Petit Coup de main pour les Exercices de la série 1 à 6

Par Joel Sandé

- Ceci est pour vous donner un coup de main (Algorithme et un bref, très bref apercu sur mon code) pour les conversions Infixe et Postfixe).

- Ce n'est pas LA solution. C'est ma façon de le faire; certains d'entre vous les ont déjà terminés avec des logiques similaires, mais d'autre n'ont aucune idée comment procéder.

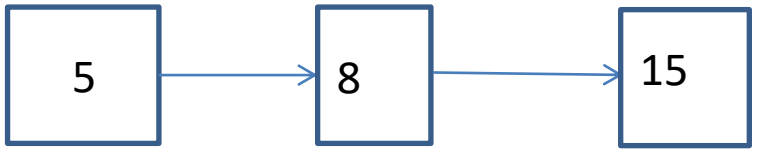
- Ceci me permet de Valider que vous savez au moins par où commencer.

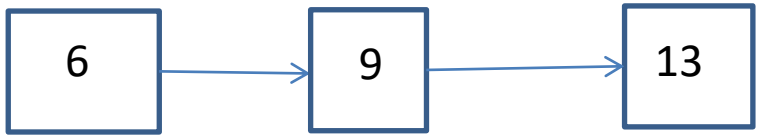
- Je ne vous demande pas de recopier la même chose. Comprenez juste le principe, et soyez créatifs.

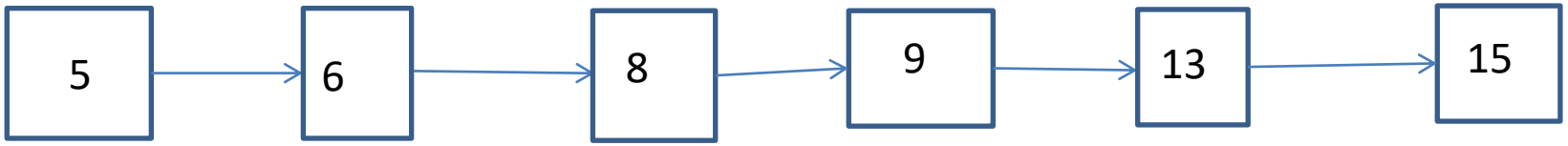
- Si donc vous avez déjà terminé, FERMEZ TOUT DE SUITE CE POWER-POINT.

