

LAPORAN PRAKTIKUM STRUKTUR DATA

PERTEMUAN KE-13



Disusun Oleh :

NAMA : Raden Isnawan Argi Aryasatya

NIM : 195410257

JURUSAN : Teknik Informatika

JENJANG : S1

Laboratorium Terpadu

Sekolah Tinggi Management Informatika Komputer

AKAKOM

YOGYAKARTA

2020

PERTEMUAN KE-13 (POHON BINER)

TUJUAN

Mahasiswa dapat mengimplementasikan penggunaan Simpul milik Double Linked List untuk pembuatan pohon

DASAR TEORI

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)


PRAKTIK 1

```
pohon.java X
1  class simpul
2  {
3  String elemen;
4  simpul kiri;
5  simpul kanan;
6  }
7  class pohon
8  {
9  public static simpul akar;
10
11  public static void deklarasiPohon()
12  {
13  akar = null;
14  }
15  public static simpul tambahSimpul(simpul Penunjuk, String ELEMEN)
16  { if (Penunjuk == null)
17    { simpul baru = new simpul();
18      baru.elemen = ELEMEN;
19      baru.kiri = null;
20      baru.kanan = null;
21      Penunjuk = baru;
22      return(Penunjuk);
23    }
24    else
25    {
26      if (ELEMEN.compareTo(Penunjuk.elemen) < 0 )
27      {
28        Penunjuk.kiri = tambahSimpul(Penunjuk.kiri, ELEMEN);
29        return(Penunjuk);
30      }
31      else
32      {
33        Penunjuk.kanan= tambahSimpul(Penunjuk.kanan, ELEMEN);
34        return(Penunjuk);
35      }
36    }
37  }
38  public static void preOrder(simpul Penunjuk)
39  {
40    if(Penunjuk != null)
41    {
42      System.out.print(Penunjuk.elemen + ",");
43      preOrder(Penunjuk.kiri);
44      preOrder(Penunjuk.kanan);
45    }
46  }
47  public static void inOrder(simpul Penunjuk)
48  { if(Penunjuk != null)
49    {
50      inOrder(Penunjuk.kiri);
51      System.out.print(Penunjuk.elemen + ",");
52      inOrder(Penunjuk.kanan);
53    }
54  }
55  public static void postOrder(simpul Penunjuk)
56  {
57    if(Penunjuk != null)
58    {
```

```

59     postOrder(Penunjuk.kiri);
60     postOrder(Penunjuk.kanan);
61     System.out.print(Penunjuk.element + ",");
62 }
63 }
64 public static void main(String[] args)
65 {
66     deklarasiPohon();
67     akar = tambahSimpul(akar, "M");
68     akar = tambahSimpul(akar, "P");
69     akar = tambahSimpul(akar, "D");
70     akar = tambahSimpul(akar, "A");
71     akar = tambahSimpul(akar, "S");
72     akar = tambahSimpul(akar, "K");
73     akar = tambahSimpul(akar, "N");
74     akar = tambahSimpul(akar, "G");
75     akar = tambahSimpul(akar, "O");
76     akar = tambahSimpul(akar, "L");
77     akar = tambahSimpul(akar, "W");
78     akar = tambahSimpul(akar, "F");
79     akar = tambahSimpul(akar, "J");
80     akar = tambahSimpul(akar, "T");
81     akar = tambahSimpul(akar, "H");
82     akar = tambahSimpul(akar, "U");
83     preOrder(akar);
84 }
85 }

```



penjelasan:

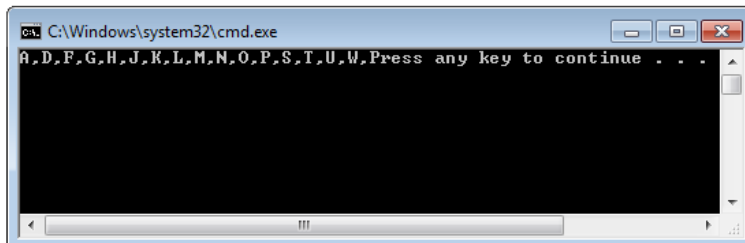
hasilnya adalah urut dari simpul pertama hingga simpul terakhir sesuai yang user masukkan. Mengapa? Karena ini adalah preorder. Preorder adalah mencetak isi node yang dikunjungi terlebih dahulu lalu baru mengunjungi cabang kiri dan cabang kanan. Alias tidak peduli urutan. Preorder hanya fokus dalam mencetak program sesuai urutan yang dimasukkan oleh pengguna

