M1 GIL – Mini-projet d'architecture logicielle 2019–2020
Djubaka – Un gestionnaire de listes de lecture

F. Nicart

À rendre avant le **11 mai 2020**

1 Présentation du mini-projet

Le projet consiste en un gestionnaire et un lecteur de listes de lecture multimédia (*playlists*). Une telle application permet à son utilisateur de programmer un enchaînement de lecture de fichiers lus par un lecteur multimédia. Nos listes de lecture pourront être composées de deux types de médias : des fichiers audio et des fichiers vidéos.

Comme nous n'allons pas réaliser la lecture à proprement parler de tels fichiers, nous utiliserons de faux fichiers multimédia qui contiendront, sous forme de texte, une description de leurs propriétés (une par ligne). Les fichiers audio (`*.mta`) contiendront la durée en secondes, le nom de la piste, le nom de l'artiste tandis que les fichiers vidéo (`*.mtv`) contiendront la durée en secondes, le titre de la vidéo, la résolution. Pour effectuer vos tests, une archive `compilation.zip` vous est fournie qui contient des fichiers (`.mta`) et (`.mtv`).

Enfin nos listes pourront contenir des sous-listes permettant de grouper des oeuvres et/ou d'autres sous-listes par thèmes en fonction de l'humeur. Une sous-liste possède un nom qui la rend visible comme une simple entrée dans la liste qui la contient. Par exemple, votre liste principale pourrait contenir une sous-liste *zen/concentration music*[1], et une liste *hacking/coding music*[2].

Les listes seront sauvegardées dans un format XML dans des fichiers `*.xpl`, consignant la programmation de lecture que l'utilisateur a créée. Les fichiers multimédias impliqués seront référencés sous forme de chemins relatifs. Bien que cela ne soit pas demandé, on prendra en compte la possibilité par la suite de lire des listes de lecture dans un autre format.

Le projet consistera en trois programmes :

1. Un éditeur de listes de lecture permettant de créer, modifier et sauvegarder des listes de lecture dans un fichier.
2. Un lecteur avec interface graphique permettant de lire (de façon simulée) une liste de lecture.
3. Un lecteur en ligne de commande permettant de lire (de façon simulée) une liste de lecture.

Ces trois programmes séparés sont obligatoires. Il n'est pas demandé de pouvoir modifier la liste de lecture depuis les lecteurs.

1.1 L'éditeur de liste

L'éditeur de liste doit permettre les opérations suivantes :

1. Créer une nouvelle liste en lui donnant un nom.
2. Charger une liste sauvegardée.
3. Sauvegarder la liste de lecture.

1. À écouter pendant que vous travaillez avec un papier et un crayon sur la modélisation. ;-)
2. Pour le moment où vous allez coder furieusement.

4. Lister les entrées de la liste actuelle avec leur durée. Dans le cas d'une sous-liste, la durée sera la somme des durées des fichiers multimédias qu'elle contient.
5. Entrer dans une sous-liste de la liste actuelle.
6. Remonter dans la liste parent (si il y en a une).
7. Importer un fichier multimédia.
8. Importer tous les fichiers multimédias contenus dans un dossier.
9. Importer une autre liste de lecture en tant que sous-liste.

Pour cette partie, et pour faire simple, aucune interface graphique n'est demandée. Un programme rudimentaire en ligne de commande sera suffisant qui pourrait consister en un interpréteur de commandes simple (ex : `list`↲ pour lister le contenu actuel, `save toto.xpl`↲ pour enregistrer sur disque, `enter 6`↲ pour entrer dans la sous-liste correspondant à l'entrée numéro 6 de la liste actuelle, etc.). Les imports pourront se faire systématiquement en fin de (sous-)liste.

1.2 Lecture de fichiers multimédias et de listes de lecture

Les lectures de fichiers multimédias seront simulées. Pour cela, on ouvrira le fichier multimédia pour en extraire les informations et les afficher dans l'interface (voir ci-dessous) et une progression de la lecture sera affichée dans l'interface et mise à jour toutes les secondes. Pour cela, on pourra utiliser l'API Timer (voir `java.util.Timer` et `java.util.TimerTask`). La lecture est terminée lorsque le temps écoulé a atteint la durée du fichier en cours de lecture.

