POLITEKNIK NEGERI MALANG

TEKNOLOGI INFORMASI

TEKNIK INFORMATIKA



#### Nama : Muhammad Nuril Huda

#### Kelas : TI-1A

#### No : 19

#### Mata Kuliah : Algoritma dan Struktur Data

## 14.2 Kegiatan Praktikum 1

14.2.1 Kode Program

* Kode Program Mahasiswa19

public class Mahasiswa19 {

    String nim;

    String nama;

    String kelas;

    double ipk;

    public Mahasiswa19() {

    }

    public Mahasiswa19(String nim, String nama, String kelas, double ipk) {

        this.nim = nim;

        this.nama = nama;

        this.kelas = kelas;

        this.ipk = ipk;

    }

    public void tampilInformasi() {

        System.out.println("NIM: "+this.nim+" " +

                           "Nama: "+this.nama+" " +

                           "Kelas: "+this.kelas+" " +

                           "IPK: "+this.ipk);

    }

}

* Kode Program Node19

public class Node19 {

    Mahasiswa19 mahasiswa;

    Node19 left, right;

    public Node19() {

    }

    public Node19 (Mahasiswa19 mahasiswa) {

        this.mahasiswa = mahasiswa;

        left = right = null;

    }

}

* Kode Program BinaryTree19

public class BinaryTree19 {

    Node19 root;

    public BinaryTree19 (){

        root = null;

    }

    public boolean isEmpty (){

        return root == null;

    }

    public void add(Mahasiswa19 mahasiswa) {

        Node19 newNode = new Node19(mahasiswa);

        if (isEmpty()) {

            root = newNode;

        } else {

            Node19 current = root;

            Node19 parent = null;

            while(true) {

                parent = current;

                if (mahasiswa.ipk < current.mahasiswa.ipk) {

                    current = current.left;

                    if (current == null) {

                        parent.left = newNode;

                        return;

                    }

                } else {

                    current = current.right;

                    if (current == null) {

                        parent.right = newNode;

                        return;

                    }

                }

            }

        }

    }

boolean find(double ipk) {

        boolean result = false;

        Node19 current = root;

        while (current != null) {

            if (current.mahasiswa.ipk == ipk) {

                result = true;

                break;

            } else if (ipk > current.mahasiswa.ipk) {

                current = current.right;

            } else {

                current = current.left;

            }

        }

        return result;

    }

    void traversePreOrder(Node19 node) {

        if (node != null) {

            node.mahasiswa.tampilInformasi();

            traversePreOrder(node.left);

            traversePreOrder(node.right);

        }

    }

    void traverseInOrder(Node19 node) {

        if (node != null) {

            traverseInOrder(node.left);

            node.mahasiswa.tampilInformasi();

            traverseInOrder(node.right);

        }

    }

    void traversePostOrder(Node19 node) {

        if (node != null) {

            traversePostOrder(node.left);

            traversePostOrder(node.right);

            node.mahasiswa.tampilInformasi();

        }

    }

Node19 getSuccessor (Node19 del){

        Node19 successor = del.right;

        Node19 successorParent = del;

        while (successor.left != null){

            successorParent = successor;

            successor = successor.left;

        }

        if (successor != del.right){

            successorParent.left = successor.right;

            successor.right = del.right;

        }

        return successor;

    }

    void delete(double ipk) {

        if (isEmpty()) {

            System.out.println("Binary tree kosong");

            return;

        }

        //cari node (current) yang akan dihapus

        Node19 parent = root;

        Node19 current = root;

        boolean isLeftChild = false;

        while (current != null) {

            if (current.mahasiswa.ipk == ipk) {

                break;

            } else if (ipk < current.mahasiswa.ipk) {

                parent = current;

                current = current.left;

                isLeftChild = true;

            } else if (ipk > current.mahasiswa.ipk) {

                parent = current;

                current = current.right;

                isLeftChild = false;

            }

        }

//penghapusan

        if (current == null) {

            System.out.println("Data tidak ditemukan");

            return;

        } else {

            //jika tidak ada anak (leaf), maka node dihapus

            if (current.left == null && current.right == null) {

                if (current == root) {

                    root = null;

                } else {

                    if (isLeftChild) {

                        parent.left = null;

                    } else {

                        parent.right = null;

                    }

                }

            } else if (current.left == null) {//jika hanya punya 1 anak (kanan)

                if (current == root) {

                    root = current.right;

                } else {

                    if (isLeftChild) {

                        parent.left = current.right;

                    } else {

                        parent.right = current.right;

                    }

                }

            } else if (current.right == null) { //jika hanya punya 1 anak (kiri)

                if (current == root) {

                    root = current.left;

                } else {

                    if (isLeftChild) {

                        parent.left = current.left;

                    } else {

                        parent.right = current.left;

                    }

                }

} else { //jika punya 2 anak

                Node19 successor = getSuccessor(current);

                System.out.println("Jika 2 anak, current = ");

                successor.mahasiswa.tampilInformasi();

                if (current == root) {

                    root = successor;

                } else {

                    if (isLeftChild) {

                        parent.left = successor;

                    } else {

                        parent.right = successor;

                    }

                }

                successor.left = current.left;

            }

        }

    }

}