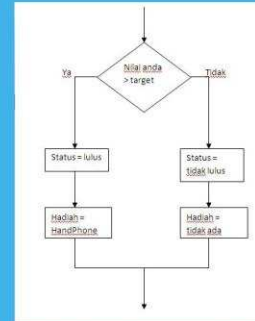


Struktur Dasar



Pengkondisian

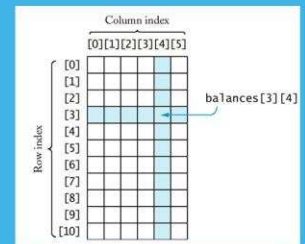
```

for (inisialisasi, kondisi, modifier)
    pernyataan;

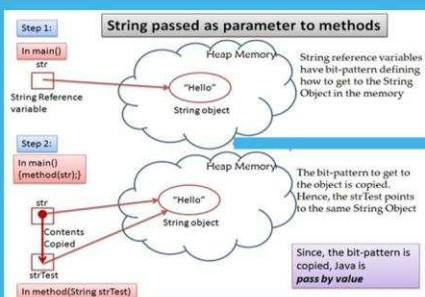
while (kondisi)
{
    pernyataan;
}

do
{
    pernyataan;
}
while (kondisi);
  
```

Perulangan



Array



Method

```

Image timer;
Thread timer;

public void init() {
    count = 0;
    lastcount = 30;
    pictures = new Image[10];
    mediaTracker tracker = new MediaTracker(this);
    for (int a = 0; a < lastcount; a++) {
        pictures[a] = getImage(a);
        tracker.addImage(pictures[a], 0);
    }
    timer = new Thread(new TimerTask());
    timer.start();
}

public void start() {
    if (timer == null) {
        timer = new Thread(new TimerTask());
        timer.start();
    }
}

public void paint(Graphics g) {
    g.drawImage(pictures[0], 0, 0, 100, 100, this);
    if (count == lastcount) {
        run();
    }
}
  
```

Implementasi

MODUL PRAKTIKUM

Algoritma Pemrograman

Tim Penyusun :

Herdianto (RDE)	116104115	Nunung Noer Hidayati (NNH)	116104117
Kholifatul Ummah (KHU)	116100075	Komang Aditya Respa. P (RPA)	116110029
Hafidh Rashemi R (HRR)	116100009	Ainu Faisal Pambudy (NOE)	116114159
Septian Hari (SHA)	116100015	Yusril Maulidan Raji (YRL)	116110025
Bambang Dwi. A (BAM)	116100018	Renantia Indriani (RNT)	116114164
Moh Fahrur Rizqon (FHR)	116100019	Fathimah Muthi Luthfiyah (FML)	116110061
Rani Auliya Syafrudin (RAS)	116100061		

Dosen Pembina :

Mardiyanto Wiyogo, ST



LABORATORIUM PROGRAMMING DAN DATABASE

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS REKAYASA INDUSTRI

INSTITUT TEKNOLOGI TELKOM 2012 / 2013

Tata Tertib, Kelengkapan dan Persyaratan Praktikum Algoritma dan Pemrograman

I. Kelengkapan dan persyaratan praktikum

1. Kelengkapan Praktikum

Kartu Praktikum

- Kartu praktikum dibagikan pada saat awal praktikum dan harus segera dilengkapi dengan data – data praktikan berikut foto dan cap laboratorium. Pemberian cap dilakukan pada saat praktikum pertama.
- Praktikan harus membawa kartu praktikum setiap mengikuti kegiatan praktikum.
- Apabila kartu praktikum hilang, praktikan harus mengganti sesuai dengan aslinya kemudian meminta cap laboratorium kepada asisten untuk legalisir , sebelum praktikum selanjutnya diadakan.

Tugas Bonus

- Tugas Bonus merupakan tugas tambahan yang bisa dikerjakan oleh praktikan sebelum pelaksanaan praktikum untuk memperoleh tambahan nilai.
- Tugas Bonus merupakan ***tugas tidak wajib***.
- Ketentuan pengerjaan Tugas Bonus akan diumumkan kemudian.

Pre Test / TP

- Pre test diadakan sebelum pelaksanaan praktikum.
- Pre test merupakan ***tugas wajib***.
- Pengumuman mengenai Pre test akan diumumkan kemudian.
- Pre test wajib dikerjakan oleh praktikan, apabila tidak mengerjakan pre test maka tidak diperbolehkan mengikuti praktikum.

Tes Awal

- Tes awal dilaksanakan sebelum pelaksanaan praktikum.
- Pengumuman mengenai tes awal akan diumumkan kemudian.

- Setiap keterlambatan, praktikan diberi kesempatan untuk mengikuti tes awal tanpa diberi perpanjangan waktu.

Praktikum

- Setiap praktikan wajib mengikuti praktikum dengan persyaratan praktikum yang telah ditentukan. Apabila salah satu atau lebih dari syarat tersebut tidak terpenuhi maka praktikan tidak diperkenankan mengikuti praktikum dan asisten berhak mengeluarkan praktikan.
- Praktikan wajib mematuhi tata tertib yang ada pada Common Lab.

Tes Akhir

- Tes akhir merupakan tes yang dilakukan pada akhir praktikum.
- Pengumuman mengenai tes akhir akan diumumkan kemudian.

2.Persyaratan Praktikum

- Memenuhi seluruh kelengkapan praktikum yang tercantum pada poin sebelumnya.

II. Tata Tertib Praktikum

1. Praktikan wajib memenuhi seluruh kelengkapan dan persyaratan Praktikum termasuk membawa kartu Praktikum.
2. Asisten dapat memberi teguran bahkan mengeluarkan praktikan yang tidak dapat menjaga ketenangan, ketertiban, kebersihan, dan kerapian laboratorium pada saat praktikum dilaksanakan.
3. Setiap praktikan wajib bertutur kata baik dan sopan dalam bersikap kepada asisten laboratorium.
4. Setiap barang yang dipinjam wajib dikembalikan ke tempat semula.
5. Tidak mengikuti praktikum ***dua modul*** atau lebih tanpa alasan yang jelas dan tidak dapat dipertanggungjawabkan maka praktikan tersebut diwajibkan mengulang praktikum pada modul bersangkutan di tahun berikutnya.
6. Tukar jadwal :
 - a. Tukar jadwal dilakukan oleh masing-masing praktikan.
 - b. Tukar jadwal dapat disetujui oleh asisten Lab apabila alasannya jelas.

7. Pengumpulan Tugas Bonus dan Tugas yang berkaitan dengan praktikum hanya dilakukan di laboratorium Prodase.
8. Seragam :
Untuk setiap kegiatan praktikum, praktikan diwajibkan untuk memakai seragam rapi dan sopan berupa kemeja putih dan memakai celana bahan warna biru gelap atau rok bahan panjang semata kaki warna biru gelap (seragam IT Telkom).
9. Sanksi Keterlambatan praktikum :
 - ❖ Terlambat 1 – 15 menit : boleh mengikuti praktikum tanpa adanya penambahan waktu.
 - ❖ Terlambat > 15 menit : tidak diperbolehkan mengikuti praktikum.
10. Selama praktikum, praktikan tidak diperkenankan meninggalkan ruangan praktikum tanpa seizin asisten jaga.
11. Alat komunikasi dinyalakan dalam mode silent atau dimatikan.
12. Pengumuman mengenai Praktikum Algoritma Pemrograman HANYA akan diinformasikan di mading Common Lab dan di web prodaslab.com.
13. Segala bentuk kecurangan akan dikenai sanksi nilai “E”.
14. Kepentingan praktikum akan dilayani pada jam kerja 08.00 – 20.00 WFLEXI.
15. Praktikum susulan:
Merupakan praktikum yang diperuntukkan bagi praktikan yang tidak mengikuti praktikum pada modul tertentu dengan alasan yang jelas dan dapat dipertanggungjawabkan.
Syarat dan ketentuan praktikum susulan:
 1. Praktikan tidak mengikuti maksimal dua (2) praktikum dengan alasan yang dapat dipertanggungjawabkan.
 2. Dilakukan pada akhir semua praktikum (setelah modul VI).
 3. Nilai yang diberikan pada praktikum susulan maksimal 90% dari nilai pada praktikum normal.
16. Hal – hal yang belum tercantum dalam peraturan ini akan ditentukan kemudian.

III. Tabel Waktu Praktikum dan Penilaian

Segi penilaian	Nilai	Waktu
Pre test / TP	10%	----
Tes awal	20%	10 menit
Praktikum	40%	60 menit
Tes akhir	30%	20 menit
Tugas Bonus	Opsional	Opsional

IV. Bentuk Penilaian

Setiap modul masing-masing memiliki point / nilai maksimal yang berbeda. Berikut tingkatan point/nilai yang terdapat dalam masing-masing modul :

Modul	Point / nilai maksimal
Pengenalan java	80
Pengkondisian	85
Perulangan	90
Array	100
Sorting & searching	115
Method	130

Dengan demikian, point maksimum praktikum Algoritma dan Pemrograman adalah 600 point dari keseluruhan modul awal hingga akhir.

Pengantar JAVA

TUJUAN PRAKTIKUM :

Praktikan dapat mengenal bentuk dasar code Java Console Application dengan sederhana, mengenal operasi dan tipe dasar.

JAVA SEKILAS PANDANG

Java merupakan bahasa pemrograman tingkat tinggi yang diciptakan berdasarkan turunan dari C++. Target utama dari penggunaan bahasa Java adalah pengkodean berarah objek yang simpel (tidak memerlukan header), menghindari manipulasi pointer secara manual (otomatis), dan lainnya. Kini, penggunaan Java sudah sangat banyak di perusahaan mengingat Java adalah *cross-platform* dan bahkan *cross-device*.

Bentuk dasar (Anatomi) Console Application :

```
public class Main {  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        System.out.println ("Give It a Try!");  
    }  
}
```

Penjelasan :

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

Awal mula kode bermula dari sini. Fungsi main() ini menandakan bahwa baris kode yang dibawah ini akan dijalankan pertama kali secara otomatis ketika program dijalankan.

```
System.out.println ("Give It a Try!");
```

Salah satu contoh baris kode yang mana ini digunakan untuk menampilkan ke console / layar. Perlu diperhatikan bahwa di bagian akhir perlu diberikan semicolon (;) untuk menandakan akhir dari satu baris kode. Serta, penulisan kode di Java membedakan huruf besar dan kecil (case-sensitive) sehingga perlu berhati-hati akan huruf besar dan kecil ketika menulis baris kode.

```
public static void main(String[] args) {  
    System.out.println ("Give It a Try!");  
}
```

Sebenarnya main() di atas adalah sebuah fungsi (penggunaannya akan dijelaskan nanti di bab selanjutnya) yang mana perlu diawali dan diakhiri oleh kurung kurawal ({ ... })

A. KOMENTAR

Komentar dalam Java berupa sebuah blok untuk menerangkan/memberikan informasi pada suatu kelas, field, method, dan lainnya agar orang lain dapat mengerti apa maksud dari kode tersebut.

Bentuk umum komentar dalam Java ada dua :

```
/* ... */
```

```
// ...
```

```
package kodingan;
```

```
public class komentar {
```

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

```
        /* menuliskan kata helloworld
```

```
        * pada console
```

```
        */
```

```
        System.out.println (args);
```

```
    }
```

```
}
```

Contoh komentar

Komentar **TIDAK AKAN** ikut dikompilasi menjadi program sehingga programmer tidak perlu khawatir programnya akan membengkak karena terlalu banyak menuliskan penjelasan (komentar).

B. INPUT / OUTPUT

Dalam java dikenal juga dengan proses input dan proses output. Berikut adalah contoh program

```
/* Program untuk proses input dan output */
import java.util.Scanner;

public class Praktikum {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("nilai a adalah ");
        //input
        int a = input.nextInt();
        //output
        System.out.println(a);
    }
}
```

untuk input dan output :

Penjelasan :

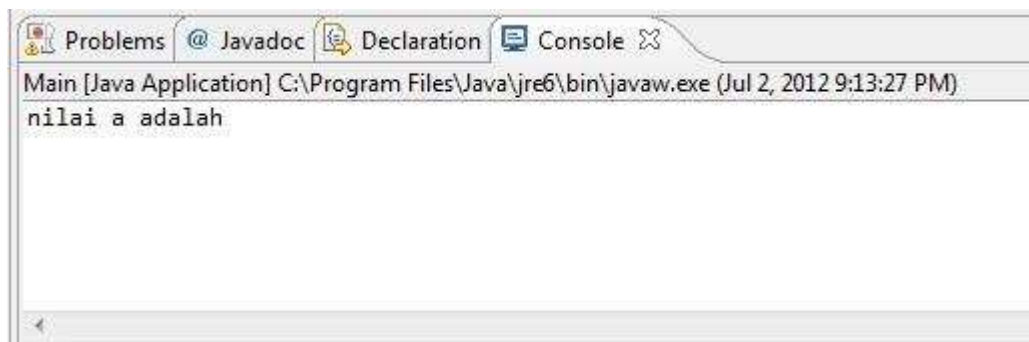
Baris kode berikut adalah baris kode yang digunakan untuk proses input atau proses pemberian nilai pada sebuah variabel (akan dijelaskan pada pembahasan berikutnya).

```
Scanner input = new Scanner(System.in);
```

Perlu diperhatikan bahwa, ketika Scanner digunakan. Pada class yang sama juga harus terdapat baris kode

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

Baris tersebut berfungsi agar Scanner dapat didefinisikan pada class tersebut. Maka ketika program dijalankan, program akan menampilkan sebagai berikut



Ketika kursor berada pada console/layar, maka kita sudah dapat mengetikkan suatu nilai, dan nilai tersebut akan menjadi nilai dari variabel a.

