Praktikum zum Programmierkurs (WI), Wintersemester 2021/2022

Aufgabe 10 (JavaFX: Eingabefenster)

Hinweis zu Bonuspunkten: *Zur Vergabe der Bonuspunkte werden wir in regelmäßigen Abständen den von Ihnen produzierten Quellcode begutachten. Informationen darüber, wann wir welche Praktikumsaufgaben begutachten, wie viele Bonuspunkte es gibt und wie Sie Ihren Quellcodestand einreichen, werden rechtzeitig bekanntgegeben.*

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

1. Vorbereitung

1. Da JavaFX eine separate Klassenbibliothek ist, müssen Sie dieses zunächst installieren. Laden Sie das für Ihr Betriebssystem passende SDK (*Public version*) hier herunter und installieren Sie es: https://gluonhq.com/products/javafx/.

   Hinweis: Für das Praktikum empfehlen wir die Nutzung von Java 11 und JavaFX 11.

2. Folgen Sie nun der Anleitung im Abschnitt "5. Anleitung: JavaFX in Eclipse einbinden" unten auf dem Praktikumsblatt, um JavaFX in Eclipse einzubinden.

3. **Wichtig:** Passen Sie Ihre `module-info.java` wie folgt an:

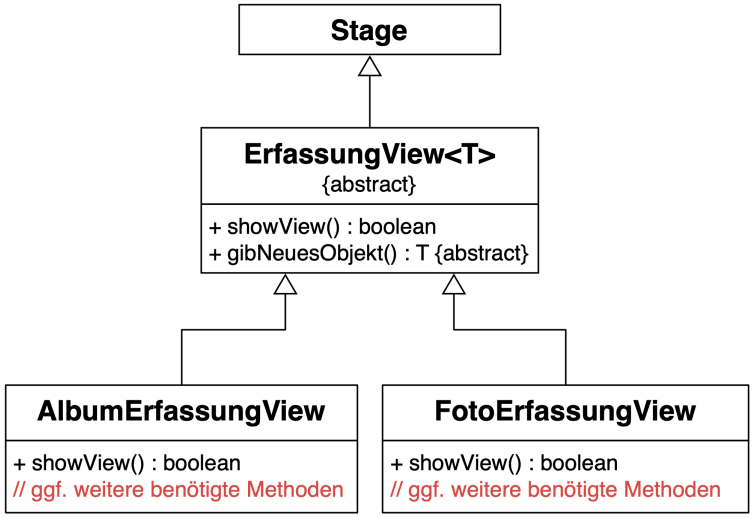
```
module pk {
    requires java.desktop;
    requires javafx.controls;
    requires javafx.graphics;

    requires metadata.extractor;
    requires xmpcore;

    opens pk.foto.ui to javafx.graphics;
}
```

2. Eingabefenster für Alben und Fotos

Als ersten Schritt zur Umsetzung einer GUI für unsere Foto-App realisieren wir nun Eingabefenster zur Erstellung neuer Alben und Fotos. Setzen Sie die Eingabefenster gemäß folgendem Klassendiagramm um:



Gehen Sie folgendermaßen vor:

1. Erstellen Sie in Ihrem Projekt ein neues (Unter-)Paket `pk.foto.ui`, in welchem alle Klassen für die GUI abgelegt werden sollen.

2. Implementieren Sie, wie im Klassendiagramm zu sehen, für jedes Eingabefenster eine eigene Klasse. Vermeiden Sie redundanten Code durch die abstrakte Oberklasse `ErfassungView<T>`: Diese soll *alle Gemeinsamkeiten* der beiden Eingabefenster enthalten.

