Praktikum zum Programmierkurs (WI), Wintersemester 2021/2022

Aufgabe 11 (GUI, Bearbeitungszeit: 2 Wochen)

Abgabe zur Erlangung der Bonuspunkte:

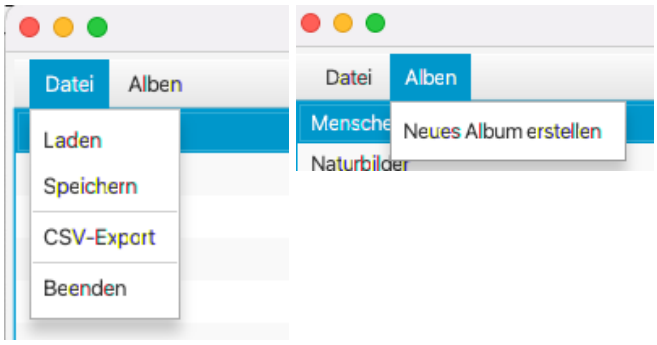
- Frist für die Abgabe: **28.01.2022 20 Uhr**
- Bis zu **6 Bonuspunkte**
- Bewertet werden die Ergebnisse der **Praktika 10 und 11**

Bitte beachten Sie die weiteren Hinweise unten im Abschnitt "5. Abgabe".

Hinweis zum entstehenden Code: *Verwenden Sie für den entstehenden Code weiterhin das in Praktikum 1 in GitLab angelegte Projekt.*

1. Menü

Erweitern Sie das Hauptfenster Ihrer Anwendung (siehe Praktikum 10, Aufgabe 3) um eine Menüleiste. Realisieren Sie die Menüs mit folgendem Aufbau und folgenden Einträgen:



2. Ereignisbehandlung für das Menü

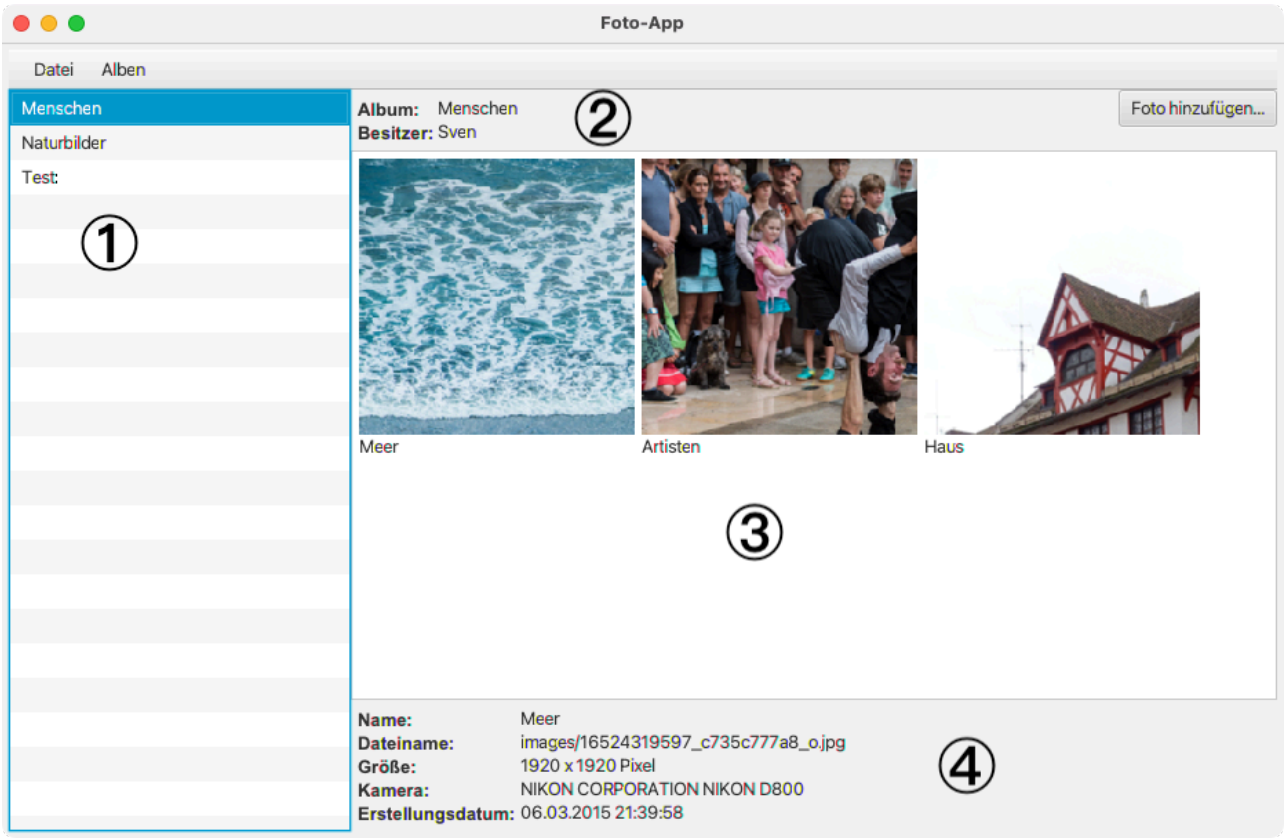
1. Versehen Sie die Menüleiste nun mit einer entsprechenden Ereignisbehandlung. Über die Menüeinträge sollen die Funktionen der Foto-App aufgerufen werden können, die Sie bereits in den vorherigen Praktika implementiert haben:

