

## Evaluación2-U2\_ÁRBOLES

**Nombre:** Juan David Jiménez Romero

**Clase main:**

```
package arbolbinarioprueba_juanj;

import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertFalse;
import static org.junit.jupiter.api.Assertions.assertTrue;

public class ArbolBinarioPrueba_JuanJ {

    public static void main(String[] args) {
        // TODO code application logic here
        Scanner scan = new Scanner(System.in);
        int opc;
        System.out.println("====MENU ARBOL BINARIO====");
        System.out.println("1.Recorrido arbol binario");
        System.out.println("2.invertir arbol binario");
        System.out.println("3.Lista del ultimo nivel");
        System.out.println("4.Profundidad maxima/Nivel");
        System.out.println("5.Validar BTS");
```

```
System.out.println("6.SubArbol");
System.out.println("7.Primer Ancestro comun");
System.out.println("0.Salir");
opc = scan.nextInt();
while(opc!=0){
    switch(opc){
        case 1:
            //Actividad 0
            System.out.println("ACTIVIDAD 0:");
            Node root = new Node(1);
            root.left = new Node(2);
            root.right = new Node(3);
            root.left.left = new Node(4);
            root.left.right = new Node(5);
            root.right.right = new Node(6);
            root.left.left.left = new Node(7);
            root.left.right.left = new Node(8);

            System.out.print("IN-ORDER TRAVERSAL: ");
            BinaryTreeTraversals.inOrdenTraversal(root);
            System.out.println();

            System.out.print("PRE-ORDER TRAVERSAL: ");
            BinaryTreeTraversals.preOrdenTraversal(root);
```

```
System.out.println();

System.out.print("POST-ORDER TRAVERSAL: ");
BinaryTreeTraversals.postOrdenTraversal(root);
System.out.println();

break;

case 2:

    //Actividad 1

    System.out.println("ACTIVIDAD 1:");

    InvertirArbolBinario invertirArbolBinario = new
InvertirArbolBinario();

    Node root1 = new Node(4);
    root1.left = new Node(2);
    root1.right = new Node(7);
    root1.left.left = new Node(1);
    root1.left.right = new Node(3);
    root1.right.left = new Node(6);
    root1.right.right = new Node(9);

    Node newRoot = invertirArbolBinario.
invertirAbol(root1);

    assertEquals(4, newRoot.value);
    assertEquals(7, newRoot.left.value);
    assertEquals(2, newRoot.right.value);
    assertEquals(9, newRoot.left.left.value);
```

```

        assertEquals(6, newRoot.left.right.value);
        assertEquals(3, newRoot.right.left.value);
        assertEquals(1, newRoot.right.right.value);
        System.out.print("PRE-ORDER TRAVERSAL: ");
        BinaryTreeTraversals.preOrdenTraversal(newRoot);
        System.out.println();
    break;
    case 3:
        //Actividad 2
        System.out.println("ACTIVIDAD 2:");
        ListOfDepths listOfDepths = new ListOfDepths();
        Node root2 = new Node(4);
        root2.left = new Node(2);
        root2.right = new Node(7);
        root2.left.left = new Node(1);
        root2.left.right = new Node(3);
        root2.right.left = new Node(6);
        root2.right.right = new Node(9);
        List<LinkedList<Node>> result =
listOfDepths.listOfDepths(root2);

        assertEquals(4, result.get(0).get(0).value);
        assertEquals(2, result.get(1).get(0).value);
        assertEquals(7, result.get(1).get(1).value);
        assertEquals(1, result.get(2).get(0).value);

```

```

    assertEquals(3, result.get(2).get(1).value);
    assertEquals(6, result.get(2).get(2).value);
    assertEquals(9, result.get(2).get(3).value);
    System.out.print("PRE-ORDER TRAVERSAL: ");
    BinaryTreeTraversals.preOrdenTraversal(root2);
    System.out.println("\nResultado: ");
    for(int i=0;i<result.size();i++){
        System.out.print(result.get(i)+" ");
    }
    System.out.println();
break;
case 4:
    //Actividad 3
    System.out.println("ACTIVIDAD 3:");
    MaximumDepth maxDepth = new MaximumDepth();
    Node root3 = new Node(4);
    root3.left = new Node(2);
    root3.right = new Node(7);
    root3.left.left = new Node(1);
    root3.left.right = new Node(3);
    assertEquals(3, maxDepth.maxDepth(root3));
    root3.left.left.left = new Node(8);
    assertEquals(4, maxDepth.maxDepth(root3));
break;

