

Arbol Binario

```
import ArbolBinarioCompleto

fun main() {
    val arbol = ArbolBinarioCompleto()
    var arbolConstruido = false
    var opcion: Int?

    do {
        println("\n=== menu arbol binario completo ===")
        println("1. insertar elementos")
        println("2. recorrido preorden")
        println("3. recorrido inorden")
        println("4. recorrido postorden")
        println("5. salir")
        print("elige una opcion: ")

        opcion = readLine()?.toIntOrNull()

        when (opcion) {
            1 -> {
                if (arbolConstruido) {
                    println("ya creaste el arbol")
                } else {
                    var niveles: Int? = null
                    while (niveles == null || niveles <= 0) {
                        print("dame los niveles del arbol: ")
                        niveles = readLine()?.toIntOrNull()
                        if (niveles == null || niveles <= 0) {

```

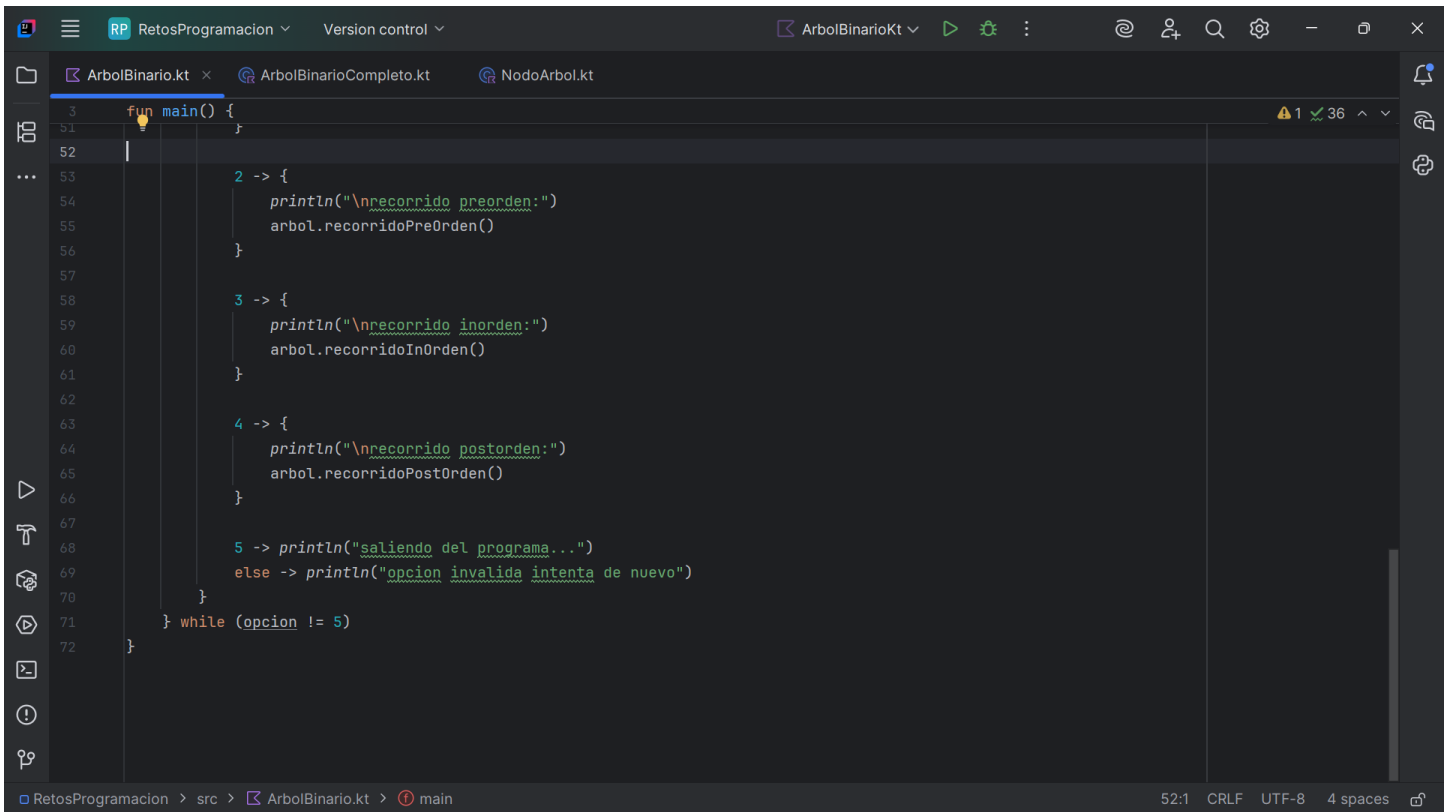
```
fun main() {
    niveles = readLine()?.toIntOrNull()
    if (niveles == null || niveles <= 0) {
        println("entrada invalida niveles > 0")
    }

    val totalNodos = (1 shl niveles) - 1
    val valores = mutableListOf<Int>()
    println("ingresa $totalNodos valores enter despues de cada valor:")

    while (valores.size < totalNodos) {
        print("valor ${valores.size + 1}: ")
        val valor = readLine()?.toIntOrNull()
        if (valor != null) {
            valores.add(valor)
        } else {
            println("valor invalido")
