VERSION 1.1

JANUARI 6, 2025

[STRUKTUR DATA]

MODUL 5, TREE

DISUSUN OLEH:

MAYLANI KUSUMA WARDHANI
MUHAMMAD HISYAM KAMIL

DIAUDIT OLEH:
MUHAMMAD ILHAM PERDANA. S.TR.T., M.T.

LAB. INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

[STRUKTUR DATA]

PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi sebelum mengerjakan tugas, materi yang tercakup antara lain:

1. Tree
2. Binary Tree
3. Binary Search Tree

TUJUAN

Mahasiswa mampu menguasai dan menjelaskan konsep dari Struktur Data Tree.

TARGET MODUL

Mahasiswa mampu memahami dan menerapkan Tree beserta contohnya:

PERSIAPAN SOFTWARE/APLIKASI

1. Java Development Kit
2. Java Runtime Environment
3. IDE (Intellij IDEA, Eclipse, Netbeans, dll)

REFERENSI MATERI

Geekgorgeeks:

https://www.geeksforgeeks.org/introduction-to-binary-tree-data-structure-and-algorithm-tutorials/

Youtube (Indonesia):

https://www.youtube.com/watch?v=YISN5L-7Nz0&ab_channel=IsnaAlfi

Youtube (English):

https://www.youtube.com/watch?v=FxaZu9KeKyI&ab_channel=GreatLearning

Javapoint:

https://www.javatpoint.com/tree

Edureka:

https://www.edureka.co/blog/java-binary-tree

Note: Dari referensi tersebut mungkin terdapat sedikit perbedaan satu sama yang lain, cukup pahami konsepnya dan terapkan pada kasus di modul ini.

MATERI POKOK

Mengapa belajar Tree dan Binary Tree:

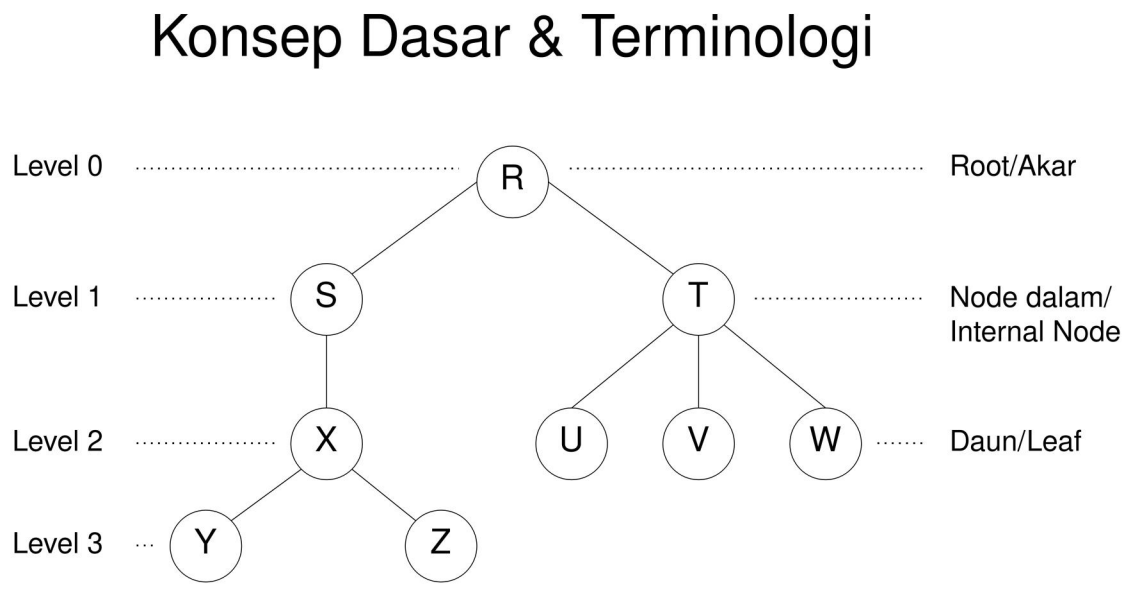
Sebelum masuk ke materi, kita harus mengetahui tujuan belajar kita, tujuan kita mempelajari Tree dan Binary Tree adalah untuk memperdalam struktur data kita, Binary Tree dan Tree banyak diimplementasikan dalam beberapa algoritma salah satunya seperti Binary Sort dan lain-lainnya. Struktur data ini juga banyak diimplementasikan pada aplikasi Backend. Selain itu jenis struktur data ini sering sekali dijadikan studi kasus dalam Technical Interview ketika kita ingin bekerja pada suatu Perusahaan khususnya untuk posisi Backend Developer.

Penjelasan:

1. Tree

Tree merupakan salah satu bentuk struktur data tidak linear yang menggambarkan hubungan yang bersifat hierarki antara elemen-elemen. Tree didefinisikan sebagai kumpulan simpul (node) dengan salah satu simpul yang dijadikan akar (root). Simpul lainnya terbagi menjadi himpunan yang saling tak berhubungan satu sama lain (subtree).

