

D4 TEKNIK INFORMATIKA



**MODUL
ALGORITMA dan STRUKTUR DATA**

Oleh
Mohamad Nurkamal Fauzan, S.T., M.T.

**POLITEKNIK POS INDONESIA
2014**

Daftar Isi

BAB 1Pendahuluan.....	1
1.1Algoritma.....	3
1.1.1Simbol diagram alur.....	3
1.1.2Memberi Harga Kepada Suatu Variabel.....	5
1.1.3Mencetak Keluaran.....	5
1.1.4Percabangan.....	6
1.1.5Pengulangan.....	6
1.1.6Panji(Flag).....	7
1.1.7Jenis struktur kontrol dasar algoritma.....	8
1.1.8Karakteristik dasar algoritma.....	8
1.2Struktur data.....	8
1.2.1Tipe data pada struktur data.....	9
BAB 2Bahasa Pemrograman Java.....	12
2.1Pendahuluan.....	13
2.2 Komponen JDK.....	13
BAB 3Tipe data.....	17
3.1Identifier.....	18
3.2Reserved Words/Keywords.....	18
3.3Variabel.....	18
3.3.1Scope dari Variabel.....	19
3.4Tipe Data.....	21
3.4.1Tipe Data Primitif.....	21
3.4.2Tipe Data Referensi.....	22
BAB 4Alur Program.....	25
4.1If.....	26
4.2If-Else.....	26
4.3Switch.....	26
BAB 5Perulangan.....	29
5.1For.....	30
5.2While.....	30
5.3Do-While.....	31
BAB 6Class.....	34
6.1Class.....	35
6.2Attribute.....	35
6.3Behavior Method.....	35
6.4Object.....	36
BAB 7Fungsi.....	38
7.1Return.....	39
7.2Void.....	39
BAB 8Constructor.....	44
BAB 9Penanganan Kesalahan.....	48
9.1Error Handling.....	49
9.2Exception.....	49
9.3Try.....	49
9.3.1Bentuk 1:.....	49
9.3.2Bentuk 2 :.....	50
9.4Catch secara bertingkat.....	50
9.5Melontarkan Exception.....	51
BAB 10Array.....	53

10.1Mendeklarasikan Variabel Array.....	54
10.2Mendefinisikan Array.....	54
10.3Array Dua Dimensi.....	55
10.4Array Multidimensi.....	55
BAB 11Pencarian dan Pengurutan.....	58
11.1Pencarian.....	59
11.2Pengurutan.....	59
11.2.1Quick sort.....	60
11.2.2bubble sort.....	62
BAB 12Linked List.....	65
12.1Linked List Data Structure.....	66
12.1.1Singly linked list implementation.....	66
12.1.2Doubly linked list implementation.....	70
BAB 13Tumpukan.....	76
13.1Stacks/Tumpukan.....	77
13.1.1Operasi Stack:.....	77
BAB 14Antrian.....	81
14.1Antrian/Queue Data Structure.....	82
14.1.1Operasi pada Queue:.....	82

BAB 1 Pendahuluan

TIK:

Mahasiswa dapat memahami konsep algoritma dan struktur data

Pokok Bahasan:

- Algoritma
- Struktur data

Sub Pokok Bahasan:

- Simbol diagram alur
- Memberi Harga Kepada Suatu Variabel
- Mencetak Keluaran
- Percabangan
- Pengulangan
- Panji(Flag)
- Karakteristik dasar algoritma
- Tipe data pada struktur data

Komputer sekarang ini memegang peranan yang penting. Komputer dapat digunakan sebagai alat bantu manusia dalam memecahkan suatu masalah. Anda dapat melihat keberadaan komputer hampir di semua lini seperti pada sekolah, industri, pasar swalayan, rumah, telepon seluler dlsb.

Seperti dijelaskan sebelumnya, komputer dapat digunakan untuk memecahkan masalah. Langkah yang harus dilakukan seorang engineer dalam memecahkan masalah adalah:

- Tentukan / kenali masalah
- Merancang solusi
- Mengimplementasikan solusi ke dalam program komputer

Program komputer ditulis dalam bahasa pemrograman yang berisikan modul-modul.

Jenis modul:

- Sebuah fungsi yang berdiri sendiri
- Sebuah metode kelas
- Sebuah kelas
- Beberapa fungsi atau kelas yang bekerja sama
- Berupa Blok kode

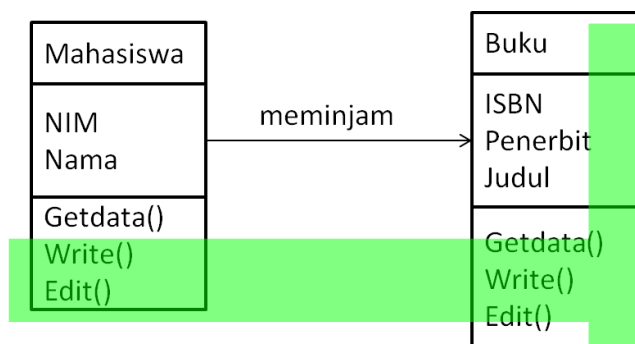
Sebuah solusi yang baik terdiri dari modul yang:

- Mengorganisir koleksi data untuk memfasilitasi operasi
- Harus dapat menyimpan, memindahkan, dan mengubah data
- Menggunakan algoritma untuk berkomunikasi antar modul

Keuntungan menggunakan modul:

- Membangun program, dapat dibuat modul-modul yang lebih kecil pada skala modul yang lebih besar
- Debug program, dapat dikonsentrasikan lebih baik
- Pembacaan kode program lebih mudah dipahami
- Modifikasi program lebih mudah dan cepat
- Dapat menghindari kode yang rangkap, dengan memanggil modul akan menghindari kode yang sama ditulis berulang kali

Contoh modul/kelas:



Di dalam modul-modulmodul tersebut terdapat algoritma

1.1 Algoritma

Asal kata Algoritma berasal dari nama Abu Ja'far Mohammed Ibn Musa al-Khowarizmi, ilmuwan Persia yang menulis kitab al jabr w'al-muqabala (rules of restoration and reduction) sekitar tahun 825 M.

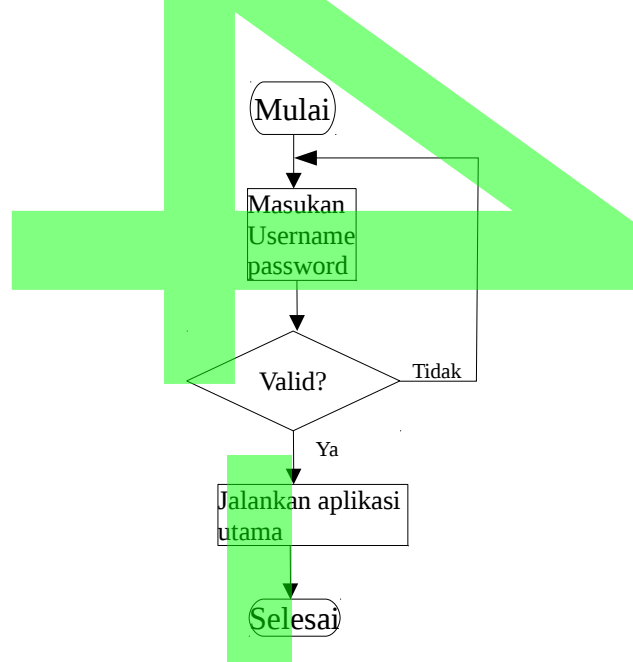
Algoritma adalah langkah-langkah kegiatan untuk memecahkan masalah berdasarkan tahapan logis dalam periode waktu tertentu.

Algoritma sering dioperasikan pada pengumpulan data, yang disimpan secara terstruktur dalam memori komputer (struktur data).

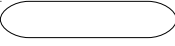
Instruksi yang terdefinisi dengan baik meliputi:

- Input, pada waktu keadaan awal
- Proses, , dilanjutkan melalui serangkaian tahapan logis yang jelas
- Output, tahapan akhir atau keluaran


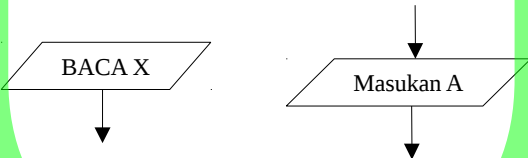


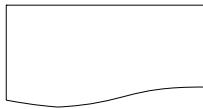

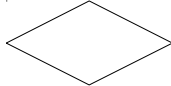
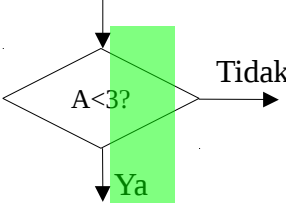


Contoh:

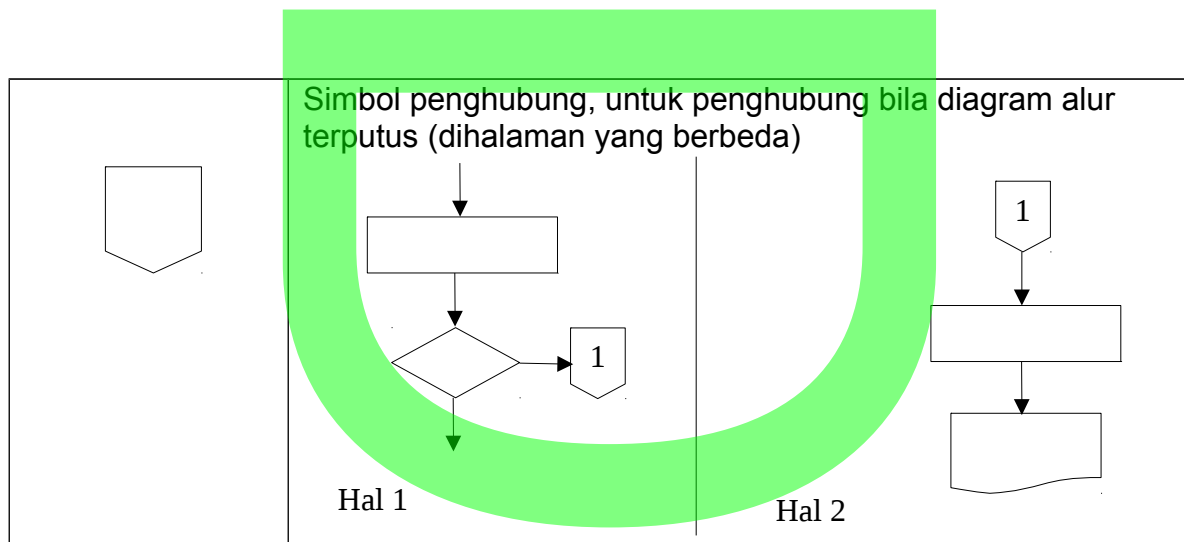


1.1.1 Simbol diagram alur

Symbol	Keterangan
	Simbol untuk menyatakan mulai (start) atau pun berhenti

The diagram shows a 'Mulai' (Start) symbol in a rounded rectangle, with an arrow pointing down to a 'Berhenti' (End) symbol, also in a rounded rectangle. This illustrates the symbols used in flowcharts to denote the beginning and end of a process.

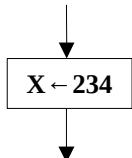
	<p>Kotak Masukan, untuk membaca data yang kemudian diberikan sebagai harga suatu variabel.</p> 
	<p>Kotak Penugasan, untuk memberi harga kepada suatu variabel atau untuk melakukan perhitungan matematika yang hasilnya diberikan sebagai harga suatu variabel</p> 
	<p>Kotak Keluaran, untuk mencetak (dan/atau menyimpan) hasil keluaran. Catatan: banyak orang yang menggunakan kotak masukan juga sebagai kotak keluaran</p> 
	<p>Kotak keputusan, untuk memutuskan arah atau percabangan yang diambil sesuai dengan kondisi benar atau salah</p> 
	<p>Simbol penghubung, untuk penghubung bila diagram alur terputus (masih dalam satu halaman)</p> 



1.1.2 Memberi Harga Kepada Suatu Variabel

Suatu variabel dapat diartikan sebagai suatu besaran yang dapat berubah-ubah harganya. Cara memberi harga kepada suatu variabel: dengan kotak penugasan.

Contoh:

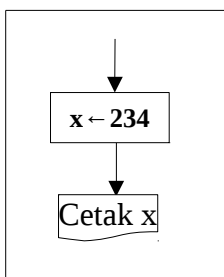


Variabel X diberi harga 234

1.1.3 Mencetak Keluaran

Untuk keperluan mencetak, kotak keluaran dapat digunakan untuk mencetak harga suatu variabel, untuk mencetak suatu konstanta bilangan serta untuk mencetak string.

Contoh:



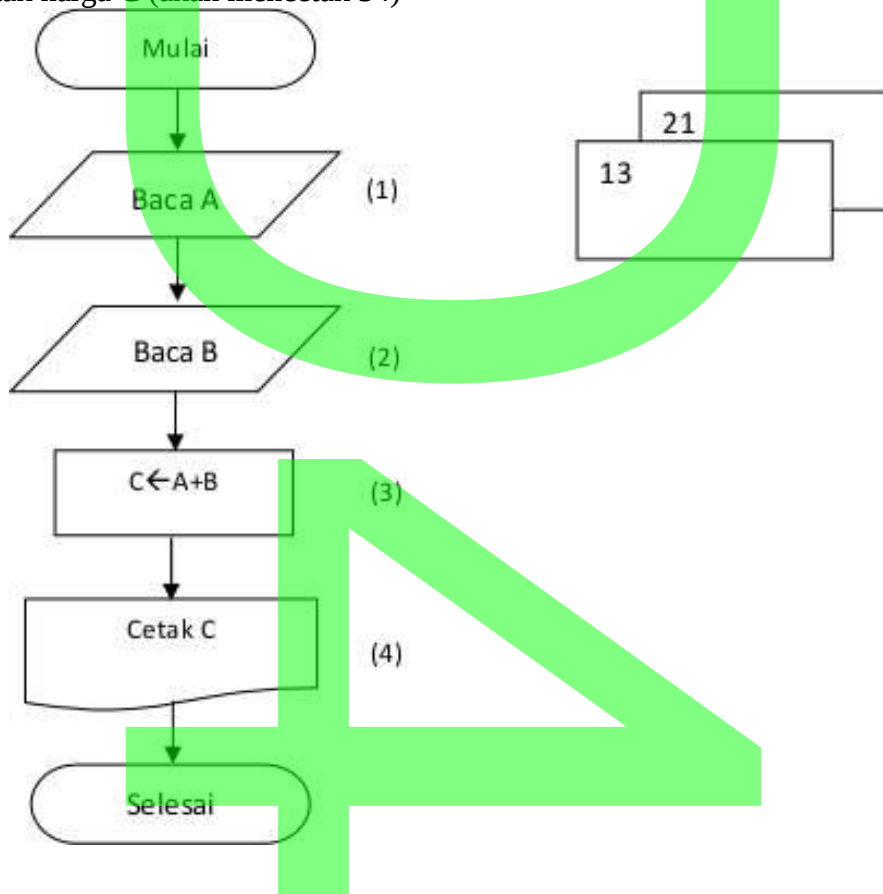
Akan mencetak harga variabel x, dalam hal ini tercetak 4

Contoh 2. Menghitung dan mencetak jumlah 2 bilangan yang dimasukan oleh user melalui keyboard, bilangan tersebut akan disimpan pada suatu variabel.