        }
    }

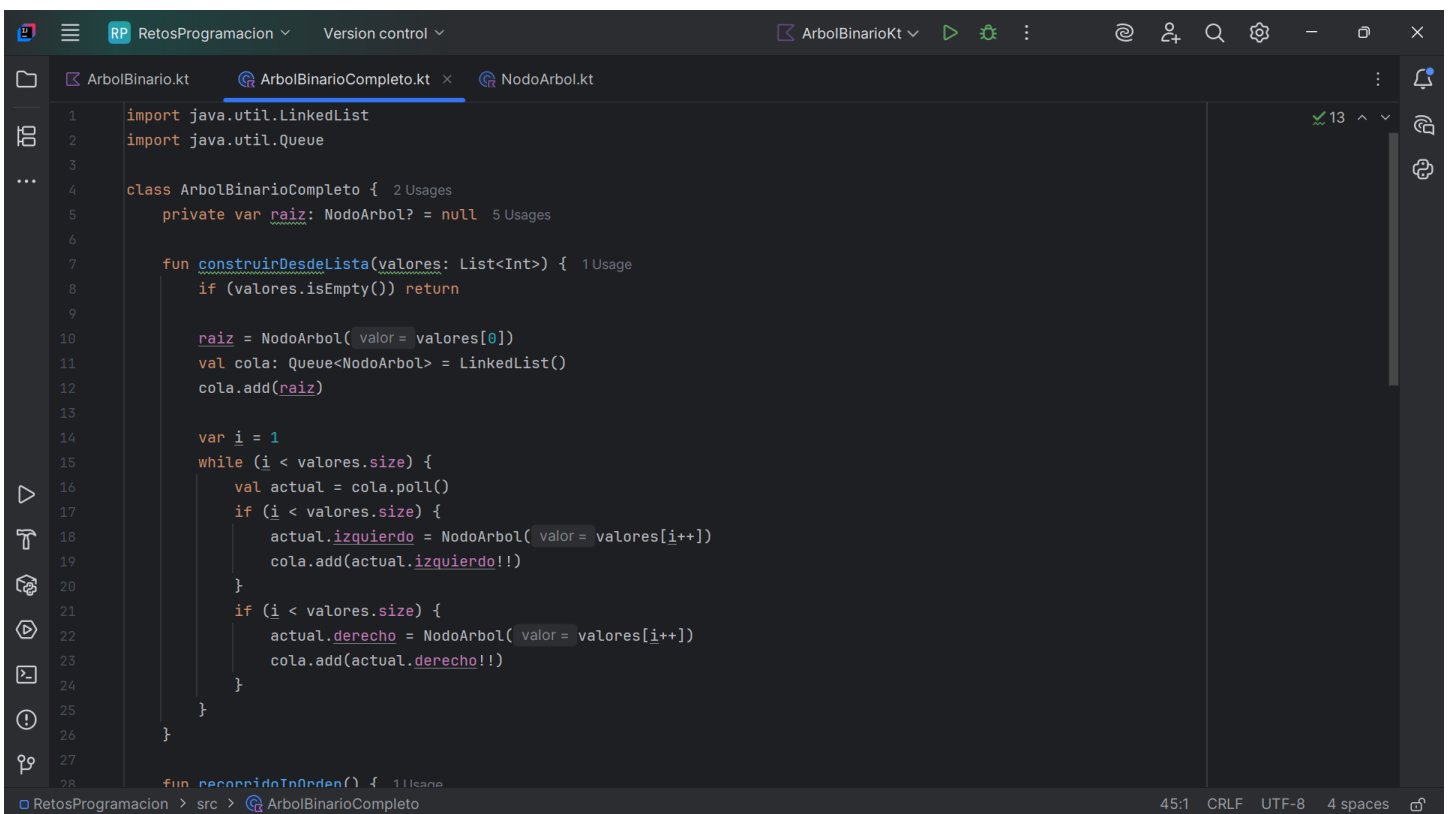
    arbol.construirDesdeLista(valores)
    arbolConstruido = true
    println("arbol construido exitosamente")
}

2 -> {
```



```
3 fun main() {
51
52
53     2 -> {
54         println("\nrecorrido preorden:")
55         arbol.recorridoPreOrden()
56     }
57
58     3 -> {
59         println("\nrecorrido inorden:")
60         arbol.recorridoInOrden()
61     }
62
63     4 -> {
64         println("\nrecorrido postorden:")
65         arbol.recorridoPostOrden()
66     }
67
68     5 -> println("saliendo del programa...")
69     else -> println("opcion invalida intenta de nuevo")
70 }
71 } while (opcion != 5)
72 }
```

Clase Arbol Binario Completo

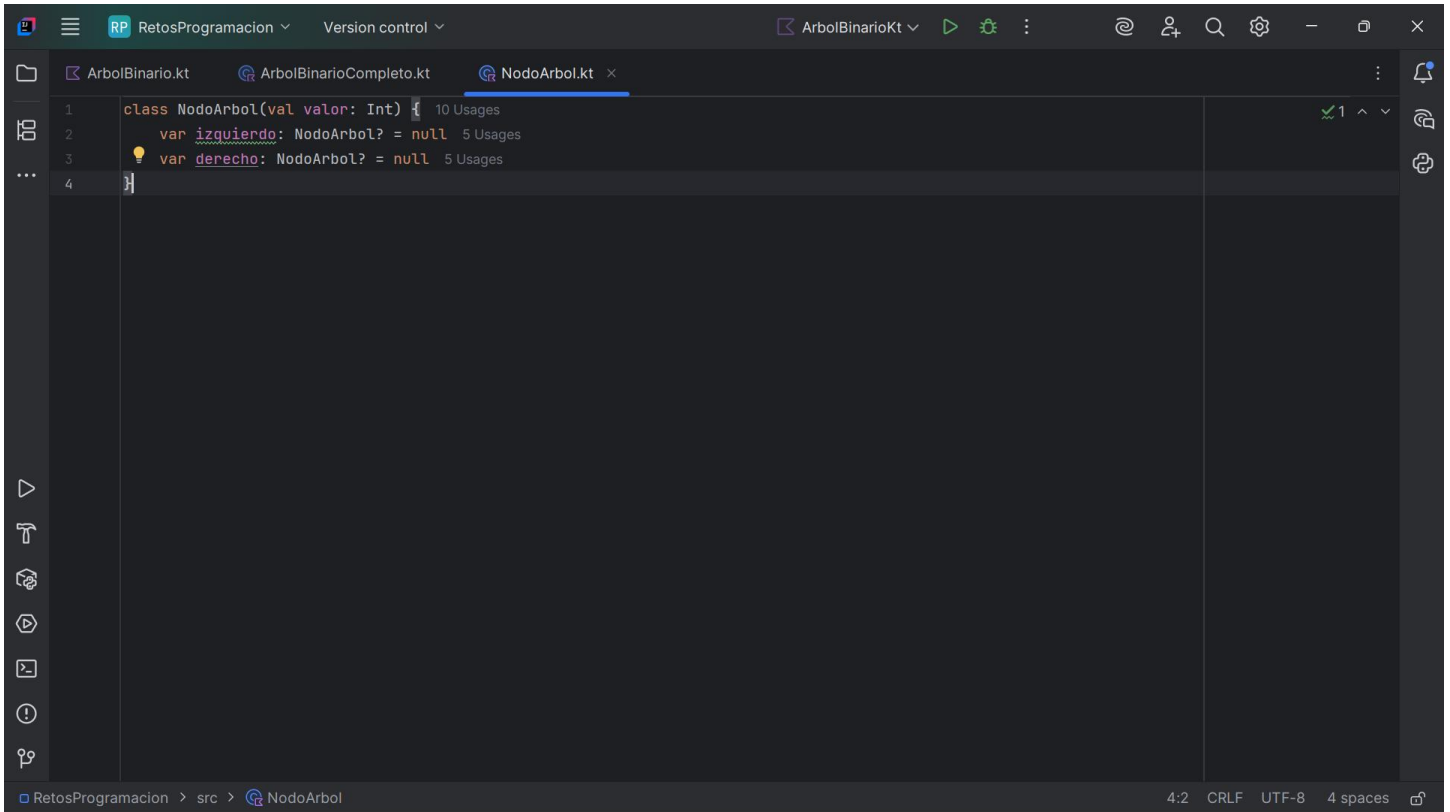


```
1 import java.util.LinkedList
2 import java.util.Queue
3
4 class ArbolBinarioCompleto { 2 Usages
5     private var raiz: NodoArbol? = null 5 Usages
6
7     fun construivDesdeLista(valores: List<Int>) { 1 Usage
8         if (valores.isEmpty()) return
9
10        raiz = NodoArbol( valor = valores[0])
11        val cola: Queue<NodoArbol> = LinkedList()
12        cola.add(raiz)
13
14        var i = 1
15        while (i < valores.size) {
16            val actual = cola.poll()
17            if (i < valores.size) {
18                actual.izquierdo = NodoArbol( valor = valores[i++])
19                cola.add(actual.izquierdo!!)
20            }
21            if (i < valores.size) {
22                actual.derecho = NodoArbol( valor = valores[i++])
23                cola.add(actual.derecho!!)
