Drs. Achmad Ridok, M.Kom.
Indriati, ST., MT.

Modul Praktikum

Algoritma dan Struktur Data

Laboratorium Komputer Dasar
Program Teknik Informatika dan Ilmu Komputer (PTIIK)
Universitas Brawijaya
Malang
2013

Kata Pengantar

Puji syukur Alhamdulillah berkat rahmat dan pertolongannya modul praktikum ASD-1 tahun 2011 telah berhasil diselesaikan.

Modul ini dimaksudkan hanyalah membantu proses pemahaman mahasiswa terhadap konsep struktur yang dipelajari di kelas. Sehingga tidak semua materi struktur data diulas pada modul ini. Sebagaimana target kuliah struktur data maka praktikum ini disusun untuk memenuhi kompetensi yang ada, yakni mahasiswa dapat memahami dan mengembangkan konsep ADT untuk menyelesaikan masalah-masalah dunia nyata.

Sistematika bahasan dalam modul ini disusun sedemikan rupa sehingga setelah mahasiswa berlatih memahami program yang ada, mahasiswa dapat mengembangkan untuk menyelesaikan masalah-masalah yang diberikan pada bagian tugas.

Namun demikian modul ini masih jauh dari sempurna yang masih perlu pembenahan, masukan dan saran dari semua pihak untuk tercapainya kompetensi yang telah ditetapkan.

Malang, 1 Februari 2013

Penyusun

Materi Praktikum

1. Pengenalan *Object Oriented Programming*
2. ADT Array 1 Dimensi
3. ADT Array 2 Dimensi
4. ADT Stack
5. ADT Single Linked List
6. ADT Double Linked List
7. ADT Sirkular Linked List
8. ADT Antrian
9. ADT Binary Tree
10. ADT AVL Tree
11. ADT Graph
12. *Sorting* (Pengurutan)

1

Pengenalan Object Oriented Programming

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman mengenai Object Oriented Programming*

Teori

1. Class

Didefinisikan Class sebagai sebuah blue print, atau prototipe, yang mendefinisikan variable-variabel dan metode-metode yang umum untuk semua objek dari jenis tertentu. Class mendefinisikan atribut dan perilaku objek yang dibuatnya. Class merupakan definisi formal suatu abstraksi. Class berlaku sebagai template untuk pembuatan objek-objek.Class berisi abstraksi yang terdiri dari nama class, atribut dan service/behaviour.

Bagian-bagian dari sebuah Class secara umum penulisan class terdiri atas 2 bagian yakni:

1.1 Class Declaration

Bentuk Umum :

```
[modifier] class <nama_kelas>
{
  ...
  ...
<class body>
...
...
}
```

[modifier] adalah pengaturan level akses terhadap kelas tersebut. Dengan kata
lain, modifier ini akan menentukan sejauh mana kelas ini dapat digunakan oleh kelas atau package lainnya. Adapun macam-macam modifier ialah :

- kosong / default / not specified
  Kelas tersebut dapat diakses oleh kelas lain dalam satu package.
- public
  Kelas tersebut dapat dipakai dimanapun, maupun kelas lain atau package

lain.

- private
  Kelas tersebut tidak dapat diakses oleh kelas manapun.

1.2 Class Body

Class Body merupakan bagian dari kelas yang mendeklarasikan kode program java. Class Body tersusun atas:

a. Konstruktor
b. Variable Instance (Atribut)
c. Method (dikenal juga sebagai function atau def)

Untuk dapat menggunakan kelas yang telah didefinisikan, anda harus membuat sebuah objek dari kelas tersebut (instance class), dengan syntax:

```
NamaKelas namaObjek = new NamaKelas ( [parameter] );
```

Contoh:

```
Hitungluas segitiga  = new Hitungluas();
```

1.3 Instance Variables (Atribut)

Suatu objek dapat dibedakan berdasarkan sifat (behavior) yang berbeda. objek juga dapat dibedakan berdasarkan atributnya. Misalnya burung dapat dibedakan berdasarkan suara kicauan, warna bulu, bentuk tubuh, dsb. . Dalam bahasa lain dikenal juga sebagai property yang mana merupakan ciri-ciri dari sebuah objek.

Atribut yang membedakan suatu instance objek burung yang satu dengan yang lainnya disebut instance variable.

Bentuk Umum :

```
[modifier] <tipe_data> <nama_variabel> = [nilai_default];
```

Contoh :

```
public double tinggi;
private int berat = 70;
```

Modifier untuk atribut, yaitu public, protected, private. Penjelasan modifier atribut
serupa dengan penjelasan modifier pada kelas.

Adapun perbedaan local dan instance variable adalah :

1. Instance variable dideklarasikan di dalam kelas tetapi tidak di dalam method.

```
class segitiga {
  double tinggi = 15.2; // ini adalah variabel instance
   String jenis; // ini adalah variabel instance
   int tambah() {
      return 3;
    }
}
```

2. Local variable dideklarasikan di dalam method.

```
int tambah() {
  int total = tinggi * 2;     // total adalah variabel
  local
  return total; }
```

1.4 Method

Sebuah method adalah bagian-bagian kode yang dapat dipanggil oleh kelas, badan program atau method lainnya untuk menjalankan fungsi yang spesifik di dalam kelas. Secara umum method dalam java adalah sebuah fungsi.

Berikut adalah karakteristik dari method :

1. Dapat mengembalikan / melaporkan nilai balikkan (return value) atau tidak (void)
2. Dapat diterima beberapa parameter yang dibutuhkan atau tidak ada parameter sama sekali. Parameter bisa juga disebut sebagai argumen dari fungsi. Parameter berguna sebagai nilai masukkan yang hendak diolah oleh fungsi.
3. Setelah method telah selesai dieksekusi, dia akan kembali pada method yang memanggilnya.

Deklarasi sebuah method

Method terdiri atas dua bagian yakni :

1. Method declaration
2. Method Body

Method dapat digambarkan sebagai sifat (behavior) dari suatu class. Untuk
mendefinisikan method pada dalam class digunakan syntax :

```
[modifier] <tipe_data_return> nama_method( [parameter] )
{
    ...
    ...
    ...
    return <tipe_data_return>;
}
```

Contoh :

```
public int Perkalian ( int y;int z )
{
    return y * z ;
}
```

1.5 Modifier pada method

Modifier menentukan level pengaksesan sebuah method. Hal ini menentukan apakah sebuah method biasa diakses oleh objek lain, objek anak, objek dalam satu paket atau tidak dapat diakses oleh suatu object sama sekali berikut adalah beberapa jenis level access:

- Public
  Atribut ini menunjukan bahwa fungsi/method dapat diakses oleh kelas lain.
- Private
  Atribut ini menunjukan bahwa fungsi atau method tidak dapat diakses oleh kelas lain
- Protected
  Atribut ini menunjukan bahwa fungsi atau method bisa diakses oleh kelas lain dalam satu paket dan hanya kelas lain yang merupakan subclass nya pada paket yang berbeda.
- Tanpa modifier
  Atribut ini menunjukan bahwa method dapat diakses oleh kelas lain dalam paket yang sama.

1.6 Method tanpa nilai balikan

Method ini merupakan method yang tidak mengembalikan nilai. Maka dari itu, kita harus mengganti tipe kembalian dengan kata kunci void. Berikut ini kode program yang dimaksud:

	Program Latihan Praktikum 1.1
1	`class Kotak{`
2	`double panjang;`
3	`double lebar;`
4	`   double tinggi;`
5	`//mendefinisikan method void (tidak mengembalikan nilai)`
6	`void cetakVolume(){`
7	`System.out.println("Volume kotak = " +(panjang*lebar*tinggi));`
8	`}`
9	`}`
10	
11	`class DemoMethod1{`
12	`  public static void main(String[] args){`
13	`    Kotak k1, k2, k3;`
14	`    //instansiasi objek`
15	`    k1=new Kotak();`
16	`    k2=new Kotak();`
17	`    k3=new Kotak();`
18	
19	`    //mengisi data untuk objek k1`
20	`    k1.panjang=4;`
21	`    k1.lebar=3;`
22	`    k1.tinggi=2;`
23	`    //mengisi data untuk objek k2`
24	`    k2.panjang=6;`
25	`    k2.lebar=5;`
26	`    k2.tinggi=4;`
27	`    //mengisi data untuk objek k3`
28	`    k3.panjang=8;`
29	`    k3.lebar=7;`
30	`    k3.tinggi=6;`
31	`    //memanggil method cetakVolume() untuk masing-masing`
32	`      //objek`

```
33      k1.cetakVolume();
34      k2.cetakVolume();
35      k3.cetakVolume();
36    }
37  }
```

Latihan 1.1

1. Jalankan program diatas
2. Berapakah volume yang ditampilkan untuk ketiga kotak diatas?

1.7 Method dengan nilai balikan

Di sini, kita akan memodifikasi program sebelumnya dengan mengganti method cetakVolume() menjadi method hitungVolume() yang akan mengembalikan nilai dengan tipe double. Berikut ini kode program yang dimaksud:

```
    Program Latihan Praktikum 1.2
1   class Kotak{
2      double panjang;
3   double lebar;
4   double tinggi;
5   //mendefinisikan method yang mengembalikan tipe double
6   double hitungVolume(){
7      //menghitung volume
8     double vol = panjang*lebar*tinggi;
9     //mengembalikan nilai
10    return vol;
11  }
12  }
13  class DemoMethod2{
14    public static void main(String[] args){
15      Kotak k1, k2, k3;
16      //instansiasi objek
17      k1=new Kotak();
18      k2=new Kotak();
19      k3=new Kotak();
20      //mengisi data untuk objek k1
21      k1.panjang=4;
22      k1.lebar=3;
23      k1.tinggi=2;
24      //mengisi data untuk objek k2
25      k2.panjang=6;
26      k2.lebar=5;
27      k2.tinggi=4;
28      //mengisi data untuk objek k3
29      k3.panjang=8;
30      k3.lebar=7;
31      k3.tinggi=6;
32      System.out.println("Volume k1 = "+k1.hitungVolume());
33      System.out.println("Volume k2 = "+k2.hitungVolume());
34      System.out.println("Volume k3 = "+k3.hitungVolume());
35    }
36  }
```

Latihan 1.2

1. Jalankan program diatas
2. Berapakah volume yang ditampilkan untuk ketiga kotak diatas?

1.8 Parameter

Dengan adanya parameter, sebuah method dapat bersifat dinamis dan general. Artinya,method tersebut dapat mengembalikan nilai yang beragam sesuai dengan nilai parameter yang dilewatkan. Terdapat dua istilah yang perlu anda ketahui dalam bekerja dengan method, yaitu parameter dan argumen. Parameter adalah variabel yang didefinisikan pada saat method dibuat, sedangkan argumen adalah nilai yang digunakan pada saat pemanggilan method. Dalam referensi lain, ada juga yang menyebut parameter sebagai parameter formal dan argumen sebagai parameter aktual. Perhatikan kembali definisi method berikut :

```
int luasPersegiPanjang(int panjang, int lebar){
   return panjang * lebar;
}
```

Di sini, variabel panjang dan lebar disebut parameter.
Luas1 = luasPersegiPanjang(10, 5);
Adapun pada statemen diatas, nilai 10 dan 5 disebut argumen. Sekarang, kita akan mengimplementasikan konsep di atas ke dalam kelas Kotak. Di sini data panjang, lebar, dan tinggi akan kita isikan melalui sebuah method. Berikut ini kode program yang dapat digunakan untuk melakukan hal tersebut.

	Program Latihan Praktikum 1.3
1	class Kotak{
2	double panjang;
3	double lebar;
4	double tinggi;
5	
6	//mendefinisikan method dengan parameter
7	void isiData(double p, double l, double t){
8	panjang = p;
9	lebar = l;
10	tinggi = t;
11	}
12	double hitungVolume(){
13	return(panjang*lebar*tinggi);
14	}
15	}
16	class DemoMethod3{
17	public static void main(String[] args){
18	Kotak k;
19	//instansiasi objek
20	k = new Kotak();
21	
22	//memanggil method isiData()

```
23    k.isiData(4,3,2);
24
25    System.out.println("Volume kotak = " + k.hitungVolume());
26    }
27  }
```

Latihan 1.3

1. Jalankan program diatas
2. Berapakah volume yang ditampilkan untuk kotak diatas?

Bagaimana jika bagian lain dari program ingin tahu juga nilai volume itu tetapi tidak ingin menampilkannya (mencetaknya). Apabila terdapat suatu fungsi yang tidak menghasilkan suatu nilai apapun maka bagian return type ini diganti dengan void .

