# Structures de données

Une référence (en anglais) :

http://java.sun.com/docs/books/

tutorial/collections/TOC.html

# Structures de données non linéaires

Exemples :

- Arbres et forêts
- Tas
- Ensembles génériques
- Graphes

# Structures de données non linéaires

Exemples :

{ 12 , {3 , 15} ,874 }

est un ensemble comportant 3 éléments :

- 12
- { 3 , 15 } qui est lui même un ensemble
- 874

# Structures de données non linéaires

Exemples :

{ 12 , {3 , 15} ,874 }

est différent de l'ensemble

{ 12 , 3 , 15 , 874 }

comportant 4 éléments

# Structures de données non linéaires

Exemples :

{12,{3,15},874}≠{12,3,15,874}

La différence est la structuration,

pas les données élémentaires

# Structures de données
# Arbres binaires

Un arbre binaire sommet-étiqueté est une structure de données récursive répondant à la spécification :

Constructeurs :

Abvide : $\rightarrow$ AB

ABetq : AB x T x AB $\rightarrow$ AB

Où T est le type des étiquettes

# Structures de données
# Arbres binaires

Un arbre binaire sommet-étiqueté est une structure de données récursive répondant à la spécification :

Observateurs :

filsGauche : AB → AB

filsDroit : AB → AB

Etiquette : AB → T

# Structures de données
# Arbres binaires

Un arbre binaire sommet-étiqueté est une structure de données récursive répondant à la spécification :

Reconnaisseurs :

estABvide : AB → bool

# Structures de données
# Arbres binaires

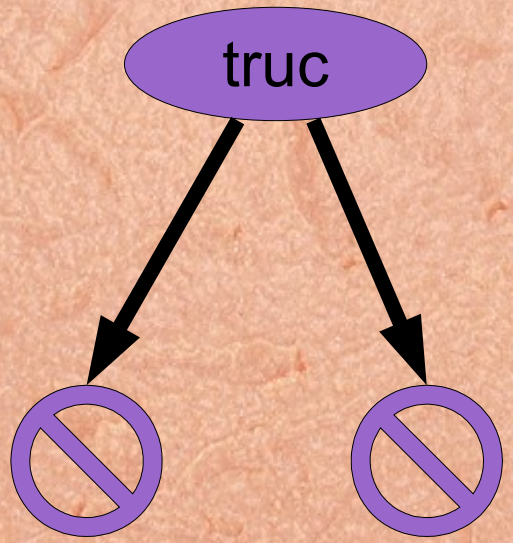
Axiomes :

```
estABvide(ABvide()) == vrai

estABvide(ABetq(.,.,.))== faux

Etiquette(ABvide()) == ERREUR

Etiquette(ABetq(.,t,.)) == t

filsGauche(ABvide()) == ERREUR

filsGauche(ABetq(g,.,.)) == g

… de même pour filsDroit
```

# Structures de données
# Arbres binaires

Exemples :

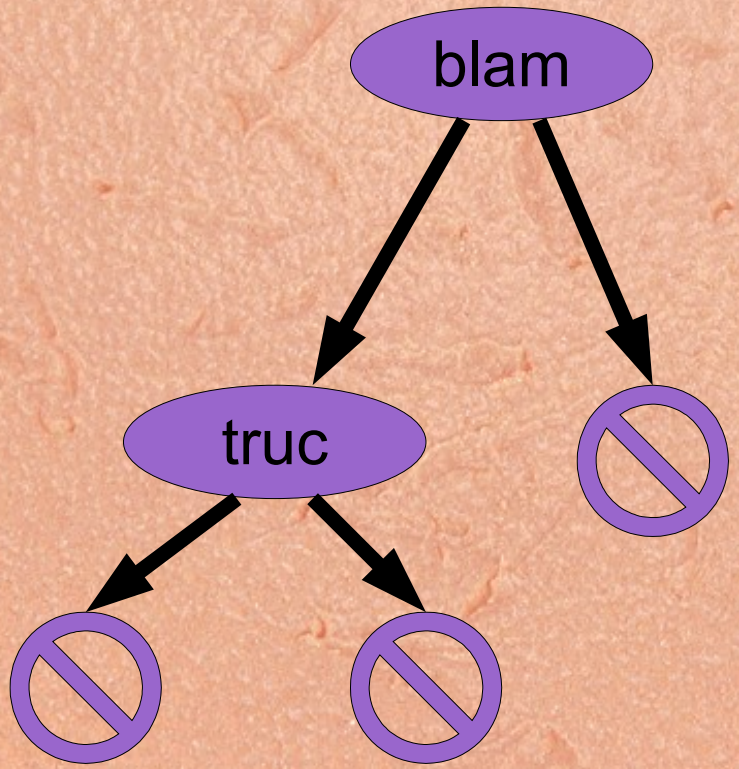
TR ← Abetq(ABvide(),“truc“,ABvide())