Node-node tersebut dihubungkan oleh sebuah vector. Setiap node dapat memiliki 0 atau lebih node anak (child). Sebuah node yang memiliki node anak disebut node induk (parent). Sebuah node anak hanya memiliki satu node induk. Sesuai konvensi ilmu komputer, tree tumbuh ke bawah, tidak seperti pohon di dunia nyata yang tumbuh ke atas. Dengan demikian, node anak akan digambarkan berada dibawah node induknya. Berikut adalah contoh ilustrasi bentuk tree.



Gambar 1. Konsep Tree

Istilah Pada Tree

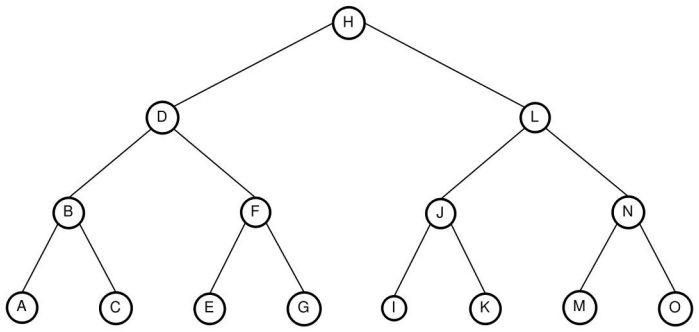
Istilah	Keterangan
Predecessor	Node yang berada di atas node tertentu
Successor	Node yang berada di bawah tertentu
Ancestor	Seluruh node yang terletak sebelum node tertentu dan terletak pada jalur yang sama
Descendant	Seluruh node yang terletak setelah node tertentu dan terletak pada jalur yang sama
Parent	Predecessor satu level di atas suatu node
Child	Successor satu level di bawah suatu node
Sibling	Node-node yang memiliki parent yang sama
Subtree	Suatu node beserta descendant-nya
Size	Banyaknya node dalam suatu tree
Height	Banyaknya tingkatan dalam suatu tree
Root	Node khusus yang tidak memiliki predecessor
Leaf	Node-node dalam tree yang tidak memiliki successor
Degree	Banyaknya child dalam suatu

2. Binary Tree

Binary Tree adalah tree dengan syarat bahwa tiap node hanya boleh memiliki maksimal 2 subtree dan kedua subtree tersebut harus dipisah. Sesuai dengan definisi tersebut, maka tiap node dalam Binary Tree hanya boleh memiliki paling banyak dua child. Berikut adalah jenis-jenis Binary Tree:

a. Full Binary Tree

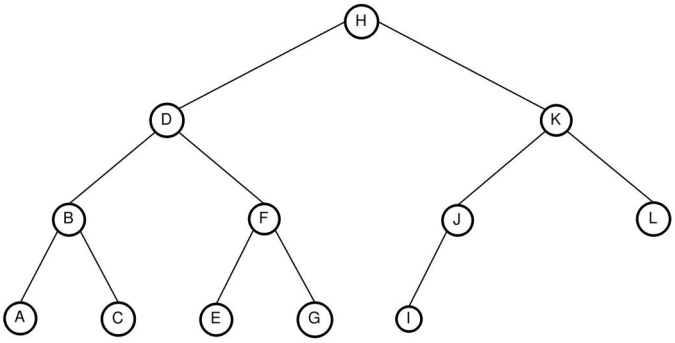
Merupakan Binary Tree yang tiap node-nya (kecuali leaf) memiliki dua child dan tiap subtree harus mempunyai Panjang path yang sama. Berikut contohnya:



Gambar 2. Visualisasi Full Binary Tree

b. Complete Binary Tree

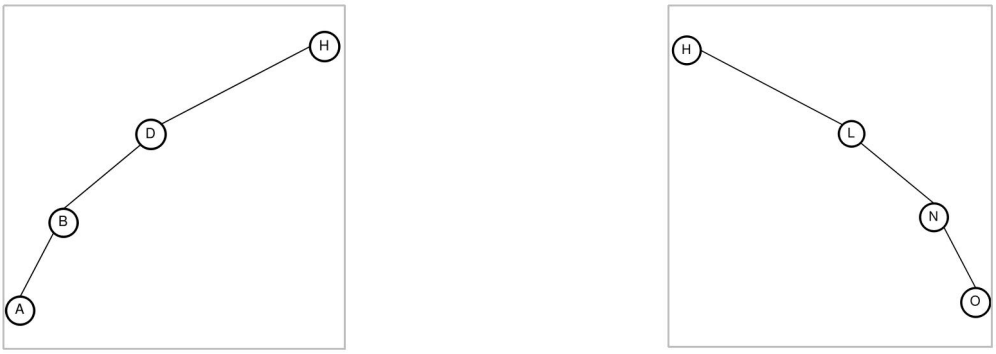
Complete Binary Tree ini hampir sama dengan Full Binary Tree, namun tiap subtree boleh memiliki panjang path yang berbeda. Node kecuali leaf memiliki 0 atau 2 child. Berikut contohnya:



Gambar 3. Visualisasi Complete Binary Tree

c. Skewed Binary Tree

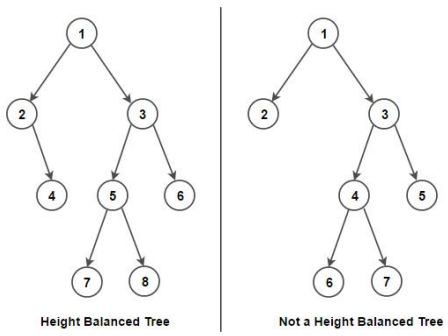
Merupakan Binary Tree yang semua node-nya (kecuali leaf) hanya memiliki satu child. Berikut contohnya:



Gambar 3. Visualisasi Skewed Binary Tree

d. Balanced Binary Tree

Balanced Binary Tree merupakan tree yang memiliki perbedaan jumlah node pada subtree kiri dan kanannya maksimal satu atau tingginya maksimal harus berselisih satu saja. Berikut contohnya:

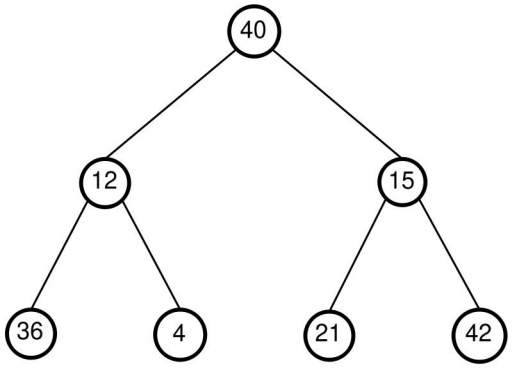


Gambar 4. Visualisasi Height Balanced Tree dan Not a Height Binary Tree

3. Binary Search Tree

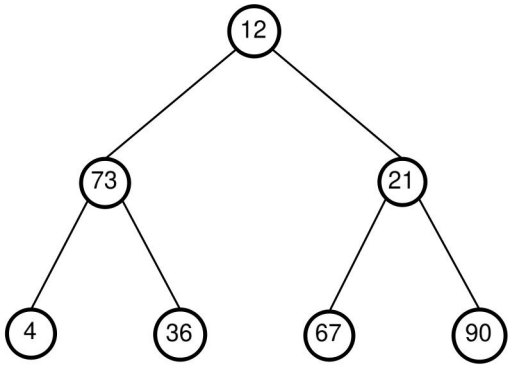
Binary Search Tree merupakan jenis khusus dari Binary Tree yang menyimpan data dalam urutan terurut. Binary Search Tree akan memudahkan pencarian, penambahan dan penghapusan data. Ciri khas yang ada dalam Binary Search ada pada di setiap node-nya. Semua nilai di subtree kanan pasti lebih besar dari nilai node di subtree kiri. Binary Search Tree sangat berguna dalam aplikasi yang memerlukan pencarian data yang cepat dan efisien. Dibawah ini merupakan gambar perbedaan diantara Binary Tree dengan Binary Search Tree.

Binary Tree :



Gambar 5. Visualisasi Binary Tree

Binary Search Tree :



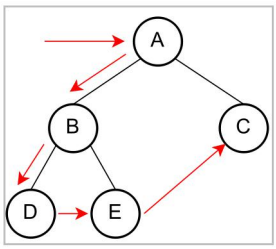
Gambar 6. Visualisasi Binary Search Tree

4. Binary Tree Traversal

Binary Tree Traversal adalah proses mengunjungi node tepat satu kali dan tiap node hanya boleh memiliki maksimal 2 subtree yang disebut sub pohon kiri (left subtree) dan sub pohon kanan (right subtree). Dengan melakukan kunjungan secara lengkap, maka akan didapatkan

urutan informasi secara linier yang tersimpan dalam sebuah Binary Tree. Terdapat 3 metode dalam Binary Tree Traversal, yaitu:

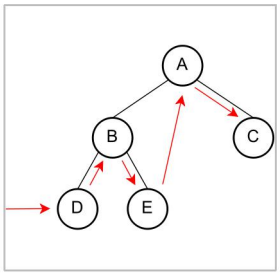
A. PreOrder



Urutan PreOrder:

- Cetak isi simpul yang dikunjungi (root)
- Kunjungi cabang kiri
- Kunjungi cabang kanan

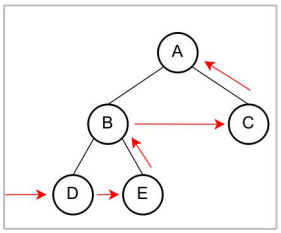
B. InOrder



Urutan InOrder:

- Kunjungi cabang kiri
- Cetak isi simpul yang dikunjungi (root)
- Kunjungi cabang kanan

C. PostOrder



Urutan PostOrder:

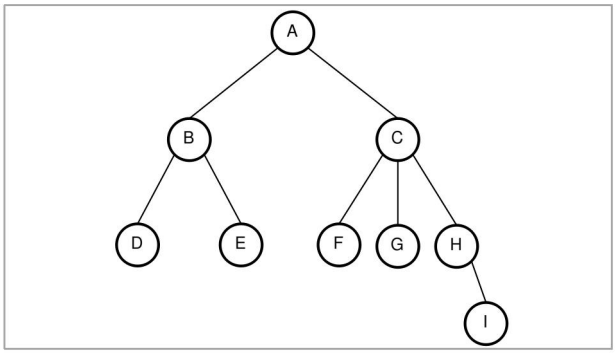
- Kunjungi cabang kiri
- Kunjungi cabang kanan
- Cetak isi simpul yang dikunjungi (root)

Kenapa Menggunakan Tree?