Contoh penggunaan return:

```
      Program Latihan Praktikum 1.4
1     package cobaCoba;
2     import java.util.Scanner;
3
4     class balok {
5     int  p, l, t;
6     int volume( int p, int l, int t)
7     {
8       return (p*l*t);
9     }
10
11    public static void main(String args[]) {
12    Scanner masuk = new Scanner(System.in);
13    //fungsi untuk menginputkan suatu nilai
14    System.out.print("Panjang = "); int a=masuk.nextInt();
15    System.out.print("Lebar = "); int b=masuk.nextInt();
16    System.out.print("Tinggi = "); int c=masuk.nextInt();
17
18    balok coba = new balok();
19    System.out.print("\nVolume balok = "+ coba.volume(a,b,c));
20    }
21    }
```

Latihan 1.4

1. Jalankan program diatas
2. Berapakah volume yang ditampilkan balok diatas?

1.9 Method Static

Sebuah method static dapat diakses tanpa harus melakukan instantiasi terlebih dahulu. Pemanggilan method static dilakukan dengan format :
Nama_kelas.nama_method();
Nama_kelas diberikan bila method tersebut dipanggil dari kelas yang berbeda.

Contoh :

```
    Program Latihan Praktikum 1.5
1   public class persegi {
2     static int hitungluas(int p, int l){
3       return p*l;
4     }
5     public static void main(String[] args) {
6        int z=0;
7        z=hitungluas(3,2);
8        System.out.println(z);
9     }
10  }
```

Latihan 1.5

1. Jalankan program diatas
2. Berapakah luas yang ditampilkan persegi diatas?

Tugas 1

1. Buatlah program untuk membuat kalkulator penjumlahan, pengurangan, perkalian dan pembagian dengan menggunakan parameter dan argumen

2

Array 1 Dimensi

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman Abstract Data Type Array 1 dimensi.*

Teori

Array adalah representas data pada lokasi memori secara berurutan dan bertipe sama. Pengaksesan data ke dalam array menggunakan indek. Karena sifatnya inilah maka array cocok digunakan untuk pengornasisasian data yang seragam.

Secara umum struktur data array telah dibuat oleh pewngembang compiler bahasa apapun termasuk bahasa pemrogram java yang mempunyai strukrut umum sebagai berikut :

typeData [] namaLarik;

Keterangan :

- typeData : bisa tipe data dasar seperti int, float dst, atau berupa tipe bentukan seperti class atau object.
- Pada dasarnya tipedata arraypun juga termasuk tipe data bentukan, sehingga atas dasar inilah kita bisa membuat varian dari applikasi array.

Contoh :

1. Int []A,B;

- mendeklarasikan dua buah array A dan B masing-masing bertipe integer.

2. Class X{}

   X []A;

- mendeklarasikan array A bertipe class X.

ADT Array 1 Dimensi

Namun demikian sebagai suatu struktur data array dapat kita dapat menyusun *Abstract Data Type*-nya (ADT) yang diberi nama Larik sebagai berikut :

Larik
Int size **Object []item**
BuatLarik(int) **setItem(int id, Object dt)** **int getSize()** **int getPos(Object dt)** **int []getPosPos(Object dt)** **Object getMax()** **Object getMin()** **Larik Sort();** **Larik Copy(int id, int n)**

Jika tipe data Object pada ADT di atas diganti dengan tipe data int, maka programnya adalah sebagai berikut :

Program Latihan Praktikum 2.1

```
1  public class Larik{
2     //data (strukur data)
3     private int size;
4     private int []itemDt;
5
6     //method
7     public void buatLarik(int n){
8        this.size = n;
9        this.itemDt = new int[this.size];
10    }
11    public Larik(int n){  buatLarik(n);}
12    public int getSize(){ return this.size;}
13    public Larik(int []dt){
14       buatLarik(dt.length);
15       for (int i=0; i<dt.length; i++)isiItem(i,dt[i]);
16    }
17
18    public void isiItem(int id, int dt){
19       this.itemDt[id] = dt;
20    }
21
```

```
22    public void cetak(String komentar){
23       System.out.println(komentar);
24       for(int i=0; i<this.size; i++){
25          System.out.print(this.itemDt[i]+" ");
26       }
27       System.out.println();
28    }
29    public int findBesar(){
30       int besar = this.itemDt[0];
31       for (int i=1;i<this.size; i++){
32          if (besar < this.itemDt[i]){
33             besar = this.itemDt[i];
34          }
35       }
36       return besar;
37    }
38    /**
39     * program ini mencari posisi suatu data tertentu di larik
40     */
41    public int getPosisi(int dtCari){
42       int pos = -99;
43       boolean ketemu = false;
44       int i=0;
45       while (!ketemu && i<this.size){
46          if (dtCari == this.itemDt[i]){
47             ketemu = true;
48             pos = i;
49          }
50          i++;
51       }
52       return pos;
53    }
54
55    private int getPosMax(int id){
56       int max = this.itemDt[id];
57       int posMax = id;
58       for (int i=id+1;i<size; i++){
59          if (max <= this.itemDt[i]) {
60             max = this.itemDt[i];
61             posMax = i;
62          }
63       }
64       return posMax;
65    }
66
67    private int getPosMin(int id){
68       int min = this.itemDt[id];
69       int posMin = id;
70       for (int i=id+1;i<size; i++){
71          if (min >= this.itemDt[i]) {
72             min = this.itemDt[i];
73             posMin = i;
74          }
75       }
76       return posMin;
77    }
78    public int PencarianBiner(int dtCari,  int awal, int akhir){
```

```
79      int pos = -99;
80      int tengah = (awal+akhir)/2;
81      if(dtCari< this.itemDt[tengah])
82         return PencarianBiner(dtCari, awal, tengah);
83      else if (dtCari > this.itemDt[tengah])
84          return PencarianBiner(dtCari,tengah+1,akhir);
85      else if (dtCari == this.itemDt[tengah]) return tengah;
86      else return pos;
87    }
88    /**
89     * program untuk mencopy isi suatu Larik
90     * mulai dari posisi k sebanyak n item hasilnya
91     * dikeluarkan sebagai array baru
92     */
93    public Larik copyLarik(int k, int n){
94      Larik lHasil = null;
95      if (n <= this.size-k){
96        lHasil = new Larik(n);
97        int j = 0;
98        for (int i=k; i<k+n; i++){
99          lHasil.isiItem(j++, this.itemDt[i]);
100       }
101     }
102     return lHasil;
103   }
104
105   /**
106    * pilihan 0 : urutkan dari kecil ke besar
107    * lainya : urutkan dari besar ke kecil
108    * Algoritma pengurutan ini menggunakan selection sort
109    */
110   public Larik SelectionSort(int pilihan){
111     Larik lsort = copyLarik(0,size);
112
113     for (int i=0; i<lsort.getSize();i++){
114       int posData;
115       if (pilihan == 0) posData = lsort.getPosMin(i);
116       else posData = lsort.getPosMax(i);
117
118       int dt1 = lsort.itemDt[i];
119       int dt2 = lsort.itemDt[posData];
120       lsort.itemDt[i] = dt2;
121       lsort.itemDt[posData] = dt1;
122     }
123     return lsort;
124   }
125
126   public static void main (String[] args) {
127     int []A = {2,34,5,7,10};
128     Larik lA = new Larik(A);
129
130
131     lA.cetak("Sebelum");
132     lA.Sort(0).cetak("Sesudah di sort");
133     //int k = lA.PencarianBiner(34,0,5);
134     //System.out.println(k);
135
```

```
136         Larik lB = lA.Sort(0);
137         int p = lB.PencarianBiner(10,0,5);
138         System.out.println(p);
139        }
140 }
```

Latihan 1.

1. Jalankan program di atas, amati hasilnya.
2. Bukalah tanda komentar pada 133 dan 134, jalankan dan amati hasilnya.
3. Perhatikan pada baris 131 dan 132, cobalah tambahkan perintah berikut pada baris sesudahnya `lA.cetak("Sesudah")`, amati hasilnya. Tuliskan kesimpulan anda.

Tugas 1.

1. Kembangkan program di atas dengan menambahkan method berikut :
   a. Mencari posisi bilangan genap yang terletak diantara bilangan x1 dan bilangan x2 dengan header sebagai berikut :

      ```
      int findPosGenap(int x1, int x2)
      ```

   b. Gunakan algoritma pengurutan yang lain BubleSort dan MergeSort

2. Pada Latihan kedua ini anda diminta untuk melengkapi bagian dari program ADT_Larik sehingga jika diberikan program utama pada gambar 1 akan menghasilkan keluaran sebagaimana gambar 2.

```
    Program Latihan Praktikum 2.2
1   package ADT_Larik;
2   /**
3    *
4    * @author achmad ridok
5    *
6    */
7   public class Larik{
8     //data (strukur data)
9     private int size;
10    private double []itemDt;
```

```
11
12      /**
13       * Contructor untuk membuat ADT larik dari suatu array
14       * @param A : array bertipe int
15       */
16      public Larik(double []A){
17        this.size = A.length;
18        this.itemDt = new double[this.size];
19        for (int i=0; i<this.size; i++){
20          this.itemDt[i] = A[i];
21        }
22      }
23      /**
24       * fungsi untu mendapatkan ukuran larik
25       * @return size dari larik
26       */
27      public int getSize(){
28        return this.size;
29      }
30
31      /**
32       * fungsi untuk mendapatkan item ke i dari suatu larik
33       * @param i : posisi item
34       * @return item larik
35       */
36      public double getItem(int i){
37        return this.itemDt[i];
38      }
39      /**
40       * fungsi static untuk menyambung dua buah larik l1 dan l2
41       * @param l1 : Larik
42       * @param l2 : Larik
43       * @return Larik
44       */
45      public static Larik sambung(Larik l1, Larik l2){
46        // Lengkapi bagian ini
47      }
48
49      /**
50       * procedure untuk isiItem suatu larik
51       * @param id : indeks larik
52       * @param dt : item data yang akan disisipkan
53       */
54      public void isiItem(int id, double dt){
55        this.itemDt[id] = dt;
56      }
57
58      /**
59       * procedure cetak suatu array
60       * @param komentar : String
61       */
62      public void cetak(String komentar){
63        System.out.println(komentar);
64        for(int i=0; i<this.size; i++){
65          System.out.printf("%.2f ",this.itemDt[i]);
66        }
67        System.out.println();
```

```
68    }
69
70    /**
71     * fungsi untuk mendapatkan nilai terbesar dari suatu larik
72     * @return : item tebesar dari larik
73     */
74    public double findBesar(){
75      double besar = this.itemDt[0];
76      for (int i=1;i<this.size; i++){
77        if (besar < this.itemDt[i]){
78          besar = this.itemDt[i];
79        }
80      }
81      return besar;
82    }
83    /**
84     * fungsi untuk mencari posisi suatu data tertentu di array
85     * @param dtCari : data yang akan dicari
86     * @return posisiData
87     */
88    public int getPosisi(double dtCari){
89      int pos = -99;
90      boolean ketemu = false;
91      int i=0;
92      while (!ketemu && i<this.size){
93        if (dtCari == this.itemDt[i]){
94          ketemu = true;
95          pos = i;
96        }
97        i++;
98      }
99      return pos;
100   }
101
102   /**
103    * fungsi static untuk mencopy isi suatu larik l
104    * @param k : posisi awal
105    * @param n : jumlah item yang akan dicopy
106    * @param l : larik asal
107    * @return Larik hasil copy
108    */
109   public static Larik copyLarik(int k, int n, Larik l){
110     // lenkapi bagian ini
111   }
112
113   /**
114    * fungsi untuk mencari posisi terbesar suatu data
115    * suatu posisi awal sampai akhir
116    * @param awal : posisi awal
117    * @param akhir : posisi akhir
118    * @return posisi data terbesar
119    */
120   public  int getPosBesar(int awal, int akhir){
121     int posBesar = -1;
122     double itemBesar;
123     if (awal <= akhir){
124       posBesar = awal;
```

```
125         itemBesar = this.getItem(awal);
126         for (int i=awal+1; i<akhir; i++){
127           double nilaiItem = this.getItem(i);
128           if (itemBesar < nilaiItem){
129             itemBesar = nilaiItem;
130             posBesar = i;
131           }
132         }
133       }
134       return posBesar;
135     }
136
137     /**
138      * fungsi untuk mencari posisi data terkecil suatu array
139      * mulai dari posisi awal sampai posisi akhir
140      * @param awal : posisi awal
141      * @param akhir : posisi akhir
142      * @return posisi data terkecil
143      */
144     public  int getPosKecil(int awal, int akhir){
145       // lenkapi bagian ini
146     }
147
148     /**
149      * fungsi pengurutan suatu larik lAsal dimana kondisi lAsal
150      * akan tetap setelah proses pengurutan
151      * @param lAsal : Array asal yang akan diurutkan
151      * @param status : 0-> urut dari kecil ke besar
152      *              1-> urut dari besar ke kecil
153      * @return Array baru hasil pengurutan
154      */
155     public static Larik SelectionSort(Larik lAsal, int status){
156       int n = lAsal.getSize();
157       Larik lhasil = Larik.copyLarik(0, n, lAsal);
158       if (status == 0){// urutkan data dari kecil ke besar
159         for (int i=0; i<n; i++){
160           int posKecil = lhasil.getPosKecil(i, n);
161           double itemKecil = lhasil.getItem(posKecil);
162           double itemI = lhasil.getItem(i);
163           lhasil.isiItem(i, itemKecil);
164           lhasil.isiItem(posKecil, itemI);
165         }
166       } else { // urutkan data dari besar ke kecil
167         for (int i=0; i<n; i++){
168           int posBesar = lhasil.getPosBesar(i, n);
169           double itemBesar = lhasil.getItem(posBesar);
170           double itemI = lhasil.getItem(i);
171           lhasil.isiItem(i, itemBesar);
172           lhasil.isiItem(posBesar, itemI);
173         }
174       }
175       return lhasil;
176     }
177 }
```

