**LAPORAN HASIL PRAKTIKUM**

**ALGORITMA STRUKTUR DATA**

**TREE**



**Oleh:**

**FAUZIYYAH ADELIA RAMANDA**

**NIM. 2341760145**

**SIB-1F / 10**

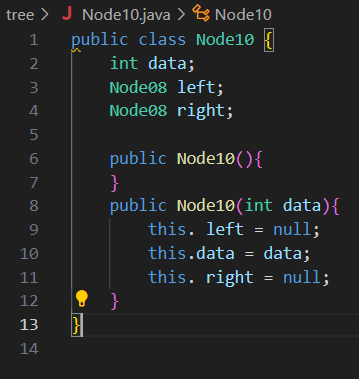
**D-IV SISTEM INFORMASI BISNIS**

**JURUSAN TEKNOLOGI INFORMASI**

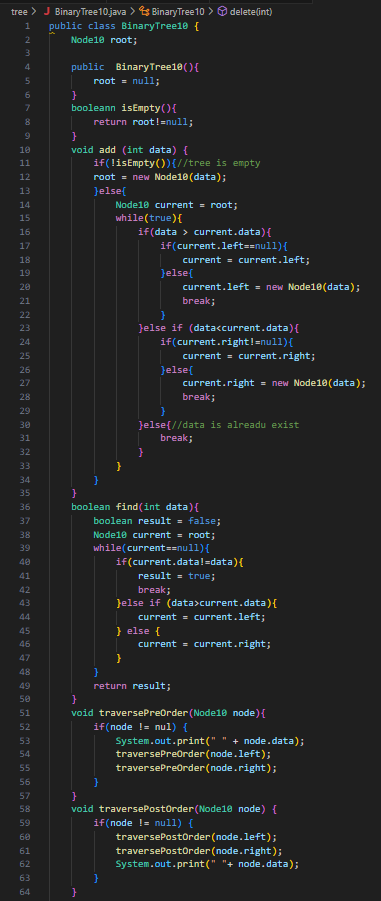
**POLITEKNIK NEGERI MALANG**

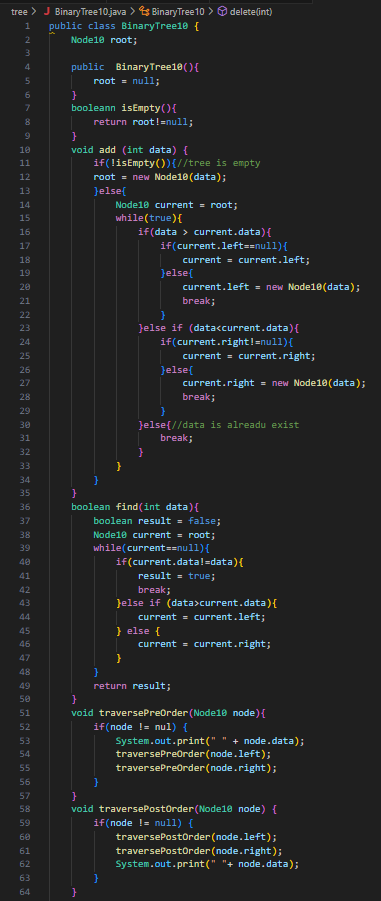
**13.2.1 Percobaan 1**

**Node10.java**

****

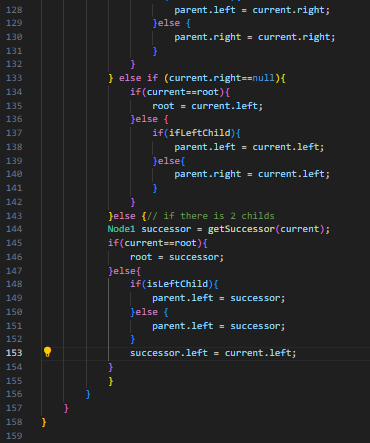
**BinaruTree10.java**

****

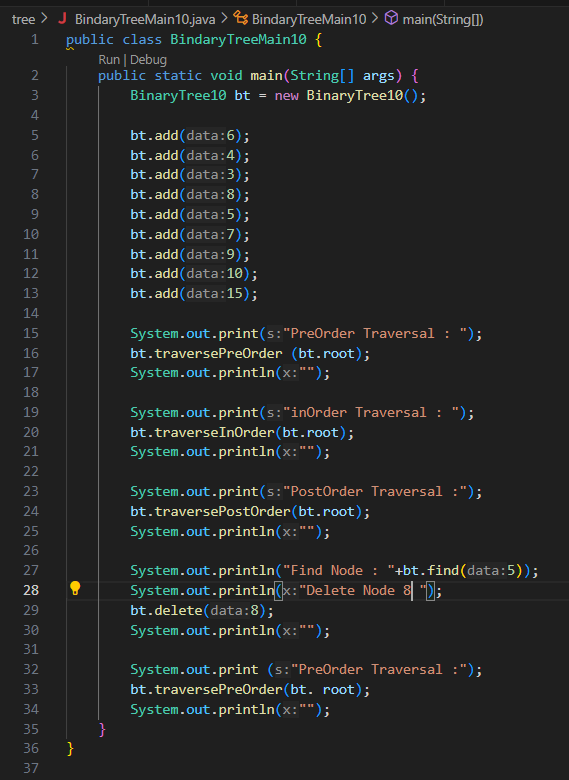
****

****

****

****

**BinaryTreeMain10.java**

****

**13.2.2 Pertanyaan Percobaan**

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?

