Praktikum zum Programmierkurs (WI), Wintersemester 2021/2022

Aufgabe 4 (Collections, Menü)

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. Arrays durch Collections ersetzen

Wir nutzen nun Collections statt Arrays, um unsere Implementierung flexibler zu gestalten. Gehen Sie wie folgt vor:

1. `FotoVerwaltung` umbauen:
    - Benennen Sie die bestehende `FotoVerwaltung`-Klasse, die noch ein Array zur Verwaltung der Alben verwendet, um in `FotoVerwaltungArray`.
    - Erstellen Sie eine neue `FotoVerwaltung`-Klasse (weiterhin im Package `pk.foto`), die zur Verwaltung der Einträge eine *typisierte Liste* verwendet. Wählen Sie eine passende Liste für diesen Zweck aus. *Eine Kapazität bzw. initiale Anzahl an Einträgen benötigen Sie bei dieser Lösung nicht mehr.*
    - Die neue `FotoVerwaltung`-Klasse soll die gleichen Methoden bieten wie die Variante mit dem Array. **Insbesondere sollen sich die Signaturen (Name, Parameter) sowie die Rückgabetypen der Methoden nicht verändern!** Passen Sie die Implementierung der einzelnen Methoden an die neue Liste an. Verwenden Sie bei der Implementierung mindestens einmal einen `Iterator` und mindestens einmal eine erweiterte For-Schleife.
    - Sorgen Sie dafür, dass in Ihrer Java-Anwendung (Klasse `Tester`) weiterhin alle verfügbaren Methoden der (neuen) `FotoVerwaltung`-Klasse mindestens einmal aufgerufen werden und weiterhin funktionieren.

    Hinweis: In den folgenden Praktika werden wir mit der neuen `FotoVerwaltung`-Klasse weiterarbeiten. Wir behalten die `FotoVerwaltungArray`-Klasse zwar, entwickeln diese im Folgenden jedoch nicht weiter.

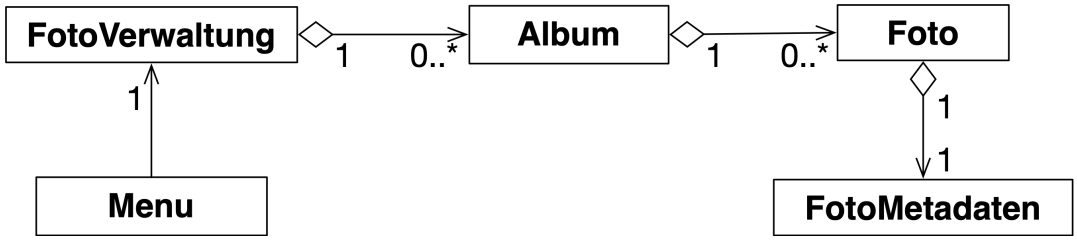
2. `Album` umbauen:

   Bauen Sie auch die Assoziation `fotos` in der Klasse `Album` so um, dass diese nicht mehr mit einem Array arbeitet, sondern mit einer geeigneten *typisierte Liste*. Passen Sie analog zu 1. die Methoden der Klasse `Album` geeignet an, ohne die Signaturen und Rückgabewerte zu ändern! *Eine Kapazität bzw. initiale Anzahl an Einträgen benötigen Sie bei dieser Lösung nicht mehr.*

   Hinweis: Im Gegensatz zu 1. können Sie die `Album` -Klasse direkt umbauen, ohne vorher eine `AlbumArray` -Variante zu erstellen.

2. Menü

Erstellen Sie die Klasse `Menu` :



Diese Klasse soll eine erste einfache Interaktion mit den AnwenderInnen realisieren. AnwenderInnen bekommen auf der Konsole folgendes Menü angezeigt, aus welchem sie eine Aktion auswählen können:

```
Foto-App

  1. Album hinzufügen
  2. Drucke alle Alben
  3. Drucke Album mit Name
  4. Beenden

  Bitte Aktion wählen:
```

Berücksichtigen Sie bei der Implementierung folgende Punkte:

- Eine Aktion soll durch Eingabe der entsprechenden Zahl ausgewählt werden können (also z.B. `3` für `Drucke Album mit Name`). Nach Eingabe der Zahl wird die entsprechende Aktion ausgeführt. Realisieren Sie die Eingabe der Zahl mit Hilfe der Klasse `Scanner` (siehe unten für Benutzungshinweise).

- Folgende Aktionen sollen möglich sein:

  1. `Album hinzufügen` :

     - Legt ein neues Album an und fügt dieses in der `FotoVerwaltung` hinzu. Die benötigten Daten (Name, Besitzer, Fotos) sollen AnwenderInnen über Eingabedialoge

(`JOptionPane` , siehe unten für Benutzungshinweise) eingeben können.
    - Insbesondere sollen AnwenderInnen bei der Erstellung des Albums beliebig viele Fotos eingeben können. Dazu soll die Anwendung nach Eingabe eines Fotos fragen, ob ein weiteres Foto eingegeben werden soll. Ermitteln Sie mit Hilfe der API-Dokumentation zu `JOptionPane` eine geeignete Möglichkeit, einen solchen Bestätigungsdialog (*confirm dialog*) anzuzeigen. Antwortet die BenutzerIn mit "Ja", so kann ein weiteres Foto eingegeben werden. Lautet die Antwort "Nein", so wird die Erstellung des Albums abgeschlossen.

