Projet MiniSpec

Mathis Baubriaud et Juliette Grosset

6 janvier 2021

1 Introduction

Un langage de modélisation de données permet la spécification d'entités représentant la structure des objets d'un domaine et la spécification des relations entre ces entités. Dans cet exercice, nous nous intéressons à un exemple d'un tel langage appelé minispec.

Dans minispec, une entité comprend un ensemble d'attributs typés. Les propriétés des entités sont modélisées par des attributs de type simple (String, Integer, Real, Boolean) ou par des collections (List ou Array) dont les élé- ments sont de type simple. Les relations entre entités sont modélisées par des attributs dont le type est une autre entité ou par des collections dont les éléments sont de type entité.

2 Méta-modèle en UML pour le langage de modélisation de données minispec

2.1 Diagramme UML du package utils

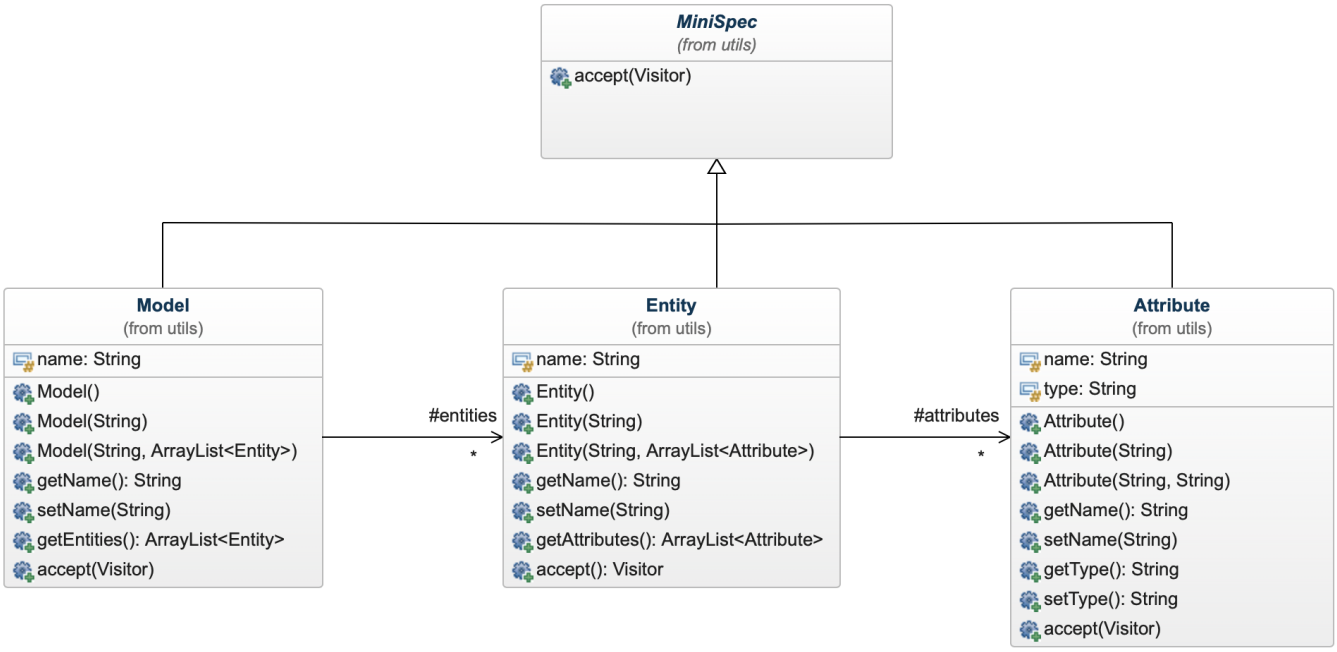

FIGURE 1 – Diagramme de classe package utils

C'est la représentation en diagramme UML de notre métamodèle du langage MiniSpec.

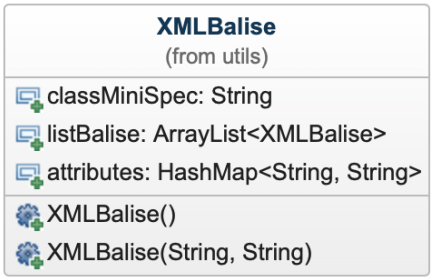

FIGURE 2 – Diagramme de la classe XMLBalise

La classe XMLBalise ci-dessus définit dans le package utils, nous est utile pour la classe DeserializerXML représenté dans la figure 4.