Penjelasan gambar pada contoh 2:

1. Satuan data pertama (berharga 13) dibaca dan diberikan kepada variabel A, sehingga harga variabel A menjadi =13
2. Selanjutnya satuan data kedua (berharga 21) dibaca dan diberikan kepada variabel B, sehingga harga variabel B=21
3. Menghitung/member harga variabel C sebesar harga variabel A ditambah dengan harga

- variabel B
4. Mencetak harga C (akan mencetak 34)

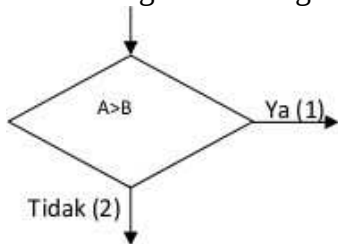


1.1.4 Percabangan

Pada contoh di atas, arus diagram alur mengalir lurus dari atas ke bawah. Diagram alur demikian biasanya untuk masalah-masalah sederhana. Masalah yang lebih rumit, pada diagram alurnya banyak terjadi alih control berupa percabangan (branching) dan pemutaran kembali (looping).

Percabangan terjadi apabila dihadapkan pada kondisi dua pilihan: BENAR dan SALAH.

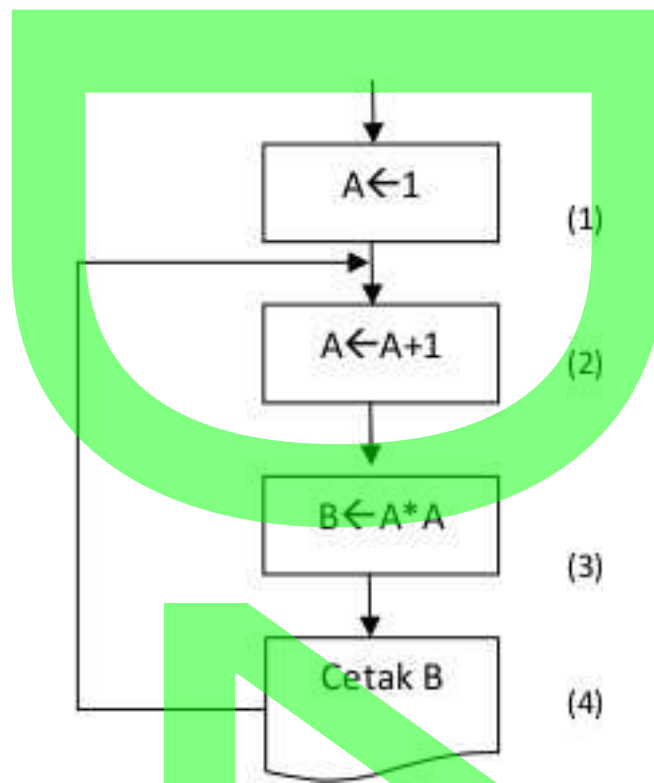
Dalam diagram alur digunakan kotak keputusan, seperti contoh gambar 1



Apabila $A > B$, kondisi benar maka mengikuti alur program(1) sedangkan bila kondisi salah mengikuti alur (2).

1.1.5 Pengulangan

Pengulangan kembali, terjadi ketika mengalihkan arus diagram alur kembali ke atas sehingga beberapa alur berulang kembali beberapa kali.



Keterangan:

(1) Variabel A diberi nilai 1

(2) Variabel A berubah nilai menjadi 2 karena melalui proses $A+1$

(3) Variabel B diberi harga sebesar harga A dikalikan harga A. Jadi variabel B berharga 4.

(4) Harga B dicetak (jadi tercetak 4). Kemudian kembali ke (2), (3), (4) dan kembali lagi ke (2) dan seterusnya.

Jadi yang akan tercetak adalah harga-harga 4,9,16....(tidak berhenti-henti). Pada contoh diatas, perulangan berlangsung tak hingga, untuk itu perulangan perlu dibatasi dengan cara memanfaatkan kotak keputusan.

1.1.6 Panji(Flag)

Kita dapat menggunakan diagram alur untuk menggambarkan proses yang berlangsung berulang kali dan untuk menandai bahwa data telah habis, digunakan teknik panji yaitu membuat suatu data nilai tertentu yang diletakkan pada bagian akhir himpunan. Contoh pembacaan record pada file diakhiri dengan EOF (end of file).



1.1.7 Jenis struktur kontrol dasar algoritma:

- Sequential
- Selection
- Repeation (Looping)

1.1.8 Karakteristik dasar algoritma:

- Solusi harus terukur
- Memiliki instruksi yang jelas
- Memiliki masukan untuk instruksi
- Memiliki keluaran sebagai hasil dari eksekusi
- Beroperasi secara efektif

Teknik pembuatan algoritma dapat berupa flowchart, pseudo code, bahasa dlsb.
Faktor yang menentukan baik buruknya algoritma adalah:

- Running time
- Jumlah penggunaan memori

1.2 Struktur data

Struktur data adalah suatu cara untuk menyimpan dan mengatur data dalam komputer sehingga dapat digunakan secara efisien. Pemilihan struktur data yang baik dan tepat dapat menghasilkan algoritma yang efisien.

Ciri-ciri desain struktur data yang baik adalah:

Memenuhi berbagai kemungkinan dari operasi yang akan dijalankan

Menggunakan sedikit sumber daya baik execution time dan penggunaan memori.

Operasi yang dapat dilakukan pada struktur data:

- Traversing, akses dan proses setiap data dalam struktur data
- Searching, mencari data dan lokasinya
- Insertion, menyisipkan item pada list item
- Deletion, menghapus item dari satu set data
- Sorting, mengurutkan data dalam urutan tertentu
- Merging, menggabungkan beberapa grup data

1.2.1 Tipe data pada struktur data

Tipe data diperlukan agar compiler tahu operasi apa yang valid dan seberapa banyak memori yang diperlukan oleh sebuah nilai yang akan disimpan atau dioperasikan. Variabel digunakan untuk menampung suatu nilai, karena itu setiap variabel pasti memiliki tipe data dan harus dideklarasikan terlebih dahulu sebelum dapat digunakan.

Dalam Java, tipe data dapat dikelompokkan menjadi dua jenis tipe data, yaitu tipe data primitif dan referensi.

- Tipe Data Primitif
- Tipe Data Referensi

TUGAS:

A. Buatlah diagram alur untuk kasus dibawah ini

1. Menghitung luas segitiga, bila diketahui Alas=10 dan tinggi=8 (Catatan:* adalah operasi perkalian, / adalah operasi pembagian)
2. Menghitung luas segitiga dengan alas dan tingginya diinputkan oleh user dari keyboard.
3. Menghitung perkalian 3 buah bilangan yang diinputkan oleh user dari keyboard
4. Mencetak data mahasiswa yang diinputkan oleh user melalui keyboard(data mahasiswa terdiri dari: nama, alamat, tgl lahir, jurusan, kelas, angkatan)
5. Terdapat 2 dua penilaian untuk mahasiswa berprestasi yaitu nilai akademik dan nilai non akademik. Mahasiswa dikatakan berprestasi “baik” bila rata-rata penilaiannya ≥ 70 dan dikatakan “kurang” bila rata-rata penilaiannya < 70 .
6. Dalam test masuk calon mahasiswa terdapat dua ujian yang harus diikuti yaitu ujian a dan b. Nilai ujian dihitung dengan menggunakan rumus $60\%a + 40\%b$. Calon mahasiswa diterima bila nilai ujiannya ≥ 70 keatas, cadangan jika nilai ujiannya $70 < \text{nilai ujian} \leq 60$, tidak diterima jika nilai ujiannya < 60 .
7. Dalam penilaian pegawai untuk naik pangkat ditetapkan salah satu yang dinilai adalah kemampuan membuat karya ilmiah. Sebagai pengarang buku mendapat Kum 3, sebagai pengarang diktat mendapat Kum 2 dan pengarang paper mendapat Kum 1. Seseorang dapat naik pangkat bila Kumnya 10 atau lebih dari 7 dalam pertimbangan, Selain itu belum berhak naik pangkat.
8. Dalam acara orientasi studi, seorang peserta harus mencari tanda tangan. Tanda tangan dosen dinilai 3, tandatangan mahasiswa senior yang menjadi panitia mendapat nilai 2. Tandatangan mahasiswa senior yang bukan menjadi panitia mendapat nilai 1. Peserta dinyatakan lulus bila berhasil mendapatkan nilai 80 atau lebih, mendapat hukuman ringan bila kurang dari 80 tetapi masih lebih dari 60, selain itu mendapat tugas berat,
9. Menentukan bilangan terbesar dalam suatu himpunan data dengan ketentuan himpunan data sebagai berikut 8 5 9 12 6 999(terdapat penggunaan harga panji)

B. Buatlah diagram alur untuk kasus dibawah ini(terdapat penggunaan berkas)

1. Untuk mengirim paket melalui P.T Paket kilat dikenakan biaya sebagai berikut:
 - Biaya administrasi Rp.5000.
 - Biaya per Kg adalah Rp 10.000,-minimal dihitung 2 kg yaitu Rp 20.000.
 - Untuk paket yang lebih dari 20 Kg dikenakan tambahan Rp 3000 per Kg tambahan. Lebih dari 40 Kg dikenakan tambahan Rp 4000 per Kg tambahan.Buatlah diagram alur yang mencetak laporan paket-paket yang dikirim, membuat nomor paket, berat paket, tujuan paket(kota) dan ongkos paket tersebut.
2. Untuk mengatur kelancaran perusahaan P.T Medal Bunga Harum, sebuah perusahaan bis kota menetapkan setoran setiap bisnya sebagai berikut:
 - Setoran untuk perawatan Rp.16.000.
 - Setoran per jam Rp.5000,-minimal dibayar 3 jam (Rp15.000)
 - Bis yang beroperasi lebih dari 6 jam dikenakan tambahan Rp 2000,-per jam tambahan dan lebih dari 10 jam tambahan Rp 3000,-per jam tambahan.Buatlah diagram alur yang mencetak laporan setoran bis, memuat nomor bis, nama sopir, jam operasi dan besarnya setoran.
3. Untuk keperluan resepsi, gedung Graha Widya Sentosa menetapkan tarip sebagai berikut:
 - Sewa gedung Rp 2.000.000
 - Sewa kursi Rp 2000 per buah minimal untuk 100 kursi

- Lebih dari 300 kursi dikenakan tambahan Rp 500 per kursi tambahan sedangkan lebih dari 500 kursi dikenakan tambahan Rp 1000 per kursi tambahan.
Buatlah diagram alur yang mencetak daftar penyewaan gedung, memuat nama penyewa, tanggal pemakaian gedung, banyak kursi serta jumlah yang harus dibayar.

BAB 2 Bahasa Pemrograman Java

TIK:

Mahasiswa dapat memahami konsep bahasa pemrograman Java

Pokok Bahasan:

- Pendahuluan
- Komponen JDK

Sub Pokok Bahasan:

- Konsep kompilasi dan eksekusi programhelloworld

2.1 Pendahuluan

Karakteristik Java :

- Berorientasi objek
- Robust
- Portable
- Multitreading
- Dinamis
- Sederhana
- Terdistribusi
- Aman
- Netral secara arsitektur
- Intrepreted
- Berkinerja tinggi

Java/Standard Development Kit (JDK/SDK) merupakan alat-alat utama bagi programmer untuk membuat dan menjalankan java.

2.2 Komponen JDK

- compiler(javac),
Bertugas untuk melaksanakan kompilasi *.java menjadi *.class
Syntax umum : javac nama_file.java
- interpreter(java) disebut juga java virtual machine atau java runtime environment,
Bertugas untuk menjalankan bytecode (*.class)
Bertugas untuk menjalankan bytecode (class)
Syntax umum : java nama_file.class
- applet viewer(appletviewer),
Digunakan untuk menjalankan applet viewer, namun sekarang sudah digantikan browser.
Syntax umum : appletviewer nama_file.html
- debugger
Bertugas untuk melakukan debugging aplikasi java.
Syntax umum : jdb option
- java class library(jcl)
- header dan stub generator(javah)
- java documentation(javadoc)

Berikut ini contoh program Java yang sederhana:

```
/*  
Contoh program java sederhana  
-----  
*/  
class Hello{  
//awal program selalu dimulai dari main()  
public static void main(String[] args) {  
System.out.println("Hello Java...");
```



```
}  
}
```

Java bersifat case sensitive. Huruf besar dan kecil adalah berbeda dalam Java. Oleh karena itu, pastikan mengetik code di atas sama persis, tanpa mengubah tipe huruf. Selain itu dalam Java, white space seperti karakter spasi, tab, pindah baris, dan karakter lainnya yang berfungsi untuk memformat tampilan, tidak memiliki arti apa pun selain untuk memudahkan kita membaca code yang ditulis. Oleh karena itu, gunakan karakter white space ini sesuka Anda untuk memformat tampilan code yang ditulis, agar memudahkan Anda atau siapa pun membacanya.

Selalu simpan setiap class yang dibuat ke dalam filenya masing-masing yang sesuai dengan nama class-nya. Contoh, jika Anda membuat class Test, simpan ke dalam file Test.java.

Lakukan kompilasi program di atas dengan mengetikkan perintah berikut pada command prompt:
javac Hello.java

Setelah kompilasi, Anda akan mendapatkan file bytecode dengan nama Hello.class. Ketik perintah berikut untuk mengeksekusi program Java ini:

```
java Hello
```

Harus diingat, pada perintah di atas, Hello adalah nama class, bukan nama file, karena itu jangan memasukkan nama ekstensi file .class. Program di atas sangat sederhana, namun memiliki beberapa bagian penting. Kita perjelas dengan melihat bagian-bagian tersebut satu per satu. Dimulai dengan bagian:

```
/*  
Contoh program java sederhana  
-----  
*/
```

Bagian ini merupakan bagian yang dikenal sebagai komentar. Komentar bukan merupakan bagian dari program karena ia tidak akan di-compile oleh compiler Java dan tidak akan mempengaruhi alur eksekusi program. Komentar ini lebih ditujukan untuk memperjelas source code yang kita tulis sehingga code akan lebih mudah dibaca oleh siapa pun. Komentar di atas ditulis di antara tanda /* dan */.

```
class Hello {
```

Baris code ini mendeklarasikan sebuah class dengan nama Hello. Perhatikan bahwa keseluruhan deklarasi dari class ini diawali dengan { dan ditutup dengan }. Semua code yang terletak di antara { } merupakan bagian dari class ini.

```
//awal program selalu dimulai dari main()
```

Baris code ini juga merupakan komentar. Perbedaanannya dengan /* */ adalah bahwa // hanya berlaku untuk satu baris ini saja dan semua kata-kata setelah tanda // merupakan komentar.

```
public static void main(String[] args) {
```

Baris code di atas mendeklarasikan suatu method dengan nama main. Nama main di sini merupakan suatu keharusan dalam Java karena Java akan mencari method yang bernama main ini sebagai titik awal eksekusi program. Keyword public merupakan access specifier yang menentukan visibility

level dari method ini.

