UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 2

LABORATORIO NIVEL 11

Objetivos

1. Iniciar con el almacenamiento de los Arboles Binarios Ordenados
2. Aplicar los conocimientos de ABO en la construcción simple de un sistema para mostrar los datos de un árbol binario Ordenado

Número de participantes

Duración de la actividad

Materiales

- Internet
- Eclipse
- Karen Julieth Perez Rivera

INSTRUCCIONES

1. Esta es una Guía de Aprendizaje para comprender el funcionamiento de un árbol binario ordenado

Objetivo:
El objetivo de este ejercicio es implementar un árbol binario ordenado (árbol de búsqueda binaria) en Java, utilizando el paradigma de programación orientada a objetos. Aprenderás a insertar valores en el árbol y a recorrerlo para imprimir sus elementos en orden ascendente.

Descripción del Problema:
Se te pide que implementes un programa en Java que represente un árbol binario ordenado. Este árbol deberá permitir las siguientes funcionalidades:

Inserción de Nodos: Permitir al usuario insertar valores enteros en el árbol. Los valores deben ser colocados de manera que para cualquier nodo:

- Todos los valores en su subárbol izquierdo son menores que el valor del nodo.
- Todos los valores en su subárbol derecho son mayores que el valor del nodo.

Recorrido del Árbol: Implementar un método para recorrer el árbol en orden (in-order) y mostrar los valores de los nodos en orden ascendente.

Entrada y Salida:

- La entrada será una serie de valores enteros que se insertarán en el árbol.
- La salida del programa debe ser una lista de los valores almacenados en el árbol, impresos en orden ascendente.

Requerimientos No funcionales:
No se permiten valores duplicados en el árbol.
El ejercicio debe realizarse en un java aplication para consola

Entregables:
Un archivo Java que contenga la implementación completa del árbol binario ordenado y un método main para probar su funcionalidad.

Requerimientos Funcionales

1. **Inserción de Nodos**:
   - El sistema debe permitir la inserción de valores enteros en el árbol binario ordenado.
   - Al insertar un nuevo valor, el sistema debe verificar que no haya valores duplicados.
   - Los valores deben ser colocados en el árbol de acuerdo con la propiedad del árbol binario: valores menores en el subárbol izquierdo y valores mayores en el subárbol derecho.
2. **Recorrido en Orden**:

ESTRUCTURAS DE DATOS 2

LABORATORIO NIVEL 11

- El sistema debe implementar un método que recorra el árbol en orden (in-order) y devuelva una lista de los valores almacenados en el árbol en orden ascendente.
- La salida debe mostrarse en la consola.

3. **Estructura del Árbol**:
   - El sistema debe utilizar clases para representar tanto los nodos del árbol como la estructura del árbol en sí.
   - Cada nodo debe contener un valor, una referencia al hijo izquierdo y una referencia al hijo derecho.
   - Interfaz de Usuario:
   - El sistema debe ejecutarse en la consola y mostrar un mensaje claro que indique que los valores del árbol se imprimirán en orden.
4. **Manejo de Errores**:
   - El sistema debe manejar adecuadamente la inserción de valores duplicados, mostrando un mensaje de advertencia o ignorando la inserción.

CREACIÓN DE CLASES

Clase NODO

```
package Clases;

public class Nodo {
	int valor;
	Nodo izquierdo;
	Nodo derecho;

	// Constructor que inicializa solo el valor del nodo
	public Nodo(int valor) {
		this.valor = valor;
		this.izquierdo = null;
		this.derecho = null;
	}

	// Constructor que inicializa con valor y nodos hijos
	public Nodo(int valor, Nodo izquierdo, Nodo derecho) {
		this.valor = valor;
		this.izquierdo = izquierdo;
		this.derecho = derecho;
	}

	// Getters y setters para el atributo 'valor'
	public int getValor() {
		return valor;
	}

	public void setValor(int valor) {
		this.valor = valor;
	}

	// Getters y setters para el nodo izquierdo
	public Nodo getIzquierdo() {
```

ESTRUCTURAS DE DATOS 2

LABORATORIO NIVEL 11

```
                return izquierdo;
        }

        public void setIzquierdo(Nodo izquierdo) {
                this.izquierdo = izquierdo;
        }

        // Getters y setters para el nodo derecho
        public Nodo getDerecho() {
                return derecho;
        }

        public void setDerecho(Nodo derecho) {
                this.derecho = derecho;
        }

        // Método que verifica si el nodo es hoja (no tiene hijos)
        public boolean esHoja() {
                return (this.izquierdo == null && this.derecho == null);
        }

        // Método para representar el nodo como cadena
        @Override
        public String toString() {
                return "Nodo{" + "valor=" + valor + ", izquierdo=" + (izquierdo != null ? izquierdo.valor : "null")
                                + ", derecho=" + (derecho != null ? derecho.valor : "null") + '}';
        }

}
```

Clase ArbolBinario

```
package Clases;

public class ArbolBinario {
        private Nodo raiz;

   // Clase interna para almacenar el nodo y su padre
   private class ResultadoBusqueda {
      Nodo nodo;   // El nodo encontrado
      Nodo padre;  // El padre del nodo encontrado

      ResultadoBusqueda(Nodo nodo, Nodo padre) {
         this.nodo = nodo;
         this.padre = padre;
      }
   }

   // Método público para insertar un nuevo valor en el árbol
```

UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 2

LABORATORIO NIVEL 11

```
public boolean insertar(int valor) {
    if (raiz == null) {
        raiz = new Nodo(valor);  // Si el árbol está vacío, creamos la raíz
        return true;
    }
    return insertarRecursivo(raiz, valor);
}


        // Método recursivo para insertar un nuevo valor en el árbol binario
        private Nodo insertarRec(Nodo raiz, int valor) {
                // Si el árbol está vacío, crea un nuevo nodo
                if (raiz == null) {
                        return new Nodo(valor); // Usamos el constructor de Nodo
                }

                // Recorrido del árbol de acuerdo al valor
                if (valor < raiz.getValor()) {
                        raiz.setIzquierdo(insertarRec(raiz.getIzquierdo(), valor));
                } else if (valor > raiz.getValor()) {
                        raiz.setDerecho(insertarRec(raiz.getDerecho(), valor));
                }

                return raiz;
        }

        // Método para recorrer el árbol en orden (in-order traversal)
        public void recorridoInOrden() {
                recorridoInOrdenRec(raiz);
        }

        // Método recursivo para el recorrido en orden
        private void recorridoInOrdenRec(Nodo raiz) {
                if (raiz != null) {
                        recorridoInOrdenRec(raiz.getIzquierdo()); // Primero, hijo izquierdo
                        System.out.print(raiz.getValor() + " "); // Luego, el nodo actual
                        recorridoInOrdenRec(raiz.getDerecho()); // Finalmente, hijo derecho
                }
        }

        // Método para buscar un número en el árbol
        public boolean buscar(int valor) {
                return buscarRec(raiz, valor);
        }

        // Método recursivo para buscar un valor en el árbol
        private boolean buscarRec(Nodo raiz, int valor) {
```

ESTRUCTURAS DE DATOS 2

LABORATORIO NIVEL 11

```
                    if (raiz == null) {
                            return false; // No se encontró el valor
                    }

                    if (raiz.getValor() == valor) {
                            return true; // Valor encontrado
                    }

                    // Buscar en el subárbol izquierdo o derecho dependiendo del valor
                    if (valor < raiz.getValor()) {
                            return buscarRec(raiz.getIzquierdo(), valor);
                    } else {
                            return buscarRec(raiz.getDerecho(), valor);
                    }
            }

}
```

Clase main

```
package Clases;

public class ArbolBinario {
        private Nodo raiz;

  // Clase interna para almacenar el nodo y su padre
  private class ResultadoBusqueda {
    Nodo nodo;   // El nodo encontrado
    Nodo padre;  // El padre del nodo encontrado

    ResultadoBusqueda(Nodo nodo, Nodo padre) {
      this.nodo = nodo;
      this.padre = padre;
    }
  }

  // Método público para insertar un nuevo valor en el árbol
  public boolean insertar(int valor) {
    if (raiz == null) {
      raiz = new Nodo(valor);  // Si el árbol está vacío, creamos la raíz
      return true;
    }
    return insertarRecursivo(raiz, valor);
  }


        // Método recursivo para insertar un nuevo valor en el árbol binario
        private Nodo insertarRec(Nodo raiz, int valor) {
                // Si el árbol está vacío, crea un nuevo nodo
```

ESTRUCTURAS DE DATOS 2

LABORATORIO NIVEL 11

```
        if (raiz == null) {
            return new Nodo(valor); // Usamos el constructor de Nodo
        }

        // Recorrido del árbol de acuerdo al valor
        if (valor < raiz.getValor()) {
            raiz.setIzquierdo(insertarRec(raiz.getIzquierdo(), valor));
        } else if (valor > raiz.getValor()) {
            raiz.setDerecho(insertarRec(raiz.getDerecho(), valor));
        }

        return raiz;
    }

    // Método para recorrer el árbol en orden (in-order traversal)
    public void recorridoInOrden() {
        recorridoInOrdenRec(raiz);
    }

    // Método recursivo para el recorrido en orden
    private void recorridoInOrdenRec(Nodo raiz) {
        if (raiz != null) {
            recorridoInOrdenRec(raiz.getIzquierdo()); // Primero, hijo izquierdo
            System.out.print(raiz.getValor() + " "); // Luego, el nodo actual
            recorridoInOrdenRec(raiz.getDerecho()); // Finalmente, hijo derecho
        }
    }

    // Método para buscar un número en el árbol
    public boolean buscar(int valor) {
        return buscarRec(raiz, valor);
    }

    // Método recursivo para buscar un valor en el árbol
    private boolean buscarRec(Nodo raiz, int valor) {
        if (raiz == null) {
            return false; // No se encontró el valor
        }

        if (raiz.getValor() == valor) {
            return true; // Valor encontrado
        }

        // Buscar en el subárbol izquierdo o derecho dependiendo del valor
        if (valor < raiz.getValor()) {
            return buscarRec(raiz.getIzquierdo(), valor);
        } else {
```

UNIVERSIDAD MARIANA
FACULTAD INGENIERÍA – PROGRAMA INGENIERÍA DE SISTEMAS

ESTRUCTURAS DE DATOS 2

LABORATORIO NIVEL 11

```
                    return buscarRec(raiz.getDerecho(), valor);
                }
        }

}
```

REPOSITORIO: https://github.com/Kareennn2/ARBOLBINARIO.git