Universidad Tecnológica del Norte de Guanajuato

Organismo Público Descentralizado del Gobierno del Estado

"Educación y progreso para la vida"

Licenciatura en Ingeniería de Tecnologías de la Información e Innovación Digital

Desarrollo de software

Unidad III.

Estructuras de datos básicas

Actividad. Implementación de un Árbol Binario

Materia

Estructura de datos avanzadas

Alumno:

Ramiro Padierna Delgado 1224100710

Profesor:

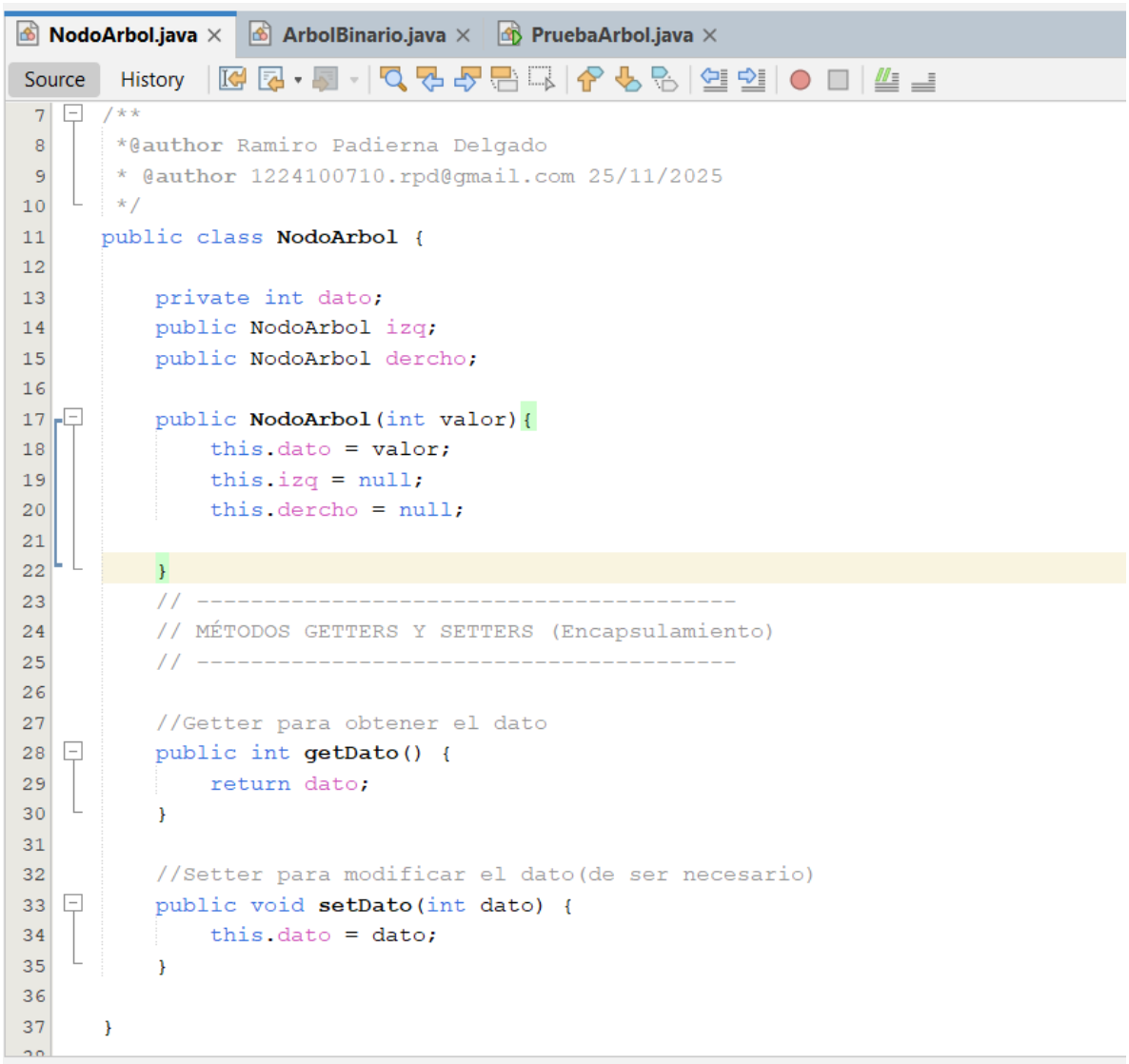
Gabriel Barrón Rodríguez

Universidad del norte de Guanajuato D.H.C.I.

25/Noviembre/2025

Implementación de un Árbol Binario

Se crea el código del NodoArbol




```java
/**
 *@author Ramiro Padierna Delgado
 * @author 1224100710.rpd@gmail.com 25/11/2025
 */
public class NodoArbol {

    private int dato;
    public NodoArbol izq;
    public NodoArbol dercho;

    public NodoArbol(int valor){
        this.dato = valor;
        this.izq = null;
        this.dercho = null;

    }
    // ----------------------------------------
    // MÉTODOS GETTERS Y SETTERS (Encapsulamiento)
    // ----------------------------------------

    //Getter para obtener el dato
    public int getDato() {
        return dato;
    }

    //Setter para modificar el dato(de ser necesario)
    public void setDato(int dato) {
        this.dato = dato;
    }

}
```

