Telecom Nancy

Université de Lorraine

Rapport de projet SD

Movie Manager

Auteurs :
Nicolas Bédrine
Vincent Albert

26 mai 2015

Table des matières

1 Présentation du sujet

1.1 Un rapide résumé

Le logiciel MovieManager est un gestionnaire de films permettant de gérer les films vus par l'utilisateur, de charger une base de films à partir d'un fichier texte, de les exporter en format json, et de créer une liste de propositions de films à voir.

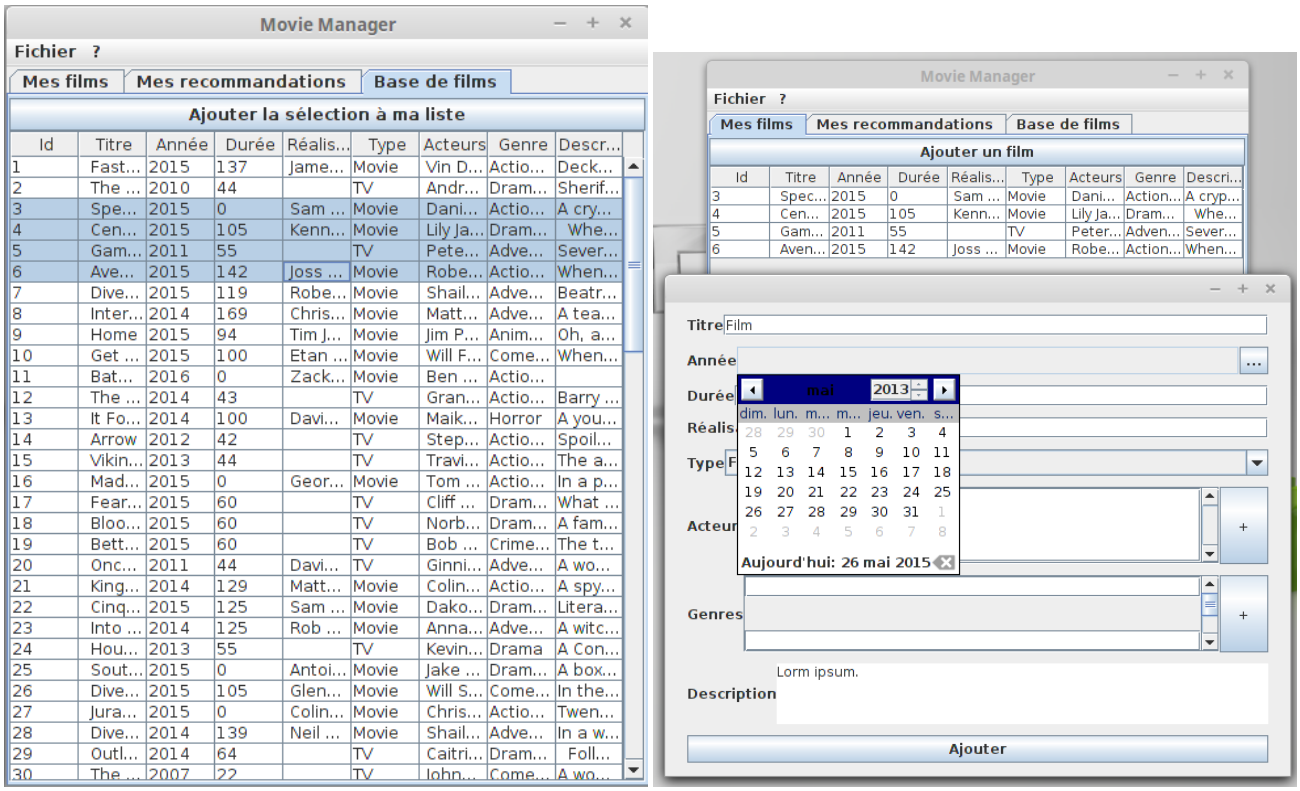

FIGURE 1 – Captures d'écran

Nous n'avons pas inclus de vue en console, mais uniquement une vue graphique. En effet, après demande de renseignements, on nous a dit qu'il n'était pas nécessaire de l'implémenter.

