Modélisation Architecturale et Patrons de Conception (2025-2026) – M1M / CC 2

2 heures - projet Gradle - rendu fichier ZIP - diagrammes UML à la racine du projet

Contexte

On désire représenter et travailler avec des expressions. Une expression peut être : un booléen (vrai, faux), un nombre (..., -1, 0, 1, ...), un test d'infériorité (x < y), une conjonction (x et y), une soustraction (x - y), un calcul de minimum (min(x,y)) ou de maximum (max(x,y)) ou une conditionnelle (si c alors x sinon y).

Ainsi, les exemples suivants sont des expressions (composées d'autres expressions, avec des parenthèses pour aider la lecture) :

- min(1 + 2, (max(3, 4) + 5)) qui vaut `3`
- (si (1 < 2) alors (1+1) sinon (2+2)) + 3 qui vaut `5`
- (si (1 < 2) alors faux sinon vrai) et (2 < 1) qui vaut `false`
- si ((3 < 5) et (5 < 8)) alors min(5-3, 8-5) sinon max(3-5, 5-8) qui vaut `2`.

Il y a en pratique d'autres types d'expressions (négation, addition) mais on ne les prend pas en compte ici (pas au CC), c'est prévu dans une extension (maintenance). Il ne doit pas y avoir plusieurs instances des nombres (utiliser FLYWEIGHT).

Les opérations qu'on souhaite faire sur les expressions sont :

- obtenir sa valeur (`valeur()`), comme cela peut être un `Integer` ou un `Boolean`, il faut en théorie utiliser la généricité
- obtenir sa représentation textuelle (`toString()`)

ATTENTION : ne pas confondre une expression de type booléenne (par exemple : (max(1,2) < 2) et vrai) et sa valeur de type `Boolean` (par exemple : `false` pour l'exemple précédent) ni une expression de type entier (par exemple : min(1,2) + 3) et sa valeur de type `Integer` (par exemple : `4` pour l'exemple précédent).

Question 1 : COMPOSITE

1.1 – représenter l'arbre de syntaxe (les noeuds de l'arbre sont les expressions / sous-expressions) correspondant à `si ((3 < x) et (x < 8)) alors min(x-3, 8-x) sinon max(3-x, x-8)` quand `x` est `5`.

1.2 – modéliser les expressions à l'aide d'un diagramme de classe en utilisant le patron COMPOSITE.

1.3 – implanter votre modèle en Java.

1.4 –

écrire une `main` pour afficher (avec `toString()`) les expressions correspondant à celle de la question 1.1 quand `x` est pris dans `{ 1, 3, 5, 8, 12 }` et leur valeur (avec `valeur()`), le résultat attendu est :

```
expression : si (et(inf(3, 1), inf(1, 8)))
        alors min(sub(1, 3), sub(8, 1))
        sinon max(sub(3, 1), sub(1, 8))
valeur = 2
expression : si (et(inf(3, 3), inf(3, 8)))
        alors min(sub(3, 3), sub(8, 3))
        sinon max(sub(3, 3), sub(3, 8))
valeur = 0
expression : si (et(inf(3, 5), inf(5, 8)))
        alors min(sub(5, 3), sub(8, 5))
        sinon max(sub(3, 5), sub(5, 8))
valeur = 2
expression : si (et(inf(3, 8), inf(8, 8)))
        alors min(sub(8, 3), sub(8, 8))
        sinon max(sub(3, 8), sub(8, 8))
valeur = 0
expression : si (et(inf(3, 12), inf(12, 8)))
        alors min(sub(12, 3), sub(8, 12))
        sinon max(sub(3, 12), sub(12, 8))
valeur = 4
```

Question 2 : VISITEUR

Dans la suite du sujet on utilisera la notion de complexité d'une expression qui est définie comme suit :

- +0 pour un booléen ou un nombre
- +1 pour un test d'infériorité, une conjonction, une soustraction, un calcul de minimum ou de maximum
- +3 pour une conditionnelle

On peut imaginer dans la suite (maintenance, pas au CC) l'écriture d'autres notions similaires à calculer.

2.1 – proposer une extension de votre diagramme de classe de la question 1.2 pour prendre en compte le calcul de complexité par un VISITEUR.

2.2 – implanter votre modèle étendu en Java.

2.3 – écrire une `main` en modifiant celle de la question 1.4 calculant la complexité des expressions en plus (avec le visiteur correspondant), le résultat attendu est :

```
expression : si (et(inf(3, 1), inf(1, 8)))
        alors min(sub(1, 3), sub(8, 1))
        sinon max(sub(3, 1), sub(1, 8))
valeur = 2
complexité = 12
expression : si (et(inf(3, 3), inf(3, 8)))
        alors min(sub(3, 3), sub(8, 3))
        sinon max(sub(3, 3), sub(3, 8))
valeur = 0
complexité = 12
expression : si (et(inf(3, 5), inf(5, 8)))
        alors min(sub(5, 3), sub(8, 5))
        sinon max(sub(3, 5), sub(5, 8))
valeur = 2
complexité = 12
expression : si (et(inf(3, 8), inf(8, 8)))
        alors min(sub(8, 3), sub(8, 8))
        sinon max(sub(3, 8), sub(8, 8))
valeur = 0
complexité = 12
expression : si (et(inf(3, 12), inf(12, 8)))
        alors min(sub(12, 3), sub(8, 12))
        sinon max(sub(3, 12), sub(12, 8))
valeur = 4
complexité = 12
```