UNIVERSIDAD TECNOLÓGICA DE PANAMÁ FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS

LICENCIATURA EN DESARROLLO Y GESTIÓN DE SOFTWARE

LAB 1 – CREACIÓN DE UN ÁRBOL BINARIO

Prof. Yolanda de Miguelena

Integrantes:

Bustamante, David
García, Eliel
Grupo: 1GS121

15 de abril de 2024

```java
import java.util.Scanner;

public class MainBusq {
    public static void main(String[] args) {
        ArbolBinario arbol2 = new ArbolBinario();
        Scanner scanner = new Scanner(System.in);
        String res;

        // Insertar elementos
        System.out.print("Ingrese el valor para la raíz del árbol:
");
        String valorRaiz = scanner.nextLine();
        arbol2.inicarBusqueda(valorRaiz); //Inicializar el arbol

        while (true) {
            System.out.print("¿Desea insertar otro valor? (s/n):
");
            String continuar = scanner.nextLine();
            if (continuar.equals("s")) {
                System.out.print("Ingrese el valor a insertar: ");
                String valor = scanner.nextLine();
                arbol2.inicarBusqueda(valor);
            } else {
                break;
            }
        }

        System.out.println("Recorrido Inorden del árbol:");
        arbol2.ejecutarInorden();
        System.out.println("\n");

        System.out.println("Recorrido Postorden del árbol:");
        arbol2.ejecutarPostorden();
        System.out.println("\n");

        System.out.println("Recorrido Preorden del árbol:");
        arbol2.ejecutarPreorden();
        System.out.println("\n");
    }
}
```

```java
public class Nodo {
    String dato;
    Nodo izquierda, derecha;

    //Constructir
    public Nodo(String dato){
        this.dato = dato;
        this.izquierda = null;
        this.derecha = null;
    }

    /*
    * Getters y setter
    * */
    public String getDato() {
        return dato;
    }
    public void setDato(String valor) {
        this.dato = valor;
    }
    public Nodo getIzquierda() {
        return izquierda;
    }
    public Nodo getDerecha() {
        return derecha;
    }
}
```

```java
import java.util.Scanner;

public class ArbolBinario {
    Nodo raiz;

    // Constructor
    public ArbolBinario() {
        raiz = null;
    }

    //Iniciar arbol de busqueda
    public void inicarBusqueda(String dato){
        raiz = insertarB(raiz, dato);
    }

    //Insertar con busqueda
```

```java
    public Nodo insertarB(Nodo padre, String dato){
        if(padre == null){
            padre = new Nodo(dato);
            return padre;
        }

        if(padre.getDato().compareTo(dato) < 0){
            padre.izquierda = insertarB(padre.izquierda, dato);
        }else{
            padre.derecha = insertarB(padre.derecha, dato);
        }

        System.out.println("\nHola");
        return padre;
    }

    //VERIFIICAR JERARQUIA
    public boolean verificarAlfabetoMenor(String pal1, String
pal2){

        for(int i = 1; i < pal1.length(); i++){
            if(pal1.charAt(i) < pal2.charAt(i)){//La palabra 1 es
menor que la 2
                return true; //Si es menor
            } else if (pal1.charAt(i) > pal2.charAt(i)) {
                return false;
            }
        }

        return false;
    }

   //Metodo para insertar nodos
    public void iniciarArbol(String dato){
        raiz = insertarH(raiz, dato);
    }

    //Metodo para insertar de manera recursiva
    public Nodo insertarH(Nodo padre, String dato){
        if(padre == null){
            padre = new Nodo(dato);
            return padre;
        }

        Scanner sc = new Scanner(System.in);
        if(padre.izquierda == null){
```

```
            System.out.println("Deseas insertar como hijo
izquierdo para el nodo con dato " + padre.getDato() + "\nDejar en
blanco si no");
            String newDato = sc.nextLine();//Nuevo dato

            if(!newDato.isEmpty()){
                //Crear nodo con el nuevo dato
                Nodo nodoTemp = new Nodo(newDato);
                padre.izquierda = nodoTemp;
                insertarH(padre.izquierda, newDato);
            }
        }

        if(padre.derecha == null){

            System.out.println("Deseas insertar como hijo derecho
para el nodo con dato " + padre.getDato() + "\nDejar en blanco si
no");
            String newDato = sc.nextLine();//Nuevo dato

            if(!newDato.isEmpty()){
                //Crear nodo con el nuevo dato
                Nodo nodoTemp = new Nodo(newDato);
                padre.derecha = nodoTemp;
                insertarH(padre.derecha, newDato);
            }
        }

        return padre;
    }

    // Método para recorrer en preorden
    public void ejecutarPreorden() {
        preorden(raiz);
    }

    private void preorden(Nodo padre) {
        if (padre != null) {
            System.out.print(padre.dato + "-");
            preorden(padre.izquierda);
            preorden(padre.derecha);
        }
    }

    // Método para recorrer en inorden
```

```java
    public void ejecutarInorden() {
        inorden(raiz);
    }

    private void inorden(Nodo padre) {
        if (padre != null) {
            inorden(padre.izquierda);
            System.out.print(padre.dato + "-");
            inorden(padre.derecha);
        }
    }

    // Método para recorrer en postorden
    public void ejecutarPostorden() {
        postorden(raiz);
    }

    private void postorden(Nodo padre) {
        if (padre != null) {
            postorden(padre.izquierda);
            postorden(padre.derecha);
            System.out.print(padre.dato + "-");
        }
    }

    public Nodo getRaiz() {
        return raiz;
    }

    public void setRaiz(Nodo raiz) {
        this.raiz = raiz;
    }
}
```

```
Recorrido Inorden del árbol:
DIRIA-JUNTO-LO-MEMORIZADO-NO-OTRA-PERSONA-TENDRIA-TODO-

Recorrido Postorden del árbol:
MEMORIZADO-LO-JUNTO-DIRIA-NO-TENDRIA-TODO-PERSONA-OTRA-

Recorrido Preorden del árbol:
OTRA-NO-DIRIA-JUNTO-LO-MEMORIZADO-PERSONA-TODO-TENDRIA-
```