Algorithmique et structures de données

TD / TP 10

Les objectifs de cette séance de TD / TP sont :

- Concevoir puis utiliser les Arbres Binaires de recherche
- Utiliser la programmation récursive
- Exploration d'un arbre binaire

TD

1.Conception d'un arbre binaire de recherche.

On reprend l'arbre binaire du dernier TD. (cf feuille)

- Quelle est la différence entre un arbre binaire et un arbre binaire de recherche ?

Nous avons vu en cours comment insérer un nouvel élément dans un ABR.
La méthode **insereTo()** se charge de placer correctement la valeur selon la relation d'ordre de l'arbre.

2.Méthode insereTo

- Ré-écrire InsereTo sans l'aide du cours. (rappel : absence de doublon)

3.Méthode de recherche

- Ecrivez la méthode rechercherABR en respectant la relation d'ordre.
  public boolean rechercherABR (Integer element)

4.Méthode Supprimer.

- Quelle stratégie allez-vous utiliser pour supprimer un élément ?
- Ecrire la méthode Supprimer (Integer element).

  Rmq : intValeur() retourne l'entier contenu dans un objet de type Integer.
  V1.compareTo(V2) compare deux Integer avec leurs valeurs numériques. Elle retourne un entier >0 si V1>V2 ; =0 si V1=V2 ; <0 si V1<V2

5.Construction d'un arbre équilibré.

- Faites la méthode récursive arbreBREnTab qui permet de placer les éléments d'un ArbreBR dans un ArrayList() en ordre croissant.
  public void arbreBREnTab (ArrayList t)

- Faites un nouveau constructeur ArbreBRCons qui va construire un ArbreBR équilibré à partir d'un tableau trié dans l'ordre croissant.

  public ArbreBRCons(ArrayList t, int debut, int fin)
  On applique la stratégie du QS ou de la dichotomie.

  Ex sur des entiers :

2	5	9	14	15	19
debut		milieu			fin

  On construit un ArbreBR en prenant l'élément de l'indice milieu.
  A sa gauche il y aura un ArbreBR du sous tableau inférieur à milieu.
  A sa droite il y aura un ArbreBR du sous tableau supérieur à milieu.

Deux cas particuliers
- debut=fin
- debut= indice et fin = indice+1

TP

- Concevez et testez la classe ArbreBR

- Testez l'équilibrage d'un arbre avec les méthodes

  public void arbreBREnTab (ArrayList t)
  public ArbreBRCons(ArrayList t, int debut, int fin)

Un Arbre Binaire de Recherche comme un Dictionnaire

- Ouvrez le fichier Fiche.java et lancez Test.java
  Vous remarquez que les deux fiches créées ont des numéros d'objet (HashCode) différents. Ce qui est normal.
- Créez dans une variable f4, une troisième fiche avec exactement les mêmes nom, prénom, statut que la première fiche. Affichez f4. Quel numéro ? Est-ce normal ?

Il n'est pas normal d'avoir des fiches identiques dans leur contenu, redondantes en terme d'objet.

- Faites un ArbreBR de Fiche, la relation d'ordre sera celle définie dans Fiche.
  (Faites une copie de vos fichiers précédents pour cette nouvelle version)

- Concevez en vous appuyant sur la méthode recherche, la méthode rechercheRef() dans ArbreBR.
  Cette méthode permet de trouver un *element* dans le dictionnaire et renvoie l'objet trouvé dans le dico.
  public Fiche rechercheRef (Fiche *element*)

Quel intérêt ?
Si vous utilisez une Fiche dans un main qui est déjà dans le dico alors il ne faut pas réserver de la place inutilement pour une Fiche déjà en mémoire RAM et présente dans le dictionnaire.

- Créez un dictionnaire avec une dizaine de fiches dans le main.
- Testez rechercherRef dans le main avec la création d'une nouvelle fiche que vous savez présente par son contenu dans le dico.
- Affichez la fiche retournée par la fonction et observez le numéro. Comparez-le numéro avec la fiche de crée.

Exercice supplémentaire sur les arbres binaires.

Document

Structurer un document avec des titres, texte, image

Construire un document

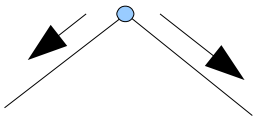
Nous allons mettre en place un programme permettant de gérer la structure d'un document composé de Titre, Texte, Image.

```
1 - Introduction
2 - Explications
    21 - Le figuratif
    22 - Peintres du 19ieme siecle
            Pierre-Auguste Renoir dit Auguste Renoir, né
             <IMAGE> : D:/renoir.jpg
            Claude Monet est né à Paris le 14 novembre 1
             <IMAGE> : D:/monet.jpg
    23 - Du figuratif à l'abstration
3 - Conclusion
```

Nous allons utiliser un arbre binaire afin de stocker les données.

