Arbres et récursivité

Les sujets de salle plate sont différents des sujets de salle machine. Les questions d'écriture de code en salle plate pourront être faite dans un langage de haut niveau.

1 Exemple initial

On considère les fonctions `test1` et `test2` suivantes :

```
void test1(int n){
  System.out.println(n);
  if(n > 0){
    test1(n - 1);
    }
  }
```

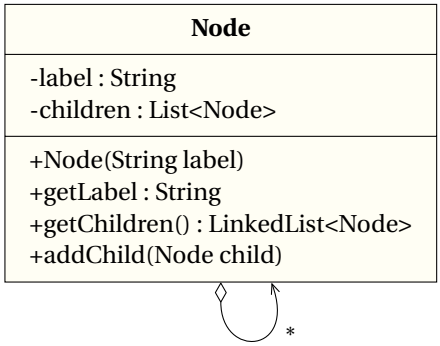
```
void test2(int n){
  if(n > 0){
    test2(n - 1);
    }
  System.out.println(n);
}
```

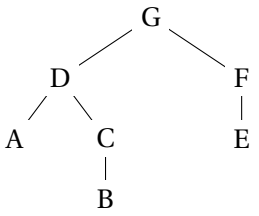
1. Qu'affichent les fonctions `test1` et `test2` ? (Essayez de deviner avant de "copier-coller" !)
2. Que feront ces fonctions si on remplace les appels récursifs par des appels croisés dans chacune des deux fonctions ? (c'est à dire si `test1` appelle `test2` au lieu de `test1` et vice versa.)

2 Parcours d'arbre

2.1 Approche classique

Considérons le diagramme de classes et l'arbre ci-dessous.





3. Écrivez les appels qui permettent de créer l'arbre dessiné avec la classe fournie.
4. Qu'affiche le code ci-dessous ?

```
 void affiche(Node tree){
    System.out.println(tree.getLabel());
    for(Node child : tree.getChildren()){
        affiche(child);
    }
}
```

5. Modifiez le code ci-dessus pour qu'il affiche les noeuds dans l'ordre postfixe, correspondant à l'ordre alphabétique sur l'exemple fourni.
6. Écrivez une fonction qui compte le nombre de noeuds de l'arbre.
7. Écrivez une fonction qui prend en paramètre une étiquette et indique si l'arbre contient un noeud avec cette étiquette.
8. Écrivez une fonction qui reçoit un arbre et qui retourne la hauteur [1] de l'arbre.
9. Proposez une fonction qui prend une profondeur k en paramètre et retourne le nombre de noeuds de l'arbre à la profondeur k.

On souhaite écrire une fonction qui permet de déterminer l'étiquette d'une feuille la plus profonde de l'arbre. Il existe deux solutions. La première (plus fonctionnelle) consiste à créer un objet spécifique (étiquette + profondeur) et retourner de tels objets lors des appels.

10. Écrivez une fonction qui retourne l'étiquette d'une feuille la plus profonde avec cette méthode.

La deuxième méthode, moins élégante mais plus simple à mettre en place, consiste à définir deux variables globales, profondeur et étiquette. On parcourt tout l'arbre et chaque fois qu'on trouve une feuille, on met à jour si nécessaire la valeur de ces deux attributs. Puis, à l'issue du parcours, on récupère ces deux valeurs. Cela implique d'avoir une deuxième fonction non récursive qui fait l'appel à la fonction récursive.

12. Écrivez cette méthode.
13. Écrivez une fonction qui affiche l'étiquette de la feuille la moins profonde. On n'utilisera aucune des deux méthodes précédentes, mais un parcours en largeur.

3 Palindrome

Un palindrome est un mot ou une phrase dont la séquence de lettre est la même si on la lit de droite à gauche ou de gauche à droite. Par exemple, *radar*, *elle*, *malayalam* [2] sont des palindromes.

On peut aussi définir un palindrome comme un mot dont la première et la dernière lettre sont identiques, et tel que le sous-mot obtenu par le retrait de ces lettres est un palindrome. Ceci donne l'idée d'une fonction récursive pour vérifier qu'un mot est un palindrome.

14. Quelle condition d'arrêt aura un programme récursif qui vérifie qu'un mot est un palindrome ?
15. Écrivez une fonction récursive qui vérifie qu'un mot est un palindrome. [3]

1. Rappelons que la profondeur d'un noeud dans un arbre est le nombre minimum de pas à faire depuis la racine pour atteindre ce noeud (0 pour la racine). La hauteur d'un arbre est la profondeur maximale des noeuds de l'arbre plus 1.
2. Une langue parlée au Kérala, dans le sud de l'Inde
3. Vous pourrez utiliser les méthodes des objets `String` `charAt(int i)` qui retourne le caractère en position i et `subString(int d, int f)` qui retourne la chaîne dont les caractères sont compris entre les position d (inclu) et f (exclu).