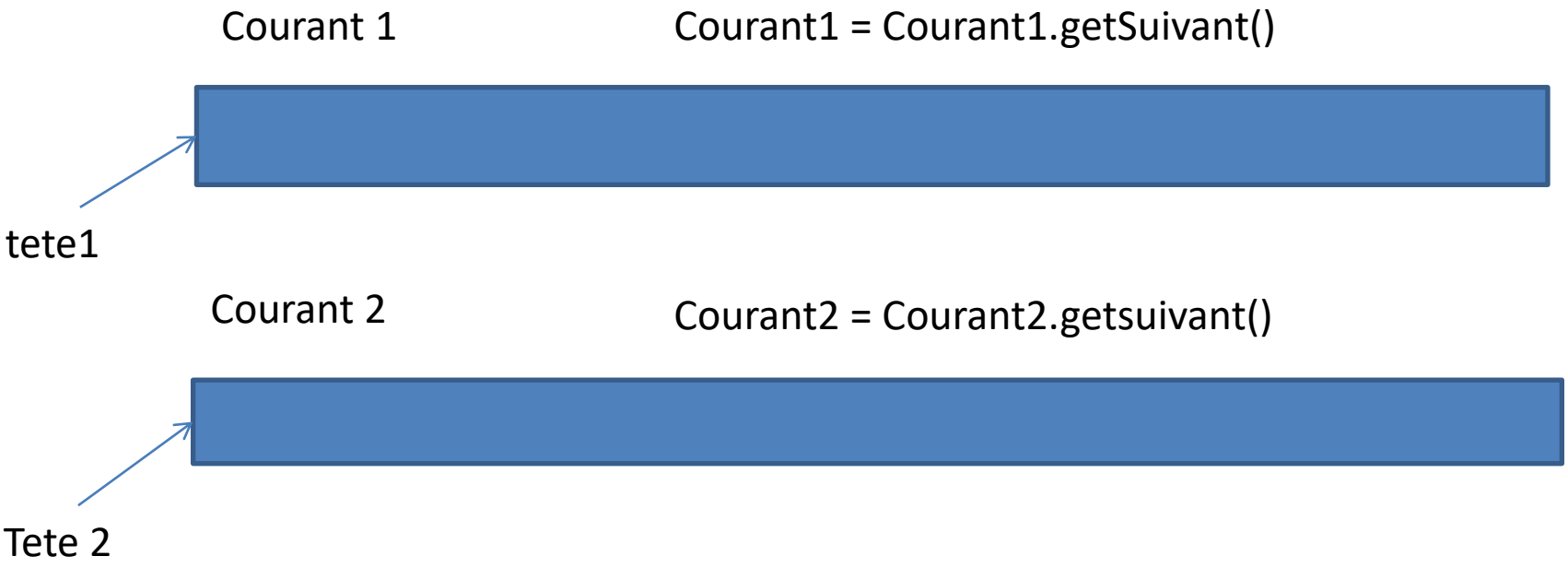
Exercice 1

- Un programme qui fusionne 2 Listes chainees ordonnées:

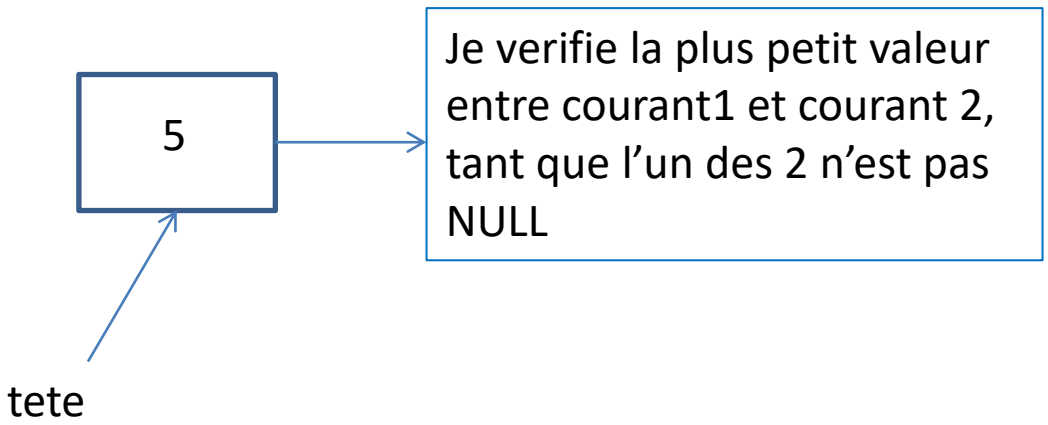








Tete1 = 5, tete2 = 6. La tete de ma Liste sera la plus petite des 2 tetes.



Remarque : Si vous ne procédez pas de cette façon, et que vous fusionnez les 2 listes directement, vous aurez par la suite à trier la Liste du bas. Ce que pas mal de personnes ont fait.

Exercice 2

- String s = “abracadabra”;

- Pour aller chercher un character à une position donnée, vous pouvez par exemple utiliser la méthode CharAt(position)

- s.ChartAt(4) = c

Exercice 3

- Ce exercice est un peu ambigue, surtout à la dernière phrase.

- Toutefois un palindrome c'est un mot qui contient une symétrie:

mum , assa, abcdefggfedcba

- Suivez le même principe que l'exercice 2 avec la particularité de la symétrie.

Exercice 4

Là, nous arrivons aux choses plus compliquées.

L'algorithme de conversion, vous les avez dejà dans vos diapos. Vous avez le choix d'utiliser l'algorithme que vous aviez avant mon arrivée dans ce cours, où celui que je vous ai donné.

Pour ce qui suit, je me base sur celui que je vous ai donné, et les règles qui viennent avec (Slides 3, 5 et 6 de Récapitulatif Infixe PostFixe).

Pas mal d'entre vous l'ont fait avec l'algorithme précédent : c'est la même chose.