C. TYPE DATA

Tipe data adalah klasifikasi antar data. Tujuannya adalah mencegah tercampurnya data lain yang memiliki “bentuk” yang berbeda. Tipe data tersebut dapat disimpan dalam sebuah wadah yang disebut **variable**. Contoh pembuatan variabel :

```
import java.util.Scanner;

public class Praktikum {
    /* Tes membuat variabel */

    public static void main(String[] args) {
        String h = "Hello!";
        System.out.println(h);
    }
    // maka di layar akan menuliskan tulisan Hello!
}
```

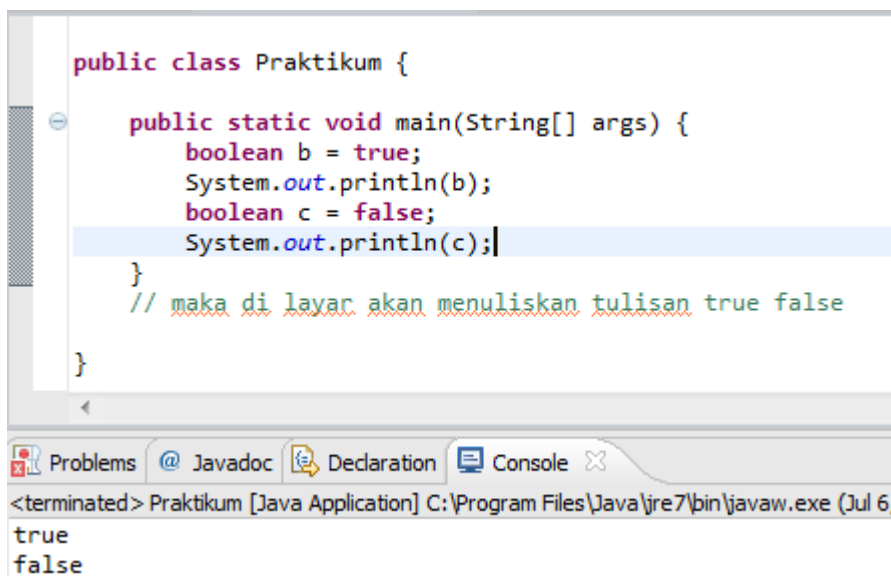
Berikut adalah tipe data yang dikenal di Java :

1. Boolean

Boolean adalah tipe data yang hanya menyatakan kondisi **true** (benar) dan **false** (salah).

Boolean pada dasarnya adalah representasi dari 1 (true) dan 0 (false).

Contoh :



```
public class Praktikum {

    public static void main(String[] args) {
        boolean b = true;
        System.out.println(b);
        boolean c = false;
        System.out.println(c);
    }
    // maka di layar akan menuliskan tulisan true false
}
```

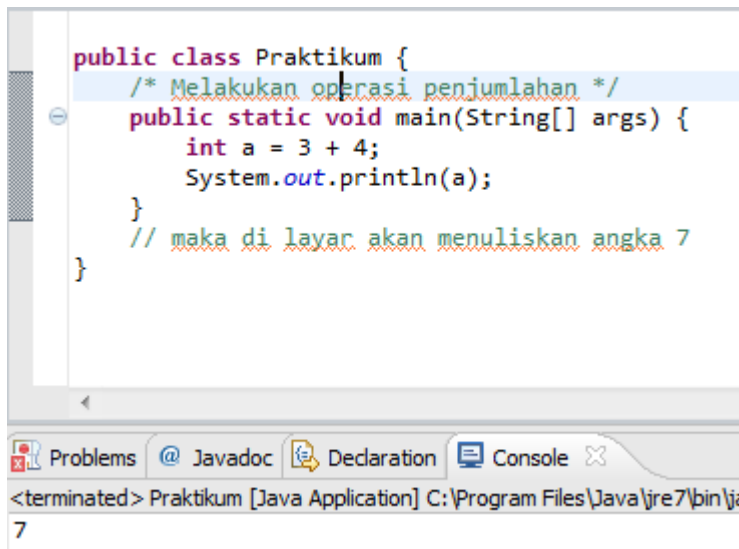
The screenshot shows an IDE window with the above code. Below the code editor, the console output is visible, showing the results of the program execution.

```
<terminated> Praktikum [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jul 6,
true
false
```

2. Integer

Integer merupakan tipe data numerik yang bulat dan dapat dilakukan proses aritmatika. Adapun tipe data yang sejenis adalah Byte, Long, Short. Perbedaannya adalah besaran bit yang dipakai.

Contoh :



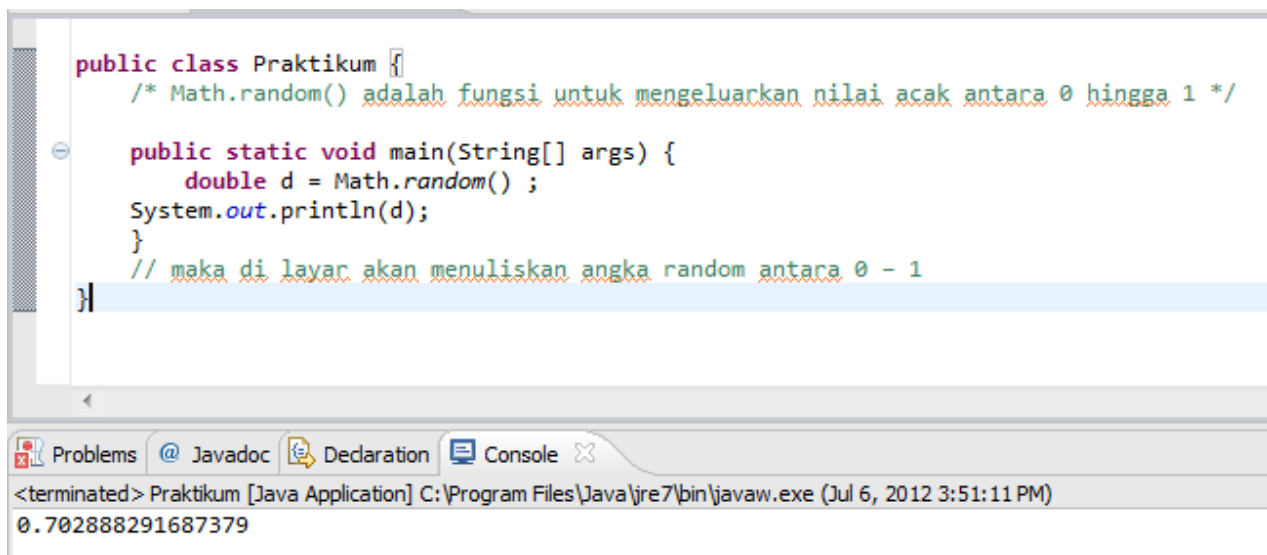
```
public class Praktikum {  
    /* Melakukan operasi penjumlahan */  
    public static void main(String[] args) {  
        int a = 3 + 4;  
        System.out.println(a);  
    }  
    // maka di layar akan menuliskan angka 7  
}
```

The screenshot shows an IDE window with tabs for Problems, Javadoc, Declaration, and Console. The Console output shows the program terminated and the number 7 printed.

3. Float

Type float digunakan untuk menangani bilangan pecahan. Tipe data jenis adalah Double untuk angka yang lebih besar dan presisi lebih tinggi.

Contoh :



```
public class Praktikum {  
    /* Math.random() adalah fungsi untuk mengeluarkan nilai acak antara 0 hingga 1 */  
    public static void main(String[] args) {  
        double d = Math.random();  
        System.out.println(d);  
    }  
    // maka di layar akan menuliskan angka random antara 0 - 1  
}
```

The screenshot shows an IDE window with tabs for Problems, Javadoc, Declaration, and Console. The Console output shows the program terminated and a random double value (0.702888291687379) printed.

4. Karakter

Char adalah tipe data untuk karakter. Pada dasarnya karakter memiliki nomor indeks yang biasa disebut ASCII code yang direpresentasikan sebagai angka. Sebagai contoh huruf 'd' memiliki nilai 100. Char dapat dilakukan operasi matematika seperti layaknya integer.

Contoh :

```
public class Praktikum {
    /* Melakukan operasi + 1 pada karakter */

    public static void main(String[] args) {
        char b = 'd' + 1;
        System.out.println(b);
    }
    // maka di layar akan menuliskan huruf e. BUKAN d+1 atau 5
}

Problems @ Javadoc Declaration Console
<terminated> Praktikum [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jul 6, 2012 3:53:26 PM)
e
```

5. String

String adalah tipe data yang dapat menyimpan sederet karakter menjadi satu seperti layaknya kalimat atau kata. Pada dasarnya string adalah sejenis array (sekumpulan) dari char yang dimanipulasi sehingga menjadi tipe data baru. Dibandingkan tipe data yang lain, tipe data string memiliki fungsi pemanipulasian paling banyak.

Contoh :

```
import java.util.Scanner;

public class Praktikum {
    /* Tes membuat variabel */

    public static void main(String[] args) {
        String h = "Hello!";
        System.out.println(h);
    }
    // maka di layar akan menuliskan tulisan Hello!
}
```

D. OPERATOR PADA JAVA

Operator merupakan tanda yang digunakan untuk melakukan suatu operasi.

Ada beberapa jenis operator yang dapat digunakan, yaitu :

- Operator assignment
- Operator aritmatik
- Operator perbandingan
- Operator logika

1. OPERATOR ASSIGNMENT

Operator assignment adalah operator yang melakukan pengisian nilai kepada suatu variabel sehingga variabel yang telah dibuat jadi menyimpan suatu nilai.

Contoh :

```
public class Praktikum {
    /* Melakukan operasi + 1 pada karakter */

    public static void main(String[] args) {
        double d;
        d = 3.14;
        d = d + 3.12; // Menambah 3.14 dengan 3.12 lalu dimasukkan ke d
        System.out.println(d);
    }
    // maka di layar akan menuliskan angka 6.26
}
```

Problems Javadoc Declaration Console

<terminated> Praktikum [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Jul 6, 2012 3:55:24 PM)
6.26

2. OPERATOR ARITMETIK ()

Ada beberapa operator aritmetik yang sudah kita kenal, yaitu

- Penjumlahan (+)
- Pengurangan (-)
- Pembagian (/)
- Perkalian (*)
- Modulus (sisa pembagian %)

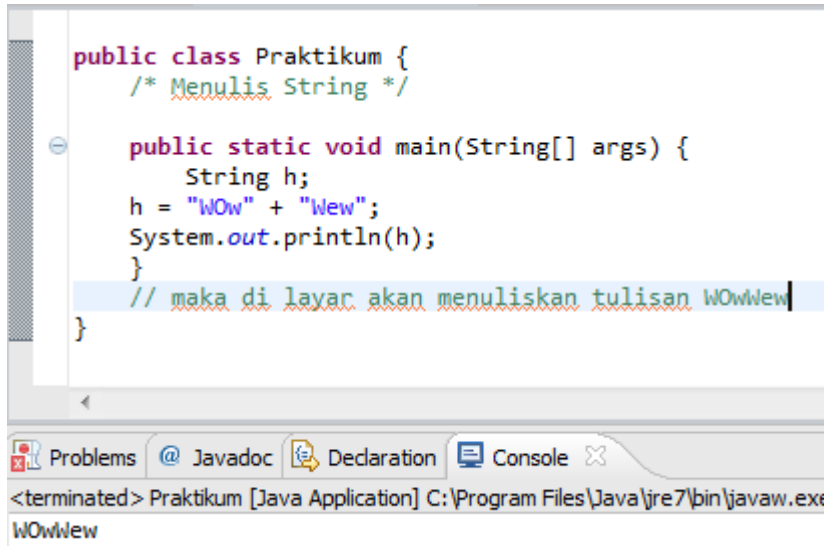
Berikut ini adalah contoh penggunaan operator aritmetik.

```
package kodingan;

public class aritmetik {
    public static void main (String[] args) {
        System.out.println("penambahan 3 + 5 = " + (3 + 5));
        System.out.println("pengurangan 5 - 3 = " + (5 - 3));
        System.out.println("perkalian 5 * 5 = " + (5 * 5));
        System.out.println("pembagian 90 / 5 = " + (90/5));
        System.out.println("modulus 52 % 6 = " + (52 % 6));
    }
}
```

Untuk beberapa kasus seperti tipe data lain, operator bisa jadi bermakna lain. Seperti pada String, dapat dilakukan operator + untuk menggabungkan string pertama dan kedua dan selanjutnya, namun tidak dapat dioperasikan -, /, * atau %.

Contoh :



```
public class Praktikum {
    /* Menulis String */

    public static void main(String[] args) {
        String h;
        h = "WOW" + "Wew";
        System.out.println(h);
    }
    // maka di layar akan menuliskan tulisan WOWWew
}
```

Problems Javadoc Declaration Console

<terminated> Praktikum [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe
WOWWew

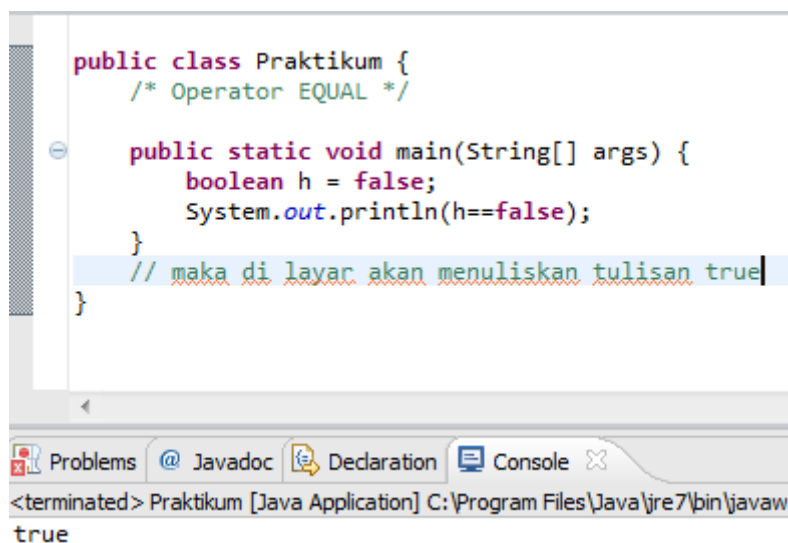
3. OPERATOR LOGIKA

Operator logika digunakan untuk melakukan operasi dan komparasi dalam nilai Boolean. Ada beberapa operator yang digunakan untuk operasi boolean, yaitu,

- **Operator == (EQUAL)**

Operator == digunakan untuk menyatakan apakah nilai di ruas kiri sama dengan ruas kanan. Mengembalikan nilai true apabila ya dan false apabila tidak.

