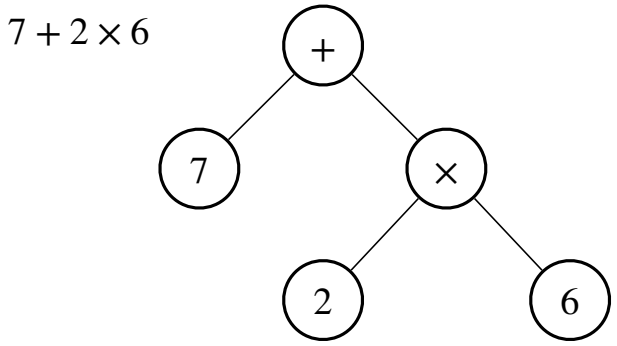
M. Gasquet, N. Lazaar, C. Nadal, N. Palleja, S. Robillard

Patrons Itérateur et Visiteur



L'objectif de cette étude c'est d'utiliser les patrons itérateur et visiteur pour représenter des expressions arithmétiques sous forme d'arbres :

Question 1 • Chaque expression peut être soit un nombre, soit une opération binaire (addition, soustraction ou multiplication) entre deux expressions. Créez la classe abstraite `Expression` et les classes concrètes `NumberExpression` et `BinaryExpression`.

Question 2 • Donnez la conception UML qui permet d'avoir au minimum deux itérateurs pour parcourir l'arbre qui représente une expression arithmétique : un pour un parcours en profondeur et un pour un parcours en largeur.

Question 3 • Créez une interface `ExpressionIterator` qui définit les méthodes nécessaires pour parcourir un arbre d'expression.

Question 4 • Créez une classe concrète `DepthFirstExpressionIterator` qui implémente l'interface `ExpressionIterator` et qui parcourt l'arbre d'expression en profondeur.

Question 5 • Créez une classe concrète `BreadthFirstExpressionIterator` qui implémente l'interface `ExpressionIterator` et qui parcourt l'arbre d'expression en largeur.

Question 6 • Donnez la conception UML qui permet d'avoir au minimum deux visiteurs des éléments qui composent une expression arithmétique.

Question 7 • Créez une interface `ExpressionVisitor` qui définit une méthode pour visiter chaque type d'expression.

Question 8 • Créez une classe concrète `EvaluateExpressionVisitor` qui implémente l'interface `ExpressionVisitor` et qui calcule la valeur d'une expression arithmétique.

Question 9 • Créez une classe concrète `OperatorExpressionVisitor` qui implémente l'interface `ExpressionVisitor` et qui affiche le nombre d'occurence de chaque opérateur arithmétique d'une expression.

La méthode principale dans la classe `App` permet de tester votre réalisation en construisant un arbre d'expression pour représenter une expression arithmétique simple : $(1+2)*(3-4)$, puis utilise les deux itérateurs pour parcourir l'arbre et utilise les deux visiteurs pour évaluer et afficher les opérateurs de l'expression.

Question 10 • Exécutez le test et créez de nouveau avec les expressions suivantes :

1. $8 + (3 * (5 - 10))$
2. $(5 * 10) - 4 + (15 - (3 * 7) + 18)$