# TD - Object Constraint Language (OCL) appliqué sur le méta-modèle OCL

Ingénierie Dirigée par les Modèles
Master informatique - Université de Montpellier - Faculté Des Sciences

13 novembre 2022

Dans ce TD, nous étudions des contrainte OCL qui servent à préciser le méta-modèle UML 2.5. Dans la première partie, ces contraintes ne demandent pas de manipuler des collections OCL. Dans la deuxième et la troisième partie, elles les utiliseront.

## 1 Quelques contraintes sur le *Multiplicities Diagram*

Le diagramme de la figure 1 représente les éléments typés et les multiplicités en OCL.

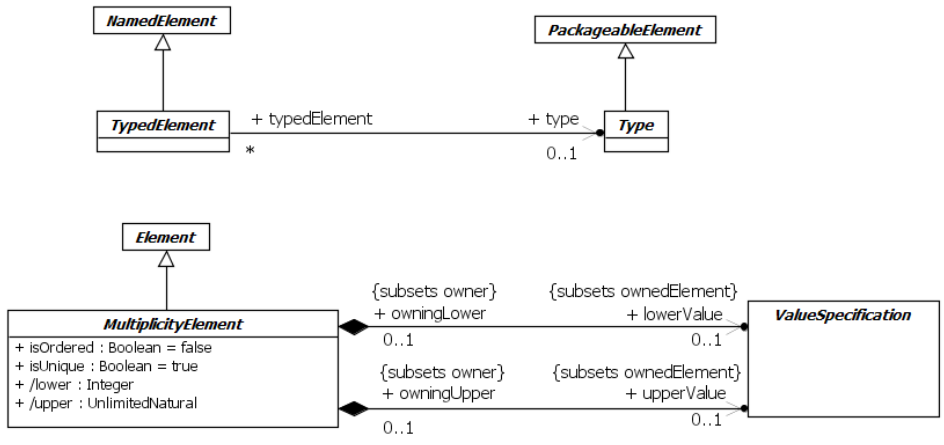


FIGURE 1 – Types et Multiplicités

Il est accompagné de plusieurs requêtes textuelles. La requête `lowerBound()` retourne la borne inférieure de multiplicité sous forme d'un entier si elle est spécifiée, 1 sinon.

```
context MultiplicityElement
  def : lowerBound() : Integer =
        if (lowerValue=null or lowerValue.integerValue()=null) then 1
        else lowerValue.integerValue() endif
```

La requête `upperBound()` retourne la borne supérieure de multiplicité sous forme d'un entier (éventuellement * pour un nombre indéfini) si elle est spécifiée, 1 sinon.

```
context MultiplicityElement
  def : upperBound() : UnlimitedNatural =
        if (upperValue=null or upperValue.unlimitedValue()=null) then 1
        else upperValue.unlimitedValue() endif
```

**Question 1.1** *Écrire les contraintes suivantes :*

- *La borne inférieure doit être positive ou nulle.*
- *La borne supérieure doit être supérieure à la borne inférieure.*
- *La valeur dérivée de* `/lower` *doit être égale à la borne inférieure donnée par la requête ci-dessus.*
- *La valeur dérivée de* `/upper` *doit être égale à la borne supérieure par la requête ci-dessus.*
- *La requête* `isMultivalued()` *retourne vrai si la propriété peut prendre plus d'une valeur ; elle ne s'applique que lorsqu'une borne supérieure a été spécifiée.*
- *La requête* `includesMultiplicity(M: MultiplicityElement)` *retourne vrai si la multiplicité de l'élément inclut $M$. Vous devez déterminer les conditions d'application.*

