

# **Modul Praktikum Algoritma dan Struktur Data**

**Laboratorium Pembelajaran  
Fakultas Ilmu Komputer (FILKOM)  
Universitas Brawijaya  
Malang  
2018**

## **Kata Pengantar**

Puji syukur Alhamdulillah berkat rahmat dan pertolongannya modul praktikum ASD tahun 2018 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 sedemikian 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 2018

Penyusun

## Daftar Isi

Kata Pengantar .....	2
Daftar Isi.....	3
Modul 1 : Algoritma .....	4
Modul 2 : Pemrograman Berorientasi Objek .....	11
Modul 3 : ADT Array.....	19
Modul 4 : ADT Single Linked List .....	33
Modul 5 : ADT Double Linked List .....	38
Modul 6 : ADT Circular Linked List .....	43
Modul 7 : ADT Stack.....	49
Modul 8 : ADT Queue.....	55
Modul 9 : ADT Binary Tree.....	61
Modul 10 : ADT AVL Tree.....	66
Modul 11 : ADT Graf.....	73
Modul 12 : Sorting .....	77
TIM PENYUSUN.....	88

## Modul 1 : Algoritma

### 1.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = 170 menit, dengan rincian sebagai berikut :

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan tugas / Studi Kasus
- 50 menit Pengayaan

### 1.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengetahui dan membuat tentang macam-macam algoritma, dan langkah-langkah membuat algoritma.
2. Mengetahui dan membuat tentang simbol-simbol flowchart, dan menyelesaikan masalah pemrograman dengan menggunakan flowchart.
3. Mengetahui dan membuat bagian-bagian pseudocode menyelesaikan masalah pemrograman dengan menggunakan pseudocode.

### 1.3 Alat & Bahan

1. Komputer
2. Microsoft Word atau Microsoft Visio

### 1.4 Dasar Teori

Logika pemrograman dapat diartikan sebagai suatu metode atau cara berpikir terstruktur yang diperlukan untuk menyelesaikan permasalahan di dalam menyusun suatu program. Ketika seseorang telah memahami langkah-langkah di dalam menyelesaikan masalah di dalam membuat suatu program, maka ia akan lebih mudah mengimplementasikannya ke dalam suatu bahasa pemrograman.

Setiap permasalahan pemrograman dapat diselesaikan dengan metode atau langkah yang berbeda-beda, tetapi penggunaan metode yang baik (optimal) ditentukan oleh tujuan dari program tersebut dibuat. Kemampuan seseorang di dalam menentukan metode atau cara yang tepat di dalam menyelesaikan masalah pemrograman tergantung pada kemampuan analisis dan pengalaman yang dimilikinya. Oleh karena itu latihan penyelesaian permasalahan pemrograman sangat diperlukan dan akan membantu melatih dan membentuk cara berpikir yang baik di dalam menyelesaikan permasalahan pemrograman.

Metode atau cara menyelesaikan permasalahan pemrograman dapat dilakukan dengan menggunakan algoritma dan *flowchart* ataupun *pseudocode*.

#### 1.4.1 Algoritma

Algoritma dapat diartikan sebagai suatu cara di dalam menyelesaikan masalah melalui serangkaian langkah-langkah atau tahap terstruktur dan ditulis secara sistematis. Pada umumnya algoritma banyak diimplementasikan dalam bentuk tulisan, tetapi ada pula yang diimplementasikan dalam bentuk gambar.

Untuk menyusun algoritma di dalam menyelesaikan permasalahan, maka kita harus memahami permasalahannya dan mengetahui langkah-langkah penyelesaian masalah tersebut untuk dituangkan ke dalam bentuk tulisan/gambar.

#### 1.4.2 Algoritma Pemrograman

Algoritma yang digunakan untuk membuat suatu program disebut sebagai algoritma pemrograman. Sebagaimana algoritma secara umum, algoritma pemrograman juga berisi

langkah-langkah terstruktur untuk menyelesaikan permasalahan di bidang pemrograman (untuk membuat suatu program).

Algoritma pemrograman yang baik adalah algoritma yang jelas dan mudah dipahami oleh yang membaca algoritma tersebut, sehingga memudahkan *programmer* ketika akan mengimplemen-tasikannya ke dalam bahasa pemrograman.

Secara umum, algoritma pemrograman dimulai dengan menentukan nilai suatu variabel, yang diinputkan oleh pengguna, ataupun yang berupa nilai awal. Kemudian dilanjutkan dengan memproses variabel tersebut sesuai dengan tujuan dibuatnya program, dan hasil proses tersebut ditampilkan kepada pengguna ataupun digunakan sebagai nilai awal untuk proses berikutnya, sampai akhirnya tujuan program tercapai dan program selesai.


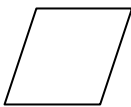
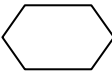

#### 1.4.3 Diagram Alir (*Flowchart*)

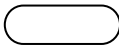


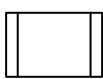

Selain menggunakan algoritma, representasi logika pemrograman di dalam menyelesaikan permasalahan di dalam pemrograman adalah diagram alir (*flowchart*). *Flowchart* dapat diartikan sebagai suatu aliran proses yang berupa simbol/bagan yang menggambarkan urutan-urutan penyelesaian permasalahan, dimana terjadi hubungan antara proses yang satu dengan yang lainnya. *Flowchart* dituangkan ke dalam bentuk gambar atau simbol yang telah menjadi kesepakatan para ahli komputer di dalam representasi logika pemrograman untuk menyusun program komputer.

#### 1.4.4 Program Flowchart

Adalah suatu *flowchart* yang digunakan untuk menyelesaikan permasalahan pada pembuatan program. Setiap penyelesaian permasalahan pemrograman dapat digambarkan dengan menggunakan simbol-simbol yang terdapat pada *program flowchart*. *Flowchart* inilah yang akan kita gunakan ketika kita akan menyelesaikan permasalahan pemrograman yang pada akhirnya akan diimplementasikan ke dalam suatu bahasa pemrograman. Simbol-simbol yang digunakan pada *program flowchart* dapat dilihat pada gambar di bawah.

Simbol-simbol yang digunakan pada *program flowchart* antara lain :

 Proses	<ul style="list-style-type: none"> <li>• Menunjukkan suatu proses/ pengolahan</li> <li>• Digunakan untuk melambangkan :             <ul style="list-style-type: none"> <li>- perhitungan</li> <li>- perubahan nilai variabel</li> </ul> </li> </ul>
 Operasi Input/Output	<ul style="list-style-type: none"> <li>• Menunjukkan operasi input/ouput</li> <li>• Digunakan untuk melambangkan :             <ul style="list-style-type: none"> <li>- menunggu input/ masukan</li> <li>- mengeluarkan output/ keluaran</li> </ul> </li> </ul>
 Persiapan ( <i>Preparation</i> )	<ul style="list-style-type: none"> <li>• Menunjukkan suatu persiapan</li> <li>• Digunakan untuk melambangkan :             <ul style="list-style-type: none"> <li>- memberikan nilai awal pada suatu variabel</li> <li>- permulaan dari suatu perulangan</li> </ul> </li> </ul>
 Keputusan ( <i>Decision</i> )	<ul style="list-style-type: none"> <li>• Menunjukkan proses pembuatan keputusan</li> <li>• Digunakan untuk melambangkan :             <ul style="list-style-type: none"> <li>- suatu pilihan/ percabangan</li> </ul> </li> </ul>

	(ya/tidak)
 Terminal (Terminator)	<ul style="list-style-type: none"> <li>Digunakan untuk menunjuk kan awal dan akhir suatu program/<i>flowchart</i></li> </ul>
 Penghubung ( <i>connector</i> )	<ul style="list-style-type: none"> <li>Digunakan sebagai penghubung antar simbol <i>flowchart</i> yang terpisah (simbol yang terpisah masih berada dalam satu halaman)</li> </ul>
 Penghubung antar halaman ( <i>Offpage Connector</i> )	<ul style="list-style-type: none"> <li>Digunakan sebagai penghubung antar simbol <i>flowchart</i> yang terpisah (simbol yang terpisah berada pada halaman yang berbeda)</li> </ul>
 Modul	<ul style="list-style-type: none"> <li>Menunjukkan suatu proses / Subproses yang telah ditentukan</li> <li>Dapat berupa suatu : <ul style="list-style-type: none"> <li>Prosedur</li> <li>Fungsi</li> </ul> </li> </ul>
Panah 	<ul style="list-style-type: none"> <li>Menunjukkan arah dari suatu proses</li> </ul>

Gambar Simbol-simbol pada *program flowchart*

Di dalam menyusun atau membuat suatu *program flowchart* ada kaidah atau aturan-aturan yang harus dipenuhi. Aturan-aturan tersebut antara lain :

1. Digunakan simbol yang sesuai dengan fungsi simbol tersebut dan berurutan berdasarkan langkah yang digunakan.
2. Jalannya metode penyelesaian dibuat sesingkat-singkatnya, tetapi tetap dapat dengan mudah dipahami.
3. Dibuat metode penyelesaian yang sederhana, mudah dipahami, dan diimplementasikan. Hilangkan langkah-langkah (logika) yang tidak perlu dan berbelit-belit.
4. Digunakan simbol penghubung, jika metode penyelesaian tidak dapat diselesaikan dengan langkah yang sederhana, atau membutuhkan penyelesaian lebih dari 1 halaman
5. Digunakan simbol pengulangan untuk menunjukkan langkah yang diulang-ulang.
6. Digunakan modul, jika terdapat langkah atau metode yang sering dilakukan
7. Penyusunan simbol digambarkan dari atas ke bawah, dan dari kiri ke kanan, dan diberi arah panah yang menunjukkan langkah yang dilakukan sebelumnya dan setelahnya. Setiap simbol (kecuali *start* dan *stop* minimal memiliki sebuah panah masuk dan sebuah panah keluar.

Sebuah *flowchart* diawali dengan simbol “Mulai” (*Start*) dan diakhiri dengan “Selesai” (*Stop*)

#### 1.4.5 Pseudocode

Representasi lain dari logika pemrograman di dalam menyelesaikan permasalahan pemrograman yang sering digunakan adalah *pseudocode*. *Pseudocode* adalah suatu urutan langkah-langkah atau prosedur penyelesaian masalah yang dituliskan dalam bentuk yang sistematis yang mendekati pernyataan/instruksi atau *syntax* yang digunakan oleh suatu bahasa pemrograman. Representasi *pseudocode* ini banyak digunakan di dalam penulisan ilmiah yang dipublikasikan (misalnya pada makalah, atau jurnal) karena penggunaan ruang tulisan yang lebih sedikit dibandingkan *flowchart*, dan juga kedekatannya dengan struktur bahasa pemrograman.

Struktur penulisan dan perintah yang digunakan tidak memiliki notasi/symbol yang baku, tetapi berdasarkan kesepakatan yang dianggap dapat mempermudah seseorang di dalam memahami langkah-langkah penyelesaian masalah di dalam pemrograman. Misalnya, untuk pernyataan-pernyataan yang saling berhubungan atau saling tergantung, biasanya ditulis dengan indentasi, seperti keputusan, dan perulangan.

Notasi pada *pseudocode* yang digunakan biasanya berkorespondensi dengan bahasa pemrograman yang umum, dan biasa disebut **notasi algoritmik**.

#### 1.4.6 Struktur Pseudocode

Walaupun tidak ada notasi baku di dalam menyusun atau membuat suatu *pseudocode*, tetapi umumnya suatu *pseudocode* memiliki suatu struktur sebagai berikut :

- Kepala *pseudocode*

Kepala *pseudocode* terdiri atas nama *pseudocode* dan komentar yang berisi penjelasan (spesifikasi) tentang *pseudocode* tersebut. Pada kepala *pseudocode* ini biasanya juga dicantumkan nama-nama pembuat program dan versi atau revisi dari program tersebut. Jika merupakan pengembangan dari *pseudocode* sebelumnya, maka biasanya juga berisi fitur-fitur baru yang telah ditambahkan pada *pseudocode* tersebut.

- Deklarasi

Bagian deklarasi biasanya berisi tentang pencantuman atau penentuan penggunaan semua nama (konstanta, peubah/*variable*, tipe atau jenis data, prosedur atau fungsi) yang digunakan di dalam *pseudocode*. Biasanya digunakan beberapa kesepakatan untuk penentuan nama untuk konstanta, dan peubah yang digunakan. Misalnya untuk nama konstanta digunakan huruf kapital, seperti PHI=3.14, AWAL=0, dll, dan huruf kecil untuk peubah, misalnya bil, nilai, dll.

Pada bagian ini juga biasanya dituliskan tentang tipe-tipe data yang digunakan dan juga tipe data baru yang dibuat, yang akan dipergunakan di dalam deskripsi atau badan *pseudocode* tersebut.

- Deskripsi atau badan

Bagian deskripsi adalah bagian yang berisi uraian langkah-langkah penyelesaian permasalahan. Deskripsi juga berisi tentang proses, pembuatan keputusan, dan perulangan. Umumnya untuk tiap-tiap bagian atau kelompok penyelesaian akan diberi komentar, yang umumnya ditulis menggunakan tanda kurung "{ dan "}", atau "//".

Bagian ini adalah bagian yang paling penting dari suatu *pseudocode*, yang merupakan bagian utama yang berisi langkah-langkah yang dilakukan di dalam penyelesaian masalah pemrograman.

*Pseudocode* di atas bertujuan untuk menampilkan bilangan yang diinputkan oleh pengguna ke layar. Pada *pseudocode* tersebut dideklarasikan (digunakan) peubah bernama "Bil" yang bertipe *integer* yang akan menyimpan nilai yang diinputkan oleh pengguna. Tipe *integer* yang digunakan, menunjukkan bahwa input yang diberikan oleh pengguna harus berupa bilangan yang memiliki nilai jangkauan (*range*) tertentu. Pada bagian deskripsi dijelaskan tentang proses yang dilakukan, yaitu proses untuk meminta masukan dari pengguna, dan proses menampilkan atau mencetak bilangan yang sudah diinputkan oleh pengguna ke layar.

### 1.5 Prosedur Praktikum

Pelajari contoh-contoh Algoritma Pemrograman, flowchart dan Pseudocode berikut :

a. Algoritma Pemrograman

Perhatikan dan analisis algoritma pemrograman berikut :

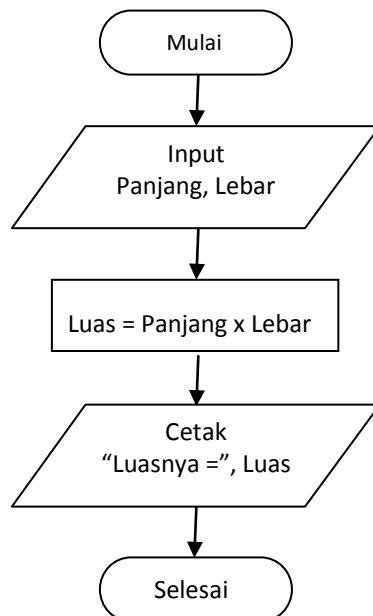
1. Tentukan isi dari variabel pertama.
2. Tentukan isi dari variabel kedua.
3. Buat variabel ketiga untuk menampung isi variabel secara sementara.
4. Masukkan isi variabel pertama ke variabel ketiga.
5. Masukkan isi variabel kedua ke variabel pertama.
6. Masukkan isi variabel ketiga ke variabel kedua.
7. Tampilkan kembali isi variabel pertama dan kedua.

Apa judul yang tepat untuk algoritma di atas ?

.....

b. Flowchart

Perhatikan dan analisis flowchart berikut :



Apa judul yang tepat untuk flowchart di atas ?

.....

c. Pseudocode



<b>Pseudocode</b> MenghitungLuasSegiempat; {pseudocode ini akan menghitung luas dari sebuah segiempat yang merupakan perkalian dari panjang dan lebarnya}
<b>Deklarasi</b> Panjang, Lebar, Luas : word;
<b>Deksripsi</b> Cetak "Inputkan nilai panjang : "; Input Panjang; Cetak "Inputkan nilai lebar : ", Input Lebar; Luas = Panjang * Lebar; Cetak "Luasnya = ", Luas;

Apa judul yang tepat untuk pseudocode di atas ?

.....

## 1.6 Analisis Hasil

Jelaskan proses yang terjadi pada Algoritma Pemrograman, Flowchart dan Pseudocode di atas :

### a. Algoritma Pemrograman

.....

.....

.....

.....

.....

