Master informatique – Université de Montpellier - 2018

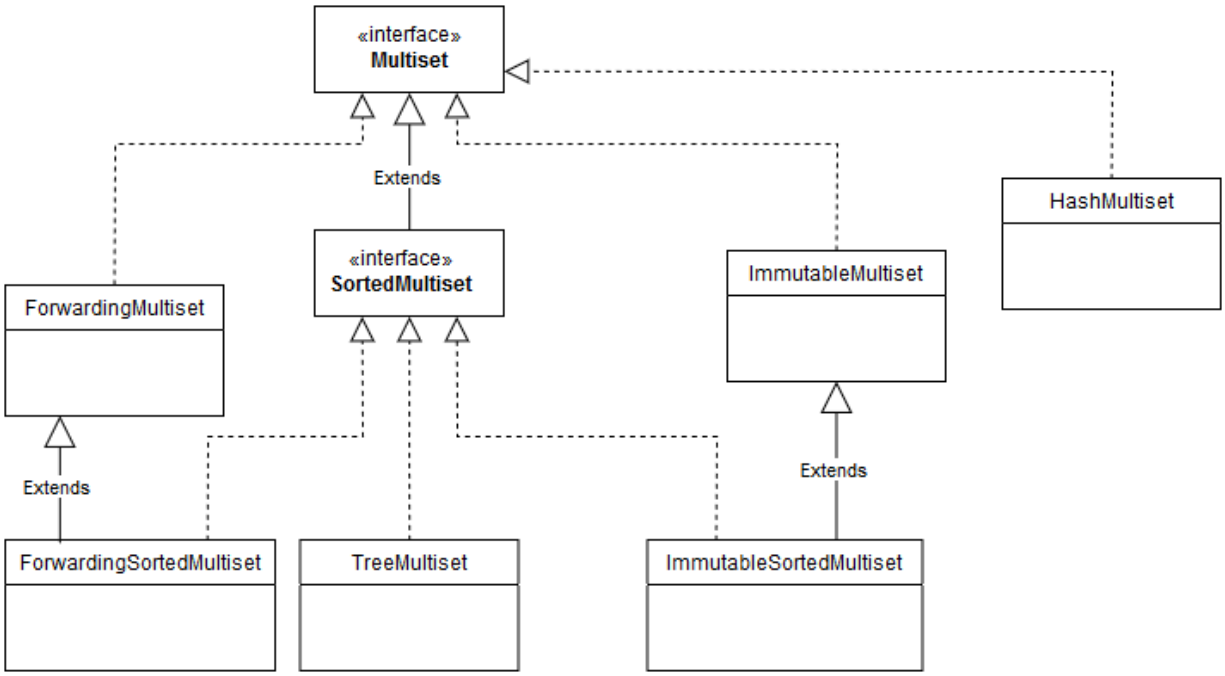
Extraction d'une hiérarchie d'interfaces Java (Big Refactoring)

TP évalué

réalisé par :
Ali LACHGUER – AIGLE

Pour la première étape dé l'étude du big refactoring, la réalisation d'un diagramme de classe décrivant la hiérarchie des classes et interfaces de l'ensemble des Multisets. Les éléments choisis pour l'étude sont :

- ForwardingMultiset
- SortedMultiset
- ImmutableMultiset
- HashMultiset
- ForwardingSortedMultiset
- immutableSortedMultiset
- treeMultiset



On a plusieurs classes qui implémentent directement l'interface Multiset, et d'autres qui héritent ou implémentent des sous-classes de l'interface Multiset. SortedMultiset est la seul sous-interface de Multiset, elle possède à son tour des classes qui l'implémentent et qui peuvent hériter d'autre classes implémentant Multiset.

Nous pouvons déjà remarquer que les classes dans le diagramme ci-dessus forment une hiérarchie d'héritage et d'implémentation, de trois nievaux. ForwardingMultiset par exemple implémente l'interface Multiset qui représente toute seule le premier niveau de la hiérarchie, puis ForwardingSortedMultiset représente donc le troisième niveau puisqu'il hérite de la classe ForwardingMultiset, et implémente l'interface SortedMultiset qui est au même niveau que ForwardingMultiset.

