Extraction de concepts

Pierre Misse, Théo Rogliano, Bruno Verley

Octobre 2018

1 Introduction

Lors de la création ou le refactoring de programme orienté objet, il est souhaitable de factoriser le maximum de propriété de chaque arborescence d'objet pour -notamment- éviter la duplication de code. Ici, nous nous somme dans un contexte orienté refactoring, et partons donc de sources existantes, crées par Google, et vérifierons si il est possible de faire une meilleure factorisation des classes.

2 Travail effectué

Nous avons extrait un treillis depuis les classes Multimap de Google (figure :1) grâce à RCAexplore. Nous avons choisi d'utiliser l'algorithme FCA, et nous avons seulement généré les treillis. Les AOC-posets étant facilement déductible : seul "concept_0" est non introducteur.

Nous avons analysé dans un premier temps les Multimap sans les interfaces. Dans ce cas, tous les concepts sont introducteurs d'un unique objet. La hiérarchie des interfaces induite est donc triviale (la même que les objets AOC-posets, figure 2). En effet, vu que tout concept est introducteur, et qu'il est facile de le nommer (nom de l'objet) alors tous les concepts sont utiles et donc potentiellement intéressants. Nous constatons que cette hiérarchie ne correspond pas à celle implémentée par Google. Par exemple, l'objet ForwardingMultimap n'as que trois sous classes dans l'implémentation de Google, alors que dans l'AOC-poset, il est superclasse de toutes les autres. De plus, il existe des interfaces dans l'implémentation Google, or l'AOC-poset suggère qu'elles ne sont pas nécessaires.

Dans un second temps, nous avons rajouter les interfaces dans le treillis afin de déterminer si elles correspondent à un concept de l'AOC-poset. Nous avons constaté que ces interfaces introduisent les mêmes features que des objets déjà présents, comme indiqué par les commentaire dans la figure 2. En guise d'exemple, l'interface Multimap introduit les mêmes features que l'objet ForwardingMultimap.

Des règles intéressante qui peuvent être extrait de ces données sont par exemple
— ImmutableListMultimap.of() => ImmutableMultimap.of()
— SortedSet.keySet() => ImmutableSetMultimap