Un lecteur multimédia est invoqué avec une liste de lecture et démarre la lecture du premier élément de la liste. Lorsque la lecture d'un fichier se termine, le lecteur passe automatiquement à l'entrée suivante de la liste. Si cette entrée est une sous-liste, le lecteur entre dans la sous-liste et commence la lecture de la première entrée. Lorsque le lecteur atteint la fin d'une sous-liste, il poursuit la lecture à l'entrée suivante dans la liste parent. Il s'arrête lorsqu'il a atteint la fin de la liste racine.

1.3 Le lecteur en ligne de commande

Ce programme sera invoqué avec le nom d'un fichier contenant une liste de lecture en paramètre et commencera automatiquement la lecture du premier fichier de la liste. À chaque démarrage d'un nouveau fichier, les informations de ce dernier seront affichés dans le terminal. Puis, pour chaque seconde écoulée, un caractère ':' sera affiché à la fin de la ligne pour indiquer la progression de la lecture. (Les plus courageux pourront effectuer à la place un affichage temps écoulé/temps restant en utilisant une astuce communément usitée qui consiste à imprimer le caractère de rang décimal 13 pour ramener le curseur en début de ligne sans passer à la ligne suivante et faire ainsi en sorte que le prochain affichage efface le précédent).

L'utilisateur pourra taper une commande pour commander au lecteur les actions détaillées ci-dessous.

1.4 Le lecteur avec interface graphique

Ce programme sera également invoqué avec le nom d'un fichier contenant une liste de lecture en paramètre (ou un menu `fichier/ouvrir` pour les plus courageux) et ouvrira une petite interface graphique contenant :

1. Une zone d'affichage des informations du fichier en cours de lecture (Nom, artiste, etc.).
2. Une zone affichant la progression de la lecture : le temps écoulé, la durée totale du fichier en cours et éventuellement une barre de progression (il n'est pas demandé que l'utilisateur puisse déplacer manuellement la barre).
3. Un panneau de commande permettant d'ordonner au lecteur les actions suivantes :
   — ▶ : démarrer la lecture.
   — ⏸ : met en pause la lecture.
   — ⏸ : met en pause la lecture.
   — ⏩ : passe au fichier suivant.
   — ⏪ : revient au fichier précédent.
   — ⏭ : termine la sous-liste actuelle et démarre la lecture de l'entrée suivante dans la liste parent.
   — ⏮ : termine la sous-liste actuelle et revient à l'entrée précédente dans la liste parent.

Enfin, si le programme est invoqué avec l'option `-d` (déverminage), il affichera, en plus de l'interface graphique, toutes les informations dans le terminal exactement comme la version en ligne de commande.

2 Modalités

— Le travail est à réaliser en binôme maximum [3]
— L'implémentation et le compte-rendu sont à rendre avant le **11 mai 2020**.

3 Travail à rendre

Vous fournirez une archive nommée `projet-al-nom1-nom2.zip` comportant un dossier éponyme. Celui-ci contiendra le code source de votre implémentation, votre rapport au format `pdf`, ainsi que des jeux d'essai de liste de lecture.

Il est important de noter que le rapport aura autant d'importance que l'implémentation. Celui-ci devra contenir :

— Une explication de l'architecture globale illustrée par un ou plusieurs diagramme UML.
— Pour chaque patron mis en oeuvre : la motivation du choix du patron, le diagramme UML de la partie de votre application correspondant au patron instancié (en prenant soin d'y indiquer les acteurs du patron).
— Éventuellement, une liste des patrons que vous avez hésité à utiliser et les raisons pour lesquelles vous les avez écartés.
— Le moins de fautes d'orthographe et de français possible.

Un autre point important à garder à l'esprit est que ce mini projet est un prétexte à l'élaboration d'une architecture de taille raisonnable mais respectant les bons principes vus en cours. Par conséquent, le logiciel n'a pas vocation à être abouti : certaines fonctionnalités énumérées peuvent ne pas être développées, en particulier lorsqu'elle n'interviennent pas dans la réalisation d'un patron. On utilisera par exemple des bouchons pour les fonctions pour lesquelles le temps a manqué en le mentionnant dans le rapport et en décrivant le travail à réaliser pour compléter la fonctionnalité.

4 Quelques indications

— Vous soignerez la modularité de votre application et justifierez dans votre rapport la répartition en paquetages.
— (Comme d'habitude,) vous soignerez la rédaction de vos interfaces. Vous détaillerez leur conception dans votre rapport.
— Vous fournirez une description du format de sauvegarde (votre DTD) en expliquant sa conception.

3. Ou en singleton ;-)