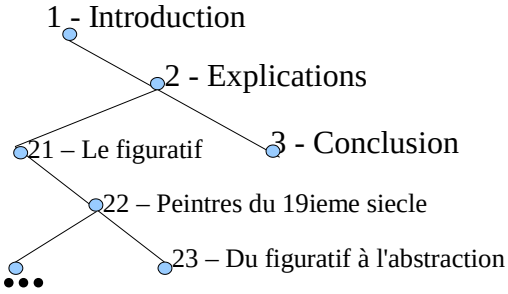
La règle de stockage choisie est la suivante :

Le contenu d'un element à gauche du noeud



Éléments consécutifs de même niveau à droite de la racine

ex :



- Ouvrez le fichier Doc.java

Prenez connaissance des classes et de l'organisation établie.

- Ouvrez les fichiers Arbre.java et TestArbre.java
- Exécutez le programme.

Nous voulons construire le document suivant grâce au main(). Complétez le code pour obtenir l'arborescence suivante :

```
1 - Introduction
2 - Explications
    21 - Le figuratif
    22 - Peintres du 19ieme siecle
            Pierre-Auguste Renoir dit Auguste Renoir, né
             <IMAGE> : D:/renoir.jpg
            Claude Monet est né à Paris le 14 novembre 1
             <IMAGE> : D:/monet.jpg
    23 - Du figuratif à l'abstration
3 - Conclusion
```

(La numérotation n'est pas à jour nous allons nous en occuper plus tard.)

Affichage

- Faites la méthode afficherTitre(d) qui permet d'afficher uniquement les Titres d'un document dans l'ordre suivant :

```
5 - Conclusion
    2 - Du figuratif à l'abstration
    3 - Peintres du 19ieme siecle
    1 - Le figuratif
3 - Explications
2 - Introduction
```

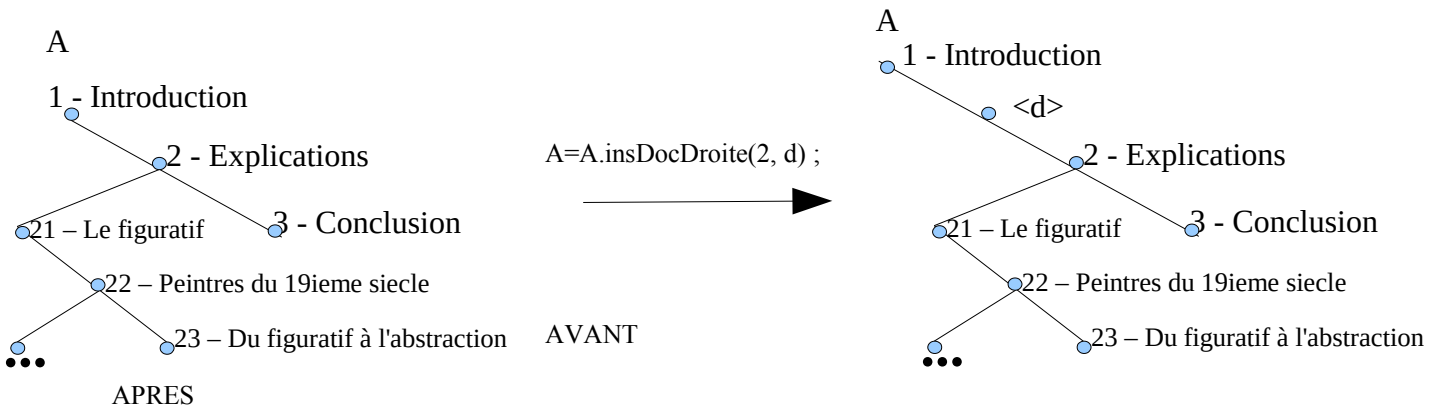
Numérotation des titres

Nous voulons que les titres et sous-titres soient correctement numérotés

- Faites la méthode récursive public void numerotation (int compteur)

Insertion d'un élément

Nous voulons insérer récursivement un élément dans un document existant. Pour le moment, on va se contenter d'insérer l'élément *d* à l'emplacement *position* sur un des nœud à droite.
public Arbre insDocDroite (int position, Doc d)



Cas particuliers : si nbNoeud à droite < position on ajoute d à la fin, si position=0 tout au début.