Public berarti method ini dapat diakses/dipanggil dari luar class di mana ia dideklarasikan. Method main() akan dipanggil dari luar oleh run-time Java pada saat program akan dieksekusi sehingga access specifier yang dimilikinya haruslah public. Keyword static memungkinkan method main() dipanggil tanpa harus terlebih dahulu membuat instance dari class Hello. Ini diperlukan karena method main() akan dieksekusi sebelum objek dari class Hello dibuat di memori. Keyword void berarti bahwa method main() tidak mengembalikan nilai apa pun setelah dipanggil/dieksekusi.

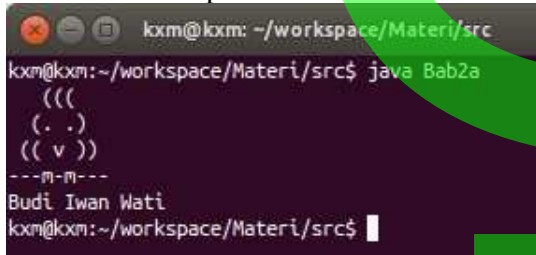
```
System.out.println("Hello Java...");
```

Baris code ini memiliki beberapa bagian penting, antara lain:

- System adalah nama dari salah satu class standar yang dimiliki oleh Java.
- out merupakan anggota dari class System dan juga merupakan objek tersendiri, out merupakan objek yang mewakili standard output stream yang dalam hal ini adalah layar komputer. Seperti halnya dengan method main() di atas, objek out ini dideklarasikan menggunakan keyword static di dalam class-nya sehingga ia dapat langsung dipanggil tanpa perlu terlebih dulu membuat instance dari class System.
- println merupakan method yang terdapat pada objek out. Berfungsi untuk mencetak keluaran ke standard output. Method ini juga mencetak karakter pindah baris. Untuk mencetak keluaran ke standard output tanpa mencetak karakter pindah baris, Anda dapat menggunakan method print().
- "Hello Java..." merupakan parameter dari method println() yang diterima oleh internal method ini dan dicetak ke standard output yang dalam hal ini adalah layar komputer sehingga pada saat Anda menjalankan program ini, di layar komputer akan tercetak Hello Java...
- Tanda ; yang Anda lihat di paling belakang baris diperlukan untuk menandakan akhir suatu statement/pernyataan/perintah.

TUGAS:

1. jalankan helloworld pada terminal atau command prompt
javac
java
2. jalankan helloworld pada eclipse, console
3. buat output:



```
kxm@kxm: ~/workspace/Materi/src
kxm@kxm:~/workspace/Materi/src$ java Bab2a
(((
(. .)
(( v ))
---m-m---
Budi Iwan Wati
kxm@kxm:~/workspace/Materi/src$
```

4. Laporan beri penjelasan (analisa logika).

BAB 3 Tipe data

TIK:

Mahasiswa dapat mengenal tipe data

Pokok Bahasan:

- Identifier
- Reserved Words/Keywords
- Variabel
- Tipe Data

Sub Pokok Bahasan:

- Scope dari Variabel
- Tipe Data Primitif
- Tipe Data Referensi

3.1 Identifier

Identifier merupakan nama yang digunakan untuk menamai class, interface, variabel, dan method. Anda dapat menentukan sendiri identifier yang akan digunakan. Yang perlu diperhatikan mengenai identifier ini adalah:

- Tidak ada batasan panjang, jadi kita dapat membuat identifier dengan panjang berapa pun.
- Identifier harus diawali dengan huruf, underscore/garis bawah (_) atau lambang dolar (\$). Untuk selebihnya dapat menggunakan karakter apa pun, kecuali karakter yang digunakan sebagai operator oleh Java (seperti +, -, *, atau /).
- Bukan merupakan keywords yang dikenal oleh Java.

Pada contoh program sebelumnya, Hello merupakan identifier yang digunakan untuk menamai class yang dibuat.

Contoh identifier yang valid:

strTemp	\$counter	b4Th33	_aValidOne
---------	-----------	--------	------------

Contoh identifier yang tidak valid:

4Ever	from-to	Good/Bad	*by4
-------	---------	----------	------

3.2 Reserved Words/Keywords

Merupakan kata-kata yang dikenal oleh kompiler Java dan memiliki arti khusus dalam program.

Berikut ini daftar kata-kata tersebut.

assert*	abstract	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
extends	enum**	false	final	finally
float	for	goto	if	implements
import	instanceof	int	interface	long
native	new	null	package	private
protected	public	return	short	static
strictfp	super	switch	synchronized	this
throw	throws	transient	true	try
void	volatile	while		

3.3 Variabel

Variabel merupakan lokasi penyimpanan yang ada di memori. Setiap variabel memiliki kemampuan menyimpan suatu informasi sesuai dengan tipe data yang dideklarasikan untuk variabel tersebut saja.

Sintaks pendeklarasian variabel secara umum adalah sebagai berikut:
`tipe-data nama-variabel;`

Tipe-data meliputi semua tipe data yang dikenal oleh Java, sedangkan nama-variabel adalah identifier yang akan digunakan untuk merujuk ke variabel tersebut di dalam program.

Contoh code:
`int counter;`

Code di atas mendeklarasikan suatu variabel yang bernama counter dengan tipe data int.

3.3.1 Scope dari Variabel

Dalam Java, secara garis besar scope dari variabel dapat dibedakan menjadi dua bagian, yaitu variabel yang dideklarasikan di dalam blok class (dikenal juga sebagai property) dan variabel yang dideklarasikan di dalam blok code.

Variabel yang dideklarasikan dalam blok class akan dikenal di bagian manapun dalam blok class tersebut. Variabel ini juga bahkan dapat diakses dari luar class menggunakan referensi objek atau instance dari class tersebut.

Pada dasarnya variabel dapat dideklarasikan di dalam blok code manapun. Yang dimaksud dengan blok code di sini adalah bagian dari code yang dimulai dengan karakter { dan ditutup dengan karakter }. Blok ini menentukan scope dari suatu variabel, yaitu apakah suatu variabel akan dikenal di bagian lain dari program atau tidak. Perhatikan juga bahwa Anda dapat membuat nested blok, di mana di dalam suatu blok code terdapat blok code lainnya. Aturan sederhana yang perlu diingat hanyalah, pendeklarasian suatu variabel dalam suatu blok code akan dikenal oleh nested blok yang ada di dalam blok code tersebut, namun tidak berlaku sebaliknya. Untuk lebih jelasnya, perhatikan

contoh code berikut ini:

```
class Scope {
    static
    public
    int
    x =
    int a = 2; //deklarasi variabel dalam blok class
    static void main(String[] args) {
        x; //variabel x ini dikenal di seluruh method main()
    }
    //variabel a juga dikenal di sini
    System.out.println("Nilai a : " + a);
    { //awal dari blok baru
        int y; //variabel ini hanya dikenal di dalam
        //blok code ini saja
        y = 5;
        //variabel x dikenal di sini
        System.out.println("Nilai x : " + x);
    }
}
```

```
//variabel a juga dikenal di sini
System.out.println("Nilai a : " + a);
{ //nested blok
int z;//variabel ini hanya dikenal di
//dalam nested blok ini saja
z = 20;
/* variabel x,y dan a dikenal di dalam
nested blok ini */
System.out.println("Nilai x+y+z+a : "
+ (x + y + z + a));
} //akhir dari nested blok
//z = 11; // -> Error variabel z tidak lagi
//dikenal di sini
/* variabel y masih dikenal di sini karena
masih dalam blok code tempat ia
dideklarasikan */
System.out.println("Nilai y : " + y);
}
//akhir dari blok baru
//y = 12; // -> Error: variabel y tidak dikenal di sini
/* variabel x masih dikenal di sini karena masih
dalam blok code tempat ia dideklarasikan */
System.out.println("Nilai x : " + x);
} //akhir dari method main
} //akhir dari deklarasi class
```

Deklarasi variabel dapat diletakkan di baris manapun dari program, tetapi perhatikan juga bahwa Anda tidak dapat menggunakan variabel yang belum dideklarasikan, sebagai contoh:

```
index = 0; // -> Error, variabel index belum dideklarasikan
int index; // -> di sini index baru dideklarasikan sebagai tipe int
```

Anda juga dapat langsung memberikan nilai pada suatu variabel pada saat mendeklarasikannya, sebagai contoh:

```
int index = 10;
```

Harus diingat bahwa variabel yang telah dideklarasikan dalam blok class, dapat dideklarasikan ulang (dengan nama yang sama) dalam blok code, namun variabel yang telah dideklarasikan dalam suatu blok code, tidak boleh dideklarasikan lagi dalam scope blok code yang sama (di dalam scope di mana variabel tersebut masih dikenal). Perhatikan contoh berikut ini:

```
class Scope {
int x; // deklarasi variabel int dengan nama
//
x dalam blok class
public static void main(String[] args) {
int x; // deklarasi variabel int
// dengan nama x dalam blok code
// baris code lainnya...
{
```

```
// awal dari blok(scope) baru
int x;// Error, variabel x telah
// dideklarasikan sebelumnya
// baris code lainnya...
}
int x; // Error, variabel x telah
// dideklarasikan sebelumnya pada
// blok code yang sama
} //akhir dari method main
}
```

3.4 Tipe Data

Tipe data diperlukan agar kompiler tahu operasi apa yang valid dan seberapa banyak memori yang diperlukan oleh sebuah nilai yang akan disimpan atau dioperasikan. Variabel digunakan untuk menampung suatu nilai, karena itu setiap variabel pasti memiliki tipe data dan harus dideklarasikan terlebih dahulu sebelum dapat digunakan. Perbedaan tipe data juga mengakibatkan setiap operasi penugasan, baik secara eksplisit maupun melalui passing parameter pada waktu pemanggilan method, selalu dicek kompatibilitas tipe datanya.

Dalam Java, tipe data dapat dikelompokkan menjadi dua jenis tipe data, yaitu tipe data primitif dan referensi.

3.4.1 Tipe Data Primitif

Tipe data primitif merupakan tipe data dasar yang dikenal oleh Java. Terdapat delapan buah tipe data primitif yang dikenal dalam Java sebagai berikut.

Nama Data	Tipe	Ukuran dalam bit	Nilai	Standar
byte		8	-2^7 s.d. 2^7-1	
short		16	-2^{15} s.d. $2^{15}-1$	
Int		32	-2^{31} s.d. $2^{31}-1$	
long		64	-2^{63} s.d. $2^{63}-1$	
float		32	Negatif: -3.4028234663852886E+38 s.d. -1.40129846432481707e-45 Positif: 1.40129846432481707e-45 s.d. 3.4028234663852886E+38	IEEE 754 floating point
double		64	Negatif: -1.7976931348623157E+308 s.d. -4.94065645841246544e-324	IEEE 754 floating point

		Positif: 4.94065645841246544e-324 s.d. 1.7976931348623157E+308	
char	16	'\u0000' s/d '\uFFFF' (0 s/d 65535)	ISO Unicode Charater Set
boolean	8	true atau false	

Kedelapan tipe data primitif ini dapat dikelompokkan ke dalam empat group:

1. Integer merupakan tipe data bilangan bulat yang terdiri atas byte, short, int, dan long.
2. Floating-Point merupakan tipe data bilangan pecahan yang terdiri atas float dan double.
3. Karakter mewakili simbol dari sebuah karakter yang terdiri atas char.
4. Boolean merupakan tipe data yang menunjukkan nilai true atau false, yang terdiri atas boolean.

Variabel dengan tipe data primitif ini dapat digunakan secara langsung untuk menyimpan suatu nilai tertentu.

Contoh:

```
int indeks = 2;
int hasil = indeks * 11;
```

3.4.2 Tipe Data Referensi

Tipe data referensi digunakan untuk memegang referensi dari suatu objek (instance dari class). Pendeklarasian variabel tipe data ini sendiri sama dengan tipe data primitif. Namun, penggunaannya agak sedikit berbeda.

Perhatikan contoh code di bawah ini untuk lebih jelasnya:

```
class ProgSederhana {
int x = 0; /* Pendeklarasian variabel dengan
tipe data int */
public static void main(String[] args) {
/*Pendeklarasian variabel dengan tipe data
class ProgSederhana*/
ProgSederhana var;
//Instantiate class ProgSederhana menjadi
//objek
var = new ProgSederhana();
/*setelah proses instantiate ini, Anda
dapat mengakses objek ProgSederhana
melalui variabel var*/
var.x = 20;
System.out.println("Nilai x : " + var.x);
}
}
```

Code berikut ini:

```
var = new ProgSederhana();
```

memerintah run-time Java untuk mengalokasikan memori membuat objek ProgSederhana di memori dan menyerahkan referensi (alamat) dari objek tersebut untuk dipegang oleh variabel var.

Inilah yang kita sebut dengan instantiate class menjadi sebuah objek. Dengan variabel var ini, Anda dapat mengakses internal objek yang telah dibuat tersebut:

```
var.x = 20;
```

Contoh penggunaan tipe data bentukan ketika kita akan menangkap input dari keyboard dengan menggunakan class Scanner.

```
import java.util.Scanner;

//http://www.programmingsimplified.com/java/source-code/java-program-take-input-
from-user

public class HelloInOut {

    public static void main(String[] args) {
        int a;
        float b;
        String s;

        Scanner in = new Scanner(System.in);

        System.out.println("Enter a string");
        s = in.nextLine();
        System.out.println("You entered string "+s);

        System.out.println("Enter an integer");
        a = in.nextInt();
        System.out.println("You entered integer "+a);

        System.out.println("Enter a float");
        b = in.nextFloat();
        System.out.println("You entered float "+b);

    }
}
```

TUGAS:

1. buat inputan dari keyboard, penjumlahan dua buah bilangan desimal, sehingga outputnya:
Masukan angka pertama?

2

Masukan angka kedua?

4

hasil variabel $x + y$ adalah

6.0

2. Laporan beri penjelasan (analisa logika).

BAB 4 Alur Program

TIK:

Mahasiswa dapat memahami alur kontrol percabangan pada program

Pokok Bahasan:

- If
- If-Else
- Switch

Sub Pokok Bahasan:

4.1 If

Percabangan IF di Java menyatakan bahwa suatu statement (pernyataan) akan dieksekusi bila memenuhi syarat/kondisi tertentu

If Ekspresi Statement

```
public class If01 {  
    public static void main(String args[]) {  
        int anInt = 0;  
        if (anInt == 0) {  
            System.out.println("Variabel anInt bernilai Nol");  
        }  
    }  
}
```

4.2 If-Else

Percabangan ini untuk memilih salah satu dari dua kemungkinan kemunculan.

if (Condition)
Statement1
else
Statement2

```
public class IfElse01 {  
    public static void main(String args[]) {  
        int anInt = 1;  
        System.out.println("Nilai Variabel anInt : " + anInt);  
        if (anInt == 0) {  
            System.out.println("Variabel anInt bernilai Nol");  
        } else {  
            System.out.println("Variabel anInt tidak bernilai Nol");  
        }  
    }  
}
```

4.3 Switch

Percabangan Switch dimaksudkan untuk menangani banyak kemungkinan kemunculan :

switch (ekspresi) {
case Constant1:
statementlist1
case Constant1:
statementlist1
default:
defaultstatementlist
}

```
public class Switch01 {  
    public static void main(String args[]) {
```

```
int a;  
a = 5;  
switch (a) {  
case 0:  
    System.out.println("Nol");  
    break;  
case 1:  
    System.out.println("Satu");  
    break;  
case 2:  
    System.out.println("Dua");  
    break;  
case 3:  
    System.out.println("Tiga");  
    break;  
case 4:  
    System.out.println("Empat");  
    break;  
case 5:  
    System.out.println("Lima");  
    break;  
case 6:  
    System.out.println("Enam");  
    break;  
case 7:  
    System.out.println("Tujuh");  
    break;  
case 8:  
    System.out.println("Delapan");  
    break;  
case 9:  
    System.out.println("Sembilan");  
    break;  
default:  
    System.out.println("Bukan Karakter Digit");  
}  
}
```

TUGAS:

1. input grade dan outputnya, gunakan class Scanner dan logika if else, sehingga outputnya:
contoh nilai inputnya: 30, 61, 80, 98

Masukan nilai Anda?