Gambar 1. Potongan program utama

```
public class AppPr1 {
```

```
   public static void main(String[] args) {
      // implementasi untuk ADT_Larik
      double []A = {3,4,1,10,5,2,10,20,16};
      double []B = {4,3,1,11,7};
      Larik L1 = new Larik(A);
      Larik L2 = new Larik(B);

      L1.cetak("L1");
      L2.cetak("L2");
      Larik L3 = Larik.sambung(L1, L2);
      L3.cetak("L3");
      Larik L4 = Larik.copyLarik(0, L1.getSize(), L1);
      L1.cetak("L1");
      L4.cetak("L4");

      Larik L5 = Larik.SelectionSort(L1,0);
      L5.cetak("L5");
      L1.cetak("L1");
      int []posisi = L1.FindPosPos(10);

      double hasil = Larik.LarikKaliLarik(L1, L4);
      System.out.printf("HASIL KALI %.3f\n",hasil);
   }
}
```

Dengan hasil keluaran sebagai berikut :

Gambar 2. Hasil keluaran

```
Isi Larik L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
Isi Larik L2
4.00 3.00 1.00 11.00 7.00
L3 = gabungan dari L1 dan L2
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00 4.00 3.00 1.00 11.00
7.00
Isi Larik L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
L4 Copy dari L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
L5 Hasil pengurutan dari L1 kecil->besar
1.00 2.00 3.00 4.00 5.00 10.00 10.00 16.00 20.00
L6 Hasil pengurutan dari L1 besar->kecil
20.00 16.00 10.00 10.00 5.00 4.00 3.00 2.00 1.00
Isi Larik L1
3.00 4.00 1.00 10.00 5.00 2.00 10.00 20.00 16.00
HASIL KALI Larik L1*L4 = 911.000
```

3

Array 2 Dimensi

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman Abstract Data Type Array 2 dimensi.*

Teori

Array 2 dimensi pada dasarnya adalah pengembangan dari array 1 dimensi, dimana tipe data masing-masing itemnya bertipe array 1 dimensi. Secara sederhana array 2 dimensi dapat didefinisikan sebagai berikut :

tipeData [ ][ ]dt.

ADT Array 1 Dimensi

Applikasi array 2 dimensi banyak digunakan pada representasi masalah matrik. Matrik sebagai representasi data dapat digambarkan ADT sebgai berikut :

Matrik
Int nBaris **Int nKolom** **double [][]itemData**
Matrik(int) **Matrik(double [][]dt)** **Larik getBaris(int)** **Larik getKolom(int)** **Matrik tambah(Matrik)** **Matrik kali(Matrik)** **Tranpos()** **Boolean kesamaan(Matrik)**

Program Latihan Praktikum 3.1

```
1  public class Matrik{
2     // Data
3     private int nBaris, nKolom;
4     private double [][]itemDt;
5
6     public Matrik(int nBrs, int nKlm){
7        nBaris = nBrs;
8        nKolom = nKlm;
9        itemDt = new double[nBaris][nKolom];
10    }
11
12    public Matrik(double [][]dt){
13       nBaris = dt.length;
14       nKolom = dt[0].length;
15       this.itemDt = new double[nBaris][nKolom];
16       for(int i=0; i<nBaris; i++){
17          for (int j=0; j<nKolom; j++){
18             this.setItem(i,j,dt[i][j]);
19          }
20       }
21    }
22    /**
23     * Fungsi untuk mendapatakan jumlah baris
24     * @return jumlah baris
25     */
26    public int getNBaris(){ return nBaris;}
27    public int getNKolom(){ return nKolom;}
28    public double getItem(int idB, int idK){
29       return this.itemDt[idB][idK];
30    }
31    public void setItem(int idB, int idK, double dt){
32       this.itemDt[idB][idK] = dt;
33    }
34
35    public Matrik tambah(Matrik x){
36       Matrik y = null;
37       if ((this.nBaris == x.getNBaris())&&
38            (this.nKolom == x.getNKolom())){
39          y = new Matrik(x.getNBaris(),x.getNKolom());
40          for (int i=0; i<this.nBaris; i++){
41             for (int j=0; j<this.nKolom; j++){
42                y.setItem(i,j, this.itemDt[i][j]+x.getItem(i,j));
43             }
44          }
45       }
46       return y;
47    }
48
49    public void cetak(String kom){
50       System.out.println(kom);
51       for (int i=0; i<this.nBaris; i++){
52          for (int j=0; j<this.nKolom; j++){
53             System.out.print(this.itemDt[i][j]+" ");
54          }
55          System.out.println();
```

```
56        }
57      }
58
59      public static void main (String[] args) {
60        // implementasi untuk ADT_Matrik
61        Matrik A,B,C;
62
63        double [][] X = {{1, 2, 3},{2, 14, 5},{16, 8, 13}};
64        double [][] Y = {{10, 12, 0},{5, 1, 5},{3, 1, 10}};
65
66        A = new Matrik(X);
67        B = new Matrik(Y);
68
69        A.cetak("Matrik A");
70        B.cetak("Matrik B");
71
72        C = A.tambah(B);
73        C.cetak("Matrik C = A+B");
74
75        Larik lb,lk;
76        lb = D.getBaris(1);
77        lb.cetak("Larik  baris ke-1");
78        lk = D.getKolom(1);
79        lk.cetak("Larik kolom ke-1");
80       }
81  }
```

Latihan 3.

1. Jalankan program di atas, amati hasilnya.

Tugas 3.1.

1. Tambahkan method untuk mengubah dari ADT Matrik ke struktur array dua dimensi dengan header :

   ```
   double [][] toDArray()
   ```

2. Tambahkan method untuk transpose matrik dengan header

   ```
   Matrik tranposeMatrik()
   ```

3. Tambahkan method untuk perkalian matrik dengan header

   ```
   Matrik kali(Matrik m)
   ```

   Interface pemanggilan method di atas sebagai berikut :

   Misalkan dideklarasikan Matrik A,B, C;

   C = A.kali(B)

4. Tambahkan method untuk mengambil isi baris tertentu dan is kolom tertentu dari matrik dan hasilnya dikeluarkan berupa larik dengan header sebagai berikut :

```
Larik getKolom(int idK)

Larik getBaris(int idK)
```

Untuk dapat menjalankan ini tambahkan program Larik pada praktikum 1 diatas program ini atau anda susun dalam bentuk paket (minta petunjuk pada asisten).
Karena larik pada program 1 tipe itemnya integer maka agar kompatibel dengan program matrik ada dua cara :

- Lakukan casting (minta petunjuk pada asisten)
- Anda edit tipe data itemDt pada larik bertipe fload

Program Latihan Praktikum 3.2

```
1  package ADT_Larik;
2
3  /**
4   * ADT Matrik
5   * @author achmad ridok
6   */
7  public class Matrik{
8    private int nBaris, nKolom;
9    private double [][]itemDt;
10   /**
11    * constructor untuk membuat suatu matrik
12    * @param nBrs : banyaknya baris
13    * @param nKlm : banyaknya kolom
14    */
15   public Matrik(int nBrs, int nKlm){
16     nBaris = nBrs;
17     nKolom = nKlm;
18     itemDt = new double[nBaris][nKolom];
19   }
20   /**
21    * constructor untuk membuat matrik dari array 2 dimensi
22    * @param A : array dua dimensi
23    */
24   public Matrik(double [][]A){
25     this(A.length,A[0].length); // panggil contructor
26     this.nBaris = A.length;
27     this.nKolom = A[0].length;
28
29     for (int i=0; i<nBaris; i++){
30       for (int j=0; j<nKolom; j++){
31         this.itemDt[i][j] = A[i][j];
32       }
33     }
34   }
35   /**
```

```
36      * Fungsi untuk mendapatakan jumlah baris
37      * @return jumlah baris
38      */
39     public int getNBaris(){ return nBaris;}
40     public int getNKolom(){ return nKolom;}
41     public double getItem(int idB, int idK){
42        return this.itemDt[idB][idK];
43     }
44     public void setItem(int idB, int idK, double dt){
45        this.itemDt[idB][idK] = dt;
46     }
47     /**
48      * fungsi tambah antara dua matrik A dan B
49      * @param A : Matrik
50      * @param B : Matrik
51      * @return Matrik hasil
52      */
53     public static Matrik tambah(Matrik A, Matrik B){
54         // tambahkan bagian ini
55     }
56
57     /**
58      * fungsi static perkalian antara vektor dengan matrik
59      * Syarat : lebar L sama dengan jumlah baris M
60      * @param L : Vector (Larik)
61      * @param M : Matrik
62      * @return Vector (Larik) berdimensi nKolom dari M
63      */
64     public static Larik VektorKaliMatrik(Larik L, Matrik M){
65       Larik lHasil = null;
66       Larik lKolom = null;
67       if (L.getSize() == M.getNBaris()){
68          lHasil = new Larik(M.getNKolom());
69          for (int i=0; i<M.getNKolom(); i++){
70             lKolom = M.getKolom(i);
71             double hasil = Larik.LarikKaliLarik(L, lKolom);
72             System.out.println(hasil);
73             lHasil.isiItem(i, hasil);
74          }
75       }
76       return lHasil;
77     }
78
79     /**
80      * fungsi static tranpos suatu matrik
81      * @param A : Matrik
82      * @return Matrik tranpos
83      */
84     public static Matrik tranpos(Matrik A){
85       // lenkapi bagian ini
86     }
87     /**
88      * fungsi untuk mendapatkan vektor baris dari matrik
89      * @param idBaris : indek baris yang akan diekstrak
90      * @return Larik representasi baris
91      */
92     public Larik getBaris(int idBaris){
```

```
93      // lenkapi bagian ini
94    }
95
96    /**
97     * fugsi untuk mendapatkan vektor kolom suatu matrik
98     * @param idKolom : id kolom yang akan diekstrak
99     * @return Larik representasi kolom
100    */
101   public  Larik getKolom(int idKolom){
102     Larik l = new Larik(this.nBaris);
103     for (int i=0; i<this.nBaris; i++){
104       double itemKolom = this.getItem(i, idKolom);
105       l.isiItem(i, itemKolom);
106     }
107     return l;
108   }
109
110   /**
111    * procedure cetak
112    * @param kom
113    */
114   public void cetak(String kom){
115     System.out.println(kom);
116     for (int i=0; i<this.nBaris; i++){
117       for (int j=0; j<this.nKolom; j++){
118         System.out.printf("%.2f ",this.itemDt[i][j]);
119       }
120       System.out.println();
121     }
122   }
```

Tugas 3.2. Lengkapi program di atas sehingga dengan potongan program driver berikut akan menghasilkan keluaran sebagaimana disamping

```
Matrik A,B,C;
double [][]data1 = {{1,2,3},{3,4,7}};
double [][]data2 = {{4,5,1},{6,1,9}};
A = new Matrik(data1);
B = new Matrik(data2);
A.cetak("A"); B.cetak("B");
C = Matrik.tambah(A,B);
C.cetak("C");
Matrik CT = Matrik.tranpos(C);
CT.cetak("Tsanpos");
Larik l1 = C.getBaris(1);
l1.cetak("Baris ke 1 dari C");
Larik l2 =
Matrik.VektorKaliMatrik(l1,CT);
l2.cetak("Hasil kali C.L1");
```

```
A
1.00 2.00 3.00
3.00 4.00 7.00
B
4.00 5.00 1.00
6.00 1.00 9.00
C
5.00 7.00 4.00
9.00 5.00 16.00
Tsanpos
5.00 9.00
7.00 5.00
4.00 16.00
Baris ke 1 dari C
9.00 5.00 16.00
144.0
362.0
Hasil kali C.L1
144.00 362.00
```

4

Stack

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman Abstract Data Type Stack.*

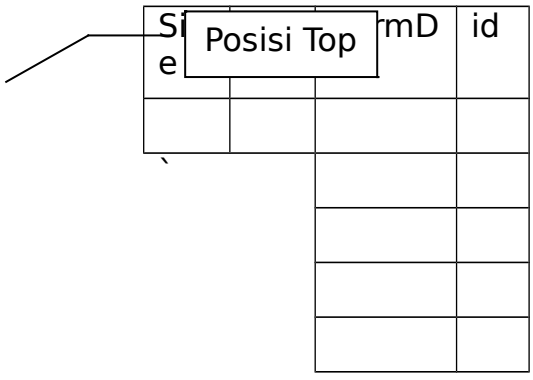
Teori

Stack atau tumpukan adalah representasi data yang meniru cara kerja suatu tumpukan dalam dunia nyata. Prinsip tumpukan adalah barang yang masuk terlebih dahulu dia akan keluar belakangan atau sebaliknya. Prinsip ini dikenal dengan LIFO (*Last In First Out*). Sesuai dengan perilaku masalah ini dalam dunia nyata, pemasukan kedalam tumpukan disebut push dan pengambilannya disebut pop.

