UNIVERSIDAD AUTÓNOMA GABRIEL RENE MORENO
FACULTAD DE INGENIERÍA Y CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES

ESTRUCTURAS DE DATOS 2

CONTENIDO: *// Breve Descripción del Contenido*

LAB-3. ABB CON LIBRERÍA DE LISTAS

PORCENTAJE TERMINADO : 100%.

GRUPO: 14

Nombre	Registro
Cristian Gabriel Gedalge Cayhuara	219022062

Fecha de Presentación: Jueves ,09 de mayo de 2024

COMENTARIO:
Los ultimos ejercicios me fueron complicados de resolverlos, pero se pudo ,me gusto que implementemos estructuras de datos que ya vienen con el lenguaje java

```
CLASE ARBOL
public class Arbol {

    public Nodo raiz;

    public Arbol() {
        this.raiz = null;
    }

// 1.   A1.generarElem(n, a, b) : Método que genera n
elementos aleatorios enteros diferentes entre a y b
inclusive.
    public void generarElem(int n, int a, int b) {
        for (int i = n; i > 0; i--) {
            int x = (int) (Math.random() * (b - a)) + a;
            insertar(x);
        }
    }
//2.       A1.insertar(x) : Método que inserta el elemento x,
en el árbol A1 en su lugar correspondiente.

    public void insertar(int ele) {
        this.raiz = insertar(this.raiz, ele);
    }

    private Nodo insertar(Nodo p, int ele) {
        if (p == null) {
            p = new Nodo(ele);
            return p;
        } else {
            if (ele < p.elem) {
                p.izq = insertar(p.izq, ele);
            } else {
                p.der = insertar(p.der, ele);
            }
        }
        return p;
    }
//3.       A1.preOrden() : Método que muestra los elementos
del árbol A1 en preOrden.

    public void preOrden() {
        preOrden(raiz);
    }

    public void preOrden(Nodo p) {
        if (p == null) {
            return;
```

```
        }
        System.out.println(p.elem);
        preOrden(p.izq);
        preOrden(p.der);
    }
//4.       A1.inOrden() : Método que muestra los elementos
del árbol A1 en inOrden.

    public void inOrden() {
        inOrden(raiz);
    }

    private void inOrden(Nodo p) {
        if (p == null) {
            return;
        }
        inOrden(p.izq);
        System.out.println(p.elem);
        inOrden(p.der);
    }
//5.       A1.postOrden() : Método que muestra los elementos
del árbol A1 en postOrden.

    public void postOrden() {
        System.out.print("[");
        postOrden(raiz);
        System.out.print("]");
    }

    public void postOrden(Nodo p) {
        if (p == null) {
            return;
        }
        postOrden(p.izq);
        postOrden(p.der);
        System.out.print(p.elem + ",");
    }
//6.       A1.niveles(): Método que muestra los elementos del
árbol A1, por niveles.
//
//7.       A1.desc(): Método que muestra los elementos del
árbol A1 de mayor a menor.

    public void desc() {
        desc(this.raiz);
    }

    private void desc(Nodo p) {
```

```
        if (p == null) {
            return;
        }
        desc(p.der);
        System.out.print(p.elem + " ,");
        desc(p.izq);

    }
//8.       A1.seEncuentra(x) : Métodos lógico que devuelve
True, si el elemento x, se encuentra en el árbol A1.

    public boolean seEncuentra(int x) {
        return seEncuentra(x, raiz);
    }

    private boolean seEncuentra(int x, Nodo p) {
        if (p == null) {
            return false;
        }
        if (x == p.elem) {
            return true;
        }
        if (x < p.elem) {
            return seEncuentra(x, p.izq);
        } else {
            return seEncuentra(x, p.der);
        }

    }

//9.       A1.cantidad() : Método que devuelve la cantidad de
nodos del árbol A1.
    public int cantidad() {
        return cantidad(raiz);
    }

    private int cantidad(Nodo p) {
        int cant;
        if (p == null) {
            return 0;
        } else {
            cant = 1 + cantidad(p.izq) + cantidad(p.der);
        }
        return cant;
    }
//10.   A1.suma() : Método que devuelve la suma de los
elementos del árbol A1.
```

```
    public int suma() {
        return suma(raiz);
    }

    private int suma(Nodo p) {
        int sum;
        if (p == null) {
            return 0;
        } else {
            sum = p.elem + (suma(p.izq) + suma(p.der));
        }
        return sum;
    }
//11.   A1.menor() : Método que devuelve el elemento menor
del árbol A1.

    public int menor() {
