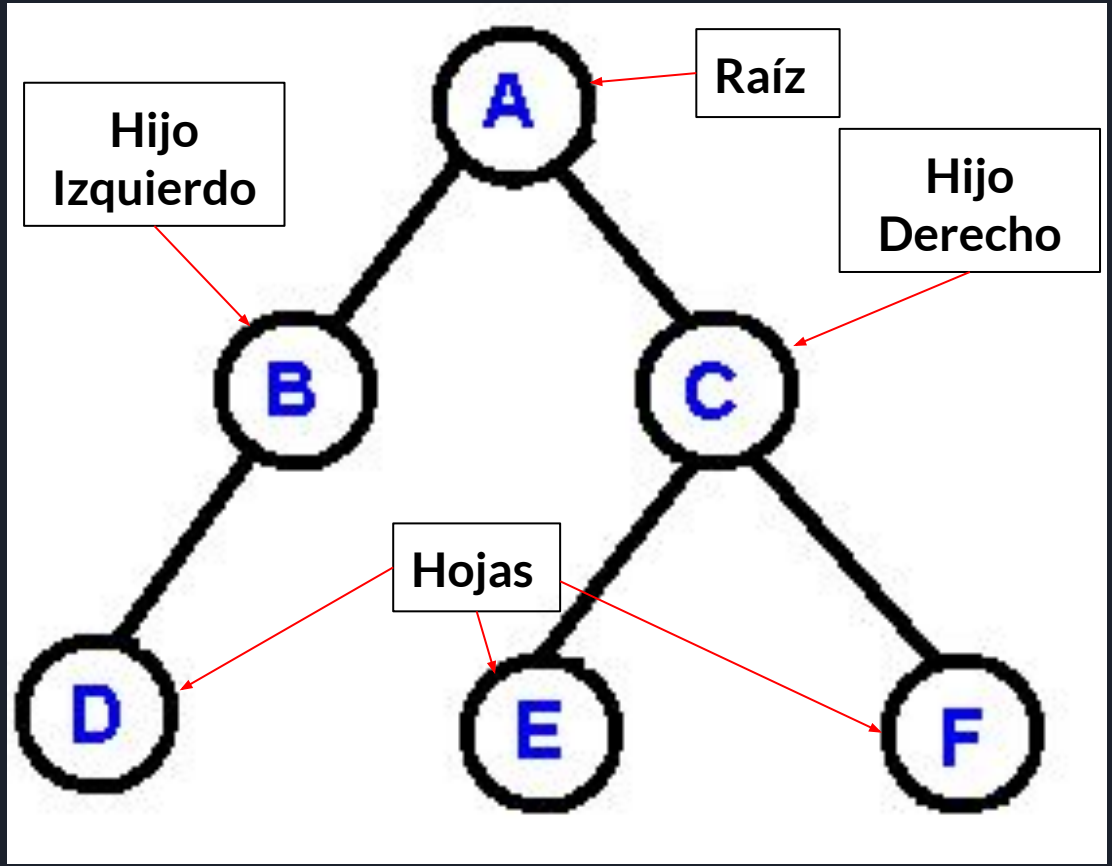
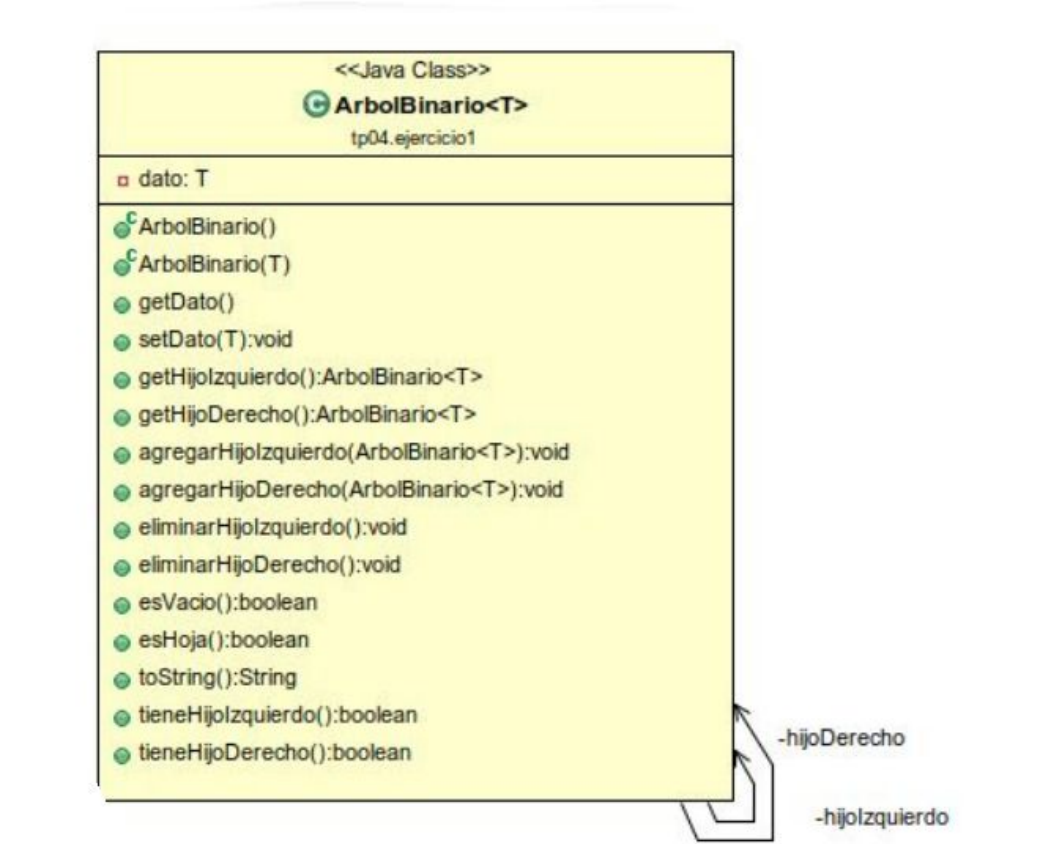
Programación 3 - Explicación Árboles Binarios