Tree merupakan teori yang sangat bermanfaat dalam struktur data karena dapat digunakan sebagai struktur dalam penyimpanan data yang baik dalam berbagai kasus. Dapat digunakan untuk menyimpan dan mencari data dalam teknik pemrograman serta bagaimana cara struktur data Tree dapat dimanfaatkan untuk menyimpan dan mencari data dengan cepat dan efisien serta mengurangi bug dalam komputer.

Contoh dan Istilah Dalam Binary Tree:

Disediakan gambar tree seperti berikut:



Gambar 7. Visualisasi Tree

Dengan melihat gambar diatas, dapat diketahui bahwa:

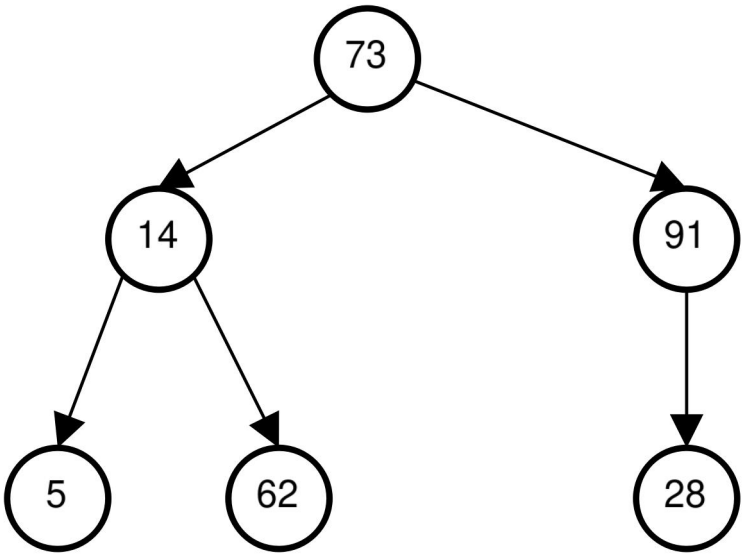
a. Predecessor (F) = **A, B, C**
b. Successor (B) = **D, E, F, G, H, I**
c. Ancestor (F) = **C, A**
d. Descendant (B) = **D, E**
e. Parent (I) = **H**
f. Child (C) = **F, G, H**
g. Sibling (G) = **F, H**
h. Degree (C) = **3**
i. Height = **4**
j. Root = **A**
k. Leaf = **D, E, F, G, I**
l. Size = **9**

*Jika dilihat dari soal diatas, yang menjadi node patokan adalah didalam tanda kurung ().

CODELAB

LATIHAN 1: Membuat Code Dari Binary Tree

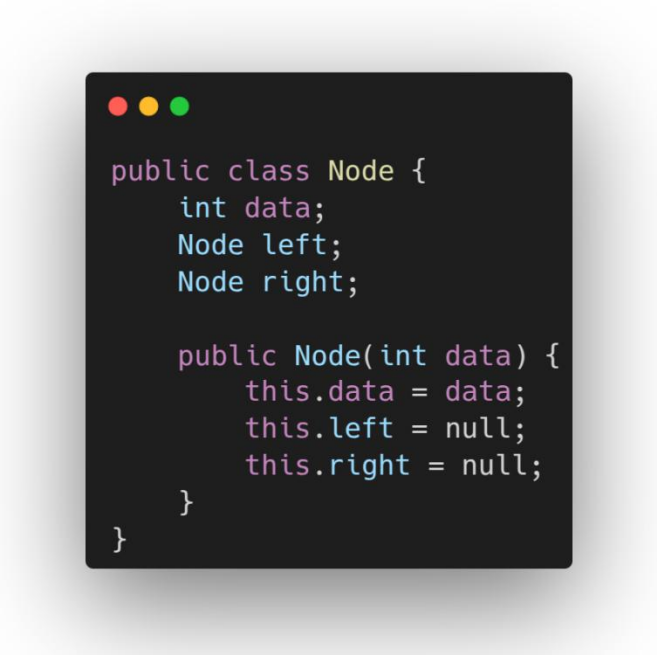
Membuat Code Binary Tree Seperti Gambar Dibawah Ini