Implementasi stack dapat dilakukan menggunakan array atau menggunakan linkedlist.

ADT Stack denga Array

ADT suatu stack menggunakan array dapat digambarkan sebagai berikut :



Dengan model representasi stack seperti di atas maka dapat disusun ADT Stack sebagai berikut :

Stack
Int size Int pos Object []temData
Stack(int) boolean isFull() boolean isEnpty() push(Object dt) Object pop()

Jika tipe Object diganti dengan tipe int maka program stack sebagai berikut :

Program Latihan Praktikum 4

```
1  public class Stack {
2     // Struktur Data
3     private int size;
4     private int top;
5     private int []data;
6
7     // method
8     public Stack(int n){
9        top = -1;
10       size = n;
11       data = new int[size];
12    }
13
14    public boolean isPalindrome(){
15       return true;
16    }
17    public boolean isFull(){
18       return top == (size-1) ? true : false;
19       //if (top == size-1)return true;
20       //else return false;
21    }
22    public boolean isEmpty(){
23       return top == -1 ? true : false;
24       //if (top == -1) return true;
25       //else return false;
26    }
27    public void push(int dt){
28       if (!isFull()){
29          data[++top] = dt;
30       }
31    }
32    public int pop(){
33       int hasil=-999;
34       if (!isEmpty()){
35          hasil = data[top--];
36       }
37       return hasil;
38    }
39
40    public static void main (String[] args) {
41       Stack st = new Stack(3);
```

```
42        st.push(0);
43        st.push(6);
44        st.push(7);
45        while (!st.isEmpty()){
46           System.out.println(st.pop());
47        }
48
49        //app stack
50        int nilai = 1234;
51        Stack s = new Stack(100);
52        while (nilai != 0){
53           int sisa = nilai % 2;
54           s.push(sisa);
55           nilai = nilai/2;
56        }
57        while (!s.isEmpty()){
58           System.out.print(s.pop());
59        }
60      }
61  }
```

Latihan 4.

1. Jalankan program di atas, amati hasilnya.
2. Modifikasilan baris `21 : data[++top] = dt` menjadi `data[top++]`, amati hasilnya, mengapa demikian ?
3. App stack pada baris 50 s.d 60 mengerjakan masalah apa?

Tugas 4

1. Buat program untuk mengkonvesi dari bilang desimal ke representasi bilangan biner menggunakan program stack di atas.
2. Kembangkan program di atas dengan membuat stack yang berisi Object sehingga isi Stack dapat diisi sembarang object seperti object Double, objeck Buku dan lain sebaginya. Dengan potongan program driver sebagaimana gambar 5 dan class Buku sebagaimna gambar 6 akan menghasilkan keluaran sebagaimana gambar 7.

Gambar 5. Driver program ADT Stack Object

```
public class AppStackObject {
   public static void main(String[] args) {
      //implementasi Stack
```

```
      StackObject st = new StackObject(3);
      st.push(new Double(5));
      st.push(new Double(8));
      st.push(new Double(7));
      while (!st.isEmpty()){
         System.out.println(st.pop());
      }

      StackObject stBuku = new StackObject(3);
      stBuku.push(new Buku("Java","Anton"));
      stBuku.push(new Buku("Algoritma dan STD","Achmad"));
      stBuku.push(new Buku("C++","Budi"));
      while (!stBuku.isEmpty()){
         System.out.println(stBuku.pop());
      }
   }
}
```

Gambar 6. Class Buku

```
Class Buku{
   private String judul;
   private String pengarang;
   public Buku(String jdl, String peng){
      this.judul = jdl;
      this.pengarang = peng;
   }
   public String toString(){
      return String.format("%s %s", this.judul, this.pengarang);
   }
}
```

Gambar 6. Keluaran program

```
   7.0
8.0
5.0
C++ Budi
Algoritma dan STD Achmad
Java Anton
```

5

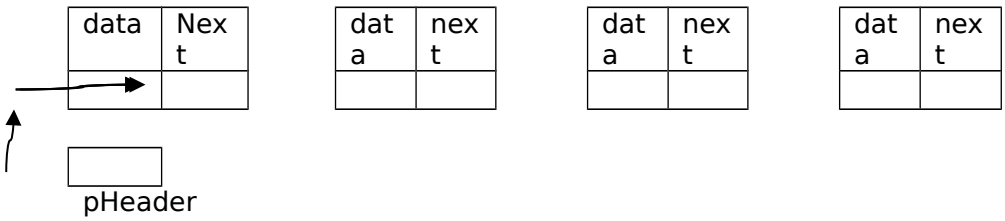
Single Linked List

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman Abstract Data Type Linked List.*

Teori

Linked List merupakan representasi data yang disimpan dalan suatu rangkaian node. Dalam setiap node menyimpan dua informasi utama yakni data dan pointer yang menunjuk ke node yang lain sehingga membentuk rangkaian node sebagaimana digambarkan berikut :



Proses pengaksesan data pada Linked List pada dasarnya sama dengan stact yakni LIFO. Namun demikian LL tidak sekaku stack, pada LL dapat disisipkan dimana saja sesuai persyaratan yang dikehendaki didepan ditengah atau dibelakang.

ADT Single Linked-list

Dari ilustrasi gambar di atas ADT single Linked-list dapat direpresentasikan sebagai berikut

SLL
Node pHeader
SLL() buatNode(Object) sisip(Object, Object) Object hapusDiDepan() cetak() sisipDtUrut(Object)

Program Latihan Praktikum 5

```
1  package ADT_LIST_BARU;
2  /**
3   * ADT_SingleLinkedList
4   * @author achmad ridok
5   */
6
7  class Node {
8      int   data;
9      Node  next;
10 }
11
12 public class SingleLinkedList {
13     private Node pointer;
14     // contructor LL
15     public SingleLinkedList() {
16         pointer = null;
17     }
18     // membuat suatu node baru
19     public void buatNode(int dt) {
20         Node nodeBaru = new Node();
21         nodeBaru.data = dt;
22         nodeBaru.next = pointer;
23         pointer = nodeBaru;
24     }
25     // menambah data dt1 setelah data dt2 dalam LL
26     public boolean sisip(int dt1, int dt2){
27      Node n = pointer;
28      while ((n != null) && (n.data != dt2))
29         n = n.next;
30      if (n == null) return false;
31      Node nn = new Node();
32      nn.data = dt1;
33      nn.next = n.next;
34      n.next = nn;
35      return true;
36     }
37     // secara normal data dihapus di depan
38     public int hapusDiDepan(){
39      Node hapus = pointer;
40      pointer = pointer.next;
41          return hapus.data;
42     }
43     public void sisipDataDiAkhir(int data){
44      Node pSblAkhir, pAkhir;
45      pSblAkhir = null;
46      pAkhir = pointer;
47      Node baru = new Node();
48      baru.data = data;
49      baru.next = null;
50      while(pAkhir != null){
51         pSblAkhir = pAkhir;
52         pAkhir = pAkhir.next;
53      }
54      pSblAkhir.next = baru;
55     }
```

```
56      public void sisipDataUrut(int data){
57        // lengkapi bagian ini
58      }
59      public void hapusData(int dataHapus){
60       // lenkapai bagian ini
61      }
62      public Node getPointer(){
63       return pointer;
64      }
65      public static SingleLinkedList gabung(SingleLinkedList L1,
66                                            SingleLinkedList 2){
67
68           // lengkapi bagian ini
69      }
70      // cetak data
71      public void cetak(String kom) {
72          System.out.println(kom);
73          Node n = pointer;
74          while (n!= null) {
75              System.out.print(n.data+"->");
76              n = n.next;
77          }
78          System.out.println("NULL");
79      }
80      public static void main(String[] args) {
81       SingleLinkedList l = new SingleLinkedList();
82          l.buatNode(11);
83          l.buatNode(2);
84          l.buatNode(30);
85          l.buatNode(14);
86          l.buatNode(5);
87          l.buatNode(16);
88          l.cetak("l : LL Asal");
89          l.sisipDataDiAkhir(56);
90          l.cetak("l : LL setelah sisip di akhir");
91          System.out.println(l.hapusDiDepan());
92          l.cetak("l : LL setelah dihapus di depan");
93          l.hapusData(30);
94          l.cetak("LL setelah 30 dihapus");
95          SingleLinkedList l2 = new SingleLinkedList();
96          l2.sisipDataUrut(5);
97          l2.sisipDataUrut(1);
98          l2.sisipDataUrut(21);
99          l2.sisipDataUrut(3);
100         l2.sisipDataUrut(9);
101         l2.sisipDataUrut(16);
102         l2.sisipDataUrut(12);
103         l2.cetak("L2 : LL terurut");
104         SingleLinkedList L3 = SingleLinkedList.gabung(l, l2);
105         L3.cetak("L3 : L gabungan L1 dan L2");
106     }
107 }
```

Latihan 5.

Lengkapi bagian program di atas sehingga akan menghasilkan keluaran sebagai berikut :

```
l : LL Asal
16->5->14->30->2->11->NULL
l : LL setelah sisip di akhir
16->5->14->30->2->11->56->NULL
16
l : LL setelah dihapus di depan
5->14->30->2->11->56->NULL
LL setelah 30 dihapus
5->14->2->11->56->NULL
L2 : LL terurut
21->16->12->9->5->3->1->NULL
L3 : L gabungan L1 dan L2
5->14->2->11->56->21->16->12->9->5->3->1->NULL
```

Tugas 5.

1. Modifikasilah program Latihan 4 di atas sehingga SLL dapat menampung sembarang object. Untuk itu anda perlu membuat class baru bernama Mahasiswa dengan data dan method sebagai berikut :

Mahasiswa
String nim **String nama** **double ipk**
Constructor Mahasiswa **double getIpk** **String getNim** **String getNama**

2. Untuk penyisipan data secara urut gunakan pengurutan berdasarkan ipk.
3. Kembangkan program stack bilangan bulat menggunakan SLL.

6

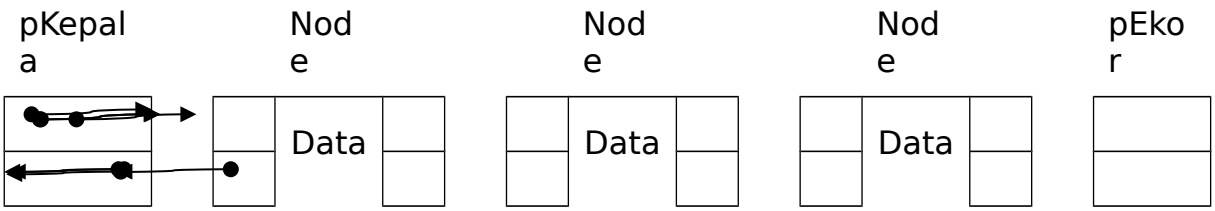
Double Linked List

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman Abstract Data Type Double Linked List.*

Teori

Double Linked List merupakan representasi data yang disimpan dalan suatu rangkaian node dimana setiap node mempunyai penunjuk ke node sebelumnya dan sesudahnya. Ilstrasi double Linked List dapat digambarkan berikut :



ADT Double Linked-list

Dari ilustrasi gambar di atas ADT Double Linked-list dapat direpresentasikan sebagai berikut

DLL
Node pKepala, pEkor
public void sisipDiHeader(int dt) public void sisipDiTrailer(int dt) public void dll.sisipDtTertentu(int ,int) public void hapusDiHeader() public void hapusDiTrailer()

Jika tipe Object diganti dengan tipe int maka program stack sebagai berikut :