Contoh :



```
public class Praktikum {
    /* Operator EQUAL */

    public static void main(String[] args) {
        boolean h = false;
        System.out.println(h==false);
    }
    // maka di layar akan menuliskan tulisan true
}
```

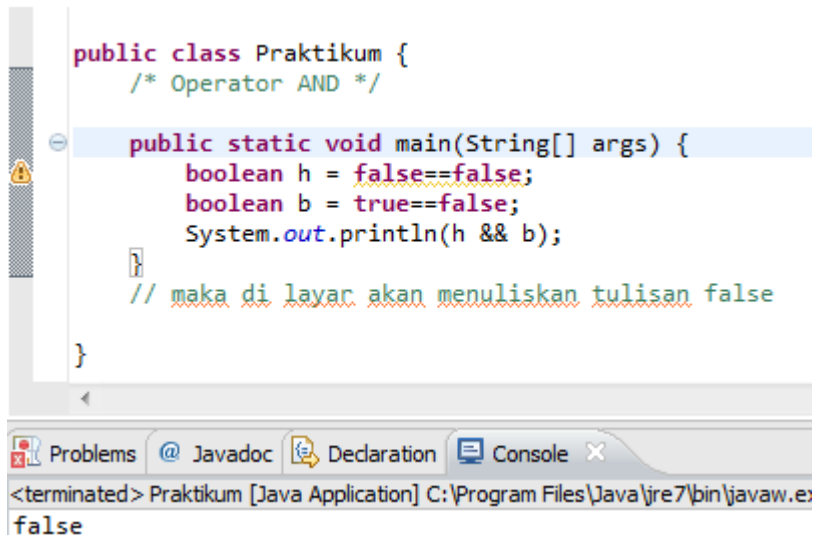
Problems Javadoc Declaration Console

<terminated> Praktikum [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe
true

- **Operator && (AND)**

Operator && (AND) sifatnya adalah konjungsi (dan), dimana mengembalikan nilai true apabila ruas kiri dan kanan sama-sama memiliki nilai true, selain itu akan dianggap salah.

Contoh :



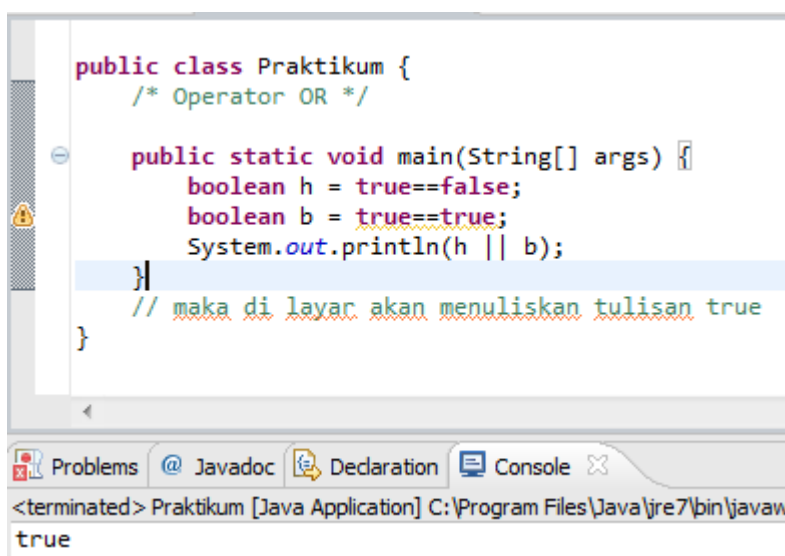
```
public class Praktikum {  
    /* Operator AND */  
  
    public static void main(String[] args) {  
        boolean h = false==false;  
        boolean b = true==false;  
        System.out.println(h && b);  
    }  
    // maka di layar akan menuliskan tulisan false  
}
```

Problems Javadoc Declaration Console
<terminated> Praktikum [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe
false

- **Operator || (OR)**

Operator || (OR) sifatnya adalah disjungsi (atau), dimana mengembalikan nilai true apabila antara ruas kiri atau kanan memiliki nilai true, salah satu atau keduanya. Mengembalikan nilai false apabila keduanya memiliki nilai false.

Contoh :



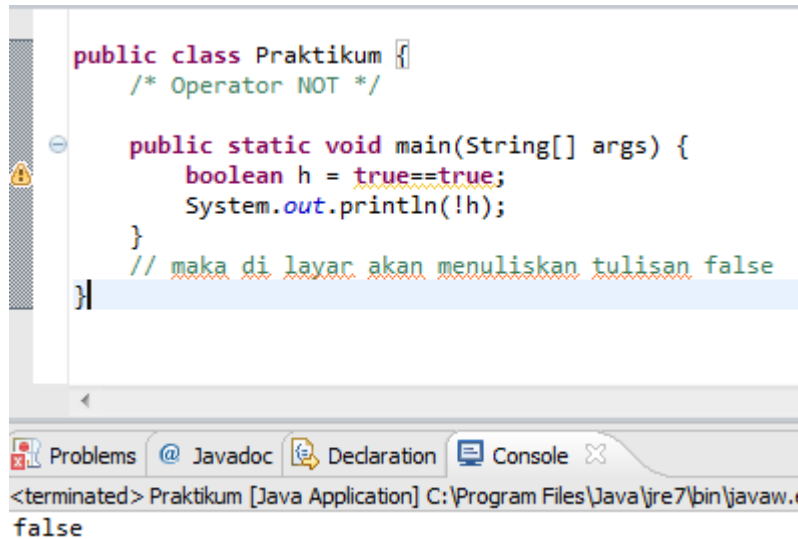
```
public class Praktikum {  
    /* Operator OR */  
  
    public static void main(String[] args) {  
        boolean h = true==false;  
        boolean b = true==true;  
        System.out.println(h || b);  
    }  
    // maka di layar akan menuliskan tulisan true  
}
```

Problems Javadoc Declaration Console
<terminated> Praktikum [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe
true

- **Operator! (NOT)**

Operator ! (NOT) akan membalikkan boolean yang dijadikan operan. Apabila nilai operan adalah true, maka hasilnya akan menjadi false.

Contoh :



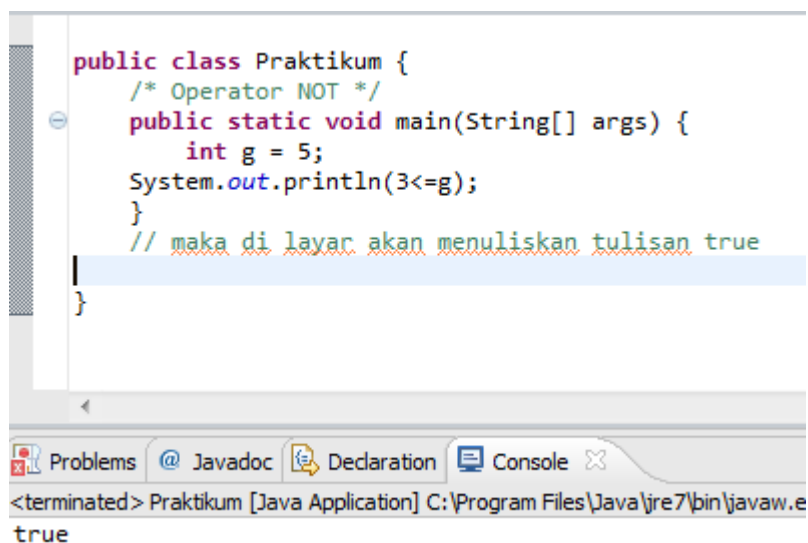
```
public class Praktikum {  
    /* Operator NOT */  
  
    public static void main(String[] args) {  
        boolean h = true==true;  
        System.out.println(!h);  
    }  
    // maka di layar akan menuliskan tulisan false  
}
```

<terminated> Praktikum [Java Application] C:\Program Files\Java\jre7\bin\javaw.e
false

4. OPERATOR LOGIKA (NUMERIC)

Untuk angka, dapat juga dilakukan hal-hal yang seperti kita lakukan dulu di logika matematika SMA. Yaitu (==) untuk sama dengan, (<) untuk kurang dari, (>) untuk lebih dari, (<=) untuk kurang dari atau sama dengan, (>=) untuk lebih dari atau sama dengan.

Contoh :



```
public class Praktikum {  
    /* Operator NOT */  
    public static void main(String[] args) {  
        int g = 5;  
        System.out.println(3<=g);  
    }  
    // maka di layar akan menuliskan tulisan true  
}
```

<terminated> Praktikum [Java Application] C:\Program Files\Java\jre7\bin\javaw.e
true

E. TOOLS YANG DIGUNAKAN

1. ECLIPSE

Dalam praktikum ini, tools yang digunakan adalah ECLIPSE. Eclipse merupakan sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan disemua platform.

2. PENGUNAAN ECLIPSE

Dalam menggunakan eclipse, ada beberapa langkah yang harus dilakukan. Berikut adalah langkah – langkah yang harus dilakukan, yaitu

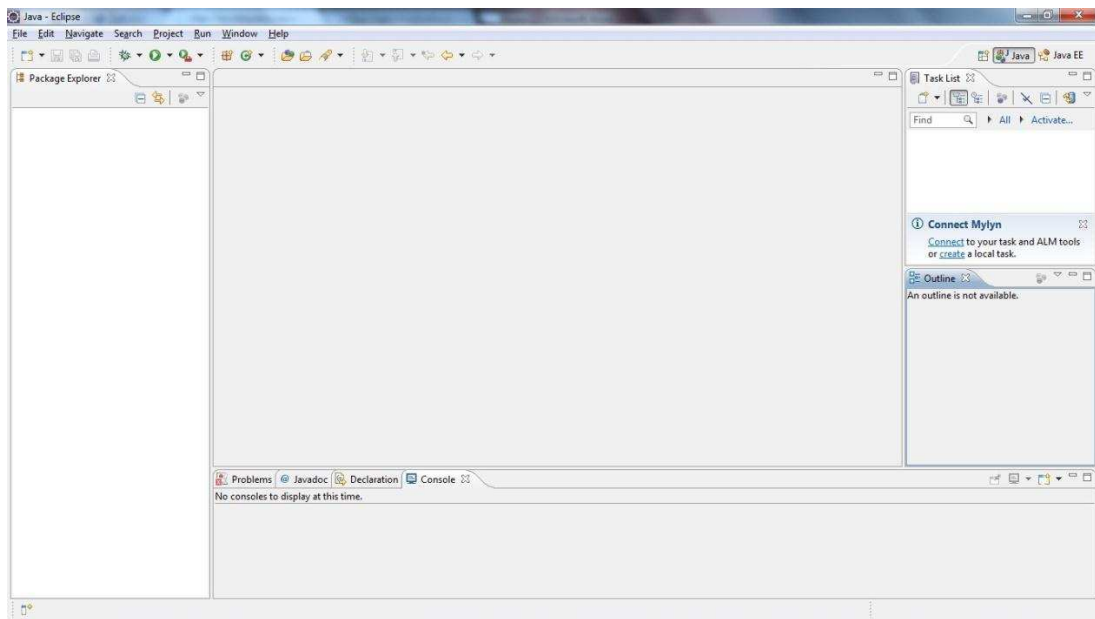
- a. Membuka eclipse, klik icon seperti gambar dibawah ini untuk membuka eclipse



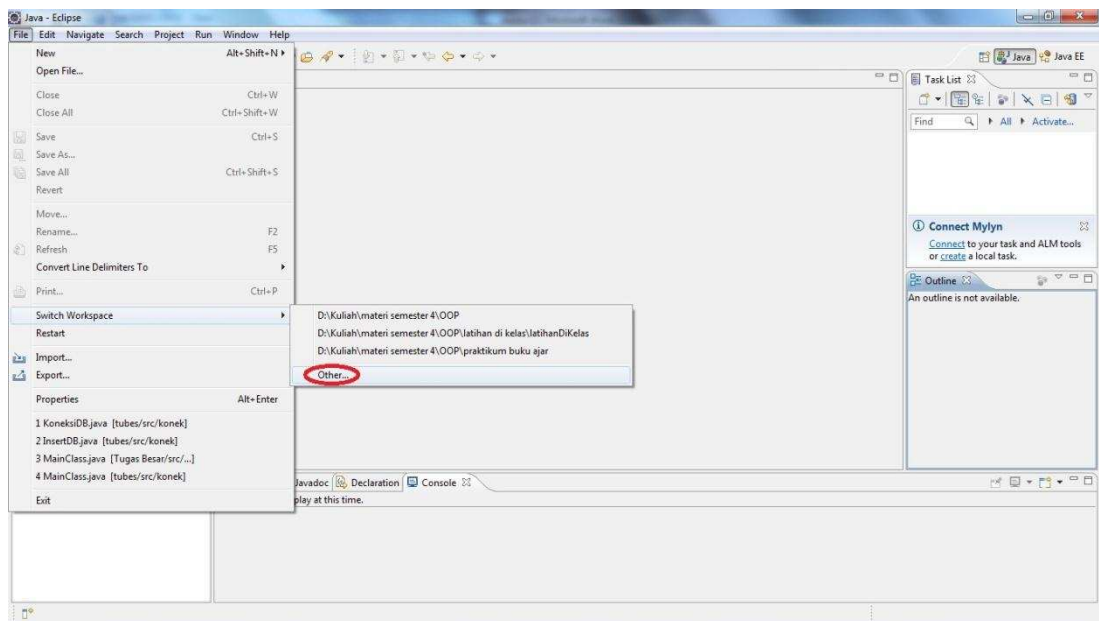
- b. Setelah klik gambar seperti itu, eclipse akan membuka workspace yang ditandai dengan proses seperti gambar dibawah ini