# Structures de données
# Arbres binaires

Exemples :

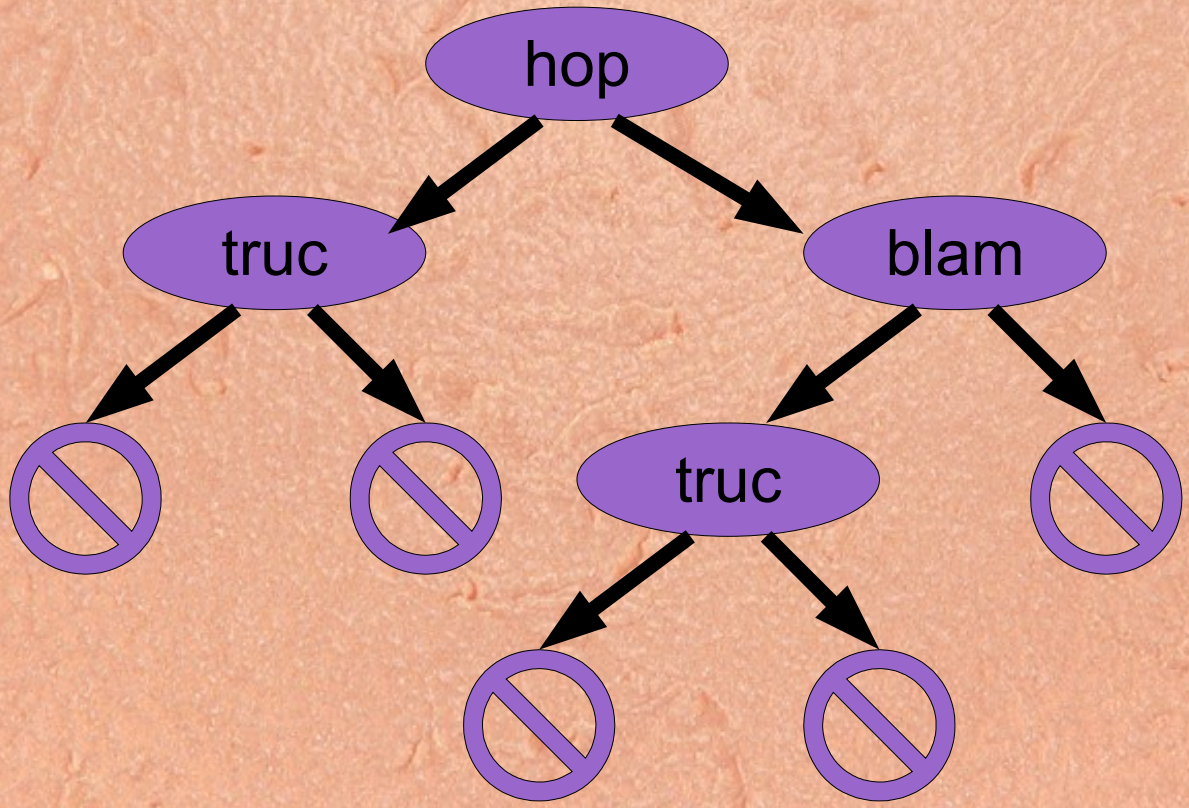
```
TR ← Abetq(ABvide(),“truc“,ABvide())
BL ← Abetq(TR,“blam“,ABvide())
```



# Structures de données
# Arbres binaires

Exemples :

```
TR ← Abetq(ABvide(),“truc“,ABvide())

BL ← Abetq(TR,“blam“,ABvide())

HO ← Abetq(TR,“hop“,BL)
```