24            }
25        }
26    }
27
28    fun recorridoInOrden() { 11 Usages
```

```
RP RetosProgramacion Version control ArbolBinarioKt ArbolBinarioCompleto.kt x NodoArbol.kt
4 class ArbolBinarioCompleto { 2 Usages
28 fun recorridoInOrden() { 1 Usage
29     inOrden( nodo = raiz)
30     println()
31 }
32
33 private fun inOrden(nodo: NodoArbol?) { 3 Usages
34     if (nodo != null) {
35         inOrden( nodo = nodo.izquierdo)
36         print("${nodo.valor} ")
37         inOrden( nodo = nodo.derecho)
38     }
39 }
40
41 fun recorridoPreOrden() { 1 Usage
42     preOrden( nodo = raiz)
43     println()
44 }
45
46 private fun preOrden(nodo: NodoArbol?) { 3 Usages
47     if (nodo != null) {
48         print("${nodo.valor} ")
49         preOrden( nodo = nodo.izquierdo)
50         preOrden( nodo = nodo.derecho)
51     }
52 }
53
54 fun recorridoPostOrden() { 1 Usage
```

```
RP RetosProgramacion Version control ArbolBinarioKt ArbolBinarioCompleto.kt x NodoArbol.kt
4 class ArbolBinarioCompleto { 2 Usages
45
46 private fun preOrden(nodo: NodoArbol?) { 3 Usages
47     if (nodo != null) {
48         print("${nodo.valor} ")
49         preOrden( nodo = nodo.izquierdo)
50         preOrden( nodo = nodo.derecho)
51     }
52 }
53
54 fun recorridoPostOrden() { 1 Usage
55     postOrden( nodo = raiz)
56     println()
57 }
58
59 private fun postOrden(nodo: NodoArbol?) { 3 Usages
60     if (nodo != null) {
61         postOrden( nodo = nodo.izquierdo)
62         postOrden( nodo = nodo.derecho)
63         print("${nodo.valor} ")
64     }
65 }
66 }
```

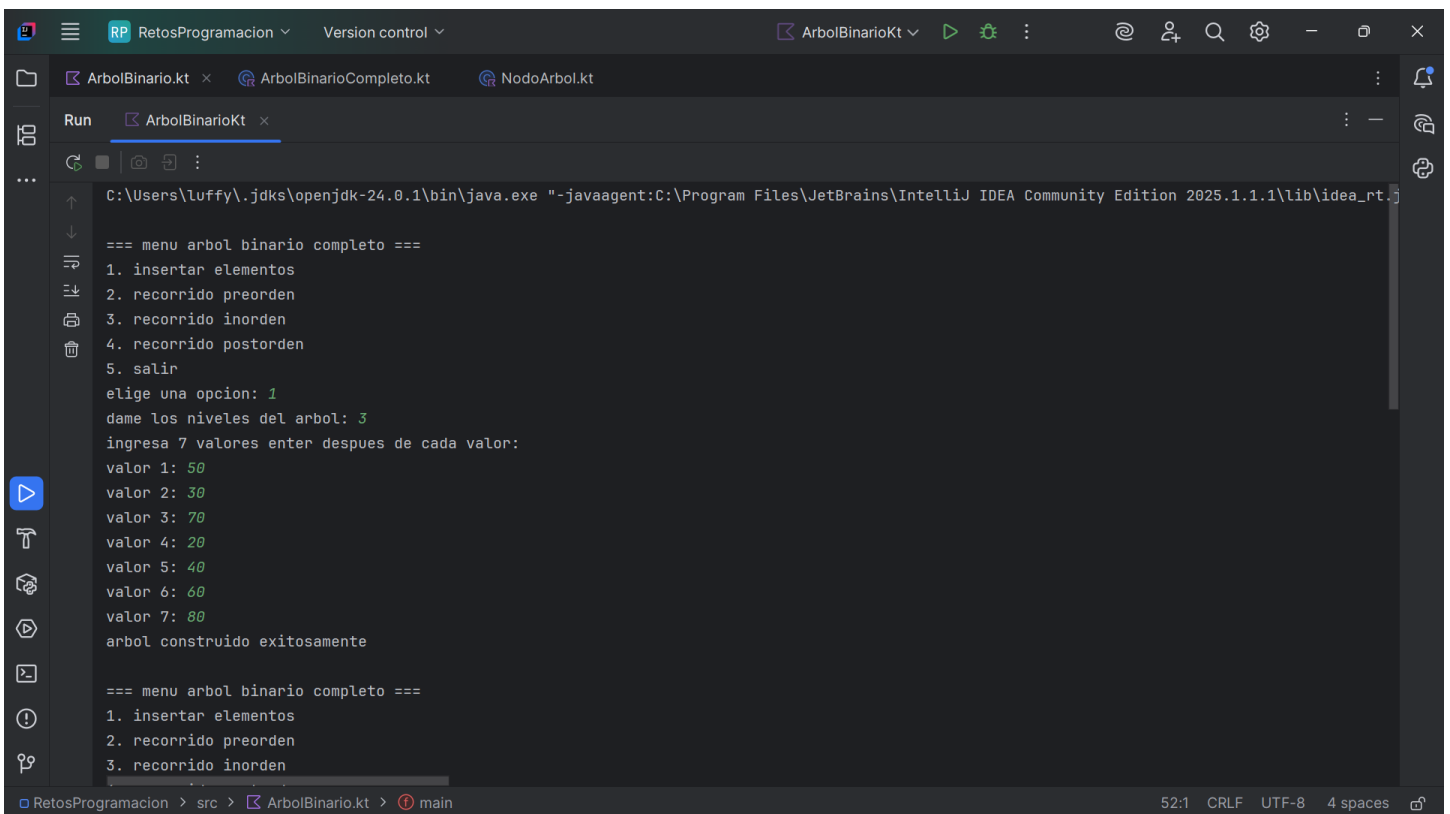
Claso Nodo



The screenshot shows the IntelliJ IDEA editor with the file 'NodoArbol.kt' open. The code defines a class 'NodoArbol' with a constructor taking an 'Int' value and two nullable 'NodoArbol?' properties, 'izquierdo' and 'derecho', both initialized to 'null'. The IDE interface includes a top toolbar with icons for running, debugging, and other actions, and a bottom status bar showing the file path 'RetosProgramacion > src > NodoArbol' and encoding details '4:2 CRLF UTF-8 4 spaces'.

```
1 class NodoArbol(val valor: Int) {