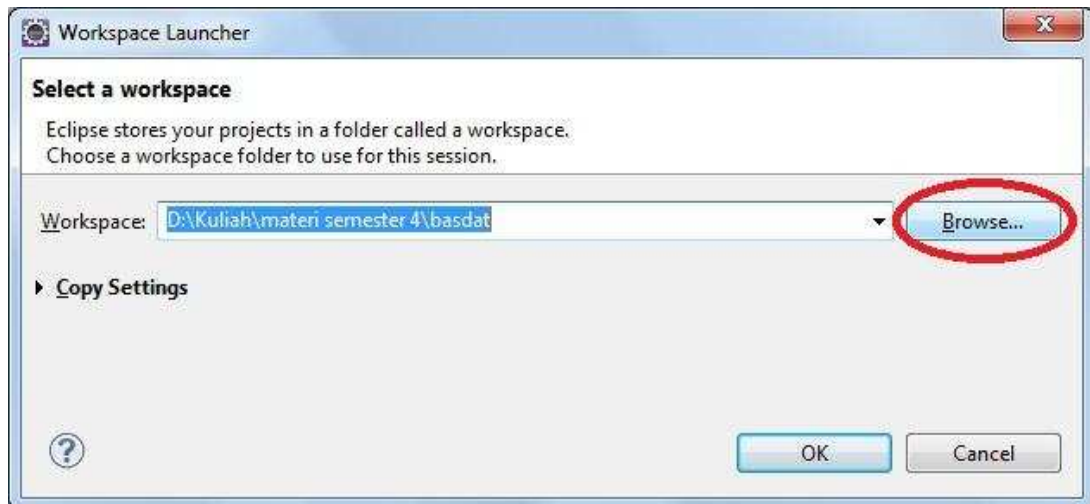
c. setelah itu akan muncul tampilan seperti dibawah ini



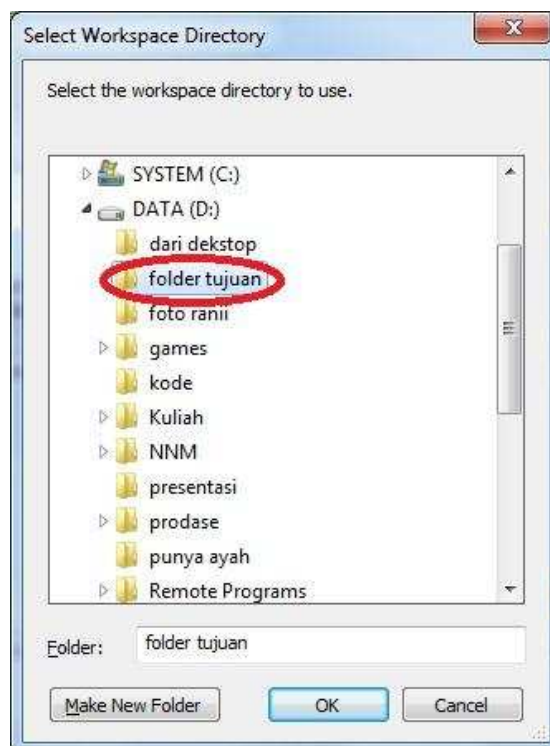
d. Untuk menyimpan project yang akan dibuat di folder yang diinginkan, maka pilih **FILE -> Switch Workspace -> other**



- e. Lalu akan muncul tabel dialog seperti gambar dibawah ini, pilih browse untuk memilih folder yang diinginkan.



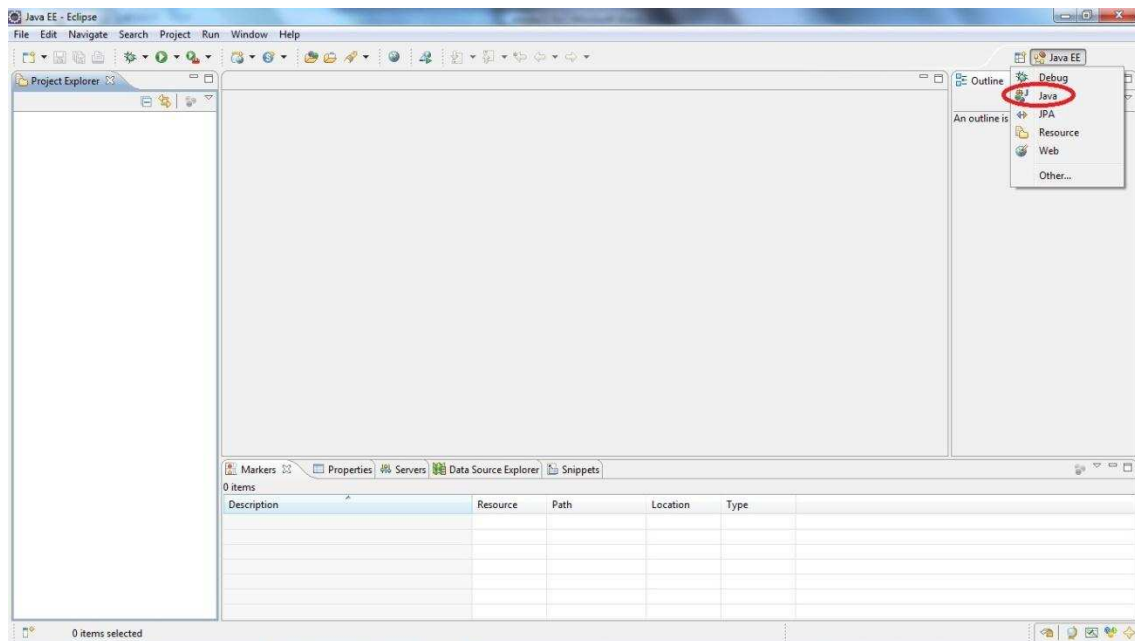
- f. lalu ketika tabel dialog seperti berikut muncul pilih folder yang anda inginkan



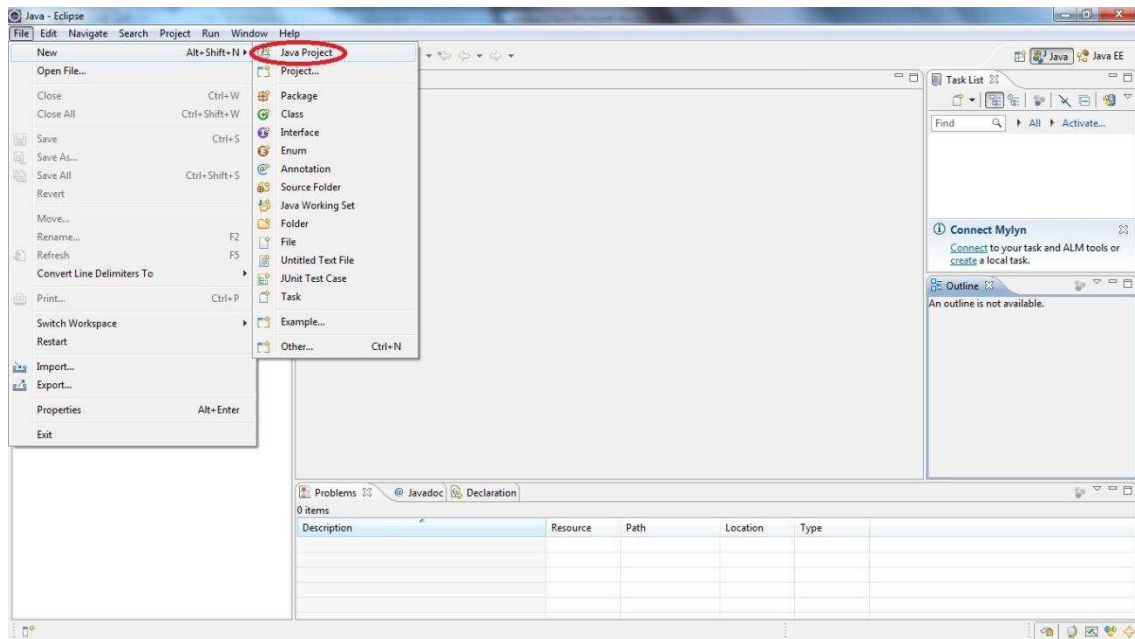
- g. setelah itu eclipse akan melakukan restart, lalu mncul tampilan sebagai berikut, pilih workbench



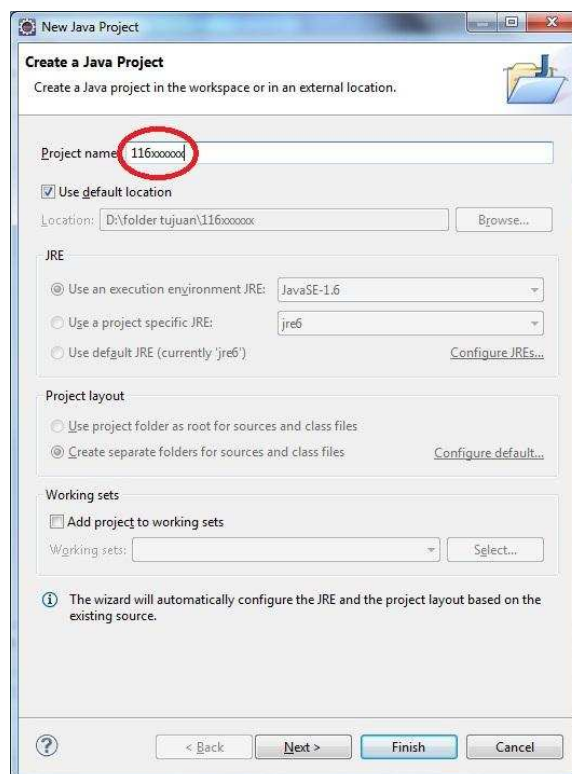
- h. lalu ubah dari java EE menjadi java pada pilihan sebagai berikut



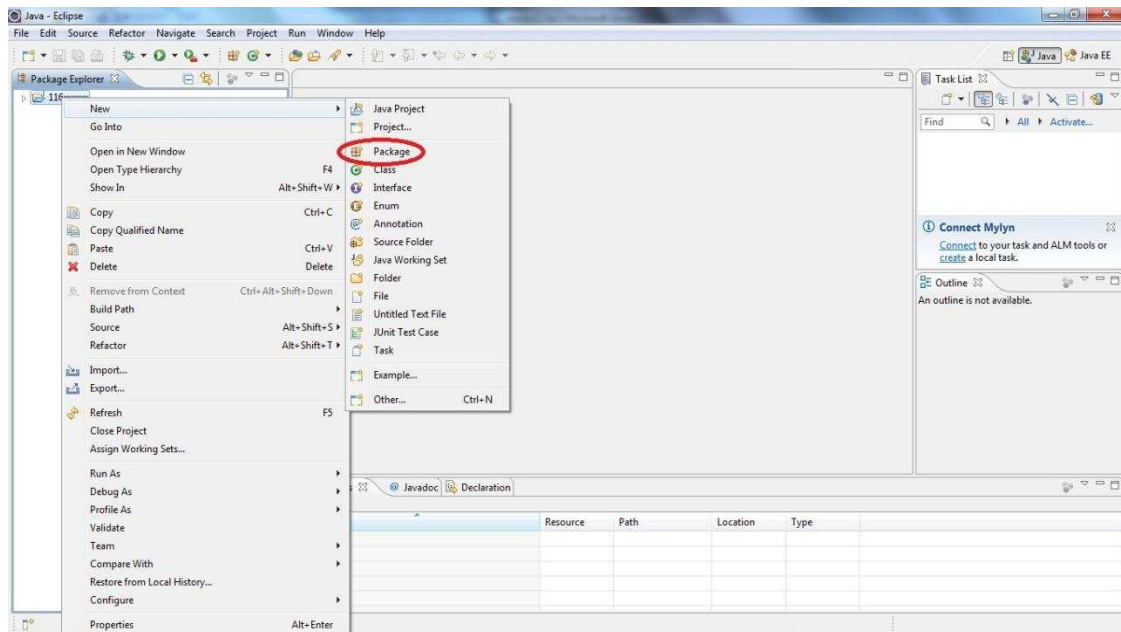
- i. Setelah itu, kita akan memulai pekerjaan kita dengan membuat project, pilih **FILE -> New -> Java Project**.



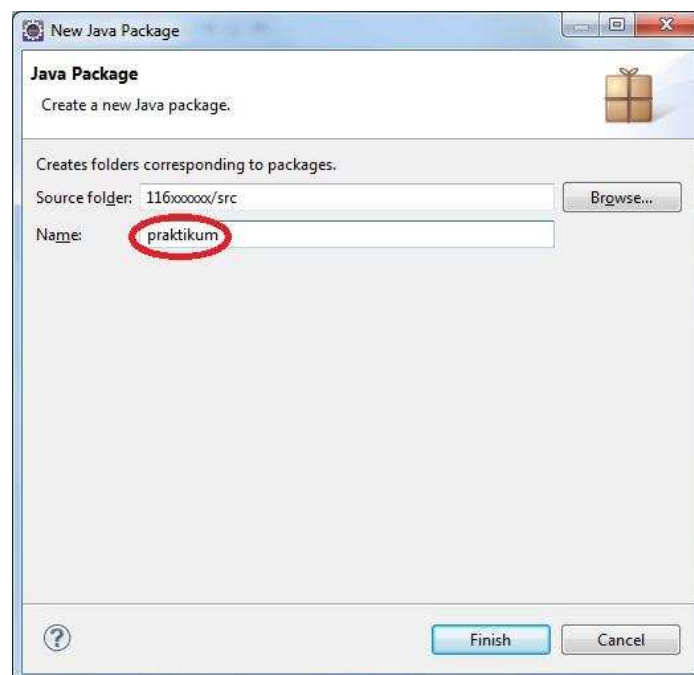
- j. Pada tabel dialog berikut, ketik nama project yang diinginkan, misal nim anda lalu klik finish.



