TP1 - Algorithmique avancée : Arbres Binaires de Recherche

CERI - Master I Informatique - 2016-2017

1 Modélisation objet

On souhaite implémenter une **classe générique** d'Arbre Binaire de Recherche (ABR). Chaque noeud d'un ABR est lui-même un ABR. De plus, chaque noeud contient une **donnée** (exemple : un mot, une image, ...). Les règles suivantes doivent être respectées :

- toutes les données d'un ABR sont du **même type** (exemple : chaque donnée de l'arbre correspond à un mot, chaque donnée de l'arbre correspond à une image, ...) ;
- toute donnée doit comporter un attribut de type chaîne de caractère nommé "**cle**" (exemple : le mot sans majuscule ni accent, le nom de l'image, ...) ;
- les données d'un ABR sont **comparables** entre elles grâce à ce champ ;
- toute donnée doit comporter une méthode "**toString**" retournant une chaîne de caractère représentant la donnée. Attention, cette méthode ne se contente pas nécessairement de retourner l'attribut clé (exemple : elle peut retourner le texte d'origine du mot alors que la clé contient le mot formatée sans majuscule ni accent).

Exercice 1 : Proposer une modélisation objet basée sur une **classe abstraite**, et utilisant une **interface** permettant d'implémenter cette classe générique.

Votre modélisation devra notamment inclure une classe ABR comportant :

- un attribut **donnee** générique ;
- un attribut ABR **pere** ;
- un attribut ABR **filsGauche** ;
- un attribut ABR **filsDroit** ;
- un constructeur prenant en argument une donnee.

2 Implémentation méthodes

Dans la suite de ce TP, vous testerez chacune des méthodes implémentées en utilisant le fichier de "ABRTest.java" que vous étofferez (ajouter au minimum un test/assert par méthode).

Afin de pouvoir utiliser le fichier ABRTest.java, vous devrez remplacer les mentions TYPE1, TYPE2, TYPE3 et TYPE4 par des types issus de votre modélisation.

Exercice 2 : Implémenter votre classe ABR et ajoutez-lui les méthodes :

- **insertionFeuille(d)** : insère un nouveau noeud contenant une donnée d en feuille de l'arbre ;
- **toString()** : renvoie une chaîne de caractères qui contient l'ensemble des données de l'arbre dans l'ordre lexicographique des clés et séparés par une virgule ;
- **recherche(String cle)** : renvoie la référence de l'objet de clé "cle" s'il existe, ou "null" sinon ;
- **minimum()** : renvoie la référence du noeud contenant l'objet de clé minimale ;
- **maximum()** : renvoie la référence du noeud contenant l'objet de clé maximale
- **hauteur()** : renvoie la hauteur de l'arbre
- **supprimer(cle)** : supprime l'objet de clé "cle" tout en maintenant la structure d'un ABR.

3 Implémentation méthodes avancées

Exercice 3 : Arbres équivalents

Définition Deux arbres binaires sont **équivalents** s'ils contiennent les mêmes éléments.

Écrire la méthode **equivalent(abr)** qui renvoie *true* si l'arbre est équivalent à *abr* et *false* sinon.

Vous mentionnerez en commentaire de cette méthode les complexités dans le pire et dans le meilleur des cas de l'algorithme que vous proposez.

Exercice 4 : Arbre contenu dans un autre

Définition Un ABR A est **contenu** dans un ABR B si et seulement si tous les éléments de A sont dans B.

Écrire la méthode *contenuDans(abr)* renvoyant *true* si l'arbre est contenu dans *abr*.

Vous mentionnerez en commentaire de cette méthode les complexités dans le pire et dans le meilleur des cas de l'algorithme que vous proposez.

Exercice 5 : Fusion d'arbres

Définition L'ABR C est dit **fusion** des arbres A et B si et seulement si U contient tous les noeuds de A et tous ceux de B.

Écrire une procédure fusionnant deux arbres, en évitant de recréer un troisième arbre par recopies des données de A et B.

4 Applications

On souhaite appliquer la classe précédemment codée au stockage des mots d'un dictionnaire de la langue. On stockera ces mots sous la forme d'un ABR pour faciliter la recherche ultérieure. Le dictionnaire est donné par le fichier "francais.mots" (source : fichier formaté mar M. Van Caneghem, Université Aix-Marseille) se trouvant sur le site e-uapv.

Proposer une classe **Dico** permettant de lire le fichier et de le stocker dans un champ ABR de la classe.

Créer une classe de test **DicoTest** permettant de tester les différentes fonctionnalités de votre classe ABR sur ce dictionnaire.