Program Latihan Praktikum 6.1

```
1  package ADT_LIST_BARU;
2
3  class NodeDLL{
4     int data;
5     NodeDLL prev,next;
6  }
7
8  public class DoubleLinkedList {
9     private NodeDLL pKepala, pEkor;
10    public DoubleLinkedList(){
11       pKepala = null;
12       pEkor = null;
13    }
14
15    public void sisipDipKepala(int dt){
16       NodeDLL baru = new NodeDLL();
17       baru.data = dt;
18       if (pKepala == null) {
19          baru.prev = pKepala;
20          baru.next = pEkor;
21          pKepala = baru;
22          pEkor = baru;
23       } else {
24          baru.next = pKepala;
25          pKepala.prev = baru;
26          pKepala = baru;
27       }
28    }
29
30    public void sisipDipEkor(int data){
31       NodeDLL baru = new NodeDLL();
32       baru.data = data;
33       if (pEkor == null) {
34          baru.prev = pKepala;
35          baru.next = pEkor;
36          pKepala = baru;
37          pEkor = baru;
38       } else {
39          baru.prev = pEkor;
40          pEkor.next = baru;
41          pEkor = baru;
42          baru.next = pEkor;
43
44       }
45    }
46
47    public void cetak(String kom){
48       System.out.println(kom);
49       NodeDLL p = pKepala;
50       while (p!=pEkor.next){
51          System.out.print(p.data+"->");
52          p = p.next;
53       }
54       System.out.println();
55    }
56
```

```
57      public void hapusDataTertentu(int dataHapus){
58         // lengkapi bagian ini
59      }
60
61      public void sisipDataTerurut(int data){
62         // lengkapi bagian ini
63      }
64
65      public static void main(String s[]){
66
67         DoubleLinkedList dll = new DoubleLinkedList();
68
69         dll.sisipDipKepala(10);
70         dll.sisipDipKepala(20);
71         dll.sisipDipKepala(30);
72         dll.cetak("Isi DLL setelah sisip data di pKepala");
73
74         dll.sisipDipEkor(55);
75         dll.sisipDipEkor(56);
76         dll.sisipDipEkor(57);
77         dll.cetak("Isi DLL setelah sisip data di pEkor");
78
79         dll.hapusDataTertentu(55);
80         dll.cetak("Isi DLL setelah data 55 dihapus");
81
82         DoubleLinkedList dll2 = new DoubleLinkedList();
83         dll2.sisipDataTerurut(30);
84         dll2.sisipDataTerurut(10);
85         dll2.sisipDataTerurut(25);
86         dll2.sisipDataTerurut(100);
87         dll2.sisipDataTerurut(20);
88         dll2.cetak("Isi dll2 sisip data terurut");
89      }
    }
```

Latihan 6

Lengkapilan program di atas sehingga jika program tersebut dijalankan akan menghasilkan keluaran sebagai berikut :

Sebelum melengkapi potongan program di atas gambarkan terlebih dahulu scenario persoalan yang akan dipecahkan kemudian susun skema, flowchart/psudocode kemudian implementasikan.

```
Isi DLL setelah sisip data di pKepala
30->20->10->
Isi DLL setelah sisip data di pEkor
30->20->10->55->56->57->
Isi DLL setelah data 55 dihapus
30->20->10->56->57->
Isi dll2 sisip data terurut
10->20->25->100->
```

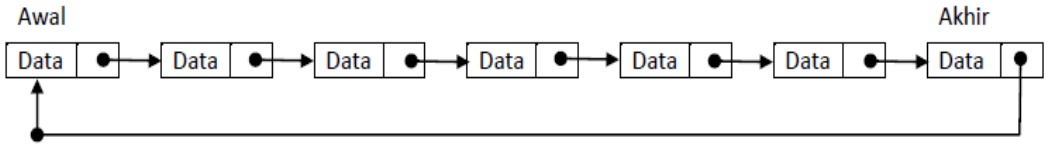
Tugas 6.

Modfikasilah program di atas dengan mengganti tipe item data pada NodeDLL dengan Object. Kemudian gunakan class Masiswa yang pernah dibuat pada praktikum 4 untuk diisikan pada DLL. Gunakan pemasukan data secara interaktif.

7

1. Sirkular Linked List

Salah satu varian dari Double Linked List adalah sirkular Linked List yang mempunyai struktur sebagaimana digambarkan berikut :



ADT Cirkular Single Linked List

SirkularLinkedList
Node head
SirkularLinkedList ()
public boolean apakahKosong()
public boolean tidakKosong()
public Object getPertama()
public void sisipDtDiAkhir(Object dt)
public void sisipDtDiAwal(Object dt)
public void cetakDt(String komentar)

Jika tipe Object diganti dengan tipe int maka program stack sebagai berikut :

	Program Latihan Praktikum 7.1
1	`class NodeCSLL {`
2	`  Object data;`
3	`  NodeCSLL next;`
4	`}`
5	`public class CircularSingleLinkedList {`
6	`  private NodeCSLL pAwal, pAkhir;`
7	`  private int counter;`
8	`  public CircularSingleLinkedList(){`
9	`  pAwal = null;`
10	`  pAkhir = null;`
11	`  counter = -1;`
12	`  }`
13	`  public void SisipDataDiAwal(Object data){`
14	`  NodeCSLL pBaru = new NodeCSLL();`
15	`  pBaru.data = data;`
16	`  pBaru.next = null;`
17	`  if (pAwal == null){`
18	`  pAwal = pBaru;`

```
19          pAkhir = pBaru;
20          pBaru.next = pAwal;
21          counter = 0;
22       } else {
23          pBaru.next = pAwal;
24          pAwal = pBaru;
25          pAkhir.next = pAwal;
26          counter++;
27       }
28    }
29    public void hapusData(Object dtHapus){
30       if(pAwal != null) {
31          NodeCSLL pSbl, pKini,pHapus;
32          pSbl = null; pKini = pAwal;
33          boolean ketemu = false;
34          int i=-1;
35          if (pAwal != null) i = 0;
36          while(!ketemu && (i <= counter) && (i != -1) ){
37             if (pKini.data.equals(dtHapus)) {
38                ketemu = true;
39             }
40             else {
41                pSbl = pKini;
42                pKini = pKini.next;
43             }
44             i++;
45          }
46          if (ketemu){
47             if(pSbl == null) {
48                pHapus = pAwal;
49                pHapus = null;
50             } else {
51                pSbl.next = pKini.next;
52                pHapus = pKini;
53                pHapus = null;
54             }
55             this.counter--;
56          }
57       }
58    }
59     public Object hapusSatuDataDiAkhir(){
60         // lengkapi bagian ini
61    }
62    public void cetak(String Komentar){
63       System.out.println(Komentar);
64       NodeCSLL pCetak;
65       pCetak = pAwal;
66       int i=-1;
67       //if (pCetak != null) i = 0;
68       //while((i < counter) && (i != -1) ){
69       while((i < counter) ){
70          System.out.print(pCetak.data+"->");
71          pCetak = pCetak.next;
72          i++;
73       }
74       System.out.println();
75    }
```

```
76    public static void main(String[] args) {
77      CircularSingleLinkedList csll =
78                    new CircularSingleLinkedList();
79      csll.SisipDataDiAwal(new Integer(50));
80      csll.SisipDataDiAwal(new Integer(60));
81      csll.SisipDataDiAwal(new Integer(70));
82      csll.SisipDataDiAwal(new Integer(8));
83      csll.SisipDataDiAwal(new Integer(9));
84      csll.SisipDataDiAwal(new Integer(90));
85      csll.SisipDataDiAwal(new Integer(19));
86      csll.cetak("csll Asal");
87      csll.hapusData(8);
88      csll.cetak("csll stl 8 di hapus");
89      csll.hapusData(90);
90      csll.cetak("csll stl 90 di hapus");
91    }
92  }
```

Latihan 7

Compilasi dan jalankan program di atas sehingga menghasilkan keluaran sebagai berikut:

```
9->90->9->8->70->60->50->
csll stl 8 di hapus
19->90->9->70->60->50->
csll stl 90 di hapus
19->9->70->60->50->
Data di hapus 50
```

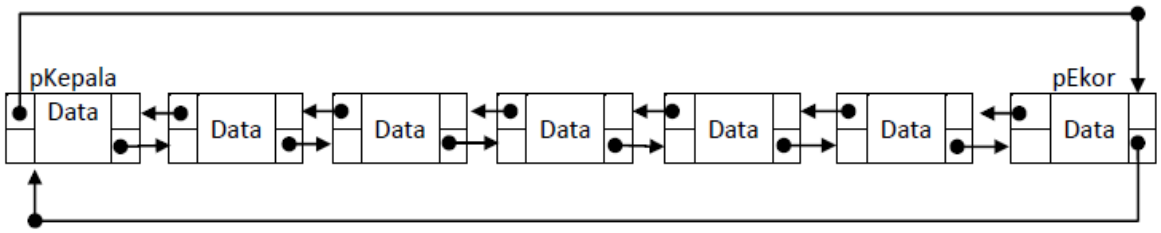
Tugas 7

1. Tambahkan fungsi untuk menghapus data dari Circular Single Linked List dari posisi Akhir dengan header sub program sebagai berikut

```
public Object hapusSatuDataDiAkhir(){
  // lengkapi bagian ini
}
```

2. Kembangkan program 7.1 dengan mengisikan data mahasiswa sebagaimana pada latihan 4 pada circular single linked list.
3. Tambahkan fungsi atau procedure untuk menampilkan data mahasiswa dari circular single linked list yang mempunyai ipk > 3.
4. Kembangkan Circular Single Linked List di atas menjadi Circukar Doubke Linked List sebagaimana gambar struktur berikut :

Gambar 7. Circular Double Linked List



8

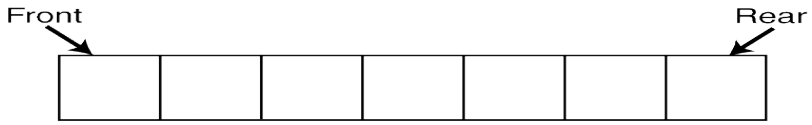
Antrian

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman Abstract Data Type Antrian.*

Teori

Antrian (Queue) merupakan representasi data yang hanya memperbolehkan pengaksesan data pada dua ujung. Penyisipan data dilakukan dibelakan (ekor) dan pengeluaran data dilakukan diujung (kepala). Berbeda dengan double lingked list pada praktikum 5 yang diperbolehkan mengakses data di sembarang tempat. Perilaku seperti ini meniru kejadian pada masalah antrian pada dunia nyata yakni yang pertama masuk dialah yang dilayani duluan (FIFO). :



Dua operasi pada antrian yakni enQueue dan deQueue. Untuk menyisipkan data pada antrian menggunakan enQueue dan menghapus data dari antrian menggunakan deQueue.

ADT Antrian

Dari ilustrasi gambar di atas ADT antrian dapat direpresentasikan sebagai berikut

Queue
List listAntrian
Queue() enqueue(Object object) Object dequeue() boolean kosong() public void cetak()

	Program Latihan Praktikum 8.1
1	**pulic class Node {**
2	**Object data;**
3	**Node next;**
4	

```
5       Node( Object object ){this ( object, null );}
6
7       Node( Object object, Node node ){
8          data = object;
9          next = node;
10      }
11
12      Object getObject(){return data;}
13
14      Node getNext() {return next;}
15   }
```

Program Latihan Praktikum 8.2

```
1    public class List {
2       private Node nodeAwal;
3       private Node nodeAkhir;
4       private String nama;
5
6       public List(){this( "list" ); }
7
8       public List( String namaList ){
9          nama = namaList;
10         nodeAwal = nodeAkhir = null;
11      }
12
13      public void sisipDiAwal( Object dt ){
14         if (kosong()) nodeAwal = nodeAkhir = new Node( dt );
15         else nodeAwal = new Node( dt, nodeAwal );
16      }
17
18      public void sisipDiAkhir( Object dt ){
19         if (kosong()) nodeAwal = nodeAkhir = new Node( dt );
20         else nodeAkhir = nodeAkhir.next = new Node( dt );
21      }
22
23      public Object hapusDrDepan(){
24        Object itemDihapus = null;
25         if (!kosong()) {
26            itemDihapus = nodeAwal.data;
27            if ( nodeAwal == nodeAkhir )
28                nodeAwal = nodeAkhir = null;
29            else nodeAwal = nodeAwal.next;
30         }
31
32         return itemDihapus;
33      }
34
35      public boolean kosong(){return nodeAwal == null;}
36
37      public void cetak(){
38        if ( kosong() ){
39           System.out.printf( "Kosong %s\n", nama );
40           return;
41        }
42
43        System.out.printf( "Isi %s adalah : ", nama );
```

