OCL : Liaison avec UML

Contexte

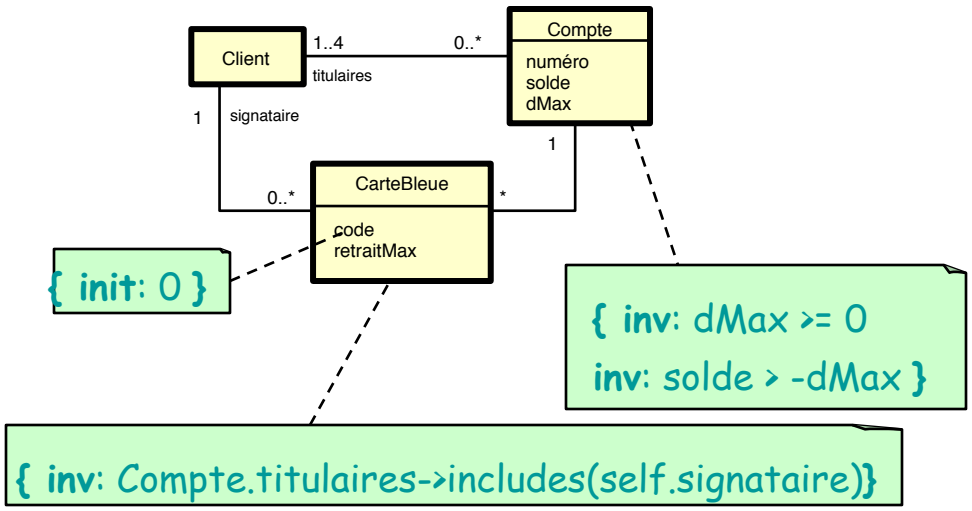
Invariant, Propriété dérivée, Valeur d'initialisation,
Précondition, Postcondition, Corps de méthode

Accès aux attributs et opérations

Navigation

Contexte d'une contrainte

- Contrainte toujours associée à un élément de modèle :
  le contexte de la contrainte.
- Deux techniques pour spécifier le contexte :



context Compte
inv: dmax >=0
inv: solde > -dMax

context CarteBleue
inv: Compte.titulaires->includes(self.signataire)
inv: code>0 and code<=9999
inv: retraitMax>10

context Compte::solde : integer
init: floor(depotInitial * 10 / 100)

Opérations spécifiques au modèle objet d 'UML

- accès à un attribut, à une opération
- navigation
  - VIA une association
  - VERS une classe associative
  - DEPUIS une classe associative
  - VIA un association qualifiée
- accès au type et super types
- accès aux instances d'une classe (extension)
- accès à l'état d'un objet
- constructions pour les post conditions

Accès à un attribut
Accès à une méthode

objet . attribut

- Accès à un attribut

self.dateDeNaissance

objet . méthode(expr1, expr2, …)

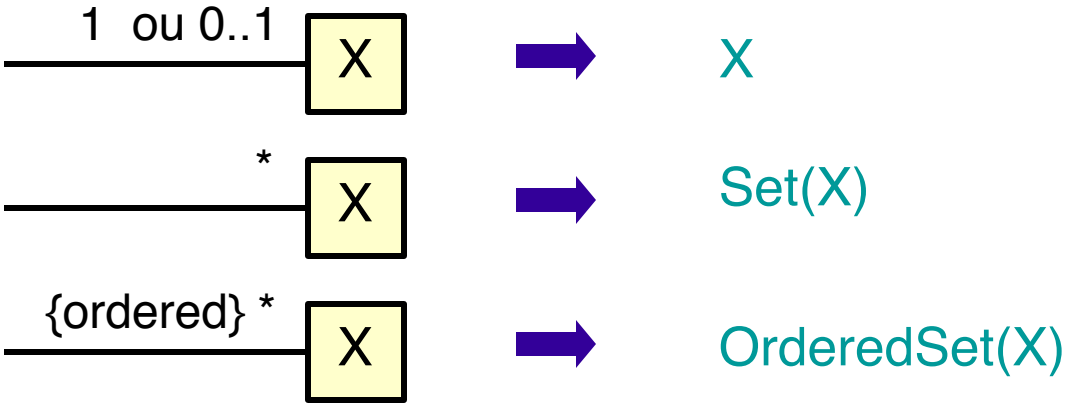
- Accès à une méthode sans effet de bord (**{query}**)

self.impôts(1998)

Navigation VIA une association

objet . nomderole

- Accéder à l 'ensemble des objets liés à un objet donné
- Le type du résultat dépend de la cardinalité et de {ordered}

