Tugas 2

Algoritma dan Struktur Data

Teknik Informatika Kelas C

Nama:

Zhafran Rama Azmi (215150207111025)
Gibran Hakim (215150200111020)

Dosen:

Putra Pandu Adikara, S.Kom., M.Kom.

Program Studi Teknik Informatika
Jurusan Teknik Informatika
Universitas Brawijaya

2022

1. Kode

```
import java.util.Scanner;

// Zhafran Rama Azmi (215150207111025)
// Gibran Hakim (215150200111020)

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        Tree dirTree = new Tree();
        while(in.hasNext()){
            String input = in.next();
            if (input.equals("TAMBAH")) {
                in.next();
                String name = in.next();
                long data = in.nextLong();
                System.out.println("TAMBAH " + name + " PADA " +
dirTree.add(name, data));
            } else if (input.equals("HITUNG")) {
                String path = in.next();
                TreeNode node =  dirTree.findWithPath(path);
                if (node == null) {
                    System.out.println("PATH " + path + " tidak
ditemukan");
                } else {
                    System.out.println("UKURAN " + path + " = " +
dirTree.countValue(node));
                }
            } else if (input.equals("CETAK")) {
                String path = in.next();
                TreeNode node = dirTree.findWithPath(path);
                if (node == null) {
                    System.out.println("PATH " + path + " tidak
ditemukan");
                } else {
                    dirTree.print(node, 0);
                    System.out.println();
                }
            } else if (input.equals("HAPUS")) {
                String path = in.next();
                TreeNode node = dirTree.findWithPath(path);
                if (node == null) {
                    System.out.println("PATH " + path + " tidak ditemukan
untuk dihapus");
                } else {
                    dirTree.delete(node);
                    System.out.println("PATH " + path + " berhasil
```

```
dihapus");
                }
            }
        }
    }
}

class TreeNode {
    TreeNode parent;
    LinkedList child;
    long size;
    String name;

    TreeNode() {
        this.parent=null;
        this.child = new LinkedList();
        this.size = -1;
        this.name = "";
    }

    TreeNode(TreeNode parent, long size, String name) {
        this.parent = parent;
        this.child = new LinkedList();
        this.size = size;
        this.name = name;
    }
}

class Tree {
    TreeNode root;
    int height;

    Tree() {
        this.root = new TreeNode(null, -1, "data");
        this.height = 0;
    }

    String add (String name, long data, TreeNode curNode, int posY) {
        String path = "/" + root.name;
        ListNode p = curNode.child.head;
        while (p!=null) {
            if (name.length() >= p.node.name.length()) {
                if (p.node.name.equals(name.substring(0,
p.node.name.length()))) {
                    path += "/" + p.node.name;
                    curNode = p.node;
                    p = curNode.child.head;
```

```
                posY++;
                continue;
            }
        }
        p = p.next;
    }
    curNode.child.add(name, data, curNode);
    if (this.height <= posY) {
        this.height = posY;
    }
    return path;
}

String add(String name, long data) {
    return this.add(name, data, root, 0);
}

TreeNode findWithPath(String path) {
    String[] nodeName = path.split("/");
    int posY = 1;
    TreeNode p = root;

    if (p.name.equals(nodeName[posY]) && posY == nodeName.length - 1) {
        return p;
    }

    ListNode pList = p.child.head;

    while (pList != null) {
        if (pList.node.name.equals(nodeName[posY + 1])) {
            if (posY + 1 == nodeName.length - 1) {
                return pList.node;
            } else {
                p = pList.node;
                pList = p.child.head;
                posY++;
                continue;
            }
        }
        pList = pList.next;
    }
    return null;
}

void print(TreeNode node, int posY) {
    if (node == root) {
        System.out.println(root.name);
```

```
        } else {
            String indent = "";
            for (int i = 0; i < posY; i++) {
                indent += "--";
            }
            if (indent.length() > 0) {
                System.out.printf("%s %s (%d)\n" , indent , node.name ,
node.size);
            } else {
                System.out.printf("%s%s (%d)\n" , indent , node.name ,
node.size);
            }
        }
        ListNode p = node.child.head;
        while (p != null) {
            print(p.node, posY + 1);
            p = p.next;
        }
    }

    void print() {
        print(root, -1);
    }

    long countValue(TreeNode cur) {
        long tmp = 0;
        if (cur.size > -1) {
            tmp = cur.size;
        }
        ListNode p = cur.child.head;
        while (p != null) {
            tmp += countValue(p.node);
            p = p.next;
        }
        return tmp;
    }

    void delete(TreeNode node) {
        node.parent.child.remove(node.name);
        node.parent = null;
    }
}

class ListNode {
    ListNode next, prev;
    TreeNode node;
```

```
    ListNode() {
        this.next = this.prev = null;
        this.node = null;
    }

    ListNode(TreeNode node) {
        this.node = node;
    }

    ListNode(TreeNode node, ListNode next, ListNode prev) {
        this.node = node;
        this.next = next;
        this.prev = prev;
    }
}

class LinkedList {
    ListNode head, tail;
    int size;

    LinkedList(){
        this.size = 0;
    }

    boolean isEmpty() {
        return size == 0;
    }

    void removeFirst(){
        if(isEmpty()){
            System.out.println("Data is empty !");
        } else{
            ListNode temp = head;
            if(head == tail){
                head = tail = null;
            } else{
                head.next.prev = null;
                head = temp.next;
            }
            size--;
        }
    }

    void removeLast(){
        if(this.isEmpty()){
            System.out.println("Data is empty !");
        } else{
```

```
            ListNode temp = tail;
            if(head == tail){
                head = tail = null;
            } else {
                tail.prev.next = null;
                tail = temp.prev;
            }
            size--;
        }
    }

    void remove(String name){
        ListNode temp = head;
        if(this.isEmpty()){
            System.out.println("Data is empty !");
        } else{
            while(temp != null){
                if(temp.node.name.equals(name)){
                    if(temp == head){
                        this.removeFirst();
                    } else if(temp == tail){
                        this.removeLast();
                    } else{
                        temp.prev.next = temp.next;
                        temp.next.prev = temp.prev;
                        size--;
                    }
                    break;
                }
                temp = temp.next;
            }
        }
    }

    void add(String name,  long val, TreeNode parent){
        ListNode curr = new ListNode(new TreeNode(parent, val, name));
        if(this.isEmpty()){
            head = tail = curr;
        } else{
            tail.next = curr;
            curr.prev = tail;
            tail = curr;
        }
        size++;
    }
}
```