.....

### b. Flowchart

.....

.....

.....

.....

.....

.....

### c. Pseudocode

.....

.....

.....

.....

.....

.....

### 1.7 Kesimpulan

Berikan kesimpulan yang kamu dapatkan dari praktikum ini :

.....

.....

.....

### 1.8 Latihan

Buatlah Algoritma Pemrograman, Flowchart dan Pseudocode untuk proses di bawah ini :

- Menghitung luas bidang datar
- Mencari hasil perhitungan aritmetika 2 buah bilangan

### 1.9 Tugas

Kerjakan soal-soal berikut :

- Buat algoritma pemrograman untuk mencari 5 buah bilangan kuadrat yang pertama.
- Buat algoritma pemrograman untuk menghitung jumlah 10 bilangan ganjil yang pertama.
- Buat *pseudocode* untuk menukar nilai dua buah peubah/variabel. Misal variabel A = 5 dan variabel B = 10, diproses sehingga menjadi variabel A = 10 dan variabel B = 5.
- Buat *flowchart* untuk menjumlahkan 5 buah bilangan pertama.
- Buat *pseudocode* untuk menjumlahkan 5 buah bilangan **kuadrat** yang pertama.
- Buat *flowchart* untuk menghitung pangkat 3 dari sebuah bilangan.

### 1.10 Daftar Pustaka

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms Using Java 6 edition", Wiley, USA, 2014.
- John R. Hubbard, "Scaum's Outline of Data Structures With Java second Edition", McGraw-Hill, New york, 2007.
- Robert Lafore, "Data Structures and Algorithm in Java second Edition", SamsPublishing, Indiana, 2003

## Modul 2 : Pemrograman Berorientasi Objek

### 2.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan tugas / Studi Kasus
- 50 menit **Pengayaan**

### 2.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Membuat program dengan menggunakan konsep pemrograman berorientasi objek
2. Memanfaatkan karakteristik pemrograman berorientasi objek di dalam mengimplementasikan program
3. Mampu mendeklarasikan suatu kelas dan menginisialisasikannya ke dalam suatu objek

### 2.3 Alat & Bahan

1. Komputer
2. Java IDE

### 2.4 Dasar Teori

#### 2.4.1 Konsep pemrograman berorientasi objek (PBO)

Secara umum pemrograman dapat dikategorikan dalam dua kategori yakni : pemrograman prosedural atau struktural dan pemrograman berorientasi objek (PBO). Pemrograman prosedural adalah suatu program yang disusun secara sistematis dalam bentuk urutan prosedur dalam menyelesaikan suatu masalah berbasis komputer. Program seperti merupakan implementasi langsung dari suatu algoritma. Contoh program seperti ini adalah PASCAL, Visual Basic, C. Sedangkan pemrograman berorientasi objek adalah program yang mencoba menyelesaikan masalah dengan memandang suatu masalah tersebut sebagai suatu objek. Konsep ini mengadopsi pengetahuan umum tentang objek dalam dunia nyata. Setiap objek dapat mempunyai nama objek, atribut yang mencirikan suatu objek tersebut dan method yang menggambarkan perilaku suatu objek. Sebagai contoh misalnya suatu objek sepeda motor, objek ini mempunyai sejumlah ciri seperti jumlah roda, warna, merk, nomer mesin, nomer rangka dan lain sebagainya serta mempunyai sejumlah perilaku seperti berjalan, berhenti adalah perilaku dari suatu objek sepeda.

Dengan sudut pandang yang berbeda antara pemrograman procedural dengan pemrograman berorientasi objek, maka cara berpikir atau logika pemrograman yang digunakan di dalam menyusun kedua program tersebut juga berbeda.

#### 2.4.2 Kelas (*Class*)

Di dalam pemrograman berorientasi objek dikenal istilah kelas (*class*). Kelas didefinisikan sebagai sebuah blue print, atau prototipe dari suatu objek, yang mendefinisikan variable-variabel dan metode-metode yang umum untuk semua objek dari jenis tertentu. Kelas mendefinisikan atribut dan perilaku objek yang dibuatnya. Kelas merupakan definisi formal suatu abstraksi, dan berlaku sebagai template untuk pembuatan objek-objek. Kelas berisi abstraksi yang terdiri dari nama kelas, atribut dan service/behaviour.

Bagian-bagian dari sebuah kelas secara umum terdiri atas 2 bagian yakni:

##### 2.4.2.1 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.

#### 2.4.2.2 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();
```

#### 2.4.2.3 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] <type_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.

### 2.4.3 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.

#### 2.4.3.1 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 ;
}
```

#### 2.4.3.2 Modifier pada method

Modifier menentukan level pengaksesan sebuah method. Modifier untuk atribut, yaitu public, protected, private. Penjelasan modifier atribut serupa dengan penjelasan modifier pada kelas.

#### 2.4.3.3 Method tanpa nilai balikan

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

#### 2.4.3.4 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.

#### 2.4.3.5 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.

#### 2.4.3.6 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.

### 2.5 Prosedur Praktikum

Ketikkan listing program – listing program di bawah ini, kemudian jalankan program tersebut.

	Program Latihan Praktikum 2.5.1
1	class Kotak{
2	double panjang;
3	double lebar;
4	double tinggi;
5	
6	//mendefinisikan method void (tidak mengembalikan nilai)
7	void cetakVolume(){
8	System.out.println("Volume kotak = "
9	+(panjang*lebar*tinggi));
10	}
11	}
12	
13	class DemoMethod1{
14	public static void main(String[] args){
15	Kotak k1, k2, k3;
16	
17	//instansiasi objek
18	k1=new Kotak();
19	k2=new Kotak();
20	k3=new Kotak();

21	
22	//mengisi data untuk objek k1
23	k1.panjang=4;
24	k1.lebar=3;
25	k1.tinggi=2;
26	
27	//mengisi data untuk objek k2
28	k2.panjang=6;
29	k2.lebar=5;
30	k2.tinggi=4;
31	
32	//mengisi data untuk objek k3
33	k3.panjang=8;
34	k3.lebar=7;
35	k3.tinggi=6;
36	
37	//memanggil method cetakVolume() untuk masing-masing
38	//objek
39	k1.cetakVolume();
40	k2.cetakVolume();
41	k3.cetakVolume();
42	}
43	}

	Program Latihan Praktikum 2.5.2
1	class Kotak{
2	double panjang;
3	double lebar;
4	double tinggi;
5	
6	//mendefinisikan method yang mengembalikan tipe double
7	double hitungVolume(){
8	//menghitung volume
9	double vol = panjang*lebar*tinggi;
10	//mengembalikan nilai
11	return vol;
12	}
13	}
14	
15	class DemoMethod2{
16	public static void main(String[] args){
17	Kotak k1, k2, k3;
18	
19	//instansiasi objek
20	k1=new Kotak();
21	k2=new Kotak();
22	k3=new Kotak();
23	
24	//mengisi data untuk objek k1
25	k1.panjang=4;
26	k1.lebar=3;

27	k1.tinggi=2;
28	
29	//mengisi data untuk objek k2
30	k2.panjang=6;
31	k2.lebar=5;
32	k2.tinggi=4;
33	
34	//mengisi data untuk objek k3
35	k3.panjang=8;
36	k3.lebar=7;
37	k3.tinggi=6;
38	System.out.println("Volume k1 = "+k1.hitungVolume());
39	System.out.println("Volume k2 = "+k2.hitungVolume());
40	System.out.println("Volume k3 = "+k3.hitungVolume());
41	}
42	}

	Program Latihan Praktikum 2.5.3
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	
13	double hitungVolume(){
14	return(panjang*lebar*tinggi);
15	}
16	}
17	
18	class DemoMethod3{
19	public static void main(String[] args){
20	Kotak k;
21	
22	//instansiasi objek
23	k = new Kotak();
24	
25	//memanggil method isiData()
26	k.isiData(4,3,2);
27	
37	System.out.println("Volume kotak = " + k.hitungVolume());
38	}
39	}

Contoh penggunaan return:



	Program Latihan Praktikum 2.5.4
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	
14	//fungsi untuk menginputkan suatu nilai
15	System.out.print("Panjang = "); int a=masuk.nextInt();
16	System.out.print("Lebar = "); int b=masuk.nextInt();
17	System.out.print("Tinggi = "); int c=masuk.nextInt();
18	
19	balok coba = new balok();
20	System.out.print("\nVolume balok = "+
21	coba.volume(a,b,c));
22	}
23	}

	Program Latihan Praktikum 2.5.5
1	public class persegi {
2	static int hitungluas(int p, int l){
3	return p*l;
4	}
5	
6	public static void main(String[] args) {
7	int z=0;
8	z=hitungluas(3,2);
9	System.out.println(z);
10	}
11	}

## 2.6 Hasil Percobaan

1. Berapakah volume yang ditampilkan untuk ketiga kotak pada praktikum 2.5.1 di atas ?
2. Berapakah volume yang ditampilkan untuk ketiga kotak pada praktikum 2.5.2 di atas ?
3. Berapakah volume yang ditampilkan untuk kotak pada praktikum 2.5.3 di atas ?
4. Berapakah volume yang ditampilkan untuk balok pada praktikum 2.5.4 di atas ?
5. Berapakah luas yang ditampilkan untuk persegi pada praktikum 2.5.5 di atas ?

## 2.7 Analisis Hasil

Lakukan analisis pada program 2.5.1 sampai 2.5.5 di atas. Jelaskan prosesnya dan perbedaan untuk masing-masing prosesnya.

## **2.8 Kesimpulan**

Berikan kesimpulan yang kamu dapatkan dari praktikum ini :

.....  
.....  
.....

## **2.9 Latihan**

Lakukan modifikasi terhadap program-program di atas, dengan melakukan perubahan program terkait penggunaan karakteristik dari atribut dan method.

## **2.10 Tugas**

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

## **2.11 Daftar Pustaka**

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms Using Java 6 edition", Wiley, USA, 2014.
- John R. Hubbard, "Scaum's Outline of Data Structures With Java second Edition", McGraw-Hill, New york, 2007.
- Robert Lafore, "Data Structures and Algorithm in Java second Edition", Sams Publishing, Indiana, 2003

## Modul 3 : ADT Array

### 3.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan tugas / Studi Kasus
- 50 menit **Pengayaan**

### 3.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami konsep array 1 dimensi dan 2 dimensi
2. Memahami cara membuat array 1 dimensi
3. Memahami cara menyimpan dan mengakses data yang tersimpan di dalam array

### 3.3 Alat & Bahan

1. Komputer
2. Java IDE

### 3.4 Dasar Teori

Array adalah object yang digunakan untuk menyimpan banyak data dengan tipe data yang sama. Tipe data dari array dapat berupa int, float, string ataupun dapat berupa class. Array dapat dibentuk mulai dari 1 dimensi, 2 dimensi sampai "n" dimensi. Sebuah array dapat diibaratkan sebuah tabel yang memiliki indeks dan isi dari array tersebut. Seluruh isi dari array memiliki tipe data yang sama dan dapat diisi sesuai kebutuhan. Sedangkan indeks dari array, adalah penomoran yang secara default diberikan kepada array tersebut. Pengisian indeks array selalu dilakukan mulai dari 0, 1, 2 dst.

Indeks	→	0	1	2	3	4
Isi array	→	9	4	2	5	4

Gambar 0.1 Array 1 Dimensi

Pada Gambar 0.1 array 1 dimensi memiliki 5 data di dalamnya, sehingga indeks terkecil dimulai dari 0 dan indeks terbesar adalah 4. Indeks merepresentasikan isi data dalam array misalkan array dengan indeks 3 memiliki data yang bernilai 5.

		Indeks				
		↓				
Indeks	→	0	1	2	3	4
0		9	5	8	2	7
1		5	8	2	7	9
2		2	9	5	7	8
3		5	8	7	9	2
4		7	2	9	8	5

Gambar 0.2 Array 2 Dimensi

Pada Gambar 0.2 tentang array 2 dimensi, sedikit berbeda. Array 2 dimensi memiliki hal seperti tabel dan kolom dengan indeks yang merepresentasikan tabel dan kolom tersebut. Indeks dimulai dari 0,0 hingga 4,4. Contoh indeks data 2,4 memiliki data yang bernilai 9.

### Mendeklarasikan variabel array

Terdapat 3 langkah untuk membuat array:

1. Mendeklarasikan variabel array
2. Men-create array beserta ukurannya
3. Memberikan Sebuah nilai pada setiap elemen array

Untuk mendeklarasikan variabel array dapat dibuat dengan beberapa format. Array 1 dimensi dan 2 ataupun “n” dimensi memiliki beberapa pemberian format yang mirip, namun pada beberapa hal tidak sama.

Array 1 dimensi dapat dibuat dengan format berikut

*“tipe\_data [ukuran] nama\_array;”*

*“tipe\_data”* adalah tipe data yang dipergunakan dalam sebuah array, seluruh data pada array akan memiliki tipe data yang sama.

*“ukuran”* adalah panjang ukuran dari sebuah array tersebut

*“nama\_array”* adalah nama variabel yang dipergunakan untuk array tersebut

Contoh:

Int [ ] angka;

String [ ] alphabet;

Double [ ] jarijari;

Sedangkan untuk array 2 dan “n” dimensi dengan format berikut

Untuk 2 dimensi *“tipe\_data [ukuran\_baris] [ukuran\_kolom] nama\_array;”*

Untuk “n” dimensi *“tipe\_data [ukuran] [ukuran] .... nama\_array;”*

Contoh :

Int [ ] [ ] angka;

Int [ ] [ ] [ ] .... angka;

*“....”* Dapat ditambahkan array / [ ] sesuai dengan kebutuhan akan data yang akan dipergunakan.

### Membuat Array (Men-create array dengan ukurannya)

Karena array adalah sebuah *object*, maka array dibuat dengan operator *“new”*. Sehingga panjang data yang dimiliki array ditentukan pada saat runtime. Contoh :

```
int [ ] desimal;  
desimal = new int[10]
```

Pada saat array dibuat, isi dari array diinisialisasi dengan default *value*. Default *value* isi dari array adalah *“0”* untuk tipe data primitif numerik, *“\u0000”* untuk tipe data char, *“false”* untuk tipe data boolean

### Memberikan nilai / isi array pada setiap elemen array

Mengisikan nilai pada array dapat dilakukan dengan cara sebagai berikut

Cara ke-1

```
double rata_rata = new double[3];  
rata_rata [0] = 21.32;  
rata_rata [1] = 11.06;  
rata_rata [2] = 80.36;
```

Cara ke-2

```
int [ ] angka = {4, 1, 5, 9, 0}
```

0	1	2
21.32	11.06	80.36

**Gambar 0.3 Array yang didapatkan dari cara ke-1**

Sehingga dari cara ke-1 didapatkan isi data seperti pada Gambar 0.3, sedangkan pada cara ke-2 didapatkan isi data dengan indeksnya seperti pada gambar Gambar 0.4 yang tiap indeksnya menyesuaikan dengan data yang diisikan.

0	1	2	3	4
4	1	5	9	0

**Gambar 0.4 Array yang didapatkan dari cara ke-2**

Pada Cara ke-1 indeks diberikan nilai isi array satu persatu. Sedangkan pada Cara ke-2 isi dari array diberikan untuk beberapa indeks secara langsung. Baik cara ke-1 maupun cara ke-2 adalah untuk pengisian nilai array pada array 1 dimensi. Sedangkan untuk array 2 ataupun “n” dimensi dapat dilakukan dengan cara ke-3 dan ke-4 berikut.

Cara ke-3

```
double ratarata = new double[2][2];
ratarata[0][0] = 21.32;
ratarata[0][1] = 11.06;
ratarata[0][2] = 80.36;
ratarata[1][0] = 40.26;
ratarata[1][1] = 85.55;
ratarata[1][2] = 13.41;
ratarata[2][0] = 37.51;
ratarata[2][1] = 74.30;
ratarata[2][2] = 88.21;
```

Pada Cara ke-3 akan menghasilkan array seperti Gambar 0.5, array dimasukkan sesuai indeks yang diberikan. Indeks masing-masing merepresentasikan baris dan kolom dari data yang dimasukkan. Tiap data yang dimasukkan harus memiliki indeks baris dan kolom.