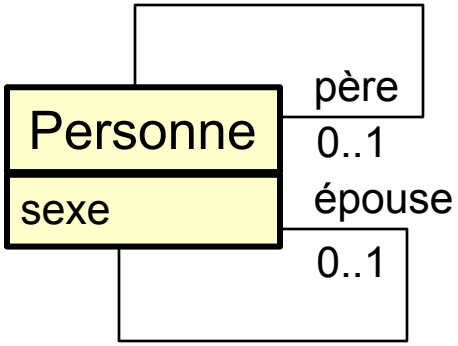


self.père	est de type	Personne
self.voitures	est de type	Set(Voiture)
self.enfants	est de type	OrderedSet(Personne)

Navigation VIA une association

- Rappel: un élément est converti en singleton lorsqu 'une opération sur collection est appliquée

self.père->**size()** = 1



- Permet de tester si la valeur est définie
(l'ensemble vide représente la valeur indéfinie)

self.père->**isEmpty()**

self.épouse->**notEmpty() implies self**.épouse.sexe = Sexe::féminin

- Si une association n 'a pas de nom de rôle alors on peut utiliser le nom de la classe destination

Navigation VERS une association

objet . nomdassociation

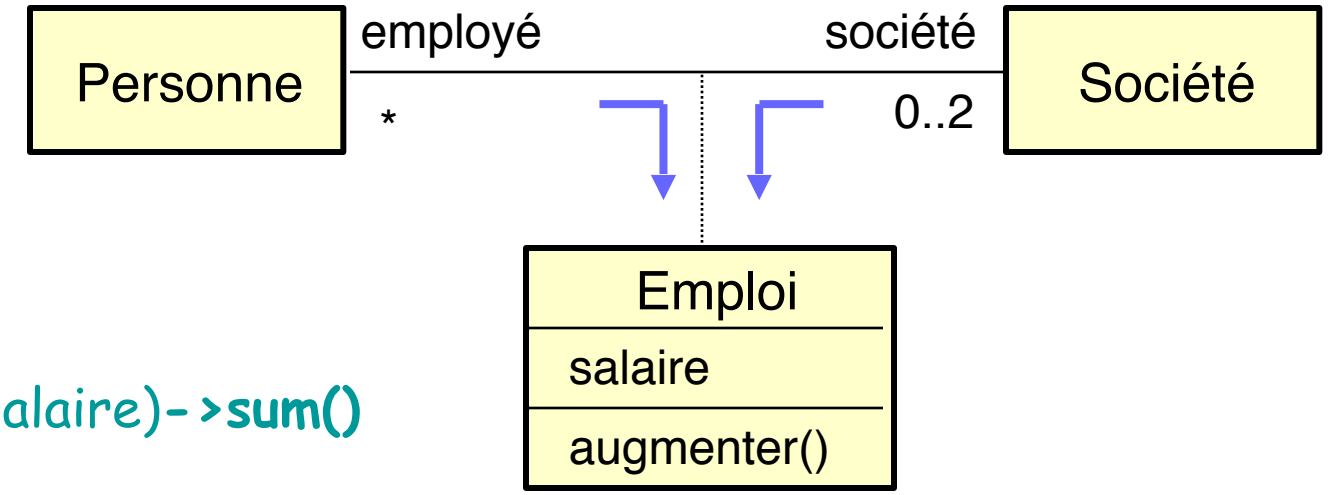
- Permet d 'accéder à l 'ensemble des liens

s.emploi

p.emploi

s.emploi->collect(salaire)->sum()

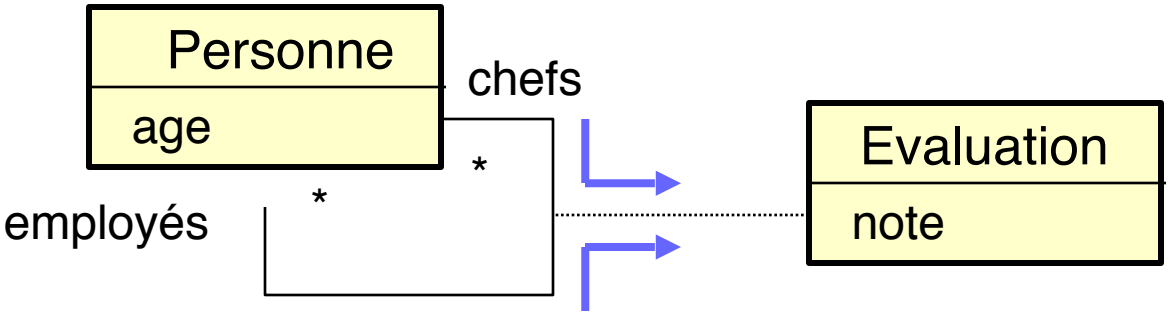
s.emploi.salaire->forall(x | x>500)