30

Anda gagal

Masukan nilai Anda?

61

belajarlal dengan giat!

Masukan nilai Anda?

80

Baik!

Masukan nilai Anda?

98

Sempurna!

2. Gunakan switch case, sehingga outputnya:

Pilih Menu makanan

=====

- 1. Nasi
- 2. Roti
- 3. lotek

3

Anda memilih makan lotek

3. Laporan beri penjelasan (analisa logika).

BAB 5 Perulangan

TIK:

Mahasiswa dapat memahami kontrol perulangan

Pokok Bahasan:

- For
- While
- Do-While

Sub Pokok Bahasan:

5.1 For

Perulangan for menyediakan sarana mengulang kode dalam jumlah tertentu, bersifat terstruktur untuk mengulangi kode sampai tercapai batas tertentu.

for (InitializationExpression; LoopCondition; StepExpression)
Statement

- InitializationExpression :
Digunakan untuk inisialisasi variabel kendali pengulangan
- LoopCondition :
Membandingkan variabel kendali pengulangan dengan suatu nilai batas
- StepExpression :
Menspesifikasikan cara variabel kendali dimodifikasi sebelum iterasi berikutnya dari perulangan.

Nested for

```
public class NestedFor {  
    public static void main(String args[]) {  
        for (int i = 1; i <= 10; i++) {  
            for (int j = i; j <= 10; j++) {  
                System.out.print(j);  
            }  
            System.out.println();  
        }  
    }  
}
```

VARIASI dari perulangan for :

```
public class VarianFor {  
    public static void main(String args[]) {  
        boolean finish = false;  
        int i = 1;  
        for (; !finish;) {  
            System.out.println("i= " + i);  
            if (i == 10)  
                finish = true;  
            i++;  
        }  
    }  
}
```

5.2 While

while (LoopCondition)
Statement

- Jika LoopCondition dievaluasi true, maka Statement dieksekusi dan proses berlanjut diulangi.
- Jika LoopCondition sejak semula dievaluasi false, maka statement tak pernah dieksekusi.

```

public class WhileLoop {
    public static void main(String args[]) {
        int i = 1;
        while (i <= 10) {
            int j = 1;
            System.out.println("i = " + i);
            i++;
        }
    }
}

```

5.3 Do-While

Perulangan do-while serupa dengan perulangan while, hanya pemeriksaan kondisi yang dilakukan adalah setelah statement.

do

Statement

while (LoopCondition)

```

public class DoWhileLoop {
    public static void main(String args[]) {
        int i = 1;
        do {
            System.out.println("i= " + i);
            i++;
        } while (i < 11);
    }
}

```

Contoh:

Loop membuat kotak

```

public class Bab5b {

    public static void main(String[] args) {
        int batasbaris = 10;
        int bataskolom = 10;

        for (int i = 1; i <= batasbaris; i++) {
            for (int j = 1; j <= bataskolom; j++) {
                if (j == 1 || j == bataskolom || i == 1 || i == batasbaris) {
                    System.out.print("x");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}

```

Mengetahui panjang karakter

```
import java.util.Scanner;
```

```
public class Bab5c {  
    public static void main(String[] args) {  
        // cek panjang nama  
        String nama;  
        Scanner inp = new Scanner(System.in);  
  
        System.out.println("Masukan nama");  
        nama = inp.nextLine();  
  
        System.out.print("Panjang nama Anda adalah:");  
        System.out.println(nama.length());  
    }  
}
```

TUGAS:

1. buat kotak, input panjang dan lebarnya
Masupakn batas baris?

5

Masupakn batas kolom?

5

xxxxx

x x

x x

x x

xxxxx

2. input nama, buat nama tersebut berada dalam kotak, sehingga:

Masukan nama

Erika Mariam

#####

#Erika Mariam#

#####

3. Laporan beri penjelasan (analisa logika).

BAB 6 Class

TIK:

Mahasiswa dapat memahami cara bekerja class

Pokok Bahasan:

- Class
- Attribute
- Behavior Method
- Object

Sub Pokok Bahasan:

6.1 Class

Class adalah struktur dasar dari OOP (Object Oriented Programming). Terdiri dari dua tipe yaitu : field (attribute/property) dan method (behavior). Class digunakan untuk mendeklarasikan sebuah variabel yang berupa objek atau dinamakan “referensi objek (object reference)”

6.2 Attribute

- Berlaku sebagai data, didefinisikan oleh class, individu, berbeda satu dengan lainnya.
- Menggambarkan tampilan, status, kualitas dari object.
- Contoh :

class motorcycle

attribute-nya = color [red, green, silver]

style [cruiser, sport bike, standart]

make [Honda, BMW]

- Didefinisikan dalam class dengan menggunakan variabel.

6.3 Behavior Method

- Berlaku sebagai method (operasi).
- Menggambarkan bagaimana sebuah instance class beroperasi misal bagaimana reaksi dari class jika diminta untuk melakukan sesuatu hal.
- Contoh :

class motorcycle

behavior-nya = start the engine

stop the engine

change gear

- Untuk menentukan behavior dari object harus membuat Methods.

Contoh:

```
class Motorcycle {
    String make;
    String color;
    boolean engineState;

    void startEngine() {
        if (engineState == true)
            System.out.println("The engine is already on.");
        else {
            engineState = true;
            System.out.println("The engine is now on.");
        }
    }

    void showAtts() {
        System.out.println("This motorcycle is a " + color + " " + make);
        if (engineState == true)
            System.out.println("The engine is on.");
        else
            System.out.println("The engine is off.");
    }
}
```

```

public static void main(String args[]) {
    Motorcycle m = new Motorcycle();
    m.make = "Yamaha RZ350";
    m.color = "yellow";
    System.out.println("Calling showAtts...");
    m.showAtts();
    System.out.println("-----");
    System.out.println("Starting engine...");
    m.startEngine();
    System.out.println("-----");
    System.out.println("Calling showAtts...");
    m.showAtts();
    System.out.println("-----");
    System.out.println("Starting engine...");
    m.startEngine();
}
}

```

6.4 Object

Setiap Object (obyek) dibangun dari sekumpulan data (atribut) yang disebut "variabel" (untuk menjabarkan karakteristik khusus dari obyek) dan juga terdiri dari sekumpulan method (menjabarkan tingkah laku dari obyek) atau Obyek adalah = sebuah perangkat lunak yg berisi sekumpulan variabel dan method yg berhubungan. Obyek mrpk.sebuah instance (keturunan) dari class. Variabel dan method diketahui sbg. Variabel instance dan method instance.

Contoh:

```

// buat kelas
class Manusia {
    String namalengkap; // variabel namalengkap bersifat public

    public void membaca() {
        System.out.println("saya membaca");
    }
}

public class Bab6a {

    public static void main(String[] args) {
        // objek tipe data primitif
        int x;
        x = 5;

        // objek tipe data dari class bentukan
        Manusia y;
        y = new Manusia();
        y.namalengkap = "Budi Iwan Wati";

        // hasil
        System.out.println(x);
        System.out.println(y.namalengkap);
        y.membaca();
    }
}

```

TUGAS:

1. Lengkapi program:

```
class Obj2D {  
    double panjang;  
    double lebar;  
  
    public double hitungLuasKotak(){  
        return panjang * lebar;  
    }  
  
    public hitungLuasSegitiga(){  
  
    }  
  
}  
  
public class Bab6b {  
  
    public static void main(String[] args) {  
        Obj2D bangunan=new Obj2D();  
  
        bangunan.lebar=10.0;  
  
        double y=bangunan.hitungLuasSegitiga();  
  
        System.out.println("luas kotag adalah=" + x);  
        System.out.println("luas sgitigq adalah=" + y);  
  
    }  
}
```

contoh output:

```
luas kotag adalah=300.0  
luas sgitigq adalah=150.0
```

2. Buat kelas dengan package yang terpisah (Anda harus menggunakan fitur import), contoh:

```
import matmatkaSD.Obj2D;  
  
public class Bab6c {  
  
    public static void main(String[] args) {  
        Obj2D bentuk = new Obj2D();  
  
        bentuk.panjang = 30;  
        bentuk.lebar = 10;  
  
        System.out.println(bentuk.hitungLuasKotak());  
    }  
}
```

2. Laporan beri penjelasan (analisa logika).

BAB 7 Fungsi

TIK:

Mahasiswa dapat memahami penggunaan method

Pokok Bahasan:

- Return
- Void

Sub Pokok Bahasan:

Pada java bethod atau fungsi merupakan kumpulan statement yang dikelompokkan untuk melakukan suatu operasi atau fungsi. Sebagai contoh pada `System.out.println()`, mesin akan melakukan instruksi untuk menampilkan output pada layar console.

Contoh:

```
public static int funcName(int a, int b) {  
    // body  
}
```

- `public static` : modifier/bentuk
- `int`: tipe nilai kembalian
- `funcName`: nama fungsi
- `a, b`: nama parameter (bervariasi)
- `int a, int b`: tipe data parameter

Method dikenal juga dengan Procedures atau Functions.

Contoh:

```
public class ExampleMinNumber{  
  
    public static void main(String[] args) {  
        int a = 11;  
        int b = 6;  
        int c = minFunction(a, b);  
        System.out.println("Minimum Value = " + c);  
    }  
  
    /** returns the minimum of two numbers */  
    public static int minFunction(int n1, int n2) {  
        int min;  
        if (n1 > n2)  
            min = n2;  
        else  
            min = n1;  
  
        return min;  
    }  
}
```

7.1 Return

keyword return digunakan untuk mengembalikan nilai hasil.

7.2 Void

Jika method melakukan instruksi yang tidak memerlukan nilai kembali. Dapat digunakan keyword void.

Contoh:

```

class Operasi {
    private int x = 100;
    private int y = 200;
    private int z;

    public int jumlahBil() {
        z = x + y;
        return z;
    }

    public void jumlahBil2() {
        z = x + y;
        System.out.println(z);
    }
}

public class Bab7a {

    public static void main(String[] args) {
        Operasi op = new Operasi();
        int x = 0;

        // test return
        x = op.jumlahBil();
        System.out.println(x);

        // test void
        op.jumlahBil2();
    }
}

```

Contoh method, fungsi dengan melibatkan parameter:

```

class Operasi2 {

    public int jumlahBil(int x, int y) {
        int z = 0;
        z = x + y;
        return z;
    }

    public void jumlahBil2(int x, int y) {
        int z = 0;
        z = x + y;
        System.out.println(z);
    }
}

public class Bab7b {

    public static void main(String[] args) {
        Operasi2 op = new Operasi2();
        int x = 0;
    }
}

```

```

        // test return
        x = op.jumlahBil(5, 4);
        System.out.println(x);

        // test void
        op.jumlahBil2(8, 7);
    }
}

```

Contoh method, fungsi dengan fitur static:

```

class Operasi2 {

    public static int jumlahBil(int x, int y) {
        int z = 0;
        z = x + y;
        return z;
    }

    public void jumlahBil2(int x, int y) {
        int z = 0;
        z = x + y;
        System.out.println(z);
    }
}

public class Bab7b {

    public static void main(String[] args) {
        Operasi2 op = new Operasi2();
        int x = 0;

        // test return
        x = op.jumlahBil(5, 4);
        System.out.println(x);

        // test void
        op.jumlahBil2(8, 7);

        // method/fungsi static dapat diakses tanpa perlu membuat objek baru
        // (instance)
        x = Operasi2.jumlahBil(45, 5);
        System.out.println(x);
    }
}

```

contoh lain penggunaan static method:

```

public class Bab7c {

    public static void main(String[] args) {
        int isinya=0;
        isinya=jumlahBil(3, 4);

        System.out.println(isinya);
    }
}

```

```
}  
  
//penggunaan method static  
public static int jumlahBil(int x, int y) {  
    return x + y;  
}  
}
```

TUGAS:

1. Apa yang dimaksud static
2. jika fitur static pada method/fungsi main Anda hapus apa yang akan terjadi?
3. Buatlah program menggunakan fungsi dan kelas baru (tanpa merubah kelas Bab7d), sehingga

```
public class Bab7d {  
  
    public static void main(String[] args) {  
        Kotak k=new Kotak();  
        k.cetak(5, 10, '&');  
    }  
}
```

dan output:

```
@@@@@@@@@  
&         &  
&         &  
&         &  
@@@@@@@@@
```

3. Laporan beri penjelasan (analisa logika).

BAB 8 Constructor

TIK:

Mahasiswa dapat memahami penggunaan constructor

Pokok Bahasan:

- constructor

Sub Pokok Bahasan:

Constructor adalah method khusus yang didefinisikan di dalam kelas dan akan dipanggil secara otomatis setiap kali terjadi instansiasi objek, dan merupakan method yang mengembalikan tipe kelas (dirinya sendiri). Fungsi dari constructor adalah untuk melakukan instansiasi nilai terhadap data-data yang terdapat pada kelas bersangkutan.

Apabila tidak mendefinisikan constructor maka secara otomatis Java akan membuatnya untuk kita. Constructor semacam ini disebut dengan “default constructor”, yang akan menginisialisasikan semua data yang ada dengan nilai nol, string dengan nilai null, variabel boolean diset ke false.

Beberapa hal yang perlu diperhatikan pada saat mendefinisikan constructor kelas adalah constructor tidak mempunyai tipe kembalian, nama constructor harus sama persis dengan nama kelas yang didefinisikan.

