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.

Une démarche possible pour résoudre les exercices :

— d'abord écrire et récupérer les réponses des deux appels récursifs sur les deux sous arbres

— puis, imaginer un cas général (this est un "gros arbre") pour trouver comment assembler ces réponses

— vérifiez que votre idée fonctionne dans les quatres cas suivants : les deux sous arbres sont vides, le fils gauche seulement est vide, le fils droit seulement est vide, les deux fils sont vides

1 Exercices

On rappelle qu'une feuille est un arbre 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 dans les feuilles

Question 3.1.
Ecrire une méthode `boolean chercheFeuille(int x)` qui renvoie vrai ssi x est une feuille de l'arbre.

Exercice 4. Recherche dans les noeuds internes

Question 4.1.
Ecrire une méthode `boolean chercheNoeudInterne(int x)` qui renvoie vrai ssi x est dans un noeud interne de l'arbre.

Exercice 5. Recherche père-fils égaux

Question 5.1.
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 6. Symétrie

Question 6.1.
Ecrire une méthode `Arbre symetrise()` qui crée un nouvel arbre indépendant en opérant une symétrie verticale passant par la racine. Par exemple, l'arbre ((() 2 (() 3 ())) 1 (() 4 ())) devient ((() 4 ()) 1 ((() 3 ()) 2 ())) (Rappel, la notation précédente est la notation obtenue en faisant un parcours infixe de l'arbre, cf cours.)

Exercice 7. Chemin

Question 7.1.
Ecrire une méthode `boolean parcours(Liste l)` qui renvoie vrai ssi il existe un chemin partant de la racine à une feuille et qui rencontrant (dans l'ordre) les entiers de l. Par exemple, avec a = ((() 5 ()) 3 ((() 8 ()) 7 ())), a.parcours(3 : :7 : :8 : :())=true, a.parcours(7 : :8 : :())=false, a.parcours(5 : :3 : :7 : :())=false, a.parcours(3 : :5 : :())=true.

Exercice 8. Meilleure traversée

Question 8.1.
Ecrire une méthode `int traverse()` 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. Par exemple, avec a = ((() 2 ()) 3 ()) 1 ((() -1 ()) 5 ())), a.traverse()=5.