TP Java : Généricité

Classe Noeud

Cette classe représentera un noeud d'un arbre binaire (*pas nécessairement d'un arbre binaire de recherche*). Un noeud a une valeur et peut avoir 2 noeuds fils : le noeud gauche et le noeud droit.
La classe `Noeud` doit être **générique**. On devra pouvoir indiquer le type `E` des valeurs qui seront conservées dans le noeud. Les éléments de `E` ne sont pas nécessairement ordonnés. Mettez dans la classe

- un constructeur sans paramètre,
- un constructeur qui prend une valeur en paramètre,
- un constructeur qui prend une valeur et les 2 noeuds gauche et droit en paramètres,
- les accesseurs et les modificateurs pour la valeur et les 2 noeuds gauche et droit,
- une méthode `toString()` qui affiche les valeurs du noeud et les sous-noeuds attachés à ce noeud.

Vous écrirez une classe `TestNoeud` pour tester votre classe. Testez seulement la création d'un noeud qui contient un noeud fils ; utilisez la méthode `toString()` pour vérifier que tout marche bien.

Classe ArbreBinaireRecherche

Cette classe sera générique. On pourra indiquer le type `E` des éléments contenus dans l'arbre.
Un arbre binaire de recherche a un nœud racine. Pour tout nœud de l'arbre

- le sous-nœud gauche est la racine d'un arbre qui contient les éléments inférieurs ou égaux à la valeur du nœud ;
- le sous-nœud droit est la racine d'un arbre qui contient les éléments supérieurs à la valeur du nœud.

Un arbre binaire de recherche doit nécessairement travailler avec une relation d'ordre sur les éléments de type E (cela n'est pas nécessaire pour le nœud d'un arbre binaire quelconque).
Vous allez commencer par écrire un arbre dont les éléments ont un ordre naturel, comme c'est le cas, par exemple, pour `Integer` ou `String`. Cet ordre naturel sera représenté par l'interface **générique** `java.lang.Comparable`.
C'est à vous de compléter la définition de `Comparable` pour trouver la bonne instanciation.
Cette classe contiendra 2 constructeurs de signatures :
`ArbreBinaireRecherche()` (construit un arbre vide)
et `ArbreBinaireRecherche(E)` (indique la valeur de la racine de l'arbre).
Elle contiendra aussi

- une méthode pour ajouter un élément dans l'arbre ;
- une méthode qui indique si un objet appartient à l'arbre ;

- une méthode pour afficher tous les éléments de l'arbre dans leur ordre naturel.

Quelle visibilité allez-vous donner à la classe `Noeud` (vous pouvez modifier cette classe si vous le souhaitez) ?

Test de ArbreBinaireRecherche avec une instanciation

Écrivez une classe `TestArbreBinaire` qui crée un arbre binaire qui contient des `Integer` et un autre qui contient des `String`. Pour tester avec des entiers, vous pouvez générer un grand nombre d'entiers de façon aléatoire en utilisant la classe `java.util.Random` et sa méthode `nextInt()` ou `nextInt(int)`.

Testez aussi avec un arbre binaire qui contient des employés. Utilisez pour cela les 2 classes `Personne` et `Employe` (classe fille de `Personne`). Les employés seront rangés dans l'arbre suivant l'ordre alphabétique de leur nom.