0	1	2
21.32	40.26	37.51
11.06	85.55	74.30
80.36	13.41	88.21

**Gambar 0.5 Array yang didapatkan dari Cara ke-3**

Cara ke-4

```
int [ ] angka = {4, 33, 1, 14}, {40, 12, 9, 66}, {21, 8, 12, 42}, {9, 37, 6, 1};
```

Pada cara ke 4, menghasilkan seperti Gambar 0.6. Tiap data pada cara ke-4 yang berada dalam “{ }” akan dimasukkan pada baris indeks ke-0 dan akan masuk pada kolom ke-0 sampai ke-4.

	0	1	2	3
0	4	33	1	14
1	40	12	9	66
2	21	8	12	42
3	9	37	6	1

**Gambar 0.6 Array yang didapatkan dari Cara ke-4**

### Mengukur Besaran Array

Untuk mengukur besar dari array yang dimiliki dapat menggunakan fungsi. Misalkan jika deklarasi dari array memiliki ukuran sebesar 4000, namun isi dari array tidak penuh. Untuk mengetahui berapa data yang ada pada sebuah array yang berukuran tersebut.

*"nama\_array.length"*

Fungsi *".length"* ditaruh dibelakang nama array dari array yang akan diukur panjangnya.

Implementasi dapat dilihat pada kode berikut untuk array 1 dimensi.

```
long diameter;
diameter = new long[4000] ;
for (int i=0 ; i<squares.length ;i++)
{ diameter[i] = i * i ;
}
```

### Mengakses Nilai Array

Mengakses nilai array adalah sesuatu yang dilakukan untuk melihat / mengambil data yang ada pada sebuah array. Data pada dapat dilihat berdasar indeks yang dimiliki. Tidak harus urut dari indeks terkecil ke indeks yang paling besar untuk dapat melihat isi dalam array. Misalkan seperti pada array 1 dimensi berikut untuk memanggil array yang dimiliki pada Gambar 0.3.

```
System.out.println("Elemen pada Indeks 2 adalah angka ke-3, nominalnya adalah: " + anArray [2]);
```

Akan menghasilkan 80.36, karena angka tersebut adalah angka ke 3. Namun yang dipanggil adalah indeks yang kedua berdasarkan fungsi *"anArray[2]"*.

Sedangkan untuk array 2 dimensi berdasarkan Gambar 0.6 Menggunakan kode berikut

```
System.out.println("Elemen pada Indeks baris ke 0 dan kolom ke 2, nominalnya adalah: " + anArray [0] [2]);
```

Akan menghasilkan angka 1, karena yang diakses dari sebuah array adalah pada baris ke-[0] dan ke-[1] dari fungsi *"anArray [0] [2]"*.

## 3.5 Prosedur Praktikum

Prosedur dari praktikum ini adalah tuliskan dan jalankan kode berikut, kemudian tampilkan pada sub-bab hasil percobaan, analisislah pada sub-bab analisis hasil serta berilah kesimpulan untuk masing-masing percobaan yang diberikan pada sub-bab 0.5 ini.

Percobaan ke-1 : Membuat array, menyimpan array dan mengakses array 1 dimensi

```
1 class ArrayDemo {
2 public static void main(String[] args) {
3     int[] anArray;
4     anArray = new int[10];
5     // initialize each element array
6     anArray[0] = 10;
7     anArray[1] = 20;
8     anArray[2] = 30;
9     anArray[3] = 40;
```

```

10     anArray[4] = 50;
11     anArray[5] = 60;
12     anArray[6] = 70;
13     anArray[7] = 80;
14     anArray[8] = 90;
15     anArray[9] = 100;
16
17     System.out.println("Index Element 0: " + anArray[0]);
18     System.out.println("Index Element 1: " + anArray[1]);
19     System.out.println("Index Element 2: " + anArray[2]);
20     System.out.println("Index Element 3: " + anArray[3]);
21     System.out.println("Index Element 4: " + anArray[4]);
22     System.out.println("Index Element 5: " + anArray[5]);
23     System.out.println("Index Element 6: " + anArray[6]);
24     System.out.println("Index Element 7: " + anArray[7]);
25     System.out.println("Index Element 8: " + anArray[8]);
26     System.out.println("Index Element 9: " + anArray[9]);
27 }
28 }

```

Apakah hanya dapat dilakukan akses pada satu indeks array saja? Misalkan langsung pada indeks ke 5 / 9 dsb?

Jika pada "*anArray = new int[10];*" angka 10 anda rubah dengan 15, maka apa isi data dari indeks ke 10 sampai indeks ke 14?

Jika tipe data diubah ke string, double, float atau boolean. Maka jelaskan yang terjadi pada indeks ke 0 sampai 9? Dan jika isi data berjumlah 15, maka apa yang terjadi pada indeks ke 10 sampai 14 tentang masing-masing tipe data tersebut?

#### Percobaan ke-2 : Mengurutkan data dan menyisipkan data pada array

```

1  import java.util.Arrays;
2  public class MainClass {
3      public static void main(String args[]) throws Exception {
4          int array[] = { 2, 5, -2, 6, -3, 8, 0, -7, -9, 4 };
5          Arrays.sort(array);
6          printArray("Sorted array", array);
7          int index = Arrays.binarySearch(array, 1);
8          System.out.println("Didn't find 1 @ " + index);
9          int newIndex = -index - 1;
10         array = insertElement(array, 1, newIndex);
11         printArray("With 1 added", array);
12     }
13
14     private static void printArray(String message, int array[]) {
15         System.out.println(message + ": [length: " + array.length + "]");
16         for (int i = 0; i < array.length; i++) {
17             if (i != 0){
18                 System.out.print(", ");
19             }
20             System.out.print(array[i]);
21         }

```

```

22     System.out.println();
23 }
24
25     private static int[] insertElement(int original[],
26     int element, int index) {
27         int length = original.length;
28         int destination[] = new int[length + 1];
29         System.arraycopy(original, 0, destination, 0, index);
30         destination[index] = element;
31         System.arraycopy(original, index, destination, index + 1, length - index);
32         return destination;
33     }
34 }

```

Berapakah isi maksimal dari array tersebut?

Proses penyisipan seperti apakah yang terjadi? Disisipkan di depan, tengah atau belakang?

Percobaan ke-3 : Membandingkan dua buah array

```

1  public class Main {
2      public static void main(String[] args) throws Exception {
3          int[] array = {1,2,3,4,5,6};
4          int[] array1 = {1,2,3,4,5,6};
5          int[] array2 = {1,2,3,4};
6          System.out.println("Variabel array = array 1?" + Arrays.equals(array, array 1));
7          System.out.println("Variabel array = array 2?" + Arrays.equals(array, array 2));
8          System.out.println("Variabel array 1 = array 2?" + Arrays.equals(array1, array 2));
9      }
10 }

```

Bagaimanakah proses pembandingan antara tiap-tiap array yang dibandingkan?

Percobaan ke-4 : Array dua dimensi

```

1  public class Array2Dimensi{
2      public static void main(String[] args) {
3          int[][] a2 = new int[10][5];
4          for (int i=0; i<a2.length; i++) {
5              for (int j=0; j<a2[i].length; j++) {
6                  a2[i][j] = i;
7                  System.out.print(" " + a2[i][j]);
8              }
9              System.out.println("");
10             }
11         }
12     }

```

Berapa tabel dan kolom kah array tersebut?

Apakah array tersebut sudah memiliki isi? Jika sudah isinya apa? Kalau belum isikanlah.

Percobaan ke-5 : Menghitung panjang baris dan kolom pada array dua dimensi

```

1  public class Main {
2      public static void main(String args[]) {
3          String[][] data = new String[4][5];
4          System.out.println("Dimension 1: " + data.length);
5          System.out.println("Dimension 2: " + data[0].length);

```



6	}
7	}

Isikanlah data dengan beberapa data, kemudian hitunglah isi data pada array tersebut.

### 3.6 Hasil Percobaan

Pada sub-bab ini jelaskan tentang apa yang anda dapatkan setelah anda menjalankan masing-masing kode program tersebut.

Percobaan ke-1

Percobaan ke-2

Percobaan ke-3

Percobaan ke-4

Percobaan ke-5

### **3.7 Analisis Hasil**

Pada sub bab ini, jelaskan tentang fenomena yang terjadi pada array yang dijalankan pada tiap percobaan.

Percobaan ke-1

Percobaan ke-2

Percobaan ke-3

Percobaan ke-4

Percobaan ke-5

### **3.8 Kesimpulan**

Percobaan ke-1

--

Percobaan ke-2

--

Percobaan ke-3

--

Percobaan ke-4

--

Percobaan ke-5

--

### 3.9 Latihan

Terdapat array dengan data 30, 87, 90, 3, 1, 50, 23, 4, 25, 23, 40, 35, 47, 2, 33.

1. Buatlah Kodenya dalam bahasa Java.
2. Jalankan dan pastikan tidak ada errornya.

### 3.10 Tugas

#### Tugas 1

Berdasar latihan yang telah anda buat maka:

1. Urutkan data pada array tersebut.
2. Hitung rata-rata data tersebut.
3. Hitung nilai maksimal dan minimalnya.
4. Tampilkan data yang bilangan ganjil saja dan prima saja.
5. Dari data tersebut, buatlah array 2 dimensi dengan format 3 baris dan 5 kolom.

Hasil
-------

## Tugas 2

Tuliskan dan jalankan program berikut

```
1 public class Matrik{
2     private int nBris, nKolom;
3     private double [][]itemDt;
4     /**
5      * constructor untuk membuat suatu matrik
6      * @param nBrs : banyaknya baris
7      * @param nKlm : banyaknya kolom
8      */
9     public Matrik(int nBrs, int nKlm){
10         nBris = nBrs;
11         nKolom = nKlm;
12         itemDt = new double[nBris][nKolom];
13     }
14     /**
15      * constructor untuk membuat matrik dari array 2 dimensi
16      * @param A : array dua dimensi
17      */
18     public Matrik(double [][]A){
19         this(A.length,A[0].length); // panggil constructor
20         this.nBris = A.length;
21         this.nKolom = A[0].length;
22
23         for (int i=0; i<nBris; i++){
24             for (int j=0; j<nKolom; j++){
25                 this.itemDt[i][j] = A[i][j];
26             }
27         }
28     }
29     /**
30      * Fungsi untuk mendapatkan jumlah baris
31      * @return jumlah baris
32      */
33     public int getNBris(){ return nBris;}
34     public int getNKolom(){ return nKolom;}
35     public double getItem(int idB, int idK){
36         return this.itemDt[idB][idK];
37     }
38     public void setItem(int idB, int idK, double dt){
39         this.itemDt[idB][idK] = dt;
40     }
41     /**
42      * fungsi tambah antara dua matrik A dan B
43      * @param A : Matrik
44      * @param B : Matrik
45      * @return Matrik hasil
46      */
47     public static Matrik tambah(Matrik A, Matrik B){
48         // tambahkan bagian ini
49     }
```

```

50
51  /**
52  * fungsi static perkalian antara vektor dengan matrik
53  * Syarat : lebar L sama dengan jumlah baris M
54  * @param L : Vector (Larik)
55  * @param M : Matrik
56  * @return Vector (Larik) berdimensi nKolom dari M
57  */
58  public static Larik VektorKaliMatrik(Larik L, Matrik M){
59      Larik IHasil = null;
60      Larik IKolom = null;
61      if (L.getSize() == M.getNBaris()){
62          IHasil = new Larik(M.getNKolom());
63          for (int i=0; i<M.getNKolom(); i++){
64              IKolom = M.getKolom(i);
65              double hasil = Larik.LarikKaliLarik(L, IKolom);
66              System.out.println(hasil);
67              IHasil.isiltem(i, hasil);
68          }
69      }
70      return IHasil;
71  }
72
73  /**
74  * fungsi static tranpos suatu matrik
75  * @param A : Matrik
76  * @return Matrik tranpos
77  */
78  public static Matrik tranpos(Matrik A){
79      // lenkapi bagian ini
80  }
81
82  /**
83  * fungsi untuk mendapatkan vektor baris dari matrik
84  * @param idBaris : indek baris yang akan diekstrak
85  * @return Larik representasi baris
86  */
87  public Larik getBaris(int idBaris){
88      // lenkapi bagian ini
89  }
90
91  /**
92  * fugsi untuk mendapatkan vektor kolom suatu matrik
93  * @param idKolom : id kolom yang akan diekstrak
94  * @return Larik representasi kolom
95  */
96  public Larik getKolom(int idKolom){
97      Larik l = new Larik(this.nBaris);
98      for (int i=0; i<this.nBaris; i++){
99          double itemKolom = this.getItem(i, idKolom);
100         l.isiltem(i, itemKolom);

```

101	return l;
102	}
103	
104	/**
105	* procedure cetak
106	* @param kom
107	*/
108	public void cetak(String kom){
109	System.out.println(kom);
110	for (int i=0; i<this.nBaris; i++){
111	for (int j=0; j<this.nKolom; j++){
112	System.out.printf("%.2f ",this.itemDt[i][j]);
113	}
114	System.out.println();
115	}
116	}
117	

Setelah program diatas dijalankan, kemudian sisipkan program dibawah sehingga program dapat berjalan

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

Setelah program tersebut disisipkan harapannya adalah keluar hasil berikut

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
Transpose
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
--

- Apakah maksud dari A?
- Apakah maksud dari B?
- Apakah maksud dari C?
- Apakah pengertian dan hasil dari Transpose?
- Apakah maksud dari Baris ke 1 dari C?
- Apakah maksud dari Hasil kali C.L1?

### 3.11 DAFTAR PUSTAKA

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms Using Java 6 edition", Wiley, USA, 2014.
- John R. Hubbard, "Scaum's Outline of Data Structures With Java second Edition", McGraw-Hill, New york, 2007.
- Robert Lafore, "Data Structures and Algorithm in Java second Edition", Sams Publishing, Indiana, 2003



## Modul 4 : ADT Single Linked List

### 4.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut :

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan tugas / Studi Kasus
- 50 menit **Pengayaan**

### 4.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

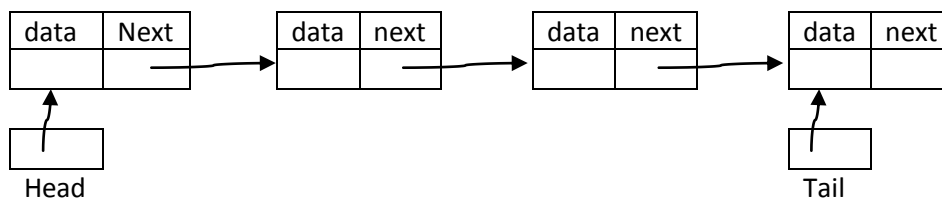
1. Mampu memahami konsep ADT linked list
2. Mampu mengaplikasikan ADT single linked list

### 4.3 Alat & Bahan

1. Komputer
2. Java IDE

### 4.4 Dasar Teori

Linked List merupakan struktur data yang dibangun dari satu atau lebih node yang menempati alokasi memori secara dinamis. Dalam setiap node menyimpan dua informasi utama yakni data dan pointer yang menunjuk ke node yang lain sehingga membentuk rangkaian node sebagaimana digambarkan berikut :



Jika linked list hanya berisi satu node maka pointer-nya akan menunjuk ke NULL. Jika linked list memiliki lebih dari satu node maka pointer menyimpan alamat dari node berikutnya. Sehingga antara node satu dengan node yang lain akan terhubung. Kecuali node paling ujung akan menunjuk ke NULL.

#### Single Linked List (SLL)

Single artinya pointer-nya hanya satu buah dan satu arah, yaitu menunjuk ke node sesudahnya. Node terakhir akan menunjuk ke NULL yang akan digunakan sebagai kondisi berhenti pada saat pembacaan isi linked list.

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

Node
Data : Object
Next : Node

SLL
head,tail: Node
size=0 : int
Inisialisasi():void
isEmpty(): boolean
size (): int
addFirst(Node input): void
addLast (Node input): void

### 4.5 Prosedur Praktikum

#### A. Percobaan Pertama: Memahami *Node* yang digunakan pada *Single Linked List*

1. Buatlah projek di IDE Java dengan nama *SingleLinkedListProject*
2. Buatlah Class *Node* di projek *SingleLinkedListProject* dan Tuliskan kode dibawah :

```

1  public class Node {
2      Object data;
3      Node next;
4
5      public static void main(String[] args)
6  {
7
8
9
10
11
12 }
13 }

```

3. Lakukan beberapa langkah berikut untuk memahami konsep Node pada Single Linked List:
  - a. Tambahkan kode `Node head = new Node();` pada baris ke 7 class Node.
  - b. Tambahkan kode `System.out.println("data : " + head.data);` pada baris ke 9 class Node kemudian jalankan program.
  - c. Tambahkan kode `System.out.println("pointer: " + head.next);` pada baris ke 10 class Node kemudian jalankan program.
  - d. Tambahkan kode `head.data = "A";` pada baris ke 8 class Node kemudian jalankan program.
4. Tuliskan hasil percobaan, analisis hasil dan kesimpulannya.