- k. Setelah itu project yang anda buat akan terlihat pada package explorer. Lalu kita buat package, klik kanan pada project yang dibuat lalu pilih **New -> Package**.

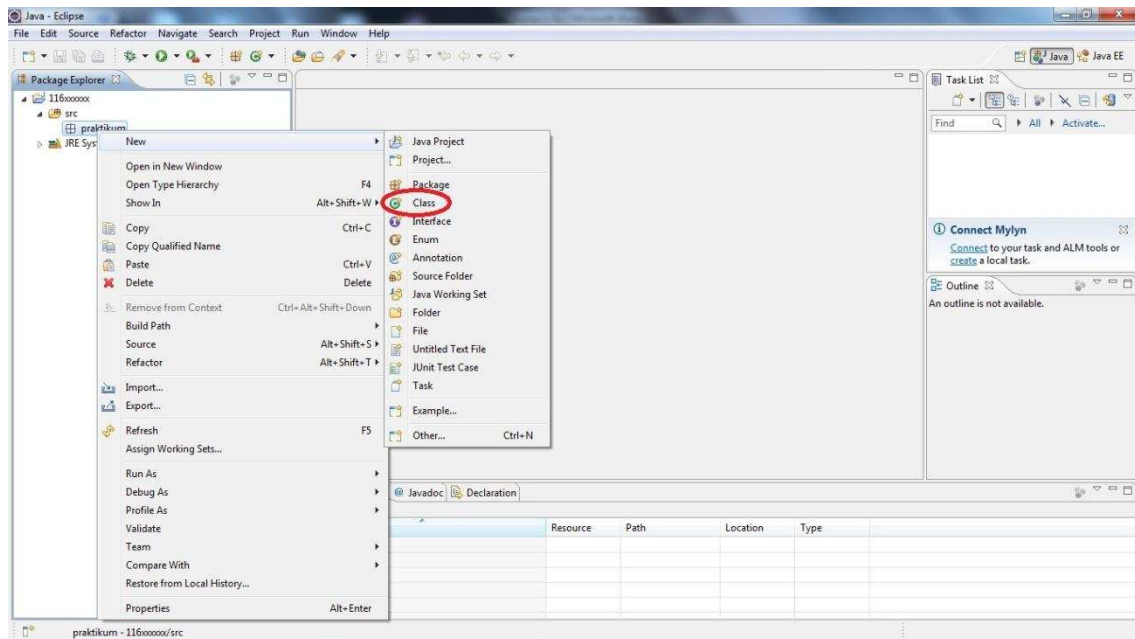


- l. Ketikkan nama package yang akan dibuat pada tabel dialog berikut lalu klik finish.



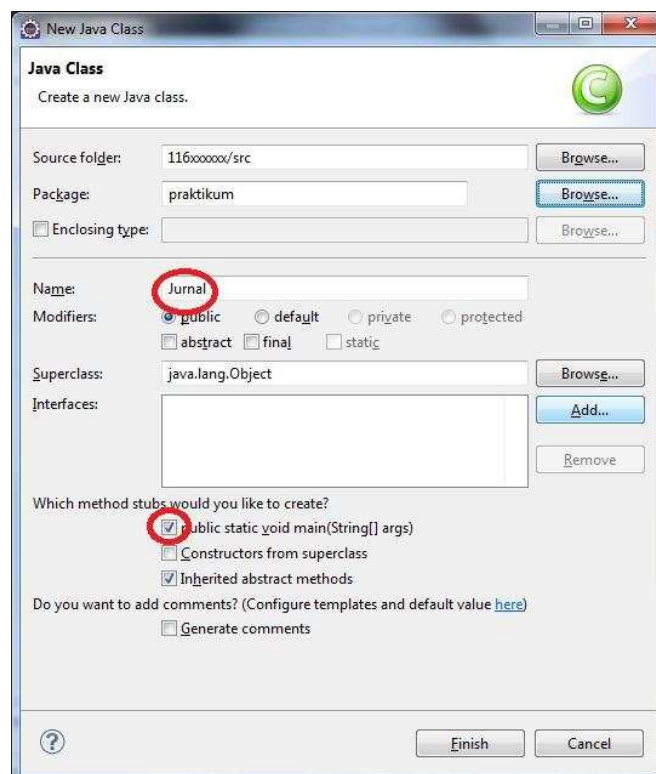
Catatan : package bersifat optional (tidak harus dibuat)

- m. Setelah itu, kita akan membuat class. Klik kanan pada package (jika ada) atau project, pilih **New -> Class**.

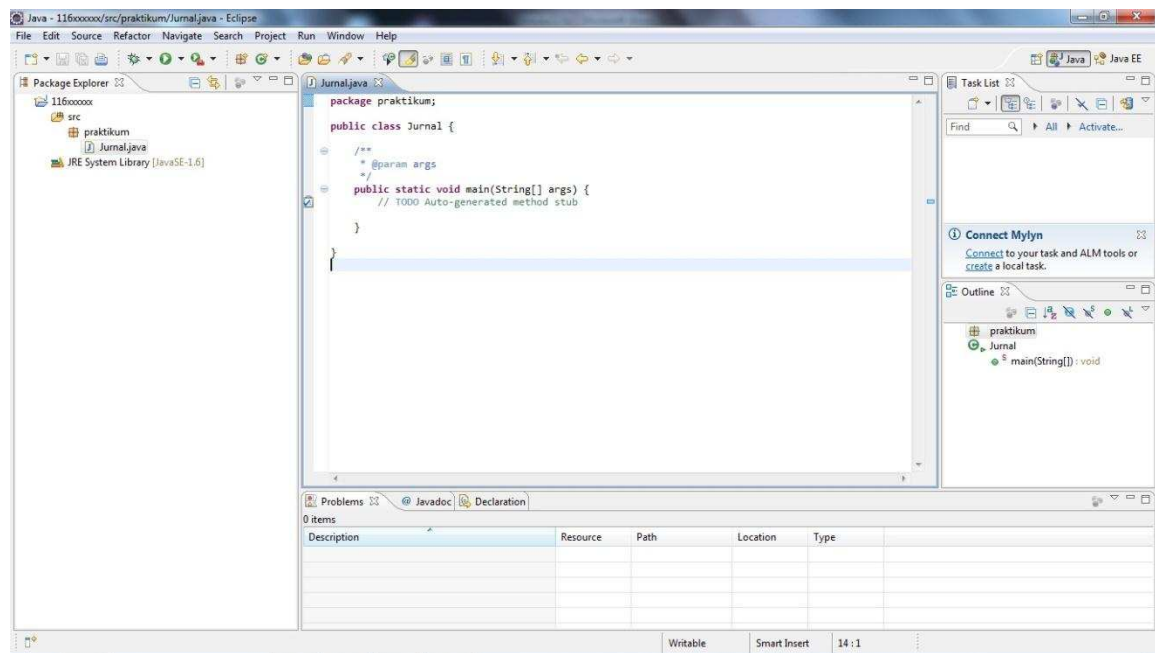


Catatan : jika kita membuat class pada project atau tidak terdapat package pada project anda maka, class berada pada package default.

- n. pada tabel dialog berikut, isikan nama dari kelas yang kita buat, kita ketikkan nama class yang diinginkan, misal **Jurnal**. Kemudian klik **public static void main** lalu klik **Finish**.



- o. Selanjutnya akan muncul tampilan seperti berikut.



p. Setelah itu, ketikkan kode program pada class yang baru dibuat.

Pengkondisian

TUJUAN PRAKTIKUM :

1. Praktikan memahami bentuk umum serta logika pengkondisian dalam Java
2. Praktikan dapat mengimplementasikan pengkondisian dalam program Java
3. Praktikan mampu memecahkan masalah sederhana dengan menggunakan analisa kasus dan mengimplementasikannya ke dalam bahasa pemrograman Java

A. PENGKONDISIAN

1. Konstruksi pengambilan keputusan

Konstruksi pengambilan keputusan adalah konstruksi yang memungkinkan program melakukan evaluasi terhadap variable / kondisi kemudian menjalankan alur program yang sesuai dengan kondisi. Dalam hal ini program dikatakan mengambil keputusan berdasarkan hasil evaluasi variable atau kondisi.

Konstruksi pengambilan keputusan :

- a. Konstruksi if
- b. Konstruksi if...else
- c. Konstruksi ...else if ...
- d. Konstruksi switch

a. Konstruksi if

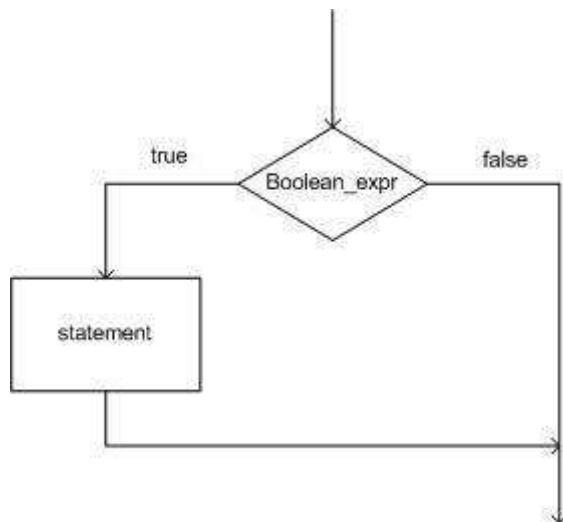


Diagram Alir untuk Konstruksi `if`

Aksi akan dieksekusi bila kondisi bernilai `true`.

Tabel 2.1 : notasi struktur IF
Dalam Java
<pre> If (kondisi) { AKSI; } </pre>
Contoh: <pre> int a = 3; if (a>0) { System.out.println(a + " bernilai positif"); } </pre>

b. Konstruksi `if – else`

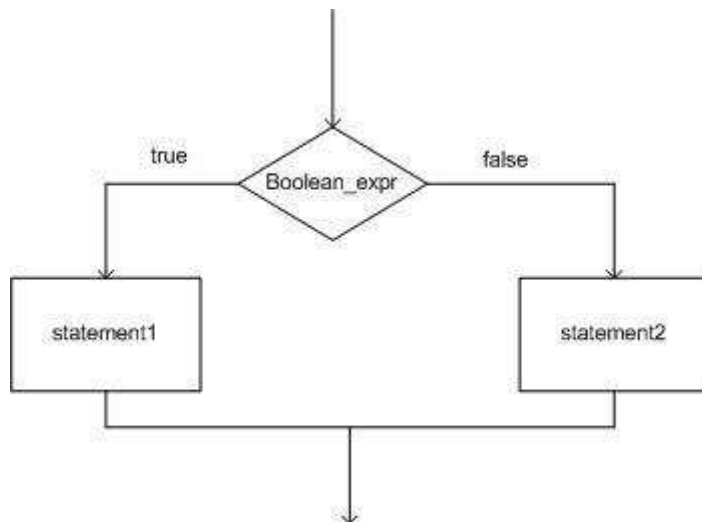


Diagram Alir untuk Konstruksi if-else

Konstruksi if-else dipakai untuk mengeksekusi salah satu dari 2 pernyataan dari syarat tertentu yang pada pada if yang dapat bernilai benar atau salah.

Tabel 2.2 : notasi struktur IF – ELSE
Dalam Java
<pre> If (kondisi) { AKSI1; } else { AKSI2; } </pre>
Contoh: <pre> int a = 3; if (a>0) { System.out.println(a + " bernilai positif"); } else { System.out.println(a + " bernilai negatif"); } </pre>

AKSI1 akan dieksekusi bila kondisi bernilai true. Kalau kondisi bernilai false, maka AKSI2 akan dieksekusi.

c. Konstruksi - else if -

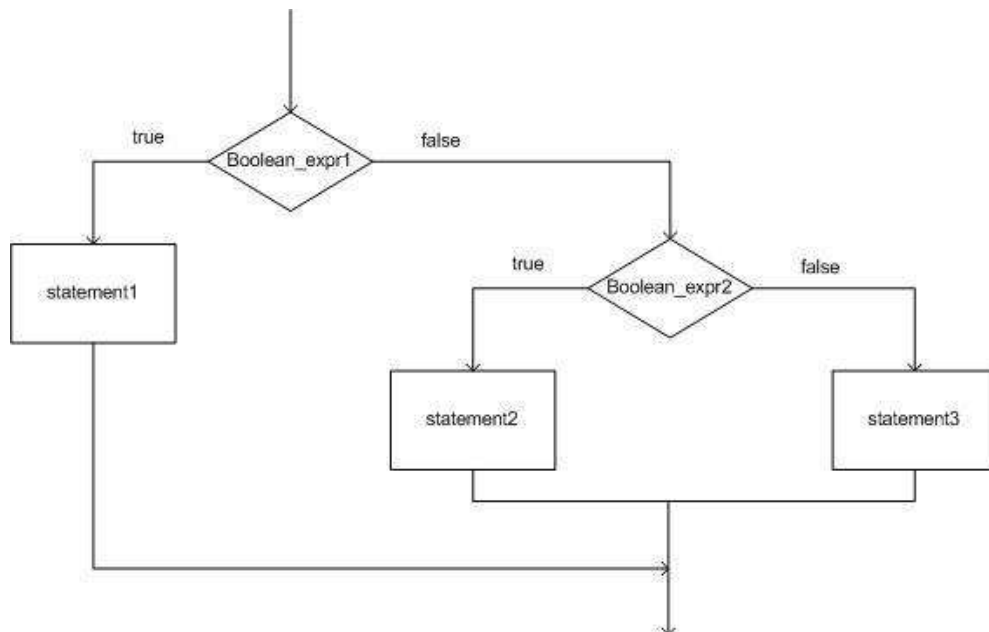


Diagram Alir untuk Konstruksi else-if

Konstruksi else-if dipakai untuk memberikan kondisi tertentu pada bagian else.

