Conception et programmation objets avancées

Sujet 2 : de la conception à la programmation

Exercice 1 On considère des expressions arithmétiques formées à partir de constantes réelles (nombres flottants) et utilisant les quatre opérations arithmétiques usuelles (addition, soustraction, multiplication et division). Une telle expression est, par exemple, $3 + ((4.7 + 2.3) * 5)$.

Ces expressions arithmétiques sont représentés par des arbres binaires. Les nœuds internes de l'arbre contiennent les opérateurs alors que les feuilles de l'arbre contiennent les constantes. Les fils gauche et droit d'un nœud interne représentent les deux sous-expressions gauche et droite. L'expression précédente est alors représentée par l'arbre donné Figure 1.

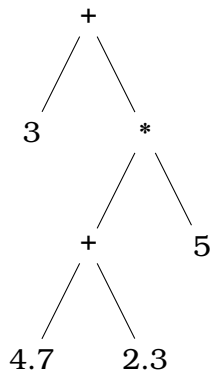

FIGURE 1 – Arbre de l'expression arithmétique $3 + ((4.7 + 2.3) * 5)$

Bien entendu, les expressions arithmétiques pourront être évaluées. Par exemple l'expression précédente vaut 38.

Pour représenter l'expression, on va créer des objets pour chacun des nœuds. Ces objets vont bien sûr être des instances de classes différentes suivant qu'il s'agisse de nœuds internes ou de feuilles. Chaque classe proposée devra contenir les constructeurs adéquats. La méthode `toString()` devra permettre d'afficher de manière naturelle l'expression donnée. Vous devrez également redéfinir la méthode `equals(Object o)` de manière à ce que deux expressions ayant la même évaluation soient égales. Il faut également définir la méthode `eval()` qui calcule la valeur de l'expression.

Vous devrez définir (au moins) les classes indiquées à la suite.

— Une classe `ExpressionsArithmetiques` déclare les fonctionnalités communes à toutes les expressions considérées.
— Une classe `Constant` représente les expressions constantes.
— Les classes `Addition` et `Multiplication` (on omet volontairement les classes `Soustraction` et `Division`, relativement identiques aux deux précédentes) représentent les opérations arithmétiques. **Attention, ces classes ont des concepts communs ...**

Soit le diagramme de classe donnée Figure 2.

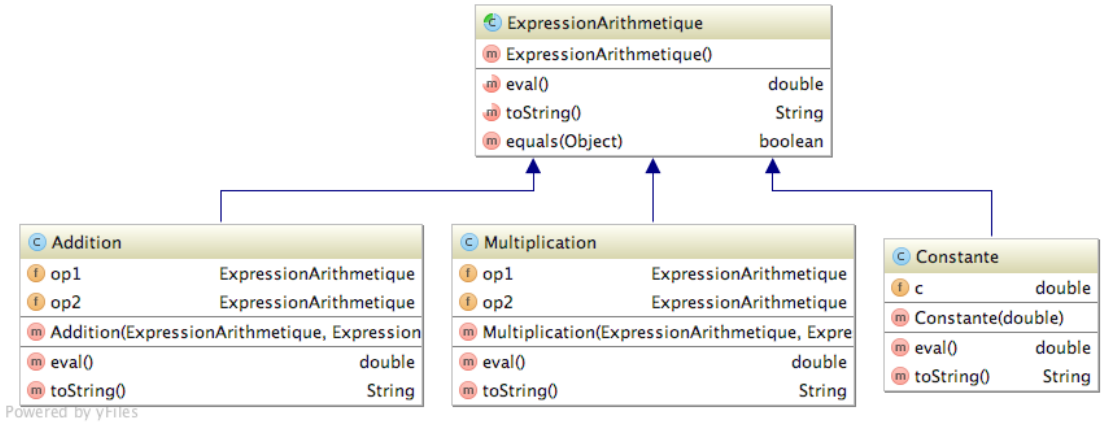

FIGURE 2 – Diagramme de classe

1. Analyse du diagramme

a. Repérez la classe abstraite. Contient-elle une méthode abstraite ?

b. Quelle est la relation entre la classe `Constante` et la classe `ExpressionArithmetique` ?

c. Pourquoi la méthode `eval()` est-elle abstraite ?

d. Pourquoi la méthode `equals(Object o)` n'est-elle pas redéfinie dans les sous-classes ?

e. Y-a-t-il des éléments à factoriser entre les classes `Addition` et `Multiplication` ? Si oui, lesquels ?

f. Quelle classe allez-vous ajouter pour assurer cette modification ?

g. Quels sont ses liens avec les classes existantes ?

2. Dans votre IDE, entrez directement le diagramme de classe avec les modifications proposées dans les questions précédentes.

3. Coder les différentes classes. Observez qu'elles sont déjà créées, vous devez seulement ajouter le code manquant.

4. Que faut-il ajouter pour que les expressions arithmétiques puissent contenir des variables comme dans l'expression 3 * x + 7 ?

5. A-t-on besoin de modifier le code des classes `Addition` etc (justifiez) ?

6. Écrire la classe `Variable`.

7. Écrire une fonction `String prefixe()` qui retourne l'expression arithmétique sous forme préfixée $+3 * +4.7\ 2.3\ 5$. Où faut il déclarer cette fonction ?

8. Visualisez le nouveau diagramme de classe complet.