Árboles Binarios



Árboles Binarios - Estructura en JAVA



Árboles Binarios - Recorridos

- **Preorden**
  - Se procesa primero la raíz y luego sus hijos, izquierdo y derecho.
- **Inorden**
  - Se procesa el hijo izquierdo, luego la raíz y último el hijo derecho
- **Postorden**
  - Se procesan primero los hijos, izquierdo y derecho, y luego la raíz

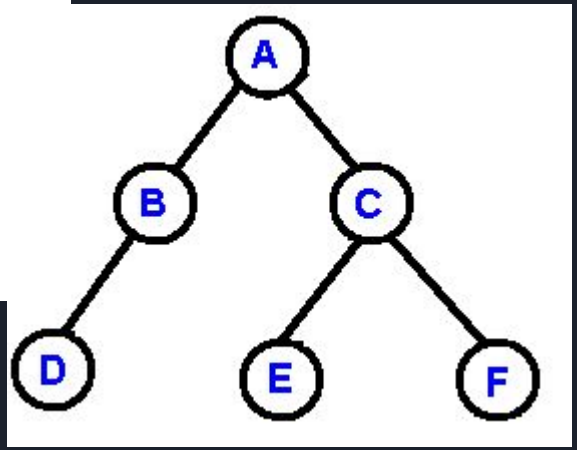
En profundidad Recursivo

- **Por niveles**
  - Se procesan los nodos teniendo en cuenta sus niveles, primero la raíz, luego los hijos, los hijos de éstos, etc.