## B. Percobaan Kedua: Memahami Operasi sederhana pada Single Linked List

1. Tambahkan Class *SLL* di projek *SingleLinkedListProject* dan Tuliskan kode dibawah :

```

1  public class SLL {
2      Node head,tail;
3      int size=0;
4
5      void inisialisasi(){
6          head=null;
7      }
8
9      boolean isEmpty(){
10         return (size==0);
11     }
12
13     int size()
14     {
15         return size;
16     }
17
18     void addFirst(Node input){
19         if (isEmpty()){
20             head=input;
21             tail=input;
22         }
23         else

```

24	{
25	input.next = head;
26	head = input;
27	} size++;
28	}
29	
30	void addLast(Node input){
31	if (isEmpty()){
32	head = input;
33	tail = input;
34	}
35	else
36	{
37	tail.next = input;
38	tail = input;
39	} size++;
40	}
41	}

2. Tambahkan method main pada class SLL dengan dengan kode sebagai berikut kemudian jalankan.

1	public static void main(String[] args)
2	{
3	SLL list = new SLL();
4	System.out.println("head : " + list.head);
5	System.out.println("tail : " + list.tail);
6	list.addFirst(new Node());
7	System.out.println("head : " + list.head);
8	System.out.println("tail : " + list.tail);
9	list.addFirst(new Node());
10	System.out.println("head : " + list.head);
11	System.out.println("tail : " + list.tail);
12	list.addLast(new Node());
13	System.out.println("head : " + list.head);
14	System.out.println("tail : " + list.tail);
15	}

3. Ganti isi dari method main pada class SLL dengan kode sebagai berikut kemudian jalankan.

1	public static void main(String[] args)
2	{
3	SLL list = new SLL();
4	System.out.println("head : " + list.head);
5	System.out.println("tail : " + list.tail);
6	list.addLast(new Node());
7	System.out.println("head : " + list.head);
8	System.out.println("tail : " + list.tail);
9	list.addLast(new Node());
10	System.out.println("head : " + list.head);
11	System.out.println("tail : " + list.tail);
12	list.addLast(new Node());

13	System.out.println("head : " + list.head);
14	System.out.println("tail : " + list.tail);
15	}

4. Tuliskan hasil percobaan, analisis hasil dan kesimpulannya.

#### 4.6 Hasil Percobaan

1. Tuliskan hasil dari percobaan pertama diatas.
2. Tuliskan hasil dari percobaan kedua diatas.

#### 4.7 Analisis Hasil

1. Tuliskan Analisis hasil dari percobaan pertama diatas.
2. Tuliskan Analisis hasil dari percobaan kedua diatas.

#### 4.8 Kesimpulan

1. Tuliskan kesimpulan dari percobaan pertama diatas.
2. Tuliskan kesimpulan dari percobaan kedua diatas.

#### 4.9 Latihan

Operasi pada Single Linked List secara lengkap adalah sebagai berikut:

1. Inisialisasi
2. isEmpty
3. size
4. Penambahan
5. Penghapusan
6. Penyisipan
7. Pencarian
8. Pengaksesan

Lengkapi kode class SLL diatas untuk operasi-operasi (method) yang belum ada pada Single Linked List.

#### 4.10 Tugas

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

<b>Mahasiswa</b>
<b>String nim</b>
<b>String nama</b>
<b>double ipk</b>
<b>Constructor Mahasiswa</b>
<b>double getIpk</b>
<b>String getNim</b>
<b>String getNama</b>

2. Tambahkan method penyisipan data yang bisa membentuk linked list urut sejak awal dengan pengurutan berdasarkan ipk.

#### 4.11 DAFTAR PUSTAKA

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms Using Java 6 edition", Wiley, USA, 2014.

- John R. Hubbard, "Scaum's Outline of Data Structures With Java second Edition", McGraw-Hill, New york, 2007.
- Robert Lafore, "Data Structures and Algorithm in Java second Edition", Sams Publishing, Indiana, 2003

## Modul 5 : ADT Double Linked List

### 5.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan tugas / Studi Kasus
- 50 menit **Pengayaan**

### 5.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

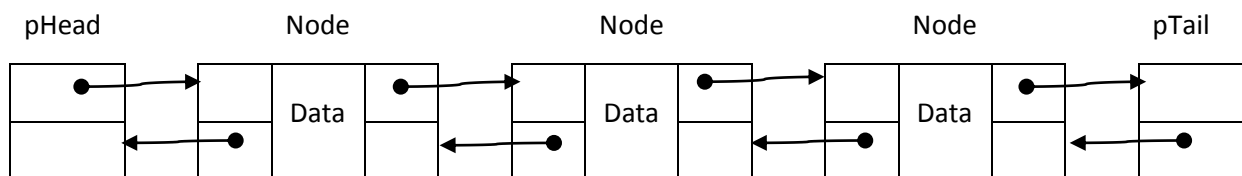
1. Mampu memahami konsep ADT double linked list
2. Mampu mengaplikasikan ADT double linked list

### 5.3 Alat & Bahan

1. Komputer
2. Java IDE

### 5.4 Dasar Teori

Double Linked List merupakan representasi data yang disimpan dalam suatu rangkaian node dimana setiap node mempunyai penunjuk ke node sebelumnya dan sesudahnya. Double : artinya field pointer-nya dua buah dan dua arah, yang menunjuk ke node sebelum dan sesudahnya. Berguna bila perlu melakukan pembacaan linkedlist dari dua arah. Double linked list memiliki 2 buah pointer yaitu pointer next dan prev. Pointer next : mengarah ke node belakang (tail). Pointer prev : mengarah ke node depan (head). Ilstrasi double Linked List dapat digambarkan berikut :



Ketika masih ada satu node maka kedua pointer (next dan prev) akan menunjuk ke NULL. Ketika double linked list memiliki banyak node maka node yang paling depan pointer prev-nya menunjuk ke NULL. Sedangkan node yang paling belakang pointer next-nya yang menunjuk ke NULL. Pada double linked dibutuhkan pointer bantu yaitu head dan tail. Head : menunjuk pada node yang paling depan. Tail : menunjuk pada node yang paling belakang. Dari ilustrasi gambar di atas ADT double Linked-list dapat direpresentasikan sebagai berikut :

Node
Data : Object
Next : Node
Prev : Node

DLL
head,tail: Node
size=0 : int
inisialisasi():void
isEmpty(): boolean
size (): int
addFirst(Node input): void
addLast (Node input): void

### 5.5 Prosedur Praktikum

#### C. Percobaan Pertama: Memahami Node yang digunakan pada Double Linked List

1. Buatlah proyek di IDE Java dengan nama DoubleLinkedListProject

2. Buatlah Class *Node* di proyek *DoubleLinkedListProject* dan Tuliskan kode dibawah :

```
1  public class Node {
2      Object data;
3      Node next;
4      Node prev;
5      public static void main(String[] args)
6      {
7
8
9
10
11
12  }
13 }
```

3. Lakukan beberapa langkah berikut untuk memahami konsep Node pada Double Linked List:
- Tambahkan kode `Node head = new Node();` pada baris ke 7 class Node.
  - Tambahkan kode `System.out.println("data : " + head.data);` pada baris ke 9 class Node kemudian jalankan program.
  - Tambahkan kode `System.out.println("pointer: " + head.next);` pada baris ke 10 class Node kemudian jalankan program.
  - Tambahkan kode `System.out.println("pointer: " + head.prev);` pada baris ke 11 class Node kemudian jalankan program.
  - Tambahkan kode `head.data = "A";` pada baris ke 8 class Node kemudian jalankan program.
4. Tuliskan hasil percobaan, analisis hasil dan kesimpulannya.

#### D. Percobaan Kedua: Memahami Operasi sederhana pada Double Linked List

5. Tambahkan Class *DLL* di proyek *DoubleLinkedListProject* dan Tuliskan kode dibawah :

```
1  public class DLL {
2      Node head,tail;
3      int size=0;
4
5      void inisialisasi(){
6          head=null;
7      }
8
9      boolean isEmpty(){
10         return (size==0);
11     }
12
13     int size()
14     {
15         return size;
16     }
17
18     void addFirst(Node input){
19         if (isEmpty()){
20             head=input;
21             tail=input;
```

```

22     }
23     else
24     {
25         input.next = head;
26         head.prev = input;
27         head = input;
28     } size++;
29 }
30
31 void addLast(Node input){
32     if (isEmpty()){
33         head = input;
34         tail = input;
35     }
36     else
37     {
38         input.prev = tail;
39         tail.next = input;
40         tail = input;
41     } size++;
42 }
43 }

```

6. Tambahkan method main pada class DLL dengan dengan kode sebagai berikut kemudian jalankan.

```

1  public static void main(String[] args)
2  {
3      DLL list = new DLL();
4      System.out.println("head : " + list.head);
5      System.out.println("tail : " + list.tail);
6      list.addFirst(new Node());
7      System.out.println("head : " + list.head);
8      System.out.println("tail : " + list.tail);
9      list.addFirst(new Node());
10     System.out.println("head : " + list.head);
11     System.out.println("tail : " + list.tail);
12     list.addLast(new Node());
13     System.out.println("head : " + list.head);
14     System.out.println("tail : " + list.tail);
15 }

```

7. Ganti isi dari method main pada class DLL dengan kode sebagai berikut kemudian jalankan.

```

1  public static void main(String[] args)
2  {
3      DLL list = new DLL();
4      System.out.println("head : " + list.head);
5      System.out.println("tail : " + list.tail);
6      list.addLast(new Node());
7      System.out.println("head : " + list.head);
8      System.out.println("tail : " + list.tail);

```



9	list.addLast(new Node());
10	System.out.println("head : " + list.head);
11	System.out.println("tail : " + list.tail);
12	list.addLast(new Node());
13	System.out.println("head : " + list.head);
14	System.out.println("tail : " + list.tail);
15	}

8. Tuliskan hasil percobaan, analisis hasil dan kesimpulannya.

#### 5.6 Hasil Percobaan

1. Tuliskan hasil dari percobaan pertama diatas.
2. Tuliskan hasil dari percobaan kedua diatas.

#### 5.7 Analisis Hasil

1. Tuliskan Analisis hasil dari percobaan pertama diatas.
2. Tuliskan Analisis hasil dari percobaan kedua diatas.

#### 5.8 Kesimpulan

1. Tuliskan kesimpulan dari percobaan pertama diatas.
2. Tuliskan kesimpulan dari percobaan kedua diatas.

#### 5.9 Latihan

Operasi pada Double Linked List secara lengkap adalah sebagai berikut:

1. Inisialisasi
2. isEmpty
3. size
4. Penambahan
5. Penghapusan
6. Penyisipan
7. Pencarian
8. Pengaksesan

Lengkapi kode class DLL diatas untuk operasi-operasi (method) yang belum ada pada Double Linked List.

#### 5.10 Tugas

1. Modifikasilah program Latihan di atas sehingga DLL dapat menampung sembarang object. Kemudian, gunakan class Mahasiswa yang pernah dibuat pada modul 4 untuk diisikan pada DLL. Berikut class mahasiswa yang dimaksud:

<b>Mahasiswa</b>
String nim String nama double ipk
<b>Constructor Mahasiswa</b> double getIpk String getNim String getNama

2. Gunakan dan modifikasi method penyisipan data pada tugas di modul 4 yang bisa membentuk linked list urut sejak awal dengan pengurutan berdasarkan ipk.

3. Lengkapi method pada DLL yang bisa menampilkan data secara *ascending* dan *descending* berdasarkan ipk.

#### **5.11 DAFTAR PUSTAKA**

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms Using Java 6 edition", Wiley, USA, 2014.
- John R. Hubbard, "Scaum's Outline of Data Structures With Java second Edition", McGraw-Hill, New york, 2007.
- Robert Lafore, "Data Structures and Algorithm in Java second Edition", Sams Publishing, Indiana, 2003

## Modul 6 : ADT Circular Linked List

### 6.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut.

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 30 menit untuk penyampaian materi
- 65 menit untuk pengerjaan tugas / Studi Kasus
- 60 menit **Pengayaan**

### 6.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

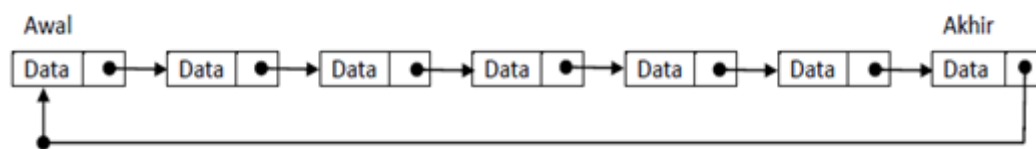
1. Mampu membedakan antara Linked List dengan Circular Linked List
2. Mampu menerapkan Circular Single Linked List dan Circular Double Linked List
3. Mampu mengimplementasikan Circular Linked List pada suatu studi kasus

### 6.3 Alat & Bahan

1. Komputer
2. Java IDE

### 6.4 Dasar Teori

Salah satu varian dari Linked List adalah Circular Linked List yang mempunyai dua field setiap node-nya, yaitu field pointer yang menunjuk ke node setelahnya dan sebuah field yang berisi data untuk node tersebut. Perbedaan antara Linked List dengan Circular Linked List adalah pada Circular Linked List pointer node terakhirnya mengarah ke node awal. Adapun struktur dari Circular Linked List ditampilkan pada gambar berikut.



Circular Linked List terdiri dari Circular Single Linked List dan Circular Double Linked List. Pada gambar di atas merupakan contoh Circular Single Linked List, karena hanya memiliki satu buah pointer dan arahnya hanya ke node setelahnya. Ketika node baru terbentuk, pointer-nya menunjuk ke dirinya sendiri. Jika sudah lebih dari satu node, maka pointer-nya menunjuk ke node setelahnya dan pointer node akhir menuju ke node awal.

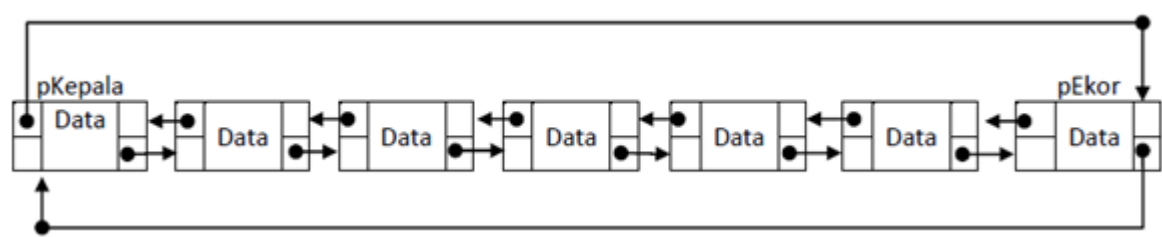
Adapun ADT dari Circular Single Linked List adalah sebagai berikut.

<b>NodeCSLL</b>
<b>Object data</b>
<b>NodeCSLLsetelah</b>

<b>CircularLinkedList</b>
<b>NodeCSLLpAwal</b>
<b>NodeCSLL pAkhir</b>
<b>int jumlah</b>
<b>CircularLinkedList ()</b>
<b>public void sisipDtDiAkhir(Object dt)</b>
<b>public void sisipDtDiAwal(Object dt)</b>
<b>public void hapusDt(Object dtHapus)</b>
<b>public void hapusSatuDataDiAwal()</b>
<b>public void hapusSatuDataDiAkhir()</b>
<b>public void cetakDt(String komentar)</b>

Berbeda dengan Circular Single Linked List, Circular Double Linked List mempunyai dua pointer, yaitu pointer yang mengarah ke node setelahnya dan ke node sebelumnya. Pada pembentukan node baru, kedua pointer mengarah ke dirinya sendiri. Jika sudah lebih dari satu node, maka pointer pada node baru akan mengarah ke

node setelahnya dan node sebelumnya. Pada node akhir pointer setelahnya mengarah ke node awal, dan pada node awal pointer sebelumnya mengarah ke node akhir. Adapun struktur dan ADT dari Circular Double Linked List adalah sebagai berikut.