Gambar 8. Visualisasi Latihan 1 Binary Tree

Selanjutnya Membuat Tree Traversal Code-nya

a. Membuat class Node.java untuk deklarasikan node



```java
public class Node {
    int data;
    Node left;
    Node right;

    public Node(int data) {
        this.data = data;
        this.left = null;
        this.right = null;
    }
}
```

b. Membuat class BinaryTree.java dan menambahkan proses traversal kedalam (PreOrder, InOrder, dan PostOrder)

```java
public class BinaryTree {
    public Node root;

    public BinaryTree() {
        root = null;
    }

    // Menambahkan root ke tree
    public void addRoot(int data) {
        root = new Node(data);
    }

    // Traversal PreOrder (Root -> Left -> Right)
    public void preOrder(Node node) {
        if (node != null) {
            System.out.print(node.data + " ");
            preOrder(node.left);
            preOrder(node.right);
        }
    }
```

```java
    // Traversal InOrder (Left -> Root -> Right)
    public void inOrder(Node node) {
        if (node != null) {
            inOrder(node.left);
            System.out.print(node.data + " ");
            inOrder(node.right);
        }
    }

    // Traversal PostOrder (Left -> Right -> Root)
    public void postOrder(Node node) {
        if (node != null) {
            postOrder(node.left);
            postOrder(node.right);
            System.out.print(node.data + " ");
        }
    }

    // Main Method untuk menjalankan program
    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();

        // Membuat struktur tree seperti gambar (Complete Binary Tree) dengan angka acak
        tree.addRoot(73); // Root
        tree.root.left = new Node(14);          // Menambahkan node ke kiri root
        tree.root.right = new Node(91);         // Menambahkan node ke kanan root
        tree.root.left.left = new Node(5);      // Menambahkan node ke kiri dari node kiri root
        tree.root.left.right = new Node(62);    // Menambahkan node ke kanan dari node kiri root
        tree.root.right.left = new Node(28);    // Menambahkan node ke kiri dari node kanan root

        // Menampilkan hasil traversal
        System.out.print("\nPreOrder: ");
        tree.preOrder(tree.root);

        System.out.print("\nInOrder: ");
        tree.inOrder(tree.root);

        System.out.print("\nPostOrder: ");
        tree.postOrder(tree.root);
    }
}
```

LATIHAN 2: Merubah Urutan Data Menjadi Binary Search Tree

Contoh Kasus Merubah Urutan Data Menjadi Binary Search Tree

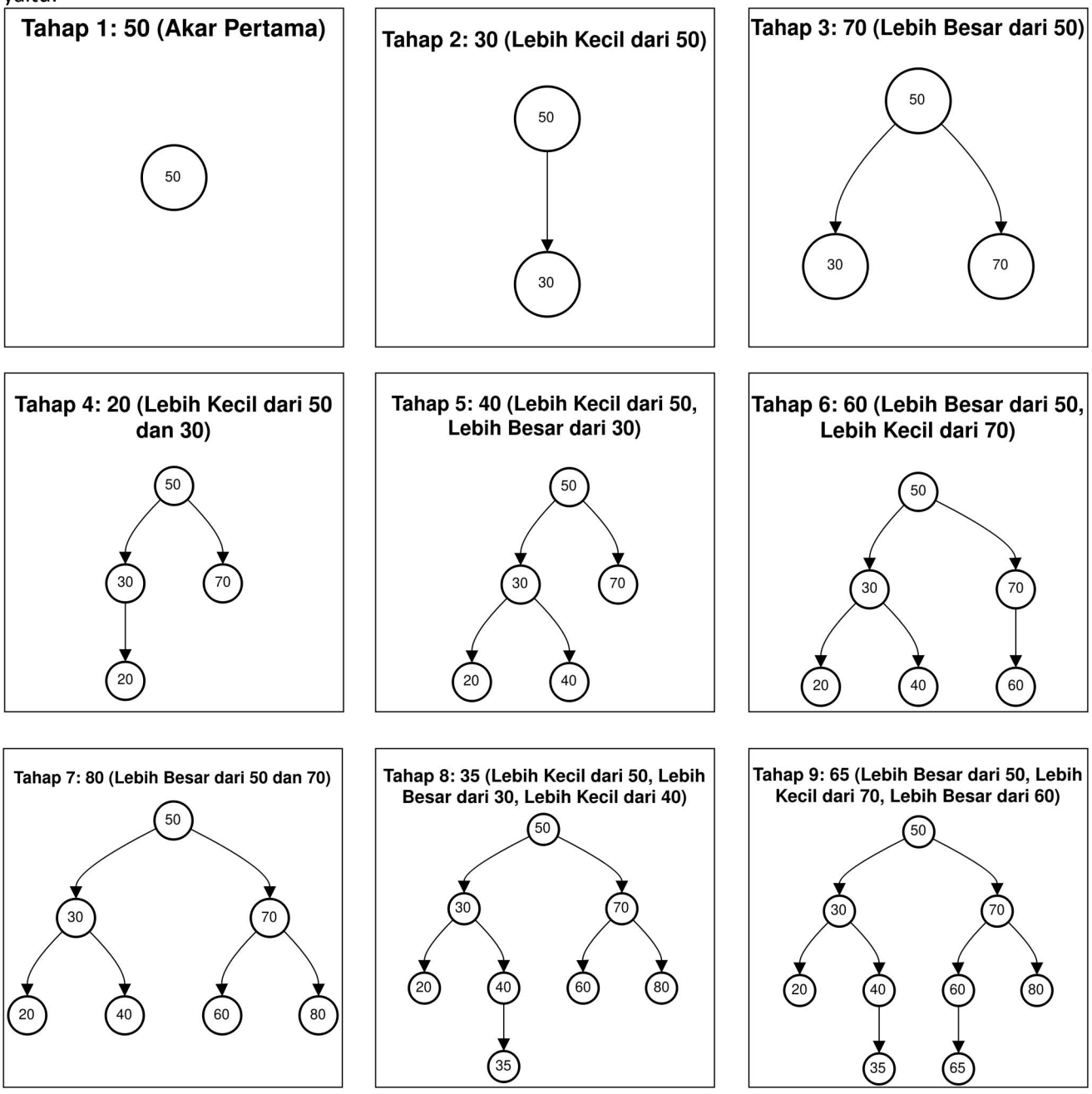
Aturan umum:

a. Jika binary Tree masih kosong, maka nilai pertama langsung menjadi root
b. Jika nilai berikutnya yang akan dimasukkan lebih kecill (<) dari node yang sudah ada, maka diletakan di bagian kiri bawah node yang sudah ada
c. Jika nilai berikutnya yang akan dimasukkan lebih besar atau sama dengan (>=) dari node yang sudah ada maka diletakkan di bagian kanan bawah node yang sudah ada tersebut
d. Pengecekan untuk peletakkan node dilakukan berulang-ulang sampai nilai yang akan dimasukkan tersebut menjadi leaf

Contoh:

Urutan data: 50, 30, 70, 20, 40, 60, 80, 35, 65

Jika diubah menjadi Binary Search Tree yaitu:



Gambar 9. Visualisasi Latihan 2 Binary Search Tree

NB: Jika data berupa selain angka, maka pengurutannya berdasarkan nilai ASCII

Tree Traversal Source Code

Dari data pohon yang sudah ada diatas, dapat dibuat kedalam program untuk menentukan Tree Traversal seperti berikut:

a. Membuat class Node. Java untuk deklarasikan node

```java
class Node {
    int data;
    Node left, right;

    public Node(int data) {
        this.data = data;
    }
}
```

b. Membuat class BinaryTree.java dan menambahkan proses traversal kedalam tree (PreOrder, InOrder, dan PostOrder)

```java
class BinaryTree {
    public Node root;

    public void NewNode(int data) {
        root = NewNode(root, new Node(data));
    }

    private Node NewNode(Node root, Node newData) {
        if (root == null) {
            return newData;
        }

        if (newData.data < root.data) {
            root.left = NewNode(root.left, newData);
        } else {
            root.right = NewNode(root.right, newData);
        }
        return root;
    }

    public void inOrder(Node node) {
        if (node != null) {
            inOrder(node.left);
            System.out.print(node.data + " ");
            inOrder(node.right);
        }
    }
```

```java
    public void preOrder(Node node) {
        if (node != null) {
            System.out.print(node.data + " ");
            preOrder(node.left);
            preOrder(node.right);
        }
    }

    public void postOrder(Node node) {
        if (node != null) {
            postOrder(node.left);
            postOrder(node.right);
            System.out.print(node.data + " ");
        }
    }

    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();

        int[] values = {50, 30, 70, 20, 40, 60, 80, 35, 65};
        for (int value : values) {
            tree.NewNode(value);
        }

        System.out.println("\nPre Order: ");
        tree.preOrder(tree.root);

        System.out.println("\nIn Order: ");
        tree.inOrder(tree.root);

        System.out.println("\nPost Order: ");
        tree.postOrder(tree.root);
    }
}
```

c. Perhatikan code diatas, apakah ada perbedaan antara logika dari code Binary Tree dengan Binary Search Tree? Jika ada tunjukkan bagian mana yang berbeda dan jelaskan kepada asisten masing-masing.

TUGAS

PERINGATAN: SETIAP PRAKTIKAN DILARANG MEMILIKI KODE PROGRAM YANG IDENTIK. JIKA DITEMUKAN KESAMAAN, MAKA NILAI AKAN DIBERIKAN 0.

Kegiatan 1: Visualisasi Keluarga dengan Binary Tree

Buatlah sebuah program Java yang dapat memvisualisasikan struktur keluarga menggunakan Binary Tree dan gambarkan **Binary Tree**-nya (gambar manual dikertas/ pakai mspaint, coredraw dll). Program ini akan menerima input dari pengguna mengenai data anggota keluarga (minimal 5 anggota) dan hubungan antar anggota keluarga tersebut. Setiap anggota keluarga akan direpresentasikan sebagai sebuah Node dalam Binary Tree.