Pour générer le fichier .rcft, j'ai modifié le programme fourni en mettant dans le tableau de chaînes de

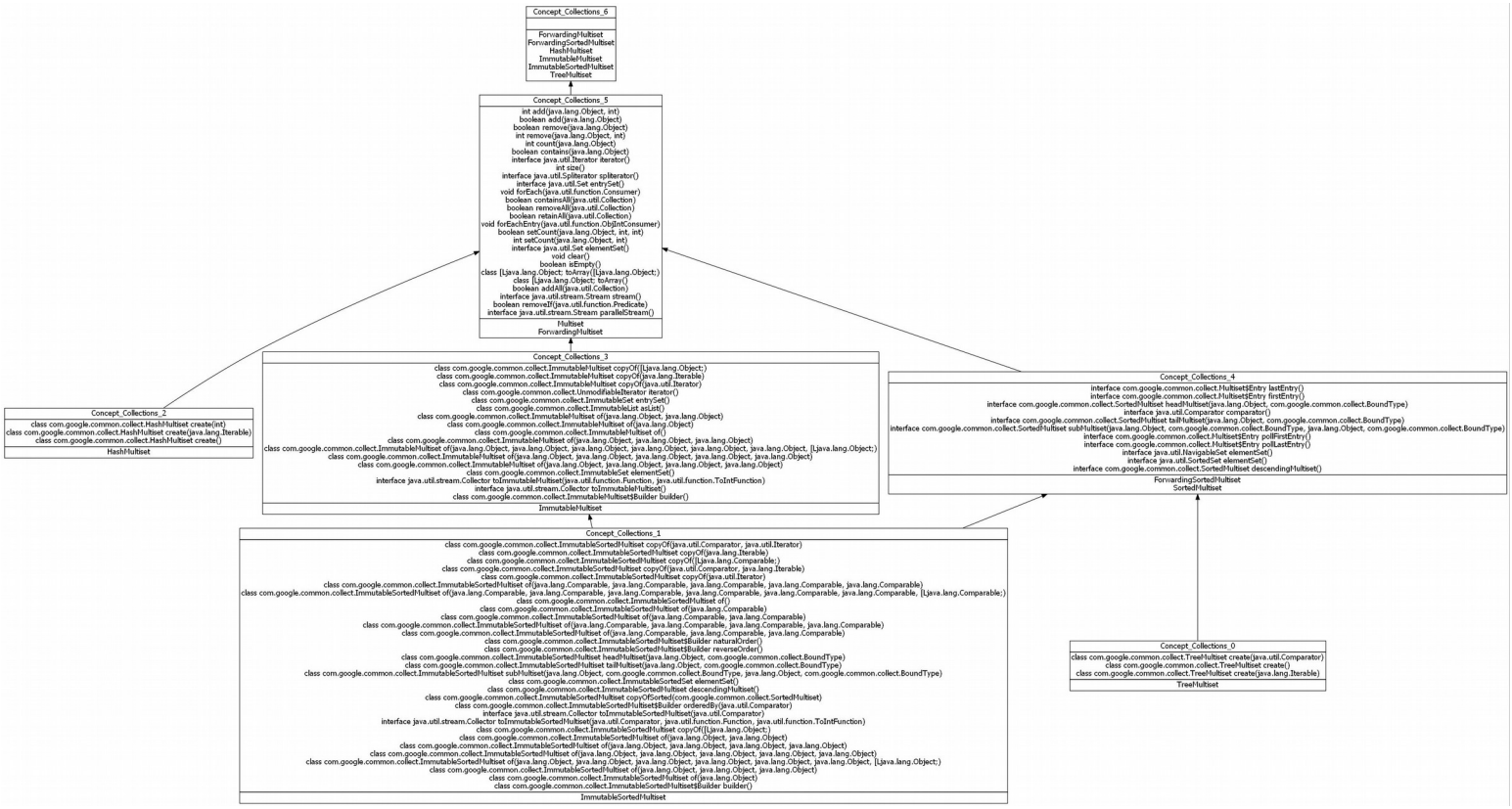
caractères la liste des éléments choisis à étudier, puis ajouter ce tableau à la liste complète qui prend des classes et interfaces, pour enfin les mettre dans la liste des éléments qui générera le tableau rcft.

```
String[] listeDesClassesInterfaces = {
        "com.google.common.collect.ForwardingMultiset",
        "com.google.common.collect.ForwardingSortedMultiset",
        "com.google.common.collect.HashMultiset",
        "com.google.common.collect.ImmutableMultiset",
        "com.google.common.collect.ImmutableSortedMultiset",
        "com.google.common.collect.TreeMultiset",
        "com.google.common.collect.SortedMultiset",
        "com.google.common.collect.ForwardingSortedMultiset"

};

ArrayList<String> listeComplete = new ArrayList<>();
listeComplete.addAll(Arrays.asList(listeDesClassesInterfaces));
listeComplete.addAll(concreteAbstractClassList(listeDesClassesInterfaces));
Extraction2018 d4 = new Extraction2018();
d4.setEntityList(listeComplete);
d4.setCharacteristicList(signatureSet(d4.getEntityList()));
```

Après avoir lancé RCAExplore avec le fichier .rcft généré en paramètre, on obtient l'AOC-poset ci-dessous.



