# Übungsblatt 2 – MVC & Observables

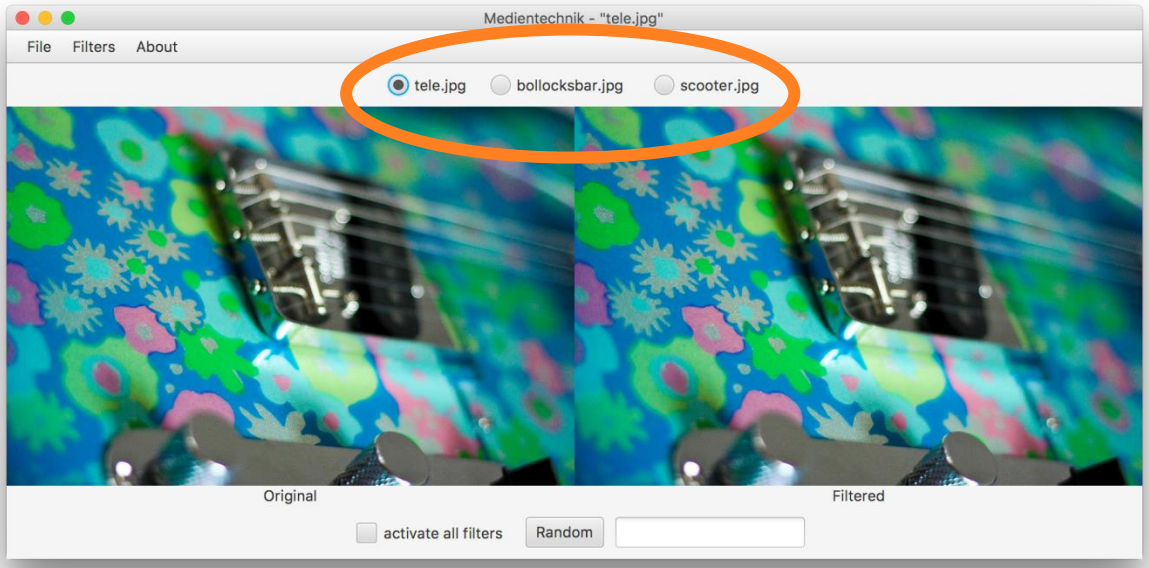
## Aufgabe 1: MVC und Observer Pattern für die Bildbetrachter GUI (15P)

Nachdem auf dem letzten Übungsblatt eine GUI für ein Bildbearbeitungsprogramm entworfen wurde, soll jetzt eine erste Funktionalität hierfür implementiert werden. Zur Implementierung soll eine **Model-View-Controller** Architektur genutzt werden, wobei die View über ein **Observer-Pattern** aktualisiert wird.

Die Bildfilter werden Teil von Übungsblatt 3 sein.

Folgende Funktionen soll das Programm bieten:

- Die Nutzer/-innen können eines von drei Bildern auswählen. Dabei wird eine Gruppe von Radio Buttons im oberen Bereich angezeigt.




  Die Pfade aller Bilder sind dabei im Model hinterlegt und werden der View z.B. im Konstruktor übergeben. Es soll also möglich sein, im Model weitere Pfade zu hinterlegen, die dann in der View automatisch zur Verfügung stehen. Es ist sinnvoll, den Pfad auch den RadioButtons mittels .setUserData() zu übergeben.
  *Hilfreiche Klassen: ToggleGroup, RadioButton, ArrayList*

- Wird zum ersten Mal ein Radio Button angeklickt, soll das Bild geladen mit dem angegebenen Pfad geladen werden. Das ausgewählte Bild soll per „Lazy Loading“ geladen werden. Das bedeutet, erst wenn die Datei das erste Mal gebraucht wird, wird das Objekt erzeugt.
  *Hilfreiche Klassen: Image, ImageView*

