Cours de Programmation Orientée-Objet

Énoncé du Travail

Année 2011–2012

Le programme Java réalisé doit parvenir par courrier électronique avant le 27 avril à midi, à l'adresse `degbomont@montefiore.ulg.ac.be`. *Les travaux rendus en retard et les travaux plagiés ne seront pas corrigés. Les détails non précisés dans l'énoncé sont libres d'interprétation. Les travaux ne compilant ou ne fonctionnant pas correctement dans l'environnement des laboratoires de l'institut Montefiore ne seront également pas corrigés.*

On demande d'implémenter en Java un ensemble de classes permettant de représenter et de manipuler des formules de l'arithmétique des entiers.

On définit une formule arithmétique par une notation préfixe :
- les constantes entières, comme par exemple -5, 0 ou 12, sont des formules,
- une variable (désignée par un identifiant) est une formule,
- si `f` et `g` sont des formules, alors :
  - la formule `add f g` représente la somme `f + g`,
  - la formule `sub f g` représente la différence `f - g`,
  - la formule `mul f g` représente le produit `f . g`,
  - la formule `div f g` représente le quotient (entier) `f / g`.

Une addition, une soustraction, une multiplication et une division entière possèdent toujours deux opérandes.

On peut naturellement représenter une formule par un arbre dont les nœuds sont des opérateurs et dont les feuilles sont des variables ou des constantes. Par exemple, la formule `add mul p q -7` est représentée par l'arbre de la figure 1.

On veut pouvoir attribuer des valeurs entières à un sous-ensemble des variables contenues dans une formule, ce qui peut mener à une formule plus simple. Par exemple, pour la formule `add p q`, si on attribue à `p` la valeur 5 et à `q` la valeur 2, on obtiendra la formule `add 5 2`, qui se simplifie en 7.

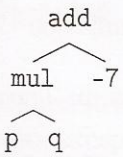

FIGURE 1 – Arbre représentant la formule `add mul p q -7`

Les règles de simplification de formules sont les suivantes :

- Une addition, une soustraction, une multiplication et une division entière ayant pour opérandes deux constantes se simplifie en la constante valant, respectivement, la somme, la différence, le produit et le quotient entier de ces deux constantes.
- Une addition dont une des opérandes est la formule `f` et la seconde la constante `0` se simplifie en la formule `f`.
- Une multiplication dont une des opérandes est la formule `f` et la seconde la constante `1` se simplifie en la formule `f`.
- Une division dont la première opérande est la formule `f` et la seconde la constante `1` se simplifie en la formule `f`.
- Une soustraction dont la première opérande est la formule `f` et la seconde la constante `0` se simplifie en la formule `f`.
- Une multiplication dont une des opérandes est la formule `f` et la seconde la constante `0` se simplifie en la constante `0`.
- Une division entière dont la première opérande est la constante `0` et la seconde une formule `f` (ne valant pas `0`) se simplifie en la constante `0`.

Une formule est complètement simplifiée si l'application des règles de simplification n'a plus d'effet sur celle-ci.

Représentation textuelle des formules

Afin de pouvoir interfacer la représentation de formules avec d'autres programmes, nous définissons une représentation textuelle des formules. Une formule est représentée par une chaîne de caractères (`java.lang.String`) qui contient des mots (suites de caractères alphanumériques) séparés par un ou plusieurs espaces (définis par la méthode `isWhiteSpace()` de la classe `java.lang.Character`). Ces mots forment la notation préfixe de la formule.

L'addition est représentée par `"add"`, la soustraction par `"sub"`, la multiplication par `"mul"` et la division entière par `"div"`. Les constantes sont représentées par une suite de caractères parmi { `'0'`, `'1'`, ... , `'9'` } avec éventuellement le symbole `'-'` en première position pour indiquer une valeur négative. Tous les autres mots qu'il est possible d'écrire représentent des variables.

Une représentation textuelle possible de la formule `add mul p q 7` serait par exemple la chaîne de caractères `"add mul p q 7"`.

Classes et interfaces

L'interface de la classe `Formula` est la suivante :

`Formula(String s)` : Constructeur permettant d'instancier une formule décrite par la chaîne de caractères `s`.

`eval(String variable, int value)` : Modifie la formule en remplaçant chaque occurrence de la variable `variable` par la constante `value`.

`add(Formula g)` : Invoquée sur la formule `f`, la méthode transforme cette dernière en la formule `add f g`.

`sub(Formula g)` : Invoquée sur la formule `f`, la méthode transforme cette dernière en la formule `sub f g`.

`mul(Formula g)` : Invoquée sur la formule `f`, la méthode transforme cette dernière en la formule `mul f g`.

`div(Formula g)` : Invoquée sur la formule `f`, la méthode transforme cette dernière en la formule `div f g`.

La formule doit toujours être complètement simplifiée après chaque opération.

Par ailleurs, la classe `Formula` devra implémenter l'interface `Cloneable`, et réimplémenter la méthode `toString()` qui devra fournir une chaîne de caractères contenant la formule en notation préfixe, où les mots sont séparés par un et un seul espace.

Gestion des erreurs et efficacité

Toutes les méthodes et constructeurs susceptibles de rencontrer une erreur doivent le signaler en déclenchant des exceptions appropriées. Retourner des valeurs particulières pour gérer les cas d'erreurs n'est pas acceptable.

Le code écrit doit être raisonnablement efficace.

Scénario de test

Le scénario suivant sera (entre autres) testé sur votre programme :

```
Formula f = new Formula("add add mul p q 5 0");
System.out.println(f.toString());
Formula g = new Formula("div 7 q");
f.sub(g);
System.out.println(f.toString());
f.eval("q", 2);
System.out.println(f.toString());
f.eval("p", 4);
System.out.println(f.toString());
```

Résultat attendu (sur la sortie standard) :

```
add mul p q 5
sub add mul p q 5 div 7 q
sub add mul p 2 5 3
10
```