INFO-H-302 Analyse et conception par objets
Object Constraint Language

F. Servais et B. Verhaegen

4 mars 2010

Object Constraint Language

- Langage de contraintes UML
- Permet de spécifier entre autre des invariants sur les classes et des pré- et post-conditions sur les méthodes
- But : enrichir les diagrammes UML

Exemple d'invariant simple

Le sexe d'une personne est soit 'M' soit 'F'

```
context Person inv:
  self.sex = 'M' or self.sex = 'F'
```

L'age d'une personne est positif

```
context Person inv:
  self.age() > 0
```

- toujours spécifier le contexte (instance d'un type)
- `self` référence l'instance courante
- `inv : <expression booléene>` définit un invariant

Opérations de base

Booléens

`and`, `or`, `xor`, `not`, `implies`, ...

Nombres

`+`, `-`, `*`, `/`, `abs`, `div`, `mod`, `max`, `min`, `<`, `>`, `<=`, `>=`, `=`, `<>`, ...

Opérations sur les collections

- `size()`
- `count()`
- `includes(aValue)`
- `excludes(aValue)`
- `isEmpty()`
- `notEmpty()`
- ...

La liste des employés d'une société n'est pas vide

```
context Company inv:
  self.employees->notEmpty()
```

Remarquez l'utilisation de -> pour les opérations sur les collections et le . pour la navigation

Action sur les classes

- `allInstances()` renvoie une collection de toutes les instances d'une classe

Le nombre de personnes ne doit pas dépasser 100

```
context Person inv:
  Person.allInstances()->size() <= 100
```

Opérations sur une collection

- `select(expression booléenne)` renvoie une collection d'objets correspondant à une condition

Les personnes qui ont plus de 18 ans

```
Person.allInstances()->select(age()>= 18)
```

- `collect(attribut ou méthode)` dérive une collection à partir d'une autre

La liste des dates de naissances

```
Person.allInstances()->collect(birthdate)
```

Opérations sur une collection

- `forAll(expression booléenne)` spécifie une expression booléene qui doit être vraie pour tous les éléments d'une collection

Toutes les personnes doivent avoir des noms uniques

```
context Person inv:
  Person.allInstances()->forAll(p1, p2 |
          p1 <> p2 implies p1.name <> p2.name)
```