Interfaces et annotations

1 Interfaces (et utilisation d'exceptions+manipulation de fichiers)

On se place globalement dans le cadre d'un logiciel permettant de gérer des fichiers musicaux et de créer des playlists. On ne réalisera ici que des portions extrêmement simplifiées.

1.1 Définition d'interfaces

Question 1. Modéliser et définir en Java une interface ElementAudio. Un élément audio est caractérisé par :

- une durée en secondes
- un titre
- un nom de fichier où est stocké l'élément (ici sous forme du chemin absolu vers le fichier)
- une taille (en octets)

Question 2. Modéliser et définir en Java une interface PlayList. Une playlist est un élément audio qui a de plus un nombre d'éléments.

1.2 Implémentations d'interfaces

Question 3. Modéliser et définir en Java une classe AbstractAudioElement qui implémente ElementAudio. En plus des attributs nécessaires à l'implémentation de l'interface, on ajoutera un attribut de type java.io.File qui représente le fichier correspondant au nom de fichier de l'élément audio. Ce fichier est créé à la création de l'élément audio abstrait. Ce fichier peut exister si le nom de fichier correspond effectivement à un fichier ou ne pas exister dans le cas contraire. Dans AbstractAudioElement, la taille en octets et la durée ne sont pas définies.

Question 4. Modéliser et définir en Java une classe Song. Une chanson est un AbstractAudioElement qui introduit de plus un attribut représentant le chanteur (sous forme de chaîne de caractères pour simplifier). La classe Song sera définie ainsi :

- le constructeur permettra de position la durée, le titre, le nom de fichier et le chanteur. On ne se sert pas des métadonnées qui pourraient être lues dans le fichier audio. Si le nom de fichier ne correspond pas à un fichier existant (méthode exists de File), on jettera une exception de type IncorrectFileNameException, exception qui véhiculera le nom de fichier fautif. On ne s'assurera pas que le fichier est bien un fichier audio.
- la taille d'une chanson est définie comme la taille du fichier la contenant (on utilisera la méthode length() de File). La méthode lenght peut lever une exception, que l'on choisit ici de propager. De quel type serait cette exception ? Devez-vous signaler que votre méthode qui retourne la taille peut également jeter cette exception ? Si vous choisissez de le faire, quelle est la conséquence sur l'interface ?

Question 5. Modéliser et définir en Java une classe SimplePlayList. Une simple playlist est un AbstractAudioElement qui implémente PlayList. SimplePlayList sera définie ainsi :

- une simple playlist contient des chansons (Song)
- le format du fichier pour une playlist est défini comme suit : la première ligne contient le nom de la playlist, les lignes suivantes contiennent chacune une description de chanson (que vous organiserez à votre goût)
- Le constructeur prend en paramètre le titre de la playlist et le nom de fichier. Si le fichier existe, le fichier est lu pour remplir la liste de chansons (gérez alors le titre à votre guise). Sinon, on considère qu'il s'agit d'une nouvelle playlist, le fichier est créé (createNewFile). Lors du remplissage de la liste de fichiers, si la Song ne se crée pas correctement à cause d'un nom de fichier incorrect, la chanson n'est simplement pas ajoutée à la playlist, et un message d'erreur avec le nom du fichier fautif est affiché sur la console.
- une méthode addSong permettra d'ajouter une chanson à la liste (à la fin). Le fichier sera modifié en conséquence.
- la durée d'une simple playlist est la somme des durées des chansons qui la composent.

- la taille d'une playlist est la taille du fichier de la playlist ajoutée à la somme des tailles des chansons la composant.
- le nombre d'éléments d'une playlist est le nombre de chansons la composant.

2 Annotations

2.1 Annotations existantes

Question 6. Dans la classe Song, on voudrait maintenant obtenir le chemin du fichier incluant ou pas l'extension du fichier, selon un paramètre booléen. Essayer de redéfinir de la sorte la méthode qui retourne le chemin du fichier. Ajoutez l'annotation @Override. Le code ne compile plus, pourquoi ? Enlevez l'annotation.

Question 7. Ajoutez un peu de Javadoc à la classe de votre choix, et générez sa javadoc.

2.2 Création d'une annotation

Question 8. Définissez une annotation Todo portant sur les méthodes, utilisable à l'exécution, dont les éléments précisent :

- le type de la tâche à effectuer sur la méthode (écrire, améliorer la complexité, refactoriser, tester)
- le numéro de la version pour laquelle ce doit être effectué
- la durée estimée de la tâche

Question 9. Annotez la classe de votre choix avec cette annotation. Vérifiez que vous ne pouvez annoter que les méthodes, et pas les autres éléments de code

Pour être capable de faire quelques premiers traitements sur les éléments disposant de cette annotation, il faut maîtriser l'introspection, nous le verrons dans un prochain TD-TP.

3 PlayList composite

Question 10. Créez un nouveau type de playlist : les playlists composites, c'est-à-dire une playlist pouvant se composer de chansons et/ou de playlists.