<i>Tabel 2.3 : notasi struktur - ELSE IF -</i>
Dalam Java
<pre> If (kondisi1) { AKSI1; } Else If (kondisi2) { AKSI2; } </pre>
<p>Contoh:</p> <pre> int a = 3; if (a % 2 == 0) { System.out.println(a + "bilangan genap"); } else if (a % 2 != 0) { System.out.println(a + "bilangan ganjil"); } </pre>

Ketika kondisi1 bernilai false, maka alur program akan menuju ke bagian else.

Selanjutnya AKSI2 diatas akan dikerjakan kalau kondisi2 bernilai true.

Silahkan anda ketik contoh program analisa kasus di bawah ini sebagai bahan latihan dan pahami kode programnya.

Contoh 2.1
Dalam Java
<pre>public class Test { public static void main (String args[]) { int A = 3; int B = 6; if (A > B) { System.out.println(A + ">" + B); } else if (A < B) { System.out.println(A + "<" + B); } else { System.out.println(A + "=" + B); } } }</pre>

d. Struktur switch

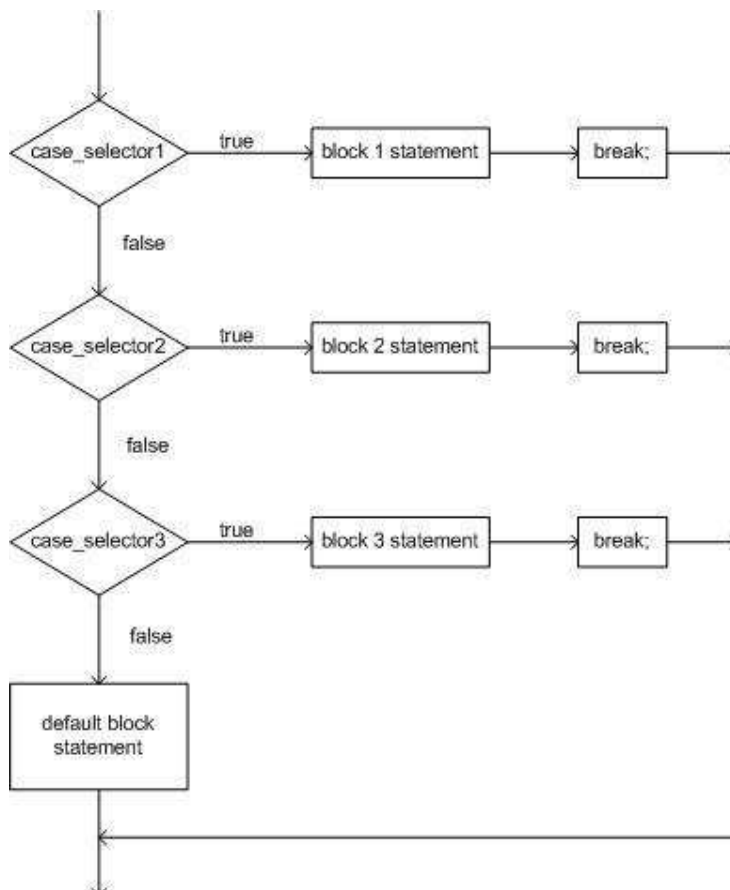


Diagram Alir untuk Konstruksi `switch`

Struktur `switch` digunakan untuk menangani banyak kemungkinan. Struktur `switch` melakukan evaluasi dan membandingkan ekspresi untuk semua konstanta `case` dan mengendalikan eksekusi program ke pernyataan `case` yang cocok

Tabel 2.4 : notasi struktur SWITCH
Dalam Java
<pre>switch (variabel) { case kondisi1 : AKSI1; case kondisi2 : AKSI2; case kondisi-n: AKSI-n ; }</pre>

Keterangan dalam penggunaan `switch` :

- Variable `switch` dalam java hanya dapat bertipe data *char*, *byte*, *short* atau *int*.
- *case* merupakan kata kunci yang mengindikasikan suatu nilai yang diuji.
- Kondisi tidak dapat berupa variable, ekspresi atau method, tetapi dapat berupa konstanta $k = \{default, 1, 2, 3, \dots, n\}$
- *break* dapat digunakan dalam `switch`. *break* adalah pernyataan yang sifatnya optional yang mengakibatkan aliran program keluar dari blok `switch`. Jika setelah aksi tidak disertai *break*, maka aliran program akan masuk ke *case* selanjutnya.
- Default merupakan kata kunci yang mengindikasikan kondisi umum yang akan dieksekusi jika semua `case` yang diuji tidak sesuai dengan nilai variabel.

Berikut ini adalah contoh pemakaian Struktur ***switch*** pada program java untuk mencari hari yang dinotasikan dengan angka sesuai dengan inputan user .

Contoh 2.2
Dalam Java

```
public class Test {  
  
    public static void main(String[] args){  
        int grade = 82;  
        switch(grade)  
        {  
            case 100:  
                System.out.println("Excellent!");  
                break;  
            case 90:  
                System.out.println("Good Job!" );  
                break;  
            case 80:  
                System.out.println("Study Harder!" );  
                break;  
            default:  
                System.out.println("Sorry, You Failed!");  
        }  
    }  
}
```

Catatan: coba contoh 2.2, ubah nilai pilihan menjadi 2 dan perhatikan perbedaan keluarannya.

soal :

1. buatlah program kelulusan matakuliah yang Inputanya berupa nilai. Jika nilainya kurang dari 55 maka dinyatakan tidak lulus
2. Buatlah program pembelian. Inputan berupa total pembelian, jika total pembelian paling rendah 50000 mendapat potongan 20%, sehingga jumlah yang dibayar adalah total pembelian dikurang potongan. Output berupa besarnya potongan dan jumlah yang harus dibayar.

Perulangan

TUJUAN PRAKTIKUM :

1. Praktikan memahami bentuk umum perulangan dalam Java
2. Praktikan dapat mengimplementasikan perulangan dalam program Java
3. Praktikan mampu memecahkan masalah sederhana dengan menggunakan analisa kasus dan mengimplementasikannya ke dalam bahasa pemrograman Java

PERULANGAN

1. Definisi Perulangan (Looping)

Perulangan dalam algoritma didefinisikan sebagai bentuk algoritma yang berfungsi untuk mengulang perintah-perintah baris program dengan aturan tertentu. Pengulangan bertujuan untuk mengefisienkan penulisan kode program, sehingga tidak perlu dilakukan berulang-ulang kali.

2. Struktur Perulangan (Looping)

Struktur perulangan secara umum terdiri atas dua bagian, yaitu:

- a. Kondisi perulangan, yaitu berupa ekspresi Boolean yang harus dipenuhi untuk melaksanakan kondisi perulangan. Kondisi ini mengakibatkan suatu kondisi perulangan akan berhenti pada saat kondisi Boolean tersebut terpenuhi.
- b. Badan (body) perulangan, yaitu suatu aksi (bagian algoritma) yang harus diulang selama kondisi yang ditentukan untuk perulangan tersebut masih terpenuhi.

3. Jenis Perulangan dalam Algoritma

Dalam modul ini akan dibahas beberapa jenis perulangan dalam bahasa pemrograman Java, antara lain :

- a. *While loop*
- b. *Do-while loop*
- c. *For loop*
- d. *Nested loop*

Berikut akan kita bahas bentuk-bentuk tersebut satu per satu:

a. While loop

While adalah bentuk perulangan yang memiliki jumlah perulangan sesuai dengan suatu kondisi logika tertentu. *Do-while loop* mirip dengan *while-loop*. Pernyataan di dalam *do-while loop* akan dieksekusi beberapa kali selama kondisi bernilai benar(*true*).

Tabel 3.1.1 : notasi struktur While

Inisialisasi
While (kondisi){
// statemen yang akan diulang
....
iterasi
}

Maksud dari bentuk di atas adalah selama kondisi_perulangan terpenuhi atau bernilai benar (*true*), maka statemen akan terus dilaksanakan sampai kondisi_perulangan bernilai salah (*false*). Jumlah perulangan ini minimal 0 kali, karena pengecekan kondisi dilakukan di awal.

```
import java.util.Scanner;

public class while1 {

    public static void main(String[] args) {

        int x = 0;

        while (x<5) {
            String nama="";
            Scanner input = new Scanner (System.in);
            System.out.println("Nama : "+nama);
            x++;
            nama = input.next ();
        }
    }
}
```

b. Do-while loop

Sama halnya dengan while, do-while juga akan menjalankan looping selama kondisi_perulangan terpenuhi atau bernilai benar (true). Berbeda pada perulangan while, pada perulangan do-while pengecekan kondisi (syarat) perulangan dilakukan setelah eksekusi statement yang diulang. Sehingga statement dalam blok do-while paling sedikit dieksekusi satu kali. Bentuk umum perulangan do-while :

Tabel 3.2.1 : notasi struktur While

Inisialisasi
Do {
// statemen yang akan diulang
...
Iterasi
} while (kondisi);

Maksud dari bentuk di atas adalah statemen akan dilakukan sebelum ada pemeriksaan kondisi perulangan.

```
import java.util.*;
public class dowhile1 {

    public static void main(String[] args) {

        int x= 0;
        do {
            String nama="";
            Scanner input = new Scanner (System.in);
            System.out.println("Nama : "+nama);
            x++;
            nama = input.next ();
        }
        while (x < 5 );

    }
}
```

c. For loop

Bentuk for loop digunakan untuk perulangan yang memiliki jumlah perulangan yang telah dipastikan sebelumnya. Bentuk umum dari perulangan traversal adalah sebagai berikut:

Tabel 3.3.1 : notasi struktur for

<pre>For (inisialisasi; kondisi; iterasi) { // statemen yang akan diulang }</pre>
--

Maksud dari bentuk di atas adalah akan dilaksanakan AKSI sebanyak N kali, dimana nilai N adalah penyesuaian kondisi perulangan dengan kondisi awal. Perulangan akan berhenti dilaksanakan jika kondisi perulangan bernilai salah (false).

```
public class looping {  
  
    public static void main(String[] args) {  
        int i;  
        for (i=1;i<=5;i++) {  
            System.out.print(i+" ");  
        }  
    }  
}
```

d. Nested loop

Nested loop merupakan perulangan di dalam perulangan. Pelajari contoh berikut dan cobalah untuk mengetahui hasilnya :

```

public class looping {

    public static void main(String[] args) {
        int i;
        int j;
        for (i=1;i<=5;i++) {
            for (j=1;j<=5;j++) {
                System.out.print("* ");
            }
            System.out.println("");
        }
    }
}

```

SOAL :

1. Buat program log-in admin lab prodase dengan :

username : si_aku

password : 12345

Bila salah memasukan username atau password,akan muncul peringatan "anda tidak berhak mengakses program ini".

2. Buatlah program dengan keluaran seperti berikut:

```

100
80 80
60 60 60
40 40 40 40
20 20 20 20 20

```

TUJUAN PRAKTIKUM :

1. Mendeklarasikan dan membuat array
2. Mengakses elemen-elemen array
3. Menentukan jumlah element dalam sebuah array
4. Mendeklarasikan dan membuat array multidimensi

A. PENGENALAN ARRAY

Dalam mendeklarasian variable, kita sering menggunakan tipe data yang sama namun dengan nama variable atau identifier yang berbeda – beda. Sebagai contoh, kita memiliki tiga variable dengan tipe data *int* dengan identifier yang berbeda tiap variabelnya.

```
int angka1;  
int angka2;  
int angka3;  
angka1 = 1;  
angka2 = 2;  
angka3 = 3;
```

Pada contoh di atas, kode tersebut kurang efektif karena harus menginisialisasi dan menggunakan tiap variable padahal dalam java atau pemrograman lain terdapat kemampuan lain untuk menampung variable – variable dengan tipe data yang sama dan dapat dimanipulasi dengan efektif.

Tipe variable ini disebut dengan **array**. Sebuah array akan menyimpan beberapa item data dengan tipe data yang sama di dalam sebuah blok memori yang berdekatan yang kemudian dibagi menjadi beberapa slot.

B. Pendeklarasian Array

Array harus di deklarasikan seperti layaknya sebuah variable, apabila Anda mendeklarasikan array, maka harus membuat sebuah list dari tipe data, yang diikuti oleh tanda kurung siku buka dan kurung siku tutup, yang diikuti oleh nama identifier.

Contoh :

```
//tipe [ ] namaArray;  
Int [ ] ages;
```

Atau

```
//tipe namaArray[];  
Int ages[];
```

Setelah pendeklarasian, kita harus membuat array dan menentukan beberapa panjangnya dengan sebuah konstruktor, proses ini didalam java disebut instantiasi (kata dalam java yang berarti membuat). Untuk meng-instantiasi sebuah objek, kita membutuhkan sebuah konstruktor. Kita akan membicarakan lagi meng-instantiasi dan pembuatan konstruktor pada praktikum selanjutnya.

Contoh :

```
//deklarasi objek  
//format penulisan = tipe namaArray[];  
Int ages[];
```

```
//instantiasi objek  
//format penulisan = variableArray = new tipe[jumlahElemen];  
Ages = new int[100];
```

Atau bisa juga ditulis

```
//deklarasi dan instantiasi  
//format penulisan = tipe namaArray[] = new tipe[jumlahElemen];  
Int ages[] = new int [100];
```

Pada contoh diatas, deklarasi akan memberitahukan kepada compiler java, bahwa identifier `ages` akan digunakan sebagai nama array yang berisi data-data integer, dan kemudian untuk membuat atau meng-instantiasi sebuah array baru yang terdiri dari 100 elemen.

Selain menggunakan sebuah keyword baru untuk menginstantiasi array, juga dapat secara otomatis mendeklarasikan array, membangun, kemudian memberitahukan sebuah nilai (value).

