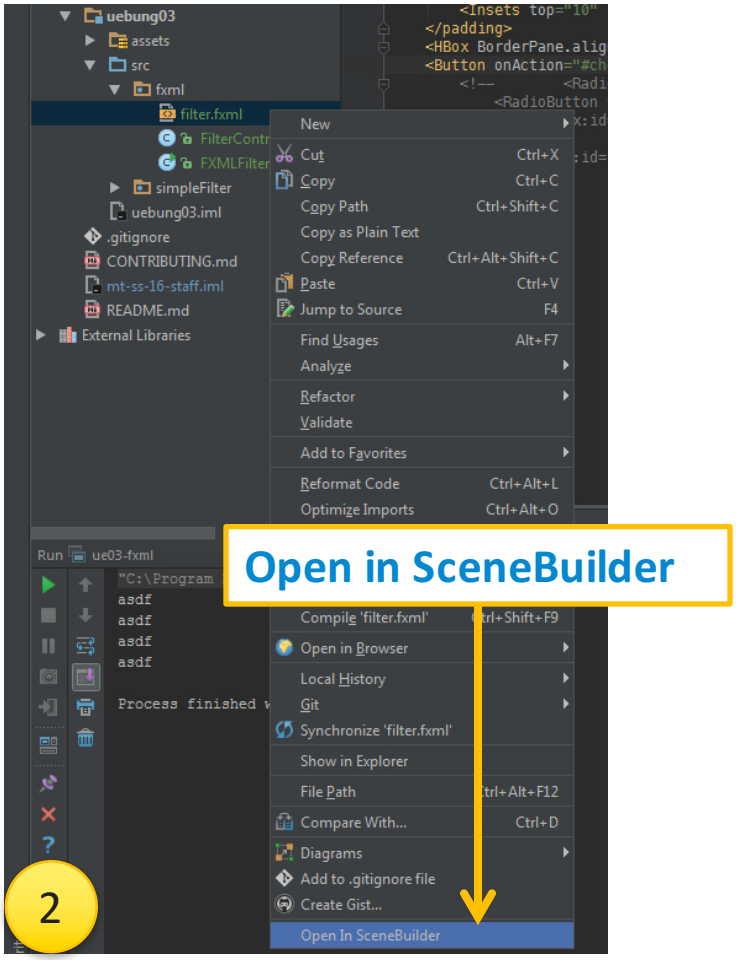


SceneBuilder in IntelliJ IDEA

- Pfad zu SceneBuilder einrichten:



- Anschließend Rechtsklick auf .fxml Datei und “Open in Scenebuilder”



Filter GUI in FXML: filter.fxml (1)

Klassenname des Controllers

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

<?import […] ?>

<BorderPane
    xmlns="http://javafx.com/javafx/8.0.65"
    xmlns:fx="http://javafx.com/fxml/1"
    fx:controller="fxml.FilterController">
    <padding>
        <Insets bottom="10" left="10" right="10" top="10" />
    </padding>
    <top>
        […  nächste Folie …]
    </top>
    <center>
        <ImageView fx:id="img">
            <image>
                <Image url="beispielBild.png" />
            </image>
        </ImageView>
    </center>
</BorderPane>
```

fx:id gibt den Instanzvariablennamen im Controller an!

Filter GUI in FXML: filter.fxml (2)

```
<top>
    <HBox prefWidth="900"
          spacing="10.0"
          BorderPane.alignment="TOP_CENTER">
        <Button id="chooseFileButton"
                onAction="#chooseFile"
                text="Choose File" />
        <RadioButton id="blurRadio"
                     fx:id="blurRadio"
                     onAction="#handleRadioButton"
                     text="blur" />
        <RadioButton id="sepiaRadio"
                     fx:id="sepiaRadio"
                     onAction="#handleRadioButton"
                     text="sepia" />
        <Slider id="effectStrengthSlider"
                fx:id="effectStrengthSlider"
                prefWidth="700.0" />
        <BorderPane.margin>
            <Insets bottom="10.0" />
        </BorderPane.margin>
    </HBox>
</top>
```

#chooseFile → Methodenname im Programmcode

Filter GUI in FXML: FilterController (1)

```
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
[… weitere Imports …]
public class FilterController implements Initializable {

    @FXML private Slider effectStrengthSlider;
    @FXML private ImageView img;
    @FXML private RadioButton blurRadio;
    @FXML private RadioButton sepiaRadio;

