

Nama: Wahda Adella Putri Febriana

Kelas: 1B / 244107020156

Absen: 24

## Percobaan 1

1. Buat file Mahasiswa24.java

```
public class Mahasiswa24 {
    String nim, nama, kelas;
    double ipk;

    Mahasiswa24(String nama, String nim, String kelas, double ipk) {
        this.nama = nama;
        this.nim = nim;
        this.kelas = kelas;
        this.ipk = ipk;
    }

    void tampilkanInformasi() {
        System.out.println("NIM " + nim + " Nama " + nama + " Kelas " + kelas + " IPK " + ipk);
    }
}
```

2. Buat file Node24.java

```
de24.java > ...
public class Node24 {
    Mahasiswa24 data;
    Node24 left, right;
    Node24(Mahasiswa24 data) {
        this.data = data;
        left = right = null;
    }
}
```

### 3. Buat file BinaryTree24.java

```

public class BinaryTree24 {
    Node24 root;
    BinaryTree24() {
        root = null;
    }
    boolean isEmpty() {
        return root == null;
    }
    void add(Mahasiswa24 data) {
        Node24 newNode = new Node24(data);
        if (isEmpty()) {
            root = newNode;
        } else {
            Node24 current = root;
            Node24 parent = null;
            while (true) {
                parent = current;
                if (data.ipk < current.data.ipk) {
                    current = current.left;
                    if (current == null) {
                        parent.left = newNode;
                        return;
                    }
                } else {
                    current = current.right;
                    if (current == null) {
                        parent.right = newNode;
                        return;
                    }
                }
            }
        }
    }
    void delete(double ipk) {
        if (isEmpty()) {
            System.out.println("Binary tree kosong");
            return;
        }
        Node24 parent = root;
        Node24 current = root;
        boolean isLeftChild = false;
        while (current != null) {
            if (current.data.ipk == ipk) {
                break;
            } else if (ipk < current.data.ipk) {
                parent = current;
                current = current.left;
                isLeftChild = true;
            } else if (ipk > current.data.ipk) {
                parent = current;
                current = current.right;
                isLeftChild = false;
            }
        }
        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
                Node24 successor = getSuccessor(current);
                System.out.println("Jika 2 anak, current = ");
                successor.data.tampilkanInformasi();
                successor.left = current.left;
            }
        }
    }
    boolean find(double ipk) {
        boolean result = false;
        Node24 current = root;
        while (current != null) {
            if (current.data.ipk == ipk) {
                result = true;
                break;
            } else if (ipk > current.data.ipk) {
                current = current.right;
            } else {
                current = current.left;
            }
        }
        return result;
    }
    void traverseInOrder(Node24 node) {
        if (node != null) {
            traverseInOrder(node.left);
            node.data.tampilkanInformasi();
            traverseInOrder(node.right);
        }
    }
    void traversePreOrder(Node24 node) {
        if (node != null) {
            node.data.tampilkanInformasi();
            traversePreOrder(node.left);
            traversePreOrder(node.right);
        }
    }
    void traversePostOrder(Node24 node) {
        if (node != null) {
            traversePostOrder(node.left);
            traversePostOrder(node.right);
            node.data.tampilkanInformasi();
        }
    }
    Node24 getSuccessor(Node24 del) {
        Node24 successor = del.right;
        Node24 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;
    }
}