Por niveles Iterativo

Recorrido preorden

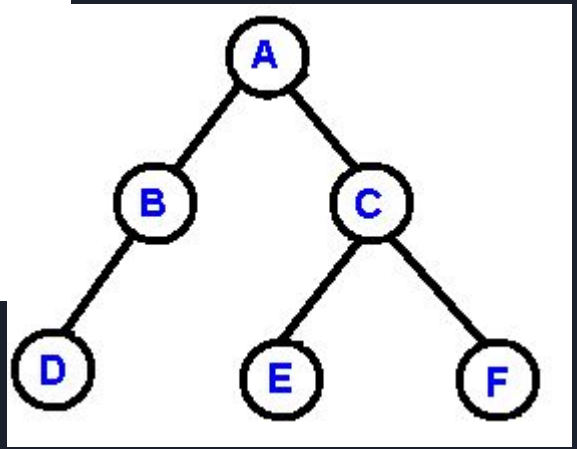
```java
public static void preOrden(ArbolBinario<Character> arbol) {
    // Proceso la raíz
    System.out.println(arbol.getDato());
    // Proceso el hijo izquierdo
    if (arbol.tieneHijoIzquierdo())
        preOrden(arbol.getHijoIzquierdo());
    // Proceso el hijo derecho
    if (arbol.tieneHijoDerecho())
        preOrden(arbol.getHijoDerecho());
}
```



El recorrido preorden, ¿qué imprime?

Recorrido preorden

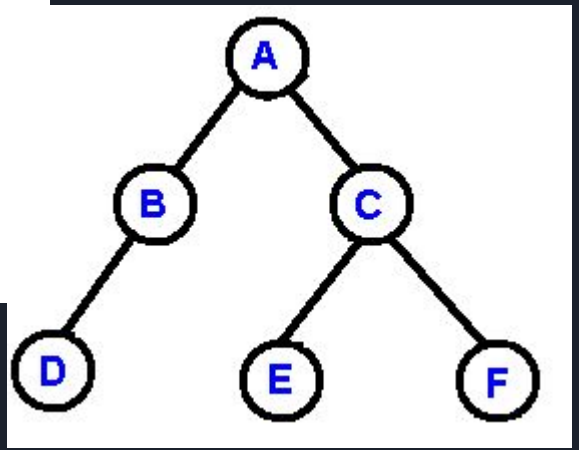
```java
public static void preOrden(ArbolBinario<Character> arbol) {
    // Proceso la raíz
    System.out.println(arbol.getDato());
    // Proceso el hijo izquierdo
    if (arbol.tieneHijoIzquierdo())
        preOrden(arbol.getHijoIzquierdo());
    // Proceso el hijo derecho
    if (arbol.tieneHijoDerecho())
        preOrden(arbol.getHijoDerecho());
}
```



A B D C E F

Recorrido inorden

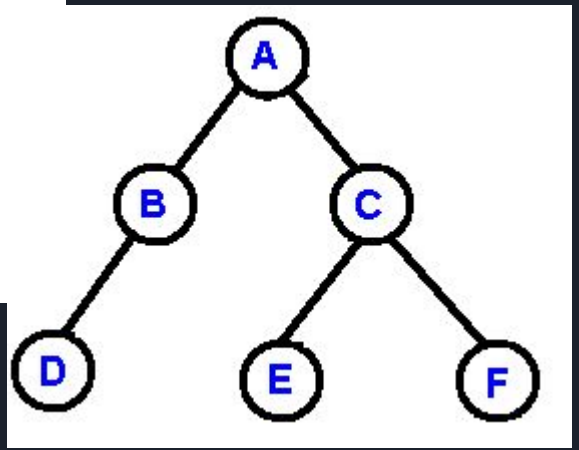
```java
public static void inOrden(ArbolBinario<Character> arbol) {
    // Proceso el hijo izquierdo
    if (arbol.tieneHijoIzquierdo())
        inOrden(arbol.getHijoIzquierdo());
    // Proceso la raíz
    System.out.println(arbol.getDato());
    // Proceso el hijo derecho
    if (arbol.tieneHijoDerecho())
        inOrden(arbol.getHijoDerecho());
}
```



El recorrido inorden, ¿qué imprime?

