Modul

6

TREE

Tujuan

Mahasiswa diharapkan dapat menerapkan penggunaan tree pada suatu kasus. Secara khusus program yang mengandung instruksi-instruksi yang menerapkan bagian dari tree. Mahasiswa juga dapat menggunakan bagian dari tree pada kasus yang akan diselesaikan. Di akhir praktikum ini mahasiswa dapat:

- Menjelaskan konsep dasar tree
- Memahami bagaimana penerapan Breadth First Search pada tree
- Memahami bagaimana penerapan Depth First Search pada Tree

Pendahuluan

Praktikum ini mengasumsikan bahwa mahasiswa telah dapat mengoperasikan Netbeans. Mahasiswa juga diharapkan sudah memahami dengan baik materi-materi yang telah diberikan sebelumnya (pemrograman dasar I). Agar mahasiswa dapat mencapai tujuan dalam pertemuan praktikum yang pertama ini, mahasiswa harus memiliki pengetahuan (minimal telah membaca) penggunaan konsep dasar tree. Penerapan Breadth First Search (BFS) dan Depth First Search (DFS) .

Proses

Praktikan membaca buku/diktat yang dilakukan selama 20 menit termasuk membuat ringkasan penting, kemudian berdiskusi sesuai dengan panduan aktifitas yang dilakukan selama 20 menit, sedangkan waktu untuk mengerjakan *project* (membuat program/*coding*) secara individual harus diselesaikan di dalam laboratorium dalam waktu 30 menit. Berikutnya untuk sesi latihan, ada soal tentang pengembangan program yang juga harus diselesaikan di laboratorium dalam waktu 50 menit. Terakhir adalah bagian tugas, yaitu *project* yang dikerjakan dirumah dan wajib dikumpulkan pada pertemuan berikutnya.

Aktifitas

1. Mahasiswa membaca buku ajar (jika ada)/diktat kuliah/materi dari sumber lain tentang konsep tree sebagai materi bab ke enam. Temukan bagian penting dalam tree ini, kemudian tulis sebagai ringkasan hasil belajar.

2. Mahasiswa berdiskusi tentang definisi tree, tujuan adanya parent, distance dan children sebagai bahan diskusi

a. Lengkapilah bagian yang kosong pada class TreeNode di bawah ini sesuai dengan perintah / command yang tertera.

```
    import java.util.ArrayList;

public class TreeNode{
    TreeNode parent;
    double distance;
    ArrayList<TreeNode> children;
    int data;

    public TreeNode(int new_data){
        this.data = new_data;
        this.parent = null;
        this.distance = 0.0;
        this.children = new ArrayList<TreeNode>();
    }

    /* set this node's parent into new parent
     * set this node's distance into distance
     * if this node's parent is not nul, then add this as parent's
child
     */
    void set_parent(TreeNode new_parent, double distance){
        ……...
        ……...
        if(……){
            ……...
        }
    }

    // alias to set_parent(new_parent, 0)
    void set_parent(TreeNode new_parent){
        this.set_parent(……...);
    }

    /* call new_child.set_parent. The new parent of new_child should
be this
     * The distance of new_child should be set to distance
     */
    void add_child(TreeNode new_child, double distance){
        new_child.set_parent(……...);
    }



    /* Simply remove child from this node's children */
    void remove_child(TreeNode child){
```

```
        ……….....
        ……….....
        this.children.remove(child);
    }

    /* print this node's data, this node's distance, and distance +
this node's distance
     * for each of this node's children, recursively call child's
print method
     */
    void print(String spaces, double distance){
        System.out.println(……..... (distance +
this.distance));
        for(int i=0; i<this.children.size(); i++){
            this.children.get(i).print(……..... );
        }
    }

    void print(){
        this.print("", 0);
    }}
```

b. Tulislah code dari class Tree di bawah ini.

```
  public class Tree{
    TreeNode root;

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

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

    void print(){
        if(this.root == null){
            System.out.println();
        }else{
            this.root.print();
        }
    }
}
```

c. Simpan class TreeNode.java dan Tree.java pada satu package yang sama. Selanjutnya buat class baru pada package yang sama ketikkan class Test.java di bawah ini dan jalankan.

```
    public class Test{
    public static void main(String Args[]){
  Tree t = new Tree(new TreeNode(1));
        t.root.add_child(new TreeNode(2), 1);
        t.root.add_child(new TreeNode(3), 1);
        t.root.add_child(new TreeNode(4), 2);
        t.root.children.get(0).add_child(new TreeNode(5), 1);
        t.root.children.get(2).add_child(new TreeNode(6), 1);
        t.root.children.get(2).add_child(new TreeNode(7), 2);
        t.print();
```

```
}
}
```

d. Setelah dijalankan akan muncul output seperti di bawah ini :

```
1 distance from parent : 0.0 distance from initial node : 0.0
2 distance from parent : 1.0 distance from initial node : 1.0
5 distance from parent : 1.0 distance from initial node : 2.0
3 distance from parent : 1.0 distance from initial node : 1.0
4 distance from parent : 2.0 distance from initial node : 2.0
6 distance from parent : 1.0 distance from initial node : 3.0
7 distance from parent : 2.0 distance from initial node : 4.0
```

Kesimpulan

Buatlah interface dari program tree di atas

Penutup

Tugas

Buatlah program yang mengimplementasikan Breadth First Search!
Buatlah program yang mengimplementasikan Depth First Search