PRAKTIK 2

```

66 deklarasiPohon();
67 akar = tambahSimpul(akar, "M");
68 akar = tambahSimpul(akar, "P");
69 akar = tambahSimpul(akar, "D");
70 akar = tambahSimpul(akar, "A");
71 akar = tambahSimpul(akar, "S");
72 akar = tambahSimpul(akar, "K");
73 akar = tambahSimpul(akar, "N");
74 akar = tambahSimpul(akar, "G");
75 akar = tambahSimpul(akar, "O");
76 akar = tambahSimpul(akar, "L");
77 akar = tambahSimpul(akar, "W");
78 akar = tambahSimpul(akar, "F");
79 akar = tambahSimpul(akar, "J");
80 akar = tambahSimpul(akar, "T");
81 akar = tambahSimpul(akar, "H");
82 akar = tambahSimpul(akar, "U");
83 inOrder(akar);
84 }
85 }

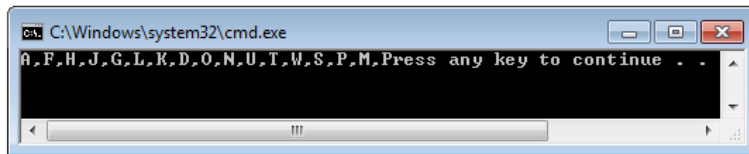
```



Hasilnya adalah urut sesuai abjad. InOrder berarti sesuai urutan. dia mengutamakan anak cabang kiri yaitu anak cabang utama lalu mencetak program

PRAKTIK 3

```
66 deklarasiPohon();
67 akar = tambahSimpul(akar, "M");
68 akar = tambahSimpul(akar, "P");
69 akar = tambahSimpul(akar, "D");
70 akar = tambahSimpul(akar, "A");
71 akar = tambahSimpul(akar, "S");
72 akar = tambahSimpul(akar, "K");
73 akar = tambahSimpul(akar, "N");
74 akar = tambahSimpul(akar, "G");
75 akar = tambahSimpul(akar, "O");
76 akar = tambahSimpul(akar, "L");
77 akar = tambahSimpul(akar, "W");
78 akar = tambahSimpul(akar, "F");
79 akar = tambahSimpul(akar, "J");
80 akar = tambahSimpul(akar, "T");
81 akar = tambahSimpul(akar, "H");
82 akar = tambahSimpul(akar, "U");
83 postOrder(akar);
84 }
85 }
```

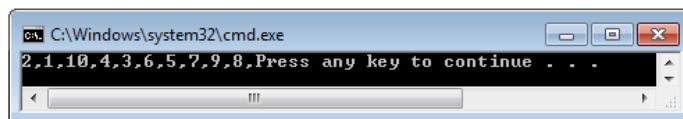


Hasilnya adalah kumpulan huruf yang acak. mengutamakan anak cabang kiri yaitu anak cabang kanan lalu mencetak program sehingga output terlihat tidak urut dan berantakan

TUGAS

PreOrder

```
66 deklarasiPohon();
67 akar = tambahSimpul(akar, "2");
68 akar = tambahSimpul(akar, "1");
69 akar = tambahSimpul(akar, "10");
70 akar = tambahSimpul(akar, "4");
71 akar = tambahSimpul(akar, "6");
72 akar = tambahSimpul(akar, "3");
73 akar = tambahSimpul(akar, "5");
74 akar = tambahSimpul(akar, "7");
75 akar = tambahSimpul(akar, "9");
76 akar = tambahSimpul(akar, "8");
77 preOrder(akar);
78 }
79 }
```



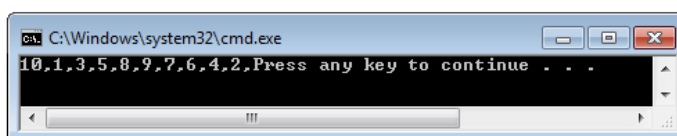
InOrder

```
64 public static void main(String[] args)
65 {
66 deklarasiPohon();
67 akar = tambahSimpul(akar, "2");
68 akar = tambahSimpul(akar, "1");
69 akar = tambahSimpul(akar, "10");
70 akar = tambahSimpul(akar, "4");
71 akar = tambahSimpul(akar, "6");
72 akar = tambahSimpul(akar, "3");
73 akar = tambahSimpul(akar, "5");
74 akar = tambahSimpul(akar, "7");
75 akar = tambahSimpul(akar, "9");
76 akar = tambahSimpul(akar, "8");
77 inOrder(akar);
78 }
79 }
```



PostOrder

```
66 deklarasiPohon();
67 akar = tambahSimpul(akar, "2");
68 akar = tambahSimpul(akar, "1");
69 akar = tambahSimpul(akar, "10");
70 akar = tambahSimpul(akar, "4");
71 akar = tambahSimpul(akar, "6");
72 akar = tambahSimpul(akar, "3");
73 akar = tambahSimpul(akar, "5");
74 akar = tambahSimpul(akar, "7");
75 akar = tambahSimpul(akar, "9");
76 akar = tambahSimpul(akar, "8");
77 postOrder(akar);
78 }
79 }
```



KESIMPULAN

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