Recorrido inorden

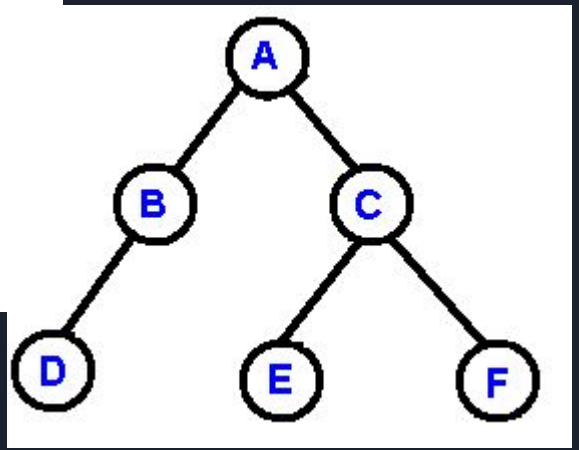
```java
public static void inOrden(ArbolBinario<Character> arbol) {
    // Proceso el hijo izquierdo
    if (arbol.tieneHijoIzquierdo())
        inOrden(arbol.getHijoIzquierdo());
    // Proceso la raíz
    System.out.println(arbol.getDato());
    // Proceso el hijo derecho
    if (arbol.tieneHijoDerecho())
        inOrden(arbol.getHijoDerecho());
}
```



D B A E C F

Recorrido postorden

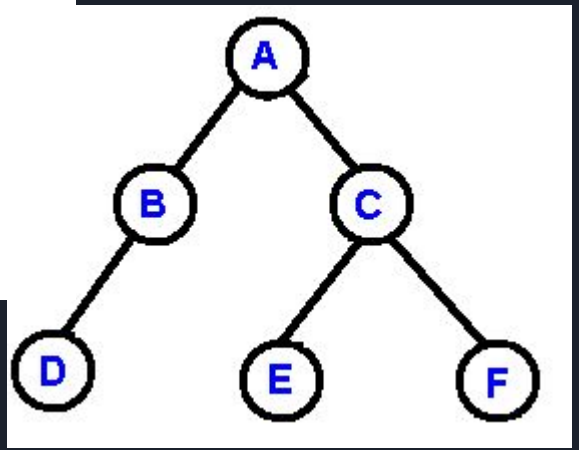
```java
public static void postOrden(ArbolBinario<Character> arbol) {
    // Proceso el hijo izquierdo
    if (arbol.tieneHijoIzquierdo())
        postOrden(arbol.getHijoIzquierdo());
    // Proceso el hijo derecho
    if (arbol.tieneHijoDerecho())
        postOrden(arbol.getHijoDerecho());
    // Proceso la raíz
    System.out.println(arbol.getDato());
}
```



El recorrido postorden, ¿qué imprime?

Recorrido postorden

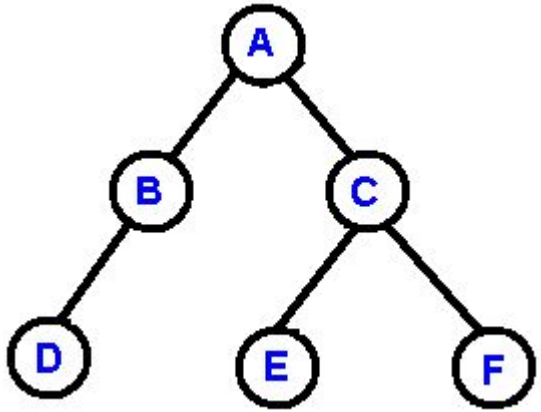
```java
public static void postOrden(ArbolBinario<Character> arbol) {
    // Proceso el hijo izquierdo
    if (arbol.tieneHijoIzquierdo())
        postOrden(arbol.getHijoIzquierdo());
    // Proceso el hijo derecho
    if (arbol.tieneHijoDerecho())
        postOrden(arbol.getHijoDerecho());
    // Proceso la raíz
    System.out.println(arbol.getDato());
}
```



