# 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;
            }
        }
    }
}
```

- Kode Program BinaryTreeMain19

```
public class BinaryTreeMain19 {
    public static void main(String[] args) {
        BinaryTree19 bst = new BinaryTree19();

        bst.add(new Mahasiswa19("244160170","Ali", "A", 3.57));
        bst.add(new Mahasiswa19("244160221","Badar", "B", 3.85));
        bst.add(new Mahasiswa19("244160185","Candra", "C", 3.21));
        bst.add(new Mahasiswa19("244160220","Dewi", "B", 3.54));

        System.out.println("\nDaftar semua mahasiswa (in order
traversal):");
        bst.traverseInOrder(bst.root);
```

```
System.out.println("\nPencarian data mahasiswa:");

        System.out.print("Cari mahasiswa dengan ipk: 3.54 : ");

        String hasilCari = bst.find(3.54) ?"Ditemukan":"Tidak
ditemukan";

        System.out.println(hasilCari);


        System.out.print("Cari mahasiswa dengan ipk: 3.22 : ");

        hasilCari = bst.find(3.22) ?"Ditemukan":"Tidak ditemukan";

        System.out.println(hasilCari);


        bst.add(new Mahasiswa19("244160131","Devi", "A", 3.72));

        bst.add(new Mahasiswa19("244160205","Ehsan", "B", 3.37));

        bst.add(new Mahasiswa19("244160170","Fizi", "B", 3.46));

        System.out.println("\nDaftar semua mahasiswa setelah
penambahan 3 mahasiswa:");

        System.out.println("InOrder Traversal:");

        bst.traverseInOrder(bst.root);

        System.out.println("PreOrder Traversal:");

        bst.traversePreOrder(bst.root);

        System.out.println("PostOrder Traversal:");

        bst.traversePostOrder(bst.root);


        System.out.println("\nPenghapusan data mahasiswa");

        bst.delete( 3.57);

        System.out.println("\nDaftar semua mahasiswa setelah
penghapusan 1 mahasiswa (in order traversal):");

        bst.traverseInOrder(bst.root);

    }

}
```

14.2.2 Hasil Kode Program

```
Daftar semua mahasiswa (in order traversal):
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160170 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85

Pencarian data mahasiswa:
Cari mahasiswa dengan ipk: 3.54 : Ditemukan
Cari mahasiswa dengan ipk: 3.22 : Tidak ditemukan

Daftar semua mahasiswa setelah penambahan 3 mahasiswa:
InOrder Traversal:
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: B IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160170 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
PreOrder Traversal:
NIM: 244160170 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160205 Nama: Ehsan Kelas: B IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
PostOrder Traversal:
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160205 Nama: Ehsan Kelas: B IPK: 3.37
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM: 244160170 Nama: Ali Kelas: A IPK: 3.57

Penghapusan data mahasiswa
Jika 2 anak, current =
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72

Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal):
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: B IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
```

14.2.3 Pertanyaan

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?
    - Pencarian data dalam binary search tree (BST) lebih efektif dibandingkan binary tree biasa karena BST memiliki aturan penempatan data yang teratur. Dalam BST, setiap nilai di sebelah kiri node selalu lebih kecil, dan setiap nilai di sebelah kanan selalu lebih besar dari node induknya. Dengan aturan ini, kita bisa langsung menentukan harus mencari ke kiri atau ke kanan saat mencari suatu nilai, tanpa harus memeriksa semua node satu per satu. Hal ini

membuat pencarian jadi lebih cepat, terutama jika struktur pohonnya seimbang. Sedangkan pada binary tree biasa, karena data tidak teratur, pencarian harus dilakukan dengan memeriksa setiap node, sehingga bisa memakan waktu lebih lama.

2. Untuk apakah di class Node, kegunaan dari atribut left dan right?
   - Pada class Node19, atribut left dan right digunakan untuk menunjuk ke node anak kiri dan anak kanan. Ini memungkinkan setiap node terhubung dengan node lain membentuk struktur pohon biner, sehingga data dapat disusun dan dicari dengan lebih mudah sesuai aturan binary search tree.
3. a. Untuk apakah kegunaan dari atribut root di dalam class BinaryTree?
   - Atribut root di dalam class BinaryTree19 berfungsi sebagai titik awal atau akar dari pohon. Semua proses seperti penambahan, pencarian, dan penghapusan data dimulai dari root, sehingga root adalah pusat dari seluruh struktur pohon.

   b. Ketika objek tree pertama kali dibuat, apakah nilai dari root?
   - Ketika objek tree pertama kali dibuat, nilai dari root adalah null, yang berarti pohon masih kosong dan belum memiliki data sama sekali.
4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?
   - Ketika tree masih kosong dan akan ditambahkan sebuah node baru, proses yang terjadi adalah node baru tersebut langsung dijadikan sebagai root. Karena belum ada data lain dalam pohon, node pertama otomatis menjadi akar pohon.
5. Perhatikan method add(), di dalamnya terdapat baris program seperti di bawah ini. Jelaskan secara detil untuk apa baris program tersebut?

```
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;
    }
}
```

   - Potongan program pada method add() tersebut digunakan untuk menentukan posisi yang tepat bagi node baru (newNode) dalam struktur binary search tree (BST) berdasarkan nilai IPK mahasiswa. Program akan membandingkan nilai IPK mahasiswa baru dengan node saat ini (current). Jika IPK mahasiswa baru lebih kecil, maka program akan bergerak ke anak kiri (current.left). Jika posisi anak kiri masih kosong (null), maka node baru ditempatkan di sana. Sebaliknya, jika IPK mahasiswa baru lebih besar atau sama, maka program akan bergerak ke anak kanan (current.right). Jika posisi anak kanan kosong, node baru akan ditempatkan di sana. Proses ini akan terus berjalan hingga node baru menemukan tempat yang sesuai, sehingga struktur BST tetap terjaga dan data tetap terurut berdasarkan IPK.
6. Jelaskan langkah-langkah pada method delete() saat menghapus sebuah node yang memiliki dua anak. Bagaimana method getSuccessor() membantu dalam proses ini?

- Ketika menghapus node yang memiliki dua anak, langkah-langkahnya adalah: pertama, cari node yang ingin dihapus. Setelah ketemu, karena node itu punya dua anak, kita tidak bisa langsung menghapusnya. Maka, digunakan method getSuccessor() untuk mencari pengganti yang paling cocok, yaitu node dengan nilai IPK terkecil dari cabang kanan (subtree kanan). Node itu disebut successor. Data node yang dihapus akan diganti dengan data dari successor, lalu successor yang sebenarnya (yang hanya punya satu anak atau tidak punya anak) dihapus dengan cara biasa. Dengan cara ini, struktur pohon tetap rapi dan aturan BST tetap terjaga.

# 14.3 Kegiatan Praktikum 2

14.3.1 Kode Program

- Kode Program BinaryTreeArray

```
public class BinaryTreeArray19 {

    Mahasiswa19[] dataMahasiswa;

    int idxLast;

    public BinaryTreeArray19() {

        this.dataMahasiswa = new Mahasiswa19[10];

    }

    void populateData (Mahasiswa19 dataMhs[], int idxLast) {

        this.dataMahasiswa = dataMhs;

        this.idxLast = idxLast;

    }

    void traverseInOrder(int idxStart) {

        if (idxStart <= idxLast) {

            if (dataMahasiswa[idxStart] != null) {

                traverseInOrder(2*idxStart+1);

                dataMahasiswa[idxStart].tampilInformasi();

                traverseInOrder(2*idxStart+2);

            }

        }

    }

}
```

- Kode Program BinaryTreeArrayMain

```
public class BinaryTreeArrayMain {

    public static void main(String[] args) {

        BinaryTreeArray19 bta = new BinaryTreeArray19();

        Mahasiswa19 mhs1 = new Mahasiswa19("244160121", "Ali", "A",
3.57);

