Algorithmique et structures de données

TD / TP 9

C. Sempé

Les objectifs de cette séance de TD / TP sont :

- Concevoir puis utiliser les Arbres Binaires
- Utiliser la programmation récursive
- Explorer un arbre binaire

TD

Nous allons concevoir une classe Arbre implémentée à base d'arbre binaire et capable de stocker des objets de type String.

Nous avons vu en cours la première partie de cette classe :

```
public abstract class Arbre
{
   public abstract String getRacine();
   public abstract Arbre getAg();
   public abstract Arbre getAd();

   public abstract void setRacine(String s);
   public abstract void setAg(Arbre Ag);
   public abstract void setAd(Arbre Ad);


   public abstract boolean estVide();

   public abstract void afficherGRD();

   public abstract String lePlusAGauche();

   public abstract Arbre supprimer( String val );

   public abstract boolean trouver( String val );

   public abstract boolean estFeuille();

   public abstract int nbFeuilles();

   public abstract int nbNoeuds ();

   public abstract int hauteur ();

}

class ArbreVide extends Arbre
{
   public ArbreVide ()
   {
   }

   public String getRacine() { return null; }
   public Arbre getAg() { return this; }
   public Arbre getAd() { return this; }

   public void setRacine(String s) {  }
   public void setAg(Arbre Ag) {  }
   public void setAd(Arbre Ad) {  }

   public boolean estVide()
   {
```

```
        return true;
    }


    public void afficherGRD()
    {
        // System.out.print(" vide ");
    }

    public boolean estFeuille()
    {
        return false;
    }

}

class ArbreCons extends Arbre
{
    private String racine;
    private Arbre Ag;
    private Arbre Ad;

    public boolean estVide()
    {
        return false;
    }

    public ArbreCons(String val, Arbre Ag, Arbre Ad)
    {
        this.racine = val; this.Ag = Ag; this.Ad = Ad;
    }

    public ArbreCons( String val) // constructeur de feuille
    {
        this.racine = val; this.Ag = new ArbreVide(); this.Ad = new ArbreVide();
    }


    public String getRacine() { return this.racine; }
    public Arbre getAg() { return this.Ag; }
    public Arbre getAd() { return this.Ad; }

    public void setRacine(String s) { this.racine = s; }
    public void setAg(Arbre Ag) { this.Ag = Ag; }
    public void setAd(Arbre Ad) { this.Ad = Ad; }



    public boolean estFeuille()
    {
        return this.getAg().estVide() && this.getAd().estVide();
    }

}
```

- Faites la méthode afficheGRD()
  public void afficheGRD ( )

- Faites une méthode qui détermine le nombre de feuilles de l'arbre
  public int nbFeuilles ()

- Faites une méthode qui effectue la recherche d'un élément dans un arbre
  public boolean trouver (String element)

- Faites une méthode qui donne l'élément le plus à droite dans l'arbre
  public String lePlusAGauche ()

- Faites la méthode qui supprime un élément dans l'arbre
  public Arbre supprimer(String valeur)

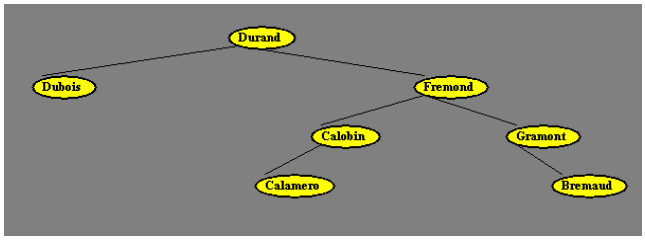
- Faites un nouveau constructeur *Arbre(int niveau)* qui construira récursivement un arbre complet. On placera, pour le moment, *null* dans le champ Racine sur chaque noeud de l'arbre.

TP

Ouvrez le projet **Tournoi**. L'application va nous servir dans un premier temps pour tester les méthodes de Arbre puis à modéliser un tournoi de tennis.

Dessiner un arbre dans une Fenêtre

- Ouvrez le fichier Fenetre.java
- Complétez l'action du bouton "Exemple". Ce bouton permet de construire un arbre en l'accrochant à la variable d'instance this.arb. Cela nous permettra de tester les méthodes de Arbre.
- Faites en sorte qu'il construise l'arbre suivant :



Remarquez que la variable d'instance supportant l'arbre à afficher se nomme ***arb***

Observez le code. Comment se dessine l'arbre ?