Navigation
VERS une association réflexive

objet . nomdassociation [nomderole]

- Si l 'association est réflexive il faut indiquer le sens de parcours de l 'association (pour éviter l'ambiguité)



p.Evaluation[chefs]
p.Evaluation[employés]
p.Evaluation[chefs].note -> **sum()**

Opérations concernant les types

objet . **oclIsTypeOf(type)**

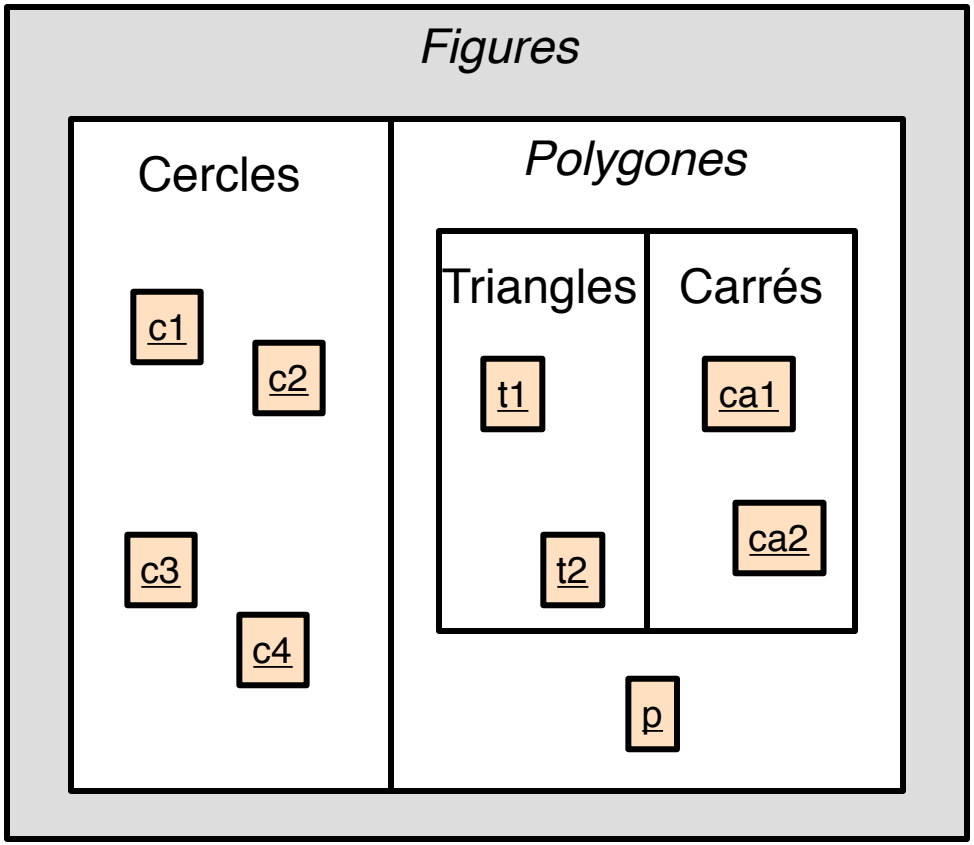
- type direct

objet . **oclIsKindOf(type)**

- type direct ou supertypes

objet . **oclAsType(type)**

- Conversion de type (casting)



Opérations concernant les types

- Contraintes sur les types

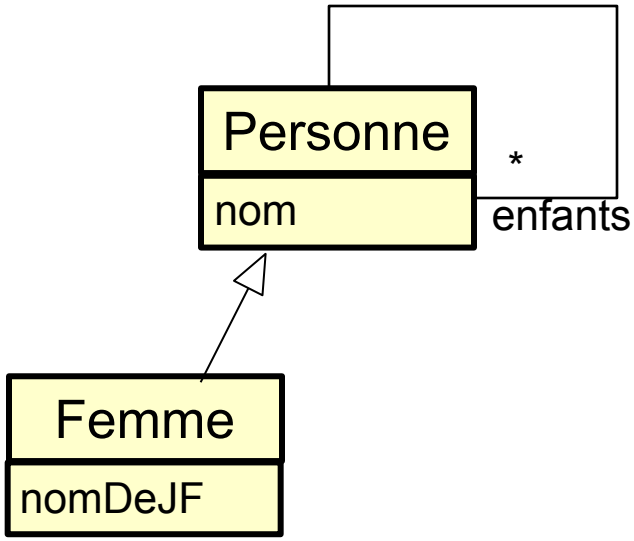
```
p.enfants->select(oclIsTypeOf(Femme))
p.enfants->select(oclIsKindOf(Femme))
```