On voit que la hiérarchie dans l'AOC-poset généré est différente que celle du diagramme de classes. Cela est dû aux attributs (méthodes) que possède les différents concepts (classes/interfaces), et qui sont utilisés pour déterminer la hiérarchie des concepts.

Le concept « Concept_collection_1 » contient le plus grand nombre d'attributs, ce concept est associé à la classe ImmutableSortedMultiset. Les méthodes définis au niveau des attributs de ce concept sont en fait des redéfinitions des méthodes de la classe ImmutableMultiset et SortedMultiset qui se trouvent respectivement dans le concept « Concept_collection_3 » et « Concept_collection_4 ». Comme nous étudions des classes et interfaces du langage Java, les méthodes sont préfixées avec le nom de leur classe correspondante, le programme RCAExplore les considère comme des valeurs différentes.

L'AOC-poset ne décrit pas efficacement pour cet exemple la hiérarchie de l'héritage, on voit que la classe ForwardingSortedMultiset et l'interface SortedMultiset se trouvent dans le même concept, alors que le diagramme de classe montre explicitement la relation d'implémentation entre eux. On peut simplement déduire que ces deux éléments possèdent les mêmes attributs, et donc l'une redéfini les méthodes de l'autre.

C'est en regardant le « Concept_collections_5 » qui contient la classe ForwardingMultiset, qu'on comprend que le concept « Concept_collections_4 » contenant ForwardingSortedMultiset et SortedMultiset a des méthodes de plus qui appartiennent à l'interface SortedMultiset, que donc ForwardingSortedMultiset est une sous-classe de ForwardMultiset car ils possédent les mêmes attributs, et que ForwardSortedMultiset implémente SortedMultiset car ils se trouvent dans le même concept, et donc possèdent les même méthodes.

On crée un deuxième AOC-poset avec cette fois, les éléments « ForwardingMultiset », « ForwardingSortedMultiset », « HashMultiset », « ImmutableSortedMultiset », et « SortedMultiset ». On obtient un AOC-poset de 4 concepts.

Le concept contenant « ImmutableSortedMultiset » (concept_collection_0) implique concept_collection_2 contenant « SortedMultiset » et « ForwardingSortedMultiset », c.à.d. « ImmutableSortedMultiset » possède les méthodes se trouvant dans « SortedMultiset » et « ForwardingSortedMultiset ». On peut décrire le concept_collection-2 comme le concept des « multiset triés » car il possède l'interface « SortedMultiset » responsable de la gestion des multisets triés. « ImmutableSortedMultiset » est donc un concept qui traite que les multiset triés.

On voit que les concepts des mutlisets triés possèdent les fonctionnalités du concept_collection-3 qui contient la classe « ForwardingMultiset », de même avec le concept_collection-1 contenant « HashMultset », cela vient du fait que le concept_collection-3 possède les méthodes de base pour un Multiset, et donc c'est normal que les autres concepts impliquent le concept possédant les méthodes de base ; si on compare cet AOC-poset avec le précèdent, on remarque que l'interface « Multiset » se trouve dans le concept_collection-3 avec la classe « ForwardingMultiset ».

En créant un treillis à partir des mêmes données, on obtient le treillis suivant :

Le treillis généré n'est pas très différent de l'AOC-poset généré à partir des mêmes données, le treillis affiche les mêmes informations, il possède un concept vide en plus des autres concepts, et qui n'apporte pas de nouvelles informations. Ce treillis n'est donc pas très utile pour notre analyse.

On peut déduire de notre AOC-poset, qu'il existe une hiérarchie au niveau des multiset triés, avec « ForwardingSortedMultiset » en haut de la hiérarchie avec l'interface « SortedMultiset » dans le même concept, cela est dû au fait que « ForwardingSortedMultiset » possède exactement les mêmes propriétés que « SortedMultiset » même si ce dernier est implémenté par « ForwaringSortedMultiset ». On peut remarquer la même chose avec « ForwardingMultiset » et « Multiset » qui se trouvent eux aussi dans le même concept. On déduit donc que « ForwardingMultiset » implémente « Multiset » sans ajouter de nouvelles méthodes en plus des méthodes implémentées.