Symbole	Stack	PostFix
(	(	
A		A
+	(+	
B		AB
/	(+/	
C		ABC
*	(+* / (*et/ meme priorité)	ABC/
(	(+*(	
D		ABC/D
+	(+*(+	
E		ABC/DE
)	(+* (+) (on vire + à droite)	ABC/DE+
-	(+* (- ne peut pas etre placé devant *)	ABC/DE+*
	(+- (first-in-first-out)	ABC/DE+*+
F	(-	ABC/DE+*+F
)	(-)	

- Remarquez que j'ai 3 colonnes : je les identifie à 3 piles.

- La 1ère pile existe déjà : c'est mon expression Infixe
- La 2e Pile est à remplir sur la base de l'Algorithme
- La 3e Pile est à remplir sur la base de l'Algorithme

```
public class Convertion {

    public static void main(String[] args) {

        Pile_2 p = new Pile_2();

        System.out.println("Ajout d'un element a la Pile");
        p.AjouterElt(new Element(" ( "));
        System.out.println("la Pile devient : " + p.ToString() + "\n");


        System.out.println("Ajout d'un element a la Pile");
        p.AjouterElt(new Element(1));
        System.out.println("la Pile devient : " + p.ToString() + "\n");


        System.out.println("Ajout d'un element a la Pile");
        p.AjouterElt(new Element(" + "));
        System.out.println("la Pile devient : " + p.ToString() + "\n");


        System.out.println("Ajout d'un element a la Pile");
        p.AjouterElt(new Element(2));
        System.out.println("la Pile devient : " + p.ToString() + "\n");
```

```
run:
Ajout d'un element a la Pile
la Pile devient :  (

Ajout d'un element a la Pile
la Pile devient :  ( 1

Ajout d'un element a la Pile
la Pile devient :  ( 1 +

Ajout d'un element a la Pile
la Pile devient :  ( 1 + 2

Ajout d'un element a la Pile
la Pile devient :  ( 1 + 2 )

Ajout d'un element a la Pile
la Pile devient :  ( 1 + 2 )  *

Ajout d'un element a la Pile
la Pile devient :  ( 1 + 2 )  * 3
```

```
        Pile_2 c;
        c = p.Convertir_Infixe_Postfixe();
        System.out.println("la Pile p3 est : " + c.ToString() + "\n");

    }
```

À présent je suis dans ma class Pile. Tout le précedent était dans le main

```
public Pile_2 Convertir_Infixe_Postfixe(){
    Pile_2 stack = new Pile_2();
    Pile_2 postFixe = new Pile_2();
    Element courant = this.tete;
    Element last_stack;

    while(courant != null){
        System.out.println("courant = " + courant.getElement() + "\n");

        if(EstOperateur(courant)){
            System.out.println("Il detecte un Operateur ");
            System.out.println("courant = " + courant.getElement() + "\n");
            last_stack = stack.getLast();
            if (memePrioritee(courant, last_stack)){
                stack.RemoveLast();
                postFixe.AjouterElt(last_stack);
                stack.AjouterElt(courant);
            }else if(interdit(last_stack, courant)){
                stack.RemoveLast();
                postFixe.AjouterElt(last_stack);
            }else{
                stack.AjouterElt(courant);
            }
        }else if(EstAccolade(courant)){
```

Voici les 2 Piles de milieu et droite définies il y a 2 slides en arriere

L' Algorithme entier est basé sur 2 choses :

- L'élément que j'ajoute à la stack
- Le dernier élément de la stack

Definissons à présent 2 fonctions

```
public boolean memePrioritee(Element a, Element b){
    boolean meme_priorite = false;

    if((a.equals(new Element(" / "))) && (b.equals(new Element(" * ")) || a.equals(new Element(" * ")) || b.equals(new Element(" / ")))) {
        meme_priorite = true;
    }else if( (a.equals(new Element(" + ")) && b.equals(new Element(" - "))) || (a.equals(new Element(" - ")) && b.equals(new Element(" + ")))) {
        meme_priorite = true;
    }
    return meme_priorite;
}
```

Pour Vérifier si un élément est un Operande, je me sers d'une Expression régulière utilisée dans la Méthode estOperande()

if(elt.ToString().matches("[0-9]+")){….}

La structure de cette ligne dépend de la façon dont vous retournez le ToString(). Il faut que ce soit un String pour que ca marche. Vous pouvez le faire autrement.

- Voilà, maintenant que vous avez l’idée, je pense que pour le reste, vous pourrez vous débrouiller comme des grands.

