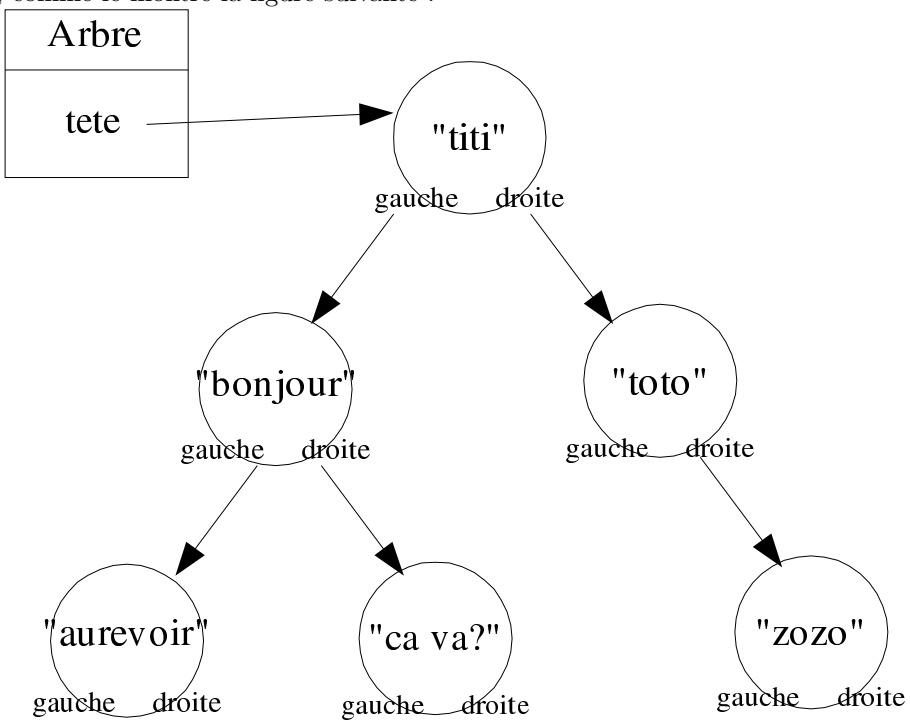
TD 6 : arbre binaire et récursivité

Les arbres binaires sont des structures efficaces pour rechercher des données par valeur (temps logarithmique). L'arbre est constitué de nœuds. Chaque nœud contient une donnée (ici un `String`) et peut avoir deux nœuds fils, comme le montre la figure suivante :



Un fils gauche aura toujours une valeur alphabétiquement inférieure au noeud parent qui lui même sera alphabétiquement inférieur à son fils droit.

Nous allons implémenter ceci à l'aide de deux classes `Arbre` et `Noeud`. La classe `Arbre` contiendra un seul attribut membre appelé `tete`, qui vaudra `null` si l'arbre est vide.

Pour comparer deux chaînes de caracteres (`String`) entre elles, on pourra utiliser la méthode `int compareTo(String)` de cette classe. Par exemple `a.compareTo(b)` renvoie une valeur négative si `a` est avant `b` en ordre alphabétique.

Voici la fonction `main` que nous souhaitons voir fonctionner :

```
class ArbreTest
{
    public static void main(String [] args)
    {
        Arbre arbre=new Arbre();
        arbre.insererValeur("titi");
        arbre.insererValeur("toto");
        arbre.insererValeur("bonjour");
        arbre.insererValeur("aurevoir");
        arbre.insererValeur("ca va?");
        arbre.insererValeur("zozo");
        Noeud n1=arbre.trouver("toto");
        Noeud n2=arbre.trouver("aurevoir");
        arbre.afficher();
    }
```

```
}
```

Implémentez les classes `Arbre` et `Noeud` pour que cette fonction `main` puisse fonctionner. Les algorithmes (très simples) de recherche et d'insertion pourront être implémentés en utilisant des méthodes récursives. Si l'arbre est correctement implémenté, la fonction afficher devra afficher les éléments de l'arbre en ordre alphabétique.