Evolution et Restructuration - TP3

Louis Daviaud, Theo Kriszt, Anthony Chaillot

October 2018

1 Extraction des données de base

1.1 Modification du code

Le code n'a pas été beaucoup modifié.

- Une constante **PATH_TO_RESULT** a été ajoutée.

  Cette constante (statique finale) permet simplement la portabilité du code sur plusieurs machines.

- Dans le main , les classes à analyser ont été ajoutées dans une liste de String.

Pour voir le code se rendre dans le zip joint **TP2_P1**.

1.2 Schéma de la hiérarchie des classes / interfaces de départ

Se référer au fichier **hierarchie_initiale.jpg** fourni en pièce jointe.

2 Maîtrise de l'outil RCAexplore

2.1 Fichiers sur lequel nous travaillons

Pour voir les fichiers **.rctf** se rendre dans le zip joints, **TP2_P2**. Cette archive contient deux dossiers :

- **multimap_fca** contenant le .rctf, .xml et le rendu au format .pdf de l'analyse via l'algorithme treilli
- **multimap_ares** contenant le .rctf, .xml et le rendu au format .pdf de l'analyse via l'algorithme AOC_poset

2.2 Commentaires sur RCAExplore

RCAExplore nous a été utile pour la génération automatique de la hiérarchie entre les interfaces et les classes (abstraites et concrètes) composant Multimap via des Concepts (grâce à **explogui**), l'outil a été également utile pour parcourirs ces concepts à l'aide du **browser**.

Il serait intéressant de pouvoir passer de l'explorateur graphique (**explogui**) à l'explorateur de hierarchie (**browser**) et vice-versa. Dans les explorateurs, il serait très appréciable pour la lisibilité de pouvoir renommer les collections de concepts à la volée pour s'y retrouver en donnant des noms plus parlants et plus pertinents que "Concept_Collections_X".

3 Compréhension et analyse du résultat

Dans les analyses, tous les concepts doivent être étudiés de manière systématique

3.1 Analysez de manière systématique les concepts de l'AOC-poset. Indiquez lesquels vous trouvez intéressants pour analyser les classes et pourquoi (par exemple pour comprendre leur organisation). Chercher à nommer les concepts d'après leur position ou leurs caractéristiques, lorsque l'on trouve facilement un nom, c'est souvent un indice que le concept est utile à garder dans une hiérarchie

3.1.1 Analyse des concepts

":" = est introduit par le/les concept(s)

- **Multimap et ForwardingMultimap** : Concept_collection_11
- **ImmutableMultimap** : Concept_collection_11 , Concept_collection_8
- **ForwardingListMultimap** : Concept_collection_11 , Concept_collection_9
- **ForwardingSetMultimap**: Concept_collection_11, Concept_collection_10
- **LinkedListMultimap** : Concept_collection_11, Concept_collection_9, Concept_collection_1
- **ArrayListMultimap** : Concept_collection_11, Concept_collection_9, Concept_collection_6
- **ImmutableListMultimap** : Concept_collection_11, Concept_collection_9, Concept_collection_8, Concept_collection_4
- **ImmutableSetMultimap** Concept_collection_3, Concept_collection_8, Concept_collection_10, Concept_collection_11
- **HashMapMultimap** : Concept_collection_11, Concept_collection_10, Concept_collection_5
- **LinkedHashMultimap** : Concept_collection_2, Concept_collection_10, Concept_collection_11
- **ForwardingSortedSetMultimap** : Concept_collection_7, Concept_collection_10, Concept_collection_11
- **TreeMultimap** : Concept_collection_0, Concept_collection_7, Concept_collection_10, Concept_collection_11

3.1.2 Renommage des concepts

Pour le nommage des concepts, la plus part d'entre eux ne possède qu'une interface, on peut donc nommer le concept avec le nom de l'interface.

Dans le cas du concept n11 qui est tout en haut de la hiérarchie et qui présente deux interfaces à savoir Multimap et ForwardingMultimap. On peut généraliser le nom par Multimap que nous savons être l'interface la plus générique des deux ou nous pouvons proposer un nom composé qui serait alors "Multimap-ForwardingMultimap".

3.2 Comparez les résultats obtenus avec un treillis versus un AOC-poset : quels sont les concepts supplémentaires dans le treillis ? Analysez-les et indiquez lesquels vous trouvez utiles / inutiles et pourquoi

Le treillis possède un concept vide en plus de l'AOC-poset.

Ce concept vide indique qu'aucune classe ne possède toute les fonctionnalités à la fois et n'est donc pas nécessaire.

3.3 Quelles règles ou motifs intéressants dans les données observez-vous ? Expliquez comment vous les avez déterminés

On observe une spécialisation des méthodes comme pour la méthode create, qui renvoie un objet du type de l'interface courante.

Par exemple : le concept qui introduit **ImmutableMultimap** déclare une méthode s'appelant *ImmutableMultimapof()* tandis que le concept qui introduit **ImmutableListMultimap** déclare la méthode spécialisée qui s'appelle *ImmutableListMultimapof()*.

3.4 Donnez la hiérarchie d'interface qui peut être extraite de votre AOC-poset

On retrouve les interfaces Multimap et ForwardingMultimap tout en haut de l'AOC-poset. Cela signifie que les interfaces MultiMap et ForwardingMultimap sont confondues au niveau hiérarchique et exposent les mêmes méthodes publiques. Autrement, le diagramme AOC-poset présente la hiérarchie des concepts à la façon d'un diagramme de classes UML.