- Accès aux propriétés redéfinies

```
e.oclAsType(Person).nom
```

- Typage dynamique

```
p.enfants->select(oclIsKindOf(Femme))
    .asTypeOf(Set(Femme))
    ->select(nomDeJF <> nom)
```



Opérations définies sur les classes

Classe.propdeclasse

- Permet d 'accéder aux propriétés de classes

Classe.allInstances

- Retourne l 'ensemble des instances de la classe c 'est à dire l 'extension de cette classe

```
Personne.allInstances->size() < 500
Personne.allInstances->forall(p1,p2 |
        p1<>p2 implies p1.numsecu <> p2.numsecu)
Personne.allInstances->isUnique(numsecu)
```

Où utiliser OCL

- OCL peut être utilisé pour décrire des prédicats
  - **inv:** invariants de classes
  - **pre:** pré-conditions d 'opérations
  - **post:** post-conditions d 'opérations

  inv: solde < decouvertMax
  pre: montantARetirer > 0
  post: solde > solde@pre

- OCL peut également être utilisé pour décrire des expressions
  - **def:** déclarer des attributs ou des opérations
  - **init:** spécifier la valeur initiale des attributs
  - **body:** exprimer le corps de méthodes {query}
  - **derive:** définir des élements dérivés (/)

  def: nbEnfants:Integer
  init: enfants->size()
  body: enfants->select(age< a)
  derive: age<18

Invariants (inv)

- Prédicat associé à une classe ou une association
- Doit être vérifié à tout instant
- Le contexte est défini par un objet
  - cet objet peut être référencé par **self**
  - l 'objet peut être nommé explicitement (possibilité supprimée en UML2.0?)
- L 'invariant peut être nommé

context Personne

inv pasTropVieux : age < 110

inv : self.age >= 0

Exemples d'invariants (inv)

context Personne

inv : age>0 and self.age<110

inv mariageLégal : marié **implies** age > 16

inv enfantsOk : enfants->size() < 20

inv : not enfants-> includes(self)

inv : enfants->includesAll(filles)

inv : enfants->forall(e | self.age - e.age < 14)

Expression de propriétés dérivées (derive)

- Préciser en OCL la valeur d'un attribut ou d'une association dérivée
- Complète la notation /

- context Personne::estMarié : Boolean
  derive : conjoint->notEmpty

- context Personne::filles : Set(Personne)
  derive : enfants->select(sexe = Sexe::Feminin)

- context Personne::grandParents : Set(Personne)
  derive: parents.parents->asSet()

Expression du corps d'une méthode (body)

- Description en OCL d'une méthode sans effet de bord ({isQuery})
- Equivalent à une requête

```
context Personne:acf( p : Personne ) : OrderedSet(Personne)
    body : self.ancestres()->intersection(p.ancestres())
                ->select(sexe = Sexe::Feminin)->sortedBy(dateDeNaissance)

context Personne
    def: ancestres : Set(Personne)
            = parents->union(parents.ancestres->asSet())
```

Pré-conditions et post-conditions (pre, post)

- Prédicats associés à une opération
  - les pré-conditions doivent être vérifiées avant l 'exécution
  - les post-conditions sont vraies après l 'exécution
  - **self** désigne l 'objet sur lequel l 'opération à lieu
- Dans une post-condition :
  - **@pre** permet de faire référence à la valeur avant l 'opération
  - **result** designe le resultat
  - **ocsIsNew()** indique si un objet n'existait pas dans l'état précédent

context Type::opération(param1 : Type1, ...) : Type

pre nom1 : param1 < ...

pre nom2 : ...

post nom2 : ... **result** > ...

Exemples

context Personne::retirer(montant : Integer)
pre : montant > 0
post : solde < solde@pre - montant

context Personne::salaire() : integer
post : result >= Legislation::salaireMinimum

context Compagnie::embaucheEmployé(p : Personne) : Contrat
pre pasPrésent : **not** employés->includes(p)
post embauché : employés = employés@pre->including(p)
post : result.oclIsNew()
post : result.compagnie = self and result.employé = p

Exercices

- Spécifier une fonction qui retourne la racine carrée d'un réel

context Math::Sqrt(x : Real) : Real

pre:

post:

- Partie entière de la racine carrée d'un entier

context Math::ISqrt(x : Integer) : Integer

pre:

post: