TD 7
Arbres

1 Rappel de Cours

Dans le cours sur les arbres, nous avons vu des arbres particuliers que sont les 'Arbres Binaires de Recherche'.

Définition 1. Un **arbre binaire** est un arbre dont les noeuds ont au maximum deux fils : un sous-arbre gauche, et un sous-arbre droit.

Ainsi, un noeud pour un arbre binaire est modélisé comme suit en JAVA :

```java
//Classe interne représentant un Noeud d'un arbre binaire
private class Noeud {
     //Attributs de la classe Noeud
    int element ;
    Noeud filsGauche ;
    Noeud filsDroit ;

    //Constructeurs d'un Noeud
    public Noeud(int element) {
        this.element = element ;
        filsGauche = null ; filsDroit = null ;
    }
    public Noeud(int element, Noeud filsGauche, Noeud filsDroit) {
        this.element = element ;
        this.filsGauche = filsGauche ;
        this.filsDroit = filsDroit ;
    }
}
```

Définition 2. Un **arbre binaire de recherche** est arbre binaire dans lequel un tri particulier des valeurs est fait, c'est à dire que pour un noeud ayant une valeur N, les noeuds de son sous-arbre gauche ont tous des valeurs $< N$, et les noeuds de son sous-arbre droit ont tous des valeurs $\geq N$.

2 Exercices

Exercice 1. Faites vos gammes

On part de la classe `ArbreBinaire` vue en cours, dont une partie est rappelée ci-dessous :

```java
public class ArbreBinaire {

    //Classe interne représentant un Noeud pour ce type d'arbre, i.e. un arbre binaire
    private class Noeud {
         //Attributs de la classe Noeud
        int element ;
        Noeud filsGauche ;
        Noeud filsDroit ;

        //Constructeurs d'un Noeud
        public Noeud(int element) {
            this.element = element ;
            filsGauche = null ; filsDroit = null ;
        }
        public Noeud(int element, Noeud filsGauche, Noeud filsDroit) {
            this.element = element ;
            this.filsGauche = filsGauche ;
            this.filsDroit = filsDroit ;
        }
    }

    //Attribut de la classe Arbre Binaire
    private Noeud racine ;
    ......
}
```

Question 1. Méthodes sur la structure de l'arbre Un peu de récursivité : pour cette classe `ArbreBinaire`, écrivez les méthodes suivantes :

- `int size()` : renvoie le nombre total de noeuds dans l'arbre.
- `int leaves()` : renvoie le nombre de noeuds feuille dans l'arbre.
- `int height()` : renvoie la « hauteur » de l'arbre, c'est-à-dire le nombre de noeud maximal qu'il y a sur un chemin de la racine à une feuille dans l'arbre. Par convention un arbre vide a une hauteur de 0, et un arbre à 1 noeud a une hauteur de 1.
- `boolean balanced()` : renvoie `true` si l'arbre est équilibré (*balanced* en anglais), et `false` sinon. Un arbre est équilibré **si** ses sous arbres gauche et droit sont équilibrés **et si** la différence de hauteur des deux sous arbres est d'au plus 1.

Question 2. Méthodes sur le contenu de l'arbre Les méthodes écrites précédemment permettent d'avoir des informations sur la structure, nous souhaitons maintenant avoir des informations sur son contenu.
Écrivez les méthodes suivantes :

- `int valeurMax()` : renvoie la valeur maximale contenue dans un noeud de l'arbre,
- `double valeurMoyenne()` : renvoie la valeur moyenne de valeurs contenues dans un noeud de l'arbre.

Exercice 2. Une classe `Ensemble`

Nous allons écrire une classe `Ensemble` ayant les fonctions d'un ensemble mathématique. Cette classe pourra stocker un ensemble de `String` (sans doublon) et vérifier si un élément est présent dans l'ensemble ou pas.

Vous utiliserez un **arbre binaire de recherche** comme structure de donnée interne. Cet exercice vous permettra donc à la fois d'implémenter un arbre binaire (ce qui n'a pas été vu en CM pour l'instant), et de de voir à quoi il sert dans un cas concret.

Question 1. Écrire la méthode d'ajout d'un élément dont la signature est : `void ajouterElement(String e)`. Cette méthode ajoute un élément à l'ensemble mais ne fait rien si l'élément est déjà dans l'ensemble.
Au niveau de l'arbre binaire, cela veut dire que vous devez essayer d'ajouter une valeur `e` dans l'arbre, mais en suivant strictement les règles de la définition 2. Vous devez donc parcourir l'ABR en descendant à gauche si la valeur est strictement plus petite que le noeud courant, et à droite si elle est strictement plus grande (rien si valeur déjà dans l'arbre). La nouvelle valeur est ajoutée comme feuille quand il n'est plus possible de descendre.

Question 2. Écrire la méthode `boolean estPresent(String s)` qui indique si la chaîne s est dans l'ensemble ou pas.
Au niveau de l'arbre binaire, cela signifie que vous devez parcourir l'arbre en cherchant la valeur `s`, toujours en suivant les règles de la définition 2.

Question 3. Écrire la méthode `String toString()` qui renvoie une représentation en chaîne de l'ensemble, de la forme "aaa, bbb, ccc". Vous êtes libres dans le choix de l'ordre de parcours des éléments (préfixe/infixe/postfixe).