Réusinage de modèles UML

Une bonne pratique de développement est d'alterner les ajouts de fonctionnalités avec des étapes de réusinage (refactoring). Les réusinages se sont tout d'abord appliqués au code : un réusinage consiste à modifier le code pour le rendre plus lisible, plus facile à comprendre et à modifier, sans changer ses fonctionnalités. Au sein d'une approche IDM, on effectue également des réusinages, mais au niveau des modèles. Nous allons écrire ici quelques méthodes de réusinage de modèle UML.

Côté technique

Nous aurons besoin de Papyrus comme modeleur UML. Il semble que Papyrus ne s'installe pas sur les dernières versions d'Eclipse. Si vous avez une telle version d'Eclipse, utilisez à la place la version Eclipse Papyrus RCP : `https://www.eclipse.org/papyrus/download.html`. Cette version inclut tous les plugins nécessaires par ailleurs pour ce qui suit.

Placez vous dans un "empty EMF project".

Vous utiliserez le fichier testSave.java présent sous Moodle. Ce fichier vous donne de quoi charger et sauver des modèles UML, et illustre l'appel de ces méthodes sur un petit modèle UML presque vide. Nous n'aurons pas besoin de générer le code pour le métamodèle UML, en effet, celui-ci est déjà présent dans un plug in eclipse. Les imports présents dans le fichier fourni ne devraient pas compiler directement. Au niveau des imports fautifs concernant xmi et UML, utilisez l'aide d'eclipse "fix project set up" et ensuite acceptez d'ajouter le plug in aux "required bundles".

Pour tester vos méthodes de réusinage, vous aurez besoin de modèles UML. Vous pouvez soit les saisir avec l'éditeur réflexif, soit utiliser le modeleur Papyrus. Le test de vos méthodes de réusinage consistera à :

- charger un modèle UML "d'entrée" stocké par exemple dans un fichier A.uml
- récupérer dans ce modèle les éléments sur lesquels vous allez appliquer votre réusinage (une classe, une méthode, un package, ...)
- appliquer votre réusinage aux éléments de modèle choisis
- sauvegarder le modèle réusiné dans un fichier par exemple Areusine.uml
- vérifier ("à l'oeil") que le réusinage a bien fonctionné comme prévu.

1. Écrire une méthode de réusinage qui permet de déplacer une classe d'un package à un autre. Pour implémenter facilement cette méthode, on supposera pouvoir récupérer une classe et un package à partir de leur nom (ce qui dans le cas général est faux, il faut disposer du nom qualifié). La méthode de refactoring prendra donc juste en paramètre le nom d'une classe, et le nom du package cible. Si le package n'existe pas, le réusinage échoue.
2. Écrire une méthode de réusinage qui remplace un attribut public par un attribut privé et une paire d'accesseurs (en lecture et en écriture). Cette méthode prend en paramètre le nom de l'attribut et le nom de la classe qui le contient. On supposera toujours que l'on peut récupérer une classe à partir de son nom.
3. Écrire une méthode de réusinage qui fait "remonter" une méthode dans une super-classe. Cette méthode prend en paramètre le nom de la classe d'origine, le nom de la super classe, et le nom de la méthode à faire remonter. Si le nom de la superclasse ne correspond pas à une super-classe de la classe origine, si le nom de la méthode donné ne correspond pas à une méthode de la classe d'origine, ou si une méthode concrète du même nom que la méthode à faire remonter existe déjà dans la super-classe spécifiée, alors le réusinage échoue.