TD Généricité

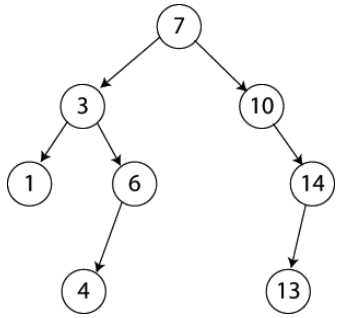
Dans ce TD vous allez écrire une classe pour représenter un arbre binaire de recherche dans lequel les éléments seront rangés selon leur ordre naturel ou selon un ordre déterminé.

Toutes les classes de ce TD seront dans un paquetage abr.

Rappel sur les arbres binaires de recherche

Un **arbre binaire de recherche** (ABR) est un arbre binaire où chaque élément possède une valeur telle que chaque élément du sous-arbre gauche aie une valeur inférieure à celle de l'élément considéré, et que chaque élément du sous-arbre droit possède une valeur supérieure à celle-ci.

On choisira dans notre implémentation d'interdire les doublons dans l'arbre.



Question 1

Ecrire la classe ArbreBinaireRecherche. Cette classe sera générique : on pourra indiquer le type E des éléments contenus dans l'arbre. Ces éléments ont un ordre naturel (comme les *Integer* ou les *String*) i.e. implémentent l'interface *Comparable*.

Cette classe contiendra 2 constructeurs :
ArbreBinaireRecherche (*élément racine)*
ArbreBinaireRecherche (*élément racine, arbregauche g, arbredroit d)*

Ecrire les méthodes d'accès et de modification d'un arbre binaire de recherche.
(valeur, affecterValeur, filsGauche, filsDroit, affecterFilsGauche, affecterFilsDroit, existeFilGauche, existeFileDroit, estFeuille).

Question 2

Ecrire une méthode *contient (*qui renvoie un booléen) et qui permet de déterminer si un élément *x* est contenu dans un arbre binaire de recherche.

Question 3

Ecrire une méthode *insérer (élément e)* qui permet d'ajouter un élément dans l'arbre.
Cette méthode renverra une exception si l'élément à ajouter est déjà présent dans l'arbre

L'insertion d'un nœud commence par une recherche: on cherche l'élément à insérer, et lorsque l'on arrive à une feuille, on ajoute l'élément comme fils de la feuille en comparant sa valeur à celle de la feuille: si elle est inférieure, le nouvel élément sera à gauche, sinon il sera à droite.

Question 4

Ecrire 3 méthodes d'affichage des éléments de l'arbre (parcours préfixe, infixe, postfixe).

Question 5

Ecrire une classe *TestArbreBinaire* qui crée un arbre binaire qui contient des *Integer* et un autre qui contient des *String* et qui utilise les méthodes écrites aux questions précédentes.

Question 6

Ecrire la méthode listeElem renvoyant la liste des éléments de l'arbre, dans l'ordre infixe.
Ecrire une nouvelle méthode permettant d'afficher les éléments de l'arbre dans l'ordre infixe qui utilise une boucle "for-each" sur la liste définie ci-dessus.

Question 7

Ecrire une classe Personne (nom, prénom, date de naissance).
Ecrire une classe Etudiant, qui en plus d'être une personne a un attribut niveau pouvant prendre l'une des valeurs : NUL, MAUVAIS, MOYEN, BON, EXCELLENT.
Ecrire une classe Enseignant, qui en plus d'être une personne a un attribut statut pouvant l'une des valeurs : CHEF, ARPETTE.
Faites le nécessaire pour que l'ordre naturel de comparaison des personnes soient l'ordre alphabétique sur le nom.

Question 8

Un arbre peut contenir des personnes, des étudiants et des enseignants. L'ordre d'insertion dans l'arbre utilisant l'ordre naturel, les éléments sont triés selon l'ordre alphabétique sur le nom (cf. question 7). Il arrive pourtant que l'on souhaite trier des personnes par age ou des étudiants selon leur niveau. Dans ce cas, on utilisera la méthode listeElem de l'Arbre, puis on procédera au tri sur la liste. Faites le nécessaire, pour pouvoir réaliser ces tris.