UNIVERSITÉ DE RENNES 1

CAO -- Ingénierie Dirigée par les Modèles

TP2 – Modification d'un modèle et Aspects

1. Modification d'un modèle

On veut rajouter dans la classe Value un entier représentant la valeur numérique de l'expression. Ecrivez les opérations Kermeta dans votre classe EcoreDocHelper pour ajouter un attribut « val » à la classe Value. Vous devrez fournir un getter et un setter à cet attribut et définir le type « Integer » de la même façon que le type « String ».

Chargement du modèle : Dans la méthode `main()` de la classe EcoreDocHelper précédemment créée, écrivez le nécessaire pour charger le modèle à l'aide de la classe utilitaire EcoreUtil fournie.

Ecrivez des opérations Kermeta pour :

- Ajouter un attribut à une classe
- Ajouter un datatype à un package
- Ajouter une opération à une classe

Utilisez les opérations créées pour ajouter (à appeler dans votre main):

- l'attribut val de type Integer à la classe Value
- l'opération getVal qui renvoie un Integer et l'opération setVal qui prend un Integer en paramètre et renvoie void
- le datatype Integer à votre package racine

Sauvegarde du modèle : Dans la méthode `main()` de la classe EcoreDocHelper précédemment créée, écrivez le nécessaire pour sauvegarder votre modèle modifié dans un autre fichier (i.e tableur_modified.ecore) à l'aide de la classe utilitaire EcoreUtil fournie.

2. Intégration d'un patron de conception (Visitor) en parcourant le modèle

Une manière d'implémenter le calcul de la valeur d'une cellule est d'appliquer le patron Visiteur, et d'écrire un visiteur concret pour l'évaluation. L'objectif est donc d'ajouter à votre modèle un nouveau paquetage « Visitors » qui contiendra une interface « Visitable » et une classe abstraite « Visitor ». Vous devrez ajouter l'interface, la classe abstraite et les opérations nécessaires.

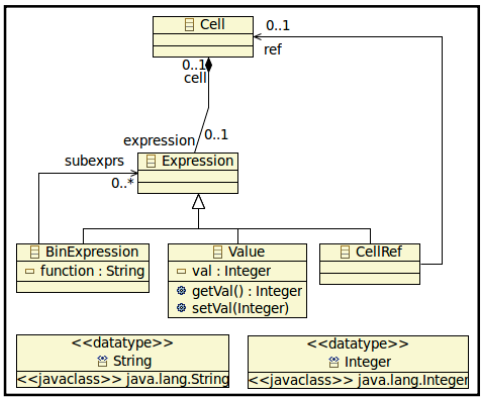

Figure 1. Modèle d'évaluation des cellules dans un tableur (modifié)

Création du visiteur : En Kermeta, écrivez les opérations nécessaires pour ajouter une interface Visitable et une classe abstraite Visitor. Ajoutez respectivement l'opération ***accept(Visitor v)*** dans l'interface et l'opération ***visit(Visitable v)*** dans la classe abstraite.

Création des opérations : Pour chaque classe de votre modèle, créez l'opération ***accept(Visitor v)*** et ajoutez le code Java sous forme d'annotations Ecore.

Pour toutes les classes Foo de votre modèle, créez l'opération ***visitFoo(Foo v)*** dans le visiteur.

Création du visiteur concret : Ajoutez un visiteur concret (i.e MyVisitor) qui hérite du visiteur abstrait.

Création du paquetage « Visitors » : Ecrivez une méthode en Kermeta qui créé un nouveau paquetage et l'ajoute à la racine de votre modèle. Vous utiliserez cette fonction dans votre `main()` et y ajouterez l'interface, la classe abstraite du visiteur et la classe concrete nouvellement créées.

Sauvegardez votre modèle obtenu dans un nouveau modèle **tableur_visitable.ecore.**

3. Génération des aspects mettant en oeuvre le patron de conception Visitor

Un visiteur peut également être implémenté par aspects.
L'objectif de cette partie est de générer du code Kermeta qui va contenir les aspects permettant d'implémenter le visiteur.

Génération des aspects :

1. Dans la classe EcoreDocHelper, créez une opération ***generate_visitor()*** dans laquelle vous allez générer le code Kermeta du fichier « tableur_visitors.kmt ». Ce code Kermeta doit définir le paquetage « tableur::visitors », l'interface « Visitable » et la classe « Visitor ». Vous devez également générer la méthode ***accept(Visitor)*** dans l'interface « Visitable » ainsi que les méthodes ***visitFoo(X)*** dans la classe « Visitor » pour chaque classe de votre modèle. Générer également un visiteur concret (i.e MyVisitor) qui hérite du visiteur abstrait.
2. Dans votre classe EcoreDocHelper, créez une seconde opération ***generate_aspects()*** qui prend en paramètre votre modèle. La méthode ***generate_aspects()*** produit le code Kermeta du fichier « tableur.kmt » qui déclare des aspects sur les classes X et qui implémente le code de l'opération ***accept(Visitor)*** dans le paquetage « tableur ».
3. Les méthodes ***visitFoo(X)*** dans la classe Visitor doivent afficher « Visiting X » et faire le parcours du visiteur (i.e l'arborescence du modèle).
4. Créez un nouveau fichier Kermeta et importez les deux fichiers nouvellement créés.
    1. Récupérez le modèle « Cell.xmi » et copiez le dans le répertoire « model ».
    2. Enregistrez votre métamodèle « Tableur » dans le repository EMF (Click droit -> Epackages registration -> Register EPackages into repository)
    3. Inspirez-vous du helper EcoreUtil pour créer une opération loadTableurModel qui charge votre modèle précédemment créé (Utilisez la méthode ***createResource(uri,meta_uri)*** à la place du ***getResource(uri)***).
    4. Faites un load de votre modèle et visitez-le (avec les méthodes fournies par votre classe concrete MyVisitor).

Tableur.kmt

```
package tableur;
require "../../metamodel/tableur_modified.ecore"
require "tableur_visitors.kmt"
require kermeta
using tableur::visitors

aspect class Foo inherits Visitable{
    method accept(v : Visitor)  from Visitable
    is do
        ...
    end
}
```

Tableur_visitors.kmt
Tableur_visitors.kmt

```
package tableur::visitors;
require "tableur.kmt"
require kermeta
using tableur

abstract class Visitor {
        ...
}

class Visitable {
        ...
}
```