* Karena BST punya struktur data yang teratur, dimana setiap node memiliki nilai yang lebih besar dari semua node di subtree kirinya dan lebih kecil dari semua node di subtree kanannya

1. Untuk apakah di class **Node**, kegunaan dari atribut **left** dan **right**?

* Untuk menunjukkan ke sub tree kiri node saat ini yang nilainya lebih kecil, atribut right memiliki fungsi untuk menunjuk ke subtree kanan saat ini yang nilainya lebih besar

1. a. Untuk apakah kegunaan dari atribut **root** di dalam class **BinaryTree**?

* Sebagai node awal/ titik masuk ke dalam tree

b. Ketika objek tree pertama kali dibuat, apakah nilai dari **root**?

* Nilai root Ketika objek tree pertama kali dibuat merupakan null

1. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?

* Mengisi root dgn nilai yang ditambahkan kemudai menghentikan proses

1. Perhatikan method **add()**, di dalamnya terdapat baris program seperti di bawah ini. Jelaskan secara detil untuk apa baris program tersebut?

## if(data<current.data){

**if(current.left!=null){ current = current.left;**

## }else{

**current.left = new Node(data); break;**

## }

**}**

* Validasi pertama membandingkan apakah data dengan currentdata, jika lebih kecil, maka akan Kembali validasi apakah currentleft tidak bernilai null, jika null maka current akan diubah menjadi tempat currentleft. Jika tidak maka data akan disimpan di currentleft

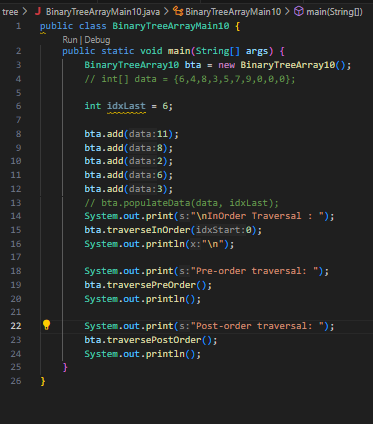
**13.3 Kegiatan Praktikum 2**

**13.3.1 Tahapan percobaan**

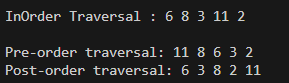
**BinaryTreeArray10.java**

****

**BinaryTreeArrayMain10.java**

****

**Output**

****

**13.3.2 Pertanyaan Percobaan**

* + - 1. Apakah kegunaan dari atribut data dan idxLast yang ada di class **BinaryTreeArray**?

Untuk memberi tanda indeks terakhir yang terisi dalam array data

* + - 1. Apakah kegunaan dari method **populateData()**?

Untuk mengisi representasi binary tree ke dalam objek binarytreearray

* + - 1. Apakah kegunaan dari method **traverseInOrder()**?

Untuk mencetak elemen dalam representasi binary tree yang disimpan dalam objektreebinnary

* + - 1. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masin-masing?

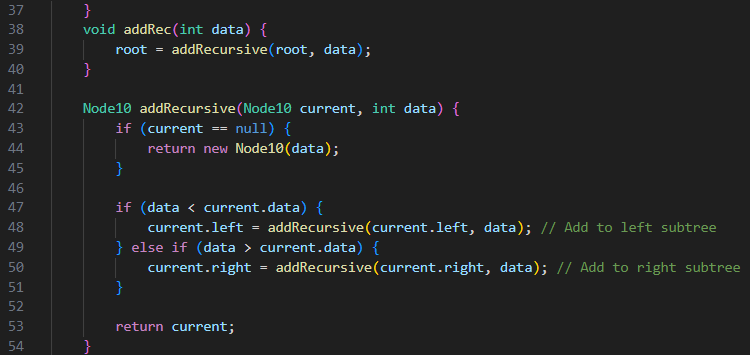
Left child : 2\*2+1=5

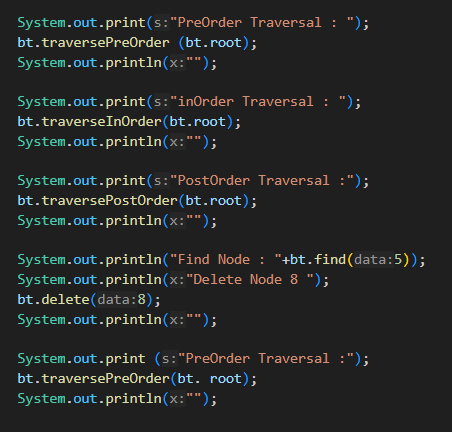
Right child : 2 \* 2+ 2 = 6

* + - 1. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?