Detail Implementasi:

1. **Input Data Keluarga:** Program harus meminta pengguna untuk memasukkan data anggota keluarga, termasuk:
   - Nama (String)
   - Jenis Kelamin (String: "Pria" atau "Wanita")
   - Relasi (String): contoh "Ayah", "Ibu", "Anak", "Saudara", "Kakek", "Nenek"
   - Untuk anggota keluarga pertama, pengguna harus menginput sebagai "Kepala Keluarga"
   - Data setiap anggota keluarga disimpan dalam Node.
2. **Struktur Binary Tree:**
   - Kepala Keluarga (anggota keluarga pertama) akan menjadi root dari Binary Tree.
   - Hubungan keluarga (misalnya, anak, saudara) akan menentukan posisi Node di dalam Tree.
   - *Asumsi*: Setiap orang hanya memiliki maksimal 2 anak (sesuai dengan struktur Binary Tree). Jika ada lebih dari 2 anak, pilih 2 saja.
   - *Tips*: Untuk memudahkan implementasi, bisa meminta pengguna memasukkan relasi anggota keluarga *terhadap Kepala Keluarga*. Contoh: "Anak ke-1 dari Kepala Keluarga", "Saudara dari Kepala Keluarga", dll.
3. **Visualisasi Binary Tree (Output):** Program harus menampilkan representasi visual dari Binary Tree yang menggambarkan struktur keluarga tersebut. Dapat menggunakan representasi berbasis teks (console) yang sederhana. Contoh:

```
Kakek
├── Ayah
│   ├── Saya
│   └── Saudara
└── Bibi
```

Penjelasan:

- **Kakek** adalah anggota keluarga paling atas (root).
- **Ayah** dan **Bibi** adalah anak dari Kakek.
- **Saya** dan **Saudara** adalah anak dari Ayah.

4. **Tree Traversal:**
   - Implementasikan metode Tree Traversal (PreOrder, InOrder, PostOrder) pada Binary Tree.
   - Tampilkan hasil Tree Traversal (nama-nama anggota keluarga) untuk setiap metode.
5. **Pertanyaan Asisten:**
   - Setelah Tree berhasil dibuat dan divisualisasikan, asisten akan memberikan pertanyaan terkait struktur keluarga dan relasi antar anggota keluarga berdasarkan Tree yang telah dibuat.
   - Dari gambar Binary Tree yang sudah kalian buat, berikan jawaban dengan jelas dari pertanyaan yang diberikan oleh asisten (pertanyaan akan diberikan langsung pada saat

praktikum, pertanyaan berhubungan dengan istilah pada BinaryTree. Berikut adalah contoh pertanyaan yang akan ditanyakan oleh asisten:

- Predessor
- Sucessor
- Ancestor

*Pastikan untuk mempelajari dan memahami semua istilah binary Tree agar dapat menjawab pertanyaan dengan benar.

6. **Contoh Output:**

```
Visualisasi Pohon Keluarga:
Kakek
├──Ayah
│   ├──Saya
│   └──Saudara
└──Bibi

PreOrder Traversal:
[Kakek] [Ayah] [Saya] [Saudara] [Bibi]

InOrder Traversal:
[Saya] [Ayah] [Saudara] [Kakek] [Bibi]

PostOrder Traversal:
[Saya] [Saudara] [Ayah] [Bibi] [Kakek]
```

KEGIATAN 2: Sistem Manajemen Koleksi Film Menggunakan BST

Buatlah sebuah program Java yang mengimplementasikan sistem manajemen koleksi film menggunakan Binary Search Tree (BST). Program ini harus menyimpan data film berdasarkan kode unik (misalnya kode produksi) sebagai kunci.

Detail Implementasi:

7. **Data Film:** Setiap film harus memiliki atribut berikut:
   - *Kode Produksi* (String, sebagai kunci BST)
   - *Judul Film* (String)
   - *Genre* (String)
   - *Tahun Rilis* (Integer)
   - *Rating Usia* (String, contoh: "SU", "R13+", "D")
   - *Jumlah Copy* (Integer, contoh: 5, 10, 20)
8. **Struktur BST:**
   - Gunakan *Kode Produksi* sebagai kunci untuk menyusun BST.
   - Implementasikan operasi dasar BST:
     - *Sisip (Insert)*: Menambahkan film baru ke dalam koleksi.
     - *Cari (Search)*: Mencari film berdasarkan *Kode Produksi*.
9. **Fitur Program:**
   - **Tambah Film:** Memungkinkan pengguna untuk menambahkan film baru ke dalam koleksi. Program harus meminta pengguna untuk memasukkan semua atribut film.
   - **Cari Film:** Memungkinkan pengguna untuk mencari film berdasarkan *Kode Produksi*. Jika film ditemukan, tampilkan semua informasi film tersebut. Jika tidak ditemukan, tampilkan pesan bahwa film tidak ditemukan.
   - **Tampilkan Koleksi (Traversal):** Menampilkan daftar semua film dalam koleksi menggunakan *InOrder Traversal* (terurut berdasarkan *Kode Produksi*).
   - **PreOrder dan PostOrder Traversal** Menampilkan semua koleksi menggunakan Preorder dan PostOrder Traversal
   - **Fitur Update/Edit Jumlah Copy Film**: Tambahkan fitur yang memungkinkan pengguna untuk

mengedit jumlah copy sebuah film yang telah ada dalam koleksi.