# Structures de données
# Arbres binaires non vides

Un arbre binaire non vide sommet-étiqueté est une structure de données récursive répondant à la spécification :

Constructeurs :

`feuille : T → ABnv`

`embr : ABnv x T x ABnv → ABnv`

Où T est le type des étiquettes

# Structures de données
# Arbres binaires non vides

Un arbre binaire non vide sommet-étiqueté répond à la spécification :

<u>Observateurs</u> :

etq : ABnv → T

filsG : ABnv → Abnv

filsD : ABnv → ABnv

# Structures de données
# Arbres binaires non vides

Un arbre binaire non vide sommet-étiqueté répond à la spécification :

Reconnaisseurs :

`estFeuille : ABnv → bool`

# Structures de données
# Arbres binaires non vides

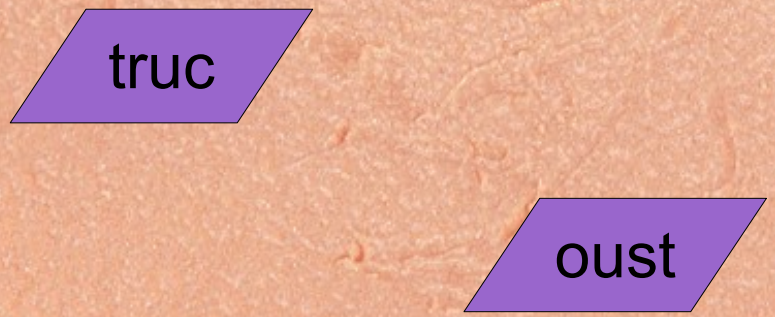
Axiomes :

```
estFeuille(feuille(.)) == vrai
estFeuille(embr(.,.,.)) == faux
etq(feuille(t)) == t
etq(embr(.,t,.) == t
filsG(feuille(.)) == ERREUR
filsG(embr(g,.,.)) == g
filsD(feuille(.)) == ERREUR
filsD(embr(.,.,d)) == d
```

# Structures de données
# Arbres binaires non vides

Exemples :

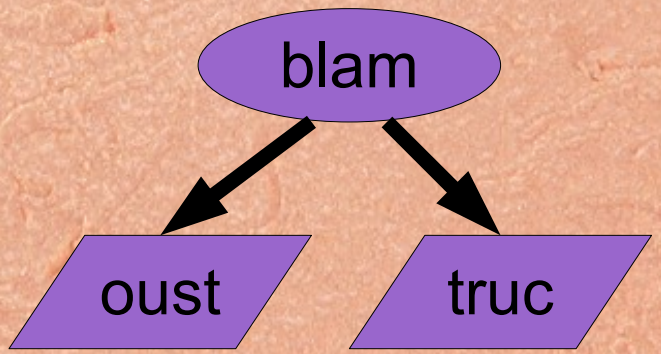
```
TR ← feuille("truc")

ST ← feuille("oust")
```



# Structures de données
# Arbres binaires non vides

Exemples :

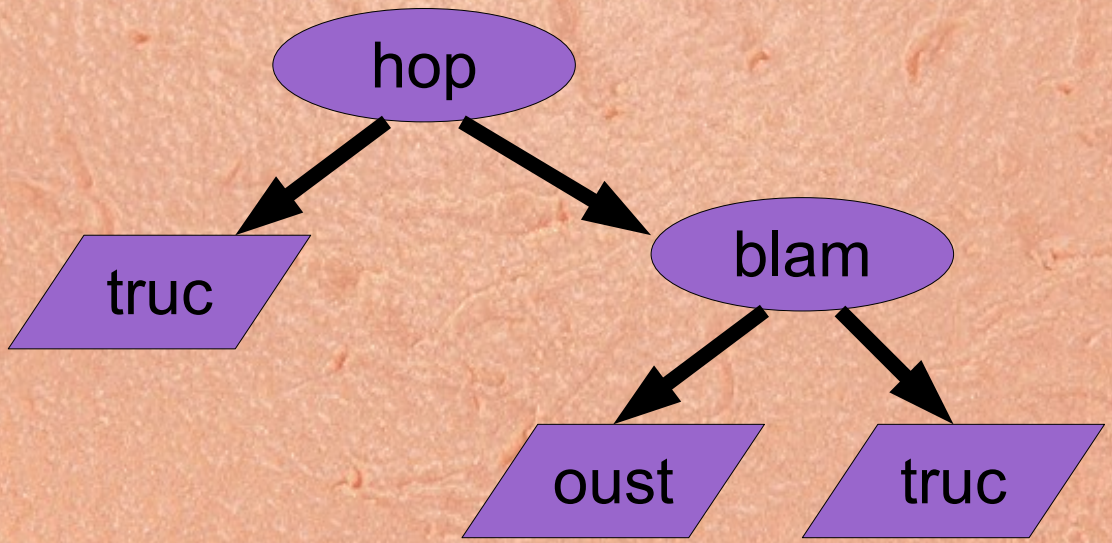
```
TR ← feuille("truc")

ST ← feuille("oust")

BL ← embr(ST,"blam",TR)
```