    - *Laden*: Lädt eine gespeicherte Menge von Alben und zugehörigen Fotos aus einer Datei.
    - *Speichern*: Speichert die aktuelle Menge von Alben und zugehörigen Fotos in eine Datei.
    - *CSV-Export*: Exportiert die aktuelle Menge von Alben und zugehörigen Fotos als CSV-Datei.
    - *Beenden*: Beendet die Anwendung.
    - *Neues Album erstellen*: Ein neues Album wird angelegt. Die Eingabe der Daten erfolgt über eine Instanz der Klasse `AlbumErfassungView`, die Sie in Praktikum 10 umgesetzt haben.

2. Verwenden Sie bei allen Funktionen zum Anzeigen von Nachrichten, Fehlermeldungen, Nachfragen etc. nun entsprechende JavaFX-Dialoge (statt `JOptionPane` wie beim bisherigen "Konsolenmenü"). Verwenden Sie dazu die bereits aus Praktikum 10 bekannte Hilfsklasse `DialogUtil`.

3. Hauptfenster

Implementieren Sie nun das Hauptfenster, so dass es folgenden Aufbau hat:



Realisieren Sie das Hauptfenster unter Verwendung passender Layout-Manager (*keine* absolute Positionierung!) wie folgt:

1. Auf der linken Seite (①) des Hauptfensters sollen die verfügbaren Alben als Liste angezeigt werden. Die Liste soll stets die aktuellen Daten aus der `FotoVerwaltung` zeigen. Insbesondere soll sich die Liste aktualisieren, sobald ein neues Album erstellt wird oder sobald Alben aus einer Datei geladen werden (vgl. Menüpunkte *Neues Album erstellen* und *Laden* aus Aufgabe 2).

   Hinweis: Um wie in obiger Abbildung eine gut lesbare Anzeige der Alben in der Liste zu erhalten, empfiehlt es sich, die `toString`-Methode der Klasse `Album` entsprechend anzupassen.

2. Sobald ein Album in der Liste (①) selektiert wird (z.B. durch Mausklick auf den entsprechenden Listeneintrag), sollen auf der rechten Seite des Hauptfensters die Details zu diesem Album angezeigt werden. Die Detailanzeige zu einem Album gliedert sich in drei Bereiche:

- einen *Kopfbereich* (②),
- einer *Galerie der Fotos* (③) zum Album, und
- bei Selektion eines Fotos in der Galerie einen *Bereich mit den Informationen des Fotos* (④).

Die drei Bereiche werden in den folgenden Punkten näher beschrieben.

Hinweis: Um auf die Selektion in der Liste zu reagieren, wird Ihnen folgender Code weiterhelfen:

```
listView.getSelectionModel().selectedItemProperty()
        .addListener(new ChangeListener<T>() {
    @Override
    public void changed(ObservableValue<? extends T> observable,
                        T oldValue, T newValue) {
        // Event behandeln
    }
});
```

3. Der *Kopfbereich* (②) zeigt den Namen und den Besitzer des ausgewählten Albums an. Zudem enthält der Bereich einen Button "Foto hinzufügen...", über welchen es möglich sein soll, ein neues Foto zum ausgewählten Album hinzuzufügen. Die Eingabe der Daten zum Foto erfolgt über eine Instanz der Klasse `FotoErfassungView`, die Sie in Praktikum 10 umgesetzt haben.