NodeCDLL
Object data
NodeCDLLsebelum
NodeCDLLsetelah

CircularDLinkedList
NodeCDLLpAwal
NodeCDLL pAkhir
int jumlah
CircularDLinkedList ()
public void sisipDtDiAkhir(Object dt)
public void sisipDtDiAwal(Object dt)
public void hapusDt(Object dtHapus)
public void cetakDt(String komentar)

## 6.5 Prosedur Praktikum

- Percobaan Pertama: Implementasi Circular Single Linked List
  - Buatlah proyek di IDE Java dengan nama CircularSingleLinkedList.
  - Buatlah Class CircularSingleLinkedList di proyek yang telah dibuat, dan tambahkan kelas Node seperti kode di bawah ini sesuai dengan ADT Class NodeCSLL.

1	class NodeCSLL {
2	Object data;
3	NodeCSLL setelah;
4	}
5	Public class CircularSingleLinkedList {
6	
7	
8	}

- Tambahkan beberapa prosedur sesuai dengan ADT Class CircularLinkedList dan implementasikan sesuai dengan kode di bawah ini.

1	class NodeCSLL {
2	Object data;
3	NodeCSLL setelah;
4	}
5	Public class CircularSingleLinkedList {
6	private NodeCSLL pAwal, pAkhir;
7	private int jumlah;
8	public CircularSingleLinkedList(){
9	pAwal = null;
10	pAkhir = null;
11	jumlah = -1;
12	}
13	Public void SisipDataDiAwal(Object data){
14	NodeCSLL pBaru = new NodeCSLL();
15	pBaru.data = data;
16	pBaru.setelah= pBaru;

```

17         if (pAwal == null){
18             pAwal = pBaru;
19             pAkhir = pBaru;
20             jumlah = 0;
21         } else {
22             pBaru.setelah= pAwal;
23             pAkhir.setelah = pBaru;
24             pAwal = pBaru;
25             jumlah++;
26         }
27     }
28     Public void SisipDataDiAkhir(Object data){
29         // lengkapi bagian ini
30     }
31     Public void hapusData(Object dtHapus){
32         if(pAwal != null) {
33             NodeCSLL pSbl, pKini,pHapus;
34             pSbl = null; pKini = pAwal;
35             boolean ketemu = false;
36             inti = 0;
37             while(!ketemu && (i <= jumlah)){
38                 if (pKini.data.equals(dtHapus)) {
39                     ketemu = true;
40                 }
41                 else {
42                     pSbl = pKini;
43                     pKini = pKini.setelah;
44                 }
45                 i++;
46             }
47             if (ketemu){
48                 if(pSbl == null) {
49                     pHapus = pAwal;
50                     pAwal = pKini.setelah;
51                     pAkhir.setelah = pAwal;
52                     pHapus = null;
53                 } else {
54                     if (pAkhir == pKini) {
55                         pAkhir = pSbl;
56                     }
57                     pSbl.setelah= pKini.setelah;
58                     pHapus = pKini;
59                     pHapus = null;
60                 }
61                 jumlah--;
62             }
63         }
64     }
65     Public void hapusSatuDataDiAwal(){
66         // lengkapi bagian ini
67     }

```

68	Public void hapusSatuDataDiAkhir(){
69	// lengkapi bagian ini
70	}
71	Public void cetak(String Komentar){
72	System.out.println(Komentar);
73	NodeCSLL pCetak;
74	pCetak = pAwal;
75	inti = -1;
76	while((i < jumlah) ){
77	System.out.print(pCetak.data+"->");
78	pCetak = pCetak.setelah;
79	i++;
80	}
81	System.out.println();
82	}
83	Public static void main(String[] args) {
84	CircularSingleLinkedList csll =
85	new CircularSingleLinkedList();
86	csll.SisipDataDiAwal(new Integer(50));
87	csll.SisipDataDiAwal(new Integer(60));
88	csll.SisipDataDiAwal(new Integer(70));
89	csll.SisipDataDiAwal(new Integer(8));
90	csll.SisipDataDiAwal(new Integer(9));
91	csll.SisipDataDiAwal(new Integer(90));
92	csll.SisipDataDiAwal(new Integer(19));
93	csll.cetak("csll Asal");
94	csll.hapusData(8);
95	csll.cetak("csll stl 8 dihapus");
96	csll.hapusData(90);
97	csll.cetak("csll stl 90 dihapus");
98	}
99	}

4. Lakukan kompilasi dan tuliskan hasil percobaan, analisis hasil, dan kesimpulannya.

B. Percobaan Kedua: Implementasi Circular Double Linked List

1. Buatlah projek di IDE Java dengan nama CircularDoubleLinkedList.
2. Buatlah Class CircularDoubleLinkedList di projek yang telah dibuat, dan tambahkan kelas Node seperti kode di bawah ini sesuai dengan ADT Class NodeCDLL.

1	class NodeCDLL {
2	Object data;
3	NodeCDLL sebelum;
4	NodeCDLL setelah;
5	}
6	Public class CircularDoubleLinkedList {
7	
8	}

3. Tambahkan beberapa prosedur sesuai dengan ADT Class CircularDLinkedList dan implementasikan sesuai dengan kode di bawah ini.

1	class NodeCDLL {
2	Object data;
3	NodeCDLL sebelum;

4	NodeCDLL setelah;
5	}
6	Public class CircularDoubleLinkedList {
7	private NodeCSLL pAwal, pAkhir;
8	private intjumlah;
9	public CircularDoubleLinkedList(){
10	pAwal = null;
11	pAkhir = null;
12	jumlah = -1;
13	}
14	Public void SisipDataDiAwal(Object data){
15	NodeCDLL pBaru = newNodeCDLL();
16	pBaru.data = data;
17	pBaru.sebelum= pBaru;
18	pBaru.setelah= pBaru;
19	if (pAwal == null){
20	pAwal = pBaru;
21	pAkhir = pBaru;
22	jumlah = 0;
23	} else {
24	pBaru.sebelum = pAkhir;
25	pBaru.setelah = pAwal;
26	pAwal.sebelum = pBaru;
27	pAkhir.setelah = pBaru;
28	pAwal = pBaru;
29	jumlah++;
30	}
31	}
32	Public void SisipDataDiAkhir(Object data){
33	// lengkapi bagian ini
34	}
35	Public void hapusData(Object dtHapus){
36	// lengkapi bagian ini
37	}
38	Public void cetak(String Komentar){
39	System.out.println(Komentar);
40	NodeCDLL pCetak;
41	pCetak = pAwal;
42	inti =-1;
43	while((i < jumlah) ){
44	System.out.print(pCetak.data+"->");
45	pCetak = pCetak.setelah;
46	i++;
47	}
48	System.out.println();
49	}
50	Public static void main(String[] args) {
51	CircularDoubleLinkedList cdll =
52	New CircularDoubleLinkedList();
53	cdll.SisipDataDiAwal(new Integer(50));
54	cdll.SisipDataDiAwal(new Integer(60));

55	<code>cdll.SisipDataDiAwal(new Integer(70));</code>
56	<code>cdll.SisipDataDiAwal(new Integer(8));</code>
57	<code>cdll.SisipDataDiAwal(new Integer(9));</code>
58	<code>cdll.SisipDataDiAwal(new Integer(90));</code>
59	<code>cdll.SisipDataDiAwal(new Integer(19));</code>
60	<code>cdll.cetak("cdll Asal");</code>
61	<code>}</code>
62	<code>}</code>

4. Lakukan kompilasi dan tuliskan hasil percobaan, analisis hasil, dan kesimpulannya.

#### 6.6 Hasil Percobaan

1. Tuliskan hasil dari percobaan pertama di atas!
2. Tuliskan hasil dari percobaan kedua di atas!

#### 6.7 Analisis Hasil

1. Percobaan pertama:
  - a. Apakah fungsi kondisi pada while baris ke 37?
  - b. Apabila data yang dihapus tidak ditemukan, apa nilai dari variabel ketemu dan i ketika berada pada baris ke 47?
2. Jelaskan apa perbedaan antara percobaan kedua dibandingkan dengan percobaan pertama!

#### 6.8 Kesimpulan

1. Tuliskan kesimpulan dari percobaan pertama di atas!
2. Tuliskan kesimpulan dari percobaan kedua di atas!

#### 6.9 Latihan

Lengkapi kode Class CircularSingleLinkedList dan CircularDoubleLinkedList di atas pada bagian prosedur yang memiliki komentar 'lengkapi bagian ini'! Tambahkan perintah untuk menjalankan prosedur tersebut di method main untuk mengetahui apakah prosedur tersebut sudah benar!

#### 6.10 Tugas

1. Modifikasi program Latihan CircularSingleLinkedList dan CircularDoubleLinkedList tanpa menggunakan variabel jumlah!
2. Modifikasi program Latihan CircularSingleLinkedList dan CircularDoubleLinkedList tanpa menggunakan pointer pAkhir!

#### 6.11 DAFTAR PUSTAKA

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms Using Java 6 edition", Wiley, USA, 2014.
- John R. Hubbard, "Scaum's Outline of Data Structures With Java second Edition", McGraw-Hill, New york, 2007.
- Robert Lafore, "Data Structures and Algorithm in Java second Edition", Sams Publishing, Indiana, 2003



## Modul 7 : ADT Stack

### 7.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan tugas / Studi Kasus
- 50 menit **Pengayaan**

### 7.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

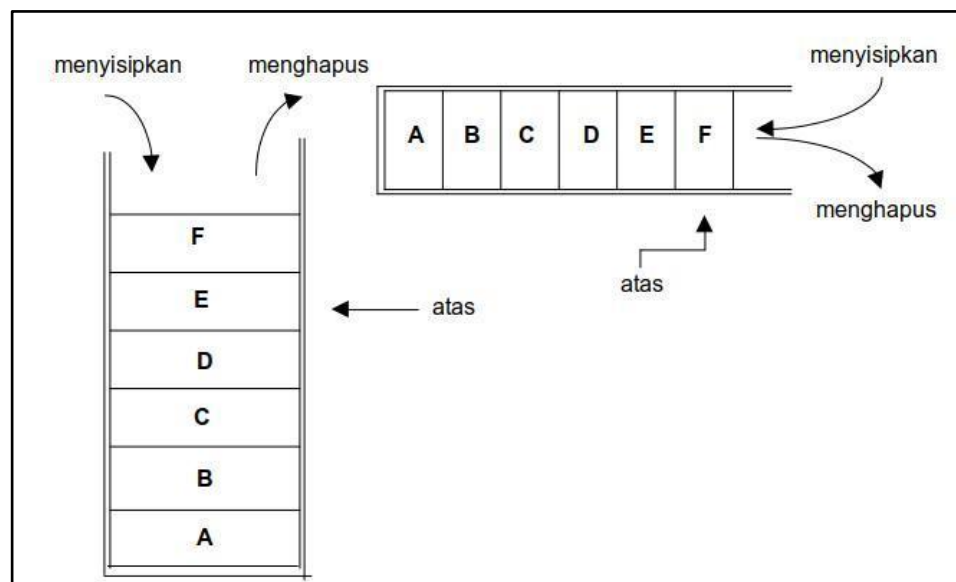
1. Mampu membuat stack menggunakan array
2. Mampu membuat stack menggunakan linked list.....

### 7.3 Alat & Bahan

1. Komputer
2. Java IDE

### 7.4 Dasar Teori Stack

Stack atau tumpukan adalah kumpulan data yang hanya bisa dilakukan penambahan (penyisipan) data dan penghapusan data pada salah satu ujung yang sama.



Dengan memperlihatkan ilustrasi-ilustrasi yang disebutkan maka kita bisa melihat bahwa stack merupakan suatu list yang mempunyai karakteristik “masuk terakhir keluar pertama” (last in first out – LIFO).

Operasi dasar pada stack antara lain pop dan push. **Push** adalah operasi menambah/menyisipkan item pada stack. **Pop** adalah proses mengambil item dari stack. Stack bisa dibuat dengan Array atau Linked List. Tabel dibawah memberikan ilustrasi operasi pada stack yang dibuat dengan array.

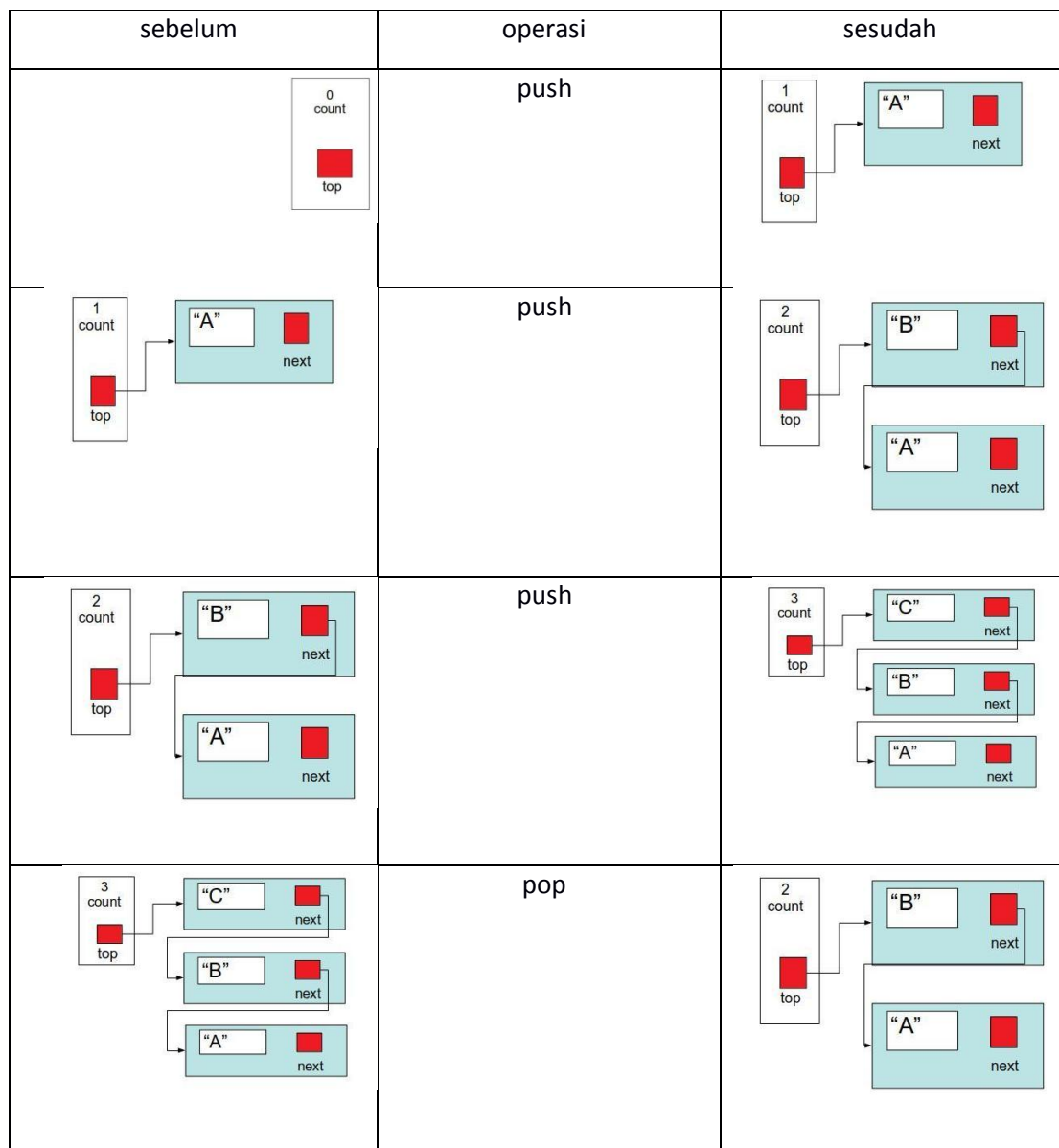
sebelum	operasi	sesudah
[]	push("A")	["A"]
["A"]	push("B")	["A","B"]
["A","B"]	push("C")	["A","B","C"]
["A","B","C"]	pop()	["A","B"]
["A","B"]	push("D")	["A","B","D"]
["A","B","D"]	pop()	["A","B"]

Dalam implementasi stack ini disimpan variabel **count** yang menghitung index atau nomor urut terakhir item dalam stack. Operasi push menambahkan item ke dalam array pada index/urutan **count+1** atau **setelah item terakhir** dalam array. Operasi pop akan mengambil item dari dalam array pada index **count** atau **item terakhir** dalam array.