Se crea el código del ArbolBinario

NodoArbol.java | ArbolBinario.java | PruebaArbol.java

Source | History

```java
public class ArbolBinario {

    private NodoArbol raiz;

    public ArbolBinario(){
        this.raiz = null;
    }
    //Se crea el metodo publico de insercion
    public void insertar(int valor){
        this.raiz = insertarRecursivo(this.raiz, valor);

    }
    private NodoArbol insertarRecursivo(NodoArbol actual, int valor) {
   // Caso Base: Si el nodo actual es null, encontramos la posición, creamos el
  //nuevo nodo y lo retornamos.
   if (actual == null) {
   return new NodoArbol(valor);
  }
   if (valor < actual.getDato()) { // Usamos getDato() por el encapsulamiento
   //estricto
    actual.izq = insertarRecursivo(actual.izq, valor);
    } else if (valor > actual.getDato()) {
    actual.dercho = insertarRecursivo(actual.dercho, valor);
    }
     // Si valor == actual.getDato(), se ignora (no permite duplicados).
     return actual; // Retorna el nodo actual sin cambios si no fue caso base.
      }
    /**
    * Método Público de Recorrido Inorden.
    * Inicia la recursión desde la raíz y muestra el resultado.
    */
```

Se crea donde se realizará la prueba del árbol

```java
/**
 * Se realizara la prueba de Arbol para revisar si ha funcionado adecuademente
 * @author Ramiro Padierna Delgado
 * @author 1224100710.rpd@gmail.com 25/11/2025
 */
public class PruebaArbol {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // 1. Crear una instancia de la clase gestora del árbol
        ArbolBinario arbol = new ArbolBinario();
        System.out.println("Insertando valores: 50, 30, 70, 20, 40");
       // 2. Insertar valores usando el método público
        arbol.insertar(50); // Raíz
        arbol.insertar(30); // Izquierda de 50
        arbol.insertar(70); // Derecha de 50
        arbol.insertar(20); // Izquierda de 30
        arbol.insertar(40); // Derecha de 30
// 3. Ejecutar el recorrido para verificar el orden
// Resultado esperado (ordenado): 20 30 40 50 70
arbol.recorrerInorden();
    }

}
```

El resultado es

```
Output - Unidad3 (run)
    run:
    Insertando valores: 50, 30, 70, 20, 40
    Recorrido Inorden: 20 30 40 50 70
    BUILD SUCCESSFUL (total time: 0 seconds)
```

Ejemplo de prueba para BTS

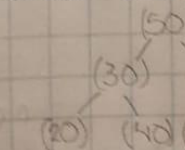
Ejemplo de prueba para BTS

Instrucciones:

Prueba 1

Datos a insertar: [50, 30, 70, 20, 40, 60, 80]

Insertar 50 sera la raiz



Al insertar
50 raiz
30 < 50 al lado izquierdo
70 > 50 al lado derecho
20 < 50 y 20 < 30 al lado izquier
de ambos
40 < 50 pero 40 > 30 va al
lado izquierdo y despues al
lado derecho

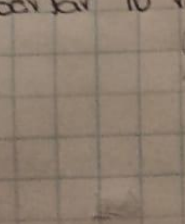
60 > 50 pero 60 < 70 va al
lado derecho y despues al
lado izquierdo
80 > 50 y 80 > 70 enlances
va al lado derecho de
ambos

Recorrido esperado
20, 30, 40, 50, 60, 70, 80

Prueba 2:

Datos a insertar: [10, 20, 30, 40, 50]

insertar 10 raiz



10 es raiz
20 > 10 del lado derecho
30 > 10, 30 > 10 de lado
derecho de ambos
40 > 10, 40 > 20, 40 > 30
del lado derecho de
todos

50 > 10, 50 > 20, 50 > 30, 50 > 40
del lado derecho de todos

Recorrido inorden esperado 10, 20, 30, 40, 50

Prueba 3

Datos a insertar [45, 25, 75, 15, 35, 65, 85, 5, 18, 30, 38

Insertar 45 raiz

Insertar 25: Izquierda de 45

Insertar 75: Derecha de 45

Insertar 15: Izquierda de 15

Insertar 35: Derecha de 25

Insertar 65: Izquierda de 75

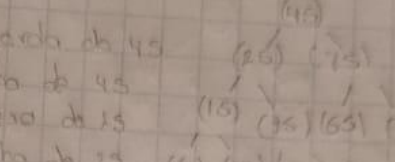
Insertar 85: Derecha de 75

Insertar 5: Izquierda de 15

Insertar 18: Derecha de 15

Insertar 30: Izquierda de 35

Insertar 38: Derecha de 35



Recorrido Inorden Esperado

5, 15, 18, 25, 30, 35, 38, 45, 65, 75, 8

Prueba 4

Datos a insertar: [50, 30, 70, 30, 50, 20, 70]

Insertar 50 Raiz

Insertar 30: Izquierda de 50

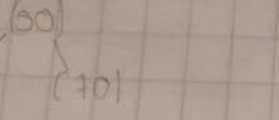
Insertar 70: Derecha de 50

Insertar 30: 30 == 30; Se ignora! CV arbol no cambia

Insertar 50: 50 == 50, ¡se ignora! CV arbol no cambia

Insertar 20: Izquierda de 30

Insertar 70: 70 == 70, ¡Se ignora! CV arbol no cambia



Recorrido Inorden Esperado

20 30 50 70