```

### 4. Buat file BinaryTreeMain24.java

```

public class BinaryTreeMain24 {
    public static void main(String[] args) {
        BinaryTree24 bst = new BinaryTree24();

        bst.add(new Mahasiswa24("nama", "264100121", "nim", "Ali", "kelas", "A", "ipk", 3.7));
        bst.add(new Mahasiswa24("nama", "264100122", "nim", "Bader", "kelas", "B", "ipk", 3.85));
        bst.add(new Mahasiswa24("nama", "264100105", "nim", "Candra", "kelas", "C", "ipk", 3.2));
        bst.add(new Mahasiswa24("nama", "264100220", "nim", "Dewi", "kelas", "D", "ipk", 3.54));

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

        System.out.println("Pencarian data mahasiswa:");
        System.out.print("Carl mahasiswa dengan ipk: 3.54 = ");
        boolean hasilCarl = bst.find(ipk: 3.54);
        System.out.println(hasilCarl);

        System.out.print("Carl mahasiswa dengan ipk: 3.22 = ");
        hasilCarl = bst.find(ipk: 3.22);
        System.out.println(hasilCarl);

        bst.add(new Mahasiswa24("nama", "264100131", "nim", "Dewi", "kelas", "A", "ipk", 3.72));
        bst.add(new Mahasiswa24("nama", "264100132", "nim", "Rahm", "kelas", "B", "ipk", 3.37));
        bst.add(new Mahasiswa24("nama", "264100133", "nim", "Fizal", "kelas", "D", "ipk", 3.46));

        System.out.println("Daftar semua mahasiswa setelah penambahan 3 mahasiswa:");
        bst.traverseInOrder(bst.root);

        System.out.println("InPreOrder Traversal:");
        bst.traversePreOrder(bst.root);

        System.out.println("InPostOrder Traversal:");
        bst.traversePostOrder(bst.root);

        System.out.println("Penghapusan data mahasiswa");
        bst.delete(ipk: 3.57);

        System.out.println("Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal):");
        bst.traverseInOrder(bst.root);
    }
}

```

## 5. Hasil Run

```
Daftar semua mahasiswa (in order traversal):
NIM Candra Nama 244160185 Kelas C IPK 3.21
NIM Dewi Nama 244160220 Kelas B IPK 3.54
NIM Ali Nama 244160121 Kelas A IPK 3.7
NIM Badar Nama 244160122 Kelas B IPK 3.85

Pencarian data mahasiswa:
Cari mahasiswa dengan ipk: 3.54 = true
Cari mahasiswa dengan ipk: 3.22 = false

Daftar semua mahasiswa setelah penambahan 3 mahasiswa:
NIM Candra Nama 244160185 Kelas C IPK 3.21
NIM Ehsan Nama 244160132 Kelas B IPK 3.37
NIM Fizzal Nama 244160133 Kelas D IPK 3.46
NIM Dewi Nama 244160220 Kelas B IPK 3.54
NIM Ali Nama 244160121 Kelas A IPK 3.7
NIM Dewi Nama 244160131 Kelas A IPK 3.72
NIM Badar Nama 244160122 Kelas B IPK 3.85

PreOrder Traversal:
NIM Ali Nama 244160121 Kelas A IPK 3.7
NIM Candra Nama 244160185 Kelas C IPK 3.21
NIM Dewi Nama 244160220 Kelas B IPK 3.54
NIM Ehsan Nama 244160132 Kelas B IPK 3.37
NIM Fizzal Nama 244160133 Kelas D IPK 3.46
NIM Badar Nama 244160122 Kelas B IPK 3.85
NIM Dewi Nama 244160131 Kelas A IPK 3.72

PostOrder Traversal:
NIM Fizzal Nama 244160133 Kelas D IPK 3.46
NIM Ehsan Nama 244160132 Kelas B IPK 3.37
NIM Dewi Nama 244160220 Kelas B IPK 3.54
NIM Candra Nama 244160185 Kelas C IPK 3.21
NIM Dewi Nama 244160131 Kelas A IPK 3.72
NIM Badar Nama 244160122 Kelas B IPK 3.85
NIM Ali Nama 244160121 Kelas A IPK 3.7
```

```
Penghapusan data mahasiswa
Data tidak ditemukan
```

```
Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal):
NIM Candra Nama 244160185 Kelas C IPK 3.21
NIM Ehsan Nama 244160132 Kelas B IPK 3.37
NIM Fizzal Nama 244160133 Kelas D IPK 3.46
NIM Dewi Nama 244160220 Kelas B IPK 3.54
NIM Ali Nama 244160121 Kelas A IPK 3.7
NIM Dewi Nama 244160131 Kelas A IPK 3.72
NIM Badar Nama 244160122 Kelas B IPK 3.85
```

## Pertanyaan

1. Karena dalam **Binary Search Tree (BST)**, data tersusun secara terurut:

- Node kiri selalu lebih kecil dari node induk.
- Node kanan selalu lebih besar dari node induk.