Pada tabel dibawah diilustrasikan operasi stack menggunakan array. Perbedaan dengan stack diatas adalah implementasi proses **push** akan menambahkan item pada **array index ke 0**. Proses **pop** akan mengambil item pada **array index ke 0**.

sebelum	operasi	sesudah
[]	push("A")	["A"]
["A"]	push("B")	["B","A"]
["B","A"]	push("C")	["C","B","A"]
["C","B","A"]	pop()	["B","A"]
["B","A"]	push("D")	["D","B","A"]
["D","B","A"]	pop()	["B","A"]

Gambar dibawah memberikan ilustrasi stack menggunakan single linked list. Top merupakan pointer yang menunjuk pada item paling atas yang merupakan item terakhir yang dimasukkan dalam stack. Dalam ilustrasi ini operasi **push** pada stack menggunakan operasi **add first** pada single linked list dan operasi **pop** menggunakan operasi **remove first**. Jika operasi **push** menggunakan operasi **add last** maka operasi **pop** akan menggunakan **remove last**.



## 7.5 Prosedur Praktikum

1. Buatlah file percobaan stack\_array.java berikut....

```

1  import java.util.Scanner;
2
3  Public class stack_array {
4  Scanner masuk = new Scanner(System.in);
5
6  int choice, i;
7  char item;
8  char arr_stack[MAX_SIZE];
9  int count = 0;
10 int keluar = 0;
11

```

```

12 Public void push(char item)
13 {
14     if (count == MAX_SIZE)
15     {
16         system.out.print ( "\n# Stack Penuh");
17     }
18     else
19     {
20         .....
21         .....
22         .....
23     }
24 }
25
26 Public void pop()
27 {
28     if (count == 0)
29         system.out.print ( "\n## Stack kosong");
30     else
31     {
32         .....
33         .....
34     }
35 }
36
37 Public void printAll()
38 {
39     system.out.print ( "\n## Stack Size : " + count);
40     for (i = (count - 1); i >= 0; i--)
41         system.out.print ( "\n## No Urut/index : " + i + ", Value : " + arr_stack[i]);
42 }
43
44 Public void menu()
45 {
46     system.out.print ( "\nMasukkan operasi yang akan dilakukan (1:push, 2:pop,
47 3:print) : ");
48     choice=masuk.nextInt();
49     switch (choice)
50     {
51     case 1:
52     {
53         system.out.print ( "\nMasukkan huruf yang akan dipush : ");
54         item= masuk.nextLine();
55         push(item);
56     }
57     case 2:pop();
58     break;
59     case 3:printAll();
60     break;
61     default:

```

62	system.out.print ( "\n1:push, 2:pop, 3:print\n");
63	keluar = 1;
64	break;
65	}
66	}
67	
68	Public static void main()
69	{
70	do{
71	menu();
72	} while (keluar == 0);
73	}
74	}

2. Tambahkan kode berikut pada baris 20 sampai 22

1	arr_stack[count] = item;
2	System.out.print ( "\n# PUSH No urut/index : " + count + ", Push : " + item);
3	count++;

3. Tambahkan kode berikut pada baris 32 dan 33

1	--count;
2	System.out.print ( "\n##POP No urut/index : " + count + ", Value : " + arr_stack[count]);

4. Jalankan program dan

- pilih menu push dan masukkan "A"
- pilih menu push dan masukkan "B"
- pilih menu push dan masukkan "C"
- pilih menu print
- pilih menu pop
- pilih menu print
- pilih menu push dan masukkan "D"
- pilih menu print

## 7.6 Hasil Percobaan

Tuliskan hasil dari percobaan diatas

## 7.7 Analisis Hasil

Tuliskan Analisis hasil dari percobaan diatas.

## 7.8 Kesimpulan

Tuliskan kesimpulan dari percobaan diatas

## 7.9 Latihan

Buat pseudocode untuk operasi pop dan push pada stack menggunakan array dengan ketentuan operasi push akan menambahkan item pada array index ke 0 dan operasi pop akan mengembalikan item pada array index ke 0

### **7.10 Tugas**

Modifikasi program `stack_array.java` sehingga operasi push akan menambahkan item pada array index ke 0 dan operasi pop akan mengembalikan item pada array index ke 0.

### **7.11 DAFTAR PUSTAKA**

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms Using Java 6 edition", Wiley, USA, 2014.
- John R. Hubbard, "Scaum's Outline of Data Structures With Java second Edition", McGraw-Hill, New york, 2007.
- Robert Lafore, "Data Structures and Algorithm in Java second Edition", Sams Publishing, Indiana, 2003

## Modul 8 : ADT Queue

### 8.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan tugas / Studi Kasus
- 50 menit **Pengayaan**

### 8.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mampu membuat queue menggunakan array
2. Mampu membuat queue menggunakan linked list.....

### 8.3 Alat & Bahan

1. Komputer
2. Java IDE

### 8.4 Dasar Teori

Queue atau antrian adalah kumpulan data yang bisa dilakukan penambahan (penyisipan) data pada satu sisi dan penghapusan data pada sisi/ujung yang lain.



Dengan memperhatikan ilustrasi diatas maka kita bisa melihat bahwa queue merupakan suatu list yang mempunyai karakteristik “masuk pertama keluar pertama” (first in first out – FIFO). Struktur data ini banyak dipakai dalam informatika misalnya untuk merepresentasi :

1. Antrian job dalam sistem operasi
2. Antrian dalam dunia nyata

Operasi dasar pada queue antara lain **enqueue** dan **dequeue**. **Enqueue** adalah operasi menambah/menyisipkan item pada akhir queue. Pada proses enqueue item ditambahkan di belakang item terakhir dalam queue. **Dequeue** adalah proses mengambil item dari queue. Ketika proses dequeue item dalam queue akan berkurang yaitu item paling depan dalam queue atau item yang pertama kali ditambahkan kedalam queue. Queue bisa dibuat dengan Array atau Linked List. Tabel dibawah memberikan ilustrasi operasi pada queue yang dibuat dengan array.

sebelum	operasi	sesudah
[]	queue("A")	["A"]
["A"]	queue("B")	["A","B"]
["A","B"]	queue("C")	["A","B","C"]
["A","B","C"]	dequeue()	["B","C"]
["A","B"]	queue("D")	["A","B","D"]
["A","B","D"]	dequeue()	["B","D"]

Dalam implementasi queue menggunakan array ini disimpan variabel **rear** yang menunjukkan index atau nomor urut terakhir item dalam queue. Operasi **enqueue** menambahkan item ke dalam array pada index/urutan **rear+1** atau **setelah item terakhir** dalam array. Operasi **dequeue** akan mengambil item dari dalam array pada index 0 atau **item pertama/paling depan** dalam array. Kemudian dilakukan pergeseran isi array ke kiri sebanyak satu dan mengurangi rear sebanyak satu(rear-1).

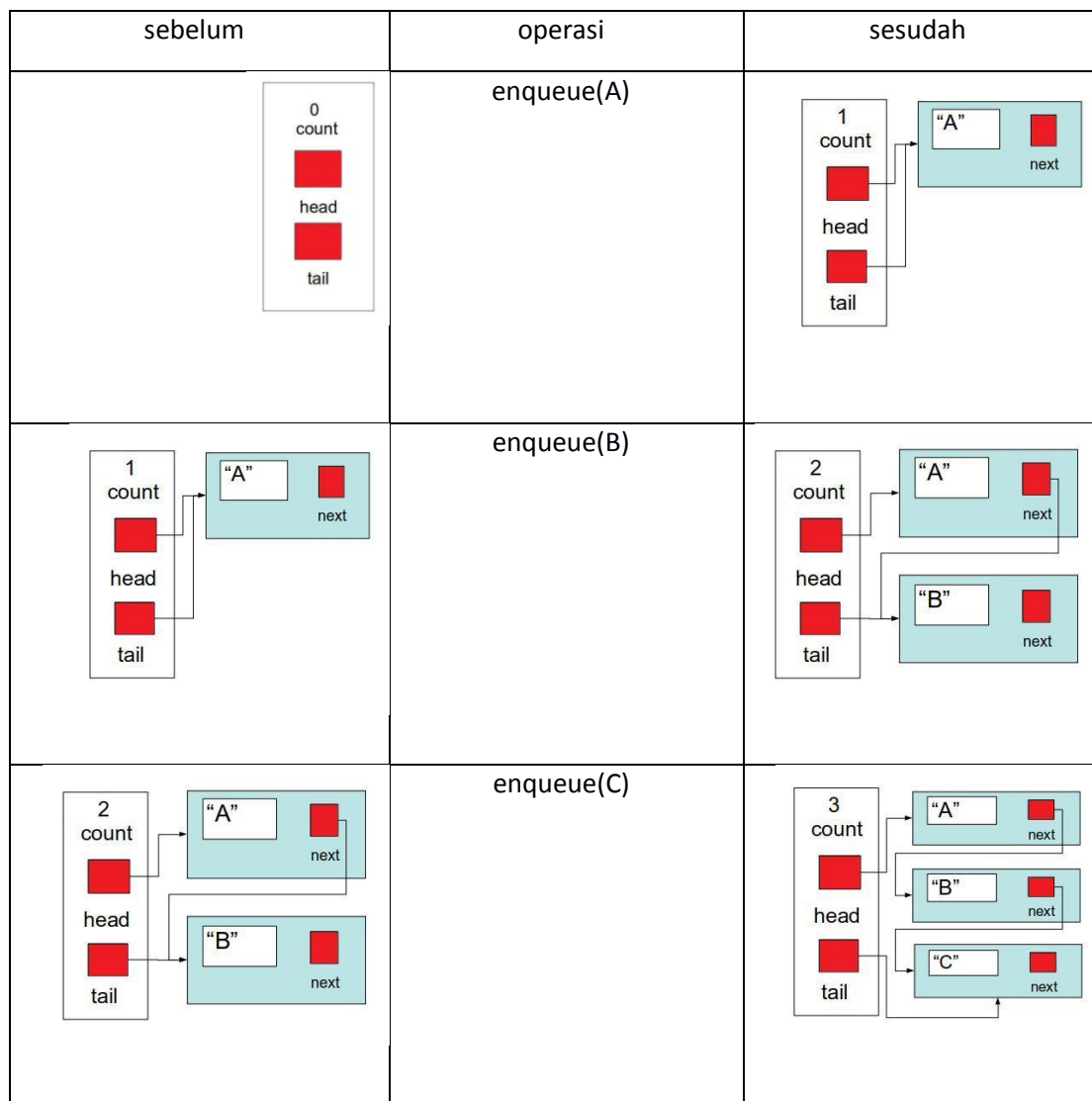
Pada tabel dibawah diilustrasikan operasi queue menggunakan array. Perbedaan dengan queue diatas adalah implementasi proses **enqueue** akan menggeser item dalam array kekanan sebanyak satu dan menambahkan/menyisipkan item pada **array index ke 0**. Proses **dequeue** akan mengambil item pada **array index terakhir atau rear**.

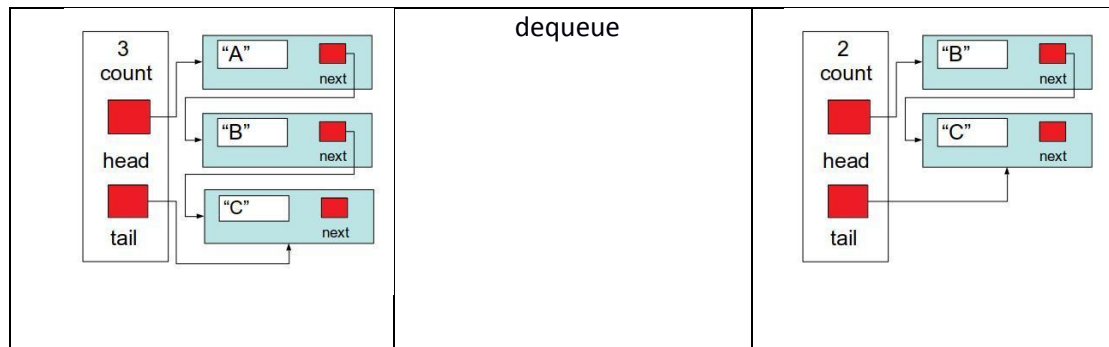
sebelum	operasi	sesudah
[]	queue("A")	["A"]
["A"]	queue("B")	["B","A"]
["B","A"]	queue("C")	["C","B","A"]
["C","B","A"]	dequeue()	["C","B"]



["C","B"]	queue("D")	["D","C","B"]
["D","C","B"]	dequeue()	["D","C"]

Gambar dibawah memberikan ilustrasi queue menggunakan single linked list. Head merupakan pointer yang menunjuk pada item paling atas yang merupakan item yang pertama dimasukkan dalam queue, item paling depan dalam antrian. Tail merupakan pointer yang selalu menunjuk pada item yang terakhir ditambahkan dalam queue, item paling belakang dalam antrian. Dalam ilustrasi ini operasi **enqueue** pada queue menggunakan operasi **add last** pada single linked list dan operasi **dequeue** menggunakan operasi **remove first**. Jika operasi **enqueue** menggunakan operasi **add first** maka operasi **dequeue** akan menggunakan **remove last**.





## 8.5 Prosedur Praktikum

1. Buatlah file percobaan queue\_array.java berikut

```

1  import java.util.Scanner;
2
3  Public class queue_array {
4  Scanner masuk = new Scanner(System.in);
5
6  int choice, i;
7  char item;
8  char arr_stack[MAX_SIZE];
9  int count = 0;
10 int keluar = 0;
11 int front,rear=0;
12
13 Public void enqueue(char item)
14 {
15     if (rear == MAX_SIZE)
16     {
17         system.out.print ( "\n# Queue Penuh");
18     }
19     else
20     {
21         system.out.print ( "\n# Queue No urut/index : "+ rear+ ", Queue :"+ item);
22         .....
23     }
24 }
25 Public void dequeue()
26 {
27     if (rear == 0)
28     system.out.print ( "\n## Queue kosong");
29     else
30     {
31         .....
32         .....
33         .....
34         .....
35         .....
36         .....

```

```

37     .....
38     }
39 }
40
41 Public void printAll()
42 {
43     system.out.print ( "\n## Queue Size : " + rear);
44     for (i = 0; i < rear; i++)
45         system.out.print ( "\n## No Urut/index : " + i + ", Value : " + arr_stack[i]);
46 }
47
48 Public void menu()
49 {
50     system.out.print ( "\nMasukkan operasi yang akan dilakukan (1:enqueue, 2:dequeue,
51     3:print) : ");
52     choice= masuk.nextInt();
53     switch (choice)
54     {
55     case 1:
56     {
57         system.out.print ( "\nMasukkan huruf yang akan di-enqueue : ");
58         item= masuk.nextLine();
59         enqueue(item);
60     }
61     break;
62     case 2:
63     {
64         dequeue();
65     }
66     break;
67     case 3:
68     {
69         printAll();
70     }
71     break;
72     default:
73     {
74         system.out.print ( "\n1:enqueue, 2:dequeue, 3:print\n");
75         keluar = 1;
76     }
77     break;
78 }
79 }
80
81 Public static void main()
82 {
83     do
84     {
85         menu();
86     } while (keluar == 0);
87 }

```

2. Tambahkan kode berikut pada baris 22

```
22 arr_stack[rear++] = item;
```

3. Tambahkan kode berikut pada baris 32 sampai 38

```

32     System.out.print ( "\n##Dequeue Value :" +
33     arr_stack[0]);
34     for(i=1;i<=rear;i++)
35     {
36         char temp=arr_stack[i];
37         arr_stack[i-1]=temp;
38     }
    rear--;

```

#### 4. Jalankan program kemudian

- pilih menu enqueue dan masukkan “A”
- pilih menu enqueue dan masukkan “B”
- pilih menu enqueue dan masukkan “C”
- pilih menu print
- pilih menu dequeue
- pilih menu print
- pilih menu enqueue dan masukkan “D”
- pilih menu print

#### 8.6 Hasil Percobaan

Tuliskan hasil dari percobaan diatas.

#### 8.7 Analisis Hasil

Tuliskan Analisis hasil dari percobaan diatas.

#### 8.8 Kesimpulan

Tuliskan kesimpulan dari percobaan diatas.

#### 8.9 Latihan

Buatlah program queue menggunakan array dengan ketentuan operasi enqueue akan selalu menambahkan item kedalam array pada index ke 0.

