Premières transformations de modèles : 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

Placez vous dans un "empty EMF project".

Vous utiliserez le fichier présent sur Moodle permettant de 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.

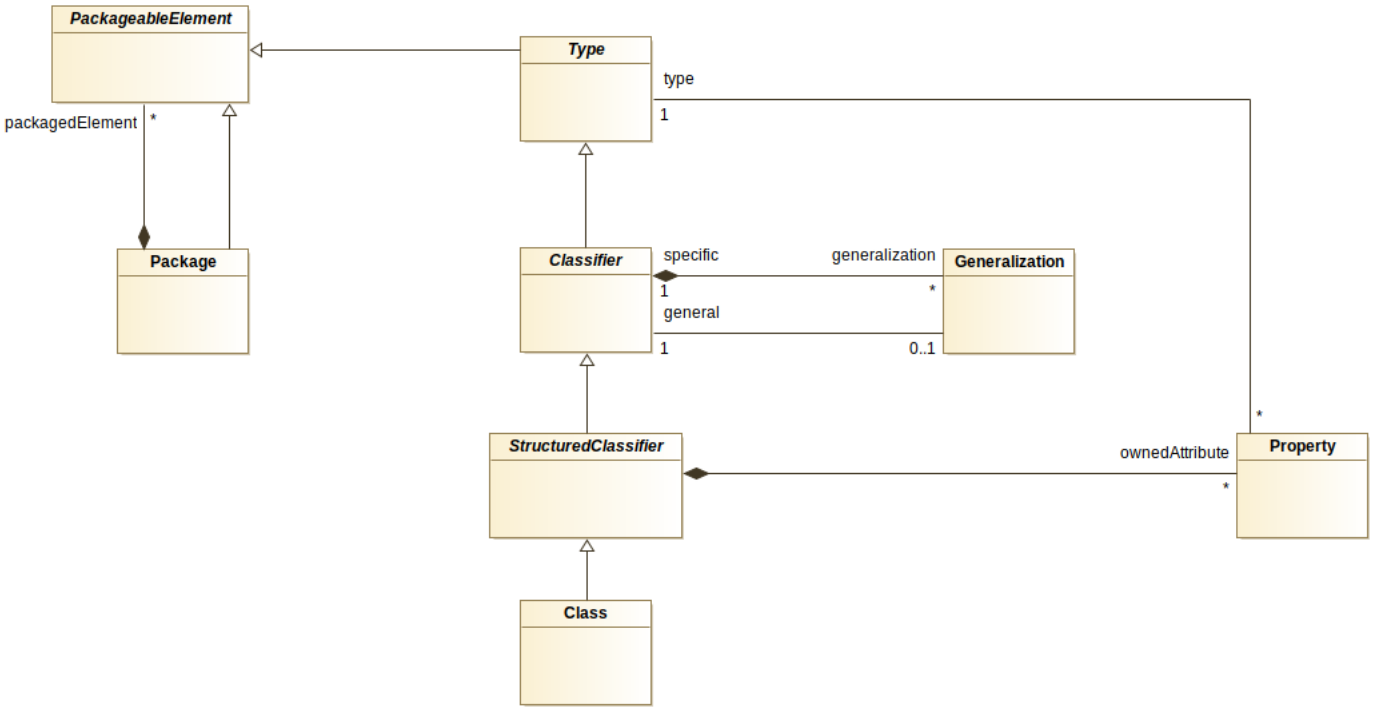
Déplacement de classe d'un package à un autre

Nous allons ici une première méthode de refactoring simplissime qui consiste à déplacer une classe d'un package à un autre.

Cela nécessite de connaître la façon dont les classes et les packages sont liés, et plus précisément, par quel biais une classe est-elle contenue dans un package. Une solution est d'explorer le métamodèle UML, ce que vous ferez plus tard dans cette UE. Nous allons ici le déduire d'un exemple, qui nous servira également de modèle de test.

- Créer un modèle papyrus InputMove, qui nous servira de modèle d'entrée pour tester le déplacement. Placez-y deux package p1 et p2, et une classe A dans le package p2.
- Ouvrez le InputMove.uml avec un éditeur de texte. Vous y voyez que la classe A est contenue dans le package p1 par un lien packagedElement. Un extrait du métamodèle vous est donné à la figure **??**, où l'on voit que packagedElement est une composition.
- Écrire une méthode de réusinage qui permet de déplacer une classe UML d'un package UML à un autre : cette méthode prend en paramètre une classe UML, et un package UML cible, et déplace la classe dans le package.
- Pour tester 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). Le code correspondant vous est donné sur moodle. Le scénario de test consistera à charger votre modèle InputMove.uml, y rechercher la classe A et le package p2 à partir de leur nom, à application votre refactoring pour déplacer A vers p2, puis à enregistrer le modèle obtenu dans un nouveau fichier suffixé par uml.

FIGURE 1 – Les packages



— Pour visualiser le modèle obtenu, il faut demander à Papyrus de construire une vue graphique (un diagramme de classes) pour le modèle généré. Pour cela :
  — Clic droit sur le fichier .uml généré, puis new Papyrus model
  — Dans les fenêtres de création de modèle papyrus à partir d'un modèle existant, choisir un diagramme de classe.
  — Le diagramme de classes s'ouvre vide, pour visualiser les éléments du modèle : cliquer sur l'icone avec le petit fantôme, et choisir "synchonized with model".

D'autres réusinages moins triviaux

Ecrivez sur le même principe les refactorings suivants. Des extraits du métamodèle UML vous sont donnés sur moodle pour vous guider.

1. Écrire une méthode de réusinage qui remplace un attribut public par un attribut privé et une paire d'accesseurs (en lecture et en écriture).
2. Écrire une méthode de réusinage qui fait "remonter" une méthode dans une super-classe. Cette méthode prend en paramètre la classe d'origine, la super classe cible, et le nom de la méthode à faire remonter. Si la superclasse ne correspond pas à une super-classe de la classe origine, si la méthode donnée ne correspond pas à une méthode de la classe d'origine, ou si une méthode concrète du même nom (pour simplifier) que la méthode à faire remonter existe déjà dans la super-classe spécifiée, alors le réusinage échoue.