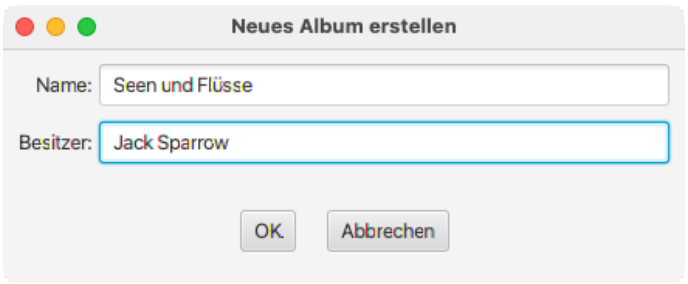
3. Die Eingabefenster sollen modale Dialoge sein (d.h. Dialoge, die im Vordergrund dargestellt werden und für die Dauer der Anzeige alle anderen Fenster für die Eingabe sperren). Implementieren Sie zu diesem Zweck für jedes Eingabefenster einen Konstruktor mit einem Parameter: Ein Objekt vom Typ `Stage`, welches das übergeordnete Fenster für den Dialog angibt (z.B. das Hauptfenster - die `primaryStage`).

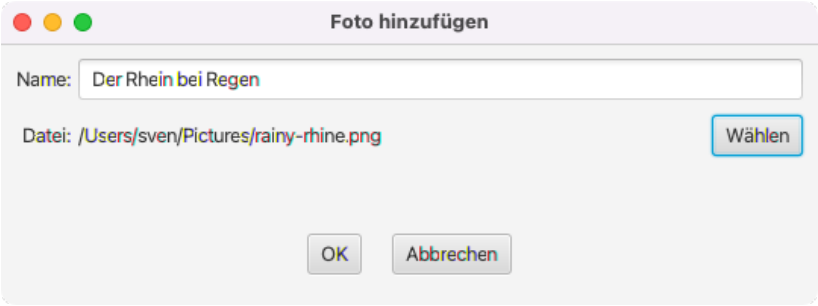
   Hinweis 1: Ist `stage` das im Konstruktor übergebene übergeordnete Fenster, so können Sie über folgenden Code einen modalen Dialog erzeugen:

   ```
   this.initOwner(stage);
   this.initModality(Modality.WINDOW_MODAL);
   ```

   Hinweis 2: Halten Sie die Implementierung der Konstruktoren möglichst knapp. Ein Eingabefenster soll erst dann angezeigt werden, wenn die Methode `showView` (siehe 6.) aufgerufen wird.

4. Realisieren Sie die Eingabefenster wie in den folgenden Abbildungen dargestellt:





Verwenden Sie passende Layout-Manager zur Positionierung der Controls (*keine* absolute Positionierung!).

5. Der `FotoErfassungView` soll die Möglichkeit bieten, eine Foto-Datei auszuwählen. Zu diesem Zweck soll eine Kombination aus zwei Controls eingesetzt werden (in der obigen Abbildung zu sehen hinter dem Label mit dem Text "Datei:"):

    1. Ein Label, welches den Dateipfad zur gewählten Datei anzeigt (in der obigen Abbildung "/Users/sven/Pictures/rainy-rhine.png"). Wurde noch keine Datei ausgewählt, so soll das Label die Beschriftung "keine Datei ausgewählt" tragen.

    2. Ein Button mit der Aufschrift "Wählen". Wird dieser Button geklickt, so soll ein Dateiauswahldialog geöffnet werden. Verwenden Sie dazu die Methode `showFileChooser` aus der Hilfsklasse `DialogUtil`, die Sie in ILIAS zum Download finden. Nach Auswahl einer Datei soll der entsprechende Dateipfad im Label (→ 5.1) angezeigt werden.

6. Die Methode `showView` soll die Anzeige des entsprechenden Eingabefensters bewirken. Berücksichtigen Sie folgende Anforderungen bei der Implementierung:

    1. Der von `showView` zurückgelieferte `boolean`-Wert zeigt an, ob die BenutzerIn die Eingabe mit "OK" bestätigt (Rückgabewert= `true`) oder mit "Abbrechen" abgebrochen hat (Rückgabewert= `false`).

    2. Nutzen Sie die Methode `showAndWait` der Klasse `Stage`, damit die Ausführung Ihres Codes wartet, bis das Eingabefenster durch die BenutzerIn geschlossen wurde (durch Klick auf "OK" oder "Abbrechen").

