TD8 : Héritage et Liaison dynamique

POO : Programmation Orientée Objet

Première année

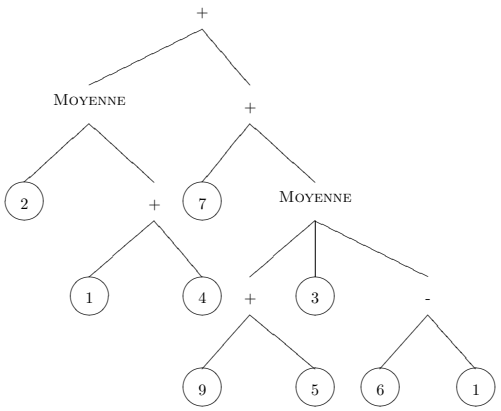
L'objectif du TD et du TP associé est de définir un ensemble de classes permettant de représenter et de manipuler des expressions arithmétiques.

1 Le problème

Les expressions sont constituées :
- d'opérandes constantes ou réelles,
- d'opérateurs binaires : $+$, $-$, $*$ et $/$,
- d'opérateurs n-aires : MAXIMUM, MOYENNE, SOMME, ...

Une expression peut être représentée sous la forme d'un arbre abstrait dont :
- la racine est l'opérateur le moins prioritaire,
- les nœuds (internes) sont des opérateurs,
- les feuilles sont les opérandes.

Par exemple, l'arbre abstrait de la figure ci-contre représente l'expression :
MOYENNE$(2, 1+4) + (7 +$ MOYENNE$(9+5, 3, 6-1))$



2 Première approche (sans héritage)

Dans un premier temps, nous allons concevoir une solution sans utiliser l'héritage.
Nous souhaitons pouvoir réaliser deux traitements sur une expression :
- `evaluer : expression` $\rightarrow$ `réel` qui permet d'évaluer la valeur d'une expression.
  Par exemple, l'évaluation de l'expression représentée par l'arbre abstrait présenté précédemment doit calculer la valeur `17.833...`
- `decompiler : expression` $\rightarrow$ `chaîne` qui transforme un arbre abstrait en une chaîne de caractères représentant l'expression.
  Ce traitement correspond au traitement inverse qu'il faut réaliser lors de la compilation (où il s'agit de transformer une chaîne de caractères en un arbre abstrait).

▷ **Question 1.** Pour simplifier, on considérera uniquement le cas d'expressions constituées d'opérateurs binaires. Proposer une solution sous la forme d'une seule classe nommée `Expr`.

▷ **Question 2.** Écrire une classe `TestExpr` permettant d'évaluer l'expression $(9 + 5)/(5 - 3)$ et de "décompiler" l'arbre abstrait que vous avez construit.

3 Seconde approche (héritage et liaison dynamique)

Bien que correcte, l'implémentation proposée dans la section précédente évolue très mal (ajout de nouveaux opérateurs, ajout de nouveaux traitement, ...). Nous souhaitons maintenant utiliser l'héritage et notamment le mécanisme de liaison dynamique afin de proposer une solution relativement aisée à maintenir et bien plus évolutive.

En utilisant l'héritage, il est possible de discriminer les opérations suivant les différentes expressions élémentaires composant une expression arithmétique. De plus, il est également possible de raffiner la classification selon l'arité (le nombre d'opérandes) des opérateurs : constantes, opérateurs binaires, opérateurs n-aires.

▷ **Question 3.** Proposer une hiérarchie de classes permettant de modéliser le problème. Identifier les classes qui seront abstraites.

▷ **Question 4.** Dessiner le schéma mémoire de l'objet qui correspond à l'expression :

MOYENNE$(2, 1+4) + (7 +$ MOYENNE$(9+5, 3, 6-1))$

▷ **Question 5.** Pour réaliser l'évaluation d'une expression, que doit-on ajouter comme primitives (attributs et méthodes) et dans quelles classes ? Étudier le comportement des classes `Exp`, `Binaire`, `NAire`, `Const`, `Plus` et `Moyenne`.

(a) Dessiner le graphe d'héritage avec les attributs et les méthodes (uniquement leur profil).

(b) Écrire le corps des différentes méthodes

▷ **Question 6.** On souhaite maintenant étendre nos expressions en autorisant la définition et l'utilisation de variables. Une même expression pourra ainsi être réduite avec différentes valeurs pour ses variables. Par exemple, on définit une variable $x = 3$ et on souhaite évaluer l'expression $max(x + 2, 7, 9)$. Il faut donc être capable de lier le nom de la variable x avec la valeur 3. Puis, il faut pouvoir remplacer x par sa valeur dans l'expression. Par exemple, si l'on a la définition $x = 6$ et l'expression $max(x + y, y, 9)$, on peut réduire cette expression en une nouvelle expression $max(6 + y, y, 9)$.

(a) Définir une nouvelle classe `Variable` qui doit permettre de manipuler des expressions particulières que sont les variables.

(b) Définir une nouvelle classe `Contexte` permettant de définir le contexte d'évaluation d'une expression en liant une valeur à un nom de variable.

(c) Pour chaque classe, définir une méthode `Exp reduire(Contexte c)` capable de réduire une expression pour un contexte donné.

Concernant l'expression `Maximum`, on souhaite pouvoir réduire l'expression même partiellement. Par exemple, si la définition $x = 3$ est présente dans le contexte, la réduction de l'expression $max(x + y, y, 7, 9)$ doit donner une nouvelle expression $max(3 + y, t, 9)$ (on a évalué la valeur de x et on a déjà calculé le maximum entre 7 et 9).

4 Préparation et sujet du TP

▷ **Question 7.** Afin de préparer le TP de cette semaine,

(a) Dessiner l'arbre d'héritage des classes

(b) Récapituler, pour chaque classe, les attributs et les méthodes.

▷ **Question 8.** Programmer les différentes classes

Remarque : Pour réaliser le dictionnaire de la classe `Contexte`, vous pouvez vous référerer à la documentation [1] de la classe `java.util.HashMap` (constructeurs, méthodes `get()` et `put()`).

▷ **Question 9.** Implémenter le processus de "décompilation" en ajoutant les méthodes nécessaires dans les différentes classes. Il s'agit donc de produire une chaîne de caractères dans un buffer de type `java.lang.StringBuffer` (renseignez-vous sur le fonctionnement de la méthode `append()`).

1. `http://download.oracle.com/javase/6/docs/api/`