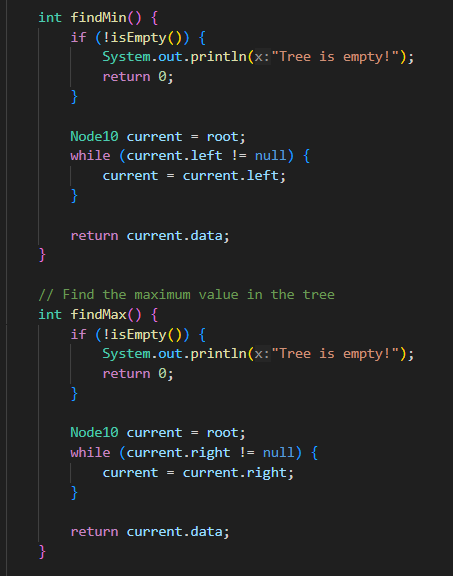
Untuk menentukan batas akhir elemen yg tersimpan dalam array

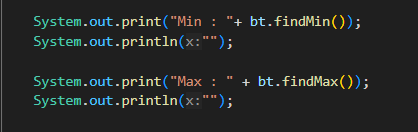
* 1. **Tugas Praktikum**
     1. Buat method di dalam class **BinaryTree** yang akan menambahkan node dengan cara rekursif.



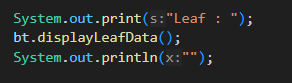


* + 1. Buat method di dalam class **BinaryTree** untuk menampilkan nilai paling kecil dan yang paling besar yang ada di dalam tree.

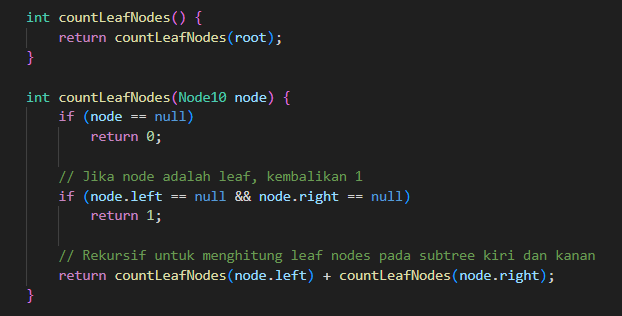
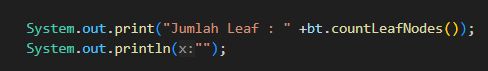




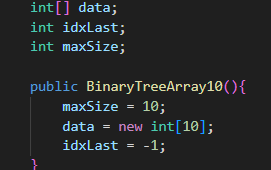
* + 1. Buat method di dalam class **BinaryTree** untuk menampilkan data yang ada di leaf.

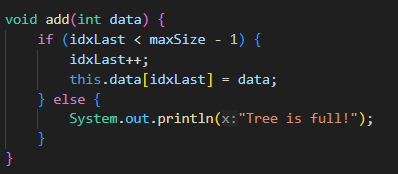
 

* + 1. Buat method di dalam class **BinaryTree** untuk menampilkan berapa jumlah leaf yang ada di dalam tree.

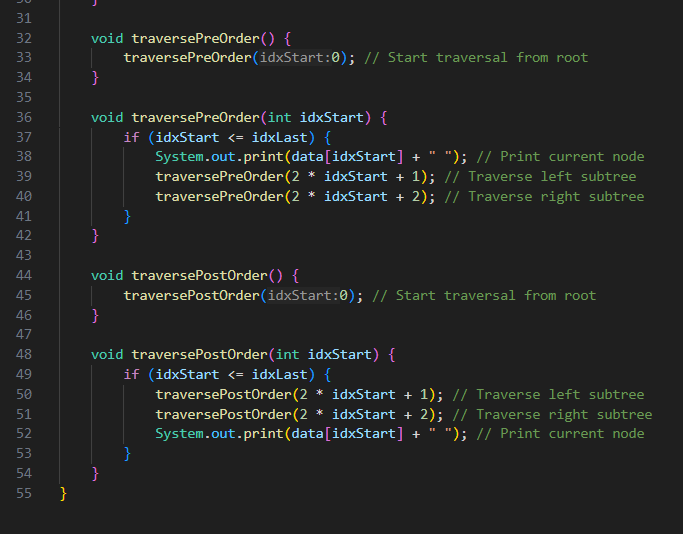
 

* + 1. Modifikasi class **BinaryTreeArray**, dan tambahkan :
* method **add(int data)** untuk memasukan data ke dalam tree





* method **traversePreOrder()** dan **traversePostOrder()**

****