    [… ein paar Instanzvariablen]
    @FXML protected void chooseFile(ActionEvent e){
          […fast gleicher Programmcode wie in SimpleFilter …]}

    @FXML protected void handleRadioButton(ActionEvent e){

        double relativeSliderValue = effectStrengthSlider.getValue() /
                                     effectStrengthSlider.getMax();
        if (sepiaRadio.isSelected()) {
            effectStrengthSlider.setMax(sepiaMax);
            currentEffect = sepiaTone;
        }
        else if (blurRadio.isSelected()){
            effectStrengthSlider.setMax(blurMax);
            currentEffect = gaussianBlur;
        }
        effectStrengthSlider.setValue(relativeSliderValue * effectStrengthSlider.getMax());
        img.setEffect(currentEffect);
    }
    […nächste Folie …]
}
```

Variablenname = fx:id

Filter GUI in FXML: FilterController (2)

```
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
[… weitere Imports ]]
public class FilterController implements Initializable {

    […vorherige Folie …]

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        this.effectStrengthSlider.setMax(this.sepiaMax);
        this.effectStrengthSlider.setValue(this.effectStrengthSlider.getValue() / 2);

        // Databinding
        gaussianBlur.radiusProperty().bindBidirectional(effectStrengthSlider.valueProperty());
        sepiaTone.levelProperty().bindBidirectional(effectStrengthSlider.valueProperty());

        ToggleGroup toggleGroup = new ToggleGroup();
        blurRadio.setToggleGroup(toggleGroup);
        sepiaRadio.setToggleGroup(toggleGroup);
    }

    public void setStage(Stage stage){
        this.stage = stage;
    }
}
```

Filter GUI in FXML: FXMLFilterMain

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
[… weitere imports …]

public class FXMLFilterMain extends Application {

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

    @Override
    public void start(Stage primaryStage) throws IOException {
        FXMLLoader loader =
              new FXMLLoader(getClass().getResource("filter.fxml"));

        Parent root = loader.load();
        FilterController controller = loader.getController();
        controller.setStage(primaryStage);

        Scene scene = new Scene(root);

        primaryStage.setScene(scene);
        primaryStage.sizeToScene();
        primaryStage.setTitle("FXML Filter");
        primaryStage.show();
    }
}
```

FXML Bemerkungen

- id vs. fx:id
  - id = CSS selector, wird nur für Styling benötigt
  - fx:id wird für die @FXML Annotation benötigt und erlaubt im Programmcode den Zugriff auf das Element zu bekommen. Die fx:id muss dem Namen der Instanzvariable entsprechen.
- Vergleich in Lines-of-Code:
  - SimpleFilter.java: **177**
  - fxml package: **193** **(davon 46 in filter.fxml)**

  → für kleine GUIs mit wenigen Elementen entsteht ein leichter Overhead (falls man das FXML von Hand schreibt). Der Vorteil wird erst bei aufwendigen GUIs richtig deutlich.

Was steckt dahinter?

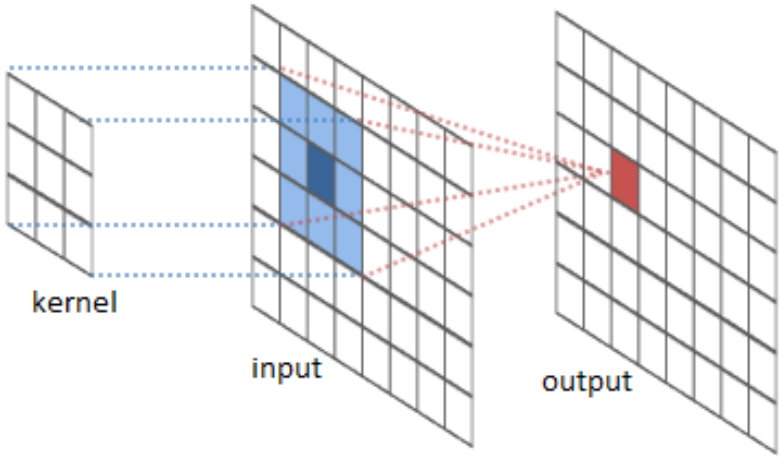
BILDFILTER THEORIE

Effekt = Filter = Pixel Manipulation

- Bilder bestehen aus “Bildpunkten” (Pixel)
- Jedem Pixel ist ein Farbwert zugeordnet, der sich i.d.R. aus drei Farben zusammensetzt (rot, grün, blau = RGB)
- Farbwerte und Helligkeiten lassen sich mit verschiedenen Mitteln verändern, z.B.
  - **Konvolution**
  - Bit-Operationen