- Testez l'action du bouton Supprimer afin qu'il supprime un élément de l'arbre après l'avoir saisi dans le champ texte txt.
- Complétez l'action du bouton Rechercher afin qu'il permette de rechercher un élément de l'arbre après l'avoir saisi dans le champ texte txt.

Tournoi de tennis

Nous allons gérer un tournoi de tennis. Pour cela nous allons avoir à mémoriser les noms des joueurs ainsi que la structure modélisant les matchs.

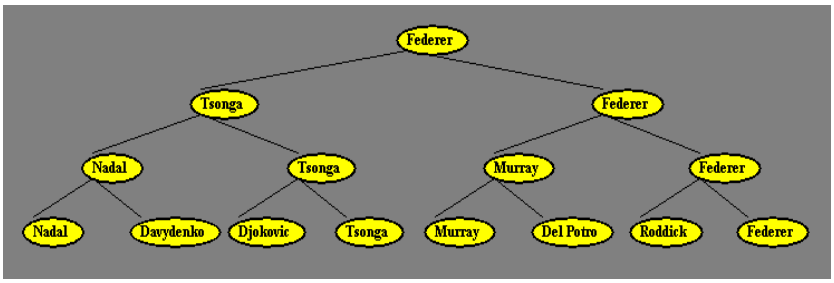
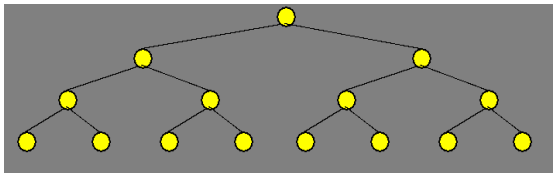


Tableau du tournoi

- Faire un nouveau constructeur *Arbre(niveau)* qui construira récursivement un arbre complet.
  On placera, pour le moment, *null* dans le champ Racine sur chaque noeud de l'arbre.



niveau=3
Joueurs possibles sur feuille
$=2^{niveau}$

- Complétez l'action du bouton "Construire tableau". Ce bouton permet de construire un arbre complet de niveau 3 accroché à la variable d'instance this.arb

- Testez l'affichage de l'arbre

Insertion d'un joueur dans le tableau du tournoi

- Concevez la méthode insereFeuille dans la classe Arbre qui va nous permettre de placer les joueurs sur les feuilles de notre arbre. Le booléen permet de savoir si l'insertion à réussie
  public boolean insereFeuille(String valeur)

Stratégie:

Si on est sur une feuille et que la racine est à null alors placer le nom sinon l'insertion à cet endroit précis à échoué.
Si on n'est pas sur une feuille alors insérez le nom à gauche, en cas d'échec essayez à droite cette fois.

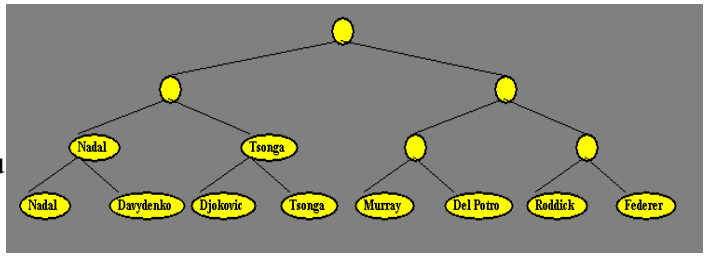
- Testez en saisissant un nom dans la zone de texte puis en appuyant sur "Inserer un joueur"

- Utiliser le bouton "Charger tournoi" afin que les joueurs se placent en bas du tableau du tournoi.

Placer le gagnant d'un match

La méthode parcours le tableau du tournoi et insère le gagnant si cela est possible.

Stratégie :



On peut insérer le nom sur un noeud de l'arbre
si ce même nom est présent sur la racine immédiate du
du sous-arbre gauche ou celui de droite.

Attention : prenez la précaution de vérifier que les
sous-arbres existent avant de vouloir les manipuler. Idem pour le champ racine.

- Concevez la méthode puis testez la avec le bouton "Saisir résultat"
  public void placerGagnant (String element)

- Faites la méthode adversaires() qui renvoie la liste des adversaires rencontraient par le gagnant du tournoi.
  Le tableau des rencontres doit être complet bien sûr.

  public void adversaires(ArrayList adv)
  adv est une variable d'instance supplémentaire dans la Fenêtre. paint() se charge d'afficher adv.