        Mahasiswa19 mhs2 = new Mahasiswa19("244160185", "Candra", "C",
3.41);

        Mahasiswa19 mhs3 = new Mahasiswa19("244160221", "Badar", "B",
3.75);

        Mahasiswa19 mhs4 = new Mahasiswa19("244160220", "Dewi", "B",
3.35);



        Mahasiswa19 mhs5 = new Mahasiswa19("244160131", "Devi", "A",
3.48);

        Mahasiswa19 mhs6 = new Mahasiswa19("244160205", "Ehsan", "D",
3.61);

        Mahasiswa19 mhs7 = new Mahasiswa19("244160170", "Fizi", "B",
3.86);



        Mahasiswa19[] dataMahasiswas =
{mhs1,mhs2,mhs3,mhs4,mhs5,mhs6,mhs7, null, null, null};

        int idxLast = 6;



        bta.populateData(dataMahasiswas,idxLast);

        System.out.println("\nInorder Traversal Mahasiswa: ");

        bta.traverseInOrder(0);

    }

}
```

14.3.2 Hasil Kode Program

```
Inorder Traversal Mahasiswa:
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.35
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.41
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.48
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.61
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.75
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.86
```

14.3.3 Pertanyaan

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class BinaryTreeArray?
   - Atribut data digunakan untuk menyimpan elemen-elemen pohon biner dalam bentuk array, di mana setiap elemen merepresentasikan data dari satu mahasiswa. Sedangkan idxLast menyimpan indeks terakhir dari elemen array yang digunakan, sehingga program tahu sampai mana data mahasiswa yang valid disimpan dalam array tersebut.
2. Apakah kegunaan dari method populateData()?
   - Method populateData() berfungsi untuk mengisi array dataMahasiswa dengan data mahasiswa yang sudah dibuat dan menentukan batas akhir data yang dimasukkan melalui parameter idxLast.
3. Apakah kegunaan dari method traverseInOrder()?
   - Method traverseInOrder() digunakan untuk melakukan traversal inorder pada pohon biner yang disimpan dalam array, yaitu mengunjungi node sebelah kiri dulu, lalu node saat ini, kemudian node sebelah kanan. Ini berguna untuk menampilkan data dalam urutan tertentu.
4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masing-masing?
   - Jika sebuah node berada di indeks 2, maka left child-nya ada di indeks 5 (2×2+1) dan right child-nya ada di indeks 6 (2×2+2).
5. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?
   - Statement int idxLast = 6 menunjukkan bahwa elemen terakhir yang berisi data valid dalam array adalah pada indeks ke-6, yang berarti hanya tujuh data mahasiswa (dari indeks 0 sampai 6) yang digunakan dalam pohon.
6. Mengapa indeks `2*idxStart+1` dan `2*idxStart+2` digunakan dalam pemanggilan rekursif, dan apa kaitannya dengan struktur pohon biner yang disusun dalam array?
   - Indeks 2*idxStart+1 dan 2*idxStart+2 digunakan karena dalam representasi pohon biner menggunakan array, anak kiri suatu node berada di indeks 2i+1 dan anak kanannya di 2i+2 (dengan i adalah indeks parent). Ini adalah rumus umum dalam struktur data pohon biner berbasis array, sehingga memungkinkan traversal dan pencarian anak node tanpa perlu pointer.

# 14.4 Tugas Praktikum

**Jawaban No 1 Sampai 3**

- Kode Program BinaryTree19

```
public void addRekursif(Mahasiswa19 mahasiswa) {
        root = addRekursif(root, mahasiswa);
    }
    Node19 addRekursif(Node19 current, Mahasiswa19 mahasiswa) {
        if (current == null) {
            return new Node19(mahasiswa);
        }
        if (mahasiswa.ipk < current.mahasiswa.ipk) {
            current.left = addRekursif(current.left, mahasiswa);
        } else {
            current.right = addRekursif(current.right, mahasiswa);
        }
        return current;
    }
    public void cariMinIPK() {
        if (isEmpty()) {
            System.out.println("Tree kosong");
            return;
        }
        Node19 current = root;
        while (current.left != null) {
            current = current.left;
        }
        System.out.println("Mahasiswa dengan IPK terkecil:");
        current.mahasiswa.tampilInformasi();
    }
```

