public class ArbolBinario<T>{

...

public int cantidadHijosPares () { INORDEN

int cant = 0;

if(!this.getHijoIzquierdo().esVacio())

cant += this.getHijoIzquierdo(). cantidadHijosPares ();

if(this.esHoja())

cant++;

if (!this.getHijoIzquierdo().esVacio() && !this.getHijoDerecho().esVacio())

cant++;

if(!this.getHijoDerecho().esVacio())

cant += this.getHijoDerecho(). cantidadHijosPares ();

return cant;

}

}

public class ContadorArbol {

private ArbolBinario<Integer> arbol;

public int contadorOcurrencias (int elem) {

int contador = 0; POSTORDEN

if (!this.getHijoIzquierdo().esVacio()) {

contador = contador + arbol.getHijoIzquierdo().contadorOcurrencias(elem);}

if (!this.getHijoDerecho().esVacio()) {

contador = contador + arbol.getHijoDerecho().contadorOcurrencias(elem);}

if (this.getDatoRaiz() == elem){

contador++;}

return contador;

}

public class ValidadorArbol {

private ArbolBinario <Integer> arbolB;

public boolean esMonodistante (int k) { PREORDEN

If (this.getArbolB().esVacio()) return false;

return esMonodistante (this.getArbolB(), k);

}

public boolean esMonodistante ( ArbolBinario <Integer> arbol, int k) {

Boolean ok=true;

If (arbol.esHoja()) return arbol.getDatoRaiz()-k==0;

else

If (!arbol.getHijoIzquierdo().esVacio() && k > 0){

ok= esMonodistante(arbol.getHijoIzquierdo(), k - arbol.getDatoRaiz())

}

If (ok && !arbol.getHijoDerecho().esVacio() && k > 0){

ok= esMonodistante(arbol.getHijoDerecho(), k - arbol.getDatoRaiz());

}

}

return ok;

}