2     var izquierdo: NodoArbol? = null
3     var derecho: NodoArbol? = null
4 }
```

Consola



The screenshot shows the IntelliJ IDEA console with the output of a Java program. The program displays a menu for a binary tree, prompts the user to select an option, and then prompts for 7 integer values to build the tree. The output shows the user selecting option 1, entering 3 levels, and providing 7 integer values (50, 30, 70, 20, 40, 60, 80). The program then prints 'arbol construido exitosamente' and displays the menu again.

```
C:\Users\luffy\.jdk\openjdk-24.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.1\lib\idea_rt.jar"

=== menu arbol binario completo ===
1. insertar elementos
2. recorrido preorden
3. recorrido inorden
4. recorrido postorden
5. salir
elige una opcion: 1
dame los niveles del arbol: 3
ingresa 7 valores enter despues de cada valor:
valor 1: 50
valor 2: 30
valor 3: 70
valor 4: 20
valor 5: 40
valor 6: 60
valor 7: 80
arbol construido exitosamente

=== menu arbol binario completo ===
1. insertar elementos
2. recorrido preorden
3. recorrido inorden
```

```
ArbolBinario.kt x ArbolBinarioCompleto.kt x NodoArbol.kt
Run ArbolBinario.kt x
4. recorrido postorden
5. salir
elige una opcion: 2

recorrido preorden:
50 30 20 40 70 60 80

=== menu arbol binario completo ===
1. insertar elementos
2. recorrido preorden
3. recorrido inorden
4. recorrido postorden
5. salir
elige una opcion: 3

recorrido inorden:
20 30 40 50 60 70 80

=== menu arbol binario completo ===
1. insertar elementos
2. recorrido preorden
3. recorrido inorden
4. recorrido postorden
5. salir
elige una opcion: 4
```

```
ArbolBinario.kt x ArbolBinarioCompleto.kt x NodoArbol.kt
Run ArbolBinario.kt x
20 30 40 50 60 70 80

=== menu arbol binario completo ===
1. insertar elementos
2. recorrido preorden
3. recorrido inorden
4. recorrido postorden
5. salir
elige una opcion: 4

recorrido postorden:
20 40 30 60 80 70 50

=== menu arbol binario completo ===
1. insertar elementos
2. recorrido preorden
3. recorrido inorden
4. recorrido postorden
5. salir
elige una opcion: 5
saliendo del programa...

Process finished with exit code 0
```