

BAB 13

PEMANFAATAN DOUBLE LINKEDLIST SEBAGAI POHON BINER

Tujuan :

Mahasiswa dapat mengimplementasikan penggunaan [Simpul milik Double Linked List](#) untuk pembuatan pohon.

13.1. Definisi-definisi

Pohon (*tree*) adalah kumpulan **akar** (*root*) , **cabang**, dan **simpul** (*node*) yang saling terhubung secara hirarki

Pohon Biner (*binary tree*) adalah pohon dimana setiap simpulnya (*node*) hanya boleh memiliki **maksimal** 2 anak (dari cabang kiri, dan kanan)

Simpul (*node*) :

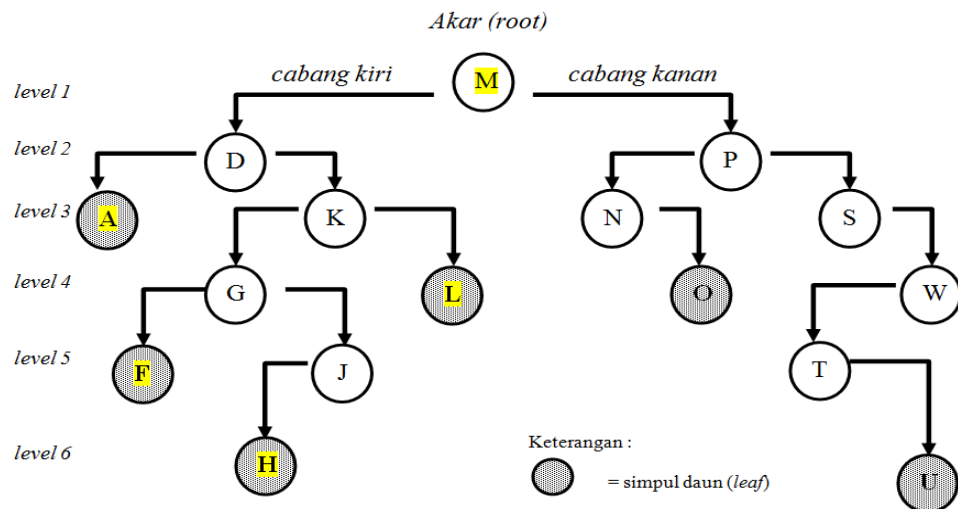
- simpul anak (*children*) hanya boleh punya 1 parent
- simpul orangtua (*parent*) hanya boleh punya **maksimal** 2 anak, namun boleh juga tidak punya anak

Simpul **Akar** adalah simpul yang tidak memiliki *parent*

Simpul **Daun** adalah simpul yang tidak memiliki anak (*children*)

Cabang di dalam pohon biner terdiri dari **cabang kiri**, **cabang kanan**

Level menunjukkan tingkat hirarki



Simpul yang digunakan untuk membentuk sebuah pohon sama dengan simpul yang digunakan pada Senarai Berantai Ganda, yaitu sebagai berikut :

```
class simpul
{
    String elemen;
    simpul kiri;
    simpul kanan;
}
```

13.2. Operasi Pada Pohon Biner

13.2.1. Menambahkan Simpul Baru ke dalam Sebuah Pohon Biner

Algoritmanya sebagai berikut.

- A. Baca **ElemenBaru**
- B. Jika pohon masih kosong :
 → jadikan **ElemenBaru** sebagai akar, menuju langkah D.
- C. Jika pohon tidak kosong :
 - C.1. **Penunjuk** = akar
 - C.2. Baca **Penunjuk.Elemen**
 - C.3. Jika **ElemenBaru < Penunjuk.Elemen** kerjakan langkah berikut:
 - C3.a. jika **Penunjuk** tidak punya anak di cabang kiri, jadikan **ElemenBaru** sebagai Anak di cabang kiri dari **Penunjuk**. Menuju langkah D.
 - C3.b. jika **Penunjuk** punya anak di cabang kiri, jadikan anak cabang kiri sebagai **Penunjuk**, ulangi langkah C.2
 - C.4. Jika **ElemenBaru >= Penunjuk.Elemen** kerjakan langkah berikut:

- C4.a. jika **Penunjuk** tidak punya anak di cabang kanan, jadikan **ElemenBaru** sebagai Anak di cabang kanan dari **Penunjuk**. Menuju langkah D.
- C4.b. jika **Penunjuk** punya anak di cabang kanan, jadikan anak cabang kanan sebagai **Penunjuk**, ulangi langkah C.2

D. Selesai.

```
class pohon
{   public static simpul akar;

    public static void deklarasiPohon()
    {   akar = null;
    }

    public static simpul tambahSimpul(simpul Penunjuk, String ELEMEN)
    {   if (Penunjuk == null)
        {   simpul baru = new simpul();
            baru.elemen = ELEMEN;
            baru.kiri = null;
            baru.kanan = null;
            Penunjuk = baru;
            return (Penunjuk) ;
        }
        else
        {   if (ELEMEN.compareTo(Penunjuk.elemen) < 0 )
            {   Penunjuk.kiri = tambahSimpul(Penunjuk.kiri, ELEMEN);
                return (Penunjuk) ;
            }
            else
            {   Penunjuk.kanan= tambahSimpul(Penunjuk.kanan, ELEMEN);
                return (Penunjuk) ;
            }
        }
    }

    .....;
    .....;

    public static void main(String[] args)
    {
        deklarasiPohon();
        .....;
        .....;
    }
}
```