## 2 Quelques requêtes sur le *Feature Diagram*

*Le diagramme de la figure 2 représente les features en OCL.*

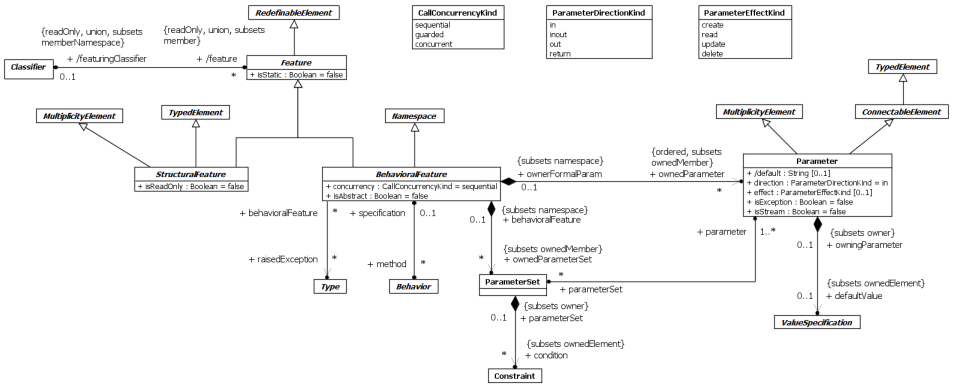


FIGURE 2 – Features

***Question 2.1*** *Les attributs sont des Properties qui sont des StructuralFeatures. Les classifiers et les features spécialisent la méta-classe* `NamedElement` *qui dispose d'un attribut* `name:String`*. Les classes sont des classifiers. Dessinez, sous forme de diagramme d'instance du métamodèle, un modèle comportant une classe* `Personne` *avec les* `StructuralFeatures` *suivantes :* `nom` *de type* `String`*, dont l'ensemble de valeurs est exactement de taille 1 ;* `adresses` *de type* `String`*, dont l'ensemble de valeurs, qui représente les adresses successives de la personne, est une séquence qui peut être vide ;* `parents`*, de type* `Personne`*, dont l'ensemble de valeurs peut être de taille 0 à 4 (il peut contenir les parents adoptifs et les parents biologiques).*

***Question 2.2*** *Écrire, dans le contexte d'un* `Classifier` *ou d'un sous-contexte à préciser, et à l'aide de la construction* `def` *les éléments suivants :*

1. *Requête retournant les features* `static`.
2. *Requête retournant les BehavioralFeatures abstraites.*
3. *Contrainte indiquant que les BehavioralFeatures abstraites n'ont pas de méthode associée.*
4. *Requête retournant les StructuralFeatures dont la collection des valeurs est un* `bag`.
5. *Requête retournant les BehavioralFeatures ordonnées par leur nombre de paramètres.*
6. *Requête retournant les noms des BehavioralFeatures dont tous les paramètres sont passés avec la direction* `in` *ou* `return`.

# 3 Quelques requêtes sur le *Classifier Diagram*

*Le diagramme de la figure 3 représente les classifiers et la relation de spécialisation/généralisation en OCL.*

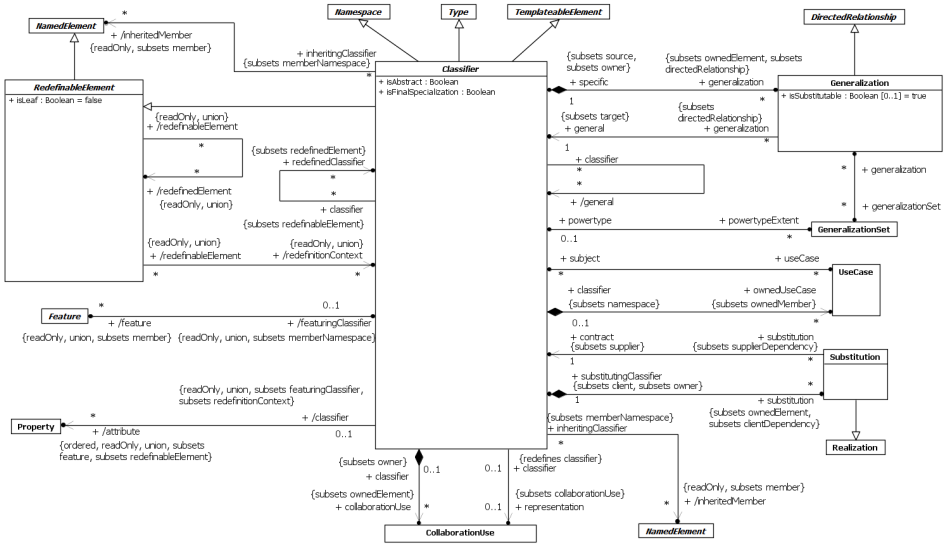


FIGURE 3 – Classifiers et Generalization

***Question 3.1*** *Dessinez, sous forme de diagramme d'instance du métamodèle, un modèle comportant une classe* `Polygone`*, ses sous-classes* `Rectangle` *et* `Losange`*, et la classe* `Carré` *sous-classe de* `Rectangle` *et* `Losange`*.*

***Question 3.2*** *Écrire, dans le contexte d'un* `Classifier` *ou d'un sous-contexte à préciser et à l'aide de la construction* `def` *les éléments suivants :*

1. *Requête* `parents():Classifier [0..*]` *retournant tous les successeurs immédiats (généralisations) d'un classifier.*
2. *Requête* `allParents():Classifier [0..*]` *retournant tous les successeurs (généralisations) d'un classifier.*
3. *Contrainte indiquant que* `/general` *est égal aux successeurs immédiats.*
4. *Contrainte indiquant qu'il n'y a pas de circuit dans les généralisations.*