- Je procède ainsi pour vous laisser le plus possible votre liberté de “création”. Je veux que ça vienne de vous.

Exercice 5

- Pareil que l'exercice 4, à la seule différence qu'il y a un petit calcul à faire avec les opérandes tirés du postFixe.

Exercice 6

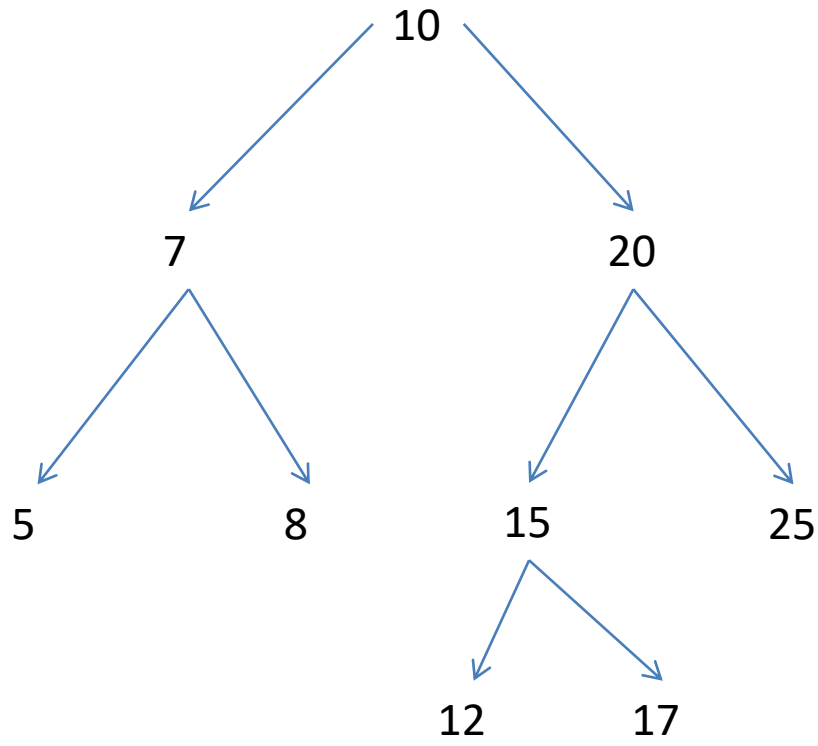
- J’avoue que c’est le plus Plat, mais heureusement, c’est le dernier de la série.

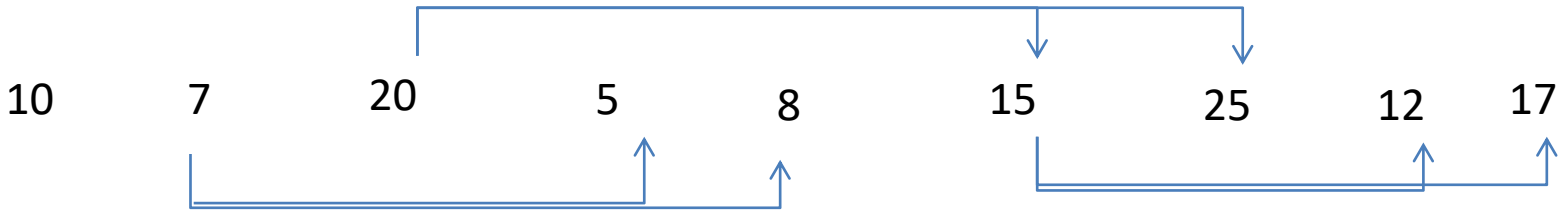
- Mais tiens, j’y pense: vous avez tous rendu le devoir sur les arbres binaires !! Vous avez donc la moitiée de la solution.

Voici une façon simple de comprendre ce qu'on vous demande de faire.

```
Insérer la racine dans la file
// Afficher la valeur de la racine
Tant qu'il y a des noeuds dans l'ARBRE
        // Obtenir le prochain noeud de la
        file
        // Afficher la valeur du noeud
        Si le noeud a un enfant gauche
                Insérer l'enfant dans la file
                Afficher la valeur du Noeud
        Si le noeud a un enfant droit
                Insérer l'enfant dans la file

                Afficher la valeur du Noeud
```





A moins que quelqu'un l'ait fait ou compri autrement.

J’espère que ça vous aide