EXPLICATION DE NOTRE LOT 2 :

Pour réaliser le lot 2, nous avons rajouter les fonctionnalités suivantes :

Dans un premier temps, nous avons décidé de créer une nouvelle classe intitulé *Opinion*. Cette classe nous permet d'utiliser 2 attributs distincts :

- *Mark* un flottant permettant de stocker la note (entre 0 et 5) de la Review.
- *Reviewer* un objet de type *Membre* permettant de stocker la référence de l'utilisateur émettant cette opinion.

Cette stratégie nous permet d'attribuer une ArrayList d'objets *Opinion* pour chaque objet *Review*. Grace à cette liste et à l'attribut *Reviewer*, nous pouvons retrouver les *Opinions* émises par un Membre et s'assurer qu'il ne dépose qu'une seule *Opinion* par *Review*.

Ensuite, nous avons réfléchi à l'implémentation du « karma ». Nous sommes arrivés à un système de coefficient propre à chaque membre. Ces membres se verront attribuer une valeur de karma à 1 au départ. Ensuite, nous ferons évoluer ce karma au gré des *mark* des *Opinions* laissées sur les *Reviews* du Membre.

Sur une review R1, après avoir rajouté une opinion O3 (sous-entendu qu'il existe déjà une Opinion O1 et O2) on calcule la moyenne des *marks* des Opinions O1, O2, O3. On obtient donc la moyennes des Opinions laissées sur une review d'un membre.

Pour calculer le « karma », on va calculer la moyenne des moyennes des Opinions laissées sur toutes les review d'un même utilisateur. A partir de cette moyenne de moyenne, on applique la formule suivante :

$$karma = 1 + \frac{(moyennedemoyenne - 2{,}5)}{5}$$

La sortie de cette formule va nous retourner une valeur entre 0.5 et 1.5. Elle va servir de pondération pour calculer les notes des Items. Si la moyenne des moyennes est supérieure à 2.5, les utilisateurs considèrent les reviews du membre pertinentes donc le karma va augmenter. A contrario si la moyenne des moyennes est inférieure à 2.5, les utilisateurs considèrent les reviews du membre impertinentes donc le karma va baisser. Ainsi, si le karma vaut 0.5, ses reviews vont compter 50 % de moins que les autres ; si le karma = 1.5, ses reviews vont compter 50 % de plus que les autres.

Lors du calcul de la note globale des Items (Book/Film) il suffit donc de multiplier chaque note laissée dans les reviews par le « karma » de l'auteur de la review puis d'en faire une moyenne pondérée.

Le fait de prendre un « karma » de 1 comme valeur par défaut est intéressant car tant que l'on ne rajoute pas d'Opinions, le calcul des moyennes est exactement le même que le LOT1, ainsi tous les tests du LOT1 continue de passer.

EXPLICATION DE NOTRE LOT 3

Pour réaliser le lot 3, nous avons rajouter les fonctionnalités suivantes :

Nous avons choisi, contrairement au sujet, de ne réaliser qu'une seule fonction de recherche d'Item. Cette fonction utilise une regex afin de chercher toute occurrence correspondant au critère de recherche. La fonction cherche dans les listes de Films et de Book et retourne une LinkedList de String contenant l'ensemble des titres des items correspondant à la recherche.

Apercu et explication de la regex :

films.get(i).getTitle().toLowerCase().replace(" " , "").contains(title.toLowerCase().replace(" " , ""))

La regex récupère le titre de chaque film et le converti en minuscule puis supprime tous les espaces présents dans le titre. Ensuite on cherche si ce titre « nettoyé » contient le titre recherché passé en paramètre, lui aussi « nettoyé ». Si c'est le cas on inscrit le titre dans la LinkedList et on passe à l'Item suivant dans la liste. A la fin de parcours des listes de Book et de Film, on retourne la LinkedList contenant toutes les occurrences du titre recherché.