Master Informatique - Université Montpellier - 2019

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

Travaux pratiques évalués

Dans ces travaux, vous étudierez un *big refactoring* consistant à extraire une hiérarchie d'interfaces Java à partir d'un ensemble de classes concrètes et abstraites.

L'objectif est de construire une hiérarchie d'interfaces décrivant les types d'un programme.

L'intérêt de cette approche est le suivant :

— La mise en évidence des interfaces permet de découpler la spécification de l'implémentation, et de donner une vue sur la structuration des types moins influencée par l'implémentation.

— Les interfaces sont des types plus abstraits, ce qui offre des possibilités d'écriture de classes abstraites et ainsi de réutilisation. De ces classes abstraites on peut dériver des classes concrètes et partager du code. Les interfaces servent de paramètres dans des méthodes qui peuvent avoir ainsi une portée très générale.

— La hiérarchie d'interfaces, dans un langage comme Java, peut être en héritage multiple (cependant sans introduction de conflits) contrairement à la hiérarchie de classes, ce qui fait que l'on a une meilleure classification conceptuelle des types (pour la compréhension) et plus d'opportunité d'écriture de méthodes générales (dans les interfaces avec des `default` méthodes ou dans des classes manipulant les objets des interfaces).

Vous travaillerez par groupes de 1 à 3 personnes. Ce document vous guide dans le travail et précise les documents à rendre pour l'évaluation de votre travail, cependant, si vous envisagez des variantes, ce n'est pas interdit, simplement documentez-les très soigneusement en les justifiant. Ce qui sera évalué ne sera pas seulement un résultat (il y a plusieurs solutions) mais le soin apporté à l'analyse et aux explications données.

Format pour rendre le travail : sur le Moodle (une seule personne du groupe dépose le travail du groupe et les noms de tous sont bien inscrits en clair dans les documents).

1 Extraction des données de base

Nous allons prendre comme données de base des classes **concrètes et abstraites** de la librairie Guava : `https://github.com/google/guava`.

Récupérez tout d'abord, sur le Moodle ou sur `https://jar-download.com/artifacts/com.google.guava` l'archive `guava-28.1-jre.jar` de la section **guava from group com.google.guava (version 28.1-jre)**. Mettez-la dans un répertoire de votre projet eclipse.

Un procédé simple consiste à créer une "User library" et à l'ajouter à votre project (clic droit sur le projet puis : Build Path -> Add Library -> User Library).

— Open the menu entry Preferences -> Java -> Build Path -> User Libraries

— Press New ..., enter a name (e.g. 'myguava')

— Press Add JARs... and search for the JAR files you downloaded

— Now select Source Attachment -> Edit..., and find the path to the jar file.

— Press OK.

Vous choisirez l'un des deux ensembles :

— les multisets
`https://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/Multiset.html`

— les multimaps
`https://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/Multimap.html`

Un brouillon de programme donné sur le Moodle pour le TP va vous aider à construire un fichier `.rcft` qui servira d'entrée à un programme capable de calculer un treillis de concepts ou un AOC-poset. Etudiez-le pour bien le comprendre et modifiez-le pour vos besoins.

Résultat attendu : Donnez un schéma de la hiérarchie des classes / interfaces de départ. Il faudra donner le programme modifié par vos soins (vous le modifierez au fur et à mesure du TP).

2 Maîtrise de l'outil RCAexplore

Vous trouverez l'outil RCAexplore (créé par Xavier Dolques), accompagné d'une documentation sur le Moodle et à cette adresse :

— `http://dataqual.engees.unistra.fr/logiciels/rcaExplore`

La documentation vous montrera comment lancer l'outil et l'utiliser avec une interface graphique et ci-dessous vous avez les principales commandes pour l'utiliser en ligne de commandes. Pour la suite, le fichier jar contenant l'outil est supposé se nommer `rcaexplore.jar`. Nous utiliserons seulement une partie des fonctionnalités de l'outil, car nous ne ferons pas d'analyse relationnelle.

Comme il s'agit d'un prototype de recherche, faites régulièrement des sauvegardes des différents fichiers (d'entrée, de sortie) pour sécuriser vos données.

2.1 L'éditeur de famille de contextes

On le lance en ligne de commande ainsi :

```
java -jar rcaexplore.jar editor &
```

Il permet de créer ou de lire des familles de contextes ; il charge et sauve des fichiers dans un format texte lisible et éditable facilement, le format RCFT (extension des fichiers .rcft). Vous ne devriez pas avoir à l'utiliser puisque vous créez vos fichiers de données automatiquement. Suivant les versions de Java, vous pourriez rencontrer des problèmes.

