L2 info S3
UE : Programmation Orientée Objets 1 - Java

TD3 - complément

Interfaces : pile par tableau, expression en polonaise inversée

On désire écrire un programme Java permettant d'évaluer des expressions écrites en polonaise inversée. Cette notation permet de noter des formules arithmétiques sans utiliser de parenthèses.

Exemples :

écriture usuelle	écriture en polonaise inversée
$(2+3)\times 4$	$2\ 3\ +\ 4\ \times$
$1+(2+3)$	$1\ 2\ 3\ +\ +$
$2-3$	$2\ 3\ -$

Pour effectuer le calcul, chaque donnée numérique est empilée, et lorsqu'un opérateur est lu, les opérandes sont dépilés (deux opérandes si l'opérateur est binaire), l'opération est effectuée et le résultat est mis au sommet de la pile.
Ainsi, par exemple, pour évaluer $2\ 3\ +\ 4\ \times$:

1) lecture de 2 qui est reconnu comme donnée numérique et de ce fait est empilé,
2) lecture de 3 qui est également empilé,
3) lecture de + qui est un opérateur binaire : 2 et 3 sont dépilés, le calcul $2+3$ est effectué, le résultat 5 est empilé,
4) lecture de 4 qui est empilé,
5) lecture de $\times$, opérateur binaire : 4 et 5 sont dépilés, le calcul 4×5 est effectué et le résultat 20 est empilé,
6) la chaîne est entièrement lue $\rightarrow$ le résultat final est au sommet de la pile.

Dans votre programme Java, la pile sera à base d'un tableau et ses éléments de type `Integer` (ou `int`).

Ce programme illustrera le notion d'interface. Il comporte une interface `Pile` ainsi que les classes `Tableau`, `PileParTableau`, `Expression` et `Test`.

Interface Pile

Elle comporte les méthodes publiques :

- `boolean pileVide()` qui renvoie *true* si la pile est vide, *false* sinon.
- `boolean pilePleine()` qui renvoie *true* si la pile est pleine, *false* sinon.
- `Integer depiler()` qui renvoie l'objet au sommet de la pile (à condition que la pile ne soit pas vide).
- `void empiler(Integer obj)` qui copie l'objet au sommet de la pile (à condition que la pile ne soit pas déjà pleine).

Classe Tableau

C'est une classe qui peut être abstraite et qui comporte :
- `Integer tab[]` : la pile.
- `int indice` qui mémorise une position dans le tableau (pour `PileParTableau` ce sera l'indice du premier élément non utilisé).
- `Tableau(int taille)` le constructeur qui instancie un nouveau tableau avec la taille transmise en argument.

La taille du tableau n'est pas mémorisée étant donné qu'elle peut être calculée (`tab.length`).

Classe PileParTableau

La classe `PileParTableau` est une sous-classe concrète de `Tableau` qui implémente l'interface `Pile`. Elle comporte :
- les méthodes de l'interface,
- un constructeur `PileParTableau(int taille)` ; l'argument est la taille de la pile (du tableau) à allouer.

Dans cet énoncé, on ne demande aucune gestion particulière des erreurs. renvoyer `null` si la pile à dépiler est vide et ne rien faire s'il faut empiler un nouvel élément dans une pile pleine.

Classe Expression

Cette classe comporte :
- un attribut `line` de type `String` qui comporte une expression en polonaise inversée.
- un constructeur `Expression(String val)` qui permet d'initialiser l'attribut `line`.
- une méthode `Integer evalue()` qui évalue l'expression. Il faut
  1. extraire de `line` les différentes sous-chaînes correspondant aux opérandes et opérateurs. Pour ce faire, utiliser la méthode `split` qui est une méthode d'instance de la classe `String` définie comme suit : `public String[] split(String regex)` Vous y ferez appel avec `regex` qui est la chaîne avec un caractère "blanc" (" ") : le séparateur dans votre expression entre les opérandes et opérateurs est un "blanc".
  2. instancier un objet de type `PileParTableau` pour effectuer votre calcul.
  3. lire les opérandes et opérateurs et effectuer les opérations correspondantes.

Vous pouvez compléter votre classe avec des méthodes privées telles que :
1. `private boolean operateur(String)` qui renvoie *true* si la chaîne transmise en argument correspond à un opérateur "reconnu " et *false* sinon ;
2. `private void calcul(PileParTableau, char)` qui a comme arguments la pile avec laquelle effectuer le calcul et l'opérateur. Dans cette méthode, les opérandes sont dépilés, l'opération effectuée et le résultat est stocké au sommet de la pile.

Classe Test

Elle comporte la méthode `main` qui permet
1. d'instancier un objet de la classe `Expression`,
2. de calculer la valeur d'une expression écrite en polonaise inversée,
3. d'afficher le résultat final.

Une expression est référencée par une variable de type `String`. Elle peut au choix :
- être directement insérée dans le code source,
- être mise sur la ligne de commande,
- être lue interactivement en cours d'exécution.

Un unique espace sépare chaque composante de l'expression.