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

ESTRUCTURA DE DATOS 2

CONTENIDO: Tarea de Arboles con Librería de Listas

PORCENTAJE TERMINADO : 100%.

GRUPO: 15

Integrantes	DT	HG	HI	EVAL
Ibarra Cuellar Gustavo	1	1	1	100

Fecha de Presentación: Jueves 24 de octubre 2024
Fecha Presentada: Jueves 24 de Octubre 2024

CLASES

```
public class Arbol {

public Nodo raiz;

public Arbol(){

   raiz=null;

}
```

A1.generarElem(n,a,b)

```
public void generarElem(int n,int a,int b){

   for(int i=0; i<n; i++){
```

```
    int                                    num=
ThreadLocalRandom.current().nextInt(a,b
+1);
     this.insertar(num);
    }
}
```

A1.insertar(x)

```
public void insertar(int x){
 raiz=insertar(x,raiz);
}

private Nodo insertar(int x,Nodo p){
 if(p==null)return new Nodo(x);
 if(x<p.elem)
   p.izq=insertar(x,p.izq);
 else  p.der=insertar(x,p.der);
 return p;
}
```

A1.preOrden()

```
public void preOrden(){
    preOrden(raiz);
}
private void preOrden(Nodo p){
   if(p==null)return;
    System.out.println(p.elem);
   preOrden(p.izq);
```

```
  preOrden(p.der);
}
```

A1.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);
}public void postOrden() {
   postOrden(raiz);
}
```

A1.posOrden()

```
private void postOrden(Nodo p) {
   if (p == null) return;
   postOrden(p.izq);
   postOrden(p.der);
   System.out.println(p.elem);
}
```

A1.desc()

```
public void desc(){
   desc(raiz);
}
```

```
private void desc(Nodo p){
 if(p==null)return;
   desc(p.der);
    System.out.println(p.elem);
    desc(p.izq);
}
```

A1.seEncuentra(x)

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

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

A1.cantidad()

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

private int cantidad(Nodo p){
if(p==null) return 0;
```

```
else
    return
cantidad(p.izq)+cantidad(p.der)+1;
}
```

A1.suma()

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

private int suma(Nodo p){
if(p==null)return 0;
return p.elem+suma(p.izq)+suma(p.der);
}
```

A1.mayor()

```
 public int mayor(){
    if(raiz==null)      {      throw      new
IllegalStateException("Arbol Vacio"); }
    return mayor(raiz);
 }

 private int mayor(Nodo p){
 while(p.der != null){
    p=p.der;
 }
 return p.elem;
 }
```

A1.menor()

```
public int menor(){
   if(raiz==null) { throw new IllegalStateException("Arbol Vacio"); }
   return menor(raiz);
}

private int menor(Nodo p){
while(p.izq != null){
   p=p.izq;
}
return p.elem;
}
```

A1.preOrden(L1)

```
public void preOrden(List<Integer> L1) {
   preOrden(raiz, L1);
    System.out.println(L1);
}

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

```
 public void inOrden(List<Integer> L1) {
   inOrden(raiz, L1);
    System.out.println(L1);
}
```

A1.inOrden(L1)

```
private void inOrden(Nodo p,
List<Integer> L1) {
   if (p != null) {
      inOrden(p.izq, L1);
      L1.add(p.elem);
      inOrden(p.der, L1);
   }
}
```

A1.posOrden(L1)

```
 public void postOrden(List<Integer> L1) {
   postOrden(raiz, L1);
    System.out.println(L1);
}

private void postOrden(Nodo p,
List<Integer> L1) {
   if (p != null) {
      postOrden(p.izq, L1);
      postOrden(p.der, L1);
      L1.add(p.elem);
```

```
    }
}
```

A1.niveles(L1)

```
public void niveles(List<Integer> L1) {
    if (raiz == null) {
        return;
    }
    Queue<Nodo> queue = new
LinkedList<>();
    queue.add(raiz);
    while (!queue.isEmpty()) {
        Nodo current = queue.poll();
        L1.add(current.elem);
        if (current.izq != null) {
            queue.add(current.izq);
        }
        if (current.der != null) {
            queue.add(current.der);
        }
    }
}
```

A1.mostrarConnivel()

```
public void mostrarConNivel() {
    mostrarConNivel(raiz, 0);
}
```

```
private void mostrarConNivel(Nodo p, int
nivel) {
   if (p != null) {
      mostrarConNivel(p.izq, nivel + 1);
      System.out.println("Elemento:    "    +
p.elem + ", Nivel: " + nivel);
      mostrarConNivel(p.der, nivel + 1);
   }
}
```

A1.elemNivel()

```
public void elemNivel(){
   elemNivel(raiz,0);
}
private void elemNivel(Nodo p,int nivel){
  if(p==null)return;
elemNivel(p.izq,nivel+1);
   System.out.println(p.elem+"\t"+ nivel);
   elemNivel(p.der,nivel+1);
}
```

A1.mostrarNiveles()

```
public void mostrarNiveles(){
 LinkedList <Nodo> L1 = new LinkedList();
 if(raiz != null)L1.add(raiz);
 while(!L1.isEmpty()){
   Nodo p=L1.removeFirst();
   if(p.izq != null) L1.add(p.izq);
```

```
  if(p.der != null) L1.add(p.der);
    System.out.println(p.elem);
 }
}
```