13.2.2. Mencetak Isi Sebuah Pohon Biner

Mencetak isi pohon (kunjungan) dilakukan dengan cara rekursif yang terdiri dari :

1. **Preorder** adalah mencetak isi pohon dengan urutan :

1. **Cetak isi node yang dikunjungi**
 2. kunjungi Anak Cabang Kiri
 3. kunjungi Anak Cabang Kanan
2. **In Order** adalah mencetak isi pohon dengan urutan :
 1. kunjungi Anak Cabang Kiri
 2. **Cetak isi node yang dikunjungi**
 3. kunjungi Anak Cabang Kanan
3. **Post Order** adalah mencetak isi pohon dengan urutan :
 1. kunjungi Anak Cabang Kiri
 2. kunjungi Anak Cabang Kanan
 3. **Cetak isi node yang dikunjungi**

13.2.2.1. Teknik Kunjungan Pre Order

13.2.2.2. Teknik Kunjungan In Order

13.2.2.3. Teknik Kunjungan Post Order

```

class pohon
{
    .....;
    .....;
    public static void preOrder(simpul Penunjuk)
    {
        if(Penunjuk != null)
        {
            System.out.print(Penunjuk.elemen + ",");
            preOrder(Penunjuk.kiri);
            preOrder(Penunjuk.kanan);
        }
    }
    public static void inOrder(simpul Penunjuk)
    {
        if(Penunjuk != null)
        {
            inOrder(Penunjuk.kiri);
            System.out.print(Penunjuk.elemen + ",");
            inOrder(Penunjuk.kanan);
        }
    }
    public static void postOrder(simpul Penunjuk)
    {
        if(Penunjuk != null)
        {
            postOrder(Penunjuk.kiri);
            postOrder(Penunjuk.kanan);
            System.out.print(Penunjuk.elemen + ",");
        }
    }
    .....;
    .....;
    public static void main(String[] args)
    {
        deklarasiPohon();
        .....;
        .....;
    }
}

```

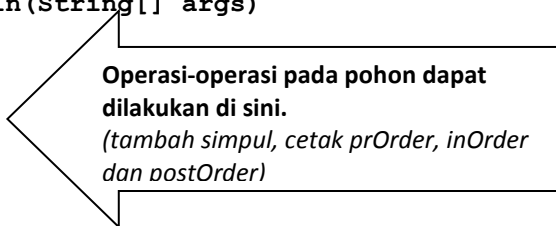
```
}
}
```

13.3. Latihan

Latihan 1 :

Tuliskan program berikut ini menggunakan textpad ataupun netbin

```
class simpul
{
    .....
}
class pohon
{
    .....
    void deklarasiPohon()
    .....
    simpul tambahSimpul(simpul Penunjuk, String ELEMEN)
    .....
    void preOrder(simpul Penunjuk)
    .....
    void inOrder(simpul Penunjuk)
    .....
    void postOrder(simpul Penunjuk)
    .....
}
class ProgramPohonBiner
{
    public static void main(String[] args)
    {
        deklarasiPohon();
        .....
        .....
        .....
        .....
    }
}
```



Operasi-operasi pada pohon dapat dilakukan di sini.
(tambah simpul, cetak prOrder, inOrder dan postOrder)

Setelah anda tulis program di atas, tambahkanlah pada program utama perintah-perintah untuk menambah simpul berikut ini :

```
akar =tambahSimpul(akar, "M");
akar =tambahSimpul(akar, "P");
akar =tambahSimpul(akar, "D");

akar =tambahSimpul(akar, "A");
```

```

akar =tambahSimpul(akar,"S");
akar =tambahSimpul(akar,"K");
akar =tambahSimpul(akar,"N");

akar =tambahSimpul(akar,"G");
akar =tambahSimpul(akar,"O");
akar =tambahSimpul(akar,"L");
akar =tambahSimpul(akar,"W");

akar =tambahSimpul(akar,"F");
akar =tambahSimpul(akar,"J");
akar =tambahSimpul(akar,"T");

akar =tambahSimpul(akar,"H");
akar =tambahSimpul(akar,"U");

```

Di bawah perintah-perintah tersebut, tambahkan perintah untuk mencetak pohon biner secara preOrder berikut :

```
preOrder(akar);
```

Kemudian eksekusilah program di atas. Hasil apakah yang didapat? Catatlah dalam laporan anda dan jelaskan mengapa bisa demikian.

Latihan 2 :

Lakukan juga eksperimen yang sama untuk cetak inOrder dan cetak postOrder berikut :

```
inOrder(akar);
```

Kemudian eksekusilah program di atas. Hasil apakah yang didapat? Catatlah dalam laporan anda dan jelaskan mengapa bisa demikian.

Latihan 3 :

Lakukan hal yang sama sebagaimana Latihan 2 untuk cetak postOrder.

```
postOrder(akar);
```

Kemudian eksekusilah program di atas. Hasil apakah yang didapat? Catatlah dalam laporan anda dan jelaskan mengapa bisa demikian.