```
    public void cariMaxIPK() {
        if (isEmpty()) {
            System.out.println("Tree kosong");
            return;
        }
        Node19 current = root;
        while (current.right != null) {
            current = current.right;
        }
        System.out.println("Mahasiswa dengan IPK terbesar:");
        current.mahasiswa.tampilInformasi();
    }
    public void tampilMahasiswaIPKdiAtas(double ipkBatas) {
        System.out.println("Mahasiswa dengan IPK di atas " + ipkBatas +
":");
        tampilMahasiswaIPKdiAtas(root, ipkBatas);
    }

    private void tampilMahasiswaIPKdiAtas(Node19 node, double ipkBatas) {
        if (node != null) {
            tampilMahasiswaIPKdiAtas(node.left, ipkBatas);
            if (node.mahasiswa.ipk > ipkBatas) {
                node.mahasiswa.tampilInformasi();
            }
            tampilMahasiswaIPKdiAtas(node.right, ipkBatas);
        }
    }
```

- Kode Program BinaryTreeMain19

```
public class BinaryTreeMain19 {
    public static void main(String[] args) {
        BinaryTree19 bst = new BinaryTree19();
        // Tambah data mahasiswa menggunakan addRekursif
        bst.addRekursif(new Mahasiswa19("244160170","Ali", "A", 3.57));
        bst.addRekursif(new Mahasiswa19("244160221","Badar", "B", 3.85));
        bst.addRekursif(new Mahasiswa19("244160185","Candra", "C", 3.21));
        bst.addRekursif(new Mahasiswa19("244160220","Dewi", "B", 3.54));
```

```
        System.out.println("\nDaftar semua mahasiswa (in order
traversal):");

        bst.traverseInOrder(bst.root);

        System.out.println("\nPencarian data mahasiswa:");

        System.out.print("Cari mahasiswa dengan ipk: 3.54 : ");

        String hasilCari = bst.find(3.54) ? "Ditemukan" : "Tidak
ditemukan";

        System.out.println(hasilCari);

        System.out.print("Cari mahasiswa dengan ipk: 3.22 : ");

        hasilCari = bst.find(3.22) ? "Ditemukan" : "Tidak ditemukan";

        System.out.println(hasilCari);

        // Tambah mahasiswa lagi

        bst.addRekursif(new Mahasiswa19("244160131","Devi", "A", 3.72));

        bst.addRekursif(new Mahasiswa19("244160205","Ehsan", "B", 3.37));

        bst.addRekursif(new Mahasiswa19("244160170","Fizi", "B", 3.46));

        System.out.println("\nDaftar semua mahasiswa setelah penambahan 3
mahasiswa:");

        System.out.println("InOrder Traversal:");

        bst.traverseInOrder(bst.root);

        System.out.println("PreOrder Traversal:");

        bst.traversePreOrder(bst.root);

        System.out.println("PostOrder Traversal:");

        bst.traversePostOrder(bst.root);

        // Tampilkan IPK minimum dan maksimum

        System.out.println();

        bst.cariMinIPK();

        bst.cariMaxIPK();

        // Tampilkan mahasiswa dengan IPK di atas 3.50

        System.out.println();

        bst.tampilMahasiswaIPKdiAtas(3.50);

        // Hapus mahasiswa dengan IPK tertentu

        System.out.println("\nPenghapusan data mahasiswa dengan IPK 3.57");

        bst.delete(3.57);

        System.out.println("\nDaftar semua mahasiswa setelah penghapusan
(in order traversal):");

        bst.traverseInOrder(bst.root);

