jMusicHub

1. Exigences

On vous demande de faire une application Java qui gère des éléments musicaux. Un élément musical peut être une Chanson ou un LivreAudio.

Une Chanson a: un Titre, un Artiste, une Durée [secondes], un ID [identifiant unique], un Contenu [fichier audio], un Genre.

Un Album contient plusieurs Chansons. Un Album a aussi: un Titre, un Artiste, une Durée [secondes], une Date (de sortie), un ID [identifiant unique]. Les Chansons sont rangées toujours dans le même ordre mais on peut y avoir accès de manière aléatoire (par exemple, pour la lecture du contenu).

Les Genres sont: Jazz, Classique, Hip-Hop, Rock, Pop, Rap.

Un LivreAudio a: un Titre, un Auteur et une Durée [secondes], un ID [identifiant unique], un Contenu [fichier audio], une Langue et une Catégorie.

Les Catégories sont : Jeunesse, Roman, Théâtre, Discours, Documentaire

Les Langues sont : Français, Anglais, Italien, Espagnol, Allemand

On peut créer des Playlists. Une Playlist a un Nom, un ID (identifiant unique) et elle contient plusieurs Chansons et LivreAudios.

Au démarrage, l'application lit depuis des fichiers CSV, ou autre:

- les albums.
- les playlists.
- les livres audios et les chansons.

Si les fichiers ne sont pas trouvés, des exceptions seront levées et des messages seront affichés à la console.

L'application permet d'afficher à la console:

- les chansons d'un album
- les titres des albums, rangés par date de sortie
- les titres des albums, rangés par genre
- les playlists, rangées par nom
- les livres audio rangés par auteur

L'application accepte les commandes suivantes, à la console :

- « c » : rajout d'une nouvelle chanson
- « a » : rajout d'un nouvel album
- « + » : rajout d'une chanson existante à un album

- « l » : rajout d'un nouveau livre audio

- « p » : création d'une nouvelle playlist à partir de chansons et livres audio existants

- « - » : suppression d'une playlist

- « s » : sauvegarde des playlists, des albums, des chansons et des livres audios dans les fichiers concernés.

- « h » : aide avec les détails des commandes précédentes.

2. Livrables

Vous développerez cette application à partir des besoins exprimés ci-dessus, à vous d'analyser et concevoir une solution fonctionnelle. Vous vous aiderez pour cela du langage UML pour conceptualiser les différentes entités et fonctionnalités de votre projet.

Avant de livrer votre solution, vous prendrez bien évidemment soin de réaliser les tests nécessaires pour s'assurer de son bon fonctionnement.

Le projet est à réaliser en binôme sous l'environnement Linux, et doit être déposé sur la plate-forme pédagogique Moodle au plus tard le 20/12/2020 à 23h55, sous la forme d'une archive .zip à vos noms et prénoms (NOM1_NOM2.zip), contenant tous les fichiers sources du projet ainsi qu'un rapport de trois pages au format pdf.

Votre programme doit obligatoirement présenter les différentes thématiques suivantes :

- Héritage
- Sérialisation
- Gestion des exceptions
- Les collections
- Au moins une classe abstraite sera implémentée et au moins une interface sera utilisée
- Les fichiers doivent contenir au moins : 20 éléments, 3 albums et 2 playlists

Le rapport ne devra pas comporter de code et uniquement décrire :

- Le diagramme de classe en UML.
- Les contributions de chaque étudiant.
- Le travail réalisé.
- Les difficultés rencontrées et les solutions qui leur ont été apportées.

Remarque : L'utilisation d'un IDE (Eclipse, IntelliJ, NetBeans etc…) est fortement déconseillée !

3. Techniques à utiliser :

Sérialisation: En informatique, la sérialisation est un processus visant à coder l'état d'une information qui est en mémoire (une structure de donnée, un objet, etc.) sous la forme condensée, par exemple une suite d'octets. Cette suite pourra par exemple être utilisée pour la sauvegarde (persistance) ou le transport sur le réseau (proxy). L'activité symétrique, visant à décoder cette suite pour créer une copie conforme de l'information d'origine, s'appelle la désérialisation.

Java intègre naturellement des méthodes de sérialisation/désérialisation permettant de sauvegarder/charger des objets dans des fichiers.

Pour cela, la classe que l'on désire sérialiser (et toutes les classes que l'on utilise au sein de cette classe) doivent simplement implémenter l'interface Serializable.

L'interface Serializable ne possède pas de méthode à redéfinir. Elle sert uniquement à identifier les classes pouvant être sérialisées.

http://docs.oracle.com/javase/7/docs/api/index.html?java/io/Serializable.html

https://jmdoudoux.developpez.com/cours/developpons/java/chap-normes-dev.php

4. Barème :

Item	**Détails**	**Nb. Points**
Rapport	*Rapport bien rédigé et contenant tous les parties citées ci-dessus.*	3
javaDoc	*Les commentaires dans le code sont pertinents et la documentation javaDoc est générée dans le répertoire doc ; fichier readme si necessaire*	1
Règles de codage*	*Les règles de codage Java sont respectées*	1
Abstraction/heritage	*Une classe abstraite et de l'héritage sont présents*	1
Exceptions	*Des classes d'exceptions sont créés et gérées*	2
Encapsulation	*Les private/public/protected sont correctement utilisés*	2
Collections	*Des collections sont utilisées*	2
Interfaces	*Des interfaces Java sont utilisées*	2
Diagramme UML	*Le diagramme UML est dessinée*	2
Respect des spécifications	*Commandes à la console*	2
Sérialisation	*La sérialisation est utilisée*	2
Total		**20**