Arbres de syntaxe abstraite d'expressions arithmétiques

Nous généralisons aux arbres la structure de liste chaînée Java, ainsi que ses manipulations, toujours en récursif : construction, affichage, parcours (pas de modification d'arbre aujourd'hui). L'expérience acquise sur les traitements récursifs des listes se généralise rapidement aux arbres, comme nous pourrons le voir tout au long des exercices. Appelez-nous pour des précisions sur ce sujet.

Par ailleurs, nous possédons maintenant l'ensemble du matériel Java à acquérir en TC (à un ou deux détails près).

Le sujet choisi pour illustrer ces concepts aujourd'hui est celui indiqué dans le titre. On se restreint aux opérateurs $+$ et $\times$ avec parenthèses.

Comme la dernière fois, les énoncés sont brefs, sauf difficulté majeure.

1. Récupérer et exécuter le fichier *CodeTd7.java* ci-dessous (ne pas le retaper, voir adresses ci-dessous, nous appeler pour des précisions). Que fait-il ?

   Notons que *x.equals("toto")* compare la variable chaîne *x* et la chaîne constante *toto*. Notons aussi que dans le cas d'un opérateur, le champ *val* n'est jamais utilisé (on a mis -1 par convention).

2. Construire un arbre constant plus riche (4 ou 5 occurences de $+$ et $\times$).

3. Afficher un arbre en notation infixe (ex : $(3+4)\times 5$).

4. Evaluer l'expression arithmétique représentée par un arbre.

5. Construire l'arbre de syntaxe abstraite d'une expression arithmétique à partir de sa notation préfixée (ex : $\times +\ 3\ 4\ 5$).

Un peu de recul : Vous venez, mine de rien, d'écrire un interpréteur pour un minuscule langage : un programme y est simplement une expression arithmétique constante. Ce n'est qu'un début, mais bravo !

Facultatif

6. Même question, mais à partir de sa notation infixe... On considére les mêmes expressions (non vides), mais dans lesquelles l'application d'un opérateur est toujours entourée de parenthèses (ex : 3, $(3+4)$, $((3+4)\times 5)$).

Codes et corrigé : `http://www.lri.fr/~waller/COURS/POLYTECHNIQUE/TC/TD7` ou fichier à copier `~waller/CodeTd7.java`

Excellente année à tous, y compris en informatique

```
import java.io.*;

class Asaear {
    String op;
    int    val;
    Asaear sabg;
    Asaear sabd;

    Asaear (String o, int v, Asaear g, Asaear d) {
        op = o;
        val = v;
        sabg = g;
        sabd = d;
    }
}

public class CodeTd7 {

    public static String Lire () {
        ... celle de d'habitude ...
    }

    static Asaear construit() {
        return(new Asaear("x", -1,
                          new Asaear("+", -1,
                                     new Asaear("feuille", 3, null, null),
                                     new Asaear("feuille", 4, null, null)),
                          new Asaear("feuille", 5, null, null)));
    }

    static int f (Asaear b) {
        if (b == null)
            return(0);
        else
            if (b.op.equals("feuille"))
                return(0);
            else
                return(1 + f(b.sabg) + f(b.sabd));
    }

    public static void main (String[] args) {
        Asaear a;
        a = construit();
        System.out.println(f(a));
    }
}
```