Règles de correspondance du Reverse Engineering

Ci-dessous les règles de correspondance entre les concepts Java et les concepts UML que nous utilisons pour automatiser le reverse engineering dans ce cours. Chaque règle est définie par un numéro.

1. À toute **classe Java** doit correspondre une **classe UML** portant le même nom que la classe Java.
2. À toute **interface Java** doit correspondre une **interface UML** portant le même nom que l'interface Java.
3. À tout **attribut d'une classe Java** doit correspondre **une propriété appartenant à la classe UML correspondant à la classe Java**. Le nom de la propriété doit être le même que le nom de l'attribut. Le type de la propriété doit être une correspondance UML du type de l'attribut Java. Si l'attribut est un tableau, la propriété peut avoir plusieurs valeurs (en fonction de la taille du tableau).
4. À toute **méthode d'une classe Java** doit correspondre **une opération UML** appartenant à la classe UML correspondant à la classe Java. Le nom de l'opération UML doit être le même que celui de l'opération Java. Pour chaque paramètre de l'opération Java doit correspondre un paramètre UML de même nom, dont la direction est in et dont le type est le type UML correspondant au type du paramètre Java.
5. Si une classe Java appartient à **un package Java**, ce dernier doit correspondre à **un package UML** correspondant au package Java qui doit contenir la classe UML correspondant à la classe Java.
6. Si une classe Java **importe un package Java**, ce dernier doit correspondre à une relation **d'import entre le package UML** de la classe UML correspondant à la classe Java et le package UML correspondant au package Java importé.
7. Si une classe Java **hérite** d'une autre classe Java, les classes UML correspondantes doivent avoir elles aussi une relation **d'héritage UML** (appelée aussi généralisation).
8. Si une classe Java **réalise une interface**, la classe UML correspondante doit aussi **réaliser l'interface UML** correspondante.
9. Si une **opération Java** possède un **code de traitement**, alors doit correspondre **une note UML contenant ce code** et qui doit être attachée à l'opération UML correspondant à l'opération Java.

Nous pouvons considérer qu'il est possible de construire une association dans le modèle UML plutôt qu'une propriété lorsqu'une classe Java référence une autre classe Java.

La **règle n° 3** deviendrait alors :

3. À tout attribut d'une classe Java, nous distinguons trois cas en fonction du type de cet attribut :

   - Si le type de l'attribut est **un type primitif**, l'attribut doit correspondre **une propriété** appartenant à la classe UML correspondant à la classe Java. Le nom de la propriété doit être le même que le nom de l'attribut. Le type de la propriété doit être une correspondance UML du type de l'attribut Java
   - Si le type de l'attribut **est une autre classe Java**, l'attribut doit correspondre à **une association UML** entre la classe UML correspondant à la classe Java de l'attribut et la classe UML correspondant au type de l'attribut Java. Cette association doit être navigable vers la classe UML correspondant au type de l'attribut Java. Le nom de rôle de la classe correspondant au type de l'attribut doit être le même que le nom de l'attribut Java (multiplicité = 0..1. ou celle du tableau).
   - Si le type de l'attribut est une **collection générique** (exemple ArrayList<B>), l'attribut doit correspondre une **association UML** entre la classe UML correspondant à la classe Java de l'attribut et la classe UML correspondant au type spécifié dans la collection générique de l'attribut Java (i.e B dans l'exemple). La multiplicité sera 0..*. Une **note UML** est ajoutée à l'association pour préciser le nom exacte de la collection générique.