`

# M09 TREE

## 1. TUJUAN

Tujuan Instruksional Umum :

Mampu membuat program menggunakan *Tree* dengan bahasa Java.

Tujuan Instruksional Khusus :

1. Dapat menjelaskan terminologi yang ada pada *Tree*.
2. Dapat memberikan contoh jenis-jenis *Tree.*
3. Dapat membuat *class Tree* menggunakan bahasa Java.
4. Dapat menggunakan class Tree.

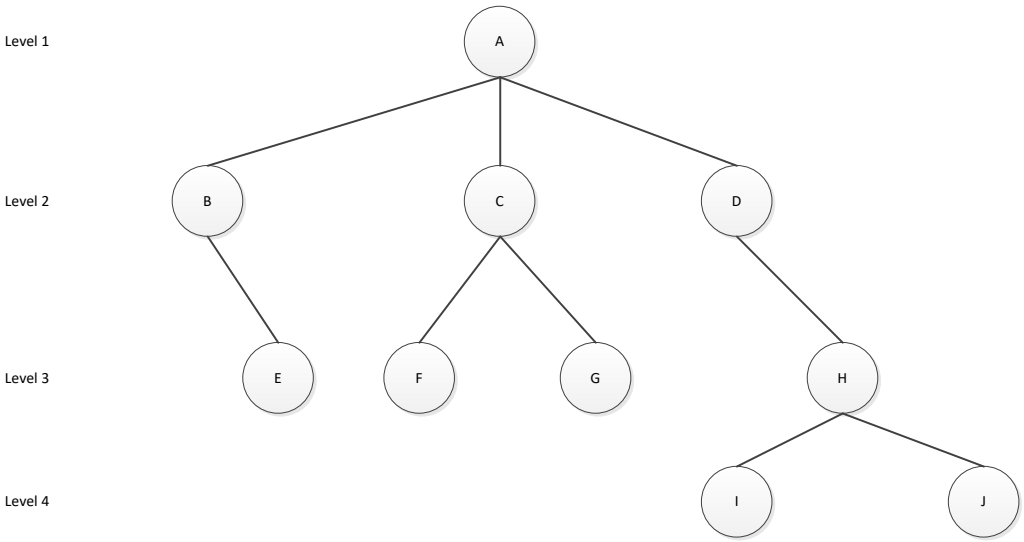
## 2. DURASI WAKTU

2 pertemuan x 4,5 jam

## 3. DASAR TEORI

Tree/pohon merupakan struktur data non linear yang digunakan untuk merepresentasikan hubungan data yang bersifat hierarkis antara elemen-elemennya. Pada tree salah satu elemennya disebut dengan root (akar) dan sisa elemen lain disebut simpul (node/vertex) yang terpecah menjadi sejumlah himpunan yang tidak saling berhubungan satu sama lain, yang disebut subtree/cabang". Perhatikan gambar Tree T berikut :

`




Tree T

Istilah-istilah pada Tree:

- Simpul adalah elemen tree yang berisi informasi / data dan penunjuk pencabangan.
- Tingkat/level suatu simpul ditentukan dari akar (root), sebagai level 1. Apabila simpul dinyatakan sebagai tingkat N, maka simpul-simpul yang merupakan anaknya berada pada tingkat N+1.
- Derajat/degree menyatakan banyaknya anak/turunan di simpul tersebut.
  Contoh : Simpul A memiliki derajat 3 (B,C dan D), simpul yang memiliki derajat 0 (nol) disebut leaf (daun) seperti : E, F, G, I, J
- Tinggi (height) atau kedalaman (depth) suatu tree adalah tingkat maksimum dari level dalam tree tersebut dikurangi 1.
  Contoh dalam tree di atas, mempunyai depth 3.
- Ancestor suatu simpul adalah semua simpul yang terletak dalam satu jalur dengan simpul tersebut, dari akar sampai simpul yang ditinjaunya.
  Contoh Ancestor J adalah A, D dan H
- Predecessor adalah simpul yang berada di atas simpul yang ditinjau.
  Contoh : Predecessor I adalah H.

`

- Successor adalah simpul yang berada di bawah simpul yang ditinjau.
  Contoh : Successor D adalah H.
- Descendant adalah seluruh simpul yang terletak sesudah simpul tertentu dan terletak pada jalur yang sama.
  Contoh : Descendant A adalah B,C dan D. Descendant C adalah F dan G.
- Sibling adalah simpul-simpul yang memiliki parent yang sama dengan simpul yang ditinjau.
  Contoh : Sibling F adalah G
- Parent adalah simpul yang berada satu level di atas simpul yang ditinjau.
  Contoh : Parent I adalah H
- Lintasan (path) adalah urutan akses untuk mendapatkan Node yang ditunjuk yang dimulai dari Akar. Path J adalah A-D-H-J.

**Pohon Biner**

Ciri : Maksimum child adalah 2 (Left Child dan Right Child).

Complete Binary Tree :

Bila semua node kecuali Leaf memiliki 0 atau 2 child.

