IUT de Montpellier **TD3 : Arbres**
M3103 Algorithmique avancée

Nous rappelons que les méthodes sont maintenant à écrire dans la classe correspondante.

Sauf mention contraire, il est interdit d'utiliser des méthodes des exercices précédents pour résoudre l'exercice courant. Les boucles sont interdites.

Conseil pour la conception de votre algorithme :

- la méthode du cours indique qu'il faut d'abord penser à une "grande entrée" : ici, un "gros" arbre sera un arbre dont les deux sous arbres sont non vides
- ensuite, vérifiez que votre code précédent est toujours correct lorsque
  - l'arbre est vide (ce qui n'est en général pas le cas, et entraîne donc l'ajout d'un cas de base)
  - l'arbre est non vide, mais l'un (ou les deux) sous arbres sont vides (ce qui peut entraîner soit des ajouts de cas de base, soit d'autres branches qui feront des appels récursifs différemment)

1 Exercices sur les arbres

On rappelle qu'une feuille est un arbre non vide dont les deux sous arbres sont vides (une feuille est donc un entier dans l'arbre n'ayant aucun entier en dessous), et qu'un noeud interne est un arbre non vide n'étant pas une feuille (un noeud interne est donc un entier dans l'arbre ayant au moins un entier en dessous).

Exercice 1. Nombre de noeuds

Question 1.1.
Ecrire une méthode `int nbNoeuds()` qui renvoie le nombre d'entiers de l'arbre courant.

Exercice 2. Nombre de feuilles

Question 2.1.
Ecrire une méthode `int nbFeuilles()` qui renvoie le nombre de feuilles de l'arbre courant.

Exercice 3. Recherche

Question 3.1.
Ecrire une méthode `boolean recherche(int x)` qui renvoie vrai ssi x est contenu dans l'arbre.

Exercice 4. Recherche père-fils égaux

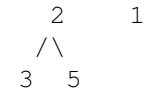
Question 4.1.
Ecrire une méthode `boolean pereFilsEgaux()` qui renvoie vrai ssi il existe une valeur x apparaissant dans un noeud de l'arbre et dans un de ses deux noeuds fils (c'est à dire si il existe un sous arbre A de this vérifiant (A.val == A.filsG.val) ou (A.val == A.filsD.val)).

Exercice 5. Symétrie

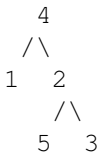
Question 5.1.
Ecrire une méthode `Arbre symetrise()` qui crée un nouvel arbre indépendant en opérant une symétrie verticale passant par la racine.

Par exemple, l'arbre

```
    4
  /   \
```



devient

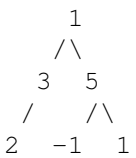


Exercice 6. Meilleur Chemin

Question 6.1.

Ecrire une méthode `int meilleurChemin()` qui renvoie le poids minimum d'un chemin partant de la racine vers une feuille, où le poids d'un chemin est défini comme la somme des entiers rencontrés. On dira que le meilleurChemin d'un arbre vide est $+\infty$

Par exemple, dans l'arbre ci dessous meilleurChemin() doit retourner 5 (et par exemple ne peut pas retourner 4 en utilisant le parcours 1 puis 3, puisque 3 n'est pas une feuille).



Exercices bonus

Exercice 7. Recherche dans les feuilles

Question 7.1.

Ecrire une méthode `boolean chercheFeuille(int x)` qui renvoie vrai ssi x est une feuille de l'arbre.

Exercice 8. Recherche dans les noeuds internes

Question 8.1.

Ecrire une méthode `boolean chercheNoeudInterne(int x)` qui renvoie vrai ssi x est dans un noeud interne de l'arbre.