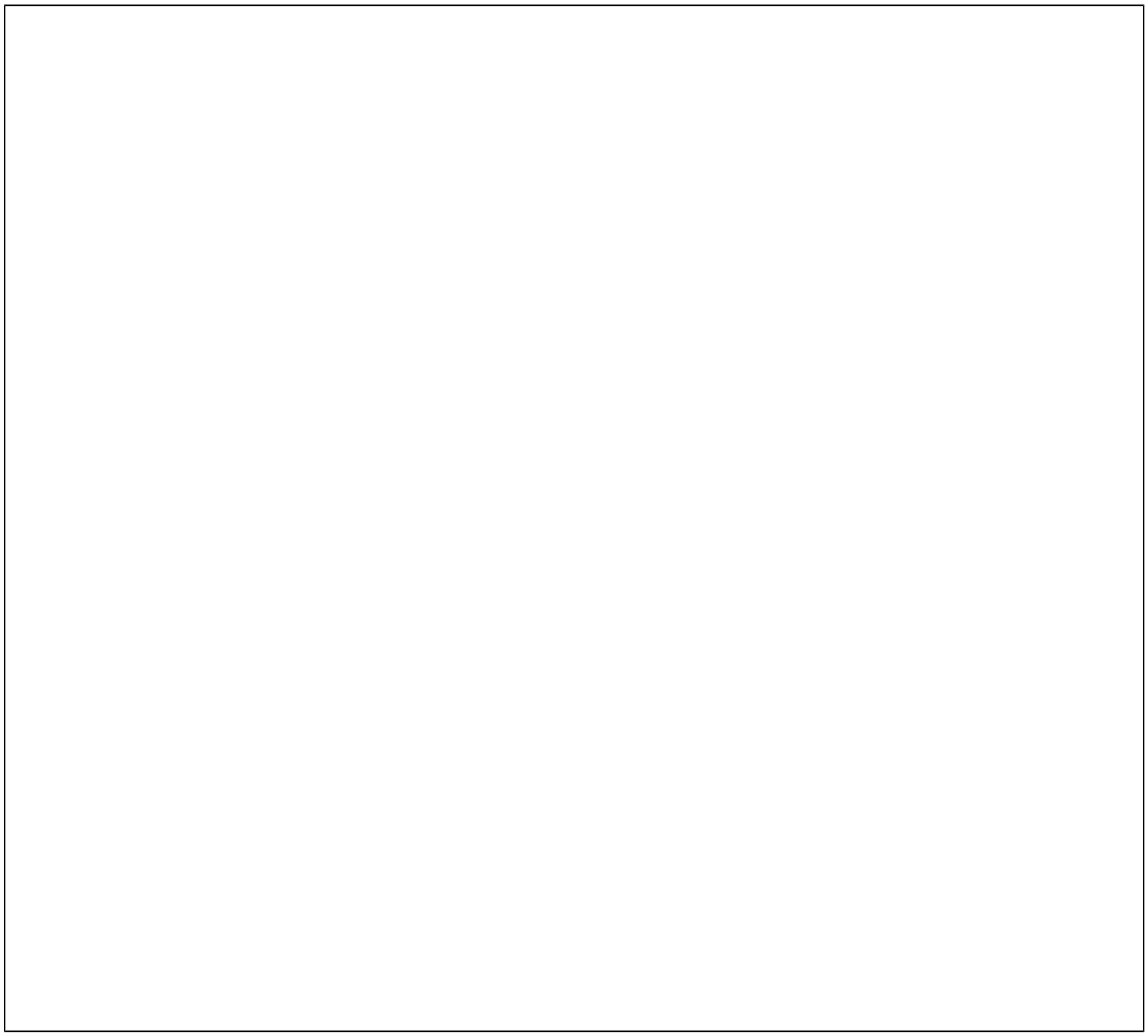
```
44      Node kini = nodeAwal;
45
46      while ( kini != null ){
47         System.out.printf( "%s ", kini.data );
48         kini = kini.next;
49      }
50
51      System.out.println( "\n" );
52    }
53  }
```

Program Latihan Praktikum 8.3

```
1   public class Queue {
2       private List listAntrian;
3       public Queue() {
4          listAntrian = new List( "queue" );
5       }
6
7       public void enqueue( Object object ){
8          listAntrian.sisipDiAkhir( object );
9       }
10
11      public Object dequeue(){
12         return listAntrian.hapusDrDepan();
13      }
14
15      public boolean kosong(){
16         return listAntrian.kosong();
17      }
18
19      public void cetak(){listAntrian.cetak();}
20
21      public static void main( String args[]){
22         Queue q = new Queue();
23         q.enqueue( 10 );
24         q.cetak();
25         q.enqueue( 40 );
26         q.cetak();
27         q.enqueue( 25 );
28         q.cetak();
29         q.enqueue( 30 );
30         q.cetak();
31
32         Object dtHapus = null;
33         while(!q.kosong()){
34             dtHapus = q.dequeue();
35             System.out.printf("%s dihapus \n",dtHapus );
36             q.cetak();
37         }
38       }
39   }
```

Latihan 8

1. Susunlah ketiga program praktikum 8.1, 8.2 dan 8.3 dengan nama file sama dengan nama classnya. Kemudian kompilasi dan jalankan dan tampilkan hasilnya pada kotak berikut :



Tugas 8

1. Kembangkan program antrian dengan menggunakan ADT Cicular Single Linked List yang telah dibuat pada praktikum 7. Perhatikan bahwa sifat antrian adalah **FIFO : First In First Out**
2. Susunlah program untuk simulasi Round Robin. Setiap proses deQueue() dilakukan pada suatu node maka isi suatu node berkurang 1, kalau nilainya sudah sama dengan 0 maka node tersebut dihapus dari List jika lebih besar dari 0 node tersebut diantrikan lagi untuk diperoses berikutnya.

9

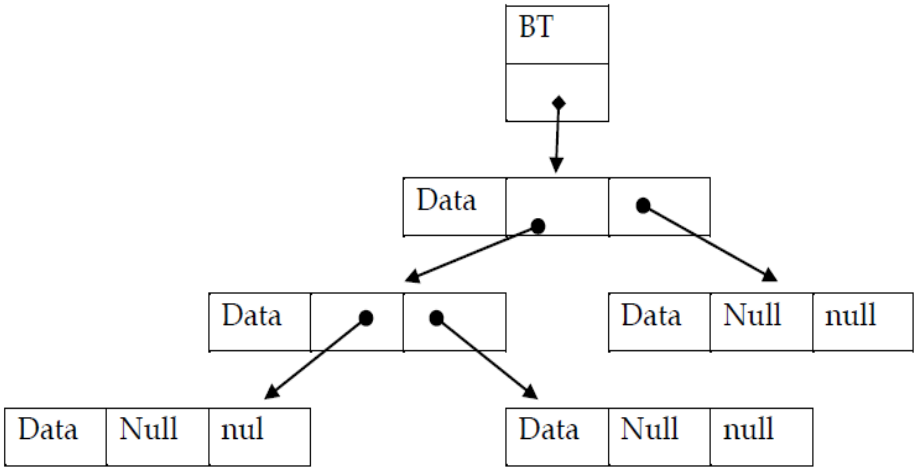
Binary Tree

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman Abstract Data Type Binary Tree.*

Teori

Binary Tree adalah non linear linked list dimana masing-masing nodenya menunjuk paling banyak 2 node yang lain. Gambaran pohon biner ini seperti gambar berikut :



ADT Binary Tree

Dari ilustrasi gambar di atas ADT antrian dapat direpresentasikan sebagai berikut

List
Node root;
Tree() sisipDtNode(int dtSisip) preorderTraversal() inorderTraversal() postorderTraversal()

Program Latihan Praktikum 9.1

```
1  import java.util.Random;
2  class Node{
3        int data;
4        Node nodeKiri;
5        Node nodeKanan;
6
7        public Node(int dt){
8           data = dt;
9           nodeKiri = nodeKanan = null;
10       }
11
12       public void sisipDt(int dtSisip ){
13          if (dtSisip < data){
14             if(nodeKiri == null)
15                nodeKiri = new Node( dtSisip );
16             else nodeKiri.sisipDt( dtSisip );
17          }
18          else if(dtSisip > data){
19             if ( nodeKanan == null )
20                nodeKanan = new Node(dtSisip);
21             else nodeKanan.sisipDt(dtSisip);
22          }
23       }
24    }
25
26    public class Tree{
27       private Node root;
28
29       public Tree() {
30          root = null;
31       }
32
33       public void sisipDtNode(int dtSisip){
34        if (root == null)
35              root = new Node( dtSisip );
36          else
37              root.sisipDt(dtSisip);
38       }
39
```

```
40        public void preorderTraversal() {
41           preorder(root);
42        }
43
44        private void preorder(Node node){
45           if(node == null) return;
46
47           System.out.printf( "%d ", node.data );
48           preorder(node.nodeKiri);
49           preorder(node.nodeKanan);
50        }
51
52        public void inorderTraversal(){
53           inorder( root );
54        }
55
56        private void inorder(Node node){
57           if (node == null) return;
58
59           inorder(node.nodeKiri);
60           System.out.printf( "%d ", node.data );
61           inorder( node.nodeKanan );
62        }
63
64        public void postorderTraversal(){
65            postorder( root );
66        }
67
68       private void postorder(Node node){
69            if (node == null) return;
70
71            postorder(node.nodeKiri);
72            postorder(node.nodeKanan);
73            System.out.printf( "%d ", node.data );
74          }
75
76       public static void main(String args[]) {
77            Tree Tree = new  Tree();
78            int nilai;
79            Random randomNumber = new  Random();
80
81            System.out.println( "sisip nilai data berikut : " );
82
83            // sisipDt 10 bilangan acak dari 0-99 ke dalam tree
84            for ( int i = 1; i <= 10; i++ ) {
85              nilai = randomNumber.nextInt( 100 );
86              System.out.print(nilai + " " );
87              Tree.sisipDtNode(nilai);
88            }
89
90            System.out.println ( "\n\nPreorder traversal" );
91            Tree.preorderTraversal();
92
93            System.out.println ( "\n\nInorder traversal" );
94            Tree.inorderTraversal();
95
96            System.out.println ( "\n\nPostorder traversal" );
```

```
97              Tree.postorderTraversal();
98              System.out.println();
99          }
100     }
```

Latihan 9

Jalankan program di atas dan tulis hasilnya pada kotak berikut :

Tugas 9

1. Tambahkan method untuk menghitung banyaknya node pada Binary Tree
2. Tambahkan method untuk menghitung banyaknya daun
3. Tambahkan method untuk menghitung tinggi dari pohon
4. Tambahkan method panjang yang menerima suatu pohon biner dan menentukan berapa level yang dimiliki pohon tersebut.
5. Buatlah program Complete Binary Tree dengan menggunakan array.

10

AVL Tree

Teori

AVL Tree adalah salah satu varian dari binary tree tetapi dengan syarat tambahan :

- Data di subtree kiri suatu node selalu lebih kecil dari data di node tersebut dan data di subtree kanan suatu node selalu lebih besar .
- Beda tinggi kiri dan tinggi kanan suatu node tidak lebih dari 1. Jika hal ini terjadi maka dilakukan rotasi kiri atau rotasi kanan

ADT AVL Tree

Dari ilustrasi gambar di atas ADT antrian dapat direpresentasikan sebagai berikut

List
Node root;
AVLT() boolean cariDt(int dt) boolean sisipDt(int dt) int tinggi() int tinggi(Node node) int jumlahNode() void inOrderTraversal() int jumlahNode(Node node)

