2. PLF POS 1: Nachtermin Filmedatenbank

4ACIF / 2023-06-21 18:50 - 20:20 / B3.08 / Georg Graf (GRG)

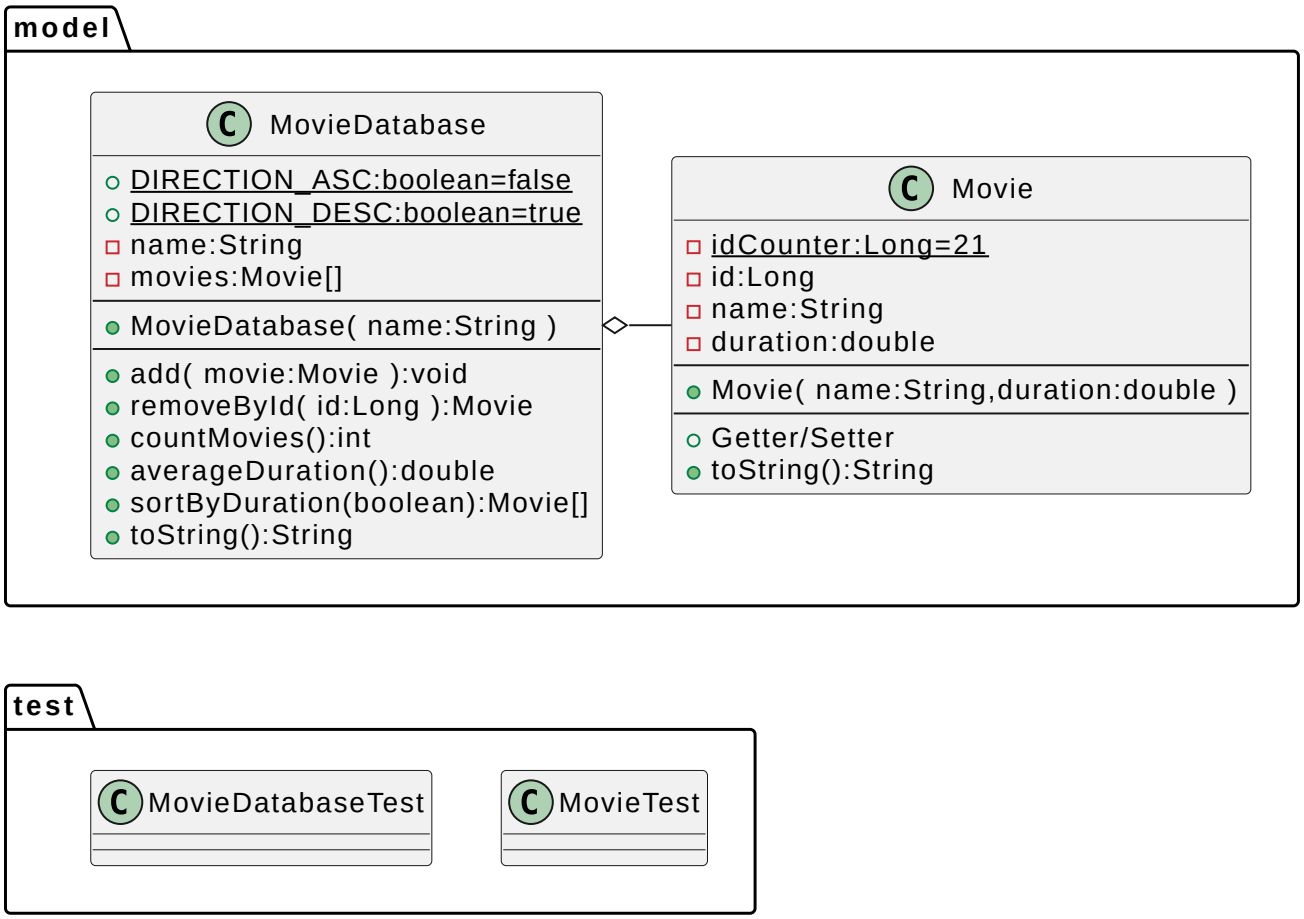
WARNING

- Achten Sie darauf, nur kompilierfähige Programme abzugeben. Sollten Sie einen Compilerfehler nicht beheben können, kommentieren Sie das entsprechende Codestück aus. Nicht kompilierfähige Programme werden negativ bewertet.
- Halten Sie sich exakt an die Vorgaben, Formate sowie Klassen- und Methodennamen.

Beschreibung

Es soll eine Filme-Datenbank entwickelt werden, in welcher Filme verwaltet werden können. Entwickeln Sie die Klassen auf Basis des UML-Diagramms und den weiter unten ausgeführten Anforderungen.