    }
```

- Hasil Kode Program

```
Daftar semua mahasiswa (in order traversal):
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160170 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85

Pencarian data mahasiswa:
Cari mahasiswa dengan ipk: 3.54 : Ditemukan
Cari mahasiswa dengan ipk: 3.22 : Tidak ditemukan

Daftar semua mahasiswa setelah penambahan 3 mahasiswa:
InOrder Traversal:
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: B IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160170 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
PreOrder Traversal:
NIM: 244160170 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160205 Nama: Ehsan Kelas: B IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
PostOrder Traversal:
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160205 Nama: Ehsan Kelas: B IPK: 3.37
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM: 244160170 Nama: Ali Kelas: A IPK: 3.57

Mahasiswa dengan IPK terkecil:
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
Mahasiswa dengan IPK terbesar:
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
```

```
Mahasiswa dengan IPK di atas 3.5:
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160170 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85

Penghapusan data mahasiswa dengan IPK 3.57
Jika 2 anak, current =
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72

Daftar semua mahasiswa setelah penghapusan (in order traversal):
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: B IPK: 3.37
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
```

**Jawaban No 4**

- Kode Program BinaryTreeArray19

```
public class BinaryTreeArray19 {
    Mahasiswa19[] dataMahasiswa;
    int idxLast;
    public BinaryTreeArray19() {
        this.dataMahasiswa = new Mahasiswa19[10];
        this.idxLast = -1;
    }
    void populateData(Mahasiswa19[] dataMhs, int idxLast) {
        this.dataMahasiswa = dataMhs;
        this.idxLast = idxLast;
    }
    void traverseInOrder(int idxStart) {
        if (idxStart <= idxLast) {
            if (dataMahasiswa[idxStart] != null) {
                traverseInOrder(2 * idxStart + 1);
                dataMahasiswa[idxStart].tampilInformasi();
                traverseInOrder(2 * idxStart + 2);
            }
        }
    }
    //Method untuk menambahkan data ke binary tree array
    void add(Mahasiswa19 data) {
        if (idxLast + 1 < dataMahasiswa.length) {
            dataMahasiswa[++idxLast] = data;
        } else {
            System.out.println("Tree is full. Cannot add more data.");
        }
    }
```

```
    //Traversal preorder:
    void traversePreOrder(int idxStart) {
        if (idxStart <= idxLast) {
            if (dataMahasiswa[idxStart] != null) {
                dataMahasiswa[idxStart].tampilInformasi();
                traversePreOrder(2 * idxStart + 1);
                traversePreOrder(2 * idxStart + 2);
            }
        }
    }
}
```

- Kode Program BinaryTreeArrayMain19

```
public class BinaryTreeArrayMain {
    public static void main(String[] args) {
        BinaryTreeArray19 bta = new BinaryTreeArray19();

        bta.add(new Mahasiswa19("244160121", "Ali", "A", 3.57));
        bta.add(new Mahasiswa19("244160185", "Candra", "C", 3.41));
        bta.add(new Mahasiswa19("244160221", "Badar", "B", 3.75));
        bta.add(new Mahasiswa19("244160220", "Dewi", "B", 3.35));
        bta.add(new Mahasiswa19("244160131", "Devi", "A", 3.48));
        bta.add(new Mahasiswa19("244160205", "Ehsan", "D", 3.61));
        bta.add(new Mahasiswa19("244160170", "Fizi", "B", 3.86));

        System.out.println("\nInorder Traversal Mahasiswa:");
        bta.traverseInOrder(0);

        System.out.println("\nPreorder Traversal Mahasiswa:");
        bta.traversePreOrder(0);
    }
}
```

- Hasil Kode Program

```
Inorder Traversal Mahasiswa:
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.35
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.41
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.48
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.61
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.75
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.86

Preorder Traversal Mahasiswa:
NIM: 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.41
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.35
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.48
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.75
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.61
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.86
```

Link Github: https://github.com/nurilhuda05/Algoritma-dan-Struktur-Data/tree/master/Pertemuan14