- **Pesan Error dan input validation:** Tambahkan pesan Error jika user salah dalam input dan validasi terhadap input.

10. **User Interface (UI):**
    - Buatlah UI berbasis teks (console) yang mudah digunakan.
    - Tampilkan menu utama dengan opsi: Tambah Film, Cari Film, Tampilkan Koleksi, dan Keluar.

NB:

* **error Handling dan input Validation:** Implementasi terhadap input data yang tidak valid (Misalnya, input huruf di angka tahun rilis)

Contoh:

Berikut adalah contoh interaksi dengan program:

```
Menu:
1. Tambah Film
2. Cari Film
3. Tampilkan Koleksi (InOrder)
4. Tampilkan Koleksi (PreOrder)
5. Tampilkan Koleksi (PostOrder)
6. Update/Edit Jumlah Copy Film
7. Keluar
Pilih opsi: 1
Kode Produksi: FIL001
Judul Film: The Shawshank Redemption
Genre: Drama
Tahun Rilis: 1994
Rating Usia: R
Jumlah Copy: 5

Film berhasil ditambahkan!

Menu:
1. Tambah Film
2. Cari Film
3. Tampilkan Koleksi (InOrder)
4. Tampilkan Koleksi (PreOrder)
5. Tampilkan Koleksi (PostOrder)
6. Update/Edit Jumlah Copy Film
7. Keluar
Pilih opsi: 2
Masukkan Kode Produksi yang dicari: FIL001
Kode Produksi: FIL001
Judul Film: The Shawshank Redemption
Genre: Drama
Tahun Rilis: 1994
Rating Usia: R
Jumlah Copy: 5
```

Catatan

Aturan umum penulisan JAVA agar mudah dikoreksi oleh asisten:

1. Untuk nama class, enum, dan yang lainnya biasakan menggunakan CamelCase(diawali dengan huruf besar pada tiap kata untuk mengganti spasi) seperti: JalanRaya, ParkiranGedung, dll.
2. Untuk penulisan nama method dan attribute diawali dengan huruf kecil di awal kata dan menggunakana huruf besar untuk kata setelahnya, seperti: getNamaJalan, namaJalan, dll.
3. Jika menggunakan IDE Inteliij jangan lupa untuk memformat kode agar terlihat rapi menggunakan menu code -> show reformat file dialog -> centang semua field dan klik ok.

KRITERIA & DETAIL PENILAIAN

Kriteria	Detail Penilaian Spesifik	Poin Maksimal
Codelab 1 & 2	Familiarisasi Awal Struktur Binary Tree & BST	**5**
	Codelab 1: Membuat struktur BT manual & traversal dasar	2
	Codelab 2: Memahami & implementasi *insert* BST sederhana & traversal	3
Pemahaman	**Penilaian Utama:** Kemampuan menjelaskan konsep fundamental Tree, BST, Traversal, dan kode yang dibuat.	**60**
	Menjelaskan Konsep Dasar Tree & Binary Tree (Hierarki, Node, Root, Child, Parent, Leaf, dll.)	10
	Menjelaskan Konsep Binary Search Tree (BST) (Aturan Urutan Kiri < Root < Kanan, Keunggulan untuk search/insert)	15
	Menjelaskan Mekanisme Traversal (PreOrder, InOrder, PostOrder – bagaimana cara kerjanya, bukan hanya urutan output)	15
	Menjelaskan Kode Tugas 1 (Representasi Node Keluarga, Logika Pembangunan Tree, Cara Kerja Traversal pada Tree Keluarga, Menjawab Istilah Tree pada kasusnya)	10
	Menjelaskan Kode Tugas 2 (Struktur Node Film, Logika *insert* BST, *search* BST, *update*, Cara Kerja Traversal pada BST Film)	10
Kegiatan 1	Implementasi Visualisasi Keluarga (Binary Tree)	**15**
	Struktur Node & Input Data (Class Node Keluarga, program bisa menerima input)	3
	Logika Pembangunan Tree (Berhasil menghubungkan Node berdasarkan relasi input, membentuk struktur tree)	4
	Implementasi Traversal (Kode PreOrder, InOrder, PostOrder berfungsi pada tree yang dibuat)	4
	Output (Menampilkan visualisasi tree teks dan hasil traversal dengan benar)	4
Kegiatan 2	Implementasi Manajemen Koleksi Film (BST)	**20**
	Struktur Node & BST (Class Node Film dengan atribut, Class BST dasar)	4
	Operasi *insert* BST (Berfungsi benar sesuai aturan BST berdasarkan Kode Produksi)	4
	Operasi *search* BST (Berfungsi benar menemukan film berdasarkan Kode Produksi)	3
	Implementasi Traversal (Kode InOrder, PreOrder, PostOrder berfungsi pada BST Film)	4
	Fitur Update Jumlah Copy (Logika mencari node lalu mengubah atributnya)	3
	UI Menu & Error Handling Dasar (Menu console berfungsi, ada pesan jika film tidak ditemukan)	2
TOTAL		**100**