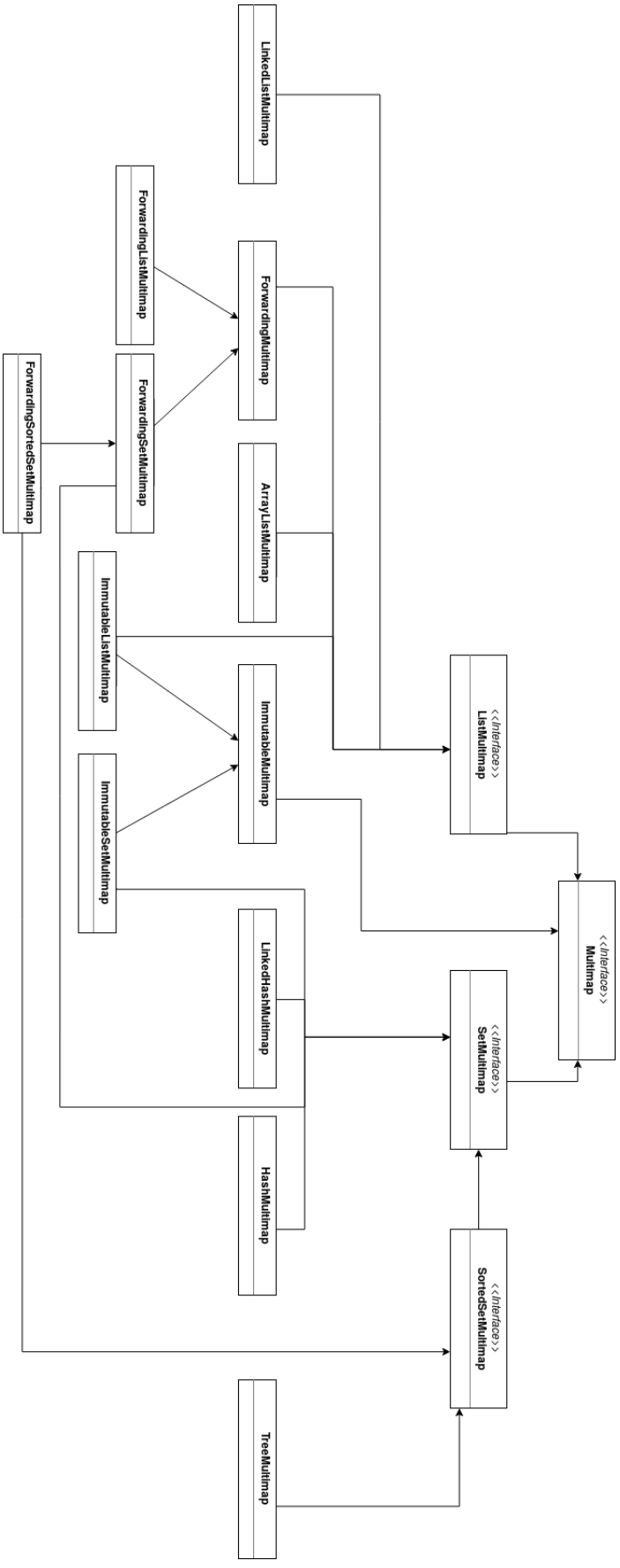

FIGURE 1 – UML décrivant l'implémentation de la multimap par Google

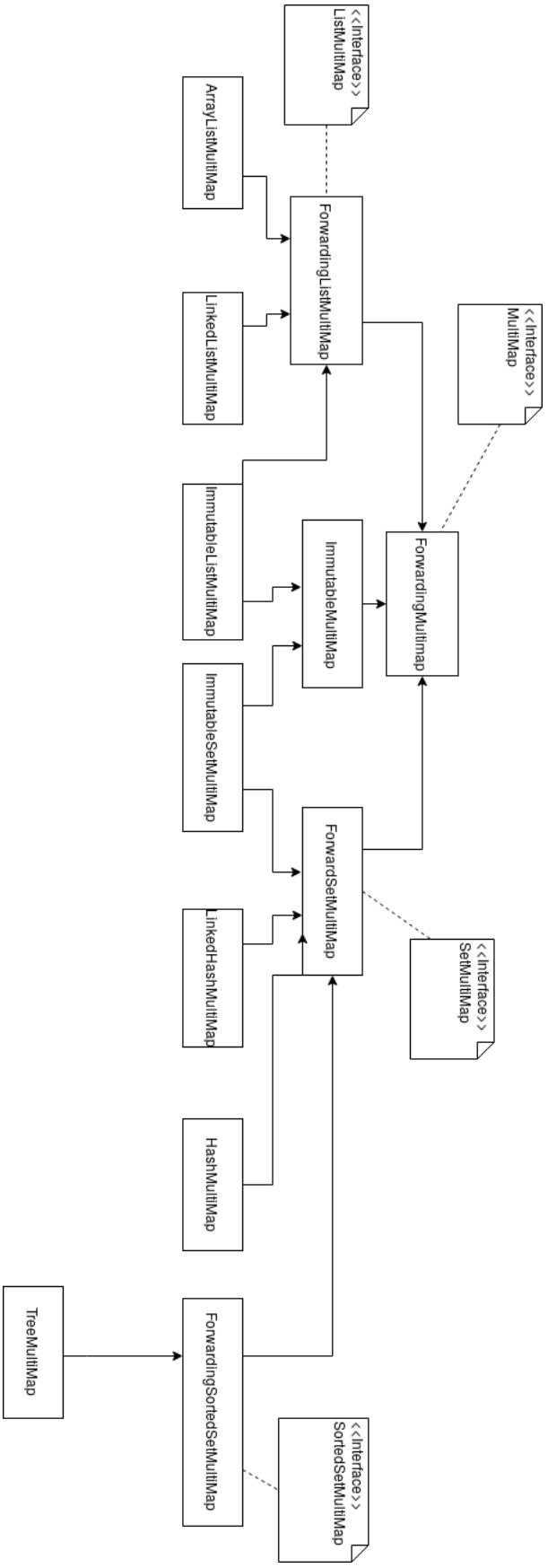

FIGURE 2 – UML construit à partir de l'AOC-posets

3 Conclusion

Finalement, nous pouvons dire que Google a plutôt bien planifié son arborescence de classe lors de la conception. Cependant, elle peut être améliorer vers une meilleure factorisation des concepts correspondant plus au treillis généré. Le refactoring que celà engendrerait (génération des interfaces, déplacements de features, etc...) ne semblent pas très coûteux pour ce cas spécifique.