# Structures de données
# Arbres binaires non vides

Exemples :

```
TR ← feuille("truc")
ST ← feuille("oust")
BL ← embr(ST,"blam",TR)
HO ← embr(TR,"hop",BL)
```



# Structures de données
# Arbres binaires non vides

## Premier exemple d'implémentation :

```
public class ABnv <T> { // version procédurale
   T etq;ABnv<T> g;ABnv<T> d;
   ABnv(T etq) // remplace feuille
        {this.etq=etq;g=null;d=null;}
   public boolean estFeuille()
      {return g==null && d==null;}
   public T etq(){return etq;}
   public ABnv<T> filsG(){return g;}
   public ABnv<T> filsD(){return d;}
   public void changeFilsG(ABnv<T> fg) {g=fg;}
   public void changeFilsD(ABnv<T> fd) {d=fd;}
   public void changeEtq(T eti) {etq=eti;}
}
```

# Structures de données
# Arbres binaires non vides

## Premier exemple d'implémentation :

```
public static void main(String[] args) {
      ABnv <String> ar ;ABnv <String> br ;ABnv <String> cr ;
      ar=new ABnv<String>("arf"); // feuille
      br=new ABnv<String>("blop"); // feuille
      cr=new ABnv<String>("crisp"); // feuille
      ar.changeFilsG(br); // un fils
      cr.changeFilsG(br);cr.changeFilsD(ar);  // deux fils
      System.out.println(br.etq());
      System.out.println(cr.etq());
      System.out.println(cr.filsG().etq());
      System.out.println(cr.filsD().filsG().etq());
      System.out.println(cr.filsD().filsD().estFeuille());

   }
```

# Structures de données
# Arbres binaires non vides

## Second exemple d'implémentation :

```
public class ABnv2 <T> { // version fonctionnelle
   T etq;ABnv2<T> g;ABnv2<T> d;
   ABnv2(T etq) {this.etq=etq;g=null;d=null;}
   public boolean estFeuille()
      {return g==null && d==null;}
   public T etq(){return etq;}
   public ABnv2<T> filsG(){return g;}
   public ABnv2<T> filsD(){return d;}
   public ABnv2<T> embr( T eti, ABnv2<T> fd) {
      ABnv2<T> resultat=new ABnv2<T>(eti);
      resultat.g=this;resultat.d=fd;
      return resultat;}
}
```

# Structures de données
# Arbres binaires non vides

## Second exemple d'implémentation :

```
public static void main(String[] args) {
      ABnv2 <String> ar ; ABnv2 <String> br ;
      ABnv2 <String> cr ; ABnv2 <String> dr ;
      ar=new ABnv2<String>("arf"); // feuille
      br=new ABnv2<String>("blop"); // feuille
      cr=ar.embr("crisp",br); // deux feuilles
      dr=cr.embr("douhouap", br); // trois feuilles
      System.out.println(br.etq());
      System.out.println(cr.etq());
      System.out.println(cr.filsG().etq());
      System.out.println(dr.filsG().filsD().etq());
      System.out.println(
            dr.filsG().filsD().estFeuille());
   }
```