```

case 5:

```
//Actividad 4  
System.out.println("ACTIVIDAD 4:");  
ValidateBST validateBst = new ValidateBST();  
Node root4 = new Node(4);  
root4.left = new Node(5);  
root4.right = new Node(7);  
root4.left.left = new Node(1);  
root4.left.right = new Node(3);  
root4.left.left.left = new Node(8);  
assertFalse(validateBst.isValidBST(root4));  
root4.left.value = 2;  
root4.left.left.left = null;  
assertTrue(validateBst.isValidBST(root4));
```

break;

case 6:

```
//Actividad 5  
System.out.println("ACTIVIDAD 5:");  
IsSubTree isSubTree = new IsSubTree();  
Node first = new Node(4);  
first.left = new Node(5);  
first.right = new Node(7);  
first.left.left = new Node(1);  
first.left.right = new Node(3);
```

```

        first.left.left.left = new Node(8);
        Node second = new Node(5);
        second.left = new Node(1);
        second.right = new Node(3);
        second.left.left = new Node(8);
        assertTrue(isSubTree.isSubtree(first, second));
        second.right.right = new Node(12);
        assertFalse(isSubTree.isSubtree(first, second));
        break;
    case 7:
        //Actividad 6
        System.out.println("ACTIVIDAD 6:");
        break;
    default:
        System.out.println("Opcion no valida");
        break;
}

System.out.println("====MENU ARBOL BINARIO====");
System.out.println("1.Recorrido arbol binario");
System.out.println("2.invertir arbol binario");
System.out.println("3.Lista del ultimo nivel");
System.out.println("4.Profundidad maxima/Nivel");
System.out.println("5.Validar BTS");
System.out.println("6.SubArbol");

```

```

        System.out.println("7.Primer Ancestro comun");
        System.out.println("0.Salir");
        opc = scan.nextInt();
    }
    System.out.println("Gracias por ingresar");
}

}

```

## Clase Node:

```

package arbolbinarioprueba_juanj;

public class Node { //declaracion de la clase Node
    //declaracion de las variables
    public int value;
    public Node left;
    public Node right;
    //constructor de la clase Node
    public Node (int value){ //formato del constructor
        this.value = value;
        this.left = null;
        this.right = null;
    }
}

```



## Clase BinaryTreeTraversals:

```
package arbolbinarioprueba_juanj;
```

```
public class BinaryTreeTraversals { //clase para el ordenamiento  
del arbol
```

```
    public static void inOrdenTraversal(Node node){ //metodo para  
el recorrido del arbol de la forma In-Orden
```

```
        if(node != null){  
            inOrdenTraversal(node.left);  
            System.out.print(node.value+" ");  
            inOrdenTraversal(node.right);  
        }  
    }
```

```
    public static void preOrdenTraversal(Node node){
```

```
        if(node != null){  
            System.out.print(node.value+" ");  
            preOrdenTraversal(node.left);  
            preOrdenTraversal(node.right);  
        }  
    }
```

```
    public static void postOrdenTraversal (Node node){
```

```
        if(node != null){  
            postOrdenTraversal(node.left);  
            postOrdenTraversal(node.right);
```

```

        System.out.print(node.value+" ");
    }
}

```

### Codigo InvertirArbolBinario:

```

package arbolbinarioprueba_juanj;

public class InvertirArbolBinario{
    public Node invertirAbol(Node root) {
        if (root == null) return null;
        Node tmp = root.left;
        root.left = invertirAbol(root.right);
        root.right = invertirAbol(tmp);
        return root;
    }
}

```

### Codigo ListOfDethps:

```

package arbolbinarioprueba_juanj;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class ListOfDepths {
    public List<LinkedList<Node>> listOfDepths(Node root) {

```

```

    if (root == null) {
        return null;
    }

    List<LinkedList<Node>> result = new ArrayList<>();
    LinkedList<Node> current = new LinkedList<>();
    current.add(root);
    while (!current.isEmpty()) {
        result.add(current);
        LinkedList<Node> parents = current;
        current = new LinkedList<Node>();
        for (Node node : parents) {
            if (node.left != null) {
                current.add(node.left);
            }
            if (node.right != null) {
                current.add(node.right);
            }
        }
    }
    return result;
}

```

### Codigo MaximumDepth:

```

package arbolbinarioprueba_juanj;

```

```

public class MaximumDepth {
    public int maxDepth(Node root) {
        if (root == null) return 0;
        int depthLeft = maxDepth(root.left) + 1;
        int depthRight = maxDepth(root.right) + 1;
        return Math.max(depthLeft, depthRight);
    }
}

