POLITEKNIK NEGERI MALANG

TEKNOLOGI INFORMASI

TEKNIK INFORMATIKA

Nama: Abdul Rahman Hanif Darmawan

NIM: 244107020232

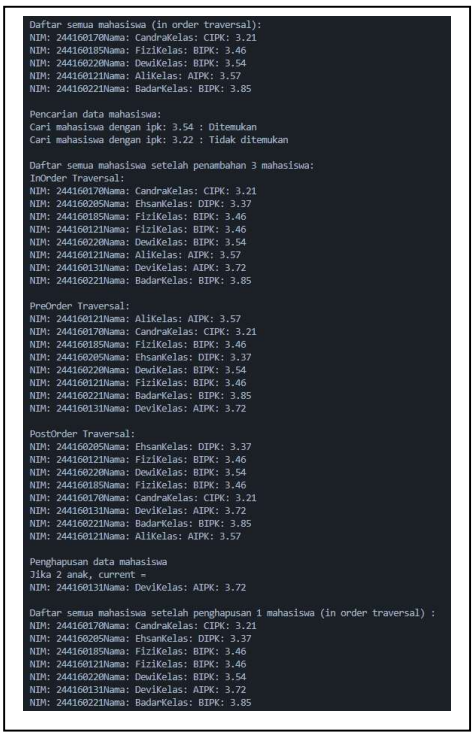
Kelas: TI-1A

Prodi: D4-TEKNIK INFORMATIKA

JOBSHEET 14
Tree

14.2.1. Verifikasi Hasil Percobaan

-



```
Daftar semua mahasiswa (in order traversal):
NIM: 244160170Nama: CandraKelas: CIPK: 3.21
NIM: 244160185Nama: FiziKelas: BIPK: 3.46
NIM: 244160220Nama: DewiKelas: BIPK: 3.54
NIM: 244160121Nama: AliKelas: AIPK: 3.57
NIM: 244160221Nama: BadarKelas: BIPK: 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: 244160170Nama: CandraKelas: CIPK: 3.21
NIM: 244160205Nama: EhsanKelas: DIPK: 3.37
NIM: 244160185Nama: FiziKelas: BIPK: 3.46
NIM: 244160121Nama: FiziKelas: BIPK: 3.46
NIM: 244160220Nama: DewiKelas: BIPK: 3.54
NIM: 244160121Nama: AliKelas: AIPK: 3.57
NIM: 244160131Nama: DeviKelas: AIPK: 3.72
NIM: 244160221Nama: BadarKelas: BIPK: 3.85

PreOrder Traversal:
NIM: 244160121Nama: AliKelas: AIPK: 3.57
NIM: 244160170Nama: CandraKelas: CIPK: 3.21
NIM: 244160185Nama: FiziKelas: BIPK: 3.46
NIM: 244160205Nama: EhsanKelas: DIPK: 3.37
NIM: 244160220Nama: DewiKelas: BIPK: 3.54
NIM: 244160121Nama: FiziKelas: BIPK: 3.46
NIM: 244160221Nama: BadarKelas: BIPK: 3.85
NIM: 244160131Nama: DeviKelas: AIPK: 3.72

PostOrder Traversal:
NIM: 244160205Nama: EhsanKelas: DIPK: 3.37
NIM: 244160121Nama: FiziKelas: BIPK: 3.46
NIM: 244160220Nama: DewiKelas: BIPK: 3.54
NIM: 244160185Nama: FiziKelas: BIPK: 3.46
NIM: 244160170Nama: CandraKelas: CIPK: 3.21
NIM: 244160131Nama: DeviKelas: AIPK: 3.72
NIM: 244160221Nama: BadarKelas: BIPK: 3.85
NIM: 244160121Nama: AliKelas: AIPK: 3.57

Penghapusan data mahasiswa
Jika 2 anak, current =
NIM: 244160131Nama: DeviKelas: AIPK: 3.72

Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal) :
NIM: 244160170Nama: CandraKelas: CIPK: 3.21
NIM: 244160205Nama: EhsanKelas: DIPK: 3.37
NIM: 244160185Nama: FiziKelas: BIPK: 3.46
NIM: 244160121Nama: FiziKelas: BIPK: 3.46
NIM: 244160220Nama: DewiKelas: BIPK: 3.54
NIM: 244160131Nama: DeviKelas: AIPK: 3.72
NIM: 244160221Nama: BadarKelas: BIPK: 3.85
```

14.2.2. Pertanyaan

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?
    - Karena dalam BST, setiap node diurutkan (left < parent < right), sehingga proses pencarian bisa menggunakan metode pencarian biner (divide and conquer), lebih cepat dibanding binary tree biasa yang tidak terurut.
2. Untuk apakah di class Node, kegunaan dari atribut left dan right?
    - left dan right digunakan untuk menunjuk ke anak kiri dan anak kanan dalam struktur pohon biner.