# Structures de données
# Arbres binaires

## Exemple d'implémentation :

```
public class AB <T>{
private class Noeud {T etq;AB<T> gche;AB<T> drte;}
            Noeud contenu;
   AB(){contenu=null;}
   public boolean estVide(){return contenu==null;}
   public AB<T> etq(T etq, AB<T> droit) {
      AB <T> resultat=new AB<T>();
      resultat.contenu=new Noeud();
      resultat.contenu.gche=this;
      resultat.contenu.etq=etq;
      resultat.contenu.drte=droit;
      return resultat;
   }
   public AB<T> filsG(){return contenu.gche;}
   public AB<T> filsD(){return contenu.drte;}
   public T etq() {return contenu.etq;}
}
```

# Structures de données
# Arbres binaires

## Exemple d'implémentation :

```
public static void main(String[] args) {
     AB<String> ar ;AB<String> br ;
     AB<String> cr ;AB<String> dr ;
     ar=new AB<String>(); // arbre vide
     br=ar.etq("gauche",ar); // feuille
     cr=br.etq("haut", ar); // un seul fils
     dr=cr.etq("hop", br);  // deux fils
     System.out.println(br.etq());
     System.out.println(cr.etq());
     System.out.println(dr.etq());
     System.out.println(dr.filsG().etq());
     System.out.println(dr.filsG().filsG().etq());
     System.out.println(
                  dr.filsG().filsD().estVide());

  }
```

# Structures de données
# Arbres binaires

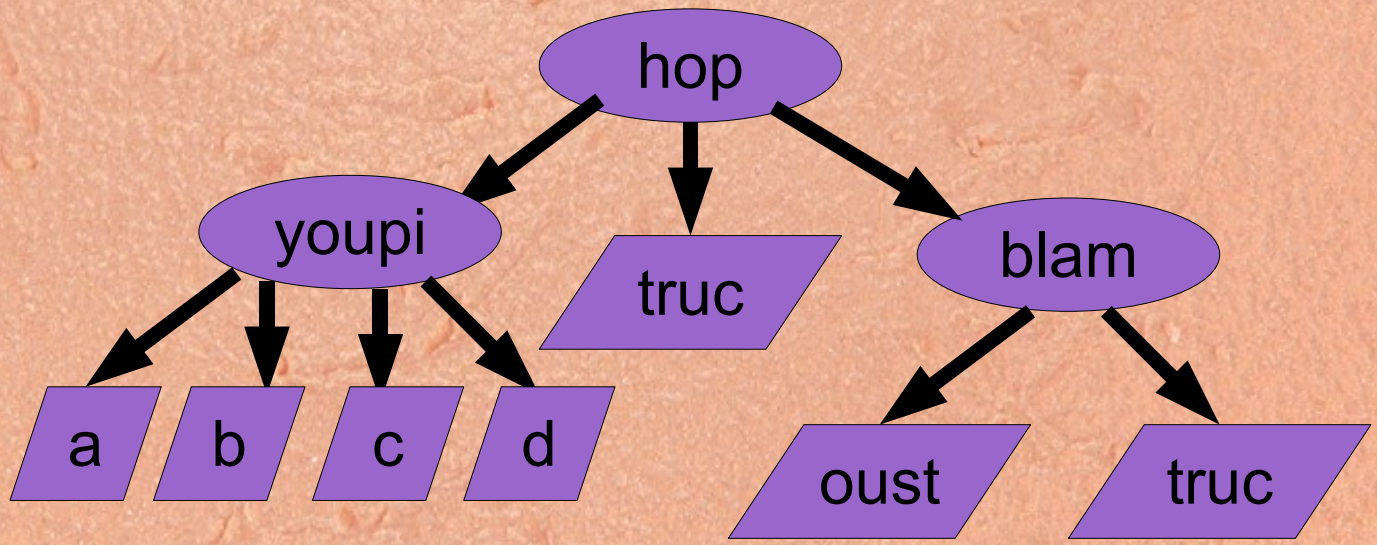
Exemple de fonction récursive :

```
public int nbNoeuds() {
      if (this.estVide())
                              {return 0;}
      else
            {return 1 + this.filsG().nbNoeuds()
                        + this.filsD().nbNoeuds(); }
   }
```

# Structures de données
# Arbres n-aires/génériques

On peut définir des arbres à 3 fils, à 4 fils, ...

On peut aussi définir des arbres à nombre de fils variable :



# Structures de données
# Arbres n-aires/génériques

On peut définir des arbres à 3 fils, à 4 fils, ...

On peut aussi définir des arbres à nombre de fils variable :