Konvolution: Idee

- Konvolution (engl. Convolution) = Faltung
- Idee: Farbwert bzw. Helligkeitswert eines einzelnen Pixels (Source Pixel) wird mit umliegenden Pixel kombiniert
- Ergebnis wird in Ziel Pixel an gleicher Stelle gespeichert



http://colah.github.io/posts/2014-07-Understanding-Convolutions/img/RiverTrain-ImageConvDiagram.png

Kernel

$$
\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
$$

Konvolution: Kernel

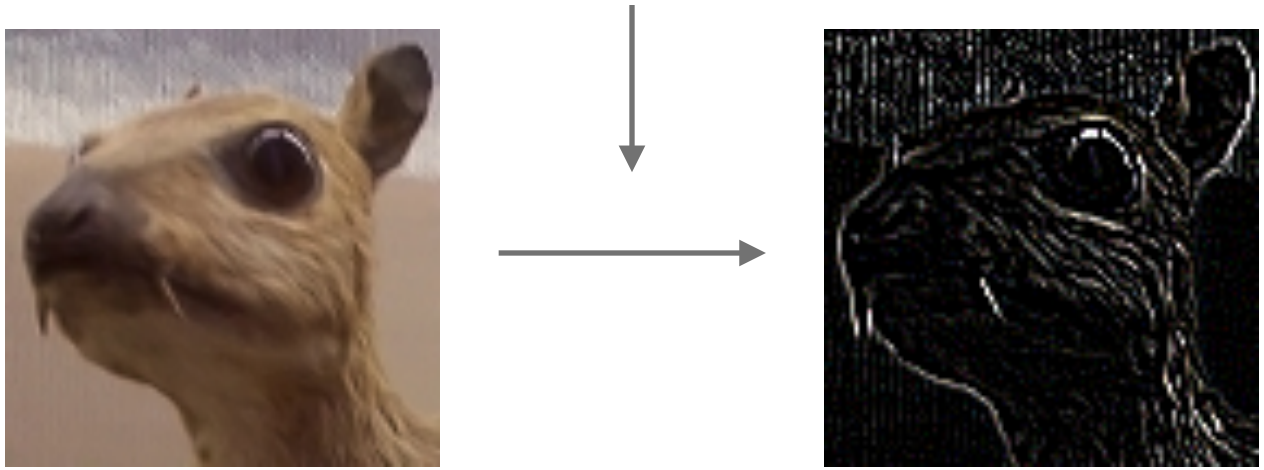
- Umsetzung durch Faltungsmatrix = Kernel
- Werte in den Matrizen werden paarweise multipliziert, dann addiert (keine Matrixmultiplikation!)
- Identitätskernel:

$$\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

$$\begin{pmatrix} 4 & 12 & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & 45 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \mathbf{1}$$

Kantenerkennung mit Kernel

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$



https://en.wikipedia.org/wiki/Kernel_(image_processing)

Kantenerkennung: Berechnung

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & 10 & 1 \\ 1 & \mathbf{10} & 1 \\ 1 & 10 & 1 \end{pmatrix} \Rightarrow$$