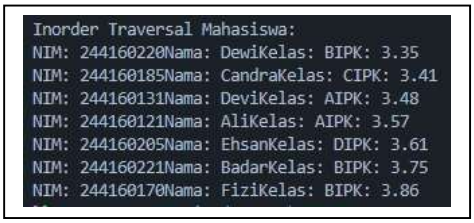
3. A. Untuk apakah kegunaan dari atribut root di dalam class BinaryTree?
   B. Ketika objek tree pertama kali dibuat, apakah nilai dari root?
   - A. root digunakan sebagai titik awal (anchor) dalam tree untuk operasi seperti pencarian, penambahan, dan penghapusan.
   - B. Ketika objek tree pertama kali dibuat, nilai **root** adalah null (belum ada node).
4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?
   - Jika tree masih kosong (root == null), maka node baru akan menjadi root dari 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;
    }
}
```

   - Baris tersebut memeriksa apakah IPK mahasiswa yang ingin ditambahkan lebih kecil daripada IPK node saat ini. Jika ya, maka pencarian berlanjut ke anak kiri (left).
6. Jelaskan langkah-langkah pada method delete() saat menghapus sebuah node yang memiliki dua anak. Bagaimana method getSuccessor() membantu dalam proses ini?
   - 1. Cari node yang akan dihapus.
     2. Jika node adalah leaf, hapus langsung.
     3. Jika node memiliki satu anak, gantikan node dengan anaknya.
     4. Jika node memiliki dua anak, cari successor (nilai terkecil di subtree kanan).
     5. Gantikan node dengan successor.
   - getSuccessor() membantu mencari node pengganti (successor) yang akan menggantikan node yang dihapus agar BST tetap terurut.

14.3.1 Verifikasi Hasil Percobaan

-



14.3.2 Pertanyaan Percobaan

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class BinaryTreeArray?
   - Atribut data menyimpan array objek Mahasiswa02 yang berisi data mahasiswa di setiap node binary tree. Atribut idxLast menyimpan indeks terakhir node dalam array yang digunakan untuk traversal agar tidak melewati batas.
2. Apakah kegunaan dari method populateData()?
   - Method populateData() digunakan untuk mengisi data binary tree dengan array Mahasiswa02 sekaligus mengatur nilai idxLast.
3. Apakah kegunaan dari method traverseInOrder()?
   - Method traverseInOrder() digunakan untuk melakukan penelusuran in-order secara rekursif pada binary tree yang disimpan dalam array.
4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masing-masing?
   - Left child = 2 × 2 + 1 = indeks 5, Right child = 2 × 2 + 2 = indeks 6.
5. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?
   - Statement int idxLast = 6 menunjukkan bahwa node terakhir yang terisi dalam array berada di indeks 6 (indeks 0-6).
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` (left child) dan `2*idxStart+2` (right child) digunakan untuk merepresentasikan struktur pohon biner dalam array sesuai dengan urutan level, agar traversal dapat dilakukan tanpa pointer secara eksplisit.

14.4. Tugas Praktikum

1. Buat method di dalam class BinaryTree00 yang akan menambahkan node dengan cara rekursif (addRekursif()).