1.2 Les objectifs recherchés

Le but du programme est d'établir un standard de films qui correspond aux goûts de l'utilisateur. Une fois ce standard défini, nous recherchons dans le modèle les films qui lui sont proches. Ainsi, on ne proposera pas forcément les suites de films déjà vus par l'utilisateur si lesdits films ne font pas partie de ses films les plus regardés.
Dans le calcul de la distance entre les films, toutes les composantes sont prises en compte. On leur attribue des poids définis arbitrairement. Ainsi, voici leur hiérarchie :

- Titre du film
- Liste des genres
- Réalisateur
- Liste des acteurs
- Type (Film ou série TV)

2 Organisation

2.1 Les outils utilisés

Afin de nous coordonner et de gérer les versions, nous avons utilisé git. Le dépôt est accessible à l'adresse `https://github.com/Neressea/MovieManager`.
Pour générer l'UML et simplifier le lancement des tests unitaires, nous avons utilisé l'IDE Eclipse.
Nous pouvons aussi noter l'utilisation de jar externes pour le JDatePicker qui sert à la sélection de la date et pour les tests unitaires.

2.2 UML du modèle

L'UML du modèle a été généré à l'aide du module ObjectAid UML d'Eclipse.

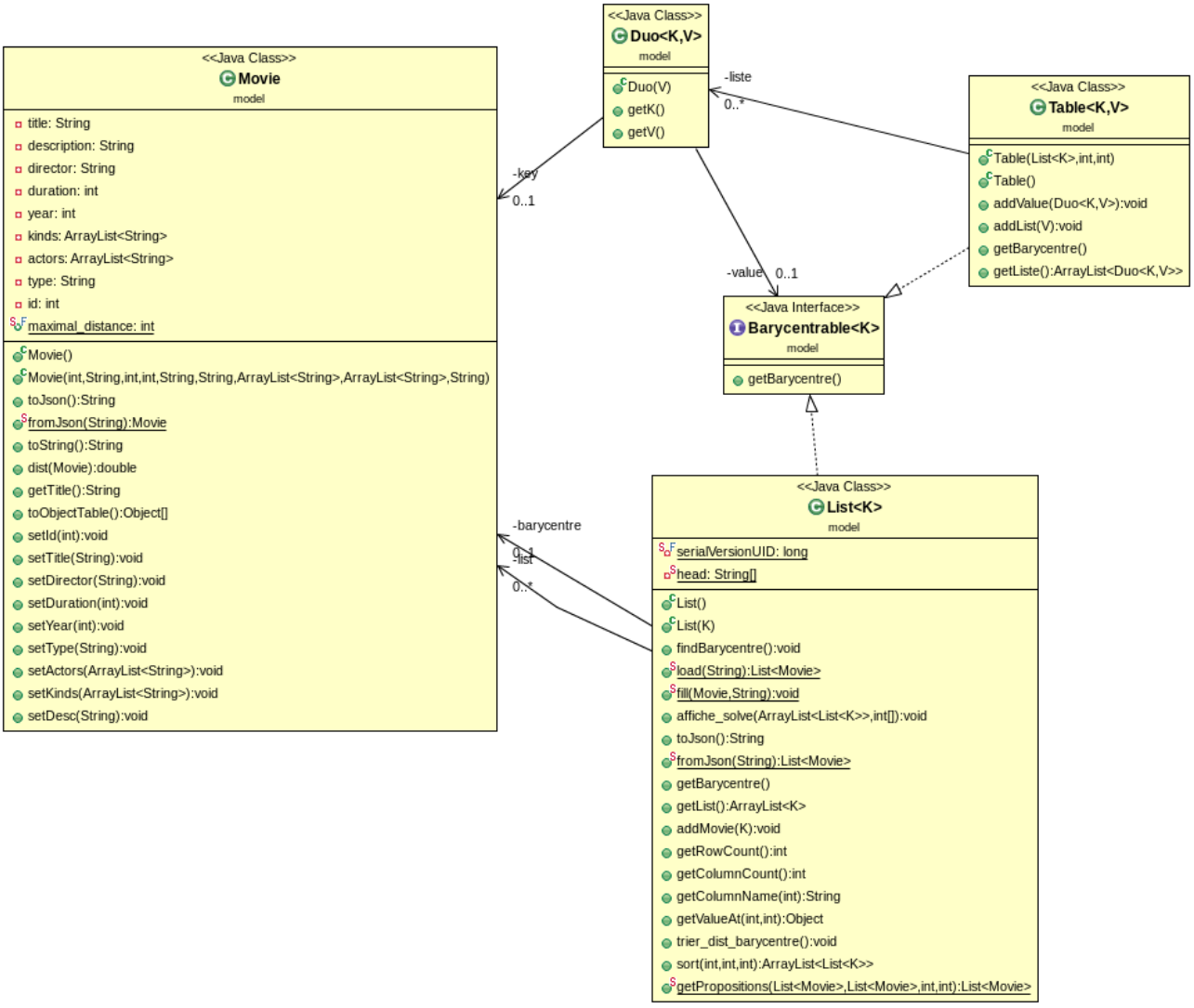

FIGURE 2 – UML du projet

Nous avons décidé de nous appuyer sur une structure en tables de hachage pour faciliter l'accès aux données lors des parcours et pour regrouper au préalable les films les plus similaires.
Les clés sont des films dits "barycentres" qui sont les plus représentatifs de leur groupe (c'est-à-dire que ce sont les films dont la distance globale à tous les autres est la plus faible). Les valeurs sont des listes de films ordonnées par rapport au film barycentre grâce au tri par sélection vu en TOP.
Le découpage en listes se fait grâce à une variante de l'algorithme des H-Means vu en mathématiques

numériques.
Pour permettre une réalisation ultérieure du code, nous avons utilisé des types génériques pour laisser le programmeur libre de son implémentation.
On peut remarquer que l'on a créé une interface Barycentrable qui nous permet de limiter l'utilisation de Duo aux objets avec l'héritage. Grâce à cela, nous pouvons appeler la méthode getBarycentre() à tout instant.

2.2.1 Les raisons du modèle

Nous avons préféré cette structure en table de hachage aux autres, notamment aux listes qui ne nous ont pas paru adaptée à la situation, autant pour les parcours lors de la recherche que pour les algorithmes de classification.