	Program Latihan Praktikum 10.1
1	**class Node{**
2	**int data;**
3	**int tinggi; //tinggi node**
4	**Node pKiri;**

```
5     Node pKanan;
6     Node pInduk;// pointer ke induk
7
8     //constructor node
9     public Node(int dt, int tg, Node pKi, Node pKa, Node pI){
10      this.data = dt;
11      this.tinggi = tg;
12      this.pKiri = pKi;
13      this.pKanan = pKa;
14      this.pInduk = pI;
15    }
16  }
17  public class AVLT {
18    private Node root;
19
20    public AVLT(){root = null;}
21
22    //cari dt di tree, mengembalikan true jika ditemukan
23    //dan false jika tidak
24    public boolean cariDt(int dt){
25      Node temp = root;
26
27      while(temp != null){
28        if(dt == temp.data) return true;
29        //cariDt subtree pKiri
30        else if(dt < temp.data) temp = temp.pKiri;
31        //cariDt subtree pKanan
32        else temp = temp.pKanan;
33      }
34      //dt tidak ditemukan
35      return false;
36    }
37    //sisip dt ke dalam tree, returns true if berhasil,
38     // false jika gagal
39    //tree diseimbangkan menggunakan algoritma AVL
40    public boolean sisipDt(int dt){
41      if(root == null){
42        //sisip dt di root
43        root = new Node(dt, 1, null, null, null);
44        return true;
45      }
46      //tree tidak kosong
47      else {
48        //mulai dari  root
49        Node temp = root;
50        Node prev = null;
51
52        //cari lokasi penyisipan dt
53        while(temp != null){
54          if(dt == temp.data)return false;
55          //sisip dt di subtree pKiri
56          else if(dt < temp.data){
57            prev = temp;
58            temp = temp.pKiri;
59          }
60          //sisip dt di subtree pKanan
61          else {
```

```
62              prev = temp;
63              temp = temp.pKanan;
64            }
65          }
66          //buat node baru
67          temp = new Node(dt, 1, null, null, prev);
68
69          if(dt < prev.data) prev.pKiri = temp;//sisip di pKiri
70          else prev.pKanan = temp; //sisipDt at pKanan
71          //mulai dari node yang disisipkan dan
72          //bergerak menuju root
73          while(temp != null){
74            //subtree pKiri dan pKanan memenuhi kondisi AVL
75            if(Math.abs(tinggi(temp.pKiri)-
76                        tinggi(temp.pKanan))  <=1){
77              temp.tinggi = Math.max(tinggi(temp.pKiri),
78                                     tinggi(temp.pKanan)) +1;
79            }
80            //kasus 1 algoritma AVL
81            else if(tinggi(temp.pKiri)-
82                    tinggi(temp.pKanan)>= 2 &&
83                     tinggi(temp.pKiri.pKiri) >=
84                    tinggi(temp.pKiri.pKanan))
85            {
86              Node parent = temp.pInduk;
87              Node pKiri = temp.pKiri;
88              temp.pKiri = pKiri.pKanan;
89              if(temp.pKiri != null) temp.pKiri.pInduk = temp;
90
91              pKiri.pKanan = temp;
92              temp.pInduk = pKiri;
93
94              pKiri.pInduk = parent;
95              if(parent == null) root = pKiri;
96              else if(parent.pKiri==temp)parent.pKiri = pKiri;
97              else parent.pKanan = pKiri;
98
99              //hitung tinggi subtree pKanan
100             temp.tinggi = Math.max(tinggi(temp.pKiri),
101                                    tinggi(temp.pKanan)) +1;
102
103             temp = pKiri;
104             //hitung tinggi dari root
105             temp.tinggi = Math.max(tinggi(temp.pKiri),
106                                    tinggi(temp.pKanan)) +1;
107           }
108           //case 2 algoritma AVl
109           else if(tinggi(temp.pKanan)-
110                   tinggi(temp.pKiri)>= 2 &&
111                    tinggi(temp.pKanan.pKanan) >=
112                   tinggi(temp.pKanan.pKiri))
113           {
114             Node parent = temp.pInduk;
115             Node pKanan = temp.pKanan;
116
117             temp.pKanan = pKanan.pKiri;
118             if(temp.pKanan != null) temp.pKanan.pInduk = temp;
```

```
119
120             pKanan.pKiri = temp;
121             temp.pInduk = pKanan;
122
123             pKanan.pInduk = parent;
124             if(parent == null) root = pKanan;
125             else if(parent.pKanan == temp)
126                  parent.pKanan = pKanan;
127             else parent.pKiri = pKanan;
128
129             //hitung tinggi subtree pKanan
130             temp.tinggi = Math.max(tinggi(temp.pKiri),
131                                    tinggi(temp.pKanan)) +1;
132
133             temp = pKanan;
134
135             //hitung tinggi dari root
136             temp.tinggi = Math.max(tinggi(temp.pKiri),
137                                    tinggi(temp.pKanan)) +1;
138          }
139          //kasus 3 dari algoritma AVL
140          else if(tinggi(temp.pKiri)-
141                  tinggi(temp.pKanan)>= 2 &&
142                   tinggi(temp.pKiri.pKanan) >=
143                  tinggi(temp.pKiri.pKiri))
144          {
145             Node parent = temp.pInduk;
146             Node pKiripKanan = temp.pKiri.pKanan;
147             temp.pKiri.pKanan = pKiripKanan.pKiri;
148             if(temp.pKiri.pKanan!= null)
149                temp.pKiri.pKanan.pInduk = temp.pKiri;
150
151             pKiripKanan.pKiri = temp.pKiri;
152             temp.pKiri.pInduk = pKiripKanan;
153
154             temp.pKiri = pKiripKanan.pKanan;
155             if(temp.pKiri != null) temp.pKiri.pInduk = temp;
156
157             pKiripKanan.pKanan = temp;
158             temp.pInduk = pKiripKanan;
159
160             pKiripKanan.pInduk = parent;
161             if(parent == null) root = pKiripKanan;
162             else if(parent.pKiri==temp)
163                  parent.pKiri = pKiripKanan;
164             else parent.pKanan = pKiripKanan;
165
166             //hitung tinggi subtree pKanan
167             temp.tinggi = Math.max(tinggi(temp.pKiri),
168                                    tinggi(temp.pKanan)) +1;
169
170             temp = pKiripKanan;
171
172             //hitung tinggi dari root
173             temp.tinggi = Math.max(tinggi(temp.pKiri),
174                                    tinggi(temp.pKanan)) +1;
175          }
```

```
176             //kasus 4 dari algoritma AVL
177             else if(tinggi(temp.pKanan)-
178                     tinggi(temp.pKiri)>= 2 &&
179                      tinggi(temp.pKanan.pKiri) >=
180                     tinggi(temp.pKanan.pKanan))
181             {
182               Node parent = temp.pInduk;
183               Node pKananpKiri = temp.pKanan.pKiri;
184
185               temp.pKanan.pKiri = pKananpKiri.pKanan;
186               if(temp.pKanan.pKiri!= null)
187                 temp.pKanan.pKiri.pInduk = temp.pKanan;
188
189               pKananpKiri.pKanan = temp.pKanan;
190               temp.pKanan.pInduk = pKananpKiri;
191
192               temp.pKanan = pKananpKiri.pKiri;
193               if(temp.pKanan != null) temp.pKanan.pInduk = temp;
194
195               pKananpKiri.pKiri = temp;
196               temp.pInduk = pKananpKiri;
197
198               pKananpKiri.pInduk = parent;
199               if(parent == null) root = pKananpKiri;
200               else if(parent.pKanan == temp)
201                 parent.pKanan = pKananpKiri;
202               else parent.pKiri = pKananpKiri;
203
204               temp.tinggi = Math.max(tinggi(temp.pKiri),
205                                      tinggi(temp.pKanan)) +1;
206
207               temp = pKananpKiri;
208
209               temp.tinggi = Math.max(tinggi(temp.pKiri),
210                                      tinggi(temp.pKanan)) +1;
211             }
212             temp = temp.pInduk;
213           }
214           //penyisipan berhasil
215           return true;
216         }
217       }
218       public int tinggi(){return root.tinggi;}
219       private int tinggi(Node node){
220         if(node == null)return 0;
221         else return node.tinggi;
222       }
223       //hitung node-node dari tree
224       public int jumlahNode() {
225         return jumlahNode(root);
226       }
227
228       public void inOrderTraversal(){
229         inOrder(root);
230       }
231
231       private void inOrder(Node r){
```

```
232       if (r == null)return;
233       inOrder(r.pKiri);
234       System.out.printf("-%d",r.data);
235       inOrder(r.pKanan);
236    }
237
238    //hitung node-node dari tree
239    private int jumlahNode(Node node){
240       if(node == null) return 0;
241       else
242          return 1 +jumlahNode(node.pKiri)
243                   +jumlahNode(node.pKanan);
244    }
245
246    public static void main (String[] args) {
247       AVLT t = new AVLT();
248       t.sisipDt(3); t.inOrderTraversal();System.out.println();
249       t.sisipDt(4); t.inOrderTraversal();System.out.println();
250       t.sisipDt(6); t.inOrderTraversal();System.out.println();
251       t.sisipDt(5); t.inOrderTraversal();System.out.println();
252       t.sisipDt(15); t.inOrderTraversal();System.out.println();
253       t.sisipDt(10); t.inOrderTraversal();System.out.println();
254       t.sisipDt(20); t.inOrderTraversal();System.out.println();
255       t.sisipDt(17); t.inOrderTraversal();System.out.println();
256       t.sisipDt(25);t.inOrderTraversal();System.out.println();
257    }
258 }
```

Latihan 10

Jalankan program di atas dan tampilkan hasilnya dan step by step sisip data secara manual

input	Manual (bentuk pohon)	Keluran program
3		
4		
6		
5		
15		
10		
20		
17		
25		

Tugas 10

1. Tambahkan method untuk menghapus suatu node pada pohon AVL
2. Dalam proses penyisipan data ke dalam pohon AVL jika kondisi pohon tidak seimbang maka harus dilakukan rotas kiri, rotasi kanan, rotasi kiri kana, atau rotasi kanan dan kiri. Dari program di atas pada baris berapakah dilakukan masing-masing proses ini?

3. Perbaikilah program di pada latihan 10.1 di atas menjadi sub-sub program dengan menambahkan sub program putarKiri, putarKanan, putarKiriKanan, putarKananKiri.

11

Graph

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman Abstract Data Type Graph*

Teori

Suatu graph adalah pasangan g = (V,E), V adalah himpunan tidak kosong terbatas vertex dari G, dan E $\subseteq$ VxV. Terdapat dua jenis graph yakni graph berarah dan graph tidak berarah.

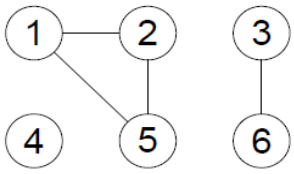
ADT Graph

ADT graph dapat direpresentasi dengan dua cara yakni adjacency matrik atau adjacentcy list. Sebagai ilustrasi perhatikan gambar graph berikut :

Graph tidak berarah dalam representasi ajd matrik

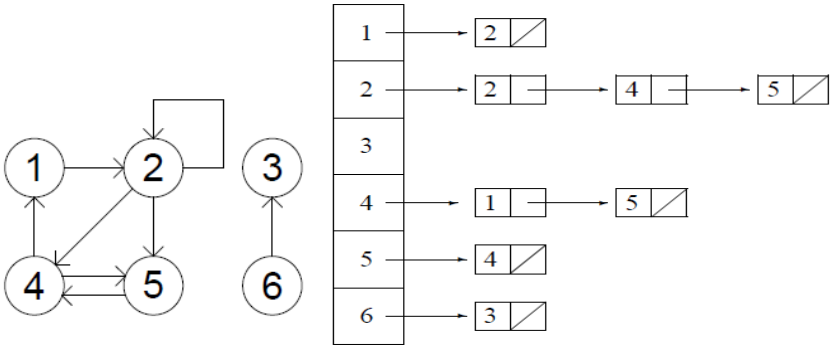
V = {1,2,3,4,5,6}

E = {{1,2},{2,5},{1,5},{3,6}}



$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Graph berarah dalam representasi ajd list



Graph
int jNode Node []
Graph(int) addAdj(int, int) cetak()

	Program Latihan Praktikum 11.1
1	`public class Graph{`
2	`  private class Node{`
3	`  private int data;`
4	`  private Node next;`
5	
6	`  public Node(int dt, Node n){`
7	`  data = dt;`
8	`  next = n;`
9	`  }`
10	`  public int getDt(){return data;}`
11	`  public Node getNext(){return next;}`
12	`  }`
13	
14	`  private Node []node;`
15	`  private int jNode;`
16	
17	`  public Graph(int n){`
18	`  jNode = n;`
19	`  node = new Node[jNode];`
20	`  for(int i=0; i<jNode; i++) node[i] = null;`
21	`  }`
22	
23	`  public void addAdj(int head, int adj){`

```
24        Node n = new Node(adj,node[head]);
25        node[head] = n;
26     }
27
28     public void cetak(){
29        for (int i=0; i<jNode; i++){
30           System.out.print("["+i+"]");
31           Node n = node[i];
32           while (n!=null){
33              System.out.print("->"+n.getDt());
34              n = n.getNext();
35           }
36           System.out.println();
37        }
38     }
39     public static void main(String s[]){
40        Graph g = new Graph(5);
41        g.addAdj(0,3);
42        g.addAdj(0,1);
43        g.addAdj(1,4);
44        g.addAdj(1,2);
45        g.addAdj(2,4);
46        g.addAdj(2,1);
47        g.addAdj(4,3);
48        g.cetak();
49     }
50  }
```

Tugas 11

1. Tambahkan method untuk mengetahui berapa jumlah edge yang masuk atau keluar dari suatu node.
2. Tambahkan pada ADT program di atas untuk merepresentasikan graph dengan matrik adjacency.
3. Buatlah sub program untuk mendapatkan tetangga dari suatu node tertentu.
4. Lakukan penulusuran node dari grap menggunakan BFS dan BFS. Gunakan struktur data yang pernah dibuat stack atau antrian.
5. Lakukan modifikasi pada ADT graph di atas sehingga suatu hubungan antar node terdapat nilai jarak.
6. Berdasarkan hasil modifikasi pada tugas 5 susun method untuk menghitung jarak terpendek dari suatu node asal ke node tujuan.

12

Sorting (Pengurutan)

Tujuan Instruksional Khusus:

- *Praktikan ini bertujuan untuk memberi pemahaman mengenai Sorting*

Teori

Pengurutan data (sorting) didefinisikan sebagai suatu proses untuk menyusun kembali himpunan obyek menggunakan aturan tertentu. Ada dua macam urutan yang biasa digunakan dalam proses pengurutan yaitu :

- urut naik (ascending) yaitu dari data yang mempunyai nilai paling kecil sampai paling besar
- urut turun (descending) yaitu data yang mempunyai nilai paling besar sampai paling kecil.

Contoh : data bilangan 5, 2, 6 dan 4 dapat diurutkan naik menjadi 2, 4, 5, 6 atau diurutkan turun menjadi 6, 5, 4, 2. Pada data yang bertipe char, nilai data dikatakan lebih kecil atau lebih besar dari yang lain didasarkan pada urutan relatif (collating sequence) seperti dinyatakan dalam tabel ASCII

Keuntungan dari data yang sudah dalam keadaan terurutkan antara lain :

- data mudah dicari (misalnya dalam buku telepon atau kamus bahasa), mudah untuk dibetulkan, dihapus, disisipi atau digabungkan. Selain itu dalam keadaan terurutkan, kita mudah melakukan pengecekan apakah ada data yang hilang atau tidak.
- melakukan kompilasi program komputer jika tabel-tabel simbol harus dibentuk
- mempercepat proses pencarian data yang harus dilakukan berulang kali.