Contoh:

```
class KotakKecil {
    double panjang;
    double lebar;
    double tinggi;

    // Mendefinisikan constructor untuk kelas Kotak
    KotakKecil() {
        panjang = 4;
        lebar = 3;
        tinggi = 2;
    }

    double hitungVolume() {
        return (panjang * lebar * tinggi);
    }
}

public class Bab8a {

    public static void main(String[] args) {
        KotakKecil k1, k2;
        k1 = new KotakKecil();
        k2 = new KotakKecil();
        System.out.println("Volume k1 = " + k1.hitungVolume());
        System.out.println("Volume k2 = " + k2.hitungVolume());
    }
}
```

Contoh:

overload constructor

```
class KotakKecil2 {
    double panjang;
    double lebar;
    double tinggi;

    // Mendefinisikan constructor untuk kelas Kotak
    public KotakKecil2() {
        panjang = 4;
        lebar = 3;
        tinggi = 2;
    }
}
```



```

    }
    //overload constructor
    public KotakKecil2(double p, double l, double t) {
        this.panjang = p;
        this.lebar = l;
        this.tinggi = t;
    }

    double hitungVolume() {
        return (panjang * lebar * tinggi);
    }
}
public class Bab8b {

    public static void main(String[] args) {
        KotakKecil2 k1, k2;
        k1 = new KotakKecil2();
        k2 = new KotakKecil2(5,8,3);
        System.out.println("Volume k1 = " + k1.hitungVolume());
        System.out.println("Volume k2 = " + k2.hitungVolume());
    }
}

```

TUGAS:

1. Lengkapi program berikut tanpa merubah class Bab8c:

```
public class Bab8c {  
  
    public static void main(String[] args) {  
        Mhs m = new Mhs("Yahya John", "50401912");  
        m.cetakInfo();  
    }  
}
```

output:

Nama saya: Yahya John dan npm: 50401912

2. Laporan beri penjelasan (analisa logika).

BAB 9 Penanganan Kesalahan

TIK:

Mahasiswa dapat menangkap error program

Pokok Bahasan:

- Error Handling
- Exception
- Try
- Catch secara bertingkat
- Melontarkan Exception

Sub Pokok Bahasan:

Kesalahan sering terjadi pada saat perancangan dan implementasi

Kesalahan dikategorikan :

- sintak error menyebabkan kesalahan kompilasi
- Semantic error , program menghasilkan keluaran yang tidak sesuai dengan harapan
- Run-time error, kebanyakan mengakibatkan terminasi program secara tidak normal atau bahkan sistem crash. Misal : penggunaan tipe data yang salah.

9.1 Error Handling

Setiap program yang berada dalam suatu kondisi yang tidak normal – Error Conditions. Program yang ‘baik’ harus dapat menangani kondisi ini. Java menyediakan suatu mekanisme untuk menangani kondisi ini – exceptions

9.2 Exception

Exception merupakan suatu keadaan yang disebabkan oleh runtime error dalam program. Memungkinkan kesalahan ditangani tanpa harus ‘mengotori’ program (dengan rutin yang menangani kesalahan) Memungkinkan pemisahan penanganan kesalahan dengan program utama

Contoh: Pembagian bil dengan nol

```
public class Bab9a {  
    public static void main(String[] args) {  
        System.out.println("Sebelum Pembagian");  
        System.out.println(5 / 0);  
        System.out.println("Setelah Pembagian");  
    }  
}
```

9.3 Try

Pernyataan try digunakan utk keperluan exception.

9.3.1 Bentuk 1:

```
try {  
    //Blok yang akan ditangkap sekiranya terjadi exception  
}  
catch(parameter)  
    //Blok yang akan dijalankan kalau terjadi exception  
}
```

Contoh :

```
public class Bab9b {  
    public static void main(String[] args) {  
        System.out.println("Sebelum Pembagian");  
        try {  
            System.out.println(5 / 0);  
        } catch (Throwable t) {  
            System.err.println("Terjadi Pembagian dengan nol");  
            System.err.println(t.getMessage());  
        }  
    }  
}
```

```

    }
    System.out.println("Setelah Pembagian");
}
// Throwable – nama kelas yg digunakan utk menangani exception.

```

9.3.2 Bentuk 2 :

```

try{
//blok yang akan ditangkap sekiranya terjadi
exception
}
finally
//blok yang akan dijalankan terakhir kali
}

```

finally selalu dijalankan baik sewaktu terjadi exception maupun sewaktu tidak terjadi exception.

Contoh :

```

public class Bab9c {
    public static void main(String[] args) {
        double bilangan = 100.0;
        System.out.println("Sebelum pembagian");
        for (int i = 5; i >= 0; i--) {
            try {
                System.out.print(bilangan + "/" + i + "=");
                System.out.println((bilangan / i));
            } finally {
                System.out.println("Bagian finally dijalankan");
            }
        }
        System.out.println("selesai");
    }
}

```

9.4 Catch secara bertingkat

Kelas Throwable memiliki sub kelas yaitu:

- Error digunakan utk menangani kesalahan spt memori habis (OutOfMemoryError) dan stack habis (StackOverflowError).
- Exception memiliki subkelas RuntimeException yg digunakan utk array tidak valid (IndexOutOfBoundsException) dan kesalahan aritmatika (ArithmeticException).

Bentuk:

```

try{
//blok yg akan ditangkap sekiranya terjadi exception
}
catch(RuntimeException r){
//blok yg akan dijalankan kalau terjadi eksepsi
RuntimeError
}

```

```

catch(Exception e){
//blok yg akan dijalankan kalau terjadi eksepsi Exception
}
catch(Throwable t){
//blok yg akan dijalankan kalau terjadi eksepsi yg lain
}

```

Contoh:

```

public class Bab9d {
    public static void main(String[] args) {
        System.out.println("Sebelum Pembagian");
        try {
            System.out.println(5 / 0);
        } catch (RuntimeException r) {
            System.err.println("Runtime exception");
        } catch (Exception e) {
            System.err.println("Exception");
        } catch (Throwable t) {
            System.err.println("Terjadi Pembagian dengan nol");
            System.err.println(t.getMessage());
        }
        System.out.println("Setelah Pembagian");
    }
}

```

9.5 Melontarkan Exception

Bentuk :

throw variabelobjek;

Variabelobjek merujuk ke suatu kelas eksepsi.

Contoh :

```

public class Bab9e {
    public static void main(String[] args) {
        int[] larik = new int[10];
        try {
            larik[50] = 77;
            System.out.println(larik[50]);
        } catch (ArrayIndexOutOfBoundsException a) {
            a = new ArrayIndexOutOfBoundsException("array harus berkisar
antara 0 dan 9");
            throw (a);
        }
    }
}

```

TUGAS:

1. modifikasi program (grade, class Bab4a) Anda dan tangkap error melalui try catch sehingga outputnya:

Masukan nilai Anda?

90

Sempurna!

Masukan nilai Anda?

a

Pastikan Anda memasukkan angka

2. Laporan beri penjelasan (analisa logika).

BAB 10 Array

TIK:

Mahasiswa dapat menggunakan array

Pokok Bahasan:

- Mendeklarasikan Variabel Array
- Mendefinisikan Array
- Array Dua Dimensi
- Array Multidimensi

Sub Pokok Bahasan:

Array adalah sekumpulan variabel yang memiliki tipe data yang sama dan dinyatakan dengan nama yang sama. Array merupakan konsep yang penting dalam pemrograman, karena array memungkinkan untuk menyimpan data maupun referensi objek dalam jumlah banyak dan terindeks.

Array menggunakan indeks integer untuk menentukan urutan elemen-elemennya, dimana elemen pertamanya dimulai dari indeks 0, elemen kedua memiliki indeks 1, dan seterusnya.

10.1 Mendeklarasikan Variabel Array

Mendeklarasikan variabel array dengan tipe data yang diinginkan dengan cara yang hampir sama dengan variabel biasa. Misalnya untuk mendeklarasikan variabel bertipe integer, dapat dilakukan dengan cara :

```
int[ ] bilangan; atau int bilangan[ ];
```

Jadi perbedaan utama pendeklarasian variabel array dengan variabel biasa adalah adanya tanda kurung [] di akhir tipe data atau di akhir nama variabel array. Pada tahap pendeklarasian variabel array ini belum ada alokasi memori untuk menyimpan data.

10.2 Mendefinisikan Array

Setelah mendeklarasikan array, kita perlu mendefinisikan array, dalam arti menentukan besar array yang diinginkan. Misalnya dengan cara :

```
Bilangan = new int[5];
```

Array memiliki ukuran yang tetap dalam arti tidak dapat membesar atau mengecil ukurannya setelah didefinisikan. Setelah didefinisikan, maka variabel dengan nama bilangan dapat menyimpan 5 nilai integer yang dapat diakses melalui indeks 0 sampai indeks 4. Setelah pendefinisian array, maka memori akan dialokasikan untuk menyimpan data dari array. Besar memori yang dialokasikan tergantung dari tipe data variabel array dan jumlah elemen array yang didefinisikan.

Contoh:

```
//SingleArray
class Bab10a {
    public static void main(String[] args) {
        int[] x;
        // Cara 1
        x = new int[3];
        x[0] = 20;
        x[1] = 10;
        x[2] = 30;
        System.out.println("Nilai x[0] : " + x[0]);
        System.out.println("Nilai x[1] : " + x[1]);
        System.out.println("Nilai x[2] : " + x[2]);
        int[] y = new int[3];
        // Cara 2
        y[0] = 20;
        y[1] = 10;
        y[2] = 30;
        System.out.println("Nilai y[0] : " + y[0]);
        System.out.println("Nilai y[1] : " + y[1]);
        System.out.println("Nilai y[2] : " + y[2]);
        int[] z = { 20, 10, 30 };
    }
}
```

```

// Cara 3 tdk menggunakan new
System.out.println("Nilai z[0] : " + z[0]);
System.out.println("Nilai z[1] : " + z[1]);
System.out.println("Nilai z[2] : " + z[2]);
}
}

```

10.3 Array Dua Dimensi

Pada Java juga menyediakan fasilitas untuk membuat array dua dimensi yang dapat membantu dalam pemrograman apabila array satu dimensi tidak mencukupi dalam menghasilkan suatu solusi. Array dua dimensi sebenarnya adalah array yang berisi array.

Contoh:

```

//Array2D
class Bab10b {
    public static void main(String[] args) {
        int[][] arrx;
        // Cara 1 Array 2 Dimensi
        arrx = new int[3][3];
        arrx[0][0] = 1;
        arrx[0][1] = 2;
        arrx[0][2] = 3;
        arrx[1][0] = 4;
        arrx[1][1] = 5;
        arrx[1][2] = 6;
        arrx[2][0] = 7;
        arrx[2][1] = 8;
        arrx[2][2] = 9;
        System.out.println("Nilai arrx[0] : " + arrx[0][0]);
        System.out.println("Nilai arrx[0] : " + arrx[0][1]);
        System.out.println("Nilai arrx[0] : " + arrx[0][2]);
        System.out.println("Nilai arrx[1] : " + arrx[1][0]);
        System.out.println("Nilai arrx[1] : " + arrx[1][1]);
        System.out.println("Nilai arrx[1] : " + arrx[1][2]);
        System.out.println("Nilai arrx[2] : " + arrx[2][0]);
        System.out.println("Nilai arrx[2] : " + arrx[2][1]);
        System.out.println("Nilai arrx[2] : " + arrx[2][2]);
        int[][] array = { { 10, 20, 30 }, { 40, 50, 60 }, { 70, 80, 90 } };
        // Cara 2 Array 2 Dimensi dgn ukuran 3 * 3 = 9
        System.out.println("Nilai array[0] : " + array[0][2]);
        System.out.println("Nilai array[1] : " + array[1][0]);
        System.out.println("Nilai array[1] : " + array[1][1]);
        System.out.println("Nilai array[1] : " + array[1][2]);
        System.out.println("Nilai array[2] : " + array[2][0]);
        System.out.println("Nilai array[2] : " + array[2][1]);
        System.out.println("Nilai array[2] : " + array[2][2]);
    }
}

```

10.4 Array Multidimensi

Selain array satu dimensi dan array dua dimensi, dapat juga membuat array multidimensi pada Java. Array multidimensi merupakan array yang terdiri dari array yang tidak terbatas hanya dua dimensi

saja. Kita dapat menggunakan kode berikut untuk mendapatkan array tiga dimensi :
`int[][][] array dimensi = new int[5][10][5];`

Dan pada array multidimensi, kita dapat menentukan ukuran array yang berbeda pada tiap array. Misalnya :
`int[][][] mdimensi = new int[5][][];`

Dari kode diatas, kita mendapatkan array pertama dengan 5 elemen, tetapi kita belum mendefinisikan ukuran array dimensi kedua dan ketiga.

Contoh:

```
//ArrayMultiD
class Bab10c {
    public static void main(String[] args) {
        int[][][] arr3 = { { { 10, 20, 30 }, { 40, 50, 60 } },
                          { { 11, 21, 31 }, { 41, 51, 61 } },
                          { { 12, 22, 32 }, { 42, 52, 62 } } };
        // ukuran 3 * 6 = 18
        System.out.println("Nilai arr3[0] : " + arr3[0][0][0]);
        System.out.println("Nilai arr3[0] : " + arr3[0][0][1]);
        System.out.println("Nilai arr3[0] : " + arr3[0][0][2]);
        System.out.println("Nilai arr3[0] : " + arr3[0][1][0]);
        System.out.println("Nilai arr3[0] : " + arr3[0][1][1]);
        System.out.println("Nilai arr3[0] : " + arr3[0][1][2]);
        System.out.println("Nilai arr3[1] : " + arr3[1][0][0]);
        System.out.println("Nilai arr3[1] : " + arr3[1][0][1]);
        System.out.println("Nilai arr3[1] : " + arr3[1][0][2]);
        System.out.println("Nilai arr3[1] : " + arr3[1][1][0]);
        System.out.println("Nilai arr3[1] : " + arr3[1][1][1]);
        System.out.println("Nilai arr3[1] : " + arr3[1][1][2]);
        System.out.println("Nilai arr3[2] : " + arr3[2][0][0]);
        System.out.println("Nilai arr3[2] : " + arr3[2][0][1]);
        System.out.println("Nilai arr3[2] : " + arr3[2][0][2]);
        System.out.println("Nilai arr3[2] : " + arr3[2][1][0]);
        System.out.println("Nilai arr3[2] : " + arr3[2][1][1]);
        System.out.println("Nilai arr3[2] : " + arr3[2][1][2]);
    }
}
```

TUGAS:

1. Kenapa suatu program java memerlukan Array Multidimensi,
2. berikan contohnya ?
3. Bagaimana mengetahui kapasitas atau panjang dari suatu array ?
4. dan berikan contoh programnya !
5. Laporan beri penjelasan (analisa logika).

BAB 11 Pencarian dan Pengurutan

TIK:

Mahasiswa dapat memahami algoritma search dan sort

Pokok Bahasan:

- Pencarian
- Pengurutan

Sub Pokok Bahasan:

- Quick sort
- bubble sort

11.1 Pencarian

Linear search atau sequential search adalah metode untuk menemukan nilai tertentu dalam suatu daftar dengan cara memeriksa setiap elemen yang dimiliki, satu persatu dan secara berurutan, sampai nilai yang diinginkan ditemukan.

Linear search (pencarian linear) merupakan teknik algoritma yang paling sederhana. Pada suatu daftar dengan sejumlah n item, kasus yang terbaik adalah ketika nilai samadengan elemen pertama dari daftar, dalam hal ini hanya satu perbandingan yang dibutuhkan. Kasus terburuk adalah ketika nilai tersebut tidak ada dalam daftar (atau hanya muncul sekali pada akhir daftar), dalam hal ini perbandingan n dibutuhkan.

Skenario terburuk untuk linear search adalah teknik ini harus melakukan loop pada seluruh koleksi data, baik itu ketika item data yang dicari ada pada akhir atau karena item data memang tidak ada.