Sebagai contoh,

```
//membuat sebuah array yang berisi variabel-variabel boolean pada sebuah identifier.
```

```
//array ini terdiri dari 4 elemen yang diinialisasikan sebagai value {true,false,true,false}
```

```
boolean result[]={true,false,true,false};
```

```
//membuat sebuah array yang terdiri dari penginialisasian 4 variabel double bagi value{100,90,80,75}
```

```
double[4]={100,90,80,75};
```

```
//membuat sebuah array String dengan identifier days. Array ini terdiri dari 7 elemen.
```

```
String days[]={"mon","tue","wed","thu","fri","sat","sun"};
```

C. MENGAkses ELEMEN ARRAY

Untuk mengakses elemen – elemen yang terdapat dalam array, kita membutuhkan nomor atau disebut dengan index atau subscript. Nomor – nomor index atau subscript sudah diberikan dalam array, sehingga program atau programmer dapat mengaksesnya bila dibutuhkan. Perlu dicatat untuk nomor index array dimulai dari angka nol dan terus bertambah hingga list value array tersebut berakhir. Index array bertipe data *int* dan perlu diingat lagi index di dalam array dimulai dari 0 sampai dengan panjang array dikurangi 1.

Sebagai contoh, perhatikan potongan kode program di bawah ini :

```
int angka [] = {1,2,3,4,5};           //elemen – elemen array

System.out.println ( angka [2] );      //mengakses elemen array
```

Maka akan ditampilkan angka 3 pada saat di run. Perlu diingat kembali bahwa index array dimulai dari 0 sehingga pada kasus di atas akan menampilkan angka 3 bukan angka 2.

Pada saat array dideklarasikan atau dikonstruksi, nilai yang disimpan dalam array akan diinisialisasikan sebagai nol. Sehingga jika kita menggunakan tipe data referensi seperti *String*, array tersebut tidak akan diinisialisasikan menjadi string kosong (""). Sehingga untuk array *String* kita harus menginisialisasi nilainya secara eksplisit.

Petunjuk penulisan program:

1. Biasanya, lebih baik menginisialisasi atau meng-instantiate array setelah anda mendeklarasikannya

```
int[] arr = new int[100];
lebih disarankan daripada,
int[] arr;
arr=new int[100];
```

2. elemen-elemen dalam n-elemen array memiliki index dari 0 sampai n-1. Perhatikan disini bahwa tidak ada elemen array `arr[n]`. Hal ini akan menyebabkan *array-index out-of-bounds exception*.

3. Anda tidak dapat mengubah ukuran dari sebuah array.

Berikut ini adalah contoh, bagaimana untuk mencetak seluruh elemen didalam array. Dalam contoh ini digunakanlah loop, sehingga kode kita menjadi lebih pendek.

```
package abc;

public class modul {

    public static void main(String[] args) {

        int[] ages = new int[100];

        for(int i=0;i<100;i++) {
```



```

        System.out.print(ages[i]);
    }
}
}

```

Contoh program :

```

1  package modul4;
2
3  import java.util.Scanner;
4
5  public class coba {
6      public static void main (String args []){
7          Scanner input = new Scanner (System.in);
8
9          //inisialisasi list nilai untuk masing-masing elemen
10         final int angka[] = new int [5];
11         System.out.println ("-Masukan nilai elemen array-");
12         for (int counter=0; counter<angka.length; counter++){
13             System.out.print ("elemen ke-" + (counter+1) + " = ");
14             angka [counter] = input.nextInt();
15         }
16         System.out.println ();
17         System.out.printf("%s %8s \n", "Index", "Values"); //kolom heading
18
19         //keluaran masing nilai element array
20         for (int counter=0; counter < angka.length; counter++){
21             System.out.printf("%5d %8d \n", counter, angka[counter]);
22         }
23     }
24 }
25

```

Output :

```

run:
-Masukan nilai elemen array-
elemen ke-1 = 1
elemen ke-2 = 2
elemen ke-3 = 3
elemen ke-4 = 4
elemen ke-5 = 5

Index  Values
    0      1
    1      2
    2      3
    3      4
    4      5
BUILD SUCCESSFUL (total time: 12 seconds)

```

final int angka = new int [5]; menunjukkan bahwa panjang array *angka* adalah 5. Inputan menggunakan *Scanner* sehingga value untuk array tersebut akan dimasukan melalui keyboard. Pada sebelumnya kita menggunakan *System.out.println* untuk mencetak.

pada kasus ini sedikit berbeda ada *System.out.printf*, dalam fungsinya sama yaitu mencetak nilai ke layar, pada *System.out.printf* untuk mengisi nilai atau variable menggunakan % (persen) di ikuti tipe variabel s (String), d (integer). Dan ada spasi di beri notasi angka menunjukkan berapa jumlah spasi yang akan di gunakan. ("*%s%8s\n*", "*Index*", "*Values*"), String "*Index*" akan di letakkan di (%s) yg awal kemudian String "*Values*" di letakkan di (%8s) ada delapan spasi dari String "*index*". **(coba diubah-ubah/modifikasi).**

D. Panjang array

untuk mengetahui berapa banyak elemen didalam sebuah array, anda dapat menggunakan **length (Panjang)** field dalam array. Panjang field dalam array akan mengembalikan ukuran dari array itu sendiri.

Sebagai contoh:

arrayName. length

pada contoh sebelumnya, kita dapat menuliskannya kembali seperti berikut ini,

Petunjuk penulisan program:

1. Pada saat pembuatan loop untuk memproses elemen-elemen dalam array, gunakanlah *length* field didalam pernyataan pengkondisian dalam loop. Hali ni akan menyebabkan loop secara otomatis menyesuaikan diri terhadap ukuran array yang berbeda-beda

2. Pendeklarasian ukuran array di java, biasanya digunakan constant untuk mempermudah. Sebagai contoh,

```
final int ARRAY_SIZE = 1000; //pendeklarasian constant
```

Contoh Program

```
import java.util.Scanner;

public class contoh1{
    public static void main (String args[]){
        int array_length = 10;
        //jumlah elemen array

        int array[] = new int [array_length];
        //besar array_length = 10

        Scanner input = new Scanner(System.in);
        System.out.println("Masukan nilai elemen array");

        for (int i =0; i < array.length; i++) {
            System.out.print("Elemen ke "+(i+1)+" = ");
            array[i] = input.nextInt();
        }
        //perulangan input data

        System.out.printf("%s %8s \n", "index", "values");

        for (int i=0; i<array.length; i++){
            System.out.printf("%5d %8d \n", i, array[i]);
        }
    }
}
```

Akan muncul pernyataan untuk mengisi nilai elemen array dari elemen ke-1 hingga elemen ke-10. Pada contoh kali ini, akan dimasukan angka sebagai berikut :

```
Masukan nilai elemen array
Elemen ke 1 = 1
Elemen ke 2 = 2
Elemen ke 3 = 3
Elemen ke 4 = 4
Elemen ke 5 = 5
Elemen ke 6 = 6
Elemen ke 7 = 7
Elemen ke 8 = 8
Elemen ke 9 = 9
Elemen ke 10 = 10
```

Maka hasil yang akan muncul adalah :

index	values
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10

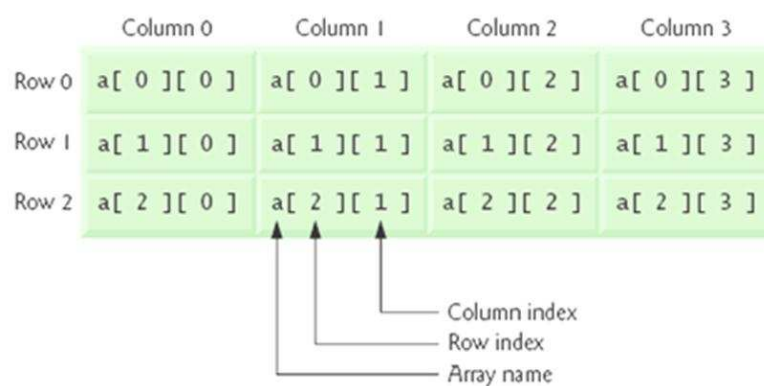
Pada contoh kasus di atas :

`int array_length = 10;` menunjukkan inisialisasi constanta `array_length` dengan nilai = 10, `int array[] = new int [array_length];` membuat dan menginisialisasi objek array kosong untuk memasukkan nilai inputan dari perulangan. Dimana di contoh program tersebut jumlah `array_length = 10`

(coba diubah-ubah/modifikasi).

E. ARRAY MULTI DIMENSI

Array multi dimensi merupakan array yang ada di dalam array. Array multi dimensi juga dapat diartikan sebagai matrix yang terdiri dari baris dan kolom. Array multi dimensi dideklarasikan dengan menambah tanda kurung siku setelah nama array.



Contoh deklarasi array :

`//array integer dengan ukuran 100 x 100`

`int twoD [] [] = new int [100] [100];`

`//contoh array String dengan ukuran 3 x 3`

`String mahasiswa[] [] = {{ "budi", "andi"},`

```
    {"tono", "rudi"},  
    {"okta", "tasya"};}
```

Untuk mengakses elemen array multi dimensi, caranya hampir sama dengan array satu dimensi. Hanya saja dalam array multi dimensi kita perlu menentukan letak kolom dan baris dari elemen tersebut. misalnya untuk mengakses elemen pertama dari array *mahasiswa* di atas kita tuliskan :

```
System.out.print ( bio [0] [0] );
```

Kode tersebut akan mencetak string "budi" pada layar.

Contoh program :

```
1  package prodase;  
2  import java.util.Scanner;  
3  class prodase{  
4  public static void main (String args[]){  
5      Scanner in = new Scanner ( System.in );  
6      String mahasiswa [] [] = new String [3][3];  
7      System.out.println ("Inputkan :");  
8      for (int i=0; i<3; i++){  
9          for (int j=0; j<3; j++){  
10             System.out.print ("baris ke-"+(i+1)+" kolom ke-"+(j+1)+" = ");  
11             mahasiswa [i][j] = in.next ();  
12         }  
13     }  
14     System.out.println ();  
15     System.out.println ("Output :");  
16     for (int i=0; i<3; i++){  
17         for (int j=0; j<3; j++){  
18             System.out.print (mahasiswa [i][j] + " ");  
19         }  
20     }  
21 }  
22 }  
23 }  
24 }  
25 }
```

Cobalah kode program di atas dan perhatikan keluaran apa yang dilakukannya. pada program di atas kita perhatikan :

```
String mahasiswa [ ] [ ] = new String [3][3];
```

Merupakan proses pendeklarasian sebuah array yang nantinya akan diinputkan elemen – elemen arraynya.

```
for (int i=0; i<3; i++){  
  
    for (int j=0; j<3; j++){  
  
        System.out.print ("baris ke-"+(i+1)+" kolom ke-"+(j+1)+" = ");  
  
        mahasiswa [i][j] = in.next ();  
  
    }  
  
}
```

Pada potongan kode di atas, merupakan kode yang digunakan untuk proses penginputan elemen – elemen array.

```
for (int i=0; i<3; i++){  
  
    for (int j=0; j<3; j++){  
  
        System.out.print (mahasiswa [i][j] + " ");  
  
    }  
  
}
```

Walaupun kode di atas menggunakan perulangan *for* seperti kode sebelumnya, namun fungsinya berbeda, pada potongan kode di atas digunakan untuk mengakses elemen array dan menampilkannya di layar.

F. Input Array menggunakan fungsi *Random*

Nilai random merupakan nilai bilangan acak, dalam program untuk men-Generate bilangan random harus membuat objek random dulu. Sintak random di java harus menggunakan java import.

```
import java.util.Random;
```

Sintaks :

Random objekRandom = new Random();

Contoh :

1. Array satu dimensi dengan input random

```
import java.util.Random;

public class ArrayRandomSatuDimensi {

    public static void main(String[] args){

        //instantiasi untuk pembentukan objek random
        Random rdm = new Random();
        int nilairandom; //deklarasi variable untuk nampung objek random

        int a[]= new int[4]; //instantiasi objek array 1 dimensi

        System.out.println("~Array satu dimensi dengan input random~\n");
        for (int i=0; i<a.length; i++){
            //assignment nilai random ke variable integer,
            //(i+1) salah untuk melakukan random setiap kali perulangan
            nilairandom=rdm.nextInt(i+1);
            a[i]=nilairandom;
            System.out.println("Nilai random array 1 dimensi indeks ke["+i+"] = " +a[i]);
        }

        System.out.println();
        System.out.printf("%s %8s \n", "index", "values");

        for (int i=0; i<a.length; i++){
            System.out.printf("%5d %8d \n", i, a[i]);
        }
    }
}
```

2. Array multidimensi dengan input random

```
import java.util.Random;

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

        //instantiasi untuk pembentukan objek random
        Random rdm = new Random();
        int nilairandom; //deklarasi variable untk nampung objek random

        int b[][] = new int[4][4]; //instantiasi objek array 2 dimensi

        System.out.println("~Array multidimensi dengan input random~\n");

        for(int i=0; i<b.length; i++){
            for(int j=0; j<b.length; j++){
                nilairandom = rdm.nextInt(i+1);
                b[i][j] = nilairandom;
                System.out.print(b[i][j]+" ");
            }
            System.out.println();
        }
    }
}
```

Coba run beberapa kali, dapat dilihat angka di dalam array tersebut berubah-ubah sesuai dengan fungsi Random.

Soal:

1. Buatlah sebuah array yang menampung nama – nama hari dalam seminggu. Gunakan Scanner untuk inputnya dan perulangan untuk menampilkannya!
2. Buatlah program penjumlahan 2 matriks berdimensi 2! (gunakan array 2 dimensi dan input random)

Searching & Shorting

TUJUAN PRAKTIKUM:

Praktikan diharapkan dapat:

1. Mengerti essensi penggunaan Searching (Pencarian) dan Sorting (Pengurutan)
2. Memahami bentuk dan jenis-jenis Searching (Pencarian) dan Sorting (Pengurutan)
3. Memecahkan masalah sederhana dengan menggunakan Algoritma Searching dan Sorting serta mengimplementasikannya kedalam sebuah bahasa pemrograman.

