Manipulation d'expressions formelles

21, 28 janvier

On souhaite manipuler de façon symbolique toute expression mathématique telle que

$$2x^2 + 3 \tag{1}$$

$$2x^2 + \frac{3y - 5xy + 1}{1 + \frac{3}{x}} \tag{2}$$

$$\sin(2\pi(x+1))e^{\frac{1}{x^2}} - \ln(x+5) \tag{3}$$

$$\sqrt{a + \frac{1+y}{1+x}} \tag{4}$$

Ici, on ne fera pas la distinction entre une variable et un paramètre. Toute expression littérale, comme a, x, y, est une variable symbolique. Une expression peut être représentée par un arbre dont chaque noeud représente une opération $(+; -; \times; /)$, une constante, une fonction ou une variable symbolique.

On distinguera trois types de constantes : les constantes entières (contenant un nombre entier), rationnelles (contenant deux nombres entiers, le numérateur et le dénominateur) et symboliques (comme π, e) permettant de manier des nombres irrationnels de façon exacte : permettant de donner de façon exacte $\sin(\pi) = 1, \ln(e) = 1$.

On souhaite représenter par une classe distincte chaque noeud possible permettant de composer des expressions. Ainsi l'expression $1 + 2$ sera représentée par un objet instanciant la classe `Addition` et ayant parmis ses champs des références vers une instance de la classe `ConstanteEntiere` dont un champ contiendra l'entier 1 et une instance de la classe `ConstanteEntiere` dont un champs contiendra l'entier 2.

Exercice 1. Expressions arithmétiques

On se limite ici aux expressions purement arithmétiques, c'est à dire, sans variable symbolique, ni fonction, et avec uniquement des constantes entières ou rationnelles. Comme par exemple

$$(9+2) \times \left(3 - \frac{1}{5+12}\right). \tag{5}$$

a. Concevoir un ensemble de classes permettant de représenter les expressions arithmétiques. Indication : utilisez une classe abstraite `ExpressionArithmétique` et exploiter les vertues de l'héritage et du polymorphisme.

b. Écrire une méthode `affiche` dans chacune de ces classes qui permette l'affichage de l'expression correspondant à l'arbre ayant pour racine le noeud courant. S'assurer que le parenthésage soit correct (mieux vaut trop que pas assez).

c. Écrire une méthode `simplifie` dans chacune de ces classes de sorte que, appelée à la racine de l'arbre, cette méthode retourne l'arbre formé d'un unique noeud de type `ConstanteEntière` ou `ConstanteRationnelle` contenant la valeur de l'expression arithmétique (sans approximation).

Exercice 2. Expressions symboliques

On souhaite maintenant représenter toute expression symbolique, comme celles listées en introduction. On utilisera pour cela une nouvelle classe abstraite `ExpressionSymbolique`.

a. Proposer une hiérarchie d'héritage entre les classes existante, cette nouvelle classe et les futures classes nécessaires aux expressions symboliques : `Variable, RacineCarree,...`

b. Implémenter des classes pour les constantes symboliques (comme π, e), les variables symboliques, l'opérateur $\sqrt{}$ et les fonctions sin, cos, ln et la fonction puissance $(x, n) \longrightarrow x^n$. Les constantes symboliques et les variables symboliques auront un champ de type `String` pour leur identifiant (ex. `Pi, x, yprime, ...`)

c. Dans chacune de ces classes, fournir également les méthodes `affiche` et `simplifie`. La sortie de `simplifie` n'est plus nécessairement une constante entière ou rationnelle. Par exemple $(2+1)x+3\times 2$ se simplifie en $3x+6$. Mettre à jour si nécessaire les méthodes `simplifie` dans les classes définie pour les expressions arithmétiques pour garantir ce bon fonctionnement.

d. Implanter aussi les identités classiques : $\ln(1)=0, \ln(e)=1, \sin(\pi)=\sin(0)=\cos(\pi/2)=0, \cos(\pi)=\sin(-\pi/2)=-1, \cos(0)=\sin(\pi/2)=1$.

e. On souhaite maintenant calculer la dérivée d'une expression par rapport à une variable symbolique. Modifier vos classes en leur ajoutant une méthode `derive(VariableSymbolique var)` prenant un seul argument `var`, représentant la variable selon laquelle faire la dérivation et retournant l'expression dérivée correspondante.

f. Vérifier votre code sur plusieurs instances tests.

Exercice 3. Pour aller plus loin : substitution et évaluation

a. Implémenter une méthode `substitue(VariableSymbolique var, ExpressionSymbolique exp)`, qui modifie l'expression courante en remplaçant toutes les occurrence de la variable symbolique `var` par l'expression `exp`.

b. Implémenter des méthodes `evalue` qui substitue aux constantes rationnelles une constante flottante (on définira une telle classe `ConstanteFlottante`).

c. Proposer des implantations de cette méthode `evalue` pour chaque classe, y compris les fonctions, en s'appuyant soit sur la bibliothèque Math de Java, soit mieux, en déduisant de développement limités de ces fonctions, un schéma d'approximation.

Ainsi toute expression symbolique dont toutes les variables ont été substituées par des expressions sans variable peuvent être évaluées.

d. Proposer plusieurs exemples d'utilisation démontrant la fonctionnalité de votre programme.