      Hinweis: Der Einfachheit halber soll es für die Erstellung eines Fotos zunächst ausreichen, Name und Dateiname anzugeben. Die `FotoMetadaten` können Sie zunächst mit Standardwerten anlegen - wir kommen in einem späteren Praktikum auf die Metadaten zurück.

2. `Drucke alle Alben` : Gibt alle in der `FotoVerwaltung` enthaltenen Alben *und deren Fotos* auf der Konsole aus.

3. `Drucke Album mit Name` : Gibt das Album mit dem gegebenen Namen (und die enthaltenen Fotos) auf der Konsole aus. Den gewünschten Namen sollen AnwenderInnen über einen Eingabedialog (`JOptionPane` , siehe unten für Benutzungshinweise) eingeben können.

4. `Beenden` : Beendet die Anwendung.

- Gehen Sie bei der Implementierung aller Eingaben davon aus, dass AnwenderInnen stets nur gültige Werte eingeben (also z.B. gültige Zahlen im Menü etc.). Eine entsprechende Fehlerbehandlung werden wir in einem späteren Praktikum ergänzen.

- Nach Ausführung einer Aktion soll automatisch wieder das Menü angezeigt werden. Ausnahme ist Aktion 4, da diese die Anwendung beendet.

- Sämtliche Ausgaben der Anwendung sollen wie bisher auf der Konsole erfolgen.

Zusatzinformationen zur Lösung:

1. Tastatureingaben über die Konsole mit `Scanner`

- Über die Klasse `java.util.Scanner` können Sie einfache Tastatureingaben auf der Konsole realisieren.

- Zu diesem Zweck müssen Sie zunächst einen Scanner erzeugen (`System.in` ist ein Eingabestrom, über den Eingaben von der Tastatur geliefert werden):
  `Scanner scanner = new Scanner(System.in);`

- `Scanner` kann primitive Datentypen und Strings einlesen (*parsen*). Zu diesem Zweck bietet die Klasse verschiedene Methoden, um die eingebenen Werte zu erhalten, z.B.:

```
// Eingabe als String holen
String eingabe = scanner.next();

// Eingabe als int holen
int intEingabe = scanner.nextInt();
```

- Weitere Informationen erhalten Sie in der API-Dokumentation zu `Scanner`.

2. Einfache Eingabedialoge mit `JOptionPane`

- Die Klasse `javax.swing.JOptionPane` bietet u.A. folgende statische Methode:

```
String showInputDialog(Component parentComponent, Object message)
```

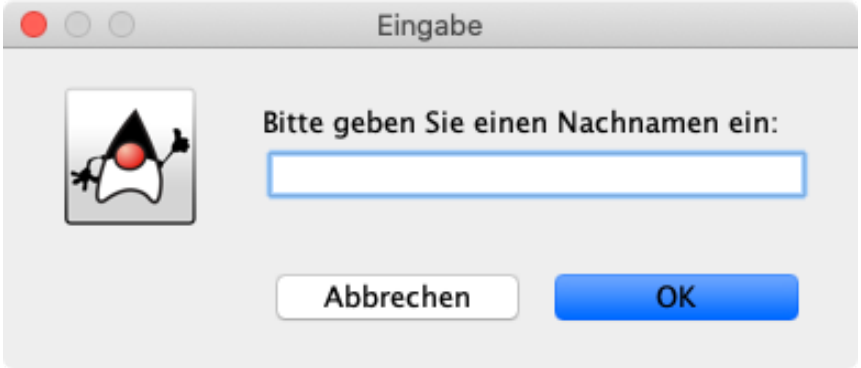
- Diese verwenden wir in diesem Praktikum folgendermaßen:

  1. Über den Parameter `parentComponent` kann man ein übergeordnetes Fenster festlegen. Da wir noch keine grafische Benutzeroberfläche haben, übergeben wir hier `null`.
  2. Über den Parameter `message` geben wir einen String ein, welcher als Text auf dem Eingabedialog angezeigt wird (Eingabeaufforderung).
  3. Die Methode `showInputDialog` liefert die Eingabe stets als String! Es kann also ggf. eine Typkonvertierung notwendig sein.

- Beispiel:

```
// Eingabedialog anzeigen
String nachname =
    JOptionPane.showInputDialog(null, "Bitte geben Sie einen Nachnamen ein: ");
```

  Es wird folgender Dialog dargestellt:



- **Wichtig:** Damit Sie die Klasse `JOptionPane` in Ihrem Quellcode benutzen können, müssen Sie die Datei `module-info.java` folgendermaßen anpassen:

```
module pk {
    requires java.desktop;
}
```

- **Hinweis:** Dieser Eingabedialog von `JOptionPane` ist sehr eingeschränkt und kann lediglich ein Eingabefeld darstellen. Benötigen Sie mehrere Eingaben, so müssen Sie mehrere Eingabedialoge nacheinander verwenden.
- Weitere Informationen erhalten Sie in der API-Dokumentation zu `JOptionPane`.

3. Commit und Push

1. Schreiben Sie den entstandenden Code per Commit in Ihrem lokalen Repository fest. Verwenden Sie als Commit-Message "Aufgabe 4: Collections, Menue".
2. Bringen Sie die Änderungen dann per Push auf den GitLab-Server. Kontrollieren Sie in GitLab, dass Ihre Änderungen angekommen sind.