Clase Arbol

public class ArbolBinario {

private Nodo raiz;

int cant;

int altura;

public ArbolBinario() {

this.raiz = null;

}

public void agregar(int dato) {

Nodo nuevo = new Nodo(dato, null, null);

insertar(nuevo, raiz);

}

public void insertar(Nodo nuevo, Nodo pivote) {

if (this.raiz == null) {

raiz = nuevo;

} else {

if (nuevo.getDato() <= pivote.getDato()) {

if (pivote.getIzq() == null) {

pivote.setIzq(nuevo);

} else {

insertar(nuevo, pivote.getIzq());

}

} else {

if (pivote.getDer() == null) {

pivote.setDer(nuevo);

} else {

insertar(nuevo, pivote.getDer());

}

}

}

}

//metodos nuevos

public boolean existe(int info) {

Nodo reco = raiz;

while (reco != null) {

if (info == reco.getDato()) {

return true;

} else if (info > reco.getDato()) {

reco = reco.getDer();

} else {

reco = reco.getIzq();

}

}

return false;

}

public int cantidad() {

cant = 0;

cantidad(raiz);

return cant;

}

private void cantidad(Nodo reco) {

if (reco != null) {

cant++;

cantidad(reco.getIzq());

cantidad(reco.getDer());

}

}

private void cantidadNodosHoja(Nodo reco) {

if (reco != null) {

if (reco.getIzq() == null && reco.getDer() == null) {

cant++;

}

cantidadNodosHoja(reco.getIzq());

cantidadNodosHoja(reco.getDer());

}

}

public int cantidadNodosHoja() {

cant = 0;

cantidadNodosHoja(raiz);

return cant;

}

public int retornarAltura() {

altura = 0;

retornarAltura(raiz, 1);

return altura;

}

private void retornarAltura(Nodo reco, int nivel) {

if (reco != null) {

retornarAltura(reco.getIzq(), nivel + 1);

if (nivel > altura) {

altura = nivel;

}

retornarAltura(reco.getDer(), nivel + 1);

}

}

public void menorValor() {

if (raiz != null) {

Nodo reco = raiz;

while (reco.getIzq() != null) {

reco = reco.getIzq();

}

System.out.println("Menor valor del árbol:" + reco.getDato());

}

}

public void mayorValor() {

if (raiz != null) {

Nodo reco = raiz;

while (reco.getDer() != null) {

reco = reco.getDer();

}

System.out.println("Mayor valor del árbol:" + reco.getDato());

}

}

int subizq = 0;

int subder = 0;

public void imprimirBalance() {

subizq = 0;

subder = 0;

Balance(this.raiz, true, 0);

System.out.println("lado Izquierdo " + subizq + " Lado Derecho " + subder);

if (subizq - subder == 0) {

System.out.println("El balance es: 0 ");

} else if (subizq - subder == -1) {

System.out.println("El balance es -1, derecha");

} else if (subizq - subder == 1) {

System.out.println("El balance 1, izquierda");

} else {

System.out.println("No es balanceado.."

+ "porque es mas grande el lado "

+ ((subizq > subder) ? "Izquierdo" : "Derecho"));

}

}

public void Balance(Nodo reco, boolean lado, int i) {

if (reco != null) {

if (reco.getDer() == null && reco.getIzq() == null) {

if (lado) {

subder = (i > subder) ? i : subder;

} else {

subizq = (i > subizq) ? i : subizq;

}

}

Balance(reco.getDer(), lado, i + 1);

if (i == 0) {

lado = false;

}

Balance(reco.getIzq(), lado, i + 1);

}

}

//altura arbol

String[] niveles;

public int alturaArbol() {

altura = 0;

alturaArbol(raiz, 0);

return altura;

}

private void alturaArbol(Nodo pivote, int nivel) {

if (pivote != null) {

alturaArbol(pivote.getIzq(), nivel + 1);

if (nivel > altura) {

altura = nivel;

}

alturaArbol(pivote.getDer(), nivel + 1);

}

}

public void imprimirNivel() {

niveles = new String[altura + 1];

imprimirNivel(raiz, 0);

for (int i = 0; i < niveles.length; i++) {

System.out.println(niveles[i] + " En nivel: " + i);

}

}

public void imprimirNivel(Nodo pivote, int nivel2) {

if (pivote != null) {

niveles[nivel2] = pivote.getDato() + ", " + ((niveles[nivel2] != null) ? niveles[nivel2] : "");

imprimirNivel(pivote.getDer(), nivel2 + 1);

imprimirNivel(pivote.getIzq(), nivel2 + 1);

}

}

public void imprimirAlturaDeCadaNodo() {

imprimirAlturaDeCadaNodo(raiz, 1);

}

private void imprimirAlturaDeCadaNodo(Nodo reco, int nivel) {

if (reco != null) {

System.out.println("Nodo contiene: "+reco.getDato() + " y su altura es: "+nivel);

imprimirAlturaDeCadaNodo(reco.getIzq(), nivel + 1);

imprimirAlturaDeCadaNodo(reco.getDer(), nivel + 1);

}

}

}

Clase Nodo

public class Nodo {

private int dato;

private Nodo izq, der;

public Nodo(int dato, Nodo izq, Nodo der) {

this.dato = dato;

this.izq = izq;

this.der = der;

}

public int getDato() {

return dato;

}

public void setDato(int dato) {

this.dato = dato;

}

public Nodo getIzq() {

return izq;

}

public void setIzq(Nodo izq) {

this.izq = izq;

}

public Nodo getDer() {

return der;

}

public void setDer(Nodo der) {

this.der = der;

}

}

Clase Main

import java.util.Scanner;

public class Main {

public static void main(String[] arg) {

ArbolBinario abo = new ArbolBinario();

int var;

int valor;

Scanner texto = new Scanner(System.in);

/\*

System.out.println(" Insertando los siguientes valores Randomicamente : ");

for (int i = 1; i <= 10; i++) {

valor = (int) (Math.random() \* 100);

System.out.print(valor + " ");

abo.agregar(valor);

}

\*/

System.out.println(" Insertando valores manualmente en el árbol: ");

abo.agregar(60); abo.agregar(75); abo.agregar(66); abo.agregar(85); abo.agregar(42);

abo.agregar(14); abo.agregar(73); abo.agregar(72); abo.agregar(2); abo.agregar(54);

System.out.println("Valores insertados: 60 75 66 85 42 14 73 72 2 54");

abo.imprimirAlturaDeCadaNodo();

do {

System.out.println(" ");

System.out.println(" ");

System.out.println("1.- Cantidad de nodos del arbol");

System.out.println("2.- cantidad de nodos hoja");

System.out.println("3.- Altura del arbol");

System.out.println("4.- BALANCE");

System.out.println("0.- Salir");

var = texto.nextInt();

switch (var) {

case 1:

System.out.println("Cantidad de nodos del árbol:" + abo.cantidad());

break;

case 2:

System.out.println("Cantidad de nodos hoja:" + abo.cantidadNodosHoja());

break;

case 3:

System.out.println("La altura del arbol es:" + abo.retornarAltura());

break;

case 4:

abo.imprimirBalance();

break;

}

} while (var != 0);

}

}