Nous l'avons aussi préférée à des structure en forme de graphe, car les temps de calcul et de recherche sont plus longs et elle n'est pas adaptée à un algorithme de partitionnement.
Les arbres auraient pu être intéressants pour la vitesse du traitement des données, mais il aurait été difficile au vu du temps imparti d'adapter des algorithmes de partitionnement aux arbres.

3 La phase de développement

3.1 Répartition des tâches

La conception a constitué la première étape du projet. Nous n'avons commencé à coder qu'une fois le schéma des structures utilisées complètement finalisé. Ensuite, nous nous sommes répartis les tâches afin de réaliser parallèlement le code et les tests associés. La rédaction quant à elle a été réalisée en dehors des étapes de codage.

- Conception
  - Nicolas : 3h
  - Vincent : 3h
- Codage
  - Nicolas : 12h
  - Vincent : 15h
- Tests
  - Nicolas : 4h
  - Vincent : 2h
- Rédaction du rapport
  - Nicolas : 1h
  - Vincent : 2h

3.2 Les problèmes rencontrés

La partie graphique n'a pas posé de problèmes particuliers étant donné l'utilisation de la bibliothèque swing. Nous avons respecté le pattern MVC en faisant hériter les ListMovie de la classe AbstractTableModel.
Cependant, le modèle n'a pas été aussi facile. En effet, nous avons voulu implémenter l'algorithme H-Means vu en cours de mathématiques numériques pour la classification des données. Nous avons eu en effet beaucoup de mal car il repose sur un aléas et les résultats retournés ne sont pas toujours identiques, ce qui a rendu les tests complexes à réaliser.
Nous avons aussi rencontrés des problèmes dont nous n'avons pas réussi à cibler l'origine ; nous avons pu alors appréhender une nouvelle fonctionnalité de git, à savoir git reset qui nous a permis de revenir à un commit précédent.
Alors que la méthode fromJson permettant de charger une liste de films à partir d'un fichier formaté existe, elle a été abandonnée au cours du projet par manque de temps, afin de privilégier l'écriture des tests.

3.3 Le lancement du logiciel

Initialement, nous avions créé un script de compilation et d'exécution. Cependant, lorsque nous avons utilisé Eclipse, nous ne l'avons plus mis à jour et il est désormais obsolète. Il n'est plus possible de lancer le projet que par un IDE.