

A. Théorie

1. Que signifie l'annotation @Override ?
2. Pourquoi est-ce qu'on a souvent besoin de redéfinir equals et hashCode ?
3. Qu'est ce que le polymorphisme statique ? Donnez un exemple.

B. Pratique

On souhaite modéliser une plateforme de streaming musical. Pour cela, on souhaite connaître quels artistes chantent quelles chansons ? De quel genre de musique ? Issu de quel album ? On souhaite également laisser la possibilité à l'utilisateur de se créer sa propre liste de lecture (playlist) pour pouvoir la partager avec les autres. Pour pouvoir écouter leurs chansons préférées, chaque utilisateur aura une liste contenant ses playlists préférées.

Une playlist est un ensemble de chansons, potentiellement effectuées par des artistes différents, conçu par un utilisateur. Il peut être « aimé (liké) » une seule fois par chaque utilisateur. Si l'utilisateur aime une playlist qu'il aime déjà, cela correspond à l'action de retirer son « like ». Nous considérerons que chaque chanson disposera d'un identifiant, d'un titre, d'une durée(*), ainsi que d'un ou plusieurs genres. Un album est également un ensemble de chansons effectuées par un même artiste, avec un identifiant, un nom, et une date de sortie. Nous définissons un « single » comme étant une chanson qui ne figure dans aucun album.

Les artistes peuvent représenter des groupes ou des artistes solo, et les deux doivent comporter : un identifiant, leur nom de scène, l'année de début, ainsi que la nationalité du groupe / chanteur(se). Chaque artiste solo se verra rajouté en plus de cela : un nom, un prénom, une date de naissance, tandis qu'un groupe est un ensemble d'artistes solo. Créez les classes Java pour implémenter cette plateforme.


Pour ce qui est des méthodes, il doit être possible d'effectuer les actions suivantes :

- addToPlaylist(...) : Ajouter une chanson ou bien un album entier X à une playlist ;
- removeById(...) : Retirer une chanson d'une playlist en utilisant son id ;
- like(...) : Aimer la playlist X et l'ajouter à la liste des playlist favorites d'un utilisateur U, si la playlist est déjà parmi les favoris, il sera retiré à la place ;
- exclude(...) : Retourne une nouvelle playlist à partir d'une playlist existante mais en excluant toutes les chansons d'un ou plusieurs genres fournis en paramètre. Par exemple, je veux bien écouter la playlist de Dr. Toky, mais en excluant tout ce qui est « Métal » ;
- countPlaylist(...) : Retourne le nombre de playlists dans lesquels figure une chanson. Par exemple : « Fly to the skies from your land » de Mustafa figure dans 14 playlists ;
- getTotalLikes(...) : Retourne le nombre de likes total qu'une playlist a reçu.

Nous vous laissons le soin de choisir dans quelles classes les placer, et quels sont les paramètres nécessaires.

(*) Exemple de durée, avec la classe Duration de Java.

java

 Copier le code

```
import java.time.Duration;

public class SongDurationExample {
    public static void main(String[] args) {
        // Creating a duration of 3 minutes and 45 seconds
        Duration duration = Duration.ofMinutes(3).plusSeconds(45);

        // Display the duration
        System.out.println("Duration: " + duration);

        // Getting minutes and seconds from the duration
        long minutes = duration.toMinutesPart();
        long seconds = duration.toSecondsPart();

        // Display minutes and seconds separately
        System.out.printf("Minutes: %d, Seconds: %d\n", minutes, seconds);
    }
}
```