Data yang diurutkan sangat bervariasi, dalam hal jumlah data maupun jenis data yang akan diurutkan. Tidak ada algoritma terbaik untuk setiap situasi yang kita hadapi, bahkan cukup sulit untuk menentukan algoritma mana yang paling baik untuk situasi tertentu karena ada beberapa faktor yang mempengaruhi efektifitas algoritma pengurutan. Beberapa faktor yang berpengaruh pada efektifitas suatu algoritma pengurutan antara lain:

- banyak data yang diurutkan
- kapasitas pengingat apakah mampu menyimpan semua data yang kita miliki
- tempat penyimpanan data, misalnya disket, flashdisk, harddisk atau media penyimpanan yang lain.

Ada 3 jenis metode sorting (pengurutan) yang akan dibahas yaitu :

1. Bubble sort (gelembung) secara ascending

Metode gelembung (bubble sort) sering juga disebut dengan metode penukaran (exchange sort) adalah metode yang mengurutkan data dengan cara membandingkan masing-masing elemen, kemudian melakukan penukaran bila perlu. Metode ini mudah dipahami dan diprogram, tetapi bila dibandingkan dengan metode lain yang kita pelajari, metode ini merupakan metode yang paling tidak efisien.

Proses pengurutan metode bubble sort ini menggunakan dua loop. Loop pertama melakukan pengulangan dari elemen ke 2 sampai dengan elemen ke N-1 (misalnya variable i), sedangkan kalang kedua melakukan pengulangan menurun dari elemen ke N sampai elemen ke i (misalnya variable j). Pada setiap pengulangan, elemen ke j-1 dibandingkan dengan elemen ke j. Apabila data ke j-1 lebih besar daripada data ke j, dilakukan penukaran.

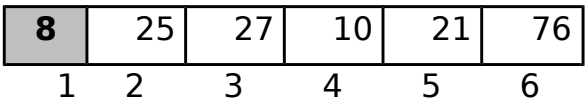
Contoh: Elemen array / larik dengan N=6 buah elemen dibawah ini.

25	27	10	8	76	21
1	2	3	4	5	6

Langkah 1:

K=N=6					21	76
K=5				8	21	76
K=4			8	10	21	76
K=3		8	27	10	21	76
K=2	8	25	27	10	21	76

Hasil akhir langkah 1:



Langkah 2:

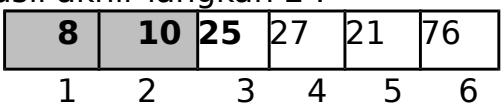
K=N= 6 — 21 76

K=5 — 10 21 76

K=4 — 10 27 21 76

K=3 — 10 25 27 21 76

Hasil akhir langkah 2 :



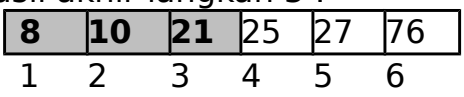
Langkah 3:

K=N=6 — 21 76

K=5 — 21 27 76

K=4 — 21 25 27 76

Hasil akhir langkah 3 :

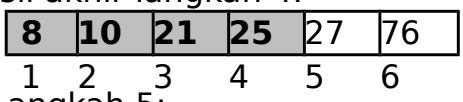


Langkah 4:

K=N=6 — 27 76

K=5 — 25 27 76

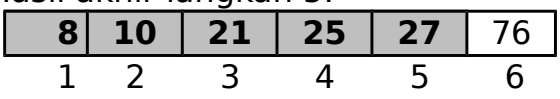
Hasil akhir langkah 4:



Langkah 5:

K=N=6 — 27 76

Hasil akhir langkah 5:



Sorting sudah selesai, karena data sudah terurutkan.

	Program Latihan Praktikum 12.1
1	public class BubbleSorter {
2	
3	int[] L = {25, 27, 10, 8, 76, 21};
4	

```
5       void bubbleSort() {
6           int i, j, Max = 6, temp;
7           for (i = 0; i < Max - 1; i++) {
8               System.out.println("Langkah " + (i + 1) + ":");
9               for (j = Max - 1; j > i; j--) {
10                  if (L[j - 1] > L[j]) {
11                      temp = L[j];
12                      L[j] = L[j - 1];
13                      L[j - 1] = temp;
14                  }
15                  System.out.println(L[j] + " index =" + (j+1));
16              }
17              System.out.println(L[j] + " index =" + (j+1));
18          }
19          System.out.println("Hasil akhir:");
20          for (i = 0; i <= 5; i++) {
21              System.out.println(L[i] + " index:" + (i+1));
22          }
23      }
24
25      public static void main(String[] args) {
26          BubbleSorter sorter = new BubbleSorter();
27          sorter.bubbleSort();
28      }
29  }
30
```

Latihan 12.1

1. Jalankan program diatas dan amati hasilnya
2. Lakukan modifikasi program diatas agar sorting secara decending. Pada baris ke berapa yang harus diubah?

2. Selection sort (maksimum/minimun)

Metode seleksi melakukan pengurutan dengan cara mencari data yang terkecil kemudian menukarkannya dengan data yang digunakan sebagai acuan atau sering dinamakan pivot.

Proses pengurutan dengan metode seleksi dapat dijelaskan sebagai berikut:

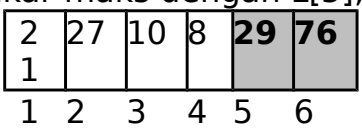
Contoh: Elemen array / larik dengan N=6 buah elemen dibawah ini.

29	27	10	8	76	21
1	2	3	4	5	6

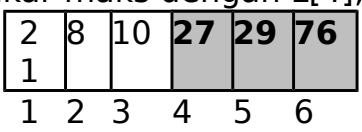
Langkah 1:
Cari elemen maksimum di dalam larik L[1..6]→ maks = L[5] = 76
Tukar maks dengan L[N], hasil akhir langkah 1:

29	27	10	8	21	**76**
1	2	3	4	5	6

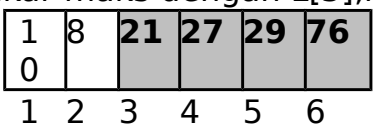
Langkah 2:
(berdasarkan susunan larik hasil langkah 1)
Cari elemen maksimum di dalam larik L[1..5]→ maks = L[1] = 29
Tukar maks dengan L[5],hasil akhir langkah 2:



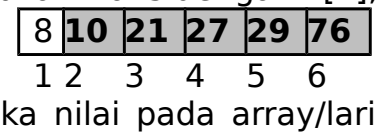
Langkah 3:
(berdasarkan susunan larik hasil langkah 2)
Cari elemen maksimum di dalam larik L[1..4]→ maks = L[2] = 27
Tukar maks dengan L[4],hasil akhir langkah 3:



Langkah 4:
(berdasarkan susunan larik hasil langkah 3)
Cari elemen maksimum di dalam larik L[1..3]→ maks = L[1] = 21
Tukar maks dengan L[3],hasil akhir langkah 4:



Langkah 5:
(berdasarkan susunan larik hasil langkah 4)
Cari elemen maksimum di dalam larik L[1..2]→ maks = L[1] = 10
Tukar maks dengan L[2],hasil akhir langkah 5:



Jika nilai pada array/larik sudah terurutkan maka proses sorting sudah selesai.

	Program Latihan Praktikum 12.2
1	`public class SelectionSorter {`
2	
3	`    int[] L = {25, 27, 10, 8, 76, 21};`
4	
5	`    void selectionSort() {`
6	`        int j, k, i, temp;`
7	`        int jmax, u = 5;`
8	`        for (j = 0; j < 6; j++) {`
9	`            jmax = 0;`
10	`            System.out.println("Langkah " + (j + 1) + ":");`
11	`            for (k = 1; k <= u; k++) {`
12	`                if (L[k] > L[jmax]) {`
13	`                    jmax = k;`
14	`                }`

```
15              }
16              temp = L[u];
17              L[u] = L[jmax];
18              L[jmax] = temp;
19              u--;
20              //melihat hasil tiap langkah
21              for (i = 0; i <= 5; i++) {
22                 System.out.println(L[i] + " index:" + (i + 1));
23              }
24          }
25
26          System.out.println("Hasil akhir:");
27          for (i = 0; i <= 5; i++) {
28              System.out.println(L[i] + " index:" + (i + 1));
29          }
30      }
31
32      public static void main(String[] args) {
33          SelectionSorter sorter = new SelectionSorter();
34          sorter.selectionSort();
35      }
36  }
```

Latihan 12.2

1. Jalankan program diatas dan amati hasilnya
2. Lakukan modifikasi program diatas agar sorting secara decending. Pada baris ke berapa yang harus diubah?

3. Insertion sort (sisip)

Proses pengurutan dengan metode penyisipan langsung dapat dijelaskan sebagai

berikut :

Data dicek satu per satu mulai dari yang kedua sampai dengan yang terakhir. Apabila ditemukan data yang lebih kecil daripada data sebelumnya, maka data tersebut disisipkan pada posisi yang sesuai. Akan lebih mudah apabila membayangkan pengurutan kartu. Pertama-tama anda meletakkan kartu-kartu tersebut di atas meja, kemudian melihatnya dari kiri ke kanan. Apabila kartu di sebelah kanan lebih kecil daripada kartu di sebelah kiri, maka ambil kartu tersebut dan sisipkan di tempat yang sesuai.

Contoh: Elemen array / larik dengan N=6 buah elemen dibawahini.

29	27	10	8	76	21
1	2	3	4	5	6

Langkah 1:
Elemen L[1] dianggap sudah terurut

29	27	10	8	76	21

1	2	3	4	5	6

Langkah 2:
(berdasarkan susunan larik pada langkah 1)
Cari posisi yang tepat untuk L[2] pada L[1..2],diperoleh :

27	**29**	10	8	76	21
1	2	3	4	5	6

Langkah 3:
(berdasarkan susunan larik pada langkah 2)
Cari posisi yang tepat untuk L[3] pada L[1..3],diperoleh :

10	**27**	**29**	8	76	21
1	2	3	4	5	6

Langkah 4:
(berdasarkan susunan larik pada langkah 3)
Cari posisi yang tepat untuk L[4] pada L[1..4],diperoleh :

8	**10**	**27**	**29**	76	21
1	2	3	4	5	6

Langkah 5:
(berdasarkan susunan larik pada langkah 4)
Cari posisi yang tepat untuk L[5] pada L[1..5],diperoleh :

8	**10**	**27**	**29**	**76**	21
1	2	3	4	5	6

Langkah 6:
(berdasarkan susunan larik pada langkah 5)
Cari posisi yang tepat untuk L[6] pada L[1..6],diperoleh :

8	**10**	**21**	**27**	**29**	**76**
1	2	3	4	5	6

Jika nilai pada array/larik sudah terurutkan maka proses sorting sudah selesai.

	Program Latihan Praktikum 12.3
1	`public class InsertionSorter {`
2	
3	`    int[] L = new int[7];`
4	
5	`    void insertionSort() {`
6	`        int k, temp, j;`
7	`        L[1] = 29;`
8	`        L[2] = 27;`
9	`        L[3] = 10;`
10	`        L[4] = 8;`
11	`        L[5] = 76;`
12	`        L[6] = 21;`
13	`        for (k = 2; k <= 6; k++) {`

```
14              temp = L[k];
15              j = k - 1;
16              System.out.println("Langkah" + (k - 1));
17              while (temp <= L[j]) {
18                  L[j + 1] = L[j];
19                  j--;
20              }
21              if ((temp >= L[j]) || (j == 1)) {
22                  L[j + 1] = temp;
23              } else {
24                  L[j + 1] = L[j];
25                  L[j] = temp;
26              }
27              for (int i = 1; i <= 6; i++) {
28                  System.out.println(L[i] + " index:" + i);
29              }
30          }
31          for (int i = 1; i <= 6; i++) {
32              System.out.println(L[i] + " index:" + i);
33          }
34      }
35
36      public static void main(String[] args) {
37          InsertionSorter sorter = new InsertionSorter();
38          sorter.insertionSort();
39
40      }
41  }
42
```

Latihan 12.3

1. Jalankan program diatas dan amati hasilnya
2. Lakukan modifikasi program diatas agar sorting secara decending. Pada baris ke berapa yang harus diubah?

Tugas 12

1. Buatlah program sorting dengan menggunakan merge sort dan quick sort untuk mengurutkan angka 29, 27, 10, 8, 76 dan 21.

13

UJian Akhir

KARTU PESERTA PRAKTIKUM

ALGORITMA DAN STRUKTUR DATA

3 X 4

NAMA MAHASISWA :

NIM :

NO	PERCOBAAN	TANGGAL	NILAI	TANDA TANGA ASISTEN
1	Pengenalan *Object Oriented Programming*			
2	ADT Array 1 Dimensi			
3	ADT Array 2 Dimensi			
4	ADT Stack			
5	ADT Single Linked List			
6	ADT Double Linked List			
7	ADT Sirkular Linked List			
8	ADT Antrian			
9	ADT Binary Tree			
10	ADT AVL Tree			
11	ADT Graph			
12	*Sorting* (Pengurutan)			

Mengetahui,

Dosen pengampu