7. Sowohl bei Klick auf den Button "OK" als auch bei Klick auf den Button "Abbrechen" soll das jeweilige Eingabefenster geschlossen werden (Methode `close` der Klasse `Stage`). Wurde das Eingabefenster mit "OK" geschlossen, soll es danach möglich sein, durch Aufruf der Methode `gibNeuesObjekt` (siehe Klassendiagramm oben) das soeben neu erstellte Objekt zu erhalten.

    Beispielablauf: Eine Instanz des `AlbumErfassungView`s wird über einen Aufruf von `showView` geöffnet. Die BenutzerIn füllt die Eingabefelder wie oben dargestellt aus und klickt auf "OK". Der `AlbumErfassungView` wird daraufhin geschlossen. Wird dann die Methode `gibNeuesObjekt` des `AlbumErfassungView`s aufgerufen, gibt diese ein `Album`-Objekt mit den eingegebenen Werten zurück (gemäß obiger Darstellung: name="Seen und Flüsse", besitzer="Jack Sparrow").

3. JavaFX-Anwendung & Testen

Erstellen Sie nun eine JavaFX-Anwendung (mittels `javafx.application.Application`) in einer neuen Klasse `FotoUI`. Diese JavaFX-Anwendung soll zunächst ein leeres Fenster als Hauptfenster öffnen (Titel: "Foto-App", Größe: 800*500 Pixel). Danach soll direkt eine Instanz einer `AlbumErfassungView` geöffnet werden. Wird diese geschlossen, soll eine `FotoErfassungView` geöffnet werden.

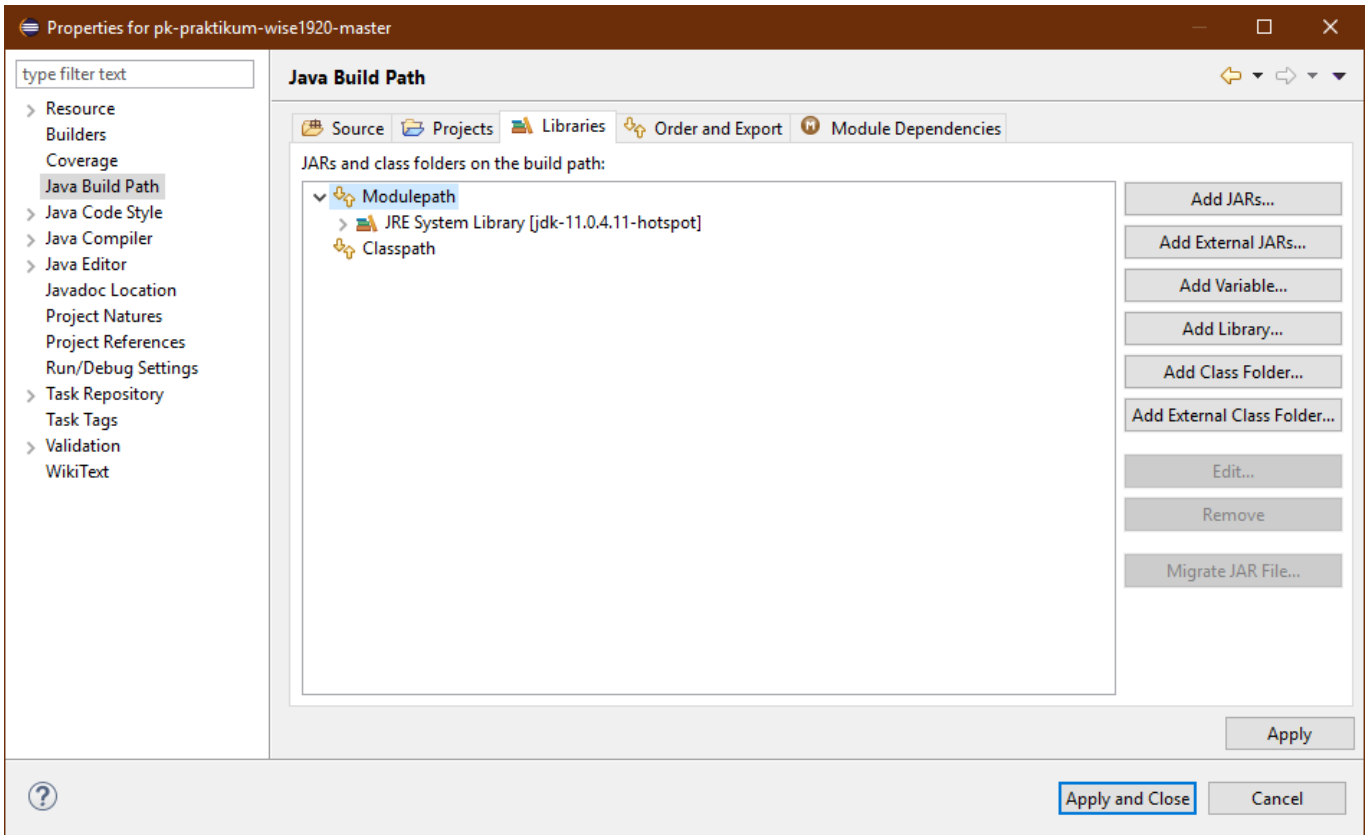
Wird die `AlbumErfassungView` und/oder die `FotoErfassungView` durch die BenutzerIn mittels "OK" geschlossen, so soll die Klasse `FotoUI` sich das jeweils erstellte `Album` bzw. `Foto` holen und dieses auf der Konsole ausgeben. Bei Klick auf "Abbrechen" erfolgt keine Ausgabe.