Dengan struktur ini, proses pencarian hanya perlu **membandingkan dan berpindah ke kiri atau kanan** secara logaritmik, sehingga kompleksitas waktu rata-rata adalah  **$O(\log n)$** .

Sementara pada binary tree biasa, pencarian bisa memakan waktu  **$O(n)$**  karena tidak ada aturan urutan data, sehingga harus menelusuri seluruh node.

2. **left**: Menunjuk ke **anak kiri** dari node tersebut. Berisi data yang lebih kecil dari node induk.

**right**: Menunjuk ke **anak kanan** dari node tersebut. Berisi data yang lebih besar dari node induk.

3. A. root digunakan sebagai **titik awal** atau **node pertama** dari pohon biner. Semua operasi seperti `add()`, `find()`, `delete()`, dan `traverse()` dimulai dari node ini.  
B. Saat pertama kali objek tree dibuat, nilai root adalah **null** karena pohon masih kosong dan belum ada data yang ditambahkan.

4. Jika tree kosong (`root == null`), maka:

- Node baru akan **langsung menjadi root**.
- Tidak perlu traversal karena tidak ada node lain.

5. Penjelasan:

- Jika `mahasiswa.ipk` lebih kecil dari `current.ipk`, maka pindah ke anak kiri (`current = current.left`).

- Jika `current.left` masih kosong (`null`), maka node baru (`newNode`) dimasukkan ke posisi tersebut (`parent.left = newNode`) dan proses berhenti (`return`).
- Kode ini menunjukkan **proses rekursif pencarian tempat kosong di anak kiri** untuk menyisipkan node baru pada Binary Search Tree.

#### 6. Langkah-langkah `delete()` untuk node dengan 2 anak:

- Cari **successor** dari node yang akan dihapus, yaitu node **paling kecil dari subtree kanan**.
- Gantikan node yang ingin dihapus dengan successor.
- Pastikan `successor.left` diisi dengan `current.left` (anak kiri dari node yang dihapus).
- Jika successor bukan anak langsung dari node yang dihapus, maka pindahkan `successor.right` ke posisi successor sebelumnya.

#### Peran `getSuccessor()`:

- Mengembalikan node yang paling layak menggantikan node yang dihapus agar **sifat BST tetap terjaga**.
- Menjamin bahwa struktur dan aturan BST tetap valid setelah penghapusan node yang punya 2 anak.

## Praktikum 2

### 1. Buat file `BinaryTreeArray24.java`

```
BinaryTreeArray24.java > ...
public class BinaryTreeArray24 {
    String[] data = new String[10];
    int idxLast = -1;

    public void populateData() {
        data[0] = "Ali";
        data[1] = "Budi";
        data[2] = "Citra";
        data[3] = "Dina";
        data[4] = "Eka";
        idxLast = 4;
    }

    public void traverseInOrder(int index) {
        if (index <= idxLast && data[index] != null) {
            traverseInOrder(2 * index + 1);
            System.out.print(data[index] + " ");
            traverseInOrder(2 * index + 2);
        }
    }
}
```

2. Buat file BlnaryTreeArrayMain24.java

```
public class BinaryTreeArrayMain24 {  
    Run | Debug  
    public static void main(String[] args) {  
        BinaryTreeArray24 tree = new BinaryTreeArray24();  
        tree.populateData();  
        System.out.println(x:"InOrder Traversal:");  
        tree.traverseInOrder(index:0);  
    }  
}
```

3. Hasil Run

```
ArrayMain24'  
InOrder Traversal:  
Dina Budi Eka Ali Citra  
D:\Project\Kuliah\SEM3\Pengetikuan AFD\Jahcheat14\
```