```
public void addRekursif(Mahasiswa02 mahasiswa) {
    root = tambahRekursif(root, mahasiswa);
}

private Node02 tambahRekursif(Node02 current, Mahasiswa02 mahasiswa) {
    if (current == null) {
        return new Node02(mahasiswa);
    }
    if (mahasiswa.ipk < current.mahasiswa.ipk) {
        current.left = tambahRekursif(current.left, mahasiswa);
    } else {
        current.right = tambahRekursif(current.right, mahasiswa);
    }
    return current;
}
```

2. Buat method di dalam class BinaryTree00 untuk menampilkan data mahasiswa dengan IPK paling kecil dan IPK yang paling besar (cariMinIPK() dan cariMaxIPK()) yang ada di dalam binary search tree

```
public void cariMinIPK() {
    if (root == null) {
        System.out.println("Tree kosong.");
    } else {
        Node02 current = root;
        while (current.left != null) {
            current = current.left;
        }
        System.out.println("Mahasiswa dengan IPK terkecil:");
        current.mahasiswa.tampilInformasi();
    }
}

public void cariMaxIPK() {
    if (root == null) {
        System.out.println("Tree kosong.");
    } else {
        Node02 current = root;
        while (current.right != null) {
            current = current.right;
        }
        System.out.println("Mahasiswa dengan IPK terbesar:");
        current.mahasiswa.tampilInformasi();
    }
}
```

3. Buat method dalam class BinaryTree00 untuk menampilkan data mahasiswa dengan IPK di atas suatu batas tertentu, misal di atas 3.50 (tampilMahasiswaIPKdiAtas(double ipkBatas)) yang ada di dalam binary search tree.

```
public void tampilMahasiswaIPKdiAtas(double ipkBatas) {
    tampilMahasiswaIPKdiAtasRekursif(root, ipkBatas);
}

private void tampilMahasiswaIPKdiAtasRekursif(Node02 node, double ipkBatas) {
    if (node != null) {
        tampilMahasiswaIPKdiAtasRekursif(node.left, ipkBatas);
        if (node.mahasiswa.ipk > ipkBatas) {
            node.mahasiswa.tampilInformasi();
        }
        tampilMahasiswaIPKdiAtasRekursif(node.right, ipkBatas);
    }
}
```

4. Modifikasi class BinaryTreeArray00 di atas, dan tambahkan :
   - method add(Mahasiswa data) untuk memasukan data ke dalam binary tree
   - method traversePreOrder()

```
void add(Mahasiswa02 data) {
    if (idxLast >= dataMahasiswa.length - 1) {
        System.out.println("Array penuh, tidak dapat menambahkan data.");
    } else {
        idxLast++;
        dataMahasiswa[idxLast] = data;
    }
}

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

Output BinaryTreeMain02.java

```
Inorder Traversal Mahasiswa:
NIM: 244160220Nama: DewiKelas: BIPK: 3.35
NIM: 244160185Nama: CandraKelas: CIPK: 3.41
NIM: 244160121Nama: AliKelas: AIPK: 3.57
NIM: 244160221Nama: BadarKelas: BIPK: 3.75

Mahasiswa dengan IPK Paling Kecil:
Mahasiswa dengan IPK terkecil:
NIM: 244160220Nama: DewiKelas: BIPK: 3.35

Mahasiswa dengan IPK Paling Besar:
Mahasiswa dengan IPK terbesar:
NIM: 244160221Nama: BadarKelas: BIPK: 3.75

Mahasiswa dengan IPK di Atas 3.50:
NIM: 244160121Nama: AliKelas: AIPK: 3.57
NIM: 244160221Nama: BadarKelas: BIPK: 3.75
```

Output BinaryTreeArrayMain02.java

```
Inorder Traversal Mahasiswa:
NIM: 244160220Nama: DewiKelas: BIPK: 3.35
NIM: 244160185Nama: CandraKelas: CIPK: 3.41
NIM: 244160131Nama: DeviKelas: AIPK: 3.48
NIM: 244160121Nama: AliKelas: AIPK: 3.57
NIM: 244160205Nama: EhsanKelas: DIPK: 3.61
NIM: 244160221Nama: BadarKelas: BIPK: 3.75
NIM: 244160170Nama: FiziKelas: BIPK: 3.86

Preorder Traversal Mahasiswa:
NIM: 244160121Nama: AliKelas: AIPK: 3.57
NIM: 244160185Nama: CandraKelas: CIPK: 3.41
NIM: 244160220Nama: DewiKelas: BIPK: 3.35
NIM: 244160131Nama: DeviKelas: AIPK: 3.48
NIM: 244160221Nama: BadarKelas: BIPK: 3.75
NIM: 244160205Nama: EhsanKelas: DIPK: 3.61
NIM: 244160170Nama: FiziKelas: BIPK: 3.86
```

Link Github

https://github.com/baynobu/ALSD/tree/87f44ec70afc264660335ae6d406083359e7ba73/Pertemuan%2015