## Analyse de la conception selon les principes SOLID avec argumentaire

1. Ouvert/Fermé

Nous avons déterminé des classes et méthodes publiques qui peuvent hériter d’autres fonctionnalités mais les symboles sont privés pour ne pas perturber le fonctionnement de notre logiciel.

« Toutes les entités logicielles doivent être ouvertes à l’extension, mais fermées à la modification »

1. Responsabilité unique

Nous pensons que la caractéristique de responsabilité unique est bien honorée puisque nous avons découpé le projet en fonctionnalité que nous avons adapté par classe. Nous avons un tat d’éléments musicaux : albums playlist morceaux et ils sont chacun menées par une classe séparée lorsqu’on ajoute des morceaux, playlists sou albums au serveur les méthodes sont respectivement dans les classes de leur type, encapsulé.

De même pour les réactions entre le client et le serveur, les méthodes sont encapsulées par un Requestmanager dans but de permettre l’envoi des commandes du client au serveur.

« Chaque module, classe, ou méthode doit être responsable d’une seule partie de la fonctionnalité que le logiciel fournit, et cette responsabilité devrait être entièrement encapsulée par la classe, le module, ou la méthode. »

1. Ségrégation des interfaces

Pour ce caractère, le logiciel IntelliJ gouverne cela en nous proposant de supprimer les interfaces qui ne sont pas utilisés. Par ailleurs nous n’avons eu en recours que très rarement à l’implémentation d’une interface.

« Une classe ne doit jamais être forcée à implémenter une interface qu’elle ne l’utilise pas ou une méthode qui n’a pas de sens pour elle »

1. Inversion des dépendances

Le classe Song comme peut le témoigner le diagramme UML de classes ne Dépend que de l’abstraction AudioElement.

« Les entités doivent dépendre uniquement des abstractions »