2.2 Diagramme UML du package tools

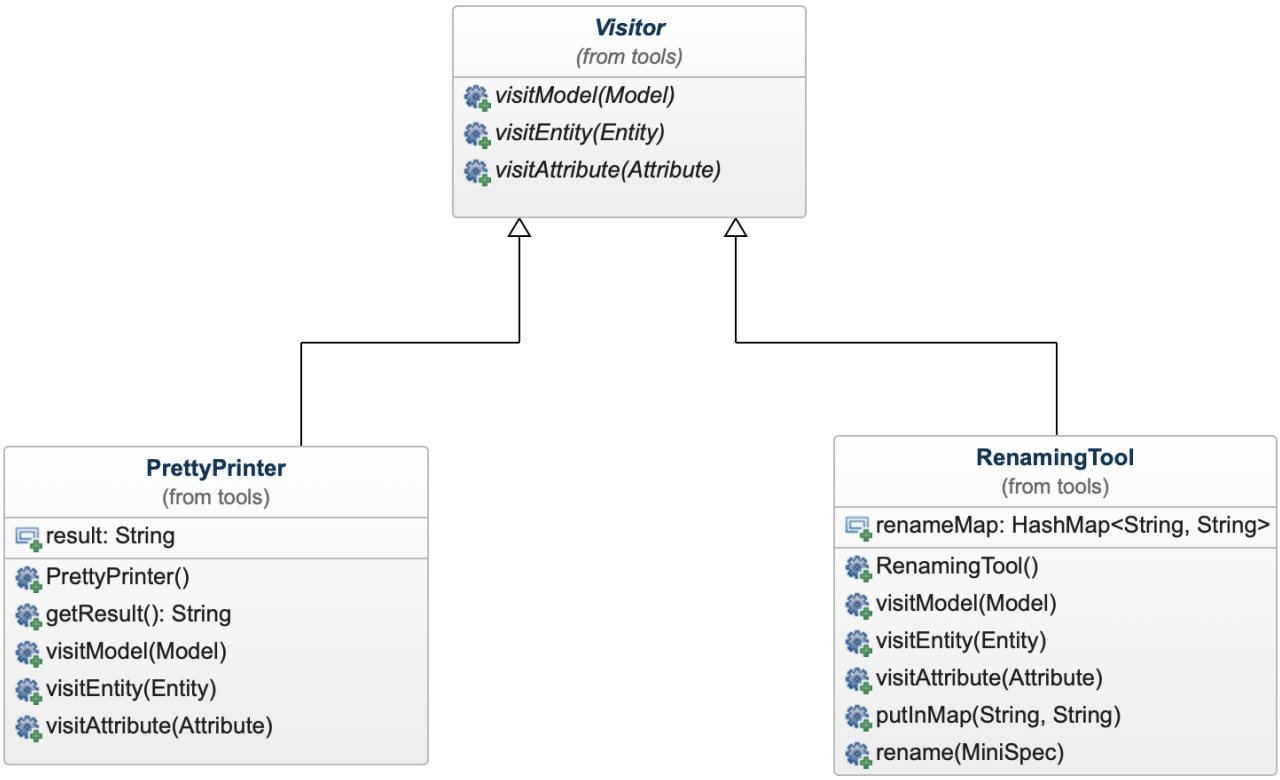

FIGURE 3 – Diagramme de classe des outils PrettyPrinter et RenamingTool

La classe Java PrettyPrinter qui permet de produire une représentation textuelle lisible dans sa syntaxe concrète.

La classe RenamingTool permet de renommer un modèle, une entité ou un attribut. La fonction putInMap prend en entrée deux strings. Le premier string correspond au nom de l'élément à renommer. Le second string correspond au nouveau nom de l'élément à renommer.