#### 8.10 Tugas

Buatlah program queue menggunakan double linked list dengan ketentuan operasi enqueue akan menambahkan item pada node head(add first) dan operasi dequeue akan mengambil item dari node pada tail(remove last).

#### 8.11 DAFTAR PUSTAKA

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, “Data Structures and Algorithms Using Java 6 edition”, Wiley, USA, 2014.
- John R. Hubbard, “Scaum’s Outline of Data Structures With Java second Edition”, McGraw-Hill, New york, 2007.
- Robert Lafore, “Data Structures and Algorithm in Java second Edition”, Sams Publishing, Indiana, 2003

## Modul 9 : ADT Binary Tree

### 9.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan tugas / Studi Kasus
- 50 menit **Pengayaan**

### 9.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengetahuidanmemahamikonsep tree, binary tree dan tree transversal
2. Menerapkankonsep tree, binary tree dan tree transversaldenganbahasapemrograman

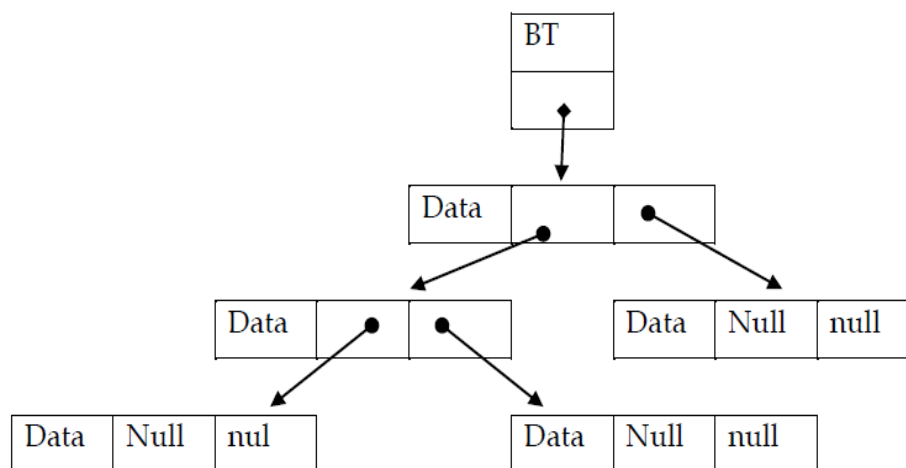
### 9.3 Alat & Bahan

1. Komputer
2. Java IDE

### 9.4 Dasar Teori

Binary Tree adalah non linear linked list dimana masing-masing nodenya menunjuk paling banyak 2 node yang lain. Elemen pada tree disebut node, dimana sebuah node hanya bias mempunyai satu parent dan bis apunya child paling banyak 2. Root adalah node yang memiliki hirarki tertinggi dan dibentuk pertama kali sehingga tidak memiliki parent. Ada 4 macam cara penelusuran node pada binary tree yaitu preorder, postorder, inorder dan level order.

Gambaran pohon biner ini seperti gambar berikut :



### ADT Binary Tree

Dari ilustrasi gambar di atas ADT antrian dapat direpresentasikan sebaga iberikut

Node
int data; Node nodeKiri; Node nodeKanan;
Node(int) sisipDt(int)

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

## 9.5 ProsedurPraktikum

- a. Buatlah program sesuai dengan source code Program Latihan Praktikum 9.1
- b. Jalankan program

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

```

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

## 9.6 Hasil Percobaan

Tulislah hasil dari Program Latihan Praktikum 9.1

## 9.7 Analisis Hasil

1. Jelaskan apa perbedaan dari 3 macam penelusuran node yang ada di Program Latihan Praktikum 9.1 yaitu pre order, inorder, dan postorder !

.....

.....

.....

2. Apa tipe data yang dapat disimpan di node Program Latihan Praktikum 9.1? Jika ingin menyimpan data yang berupa String bagian kode program manakah yang harus diganti?

.....

.....

.....

## 9.8 Kesimpulan

## 9.9 Latihan

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.



### **9.10 Tugas**

1. Buatlah program Complete Binary Tree dengan menggunakan array.

### **9.11 DAFTAR PUSTAKA**

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms Using Java 6 edition", Wiley, USA, 2014.
- John R. Hubbard, "Scaum's Outline of Data Structures With Java second Edition", McGraw-Hill, New york, 2007.
- Robert Lafore, "Data Structures and Algorithm in Java second Edition", Sams Publishing, Indiana, 2003

## Modul 10 : ADT AVL Tree

### 10.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan tugas / Studi Kasus
- 50 menit **Pengayaan**

### 10.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengetahui dan memahami konsep AVL Tree
2. Menerapkan konsep AVL Tree dengan menggunakan bahasa pemrograman
3. Menganalisis penerapan konsep AVL Tree

### 10.3 Alat & Bahan

1. Komputer
2. Java IDE

### 10.4 Dasar 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

Node
int data
int tinggi
Node pKiri
Node pKanan
Node pInduk
Node(int, int, Node, Node, Node)

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)

### 10.5 Prosedur Praktikum

	Program Latihan Praktikum 10.1
1	<code>class Node{</code>
2	<code>    int data;</code>
3	<code>    int tinggi; //tinggi node</code>
4	<code>    Node pKiri;</code>
5	<code>    Node pKanan;</code>
6	<code>    Node pInduk; // pointer ke induk</code>
7	<code></code>
8	<code>    //constructor node</code>
9	<code>    public Node(int dt, int tg, Node pKi, Node pKa, Node pI){</code>
10	<code>        this.data = dt;</code>
11	<code>        this.tinggi = tg;</code>
12	<code>        this.pKiri = pKi;</code>

```

13         this.pKanan = pKa;
14         this.pInduk = pI;
15     }
16 }
17 public class AVLTree {
18     private Node root;
19
20     public AVLTree(){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
70             else prev.pKanan = temp; //sisipDt at
71
72             //mulai dari node yang disisipkan dan
73             //bergerak menuju root
74             while(temp != null){
75                 //subtree pKiri dan pKanan memenuhi
76                 kondisi AVL
77                 if(Math.abs(tinggi(temp.pKiri)-
78                     tinggi(temp.pKanan)) <=1){
79
80                     temp.tinggi =

```

```

81 Math.max(tinggi(temp.pKiri),
82
83 tinggi(temp.pKanan)) +1;
84     }
85     //kasus 1 algoritma AVL
86     else if(tinggi(temp.pKiri)-
87 tinggi(temp.pKanan)>= 2 &&
88         tinggi(temp.pKiri.pKiri)
89 >=
90     tinggi(temp.pKiri.pKanan))
91     {
92         Node parent = temp.pInduk;
93         Node pKiri = temp.pKiri;
94
95         temp.pKiri = pKiri.pKanan;
96         if(temp.pKiri != null)
97 temp.pKiri.pInduk = temp;
98
99         pKiri.pKanan = temp;
100        temp.pInduk = pKiri;
101
102        pKiri.pInduk = parent;
103        if(parent == null) root = pKiri;
104        else
105        if (parent.pKiri==temp)parent.pKiri = pKiri;
106        else parent.pKanan = pKiri;
107
108        //hitung tinggi subtree pKanan
109        temp.tinggi =
110 Math.max(tinggi(temp.pKiri),
111
112 tinggi(temp.pKanan)) +1;
113
114        temp = pKiri;
115
116        //hitung tinggi dari root
117        temp.tinggi =
118 Math.max(tinggi(temp.pKiri),
119
120 tinggi(temp.pKanan)) +1;
121    }
122    //case 2 algoritma AVL
123    else if(tinggi(temp.pKanan)-
124 tinggi(temp.pKiri)>= 2 &&
125
126 tinggi(temp.pKanan.pKanan) >=
127 tinggi(temp.pKanan.pKiri))
128    {
129        Node parent = temp.pInduk;
130        Node pKanan = temp.pKanan;
131
132        temp.pKanan = pKanan.pKiri;
133        if(temp.pKanan != null)
134        temp.pKanan.pInduk = temp;
135
136        pKanan.pKiri = temp;
137        temp.pInduk = pKanan;
138
139        pKanan.pInduk = parent;
140        if(parent == null) root = pKanan;
141        else if(parent.pKanan == temp)
142        parent.pKanan = pKanan;
143        else parent.pKiri = pKanan;
144
145        //hitung tinggi subtree pKanan
146        temp.tinggi =
147 Math.max(tinggi(temp.pKiri),
148

```

```

149 tinggi(temp.pKanan)) +1;
150
151 temp = pKanan;
152
153 //hitung tinggi dari root
154 temp.tinggi =
155 Math.max(tinggi(temp.pKiri),
156
157 tinggi(temp.pKanan)) +1;
158 }
159 //kasus 3 dari algoritma AVL
160 else if(tinggi(temp.pKiri)-
161 tinggi(temp.pKanan)>= 2 &&
162 tinggi(temp.pKiri.pKanan)
163 >=
164 tinggi(temp.pKiri.pKiri))
165 {
166 Node parent = temp.pInduk;
167 Node pKiripKanan =
168 temp.pKiri.pKanan;
169 temp.pKiri.pKanan =
170 pKiripKanan.pKiri;
171 if(temp.pKiri.pKanan!= null)
172 temp.pKiri.pKanan.pInduk =
173 temp.pKiri;
174
175 pKiripKanan.pKiri = temp.pKiri;
176 temp.pKiri.pInduk = pKiripKanan;
177
178 temp.pKiri = pKiripKanan.pKanan;
179 if(temp.pKiri != null)
180 temp.pKiri.pInduk = temp;
181
182 pKiripKanan.pKanan = temp;
183 temp.pInduk = pKiripKanan;
184
185 pKiripKanan.pInduk = parent;
186 if(parent == null) root =
187 pKiripKanan;
188 else if(parent.pKiri==temp)
189 parent.pKiri = pKiripKanan;
190 else parent.pKanan = pKiripKanan;
191
192 //hitung tinggi subtree pKanan
193 temp.tinggi =
194 Math.max(tinggi(temp.pKiri),
195
196 tinggi(temp.pKanan)) +1;
197
198 temp = pKiripKanan;
199
200 //hitung tinggi dari root
201 temp.tinggi =
202 Math.max(tinggi(temp.pKiri),
203
204 tinggi(temp.pKanan)) +1;
205 }
206 //kasus 4 dari algoritma AVL
207 else if(tinggi(temp.pKanan)-
208 tinggi(temp.pKiri)>= 2 &&
209 tinggi(temp.pKanan.pKiri)
210 >=
211 tinggi(temp.pKanan.pKanan))
212 {
213 Node parent = temp.pInduk;
214 Node pKananpKiri =
215 temp.pKanan.pKiri;
216

```