UML-Diagramm



Angabe und Abgabe Verfahren

Angabe

- Auf **Z:** finden Sie das Zipfile der Angabe, dieses auf den Desktop **verschieben**.
- Am Desktop - in den Desktop entpacken - und **die Aufgaben lösen**

Abgabe

- In Intellij mittels `File → Export → Project to Zip File` auf das Laufwerk `Z:\\` abgeben. (`Vorname_Nachname.zip`)

1. Aufgabe - Movie

Implementieren Sie die Model-Klasse `Movie` gemäß UML-Diagramm und folgender Klassenbeschreibung. Sie können die vorhandenen Testklassen für JUnit-Tests verwenden.

Table 1. Klassenbeschreibung

Attribut/Methode	Beschreibung
`idCounter`	• Counter, der für jede erzeugte `Movie`-Instanz eine eindeutige Id vergibt. • Der Counter soll beim Wert 21 starten.
`id`	• Jede `Movie`-Instanz muss vom `idCounter` eine eindeutige Id erhalten, die nach ihrere Zuweisung **nicht** mehr verändert werden darf. Natürlich muß der `idCounter` (statisch) nach jeder ID-Vergabe inkrementiert werden, wie im Unterricht besprochen.
`name`	• Repräsentiert den Namen eines Filmes. • Der Name des Movies darf nicht null sein. • Der Name des Movies muss mindestens 3 Zeichen lang sein. • Werfen Sie pro Fehlerfall eine `IllegalArgumentException` mit einer sprechenden Fehlermeldung.
`duration`	• Dauer des Filmes in Sekunden.

`toString (String)`	• Die Methode gibt einen String über alle Informationen eines Filmes zurück.
	
	3 Beispiele:
	
	`21: Kill Bill: Vol. 1 (6660 sec)`
	`22: From Dusk Till Dawn (6480 sec)`
	`24: Pulp Fiction (9240 sec)`

2. Aufgabe - MovieDatabase

Implementieren Sie die Klasse `MovieDatabase` gemäß UML-Diagramm und folgender Klassenbeschreibung. Sie können die vorhandenen Testklassen für JUnit-Tests verwenden.

Table 2. Klassenbeschreibung

Attribut/Methode	Beschreibung
`name`	Der Name der Filme Datenbank.
`movies`	• Alle Filme sollen im Array `movies` vom Typ `Movie[]` gespeichert werden. • Insgesamt sollen 7 Filme gespeichert werden können. • Achten Sie darauf, dass keine Methode eine Referenz auf das private Array zurückliefert.
`add( movie )` `(void)`	• Die Methode fügt einen neuen Film in die Movie-Database ein. • Ein Film darf nicht doppelt aufgenommen werden. Wenn schon vorhanden, muß eine `IllegalArgumentException` geworfen werden. • `null` ist kein Film und darf nicht aufgenommen werden. In dem Fall soll ebenfalls eine `IllegalArgumentException` fliegen.
`removeById( id )` `(Movie)`	• Die Methode dient zum Entfernen eines Filmes. • Wenn der Film mit der übergebenen `id` in der Movie-DB vorhanden ist, so soll er aus dem Array entfernt werden. Geben Sie das entfernte Objekt zurück. • Ist der Film mit der gesuchten `id` nicht vorhanden, so soll der Wert `null` zurückgegeben werden.
`countMovies()` `(int)`	• Zählt alle in der Movie-DB vorhandenen Filme und liefert den Wert zurück. (Dh. Anzahl aller nicht-null Werte im Array)
`averageDuration()` `(double)`	• retourniert die durchschnittliche Dauer aller Filme in der Datenbank. (Arithmetisches Mittel: Gesamtlänge/Anzahl)
`sortByDuration(boolean)` `(Movie[])`	• Die Methode soll ein **neues** Array mit den sortierten Werten zurückliefern. • Ist der boolean Parameter `DIRECTION_ASC` soll der kürzeste Film am Beginn sein. • Ist der boolean Parameter DIRECTION_DESC` soll der längste Film am Beginn sein. • Alle `null` Werte sollen an das Ende des Arrays sortiert werden. • Das bestehende Array `movies` soll unsortiert bleiben. **Hinweis** Kopieren Sie die Daten vor dem Sortieren zuerst in ein **neues** Array mit der gleichen Größe wie `movies`.

Attribut/Methode	Beschreibung
`toString()` `String`	• Liefert detaillierte Informationen über den Movie-DB zurück. Beispiel: ``` Filme-Datenbank: Quentin Lovers 3 Vorhandene Filme: 21: Kill Bill: Vol. 1 (6660 sec) 22: From Dusk Till Dawn (6480 sec) 24: Pulp Fiction (9240 sec) ``` • Verwenden Sie eine Instanz der Klasse `StringBuilder` um die Informationen der Filme zu sammeln. Lassen Sie sich ggf. von Intellij dabei unterstützen!

3. Aufgabe - JUnit Tests

- Projektordner `s4-plf-001_online_mdb` mit dem Windows Explorer öffnen.
- Datei `abgabe.cmd` ausführen.
- **Z: Laufwerk** öffnen und Abgabe kontrollieren.
- Vom Prüfungsrechner **abmelden**.

Viel Erfolg!!!