Projet POO – Java 2020 – 2021

jMusicHub

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 XML:

- les albums (« albums.xml »)
- les playlists (« playlists.xml »)
- les livres audios et les chansons (« elements.xml »)

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 chansons d'un album, rangées par genre
- les playlists
- 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 xml respectifs

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

La documentation javaDoc sera rédigée en Anglais.

La documentation du programme contiendra le diagramme de classe en UML.

Au moins une classe abstraite sera implémentée et au moins une interface sera utilisée.

Le programme peut être livré sous Linux ou sous Windows.

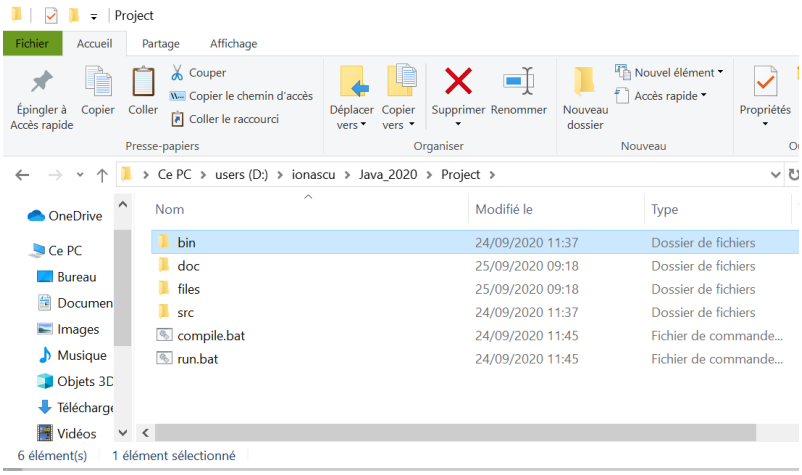
Les fichiers .xml doivent contenir au moins : 20 éléments, 3 albums et 2 playlists

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

Livrables

Le programme respectera la structure des répertoires/packages décrite ci-dessous. Le package **util** contiendra les classes qui gèrent la lecture/écriture des fichiers xml

→project				
→	bin			
	→	musichub		
		→	main	*(.class files)*
		→	business	*(.class files)*
		→	util	*(.class files)*
→	src			
	→	musichub		
		→	main	*(.java files)*
		→	business	*(.java files)*
		→	util	*(.java files)*
→	doc			*(javadoc documentation)*
→	files			*(.xml files)*
→	compile.bat			*(script de compilation: extension .bat sous Windows, .sh sous Linux)*
→	run.bat			*(script d'exécution: extension .bat sous Windows, .sh sous Linux)*



Barème

Item	Détails	Nb. Points
Build/execution sans erreurs	*Les scripts compile et run fonctionnent correctement*	2
Structure des packages	*La structure donnée ci-dessus est respectée*	1
javaDoc en anglais*	*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
	Sauvegarde/lecture XML	2
Total		**20**

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