        return menor(raiz);
    }

    private int menor(Nodo p) {
        if (p.izq == null) {
            return p.elem;
        } else {
            return menor(p.izq);
        }
    }

//12.   A1.mayor() : Método que devuelve el elemento mayor
del árbol A1.
    public int mayor() {
        return mayor(raiz);
    }

    private int mayor(Nodo p) {
        if (p.der == null) {
            return p.elem;
        } else {
            return menor(p.der);
        }
    }
//13.   A1.preoOrden(L1) : Método que encuentra en la lista
L1, el recorrido de preOrden de los elementos del árbol A1.

    public void preOrden(ArrayList<Integer> L1)
    {
        preOrden(this.raiz,L1);
    }
```

```
    private void preOrden(Nodo p,ArrayList<Integer> L1){
        if(p==null)return;
        L1.add(p.elem);
        preOrden(p.izq,L1);
        preOrden(p.der,L1);

    }
//14.   A1.inOrden(L1) : Método que encuentra en la lista L1,
el recorrido de inOrden de los elementos del árbol A1.
    public void inOrden(ArrayList<Integer> L1)
    {
        inOrden(this.raiz,L1);
    }
    private void inOrden(Nodo p,ArrayList<Integer> L1){
        if(p==null)return;

        inOrden(p.izq,L1);
        L1.add(p.elem);
        inOrden(p.der,L1);

    }
//15.   A1.postOrden(L1) : Método que encuentra en la lista
L1, el recorrido de postOrden de los elementos del árbol A1.
        public void postOrden(ArrayList<Integer> L1) {
        postOrden(raiz,L1);
    }

    public void postOrden(Nodo p,ArrayList<Integer> L1) {
        if (p == null) {
            return;
        }
        postOrden(p.izq,L1);
        postOrden(p.der,L1);
        L1.add(p.elem);
    }
//16.   A1.niveles(L1) : Método que encuentra en la lista L1,
el recorrido por niveles de los elementos del árbol A1.
    public void niveles(ArrayList<Integer> L1,int nivel)
    {
        niveles(this.raiz,L1,nivel);
    }

    private void niveles(Nodo p,ArrayList<Integer> L1,int
nivel)
    {
        if (p == null) {
            return;
        }
```

```
        elementoNivel(p.izq, nivel + 1);
        System.out.println(p.elem + "t" + nivel);
        L1.add(nivel);
        elementoNivel(p.izq, nivel + 1);
    }
//17.   A1.mostrarNivel(): Método que muestra los elementos
del árbol y el nivel en el que se encuentran. (Recorrer el
árbol en cualquier orden)

    public void mostrarnivel() {
        LinkedList<Nodo> L1 = new LinkedList();
        if (raiz == null) {
            return;
        }
        L1.add(raiz);
        while (!L1.isEmpty()) {
            Nodo p = L1.getFirst();
            System.out.print(p.elem);
            if (p.izq != null) {
                L1.add(p.izq);
            }
            if (p.der != null) {
                L1.add(p.der);
            }
            L1.removeFirst();
        }
    }
//18. A1.sumarNivel(L1) : Método que encuentra en la Lista de
acumuladores por nivel L1, la suma de los elementos de cada
nivel.

    public void sumaNivel() {
        int max = cantidad();
        ArrayList<Integer> L1 = new ArrayList(max);
        for (int i = 0; i < max; i++) {
            L1.add(0);
        }
        sumaNivel(this.raiz, 0, L1);
        int i = 0;
        while (L1.get(i) != 0) {
            System.out.println(i + 1 + "\t" + L1.get(i));
            i++;
        }

    }

    public void sumaNivel(Nodo p, int nivel,
ArrayList<Integer> L1) {
```

```
        if (p == null) {
            return;
        }
        L1.set(nivel, L1.get(nivel) + p.elem);
        sumaNivel(p.izq, nivel + 1, L1);
        sumaNivel(p.der, nivel + 1, L1);

    }

    public void elementoNivel(int nivel) {
        elementoNivel(this.raiz, nivel);
    }

    private void elementoNivel(Nodo p, int nivel) {
        if (p == null) {
            return;
        }
        elementoNivel(p.izq, nivel + 1);
        System.out.println(p.elem + "t" + nivel);
        elementoNivel(p.izq, nivel + 1);
    }

    public static void main(String[] arg) {
        Arbol A1 = new Arbol();
        A1.insertar(14);
        A1.insertar(15);
        A1.insertar(13);
        A1.insertar(4);
        A1.insertar(5);
        A1.mostrarnivel();
    }

}
```

CLASE NODO

```
public class Nodo {
    Nodo izq;
    Nodo der;
    int elem;
    public Nodo(int ele)
    {
        this.elem=ele;
        this.izq=this.der=null;
    }
}
```