4. Die *Galerie der Fotos* (③) zeigt alle Fotos an, welche im ausgewählten Album enthalten sind. Dabei werden zur besseren Übersicht gleich große Miniaturen (*Thumbnails*) der Fotos angezeigt. Unter jeder Miniatur wird der Name des jeweiligen Fotos angezeigt. Die Galerie soll sich automatisch aktualisieren, sobald ein neues Foto hinzugefügt wird (siehe Punkt 3).

   Hinweis 1: Um eine Miniatur (*Thumbnail*) zu einer Bilddatei zu erzeugen, wird Ihnen folgender Code weiterhelfen:

   ```
   private ImageView createThumbnail(String imageFile, int width) throws FileNotF
   oundException {
       ImageView iv = new ImageView(new Image(new FileInputStream(new File(imageFil
                                              width * 2, 0, true, true));
       iv.setViewport(new Rectangle2D(0, 0, width, width));
       return iv;
   }
   ```

   Das resultierende `ImageView`-Objekt ist ein `Node`, welchen Sie wie ein Control in einem Scene-Graph platzieren können.

   Hinweis 2: Um auch die Anzeige einer größeren Menge von Fotos zu ermöglichen, ist die Verwendung eines ScrollPanes für die Galerie empfohlen.

5. Bei Klick auf ein Foto in der Galerie (③) sollen in Bereich ④ die Informationen (Name, Dateiname, Größe, Kamera, Erstellungsdatum) zum ausgewählten Foto angezeigt werden. Bereich ④ soll nur

sichtbar sein, wenn ein Foto in der Galerie ausgewählt wurde.

Hinweis: Um auf einen Klick auf ein Foto reagieren zu können, wird Ihnen folgender Code weiterhelfen:

```
imageView.setOnMouseClicked(new EventHandler<Event>() {
    @Override
    public void handle(Event evt) {
        // Event behandeln
    }
});
```

Weitere Tipps (optional):

1. Sie können in JavaFX das Aussehen vieler Komponenten mittels CSS (Cascading Style Sheets) anpassen, z.B.:

   ```
   var fp = new FlowPane();
   // Hintergrundfarbe auf weiß setzen
   fp.setStyle("-fx-background-color: WHITE");
   // einen dickeren Rand setzen
   fp.setStyle("-fx-border-width: 1");
   ```

   Mehr dazu erfahren Sie im JavaFX CSS Reference Guide.

2. Um ein Label mit fettgedrucktem Text zu erzeugen, wird Ihnen folgender Code weiterhelfen:

   ```
   private Label createBoldLabel(String text) {
       var lbl = new Label(text);
       lbl.setFont(Font.font("Arial", FontWeight.BOLD, Font.getDefault().getSize()))
       return lbl;
   }
   ```

4. Commit und Push

1. Schreiben Sie den entstandenden Code per Commit in Ihrem lokalen Repository fest. Verwenden Sie als Commit-Message "Aufgabe 11: GUI".

2. Bringen Sie die Änderungen dann per Push auf den GitLab-Server. Kontrollieren Sie in GitLab, dass Ihre Änderungen angekommen sind.

