**Laboratoire 4**

Objectifs : Concevoir une application qui met en pratique les notions vues à ce jour.

Directives : Le travail compte pour 15% de la note finale, réalisé seul

Date de remise : Précisée dans Léa

1. **Description du travail à réaliser**

L'homme écrit généralement des expressions telles que **3 + 4** et **7 / 9** où l'opérateur (**+** ou **/**) s'écrit entre les opérandes; c'est ce qu'on appelle la notation *infixée*. Pour évaluer une expression à l'aide de l'ordinateur, il est plus facile d'utiliser la notation *suffixée*, où l'opérateur apparaît à droite des deux opérandes. Les expressions infixées précédentes apparaîtraient en notation suffixée respectivement sous la forme **3 4 +** et **7 9 /.**

L'évaluation d'une expression infixée complexe implique que le compilateur convertisse d'abord l'expression en sa notation suffixée, puis évalue la version suffixée de l'expression. Vous aurez à coder deux classes Java mettant en œuvre les algorithmes reliés à chacune de ces étapes. Chaque algorithme utilise un objet de type pile.

Dans ce travail, vous allez écrire un programme Java qui,

1. Accepte en entrée une expression en notation infixée
2. Affiche cette même expression en notation suffixée
3. Évalue le résultat de cette expression
4. Affiche le résultat de cette évaluation

Nous ne traiterons que les expressions arithmétiques :

1. Valides
2. Constituées de constantes, d'opérateurs et de parenthèses
3. Dont les opérateurs arithmétiques autorisés sont : + et – (binaires), \* et /
4. Et dont les constantes sont des entiers à un seul chiffre
5. **Structure du programme**

On vous demande de diviser vos classes en trois paquetages, comme suit : **demarrage**, **expressionsInfixes** et **pileGenerique**.

|  |  |
| --- | --- |
| Paquetage | Classe(s) contenue(es) dans ce paquetage |
| *demarrage* | *TestProgramme pour tester le programme* |
| *expressionsInfixes* | *ConvertisseurInfixeEnSuffixe* et *EvaluateurSuffixe* |
| *pileGenerique* | Classe générique *Pile* présentée dans la partie théorique sur les piles. |

1. **Création de la classe ConvertisseurInfixeEnSuffixe**

Cette classe permet de convertir une expression arithmétique infixée telle que :

**(6+2)\*5-8/4**

en une expression suffixée, en tenant compte de la priorité des opérateurs. La version suffixée de l'expression infixée précédente est (remarquez qu'aucune parenthèse n'est nécessaire) :

**62+5\*84/-**

* 1. L'algorithme de création d'une expression suffixée est :

1. Pousser une parenthèse gauche ' ( ' sur la pile
2. Ajouter une parenthèse droite ' ) ' à la fin de l'expression infixée
3. Tant que la pile n'est pas vide, traiter les caractères de l'expression infixée de gauche à droite et effectuer ce qui suit :

Si le caractère courant est un chiffre, l'ajouter à l'expression suffixée

Si le caractère courant est une parenthèse gauche, l'empiler

Si le caractère courant est un opérateur :

Retirer les opérateurs, s'il y en a, du sommet de la pile tant qu'ils ont une priorité égale ou supérieure à celle de l'opérateur courant et ajouter les opérateurs retirés à l'expression suffixée

Empiler le caractère courant

Si le caractère courant est une parenthèse droite :

Tant que le sommet de la pile n'est pas une parenthèse gauche, retirer les opérateurs du sommet de la pile et les ajouter à l'expression suffixée

Retirer (et supprimer) la parenthèse gauche de la pile.

* 1. Les méthodes de la classe ConvertisseurInfixeEnSuffixe sont :

1. La méthode **convertirEnSuffixe**, qui convertit une expression infixée en notation suffixée (et retroune la version suffixée).
2. La méthode **estUnOperateur**, qui détermine si un caractère est un opérateur.
3. La méthode **prioriteEgaleOuGrande**, qui détermine si la priorité d'un premier opérateur est égale ou plus grande à la priorité d'un deuxième opérateur. La méthode renvoie **true** si le premier opérateur est de priorité égale ou plus grande que la priorité du deuxième opérateur. Sinon, elle renvoie **false**.
   1. Méthode voir de la classe PileGenerique

Assurez-vous que la méthode **voir**, qui retourne l'élément au sommet de la pile (sans le dépiler), est présente dans la classe PileGenerique.

1. **Création de la classe EvaluateurSuffixe**

Cette classe parcourt et évalue une expression suffixée telle que :

**6 2 + 5 \* 8 4 / -**

* 1. L'algorithme permettant d'évaluer une expression suffixée est :

Pour chaque caractère de l'expression suffixée :

Si le caractère courant est un chiffre,

Empiler sa valeur entière sur la pile;

Sinon, si le caractère courant est un opérateur,

Retirer les deux éléments du sommet de la pile et les déposer dans les variables **operande2** et **operande1**;

Calculer **operande1** opérateur **operande2**;

Empiler le résultat du calcul sur la pile;

* 1. Les méthodes de la classe EvaluateurSuffixe sont :

1. La méthode **evaluerSuffixe** qui retourne la valeur d'une expression suffixée
2. La méthode **resultat**, qui retourne le résultat de l'évaluation de **opérande1** **opérateur** **opérande2**.
3. **Directives**

* 1. Les classes que vous devez coder ne doivent pas avoir de propriétés.
  2. Utiliser comme fichier de départ le fichier **tp2DepartV2.zip** disponible dans les documents distribués de Léa dans la catégorie **Tps**.
  3. Remettre votre projet sous la forme d'un fichier .**zip**.   
     Essayer d'importer ("Import") votre projet à partir du fichier .zip.  
     Si vous pouvez exécuter votre programme alors remettre votre fichier .zip.
  4. Le travail doit être réalisé individuellement.