Boolean esArbolCreciente(){

ColaGenerica<ArbolBinario<T>>cola= new ColaGenerica<ArbolBinario<T>>();

int nivelActua l = 0;

int cantNivel = 0;

Boolean esCreciente =true; POR NIVELES

cola.encolar(this);

cola.encolar(null);

while( ! cola.esVacia ( ) && esCreciente) {

ArbolBinario<T> e = cola.desencolar();

if (e == null){

if (cantNivel != nivelActual+1){

esCreciente= false; // el árbol NO es creciente

}else{

nivelActual++;

if( ! cola.esVacia ( ) )

cola.encolar ( null) ;

cantNivel = 0; POR NIVELES

}else{

cantNivel++;

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

cola.encolar (this.getHijoIzquierdo());

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

cola.encolar (this.getHijoDerecho());

}

return esCreciente;

}

public ArbolBinario <Integer> minEnNiveldeAB (int n) {

ColaGenerica<ArbolGeneral<T>> cola = new ColaGenerica<ArbolGeneral<T>>();

cola.encolar(this);

cola.encolar(null);

int nivel = 1;

ArbolGeneral<T> menor = null; POR NIVELES

while (!cola.esVacia()) {

arbol = cola.desencolar();

If (arbol == null) {

Nivel = nivel +1;

If (!cola.esVacia()) {

cola.encolar(null);

}

} else {

If (nivel == n) {

If (arbol.esHoja()) {

If (menor == null || menor.getDatoRaiz() < arbol.getDatoRaiz()) {

menor = arbol;

}

} POR NIVELES

} else { /\* arbol no vacio y nivel menor a n \*/

If (!arbol.getHijoIzquierdo().esVacio()) {

cola.encolar(arbol.getHijoIzquierdo());

}

If (!arbol.getHijoDerecho().esVacio()) {

cola.encolar(arbol.getHijoDerecho());

}

}

}

return menor;

}