Skewed Binary Tree (Miring) :

Bila semua node, kecuali Leaf memiliki hanya 1 child

Full Binary Tree :

Bila semua node kecuali Leaf memiliki 2 Child dan semua subtree harus memiliki path yang sama

`

## 4. PERCOBAAN

### PERCOBAAN 1: Membuat Class Tree

1. Buatlah Proyek Baru – Java Application - dengan nama CobaBinaryTree
2. Buatlah class baru dengan nama node.java, source code tampak seperti berikut:

```
public class Node {
    private String data;
    private Node LeftChild;
    private Node RightChild;
    private Node parent;

    public String getData() {
        return data;
    }

    public void setData(String data) {
        this.data = data;
    }

    public Node getLeftChild() {
        return LeftChild;
    }

    public void setLeftChild(Node LeftChild) {
        this.LeftChild = LeftChild;
    }

    public Node getRightChild() {
        return RightChild;
    }

    public void setRightChild(Node RightChild) {
        this.RightChild = RightChild;
    }

    public Node (String data){
        this.data = data;
        LeftChild = null;
        RightChild = null;
    }

    public boolean isGreater(Node compare){
        return this.data.compareTo(compare.getData()) > 0;
    }

}
```

`

3. Buatlah class baru dengan nama BinaryTree.java, dengan source tampak seperti berikut:

```
public class Tree {

    private Node root;

    public Tree(){
        root = null;
    }

    public Tree(Node root){
        this.root = root;
    }

    public Tree(String data){
        this.root = new Node(data);
    }

    public void insert(String data){
        insert(new Node(data));
    }

    public void insert(Node child){
        insert(root, child);
    }

    public void insert(Node parent, Node child){
        if(root == null) {
            root = child;
            System.out.println("Add " + child.getData() + " as
Root");
        }
        else{
            if(child.isGreater(parent)){
                if(parent.getRightChild() == null) {
                    parent.setRightChild(child);
                    System.out.println("Add " + child.getData() +
" RightChild " + parent.getData());
                }
                else insert(parent.getRightChild(), child);
            }
            else {
                if(parent.getLeftChild() == null) {
                    parent.setLeftChild(child);
                    System.out.println("Add " + child.getData() +
" LeftChild " + parent.getData());
                }
                else insert(parent.getLeftChild(), child);
            }
```

`

```
        }
    }


}
```

4. Mencoba Program. Modifikasi method main pada Class CobaBinaryTree sehingga terlihat seperti source berikut:

```
    public static void main(String[] args) {
        Tree pohon = new Tree();
        // String data = "HAKCBDJL";
        Scanner input = new Scanner(System.in);
        System.out.print("String : ");
        String data = input.nextLine();
        for(int          i=0;          i<data.length();          i++)
pohon.insert(String.valueOf(data.charAt(i)));
    }
```

Debuglah program tersebut, perhatikanlah isi dari object Pohon.

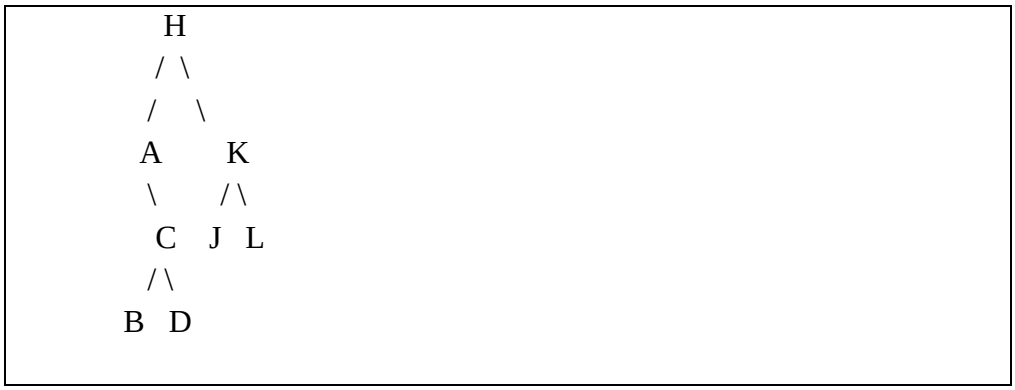
Input string: HAKCBDJL

`

| Iterasi | Isi Object Pohon |
|---|---|
| Iterasi 1 | Add H as Root<br>(ada H) |
| Iterasi 2 | Add A as LeftChild of H<br>(Ada H root dan A anak H) |
| Iterasi 3 | Add K as RightChild of H<br>(Ada H root, dan ada K dan A anak H) |
| Iterasi 4 | Add C as RightChild of A<br>(Ada H root, dan ada K dan A anak H, serta C sebagai anak A) |
| Iterasi 5 | Add B as LeftChild of C<br>(Ada H root, dan ada K dan A anak H, serta C sebagai anak A serta B menjadi anak C) |
| Iterasi 6 | Add D as RightChild of C<br>(Ada H root, dan ada K dan A anak H, serta C sebagai anak A serta B dan D menjadi anak C) |
| Iterasi 7 | Add J as LeftChild of K<br>(Ada H root, dan ada K dan A anak H, serta C sebagai anak A serta B dan D menjadi anak C, serta J menjadi anak K) |

`

| | |
|---|---|
| Iterasi 8 | Add L as RightChild of K<br>(Ada H root, dan ada K dan A anak H, serta C sebagai anak A serta B dan D menjadi anak C, serta J dan L menjadi anak K) |




Gambarkan bentuk pohonnya !

## 5. LATIHAN

Modifikasilah class BinaryTree, tambahkan operasi untuk Finding Node.

`