4. Commit und Push

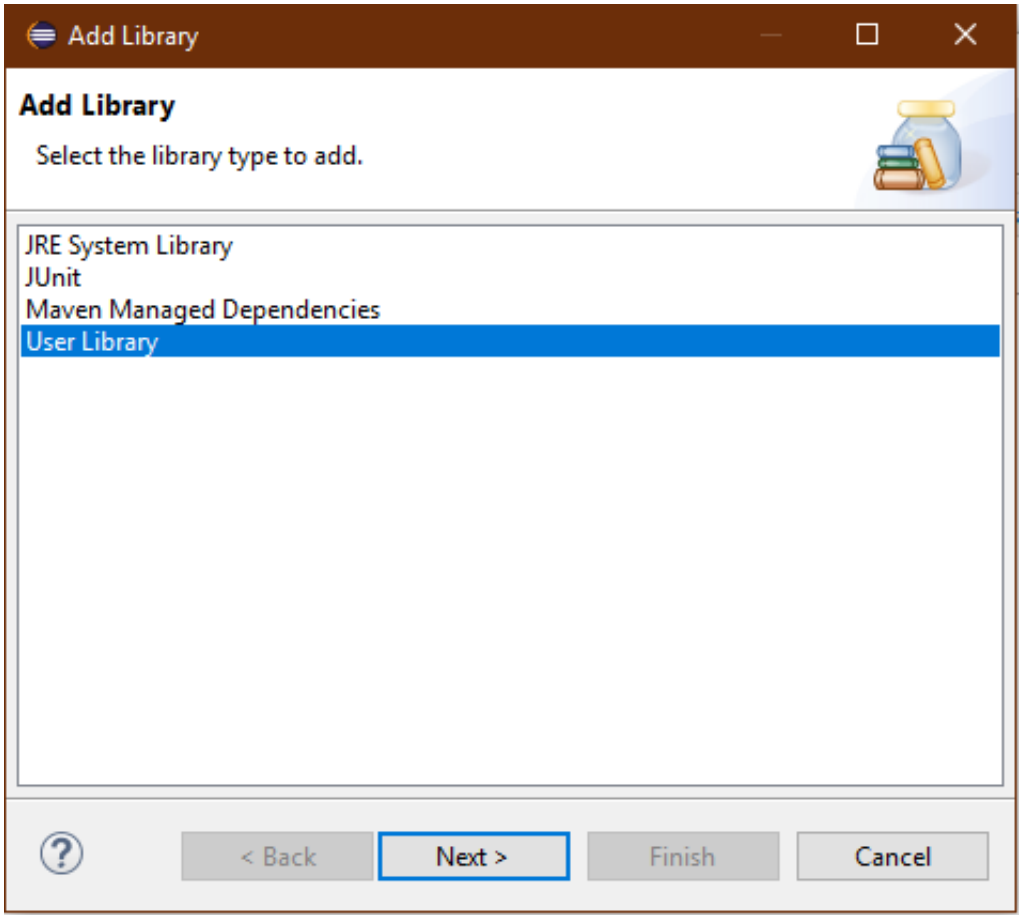
1. Schreiben Sie den entstandenden Code per Commit in Ihrem lokalen Repository fest. Verwenden Sie als Commit-Message "Aufgabe 10: Eingabefenster".
2. Bringen Sie die Änderungen dann per Push auf den GitLab-Server. Kontrollieren Sie in GitLab, dass Ihre Änderungen angekommen sind.