```

### Codigo ValidateBST:

```

public class ValidateBST {
    public boolean isValidBST(Node root) {
        return isValidBST(root, null, null);
    }
    public boolean isValidBST(Node root, Integer min, Integer max)
    {
        if (root == null) return true;
        if ((min != null && root.value <= min) || (max != null &&
root.value >= max)) return false;
        return isValidBST(root.left, min, root.value) &&
isValidBST(root.right, root.value, max);
    }
}

```

### Codigo IsSubTree:

```

public class IsSubTree {

```

```

public boolean isSubtree(Node first, Node second) {
    StringBuilder stb1 = new StringBuilder();
    StringBuilder stb2 = new StringBuilder();
    preOrder(first, stb1);
    preOrder(second, stb2);
    return stb1.toString().contains(stb2.toString());
}

private void preOrder(Node node, StringBuilder stb) {
    if (node == null) {
        stb.append("X");
        return;
    }
    stb.append(node.value);
    preOrder(node.left, stb);
    preOrder(node.right, stb);
}
}

```

### Codigo FirstCommonAncestor:

```

package arbolbinarioprueba_juanj;

public class FirstCommonAncestor {
    class AncestorNode {
        boolean nodeFound;
        Node ancestor;
    }
}

```

```

    }

    public Node firstCommonAncestor(Node root, Node firstNode,
Node secondNode) {

        return postOrderSearch(root, firstNode,
secondNode).ancestor;

    }

    private AncestorNode postOrderSearch(Node current, Node
firstNode, Node secondNode) {

        if (current == null) {

            return new AncestorNode();

        }

        AncestorNode leftResult = postOrderSearch(current.left,
firstNode, secondNode);

        if (leftResult.ancestor != null) return leftResult;

        AncestorNode rightResult = postOrderSearch(current.right,
firstNode, secondNode);

        if (rightResult.ancestor != null) return rightResult;

        AncestorNode result = new AncestorNode();

        if (leftResult.nodeFound && rightResult.nodeFound) {

            result.ancestor = current;

            return result;

        }

        result.nodeFound =

current == firstNode

|| current == secondNode

```

```

        || leftResult.nodeFound

        || rightResult.nodeFound;

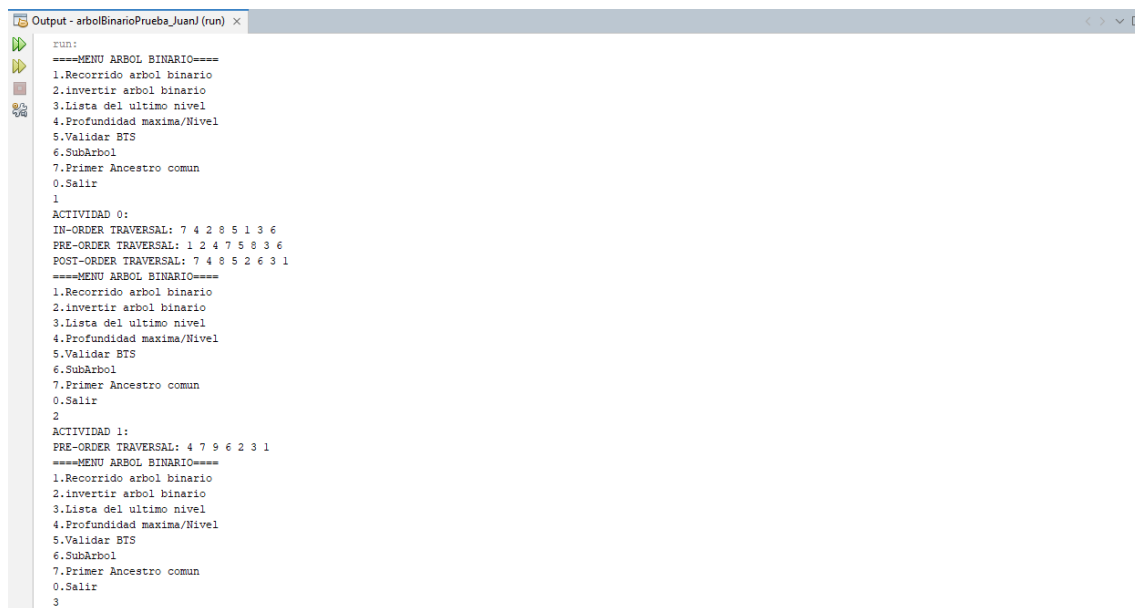
    return result;

}

}