D B E F C A

Recorrido por niveles

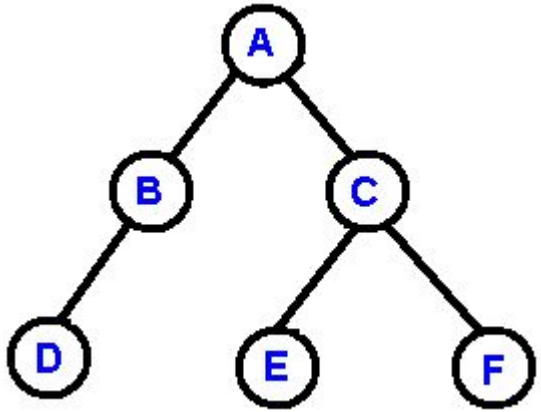
```java
public static void porNiveles(ArbolBinario<Character> arbolDeLetras) {
    ColaGenerica<ArbolBinario<Character>> cola = new ColaGenerica<ArbolBinario<Character>>();
    ArbolBinario<Character> arbol = null;
    cola.encolar(arbolDeLetras);
    cola.encolar(null);
    while (!cola.esVacia()) {
        arbol = cola.desencolar();
        if (arbol != null) {
            System.out.println(arbol.getDato());
            if (arbol.tieneHijoIzquierdo())
                cola.encolar(arbol.getHijoIzquierdo());
            if (arbol.tieneHijoDerecho())
                cola.encolar(arbol.getHijoDerecho());
        } else if (!cola.esVacia()) {
            cola.encolar(null);
        }
    }
}
```



El recorrido por niveles , ¿qué imprime?

Recorrido por niveles

```java
public static void porNiveles(ArbolBinario<Character> arbolDeLetras) {
    ColaGenerica<ArbolBinario<Character>> cola = new ColaGenerica<ArbolBinario<Character>>();
    ArbolBinario<Character> arbol = null;
    cola.encolar(arbolDeLetras);
    cola.encolar(null);
    while (!cola.esVacia()) {
        arbol = cola.desencolar();
        if (arbol != null) {
            System.out.println(arbol.getDato());
            if (arbol.tieneHijoIzquierdo())
                cola.encolar(arbol.getHijoIzquierdo());
            if (arbol.tieneHijoDerecho())
                cola.encolar(arbol.getHijoDerecho());
        } else if (!cola.esVacia()) {
            cola.encolar(null);
        }
    }
}
```



A B C D E F

Método para contar hojas de un Árbol Binario

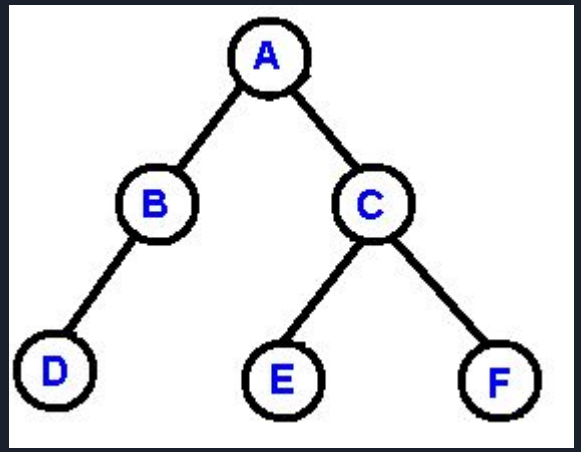
```java
public static int contarHojas(ArbolBinario<Character> arbol) {
    int hojas = 0;
    if (arbol.esHoja())
        return 1; // también puede ser hojas=1;
    else {
        if (arbol.tieneHijoIzquierdo())
            hojas = contarHojas(arbol.getHijoIzquierdo());

        if (arbol.tieneHijoDerecho())
            hojas = hojas + contarHojas(arbol.getHijoDerecho());
    }
    return hojas;
}
```

¿Qué cambios debe realizar si el método contarHojas() se implementa dentro de la clase ArbolBinario?

Frontera

Escribir un método que devuelva en una lista la frontera de un Árbol Binario.

¿Cuál sería la frontera de éste árbol?



D E F

Frontera

```java
public static ListaGenerica<Character> frontera(ArbolBinario<Character> arbol) {
    ListaGenerica<Character> listaFrontera = new ListaGenericaEnlazada<Character>();
    fronteraRecursivo(listaFrontera, arbol);
    return listaFrontera;
}

private static void fronteraRecursivo(ListaGenerica<Character> listaFrontera, ArbolBinario<Character> arbol) {
    if (arbol.tieneHijoIzquierdo())
        fronteraRecursivo(listaFrontera, arbol.getHijoIzquierdo());
    if (arbol.esHoja())
        listaFrontera.agregarFinal(arbol.getDato());
    if (arbol.tieneHijoDerecho())
        fronteraRecursivo(listaFrontera, arbol.getHijoDerecho());
}
```