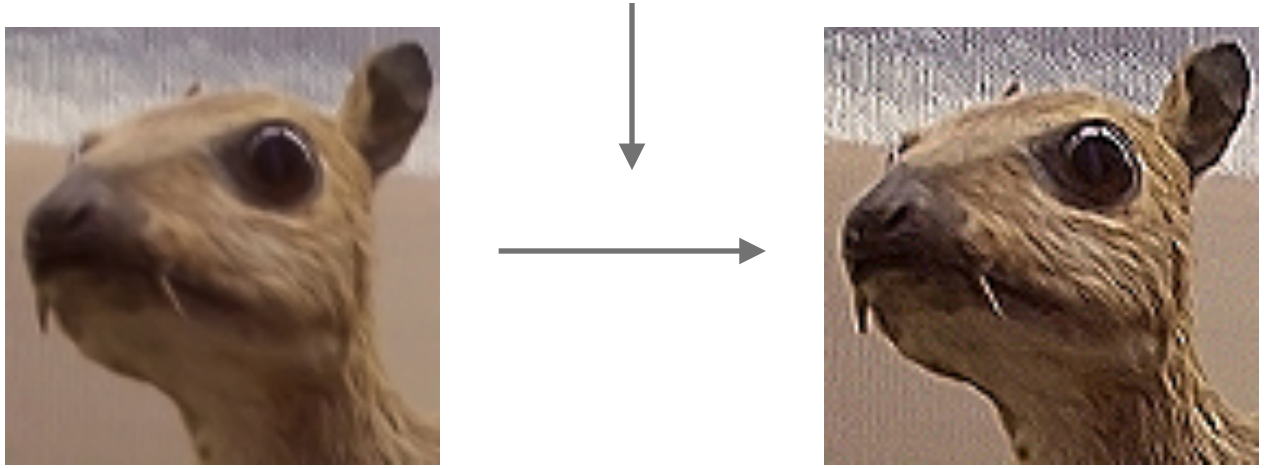
$$\begin{aligned} &(-1) * 1 + (-1) * 10 + (-1) * 1 + \\ &(-1) * 1 + 8 * \mathbf{10} + (-1) * 1 + \\ &(-1) * 1 + (-1) * 10 + (-1) * 1 \end{aligned}$$

$$=$$

$$\mathbf{54}$$

Schärfen mit Kernel

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$



https://en.wikipedia.org/wiki/Kernel_(image_processing)

Weichzeichnen

$$\frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

https://en.wikipedia.org/wiki/Kernel_(image_processing)

Breakout: Rechnen

- Gegeben ist dieser Konvolutionskernel:

$$\frac{1}{10}\begin{pmatrix}1 & 5 & 1\\ 5 & 10 & 5\\ 1 & 5 & 1\end{pmatrix}$$

- Mit folgenden Bildpunkten:

$$\begin{pmatrix}10 & 10 & 10\\ 20 & 100 & 20\\ 20 & 10 & 10\end{pmatrix}$$

GaussianBlur in JavaFX

- `radius` property
  Gibt an, wie weit ein anderes Pixel vom aktuell betrachteten entfernt sein darf, damit es in der Berechnung miteingeschlossen wird.
  Beispiel mit Radius = 2 (nur schematisch, Details weichen ab)

$$\frac{1}{16}\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & 4 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Konvolution: Probleme

- Wie gehen wir mit dem Bildrand um?
  - Ignorieren = Unverändert lassen = Source Pixel übernehmen
  - Erweitern
    - Wert vom nächsten Pixel übernehmen
    - Wert von der gegenüberliegenden Seite übernehmen
  - Ignorieren und Bild zuschneiden

- Werte außerhalb des Farbraums können wir normalisieren: höchsten und niedrigsten Wert herausfinden und die dazwischenliegenden Werte auf eine neue Skala übertragen (z.B. (Min) 100 → 0, (Max) 500 → 255 für RGB Werte)

Bit-Operatoren

- Beispiel RGB Farbwert einen Pixels in Binärdarstellung:

```
01001010 10011100 01111010
```

- darauf können Bit-Operatoren angewendet werden:
  - & UND
  - | ODER
  - >> nach RECHTS verschieben (Bitshift)
  - << nach LINKS verschieben

Beispiel: Grünwerte herausfiltern Teil 1

```
01001010 10011100 01111010
   R        G        B
```

- Wir wollen nur die Grünwerte extrahieren → Bit-Maske

```
  01001010 10011100 01111010
& 00000000 11111111 00000000
---------------------------
  00000000 10011100 00000000
```

- Ergebnis: Rot und Blau wurden auf null gesetzt

Beispiel: Grünwerte herausfiltern Teil 2

- Zweiter Schritt: Grünwert nach rechts verschieben

```
  00000000 10011100 00000000
                       >> 8
----------------------------
  00000000 00000000 10011100
```

- Ergebnis: Grünwerte wurden extrahiert

Wrap-Up Quiz

1. Wie verbindet man Controller und FXML Datei?
2. Wie lautet der deutsche Begriff für “Konvolution”?
3. Kann man mit einer Konvolution ein Bild in Schwarz-Weiß umwandeln?

Vielen Dank!

WELCHE FRAGEN HABT IHR?

Credits

- Philip Ngyuen: Simple Filter Beispiel
- Lea Rieger: Bildfilter Theorie Folienupdate
- Links:
  - http://blog.axxg.de/javafx-databinding-fxml/
  - http://www.javafxtutorials.com/tutorials/radio-buttons-in-javafx-and-fxml/