```

## EJECUCION:



```

Output - arbolBinarioPrueba_JuanJ (run) x
run:
====MENU ARBOL BINARIO====
1.Recorrido arbol binario
2.invertir arbol binario
3.Lista del ultimo nivel
4.Profundidad maxima/Nivel
5.Validar BTS
6.SubArbol
7.Primer Ancestro comun
0.Salir
1
ACTIVIDAD 0:
IN-ORDER TRAVERSAL: 7 4 2 8 5 1 3 6
PRE-ORDER TRAVERSAL: 1 2 4 7 5 8 3 6
POST-ORDER TRAVERSAL: 7 4 8 5 2 6 3 1
====MENU ARBOL BINARIO====
1.Recorrido arbol binario
2.invertir arbol binario
3.Lista del ultimo nivel
4.Profundidad maxima/Nivel
5.Validar BTS
6.SubArbol
7.Primer Ancestro comun
0.Salir
2
ACTIVIDAD 1:
PRE-ORDER TRAVERSAL: 4 7 9 6 2 3 1
====MENU ARBOL BINARIO====
1.Recorrido arbol binario
2.invertir arbol binario
3.Lista del ultimo nivel
4.Profundidad maxima/Nivel
5.Validar BTS
6.SubArbol
7.Primer Ancestro comun
0.Salir
3

```

La primera y segunda actividad si funcionan correctamente y muestran el arreglo en varias formas y el 2do muestra el arreglo inverso mediante el pre-orden traversal

```
Output - arbolBinarioPrueba_Juan (run) x
=====MENU ARBOL BINARIO=====
1.Recorrir arbol binario
2.invertir arbol binario
3.Lista del ultimo nivel
4.Profundidad maxima/Nivel
5.Validar BTS
6.SubArbol
7.Primer Ancestro comun
0.Salir
3
ACTIVIDAD 2:
PRE-ORDER TRAVERSAL: 4 2 1 3 7 6 9
Resultado:
[arbolbinarioprueba_juanj.Node@6576fe71] [arbolbinarioprueba_juanj.Node@300ffa5d, arbolbinarioprueba_juanj.Node@1f17ae12] [arbolbinarioprueba_juanj.Node@4d405ef7, arbolbi
=====MENU ARBOL BINARIO=====
1.Recorrir arbol binario
2.invertir arbol binario
3.Lista del ultimo nivel
4.Profundidad maxima/Nivel
5.Validar BTS
6.SubArbol
7.Primer Ancestro comun
0.Salir
4
ACTIVIDAD 3:
=====MENU ARBOL BINARIO=====
1.Recorrir arbol binario
2.invertir arbol binario
3.Lista del ultimo nivel
4.Profundidad maxima/Nivel
5.Validar BTS
6.SubArbol
7.Primer Ancestro comun
0.Salir
5
```

El 3ro muestra mediante el pre-orden trasversal el árbol pero al momento de tener que mostrar que valores son los que se encuentran en el último nivel salen valores que no son los que se desea esto se ha de deber a que los datos solicitados no están siendo bien llamado y debería buscar otra forma para mostrarlos correctamente y el de profundidad del árbol y las opciones que están para abajo no lo eh realizado debido a que estaba buscando la manera de mostrar bien los datos.

```
Output - arbolBinarioPrueba_Juan (run) x
ACTIVIDAD 4:
=====MENU ARBOL BINARIO=====
1.Recorrir arbol binario
2.invertir arbol binario
3.Lista del ultimo nivel
4.Profundidad maxima/Nivel
5.Validar BTS
6.SubArbol
7.Primer Ancestro comun
0.Salir
6
ACTIVIDAD 5:
=====MENU ARBOL BINARIO=====
1.Recorrir arbol binario
2.invertir arbol binario
3.Lista del ultimo nivel
4.Profundidad maxima/Nivel
5.Validar BTS
6.SubArbol
7.Primer Ancestro comun
0.Salir
7
ACTIVIDAD 6:
=====MENU ARBOL BINARIO=====
1.Recorrir arbol binario
2.invertir arbol binario
3.Lista del ultimo nivel
4.Profundidad maxima/Nivel
5.Validar BTS
6.SubArbol
7.Primer Ancestro comun
0.Salir
0
Gracias por ingresar
BUILD SUCCESSFUL (total time: 6 seconds)
```



Lo explicado anteriormente el dato es que funcionan solo que esta de encontrar la manera que estos datos que realizan cada actividad se muestren en pantalla para poder visualizar los datos que sería lo único faltante del trabajo.