Linear search tidak membutuhkan data terurut.

```
package bab11;

//package com.java2novice.algos;
//sumber http://www.java2novice.com/java-search-algorithms/linear-search/

public class MyLinearSearch {

    public static int linerSearch(int[] arr, int key) {

        int size = arr.length;
        for (int i = 0; i < size; i++) {
            if (arr[i] == key) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String a[]) {

        int[] arr1 = { 23, 45, 21, 55, 234, 1, 34, 90 };
        int searchKey = 34;
        System.out.println("Key " + searchKey + " found at index: "
            + linerSearch(arr1, searchKey));
        int[] arr2 = { 123, 445, 421, 595, 2134, 41, 304, 190 };
        searchKey = 421;
        System.out.println("Key " + searchKey + " found at index: "
            + linerSearch(arr2, searchKey));
    }
}
```

11.2 Pengurutan

Algoritma sorting merupakan algoritma yang menempatkan unsur-unsur data dalam urutan tertentu. Penyortiran yang efisien penting untuk mengoptimalkan penggunaan algoritma yang lain seperti

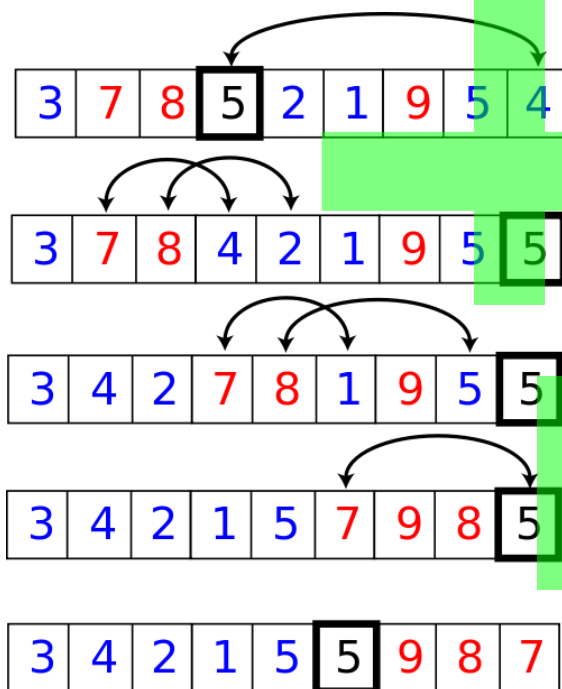
algoritma search dan merge, yang membutuhkan input data terurut.

11.2.1 Quick sort

Quicksort merupakan teknik algoritma yang cepat dengan menggunakan strategi divide and conquer. Quicksort, pertama membagi daftar data menjadi sub-sub daftar data, elemen rendah dan tinggi. Quicksort dapat secara rekursif mengurutkan sub-sub daftar.

Penerapan Quick sort:

1. Pilih suatu elemen (disebut pivot/poros tengah) dari daftar data. Umumnya pivot berada pada pertengahan elemen indeks.
2. Susun ulang daftar data sehingga semua elemen dengan nilai yang kurang dari pivot dikelompokkan sebelum pivot, sedangkan nilai yang lebih besar diletakkan setelah pivot. Operasi divide ini disebut dengan operasi partisi.
3. Secara rekursif dilakukan langkah-langkah di atas untuk sub daftar dengan nilai yang lebih kecil dan besar.



```
package bab11;

//package com.java2novice.sorting;

public class MyQuickSort {

    private int array[];
    private int length;

    public void sort(int[] inputArr) {

        if (inputArr == null || inputArr.length == 0) {
            return;
        }
    }
}
```

```

        this.array = inputArr;
        length = inputArr.length;
        quickSort(0, length - 1);
    }

    private void quickSort(int lowerIndex, int higherIndex) {
        int i = lowerIndex;
        int j = higherIndex;
        // calculate pivot number, I am taking pivot as middle index number
        int pivot = array[lowerIndex+(higherIndex-lowerIndex)/2];
        // Divide into two arrays
        while (i <= j) {
            /**
             * In each iteration, we will identify a number from left side which
             * is greater then the pivot value, and also we will identify a
number search
             * from right side which is less then the pivot value. Once the
             * is done, then we exchange both numbers.
            */
            while (array[i] < pivot) {
                i++;
            }
            while (array[j] > pivot) {
                j--;
            }
            if (i <= j) {
                exchangeNumbers(i, j);
                //move index to next position on both sides
                i++;
                j--;
            }
        }
        // call quickSort() method recursively
        if (lowerIndex < j)
            quickSort(lowerIndex, j);
        if (i < higherIndex)
            quickSort(i, higherIndex);
    }

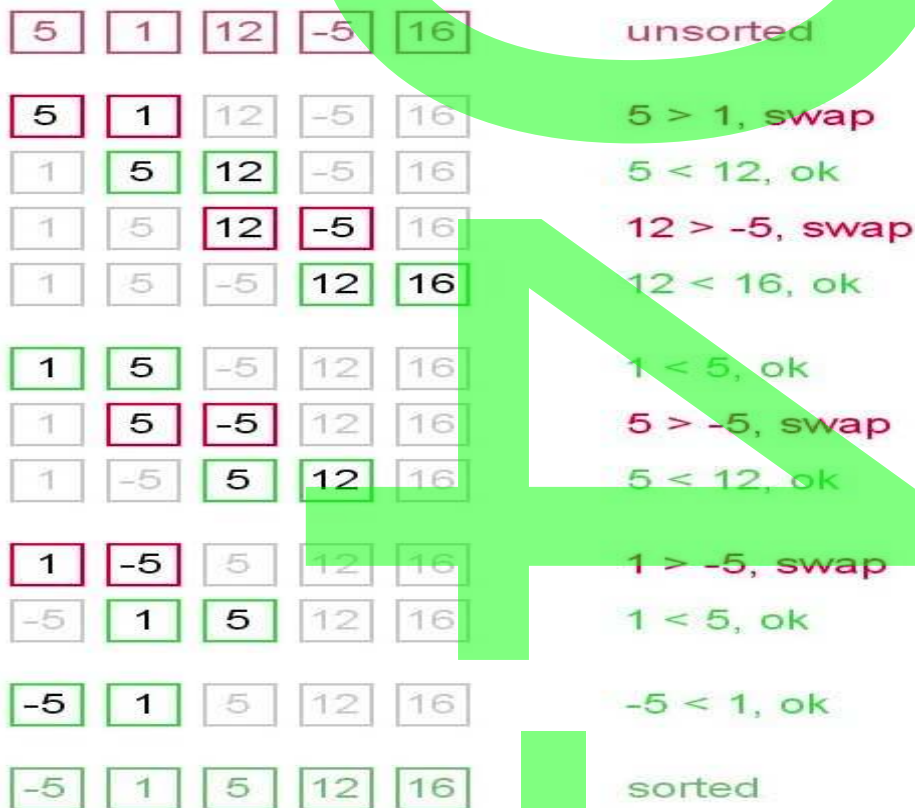
    private void exchangeNumbers(int i, int j) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    public static void main(String a[]){
        MyQuickSort sorter = new MyQuickSort();
        int[] input = {24,2,45,20,56,75,2,56,99,53,12};
        sorter.sort(input);
        for(int i:input){
            System.out.print(i);
            System.out.print(" ");
        }
    }
}

```


11.2.2 bubble sort

Bubble sort merupakan algoritma sorting sederhana yang bekerja berulang kali melalui daftar data yang disortir, dengan membandingkan setiap pasang item data yang berdekatan dan menukarkan posisi jika tidak terurut. Pengulangan dilakukan hingga proses penukaran posisi tidak diperlukan lagi.



```
package bab11;
//package com.java2novice.algos;

public class MyBubbleSort {

    // logic to sort the elements
    public static void bubble_srt(int array[]) {
        int n = array.length;
        int k;
        for (int m = n; m >= 0; m--) {
            for (int i = 0; i < n - 1; i++) {
                k = i + 1;
                if (array[i] > array[k]) {
                    swapNumbers(i, k, array);
                }
            }
            printNumbers(array);
        }
    }

    private static void swapNumbers(int i, int j, int[] array) {
```

```
        int temp;
        temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    private static void printNumbers(int[] input) {
        for (int i = 0; i < input.length; i++) {
            System.out.print(input[i] + ", ");
        }
        System.out.println("\n");
    }

    public static void main(String[] args) {
        int[] input = { 4, 2, 9, 6, 23, 12, 34, 0, 1 };
        bubble_srt(input);
    }
}
```

TUGAS:

Cari
search

Implement Binary search in java using divide and conquer technique
Implement Binary search in java using recursive algorithm.

sort

Implement selection sort in java.

Implement insertion sort in java.

Implement merge sort in java.

BAB 12 Linked List

TIK:

Mahasiswa dapat memahami bentuk struktur data linked list

Pokok Bahasan:

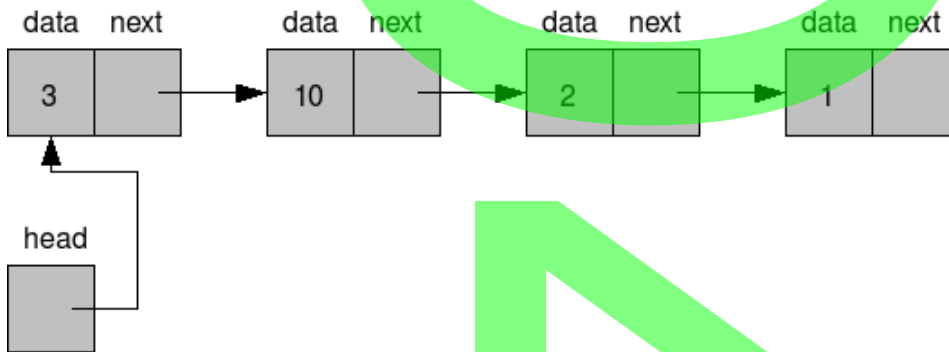
- Linked List Data Structure

Sub Pokok Bahasan:

- Singly linked list implementation
- Doubly linked list implementation

12.1 Linked List Data Structure

Linked list adalah struktur data yang terdiri dari sekelompok simpul yang bersama-sama membangun suatu urutan. Setiap node/simpul terdiri dari data dan link atau referensi ke simpul berikutnya dalam urutan tersebut. Struktur ini memungkinkan untuk penyisipan yang efisien atau penghapusan elemen pada posisi apapun dalam simpul akhir urutan. Simpul terakhir dikaitkan dengan terminasi yang digunakan sebagai tanda akhir daftar elemen.



Linked list merupakan struktur data yang paling sederhana dan umum. Linked list dapat digunakan untuk mengimplementasikan beberapa jenis data abstrak lainnya, termasuk list/daftar, stack/tumpukan, queue/antrian, array asosiatif, S-expressions, dll.

Manfaat dari linked list dibandingkan dengan array konvensional adalah bahwa elemen daftar link dengan mudah dapat dimasukkan atau dihapus tanpa realokasi atau reorganisasi seluruh struktur karena item data tidak perlu disimpan contiguous dalam memori atau pada disk. Daftar terkait memungkinkan penyisipan dan penghapusan node pada setiap titik dalam daftar.

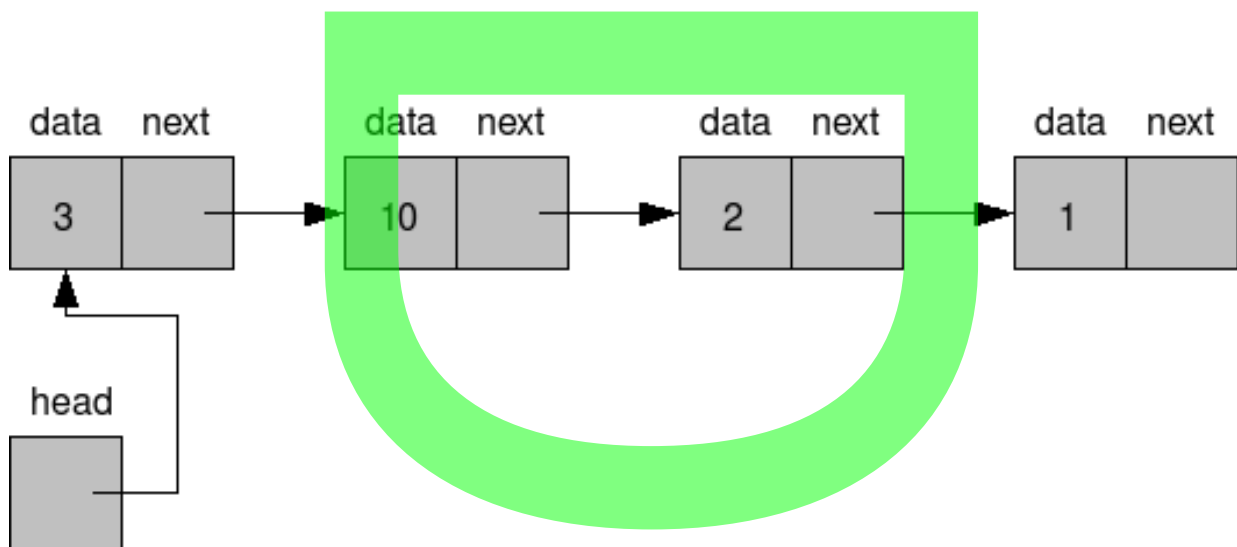
Di sisi lain, linked list sederhana tidak mengizinkan akses acak pada data, atau dengan menggunakan indeks. Dengan demikian, pencarian simpul terakhir, atau mencari simpul dengan data data yang dibutuhkan atau hendak mencari tempat untuk simpul baru diperlukan proses scanning pada daftar elemen.

12.1.1 Singly linked list implementation

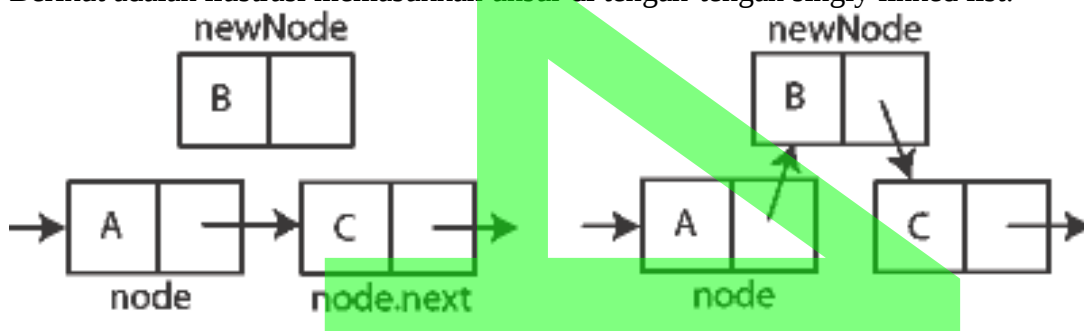
Singly Linked Lists adalah jenis struktur data dan merupakan suatu tipe pada suatu daftar. Pada singly linked list setiap simpul menyimpan isi dan sebuah penunjuk alamat (pointer) ke simpul (node) berikutnya. Pada singly linked list tidak dicatat pointer ke simpul sebelumnya.

Singly linked list, setiap node hanya memiliki satu link ke node lain. Untuk menyimpan sebuah linked list tunggal, hanya diperlukan menyimpan referensi atau pointer ke node pertama dalam daftar itu. Node terakhir memiliki pointer ke ketiadaan (null) untuk menunjukkan bahwa itu adalah node terakhir.