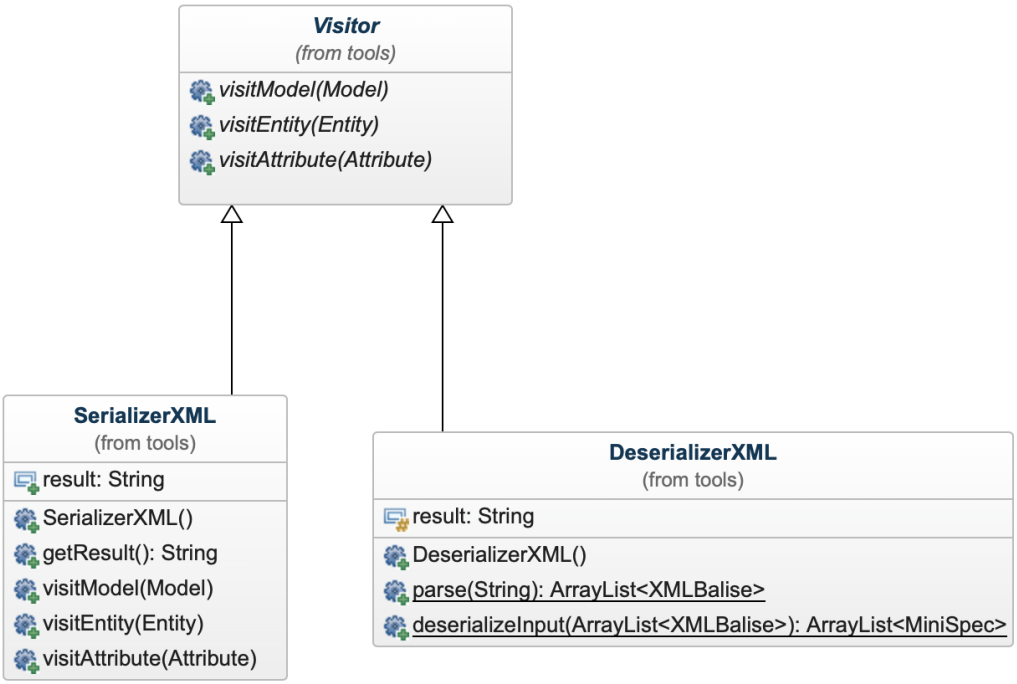

FIGURE 4 – Diagramme de classe des outils de sérialisation et désérialisation XML

La classe SerializerXML permet de sérialiser vers une représentation XML.

De manière inverse, la classe DeserializerXML qui utilise la classe utilitaire XMLBalise du package utils, permet de désérializer une représentation XML à l'aide d'une fonction parser.

3 Tests JUnit des différents outils

```
Test of the tool : Pretty Printer
---------------------------------

model Flotte;
        entity Satellite ;
                nom : String ;
                id : Integer ;
        end_entity ;
end_model;

Test of the tool PrettyPrinter réussi !
```

FIGURE 5 – Test JUnit de l'outil PrettyPrinter

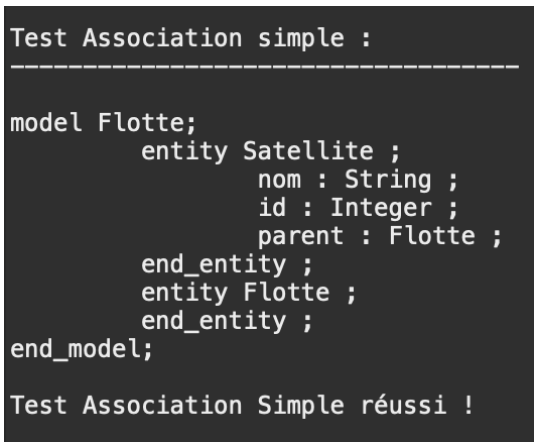

FIGURE 6 – Test JUnit montrant la possibilité de l'association simple

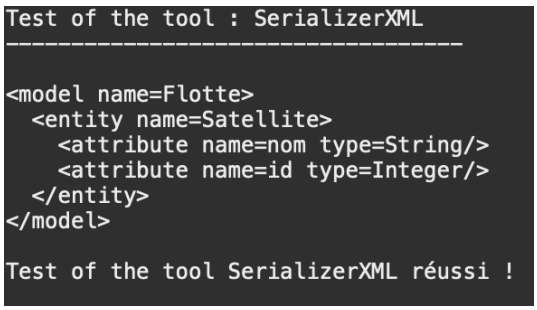

FIGURE 7 – Test JUnit Serialisation vers XML

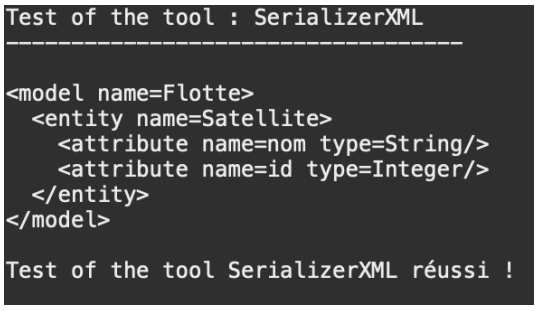

FIGURE 8 – Test JUnit Déserialisation depuis XML puis sérialisation vers XML en vérification

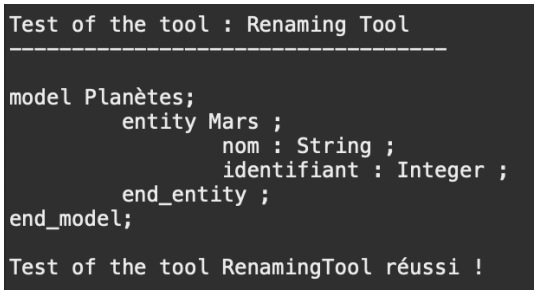

FIGURE 9 – Test JUnit Déserialisation depuis XML puis sérialisation vers XML en vérification