2.2 Le générateur de treillis de concepts, AOC-posets, et treillis Iceberg

On lance le générateur en ligne de commande par :

```
java -jar rcaexplore.jar explogui <fichier.rcft> <dossier de sortie>
```

pour lancer l'explorateur RCA sur le fichier rcft avec une interface graphique

En choisissant les options correspondant au démarrage, l'exploration produit le fichier `result.xml` et divers autres fichiers qui peuvent vous intéresser. Elle produit toujours un fichier `.dot` contenant l'ordre partiel construit (que l'on peut réouvrir avec Graphviz), un fichier `trace.csv` qui contient les options de configuration de chaque étape, et un fichier `latticebuilder.sh` qui sert à générer, à partir des fichiers `.dot`, des fichiers `.pdf` pour visualiser les treillis en ligne de commande.

Observez la configuration courante, qui est celle de l'étape 0, par `display configuration`. Dans l'étude que nous faisons ici, nous jouerons seulement sur les algorithmes de construction car nous ne ferons pas d'analyse relationnelle, et il n'y aura que deux étapes aux résultats identiques (0 et 1).

On peut créer (`choose construction algorithm`) des treillis (item de menu `fca`), des AOC-posets (item de menu `ares`), des ordres restreints aux object-concepts (item de menu `ocposet`), ou restreints aux attribute-concepts (item de menu `acposet`).

Une fois l'algorithme choisi, lancez la construction (par `auto`), qui se fait avec la configuration de départ. Lorsque l'outil vous propose de visualiser les résultats, vous pouvez accepter et vous aurez une représentation graphique permettant de voir facilement les ancêtres et descendants d'un sommet.

Les structures construites seront dans le répertoire `<dossier de sortie>`.

Les fichiers `stepi.dot` et `stepi-j.dot` vous présentent les structures construites à chaque étape de l'exploration par RCA. Ils peuvent être transformés en fichiers SVG, PDF, PNG, PS ou autres formats grâce

à la commande dot en ligne de commande (logiciel GraphViz). Le fichier `latticebuilder.sh` contient les commandes pour créer ces fichiers sans écrire vous-même toutes les commandes `dot` nécessaires.

A la fin, `result.xml` correspond à l'ensemble des structures conceptuelles générées pour toutes les étapes du processus, et une option de visualisation vous montre les structures directement. Il peut être analysé automatiquement pour tirer différents résultats a posteriori.

2.3 Le navigateur de concepts

Il permet de naviguer de concept à concept. On le lance en ligne de commande uniquement :
`java -jar rcaexplore.jar browser <fichier.xml>`

Pour voir le contenu d'un concept il faut choisir l'étape, le contexte et taper le nom du concept (dans le champ du haut) puis faire `update` (bouton en haut à droite). Les noms des concepts d'un contexte sélectionné sont affichés avec (entre parenthèses) leur support (taille de l'extension). Il est possible de faire un filtrage dans cette liste pour ne voir que les concepts dont le support est supérieur à une valeur donnée. On peut voir à partir d'un concept ses parents, ses enfants, son extension, son intension simplifiée et son intension. Si on sélectionne un élément de l'intension simplifiée et un élément de l'intension on peut voir juste au-dessous une représentation de la règle d'implication correspondante.

Résultat attendu : Donnez le(s) fichier(s) `.rcft` avec lesquels vous travaillez et présentez clairement les structures obtenues sur vos données suivant les différentes variantes d'utilisation. Créez des treillis et des AOC-posets pour voir la différence (donnez les fichiers .dot et .pdf). Commentez la manière dont RCAexplore vous a aidé, ou quelles améliorations de l'outil vous auraient été utiles.

3 Compréhension et analyse du résultat

Résultat attendu : un texte comprenant une discussion sur les points qui suivent.

- **Analysez de manière systématique les concepts de l'AOC-poset (algorithme ares). 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.**
- **Construisez également un treillis : quels sont les concepts supplémentaires qui apparaissent dans le treillis ? Analysez-les et indiquez lesquels vous trouvez utiles / inutiles et pourquoi.**
- **Donnez la hiérarchie d'interfaces qui peut être extraite de votre AOC-poset.**