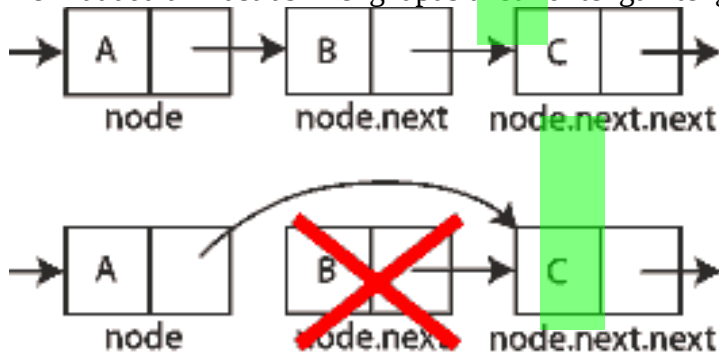
Ilustrasi singly linked list:



Berikut adalah ilustrasi memasukkan unsur di tengah-tengah singly linked list:



Berikut adalah ilustrasi menghapus unsur di tengah-tengah singly linked list:



implementasi singly linked list pada program java:

```
package bab11;

//package com.java2novice.ds.linkedlist;

public class SinglyLinkedListImpl<T> {

    private Node<T> head;
    private Node<T> tail;

    public void add(T element){

        Node<T> nd = new Node<T>();
        nd.setValue(element);
        System.out.println("Adding: "+element);
        /**
```

```

    * check if the list is empty
    */
    if(head == null){
        //since there is only one element, both head and
        //tail points to the same object.
        head = nd;
        tail = nd;
    } else {
        //set current tail next link to new node
        tail.setNextRef(nd);
        //set tail as newly created node
        tail = nd;
    }
}

public void addAfter(T element, T after){

    Node<T> tmp = head;
    Node<T> refNode = null;
    System.out.println("Traversing to all nodes..");
    /**
     * Traverse till given element
     */
    while(true){
        if(tmp == null){
            break;
        }
        if(tmp.compareTo(after) == 0){
            //found the target node, add after this node
            refNode = tmp;
            break;
        }
        tmp = tmp.getNextRef();
    }
    if(refNode != null){
        //add element after the target node
        Node<T> nd = new Node<T>();
        nd.setValue(element);
        nd.setNextRef(tmp.getNextRef());
        if(tmp == tail){
            tail = nd;
        }
        tmp.setNextRef(nd);
    }
    else {
        System.out.println("Unable to find the given element...");
    }
}

public void deleteFront(){

    if(head == null){
        System.out.println("Underflow...");
    }
    Node<T> tmp = head;
    head = tmp.getNextRef();
    if(head == null){
        tail = null;
    }
}

```

```

        System.out.println("Deleted: "+tmp.getValue());
    }

    public void deleteAfter(T after){
        Node<T> tmp = head;
        Node<T> refNode = null;
        System.out.println("Traversing to all nodes..");
        /**
         * Traverse till given element
         */
        while(true){
            if(tmp == null){
                break;
            }
            if(tmp.compareTo(after) == 0){
                //found the target node, add after this node
                refNode = tmp;
                break;
            }
            tmp = tmp.getNextRef();
        }
        if(refNode != null){
            tmp = refNode.getNextRef();
            refNode.setNextRef(tmp.getNextRef());
            if(refNode.getNextRef() == null){
                tail = refNode;
            }
            System.out.println("Deleted: "+tmp.getValue());
        } else {
            System.out.println("Unable to find the given element...");
        }
    }

    public void traverse(){
        Node<T> tmp = head;
        while(true){
            if(tmp == null){
                break;
            }
            System.out.println(tmp.getValue());
            tmp = tmp.getNextRef();
        }
    }

    public static void main(String a[]){
        SinglyLinkedListImpl<Integer> sl = new SinglyLinkedListImpl<Integer>();
        sl.add(3);
        sl.add(32);
        sl.add(54);
        sl.add(89);
        sl.addAfter(76, 54);
        sl.deleteFront();
        sl.deleteAfter(76);
        sl.traverse();
    }
}

```



```

class Node<T> implements Comparable<T> {
    private T value;
    private Node<T> nextRef;

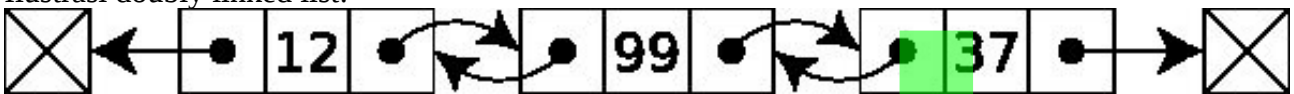
    public T getValue() {
        return value;
    }
    public void setValue(T value) {
        this.value = value;
    }
    public Node<T> getNextRef() {
        return nextRef;
    }
    public void setNextRef(Node<T> ref) {
        this.nextRef = ref;
    }
    @Override
    public int compareTo(T arg) {
        if(arg == this.value){
            return 0;
        } else {
            return 1;
        }
    }
}

```

12.1.2 Doubly linked list implementation

Doubly-linked list adalah struktur data yang terdiri dari satu set sequentially linked records yang disebut node. Setiap node berisi dua bidang, yang disebut link, yang memiliki referensi ke node sebelumnya dan ke node berikutnya dalam urutan node. Awal dan akhir node link sebelumnya dan berikutnya, masing-masing, arahkan ke beberapa jenis terminator, biasanya node sentinel atau null, untuk memfasilitasi traversal dari daftar. Jika hanya ada satu simpul sentinel, maka daftar tersebut sirkuler dihubungkan melalui node sentinel. Hal ini dapat dikonseptualisasikan sebagai dua singly linked lists terbentuk dari item data yang sama, tetapi dalam perintah berurutan berlawanan.

Ilustrasi doubly linked list:



Dua link simpul(node) memungkinkan traversal dari list di kedua arah. Sementara menambahkan atau menghapus node dalam doubly-linked list membutuhkan perubahan lebih banyak link dari operasi yang sama pada singly linked list, operasi lebih sederhana dan berpotensi lebih efisien, karena tidak ada kebutuhan untuk melacak node sebelumnya selama traversal atau tidak perlu melintasi daftar untuk menemukan node sebelumnya, sehingga link-nya dapat dimodifikasi.

Berikut adalah ilustrasi memasukkan unsur di tengah-tengah doubly linked list:

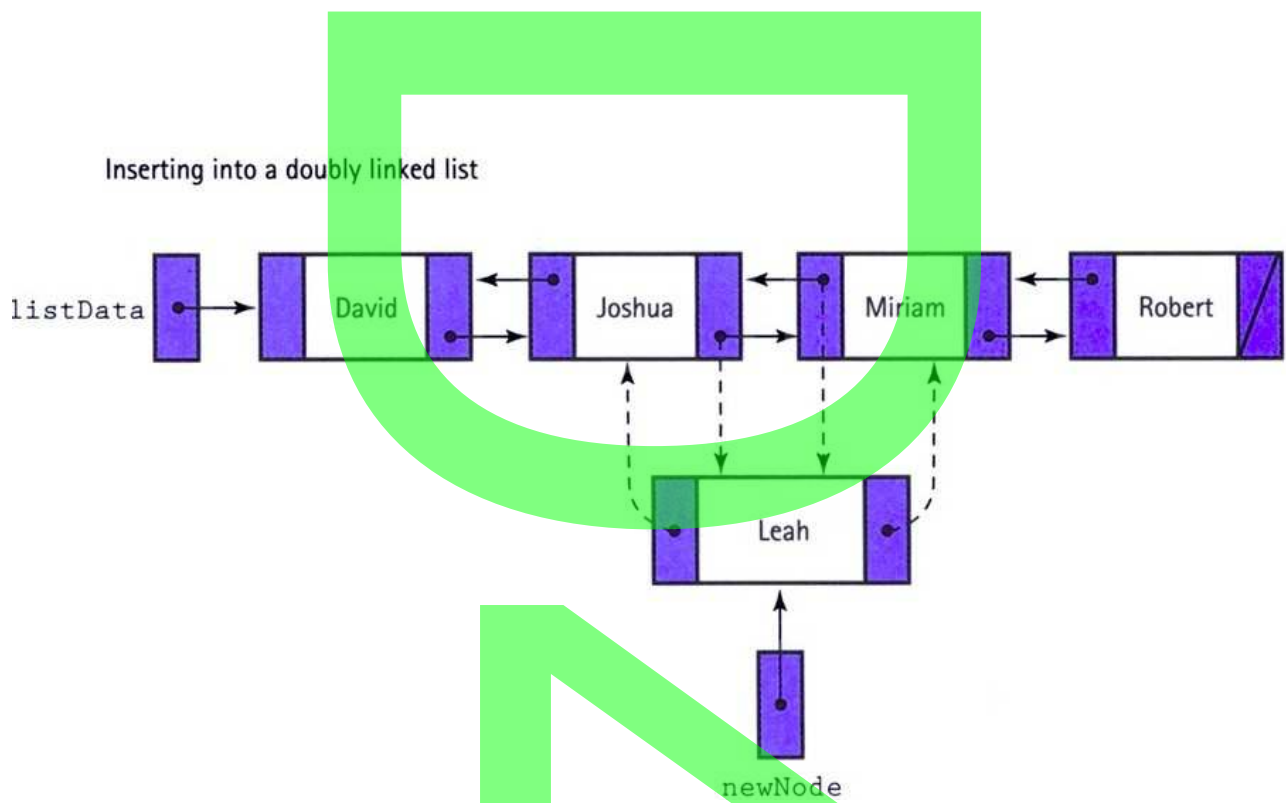


Image Reference: younginc.site11.com

Berikut adalah ilustrasi menghapus unsur di tengah-tengah doubly linked list:

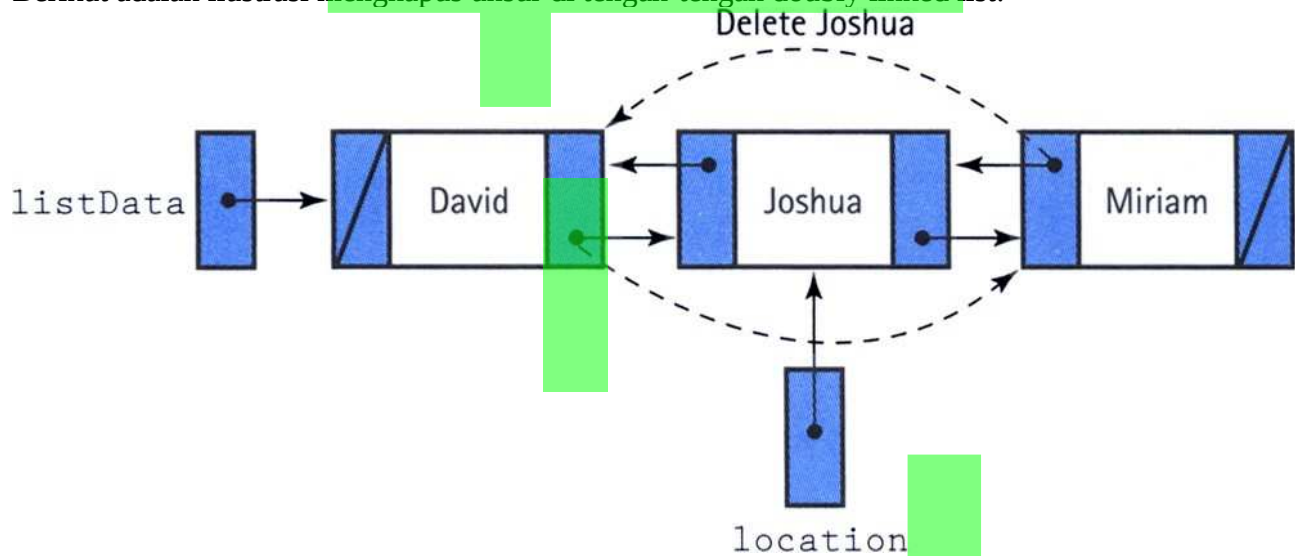


Image Reference: younginc.site11.com

implementasi doubly linked list pada program java:

```
package bab11;

//package com.java2novice.ds.linkedlist;

import java.util.NoSuchElementException;

public class DoublyLinkedListImpl<E> {

    private Node head;
```

```

private Node tail;
private int size;

public DoublyLinkedListImpl() {
    size = 0;
}
/**
 * this class keeps track of each element information
 * @author java2novice
 */
private class Node {
    E element;
    Node next;
    Node prev;

    public Node(E element, Node next, Node prev) {
        this.element = element;
        this.next = next;
        this.prev = prev;
    }
}
/**
 * returns the size of the linked list
 * @return
 */
public int size() { return size; }

/**
 * return whether the list is empty or not
 * @return
 */
public boolean isEmpty() { return size == 0; }

/**
 * adds element at the starting of the linked list
 * @param element
 */
public void addFirst(E element) {
    Node tmp = new Node(element, head, null);
    if(head != null) {head.prev = tmp;}
    head = tmp;
    if(tail == null) { tail = tmp;}
    size++;
    System.out.println("adding: "+element);
}

/**
 * adds element at the end of the linked list
 * @param element
 */
public void addLast(E element) {
    Node tmp = new Node(element, null, tail);
    if(tail != null) {tail.next = tmp;}
    tail = tmp;
    if(head == null) { head = tmp;}
    size++;
    System.out.println("adding: "+element);
}

```

```

}

/**
 * this method walks forward through the linked list
 */
public void iterateForward(){
    System.out.println("iterating forward..");
    Node tmp = head;
    while(tmp != null){
        System.out.println(tmp.element);
        tmp = tmp.next;
    }
}

/**
 * this method walks backward through the linked list
 */
public void iterateBackward(){
    System.out.println("iterating backward..");
    Node tmp = tail;
    while(tmp != null){
        System.out.println(tmp.element);
        tmp = tmp.prev;
    }
}

/**
 * this method removes element from the start of the linked list
 * @return
 */
public E removeFirst() {
    if (size == 0) throw new NoSuchElementException();
    Node tmp = head;
    head = head.next;
    head.prev = null;
    size--;
    System.out.println("deleted: "+tmp.element);
    return tmp.element;
}

/**
 * this method removes element from the end of the linked list
 * @return
 */
public E removeLast() {
    if (size == 0) throw new NoSuchElementException();
    Node tmp = tail;
    tail = tail.prev;
    tail.next = null;
    size--;
    System.out.println("deleted: "+tmp.element);
    return tmp.element;
}

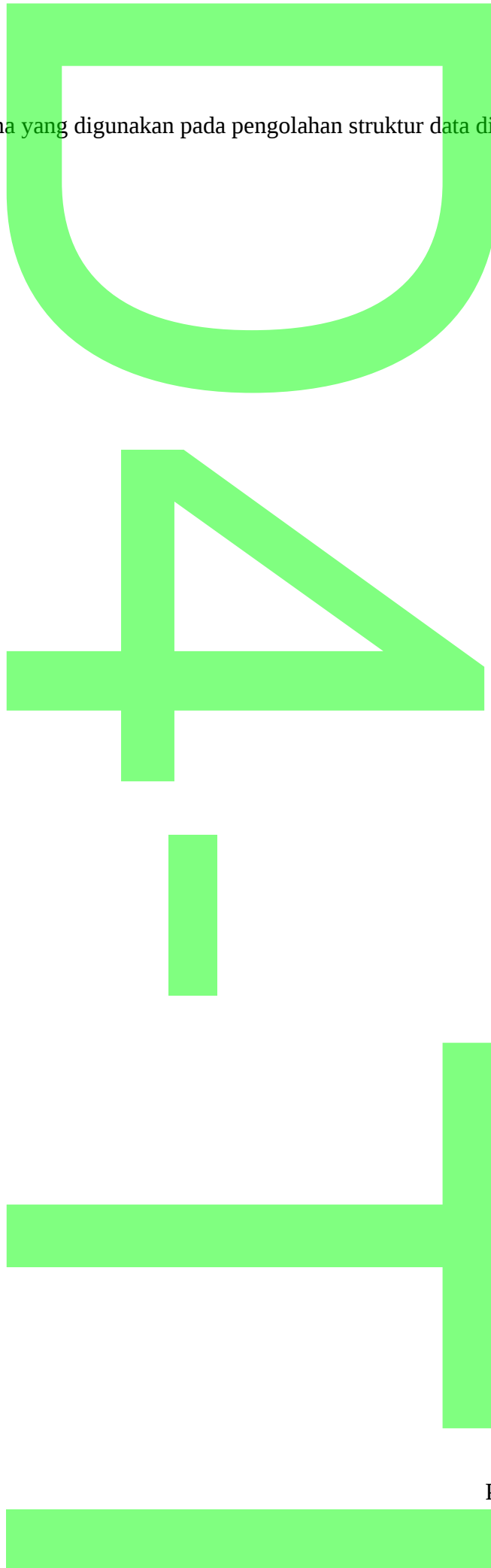
public static void main(String a[]){
    DoublyLinkedListImpl<Integer> dll = new DoublyLinkedListImpl<Integer>();