5. Anleitung: JavaFX in Eclipse einbinden

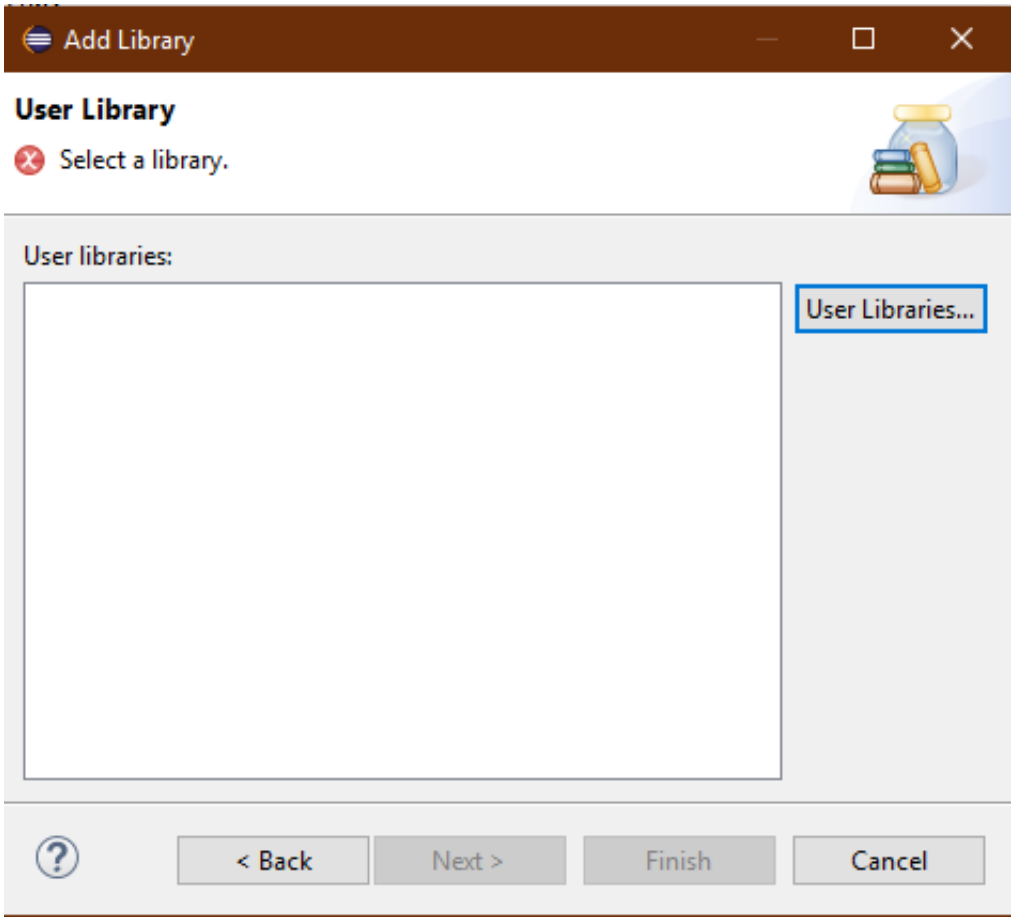
1. Öffnen Sie die Projekteinstellungen über einen Rechtsklick auf Ihr Projekt und Auswahl des Menüpunktes *Properties*. Wählen Sie im folgenden Dialog links den Punkt *Java Build Path* aus und wechseln Sie dann rechts auf den Reiter *Libraries*:



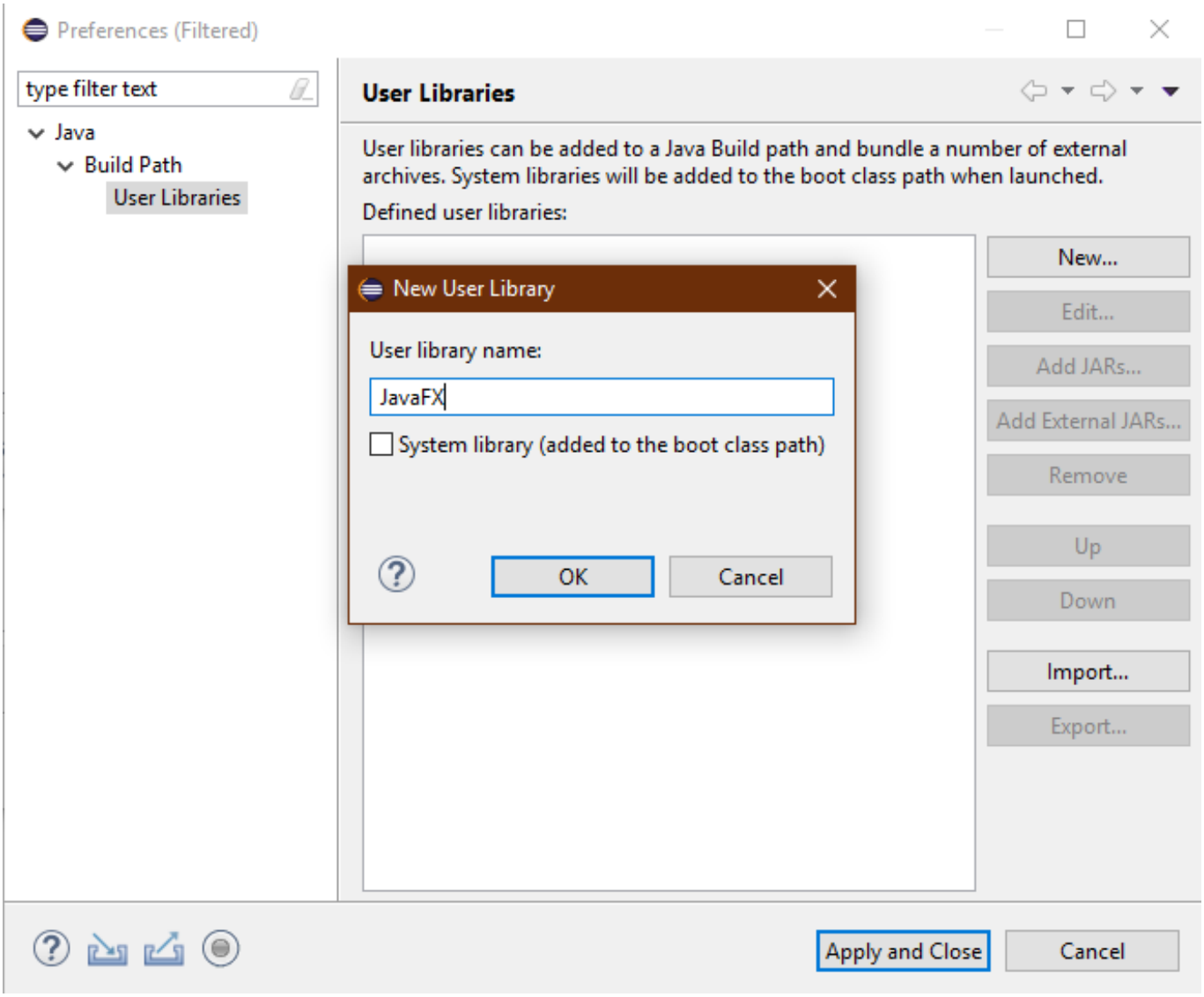
2. Wählen Sie *Modulepath* aus und klicken Sie dann rechts auf *Add Library...*. Wählen Sie im sich öffnenden Dialog den Eintrag *User Library* und bestätigen Sie dies mit Klick auf *Next >*:



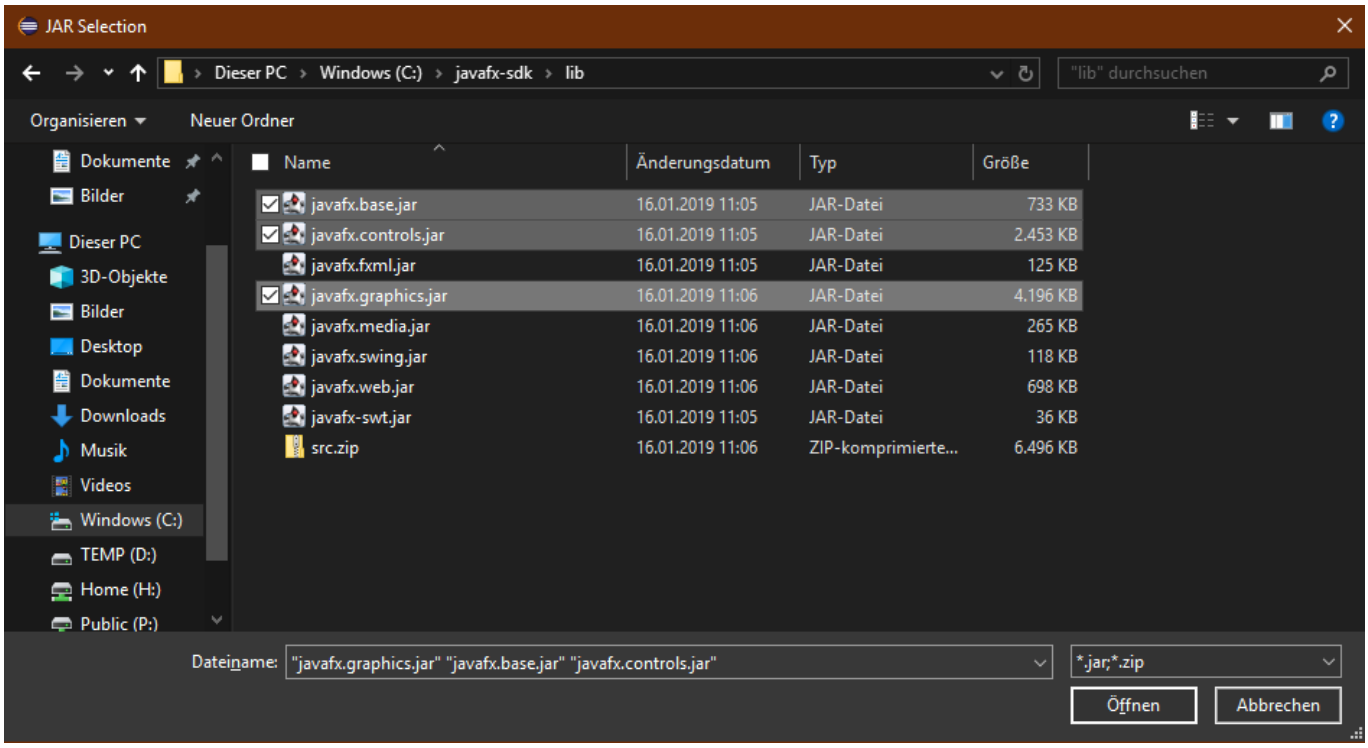
3. Wählen Sie im nächsten Dialog die Schaltfläche *User Libraries...*:

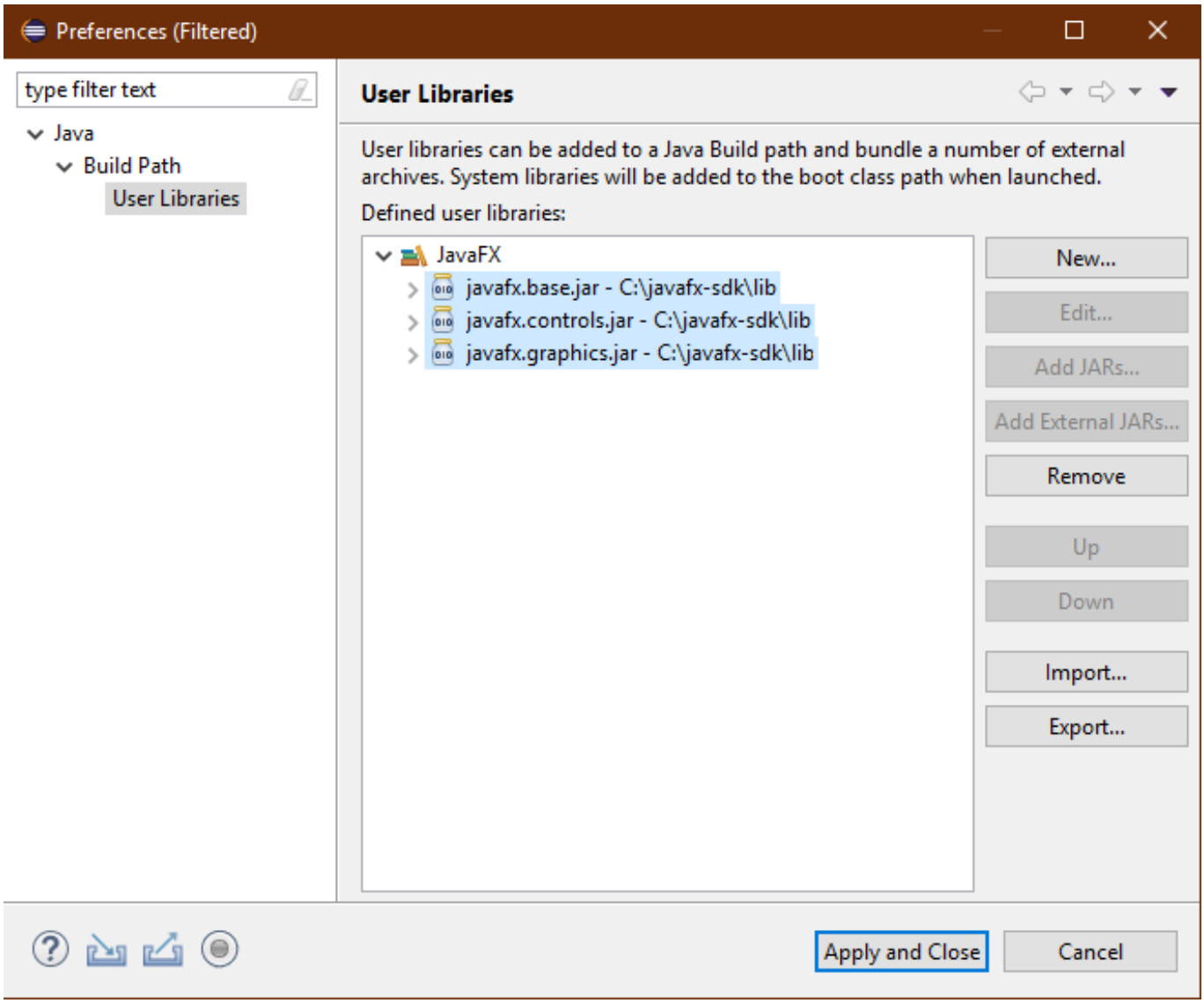


4. Legen Sie über Klick auf *New...* eine neue User-Library mit dem Namen "JavaFX" an. Bestätigen Sie die Eingabe des Namens mit Klick auf *OK*.



5. Fügen Sie der neu angelegten User-Library nun über *Add External JARs...* die notwendigen Bibliotheken hinzu. Navigieren Sie dazu im folgenden Dateiauswahldialog zu dem Verzeichnis, in welches Sie JavaFX installiert haben und dort in das *lib*-Verzeichnis (bei den Laborrechnern ist dies *C:\javafx-sdk\lib*). Wählen Sie die Dateien *javafx.base.jar*, *javafx.controls.jar* und *javafx.graphics.jar* und bestätigen Sie die Auswahl mit *Öffnen*.





6. Bestätigen Sie die noch offenen Dialoge mit *Apply and Close*, *Finish* und *Apply and Close*.