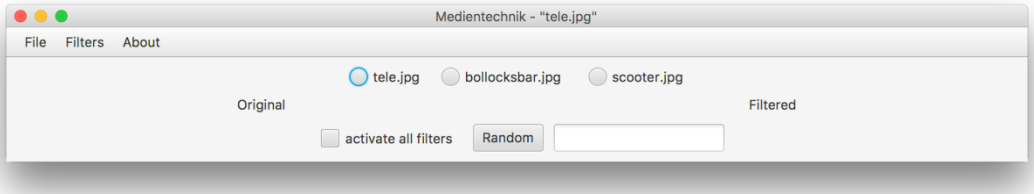



*Abbildung 1 - zu Beginn wird noch kein Bild angezeigt.*

- Das ausgewählte Bild soll sowohl im linken als auch im rechten „Slot“ angezeigt werden. Achten Sie auf die korrekte **Skalierung** des Bildes bzw. des Frames.
- Der Menüpunkt „**Exit**“ im Menü soll das Programm beenden.
- Der Menüpunkt „**About Medientechnik**“ liefert eine kurze Info, z.B. zum Programm-Autor.
  *Hilfreiche Klassen: Alert (benötigt Java Version ≥ 8u40)*
- Die Checkboxen im Menüpunkt „**Filters**“ sind einzeln (de-)aktivierbar. Da noch keine Auswirkungen (sprich: Bildfilter) implementiert werden, reicht es vorerst, wenn im Model des Programms entsprechende Variablen gesetzt werden, die verdeutlichen, dass die Filter aktiv sind.
  *Hilfreiche Klassen: HashMap<String, Boolean>*
- Im „Filters“ **Textfeld** werden immer die Namen aller aktiven Filter angezeigt.
  *Hilfreiche Methode: String.join(…)*

Im Model des Programms sollen dabei u.a. folgende Informationen gespeichert werden:

- Die Pfade aller verfügbaren Bilder, sowie der Pfad des aktuell geladenen Bilds
- die HauptBühne (`primaryStage`)
- die verfügbaren Filternamen (z.B. als ArrayList)
- Aktivierte Filter (z.B. mit HashMap oder mehrere Boolean Variablen)

**Packen Sie alle erstellten Dateien in ein ZIP Archiv (kein .rar) mit dem Namen „aufgabe1.zip“. Falls Ihre Lösung nicht lauffähig sein sollte, legen Sie bitte der Abgabe eine „README“ Datei bei, die folgendes enthält:**

- Was ist das Problem?
- Was haben Sie alles zur Lösung des Problems schon versucht (inkl. Links im Web)?
- Welche konkreten Fragen haben Sie dabei?

## Aufgabe 2: Fortgeschrittene Themen (5P):

Hinweis: ein Teil des Stoffs, der zum Laden einer neuen Datei notwendig ist, wird Ihnen erst in der 3. Übungsstunde präsentiert.

- Sobald ein Bild geladen wurde, soll der **Name der Datei in der Titelleiste** des Fensters entsprechend aktualisiert werden.
- Der „**Alle Effekte**“ Button ermöglicht die (De-)Aktivierung aller Effekte. **Besonderheit**: Sind bereits Filter aktiv, bleiben diese natürlich aktiviert, wird der „Alle Effekte“ Button aber deaktiviert, sollen die *zuvor aktiven Filter weiterhin aktiv* bleiben!
- Der Button „**Random**“ aktiviert zufällige Effektkonstellationen.
- **Beliebiges Bild laden.** Über den entsprechenden Eintrag im Menü soll eine Bild-Datei geöffnet und anschließend in der GUI angezeigt werden.

**Ergänzen Sie ggf. das Archiv von Aufgabe 1 und vermerken bitte in einer separaten Datei, welche Funktionalitäten Sie umgesetzt haben.**

Nutzen Sie außerdem Javadoc (https://docs.oracle.com/javase/1.5.0/docs/guide/javadoc/index.html, http://www.java-doc.de/), um Ihren Code sinnvoll zu kommentieren und eine eigene API generieren zu können! Lesen Sie sich Kapitel 6 (S.61 -76) im Buch „Weniger schlecht programmieren“ von K. Passig & J. Jander (2013) durch, um die Best Practices zum Thema Kommentare kennenzulernen.

**Geben Sie die erstellten Dateien gepackt (zip, kein .rar) mit dem Namen „aufgabe1.zip“ ab. Halten Sie sich dabei unbedingt an das vorgegebene Template auf der Website (bzgl. Dateinamen, Ordnerstruktur, …) – es werden ausschließlich lauffähige Lösungen im korrekten Format bewertet!**

**Abgabe:** Packen Sie alle Dateien dieses Übungsblatts in eine Datei („blatt2.zip“) und geben Sie diese bis spätestens

**11.06.2017, 23:59 Uhr (MESZ)**

über UniWorX ab. Eine spätere Abgabe oder eine Abgabe per E-Mail ist nicht möglich!

Bei Problemen oder Fragen können die Tutoren oder die Übungsleitung kontaktiert werden. Gerne beantworten wir Fragen in unserem Slack Channel (https://mimuc.slack.com/messages/mt-ss17)