```

```
dll.addFirst(10);  
dll.addFirst(34);  
dll.addLast(56);  
dll.addLast(364);  
dll.iterateForward();  
dll.removeFirst();  
dll.removeLast();  
dll.iterateBackward();  
}  
}
```

TUGAS:

Buatlah analisa algoritma yang digunakan pada pengolahan struktur data di atas



BAB 13 Tumpukan

TIK:

Mahasiswa dapat mengenal struktur data stack

Pokok Bahasan:

- Stacks/Tumpukan

Sub Pokok Bahasan:

- Operasi Stack

13.1 Stacks/Tumpukan

Stack adalah tipe data abstrak atau koleksi. Terjadi aksi: Push dan Pop. Pada Push, penambahan elemen data untuk koleksi, dan Pop, penghapusan elemen data dari koleksi. Push dan Pop dilakukan hanya pada salah satu ujung Stack yang disebut sebagai 'tumpukan teratas' (top of the stack).

Dengan kata lain, Stack dapat hanya didefinisikan sebagai struktur data Last In First Out (LIFO), yaitu, elemen terakhir ditambahkan di bagian atas tumpukan (In) harus menjadi elemen pertama yang akan dihapus (Out) dari stack.

13.1.1 Operasi Stack:

- Push: Sebuah entitas baru dapat ditambahkan ke puncak koleksi.
- Pop: Suatu entitas akan dihapus dari atas koleksi.
- Peek / Top: Mengembalikan info entitas teratas tanpa menghapusnya.

Overflow State: Tumpukan dapat diimplementasikan untuk memiliki kapasitas terbatas. Jika stack sudah penuh dan tidak mengandung cukup ruang untuk menerima suatu entitas yang di Push, stack kemudian dianggap dalam keadaan overflow.

Underflow State: Operasi pop menghapus item dari atas tumpukan. Jika stack kosong dan dilakukan pop maka hal ini mengakibatkan underflow, yang berarti tidak ada item dalam stack untuk dihapus.

Stack adalah struktur data terbatas, karena hanya sejumlah kecil operasi yang dapat dilakukan. Operasi Push dan Pop mengikuti aturan LIFO.

Efisiensi Stacks

Dalam stack, elemen dapat di push atau pop satu per satu konstan $O(1)$ waktu. Artinya, waktu tidak tergantung pada berapa banyak item dalam stack dan karena itu stack sangat cepat. Tidak ada perbandingan atau swap yang diperlukan.

```
package bab13;

//package com.java2novice.ds.stack;

public class MyStackImpl {

    private int stackSize;
    private int[] stackArr;
    private int top;

    /**
     * constructor to create stack with size
     * @param size
     */
    public MyStackImpl(int size) {
        this.stackSize = size;
        this.stackArr = new int[stackSize];
        this.top = -1;
    }

    /**
```



```

* This method adds new entry to the top
* of the stack
* @param entry
* @throws Exception
*/
public void push(int entry) throws Exception {
    if(this.isStackFull()){
        throw new Exception("Stack is already full. Can not add element.");
    }
    System.out.println("Adding: "+entry);
    this.stackArr[++top] = entry;
}

/**
 * This method removes an entry from the
 * top of the stack.
 * @return
 * @throws Exception
 */
public int pop() throws Exception {
    if(this.isStackEmpty()){
        throw new Exception("Stack is empty. Can not remove element.");
    }
    int entry = this.stackArr[top--];
    System.out.println("Removed entry: "+entry);
    return entry;
}

/**
 * This method returns top of the stack
 * without removing it.
 * @return
 */
public int peek() {
    return stackArr[top];
}

/**
 * This method returns true if the stack is
 * empty
 * @return
 */
public boolean isEmpty() {
    return (top == -1);
}

/**
 * This method returns true if the stack is full
 * @return
 */
public boolean isStackFull() {
    return (top == stackSize - 1);
}

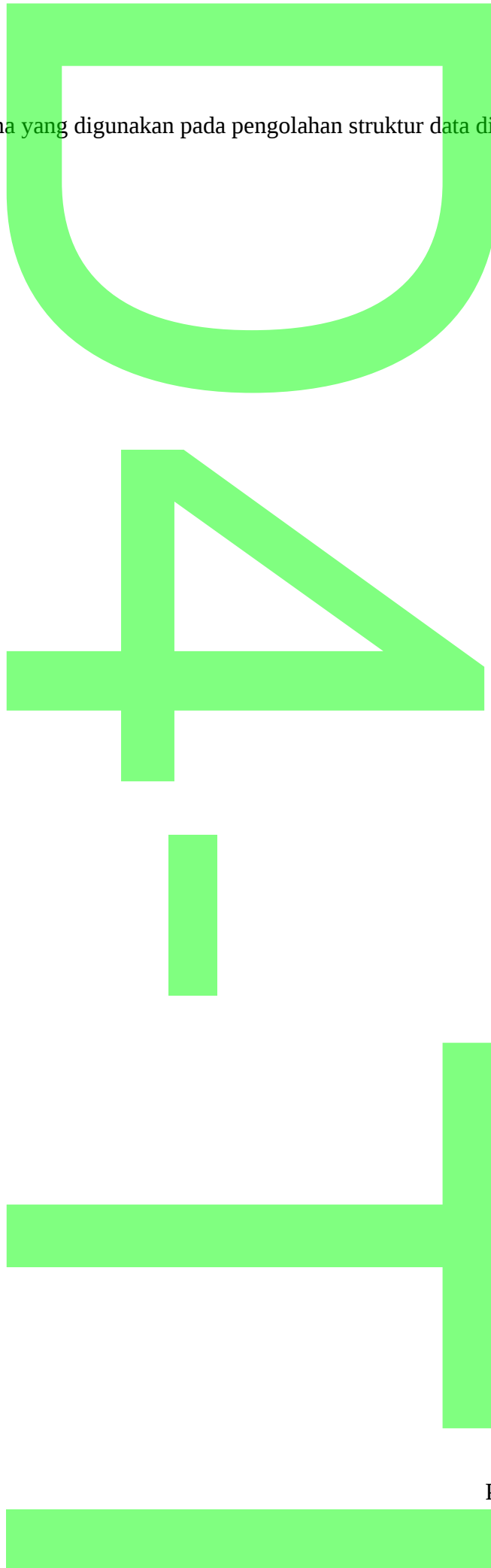
public static void main(String[] args) {
    MyStackImpl stack = new MyStackImpl(5);
    try {
        stack.push(4);
        stack.push(8);
    }
}

```

```
stack.push(3);
stack.push(89);
stack.pop();
stack.push(34);
stack.push(45);
stack.push(78);
} catch (Exception e) {
    System.out.println(e.getMessage());
}
try {
    stack.pop();
    stack.pop();
    stack.pop();
    stack.pop();
    stack.pop();
    stack.pop();
} catch (Exception e) {
    System.out.println(e.getMessage());
}
}
```

TUGAS:

Buatlah analisa algoritma yang digunakan pada pengolahan struktur data di atas



BAB 14 Antrian

TIK:

Mahasiswa dapat mengenal struktur data Queue

Pokok Bahasan:

- Antrian/Queue Data Structure

Sub Pokok Bahasan:

- Operasi pada Queue

14.1 Antrian/Queue Data Structure

Antrian adalah jenis tipe data abstrak atau koleksi di mana entitas dalam koleksi yang disimpan dalam rangka dan satu-satunya operasi pada koleksi adalah penambahan entitas ke posisi terminal belakang, disebut sebagai enqueue, dan penghapusan entitas dari posisi terminal depan, disebut sebagai dequeue.

Antrian disebut sebagai struktur data First-In-First-Out (FIFO). Dalam struktur data FIFO, elemen pertama ditambahkan ke antrian akan menjadi yang pertama untuk dihapus. Hal ini sama dengan bahwa sekali elemen baru ditambahkan, semua elemen yang sudah ditambahkan sebelumnya harus dihapus sebelum elemen baru dapat dihapus. Sebuah antrian adalah contoh dari struktur data linier.

Antrian menyediakan layanan dalam ilmu komputer, transportasi, dan riset operasi di mana berbagai entitas seperti data, benda, orang, atau peristiwa disimpan dan diadakan untuk diproses kemudian. Dalam konteks ini, antrian melakukan fungsi buffer.

14.1.1 Operasi pada Queue:

- enqueue: Menambahkan item ke akhir antrian.
- front: Mengembalikan item di depan antrian.
- dequeue: Menghapus item dari depan antrian.

Overflow State: antrian dapat diimplementasikan memiliki kapasitas terbatas. Jika antrian sudah penuh dan tidak mempunyai cukup ruang untuk menerima suatu entitas yang harus di push, antrian ini kemudian dianggap dalam keadaan overflow.

Underflow State: Operasi dequeue menghapus item dari bagian atas antrian. Sebuah operasi dequeue menampilkan item sebelumnya atau menampilkan antrian kosong. Jika antrian kosong kemudian operasi pop dilakukan hal ini menyebabkan keadaan underflow, yang berarti tidak ada item dalam antrian untuk dihapus.

Efisiensi Antrian

Waktu yang dibutuhkan untuk menambah atau menghapus item adalah konstan dan independen dari jumlah item dalam antrian.

```
package bab14;

//package com.java2novice.ds.queue;

public class QueueImpl {

    private int capacity;
    int queueArr[];
    int front = 0;
    int rear = -1;
    int currentSize = 0;

    public QueueImpl(int queueSize){
        this.capacity = queueSize;
        queueArr = new int[this.capacity];
    }
}
```

```

/**
 * this method adds element at the end of the queue.
 * @param item
 */
public void enqueue(int item) {
    if (isQueueFull()) {
        System.out.println("Overflow ! Unable to add element: "+item);
    } else {
        rear++;
        if(rear == capacity-1){
            rear = 0;
        }
        queueArr[rear] = item;
        currentSize++;
        System.out.println("Element " + item+ " is pushed to Queue !");
    }
}

/**
 * this method removes an element from the top of the queue
 */
public void dequeue() {
    if (isQueueEmpty()) {
        System.out.println("Underflow ! Unable to remove element from
Queue");
    } else {
        front++;
        if(front == capacity-1){
            System.out.println("Pop operation done ! removed:
"+queueArr[front-1]);
            front = 0;
        } else {
            System.out.println("Pop operation done ! removed:
"+queueArr[front-1]);
        }
        currentSize--;
    }
}

/**
 * This method checks whether the queue is full or not
 * @return boolean
 */
public boolean isQueueFull(){
    boolean status = false;
    if (currentSize == capacity){
        status = true;
    }
    return status;
}

/**
 * This method checks whether the queue is empty or not
 * @return
 */
public boolean isQueueEmpty(){
    boolean status = false;
    if (currentSize == 0){

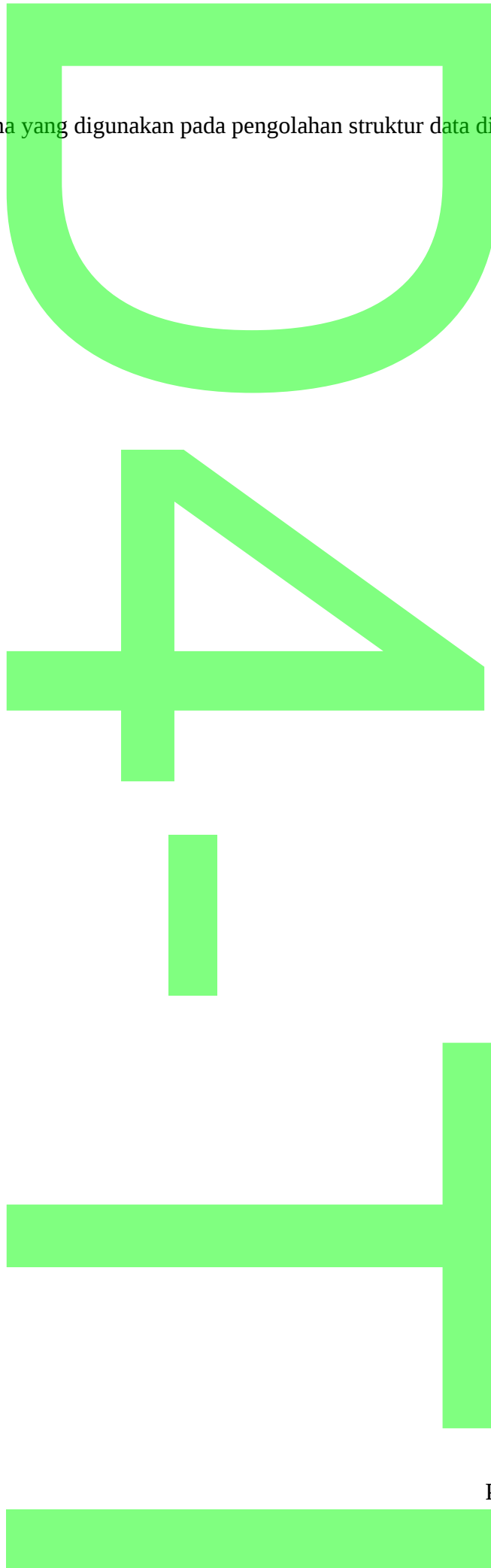
```

```
        status = true;
    }
    return status;
}

public static void main(String a[]){
    QueueImpl queue = new QueueImpl(4);
    queue.enqueue(4);
    queue.dequeue();
    queue.enqueue(56);
    queue.enqueue(2);
    queue.enqueue(67);
    queue.dequeue();
    queue.dequeue();
    queue.enqueue(24);
    queue.dequeue();
    queue.enqueue(98);
    queue.enqueue(45);
    queue.enqueue(23);
    queue.enqueue(435);
}
}
```

TUGAS:

Buatlah analisa algoritma yang digunakan pada pengolahan struktur data di atas



Pustaka

oracle

Universitas Gunadarma

Jeni

java2novice.com

tutorialspoint.com

