# Rapport pour le TP numéro 5

CHASTEL Paul

Mai 2021

## 1 Introduction

L'objectif de ce Tp de POO était de nous faire créer une structure **Java** qui nous permattait de rentrer une expression préfixée du type: (+(* 2 x)( / 3 y)) qui est déduite de l'expression: ( ( 2*x ) + ( 3/y ) ) ; pour nous faire ressortir en premier lieu l'expression non-préfixée et après avoir rentré les valeurs des variables, le programme nous resort le résultat de l'expression.

Exemple tiré du sujet:
p - (+ (* 2 x) (/ y 3))
(2 * x) + (y / 3)
p(x=3, y=6)
8

## 2 Mise en place des classes "simples":

Pour réaliser cette structure, j'ai créé une classe par opération binéaire (Addition, Soustraction, Division, Multiplication), une classe pour les constante(Constante) et enfin une classe pour les variables(Variable). Ces classes seront plus tard sous-classes d'une super classe appelée Opération . Composée d'un constructeur pour les variables et constante et d'un super-constructeur sur Opération pour les opérations binéaires. A ces classes je suis venu y implémenter la méthode *toString()* comme demandé dans la question 2) qui viendra afficher l'opération ou la variable/constante en écriture standardisée.

Toute ces classes seront publiques car ré-utilisées.

## 3 Interface Expression:

Pour la question 3) qui nous demandais de créer la méthode *eval()* j'ai décidé de créer une interface qui me permettait d'implementer cette méthode.

Cette méthode permet d'affecter une valeur pour chaque variables, elle a été implementée dans la super-classe Operation puis reprise dans chaque sous-classe.

# 4 Mise en place de la super-classe Operation:

Cette super-classe est composée de deux constructeurs, le premier simple qui prends deux expressions en entrée et celui plus complexe qui prends une chaine de caractères en entrée qui est en fait la partie fonctionnelle de mon programme celle qui va venir traiter l'expression préfixée en enlevant d'abord les paranthèses et en la lisant en partant de la droite ce qui me permet de traiter d'abord les opérations de poids faibles puis de poids de plus en plus fort.

Les variables *tmp1* et *tmp2* font office de mémoire pour les valeurs que l'on va trouver que ce soit des variables ou des constantes d'ailleurs, vérifiant au passage de ne pas récupérer d'expressions mal formatées, une fois ces deux mémoires remplies, je viens chercher le prochain opérateur binénaire (+,-,*,/), je créer ensuite l'opération et je viens la stocker dans une liste d'operation: *memList[]*, une fois que je croise un opérateur je fais fusionner les deux dernières opérations de *memList()* qui représente les deux dernières opérations de poids faible rencontrées et ce jusqu'à ce qu'il me reste deux opération dans *memList()* qui correspondront aux deux fils d'opération que je viendrais afficher et/ou traiter dans mon *Main()*.

# 5 Création du Main:

Le *Main()* vient saisir les expressions rentrées dans la console mais ces expressions doivents être formatées du type: *p 'signe infèrieur du clavier Latex ne le prends pas'- (+ (* 2 x) (/ y 3))* et l'affectation doit être formatée au type: *p(x=3, y=6)*. Il va prends les deux fils d'opèration et l'afficher séparé du signe du premier opérateur binéaire trouvé dans l'expression préfixée.