2. Output

```
TAMBAH a PADA /data
TAMBAH b PADA /data
TAMBAH c PADA /data
TAMBAH ba PADA /data/b
TAMBAH bc PADA /data/b
TAMBAH bd PADA /data/b
TAMBAH be PADA /data/b
TAMBAH baa PADA /data/b/ba
TAMBAH bac PADA /data/b/ba
TAMBAH cb PADA /data/c
TAMBAH ce PADA /data/c
data
-- a (1024)
-- b (256)
---- ba (100)
------ baa (500)
------ bac (25)
---- bc (150)
---- bd (125)
---- be (75)
-- c (35)
---- cb (30)
---- ce (50)

c (35)
-- cb (30)
-- ce (50)

UKURAN /data/c = 115
PATH /data/d tidak ditemukan
```

```
➜ cat input0 | java Main.java
TAMBAH a PADA /data
TAMBAH b PADA /data
TAMBAH c PADA /data
TAMBAH ba PADA /data/b
TAMBAH bc PADA /data/b
TAMBAH bd PADA /data/b
TAMBAH be PADA /data/b
TAMBAH baa PADA /data/b/ba
TAMBAH bac PADA /data/b/ba
TAMBAH cb PADA /data/c
TAMBAH ce PADA /data/c
data
-- a (1024)
-- b (256)
---- ba (100)
------ baa (500)
------ bac (25)
---- bc (150)
---- bd (125)
---- be (75)
-- c (35)
---- cb (30)
---- ce (50)

c (35)
-- cb (30)
-- ce (50)

UKURAN /data/c = 115
PATH /data/d tidak ditemukan
```

```
TAMBAH aab PADA /data
TAMBAH a PADA /data
TAMBAH aab PADA /data/aab
TAMBAH aabc PADA /data/aab/aab
data
-- aab (100)
---- aab (300)
------ aabc (300)
-- a (200)

a (200)

UKURAN /data/a = 200
PATH /data/a/ab tidak ditemukan
```

```
UKURAN /data/aab = 700
UKURAN /data/aab/aab = 600
```

Refleksi Diri:

Yang paling sulit saya dan teman kelompok saya hadapi adalah dalam cara berpikir secara rekursif.

```
➜ cat input1 | java Main.java
TAMBAH aab PADA /data
TAMBAH a PADA /data
TAMBAH aab PADA /data/aab
TAMBAH aabc PADA /data/aab/aab
data
-- aab (100)
---- aab (300)
------ aabc (300)
-- a (200)

a (200)

UKURAN /data/a = 200
PATH /data/a/ab tidak ditemukan
UKURAN /data/aab = 700
UKURAN /data/aab/aab = 600
```