```

217 temp.pKanan.pKiri =
218 pKananpKiri.pKanan;
219 if(temp.pKanan.pKiri!= null)
220     temp.pKanan.pKiri.pInduk =
221     temp.pKanan;
222
223 pKananpKiri.pKanan = temp.pKanan;
224 temp.pKanan.pInduk = pKananpKiri;
225
226 temp.pKanan.pInduk = temp;
227
228 pKananpKiri.pKiri = temp;
229 temp.pInduk = pKananpKiri;
230
231 pKananpKiri.pInduk = parent;
232 if(parent == null) root =
233 pKananpKiri;
234 else if(parent.pKanan == temp)
235     parent.pKanan = pKananpKiri;
236 else parent.pKiri = pKananpKiri;
237
238 temp.tinggi =
239 Math.max(tinggi(temp.pKiri),
240 tinggi(temp.pKanan)) +1;
241
242
243 temp = pKananpKiri;
244
245 temp.tinggi =
246 Math.max(tinggi(temp.pKiri),
247 tinggi(temp.pKanan)) +1;
248
249     }
250     temp = temp.pInduk;
251     }
252     //penyisipan berhasil
253     return true;
254 }
255
256 public int tinggi(){return root.tinggi;}
257 private int tinggi(Node node){
258     if(node == null)return 0;
259     else return node.tinggi;
260 }
261 //hitung node-node dari tree
262 public int jumlahNode() {
263     return jumlahNode(root);
264 }
265
266 public void inOrderTraversal(){
267     inOrder(root);
268 }
269
270 private void inOrder(Node r){
271     if (r == null)return;
272     inOrder(r.pKiri);
273     System.out.printf("-%d",r.data);
274     inOrder(r.pKanan);
275 }
276
277 //hitung node-node dari tree
278 private int jumlahNode(Node node){
279     if(node == null)    return 0;
280     else
281         return 1 +jumlahNode(node.pKiri)
282         +jumlahNode(node.pKanan);

```

283	}
284	
285	public static void main (String[] args) {
286	AVLT t = new AVLT();
287	t.sisipDt(3); t.inOrderTraversal();System.out.println();
288	t.sisipDt(4); t.inOrderTraversal();System.out.println();
289	t.sisipDt(6); t.inOrderTraversal();System.out.println();
290	t.sisipDt(5); t.inOrderTraversal();System.out.println();
291	t.sisipDt(15); t.inOrderTraversal();System.out.println();
292	t.sisipDt(10); t.inOrderTraversal();System.out.println();
293	t.sisipDt(20); t.inOrderTraversal();System.out.println();
294	t.sisipDt(17); t.inOrderTraversal();System.out.println();
295	t.sisipDt(25);t.inOrderTraversal();System.out.println();
296	}
297	}

## 10.6 Hasil Percobaan

Tulislah hasil dari Program Latihan Praktikum 10.1

## 10.7 Analisis Hasil

1. Jelaskan apa perbedaan dari 4 kasus yang ada pada Program Latihan Praktikum 10.1

2. Dalam proses penyisipan data kedalam pohon AVL jika kondisi pohon tidak seimbang maka harus dilakukan rotasi kiri, rotasi kanan, rotasi kiri kanan, atau rotasi kanan dan kiri. Dari program di atas pada baris berapakah dilakukan masing-masing proses ini?

## 10.8 Kesimpulan

## 10.9 Latihan

1. Tambahkan method putar Kiri pada Program Latihan Praktikum 10.1 untuk menyeimbangkan AVL Tree
2. Tambahkan method putar Kanan pada Program Latihan Praktikum 10.1 untuk menyeimbangkan AVL Tree
3. Tambahkan method putar Kiri Kanan pada Program Latihan Praktikum 10.1 untuk menyeimbangkan AVL Tree
4. Tambahkan method putar Kanan Kiri pada Program Latihan Praktikum 10.1 untuk menyeimbangkan AVL Tree

### 10.10 Tugas

Buatlah Program Latihan Praktikum 10.1 dengan menambahkan method untuk menghapus suatu node pada pohon AVL dengan 3 kondisi yaitu jika node yang dihapus adalah daun, jika node yang dihapus mempunyai satu anak dan jika node yang dihapus mempunyai 2 anak

### 10.11 DAFTAR PUSTAKA

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, "Data Structures and Algorithms Using Java 6 edition", Wiley, USA, 2014.
- John R. Hubbard, "Scaum's Outline of Data Structures With Java second Edition", McGraw-Hill, New york, 2007.
- Robert Lafore, "Data Structures and Algorithm in Java second Edition", Sams Publishing, Indiana, 2003



## Modul 11 : ADT Graf

### 11.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut.

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 45 menit untuk penyampaian materi
- 55 menit untuk pengerjaan tugas / Studi Kasus
- 55 menit **Pengayaan**

### 11.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

- Memahami representasi graf dalam adjacency matrix dan adjacency list
- Mengimplementasikan graf dalam bentuk adjacency list

### 11.3 Alat & Bahan

- Komputer
- Java IDE

### 11.4 Dasar Teori

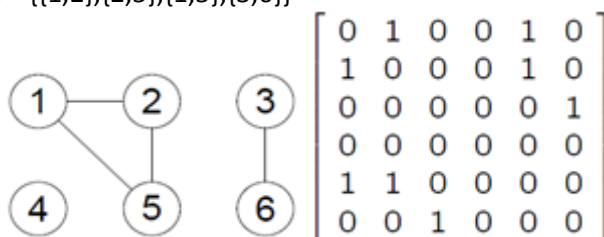
Graf merupakan struktur data yang memiliki relasi many to many, yaitu setiap elemen dapat memiliki nol atau lebih dari satu cabang. Suatu graf terdiri dari dua bagian, yaitu node (verteks) dan edge. Node digunakan untuk menyimpan data, sedangkan edge berguna untuk menghubungkan node satu dengan node lainnya. Suatu graf dapat ditulis dalam bentuk  $G = (V, E)$ , yang mana  $V$  adalah suatu verteks berupa himpunan tidak kosong dari  $G$ , dan  $E \subseteq V \times V$ . Terdapat dua jenis graf, yaitu graf berarah dan graf tak berarah.

Suatu graf dapat direpresentasikan dalam bentuk adjacency matrix dan adjacency list. Adjacency matrix direpresentasikan dengan matriks dua dimensi, dan adjacency list direpresentasikan dengan array maupun linked list. Berikut masing-masing representasi graf.

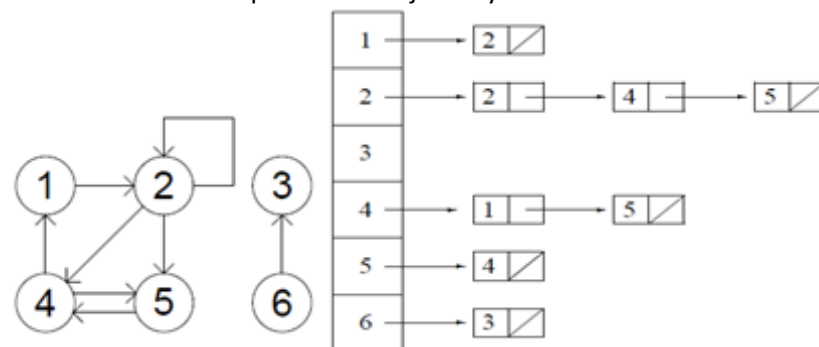
Graf tak berarah dalam representasi adjacency matrix

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

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



Graf berarah dalam representasi adjacency list



Adapun ADT dari graf adalah sebagai berikut.

Node
Object data;

Node next;
Node(int,Node)
Object getDt()
Node getNext()

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

### 11.5 Prosedur Praktikum

1. Buatlah projek di IDE Java dengan nama Graph
2. Buatlah Class Node di projek yang telah dibuat, dan implementasikan masing-masing method sesuai dengan ADT graf.

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	}
21	
22	public void addAdj(int head, int adj){
23	Node n = new Node(adj,node[head]);
24	node[head] = n;
25	}
26	
27	public void cetak(String komentar){
28	System.out.println(komentar);
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	
40	public static void main(String args){
41	Graph g = new Graph(5);
42	g.addAdj(0,3);
43	g.addAdj(0,1);
44	g.addAdj(1,4);
45	g.addAdj(1,2);
46	g.addAdj(2,4);
47	g.addAdj(2,1);

48	<code>g.addAdj(4,3);</code>
49	<code>g.cetak("Kondisi awal");</code>
50	<code>}</code>
51	<code>}</code>

3. Lakukan kompilasi dan tuliskan hasil percobaan, analisis hasil, dan kesimpulannya.

#### 11.6 Hasil Percobaan

Tuliskan hasil dari percobaan di atas!

#### 11.7 Analisis Hasil

Tuliskan analisis hasil dari percobaan di atas!

#### 11.8 Kesimpulan

Tuliskan kesimpulan dari percobaan di atas!

#### 11.9 Latihan

1. Tambahkan method untuk mengetahui berapa jumlah edge yang masuk atau keluar dari suatu node!
2. Tambahkan method untuk mengetahui jumlah tetangga dari suatu node!

#### 11.10 Tugas

1. Buatlah ADT untuk merepresentasikan graf dengan adjacency matrix!
2. Implementasikan penelusuran node dari graf menggunakan BFS dan DFS! Gunakan struktur data yang pernah dibuat pada Stack atau Queue!
3. Lakukan modifikasi pada ADT graf di atas sehingga suatu hubungan antar node terdapat nilai jarak!
4. Berdasarkan hasil modifikasi nomor 3, susunlah metode untuk menghitung jarak terpendek dari suatu node asal ke node tujuan!

.....

#### **11.11 DAFTAR PUSTAKA**

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, “Data Structures and Algorithms Using Java 6 edition”, Wiley, USA, 2014.
- John R. Hubbard, “Scaum’s Outline of Data Structures With Java second Edition”, McGraw-Hill, New york, 2007.
- Robert Lafore, “Data Structures and Algorithm in Java second Edition”, Sams Publishing, Indiana, 2003

## Modul 12 : Sorting

### 12.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum = **170 menit**, dengan rincian sebagai berikut (misalkan):

- 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
- 60 menit untuk penyampaian materi
- 45 menit untuk pengerjaan tugas / Studi Kasus
- 50 menit **Pengayaan**

### 12.2 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami konsep dasar sorting
2. Mengimplementasikan Bubble dan Shell Sort
3. Mengimplementasikan Selection dan Insertion Sort
4. Memahami perbedaan masing-masing metode sorting

### 12.3 Alat & Bahan

1. Komputer
2. Java IDE

### 12.4 Dasar Teori

#### Sorting

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 :

- a) urut naik (ascending) yaitu dari data yang mempunyai nilai paling kecil sampai paling besar
- b) 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 :

- a) data mudah dicari (misalnya dalam buku telepon atau kamus bahasa), mudah untuk dibetulkan, dihapus, disisipi atau digabungkan.
- b) Selain itu dalam keadaan terurutkan, kita mudah melakukan pengecekan apakah ada data yang hilang atau tidak.
- c) melakukan kompilasi program komputer jika tabel-tabel simbol harus dibentuk
- d) 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:

- a) banyak data yang diurutkan
- b) kapasitas mengingat apakah mampu menyimpan semua data yang kita miliki
- c) tempat penyimpanan data, misalnya disket, flashdisk, harddisk atau media penyimpanan yang lain

#### Bubble Sort

Bubble sort (metode gelembung) adalah metode/algorithm pengurutan dengan cara melakukan penukaran data dengan tepat disebelahnya secara terus menerus sampai bisa dipastikan dalam satu iterasi tertentu tidak ada lagi perubahan. Jika tidak ada perubahan berarti data sudah terurut. Disebut pengurutan gelembung karena masing-masing kunci akan dengan lambat menggelembung ke posisinya yang tepat.

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:

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

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 :

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

Langkah 3:

K=N=6					21	76
K=5				21	27	76
K=4			21	25	27	76

Hasil akhir langkah 3 :

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

Langkah4:

K=N=6	27	76			
K=5	25	27	76		
Hasil akhir langkah 4:					
8	10	21	25	27	76
1	2	3	4	5	6

Langkah 5:

K=N=6
27
76

Hasil akhir langkah 5:

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

Sorting sudah selesai, data sudah terurutkan.

### Shell Sort

Metode shell sort dikembangkan oleh Donald L. Shell pada tahun 1959. Dalam metode ini jarak antara dua elemen yang dibandingkan dan ditukarkan tertentu. Secara singkat metode ini dijelaskan sebagai berikut. Pada langkah pertama, kita ambil elemen pertama dan kita bandingkan dan kita bandingkan dengan elemen pada jarak tertentu dari elemen pertama tersebut. Kemudian elemen kedua kita bandingkan dengan elemen lain dengan jarak yang sama seperti jarak yang sama seperti diatas. Demikian seterusnya sampai seluruh elemen dibandingkan. Pada langkah kedua proses diulang dengan langkah yang lebih kecil, pada langkah ketiga jarak tersebut diperkecil lagi seluruh proses dihentikan jika jarak sudah sama dengan satu.

Proses pada Shell sort sebelum memulai sorting adalah kita lihat jumlah data, misalkan pada tabel dibawah data awal adalah pada baris langkah ke-1. Kemudian kita berikan jarak untuk pertukaran data. Biasanya untuk proses awal, jarak untuk pertukaran data biasanya separuh dari total jumlah data. Misalkan pada contoh adalah 6, sehingga kita beri jarak 3 untuk dilakukan perbandingan dan pertukaran data.

Proses ke-	Jarak	0	1	2	3	4	5
1	Awal	25	40	5	8	33	13
2	Jarak = 3	<u>25</u>	40	5	<u>8</u>	33	13
3		8	<u>40</u>	5	25	<u>33</u>	13
4		8	33	<u>5</u>	25	40	<u>13</u>
5	Jarak = 1	<u>8</u>	<u>33</u>	5	25	40	13
6		8	<u>33</u>	<u>5</u>	25	40	13
7		8	5	<u>33</u>	<u>25</u>	40	13
9		8	5	25	<u>33</u>	<u>40</u>	13
10		8	5	25	33	<u>40</u>	<u>13</u>
11		<u>8</u>	<u>5</u>	25	33	13	40
12		5	<u>8</u>	<u>25</u>	33	13	40
13		5	8	<u>25</u>	<u>33</u>	13	40
14		5	8	25	<u>33</u>	<u>13</u>	40
15		5	8	25	13	<u>33</u>	<u>40</u>
16		<u>5</u>	<u>8</u>	25	13	33	40
17		5	<u>8</u>	<u>25</u>	13	33	40
18		5	8	<u>25</u>	<u>13</u>	33	40

19		5	8	13	<u>25</u>	<u>33</u>	40
20		<u>5</u>	<u>8</u>	<u>13</u>	<u>25</u>	<u>33</u>	<u>40</u>

#### Keterangan

Proses ke-1: Data dalam bentuk array awal yang dimiliki, data berjumlah 6. Kemudian kita tentukan dengan jarak 3.

Proses ke-2: Dengan jarak 3, kita bandingkan data pada indeks ke 0 dengan indeks ke 3. Data pada indeks ke-0 bernilai 25 dibandingkan dengan indeks ke-3 bernilai 8. Karena data pada indeks ke-3 bernilai lebih kecil, maka dilakukan pertukaran data. Sehingga data yang bernilai 8 ditaruh pada indeks ke-0 dan sebaliknya. Setelah dilakukan pertukaran, akan maju 1 indeks hingga selesai.

Proses ke-3: Dengan cara yang sama dilakukan seperti proses ke-2, antara indeks ke-1 dan indeks ke-4, indeks ke-1 bernilai 40 dan indeks ke-4 bernilai 33 kemudian dilakukan pertukaran.

Proses ke-4: Indeks ke-2 dibandingkan dengan indeks ke-5, karena data sudah sesuai, sehingga tidak dilakukan pertukaran.

Proses ke-5 sampai dengan proses ke-19 : proses pembandingan data dan pertukaran data mirip dengan proses ke-2 hingga proses ke-4, namun perbedaannya adalah nilai jarak. Nilai jarak biasanya akan dilakukan setting decrement 1 dari indeks awal, atau dapat ditentukan untuk proses selanjutnya. Pada contoh tersebut dilakukan pemberian jarak sebesar 1 untuk proses ini. Proses ini akan terus berjalan dan panjang proses tidak menentu hingga tidak ada lagi data yang akan ditukarkan dan dibandingkan hingga tidak ada data yang dapat dibandingkan (lebih kecil atau lebih besar)

Proses ke-20 akan dilakukan lagi pembandingan terakhir pada masing-masing data untuk memastikan tidak ada data yang lebih kecil dari indeks sebelumnya.

#### Selection Sort

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 selection sort secara ascending dapat dijelaskan sebagai berikut:

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

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

##### Langkah 1:

Cari elemen maksimum di dalam larik  $L[1..6] \rightarrow \text{maks} = L[5] = 76$

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  $L[1..5] \rightarrow \text{maks} = L[1] = 29$

Tukar maks dengan  $L[5]$ , hasil akhir langkah 2:

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

##### Langkah 3:

(berdasarkan susunan larik hasil langkah 2)

Cari elemen maksimum di dalam larik  $L[1..4] \rightarrow \text{maks} = L[2] = 27$



Tukar maks dengan L[4], hasil akhir langkah 3:

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

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:

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

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:

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

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

### Insertion Sort

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.

## 12.5 ProsedurPraktikum

Jalankan masing-masing program dibawah ini

### Bubble Sort

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	}

## Shell Sort

1	public class Shell {
2	public static <T extends Comparable<? super T>> void shellSort(T[] arr){
3	int i, jarak;
4	Boolean did_swap=true;
5	T temp;
6	jarak=arr.length;
7	while(jarak>1){
8	jarak=jarak/2;
9	did_swap=true;
10	while(did_swap){
11	did_swap=false;
12	i=0;
13	while(i<(arr.length-jarak)){
14	if(arr[i].compareTo(arr[i+jarak])>0){
15	temp=arr[i];
16	arr[i]=arr[i+jarak];
17	arr[i+jarak]=temp;
18	did_swap=true;
19	}
20	i++;
21	}
22	}
23	}
24	}
25	public static <T> void tampil(T data[]) {
26	for (T objek : data) {
27	System.out.print(objek + " ");
28	}
29	System.out.println("");
30	}
31	public static void main(String[] args) {
32	Integer data[] = new Integer[10];
33	for(int a=0 ; a<data.length ; a++){
34	data[a]= (int)(Math.random()*13+1);
35	}
36	long awal = System.currentTimeMillis();
37	shellSort(data);
38	long sisaWaktu = System.currentTimeMillis() - awal;
39	tampil(data);
40	System.out.println("Waktu " + sisaWaktu);
41	}
42	}

## Selection Sort

1	public class SelectionSorter {
2	
3	int[] L = {25, 27, 10, 8, 76, 21};
4	void selectionSort() {
5	int j, k, i, temp;
6	int jmax, u = 5;
7	for (j = 0; j < 6; j++) {
8	jmax = 0;
9	System.out.println("Langkah " + (j + 1) + ":");
10	for (k = 1; k <= u; k++) {
11	if (L[k] > L[jmax]) {
12	jmax = k;
13	}
14	}
15	temp = L[u];
16	L[u] = L[jmax];
17	L[jmax] = temp;
18	u--;
19	//melihat hasil tiap langkah
20	for (i = 0; i <= 5; i++) {
21	System.out.println(L[i] + " index:" + (i + 1));
22	}
23	}
24	
25	System.out.println("Hasil akhir:");
26	for (i = 0; i <= 5; i++) {
27	System.out.println(L[i] + " index:" + (i + 1));
28	}
29	}
30	
31	public static void main(String[] args) {
32	SelectionSorter sorter = new SelectionSorter();
33	sorter.selectionSort();
34	}
35	}

## Insertion Sort

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	}

## 12.6 Hasil Percobaan

1. Apakah yang anda dapatkan dari Bubble dan Shell Sort tersebut?

.....

.....

2. Apakah yang anda dapatkan dari Selection dan Insertion tersebut?

.....

.....

## 12.7 Analisis Hasil

1. Apakah perbedaan mendasar pada masing-masing algoritma sorting tersebut?

- .....
- .....
2. Jika diberikan data secara berbeda (misal 10, 20, 30 dst) proses sorting manakah yang paling efektif? Sebutkan alasan anda dengan tepat!

.....

.....

### 12.8 Kesimpulan

Berilah kesimpulan tentang masing-masing metode sorting setelah anda menjalankan program tersebut

.....

.....

### 12.9 Latihan

1. Pilihlah dua buah Sorting dari beberapa algoritma sorting tersebut dalam Bentuk Descending

.....

.....

2. Dari dua buah metode sorting yang anda pilih, tambahkan kode program untuk menampilkan perubahan setiap iterasi.

.....

.....

3. Tambahkan kode program untuk menghitung banyaknya perbandingan dan pergeseran pada dua algoritma yang anda pilih.

.....

.....

### 12.10 Tugas

1. Pilihlah dua algoritma sorting, berilah data secara acak sebanyak 50 data. Masukkan data tersebut, dan berapa banyak iterasi yang dibutuhkan untuk sorting data tersebut?

.....

.....

2. Pilih salah satu antara sorting yang ada, implementasikan pengurutan data Pegawai pada tugas pendahuluan dengan ketentuan sebaga berikut
  - a. Proses pengurutan dapat dipilih secara ascending maupun descending
  - b. Pengurutan dapat dilakukan berdasarkan NIP dan Nama  
Gunakan struktur data array

### 12.11 DAFTAR PUSTAKA

- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, “Data Structures and Algorithms Using Java 6 edition”, Wiley, USA, 2014.
- John R. Hubbard, “Scaum’s Outline of Data Structures With Java second Edition”, McGraw-Hill, New york, 2007.
- Robert Lafore, “Data Structures and Algorithm in Java second Edition”, Sams Publishing, Indiana, 2003

**TIM PENYUSUN**

1. Issa Arwani, S.Kom, M.Sc
2. Bayu Rahayudi, S.T, M.T
3. Mochammad Hannats Hanafi, S.ST, M.T
4. Indriati S.T, M.Kom
5. Lailil Muflikhah, S.Kom, M.Sc
6. Tri Afirianto, S.T, M.T