5. Abgabe

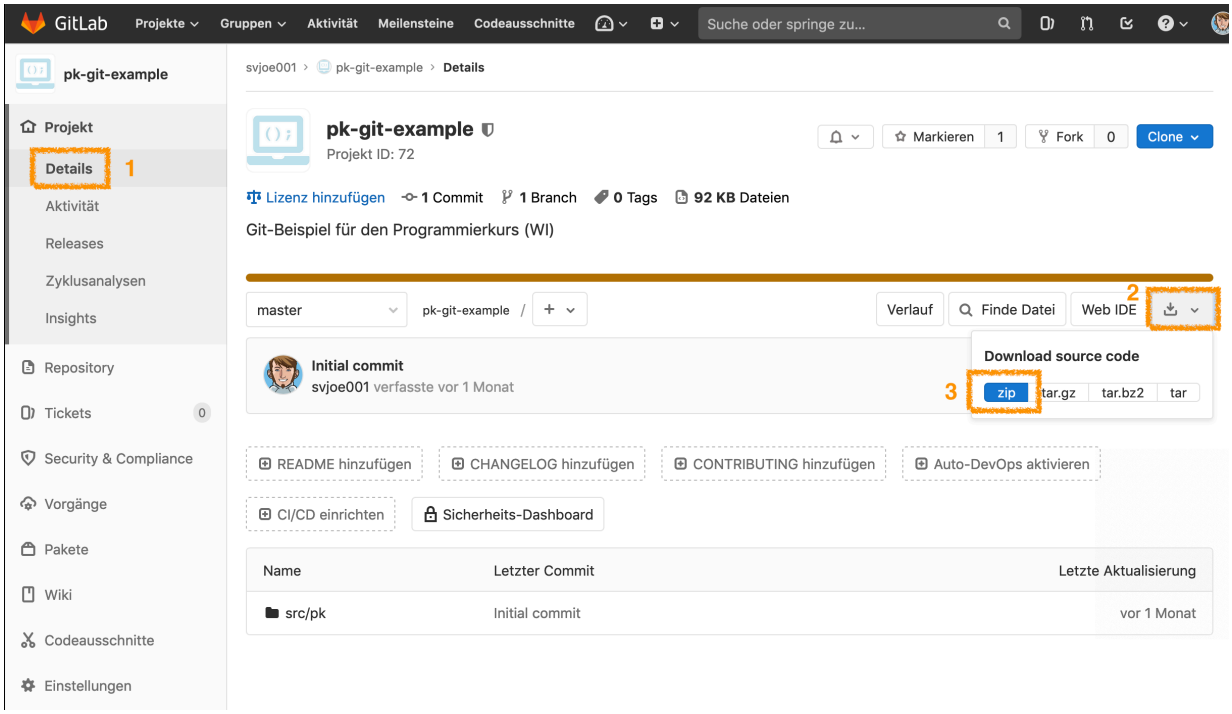
Wichtig: Die folgenden Schritte müssen pro Team nur einmal durchgeführt werden. Es wird also pro

Team nur eine Abgabe in SmartAssign hochgeladen!

Gehen Sie zum Durchführen der Abgabe folgendermaßen vor:

1. Fügen Sie Ihrem Projekt auf oberster Ebene eine Datei `readme.txt` hinzu. Notieren Sie in dieser Datei die *Matrikelnummern und Namen aller Team-Mitglieder*. Fügen Sie die Datei dann per Commit und Push in Ihrem Git-Repository zu. Kontrollieren Sie ggf. auf GitLab, ob die Datei auch wirklich da ist. **Dieser Schritt ist extrem wichtig, da wir auf dieser Basis die Bonuspunkte buchen!**

2. Öffnen Sie Ihr Projekt auf GitLab. Wählen Sie im Bereich "Details" (1) die Download-Schaltfläche (2) und dort die Schaltfläche "zip" (3).



   Es wird nun eine ZIP-Datei auf Ihren Rechner heruntergeladen, die den aktuellen Stand Ihres Projektes enthält.

3. **Kontrollieren Sie, dass die ZIP-Datei alle für die Abgabe relevanten Dateien enthält** (durch Hineinschauen bzw. Entpacken der ZIP-Datei). Es sollte der Quellcode zu den Praktika 10 und 11 sowie die Datei `readme.txt` enthalten sein.

4. Zudem sollte die ZIP-Datei nicht größer als 16 MB sein, da sonst Probleme beim Upload in SmartAssign zu erwarten sind.

5. **Kontrollieren Sie weiterhin, dass der in der ZIP-Datei enthaltene Quellcode vollständig kompilierbar ist** (z.B. durch testweises Einbinden in Eclipse über *New Project → Java Project*). **Eine nicht kompilierbare Abgabe führt zu starkem Punktabzug!**

6. Laden Sie die ZIP-Datei bis zum **28.01.2022 20 Uhr** in SmartAssign (http://smartassign.inf.fh-dortmund.de/) hoch.

Wichtige Hinweise zur Abgabe:

- Laden Sie Ihre Ergebnisse unbedingt bis zum **28.01.2022 20 Uhr** hoch. Eine pünktliche Abgabe ist Voraussetzung zum Erlangen der Bonuspunkte - eine spätere Abgabe ist nicht möglich!
- Laden Sie pro Team nur **eine Abgabe** hoch.
- Sie können mit dieser Abgabe bis zu **6 Bonuspunkte** erlangen. Bewertet werden die Ergebnisse der **Praktika 10 und 11**.
- Kopierte Lösungen werden von uns als Täuschungsversuch eingestuft und mit 0 Bonuspunkten bewertet.