Master Informatique - Université Montpellier - 2016

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

Travaux pratiques évalués (partie 1)

Dans ces travaux, vous étudierez un *big refactoring* consistant à extraire une hiérarchie d'interfaces Java. 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).

Le cas d'étude portera sur l'organisation d'une hiérarchie de types (interfaces) de collections extraite à partir d'une partie des classes concrètes décrivant les séquences en Java et sa comparaison avec la hiérarchie actuelle (classes abstraites et interfaces).

Vous travaillerez par groupes de 1 à 3. 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 : par mail à `huchard@lirmm.fr` avant le 5 octobre 2016 (si possible à la fin du TP du lundi 3 octobre). Le sujet du mail sera TP1 HMIN306, vos documents seront réunis dans un fichier `.zip`. Vérifiez que vous recevez un accusé de réception avant le 8 octobre.

1 Compréhension de la hiérarchie des séquences initiale

Etudiez un extrait de la hiérarchie des classes concrètes de collections restreinte à ces séquences :

ArrayBlockingQueue, ArrayDeque, ArrayList, AttributeList, ConcurrentLinkedDeque, ConcurrentLinkedQueue, CopyOnWriteArrayList, DelayQueue, LinkedBlockingDeque, LinkedBlockingQueue, LinkedList, LinkedTransferQueue, PriorityBlockingQueue, PriorityQueue, RoleList, RoleUnresolvedList, Stack, SynchronousQueue, Vector

Construire une représentation graphique de la hiérarchie des classes et interfaces portant sur les collections qui vous servira à comparer vos résultats à l'existant.

Différentes sources peuvent vous aider :

— Tutorial Oracle
`https://docs.oracle.com/javase/tutorial/collections/`
`https://docs.oracle.com/javase/8/docs/api/`
— Code source des classes et interfaces du package util
`http://hg.openjdk.java.net/jdk8u/jdk8u/jdk/file/5910b94ea083/src/share/classes/java/util/`

Résultat attendu : Schéma de la hiérarchie initiale des séquences commenté, distinguant clairement ce qui est interface, classe abstraite ou classe concrète. Indiquez quelles sources d'information vous avez utilisées et comment vous avez procédé pour construire la hiérarchie. Etablir trois fichiers contenant respectivement les listes de : classes concrètes (celles qui sont données ci-dessus dans l'énoncé), classes abstraites (super-classes de ces classes concrètes), interfaces de séquences (implémentées par ces classes concrètes), interfaces hors de la hiérarchie des collections et implémentées par ces classes concrètes.

2 Extraction des données de base

Vous partirez des classes concrètes de la hiérarchie des collections Java données ci-dessus (séquences). Un brouillon de programme d'analyse est disponible à l'adresse (attention à le vérifier soigneusement et à le compléter) :
https ://www.lirmm.fr/content/download/10869/163682/file/ExtractionInterfaces2016.java

Il vous faudra être capable de connaître pour chaque classe les signatures de méthodes :

— nom
— type de retour
— liste de types de paramètres
— optionnellement : liste des exceptions potentiellement signalées

De plus :

— Réfléchissez à ce que vous devez faire des interfaces telles que `Serializable` qui ne sont pas dans la hiérarchie des collections mais dont dérivent certains types de la hiérarchie des collections. Proposez une solution (les traiter et comment, ou ne pas les traiter).
— Donnez des idées pour récupérer les méthodes statiques qui seraient dignes de figurer dans des interfaces.
— Donnez des idées pour récupérer les méthodes par défaut qui seraient dignes de figurer dans des interfaces.
— Y a-t-il des attributs `public final static` dans les classes concrètes que vous étudiez ?

Résultat attendu : Programme commenté utilisant le package `reflect` ou procédé incluant un outil d'analyse statique réalisant l'extraction des données et fichier descriptif des classes concrètes.

3 Extraction d'une hiérarchie d'interfaces et analyse

— Construisez une hiérarchie d'interfaces à partir des classes concrètes de séquences, comparez-la avec la hiérarchie de classes et d'interfaces initiale. Analysez chaque interface individuellement (parmi celles que vous proposez et celles qui existaient déjà) en indiquant pourquoi vous la trouvez pertinente ou peu intéressante.

Résultat attendu : Document présentant de manière détaillée avec des graphiques et du texte explicatif la hiérarchie d'interfaces que vous proposez et l'analyse des résultats. Expliquez la méthode (manuelle ou basée sur un programme que vous avez écrit) que vous avez employée pour trouver les interfaces. Indiquez exactement quelles méthodes contient chaque interface.