A. Searching

Searching adalah proses menemukan nilai (data) tertentu dari dalam sekumpulan nilai yang bertipe sama (tipe dasar maupun tipe bentukan). Proses pencarian seringkali diperlukan pada saat program perlu mengubah atau menghapus nilai tertentu (sebelum bisa mengubah atau menghapus, perlu mencari dulu apakah nilai tersebut ada dalam kumpulan nilai tersebut). Kasus lain yang memerlukan algoritma pencarian adalah penyisipan data ke dalam kumpulan data (perlu dimulai dengan pencarian apakah data tersebut telah ada sehingga terhindar dari duplikasi data).

Metode-metode Searching

Ada beberapa algoritma searching yang telah diciptakan seperti interpolation search, tree search, sequential search, binary search dan graph search. Setiap algoritma pasti mempunyai kelebihan dan kekurangan masing-masing. Pada modul ini hanya ada dua (2) metode searching yang dibahas, yaitu: sequential search dan binary search.

1. Sequential Search

Sequential search (pencarian sekuensial) adalah proses membandingkan setiap elemen larik (array) satu persatu dengan nilai yang dicari secara beruntun, mulai dari elemen pertama sampai elemen yang dicari sudah ditemukan, atau sampai seluruh elemen sudah diperiksa.

Kelebihan sequential search:

- Jika elemen yang dicari berada di depan (indeks awal array) maka elemen tersebut akan cepat ditemukan.
- Data yang dicari tidak harus dalam keadaan terurut.
- Algoritmanya cukup sederhana.

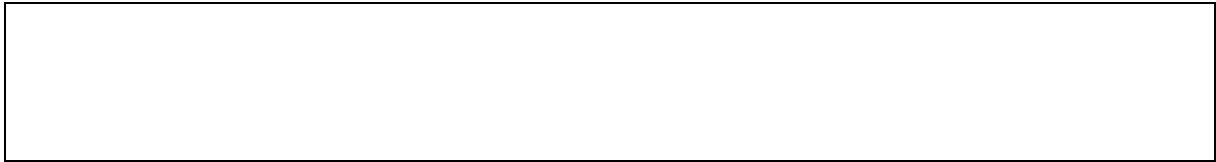
Kekurangan sequential search:

- Jika elemen yang dicari berada di akhir (indeks akhir array) maka pencarian akan berlangsung sangat lama, sehingga beban komputasi akan meningkat.
- Tidak cocok untuk mencari sebuah elemen pada kumpulan data yang sangat banyak.

Contoh 6.1

Program Sequential Search dalam Java

```
import java.util.Scanner;
public class SekuensialSearch {
    public static void main(String[] args) {
        int arr[];
        arr = new int[] { 1, 4, 2, 3, 6, 5, 7 };
        // membuat metode input, dalam contoh ini adalah scanner
        Scanner input = new Scanner(System.in);
        boolean get = false;
        int point = 0, searchItem;
        // memasukkan nilai yang dicari kedalam variabel searchItem
        searchItem = input.nextInt();
        // proses mencari apakah nilai searchItem ada didalam variabel arr
        for (int i = 0; i < arr.length; i++) {
            if (searchItem == arr[i]) {
                // status get(dapat) akan berubah menjadi true saat ada
                // nilai di dalam arr yang sama dengan searchItem
                get = true;
                // menyimpan posisi index ditemukannya nilai tersebut
                point = i;
                // break berguna untuk keluar dari perulangan
                break;
            }
        }
        // jika nilai ditemukan, maka....
        if (get)
            System.out.println("Angka " + searchItem +
                               " ditemukan dalam array ke-" + (point + 1));
        // jika tidakdi temukan, maka...
        else
            System.out.println("Angka tidak ditemukan");
    }
}
```



Program di atas merupakan contoh program sequential search dimana akan ada sebuah proses pencarian di dalam sebuah array dengan tujuh elemen.

2. Binary Search

Binary Search (pencarian biner) adalah proses mencari data dengan membagi data atas dua bagian secara terus menerus sampai elemen yang dicari ditemukan, atau indeks kiri lebih besar dari indeks kanan (data sudah tidak bisa lagi dibagi menjadi dua bagian). Kekurangan dalam pencarian ini adalah data yang dicari harus sudah dalam keadaan terurut (ascending maupun descending). Kelebihannya, jika data sudah dalam keadaan terurut maka pencarian akan berjalan jauh lebih cepat dari pada pencarian dengan sequential search.

Kelebihan binary search:

- Cocok untuk mencari data yang jumlahnya sangat banyak.
- Elemen dalam data dapat ditemukan dalam waktu yang singkat.

Kekurangan binary search:

- Data yang dicari harus dalam keadaan terurut, oleh karena itu perlu belajar metode pengurutan data (sorting) terlebih dahulu sebelum melakukan searching dengan metode ini.
- Algoritmanya lebih kompleks dari pada sequential search.

Contoh 6.2

Program Binary Search dalam Java

Dengan data terurut secara ascending

```
import java.util.Scanner;
public class BinarySearch {
    public static void main(String[] args) {
        int arr[];
        arr = new int[] { 2, 7, 11, 13, 90, 101 };
        // membuat metode input, dalam contoh ini adalah scanner
        Scanner input = new Scanner(System.in);
        // inisialisasi l(lengan kiri) dan r(lengan kanan)
        int l = 0, r;
        r = arr.length;
        int searchItem;
        // memasukkan nilai yang dicari kedalam variabel searchItem
        searchItem = input.nextInt();
        // kita akan mencari hingga batas l - r = -1 atau l + 1 = r
        while (l + 1 < r) {
            // membuat index tengah (l+r)/2
            int indexTengah;
            indexTengah = (l + r) / 2;
            // jika nilai array pada index tengah lebih dari nilai yang dicari
            // maka index tengah adalah l (lengankiri)
            if (arr[indexTengah] < searchItem)
                l = indexTengah;
            // jika bukan, maka index tengah adalah r (lengan kanan)
            else
                r = indexTengah;
        }
        // jika nilai arr pada lengan kiri adalah nilai yang dicari
        if (arr[l] == searchItem)
            System.out.println("Data ditemukan pada index ke-" + (l + 1));
        // jika nilai arr pada lengan kanan adalah nilai yang dicari
        else if (arr[r] == searchItem)
            System.out.println("Data ditemukan pada index ke-" + (r + 1));
        // jika bukan keduanya
        else
            System.out.println("Data tidak ditemukan");
    }
}
```

Contoh berikut ini untuk memperjelas cara kerja binary search pada deretan angka
"2 7 11 13 90 101" :

Bentuk penyimpanan dalam array "tabel[index]":

Angka	2	7	11	13	90	101
Index	1	2	3	4	5	6

1. Proses pencarian angka 11

Iterasi 1 $i=1, j=6$ dan $k=(1+6) \div 2=3$
karena "tabel[3]=11" maka pencarian dihentikan dan ketemu=true

2. Proses pencarian angka 7

Iterasi 1 $i=1, j=6$ dan $k=(1+6) \div 2=3$
karena (tabel[3]!=7 dan $i \leq j$) maka pencarian masih dilanjutkan

Iterasi 2 Karena tabel[k]>x ($11 > 7$) maka $j=k-1$, sehingga
 $j=3-1=2$, $i=1$, dan $k=(1+2) \div 2=1$
karena (tabel[1]!=7 dan $i \leq j$) maka pencarian masih dilanjutkan

Iterasi 3 Karena tabel[k]<x ($2 < 7$) maka $i=k+1$, sehingga
 $i=1+1=2$, $j=2$, dan $k=(2+2) \div 2=2$
karena tabel[2]=7 maka pencarian dihentikan dan ketemu=true

3. Proses pencarian angka 6

Iterasi 1 $i=1, j=6$ dan $k=(1+6) \div 2=3$
karena (tabel[3]!=6 dan $i \leq j$) maka pencarian masih dilanjutkan

Iterasi 2 Karena tabel[k]>x ($11 > 6$) maka $j=k-1$, sehingga
 $j=3-1=2$, $i=1$, dan $k=(1+2) \div 2=1$
karena (tabel[1]!=6 dan $i \leq j$) maka pencarian masih dilanjutkan

Iterasi 3 Karena tabel[k]<x ($2 < 6$) maka $i=k+1$, sehingga
 $i=1+1=2$, $j=2$, dan $k=(2+2) \div 2=2$
karena (tabel[2]!=6 dan $i \leq j$) maka pencarian masih dilanjutkan

Iterasi 4 Karena tabel[k]<x ($7 > 6$) maka $j=k-1$, sehingga
 $j=2-1=1$, $i=2$, dan $k=(2+1) \div 2=1$
karena (tabel[1]!=6 dan $i > j$) maka pencarian dihentikan dan ketemu=false

B. Sorting

Sorting merupakan sebuah proses untuk mengatur item dalam suatu urutan tertentu (menaik atau menurun). Misalnya untuk mengurutkan bilangan, mengurutkan NIM, mengurutkan nama, dsb.

Operasi Dasar Sorting:

1. Membandingkan nilai
2. Memindahkan nilai-nilai dalam daftar ke posisi yang sesuai

Metode-metode Sorting

1. Selection Sort (Pengurutan dengan Seleksi)

Selection Sort merupakan cara mencari (searching) nilai ekstrim pada kumpulan sebuah data. Jadi sebelum menggunakan metode ini harus mempelajari metode searching terlebih dahulu.

Cara pengurutan dalam Selection Sort:

- a. Cari nilai ekstrim (minimum atau maximum, pilih salah satu) pada semua elemen array dan pertukarkan dengan elemen array yang seharusnya (minimum di pertama atau maksimum di akhir). Lalu elemen array tersebut di isolasi (tidak diganggu gugat).
- b. Temukan elemen array bernilai ekstrim dari array index kedua dari awal jika minimum atau kedua dari akhir jika maksimum, kemudian pertukarkan dengan elemen array di posisi (indeks) kedua (dari awal atau dari akhir tergantung penggunaan maksimum atau minimum). Lalu isolasi elemen array tersebut ditambah elemen sebelumnya.
- c. Ulangi langkah-langkah di atas untuk elemen array berikutnya hingga elemen array terakhir.

Contoh berikut ini menunjukkan cara kerja selection sort pada deretan angka "5 1 4 8 2" yang diurutkan menaik (ascending):

5	1	4	8	2	{nilai terendah=1, tukar dengan posisi 1}
1	5	4	8	2	{nilai terendah=2, tukar dengan posisi 2}
1	2	4	8	5	{nilai terendah=4, posisi tetap}

1	2	4	8	5	{nilai terendah=5, tukar dengan posisi 4}
1	2	4	5	8	{nilai terendah=8, posisi tetap}
1	2	4	5	8	{selesai}

Contoh 6.3

Program Selection Sort dalam Java

```

public class SelectionSort {
    public static void main(String[] args) {
        int arr[];
        arr = new int[] { 2, 6, 3, 5, 1, 2, 8, 9 };
        // pengulangan sesuai panjang array - 1
        // mengapa - 1 ?
        int i = 0;
        while (i < arr.length - 1) {
            // menyimpan nilai diposisi ini untuk dibandingkan dengan nilai
            // ditempat lain
            int tmp = i;
            // melakukan pengulangan sepanjang array untuk mencari nilai
            // terbesar atau terkecil setelah posisi awal
            for (int j = i + 1; j < arr.length; j++) {
                // dalam contoh ini, kita mencari nilai terbesar
                if (arr[j] > arr[tmp])
                    tmp = j;
            }
            // setelah keluar dari perulangan, kita telah mendapatkan index
            // nilai terbesar selain bilangan yang 'tidak dapat dipilih' dalam
            // variabel tmp
            // sekarang kita akan menukar nilai di index i dengan nilai di index
            // tmp
            int helper = arr[i];
            arr[i] = arr[tmp];
            arr[tmp] = helper;
            // sekarang, kita isolasi index ke-i agar tidak dapat dirubah dengan
            // berpindah ke index depannya
            i++;
        }

        // coba kita tampilkan hasilnya
        for (int iterator = 0; iterator < arr.length; iterator++) {
            if (iterator != 0)
                System.out.print(",");
            System.out.print(arr[iterator]);
            // lho, tadi kita menggunakan arr[i], sekarang mengapa menggunakan
            // arr[iterator] ?
        }
    }
}

```

2. Insertion Sort (Pengurutan dengan Penyisipan)

Insertion sort adalah sebuah algoritma pengurutan yang membandingkan dua elemen data pertama, mengurutkannya, kemudian mengecek elemen data berikutnya satu persatu dan membandingkannya dengan elemen data yang telah diurutkan.

Cara pengurutan dalam Selection Sort:

- Membandingkan dua elemen data pertama dan mengurutkannya.
- Mengambil satu elemen data berikutnya dan membandingkannya dengan dua elemen data pertama yang telah terurut, kemudian mengurutkannya. Elemen data ketiga ini bisa diletakkan sebelum elemen data pertama, setelah elemen data kedua, atau disisipkan diantara elemen data pertama dan kedua.
- Mengulang langkah kedua hingga seluruh elemen data dalam daftar sudah diurutkan.

Contoh berikut ini menunjukkan cara kerja insertion sort pada deretan angka "5 1 4 8 2" yang diurutkan menaik (ascending):

5	1	4	8	2	{kondisi awal}
1	5	4	8	2	{pengurutan 2 elemen data}
1	4	5	8	2	{pengurutan 3 elemen data}
1	4	5	8	2	{pengurutan 4 elemen data}
1	2	4	5	8	{pengurutan 5 elemen data → selesai}

Contoh 6.4

Program Insertion Sort dalam Java

```
public class InsertionSort {  
    public static void main(String[] args) {  
        int arr[] = { 4, 3, 2, 5 };  
        // kita mengulangi dari array index ke-1 hingga terakhir  
        // kenapa tidakdi mulai dari index ke-0 ?  
        for (int i = 1; i < arr.length; i++) {  
            // kita menyimpan nilai di index i dan 'menganggarnya' nilai  
            // terkecil  
            int min = arr[i];  
            // pengulangan hanya dimulai dari index ke-i karena index setelah  
            // nyatidak perlu diperiksa  
            int j = i;  
            // ini alasan mengapa tidakdi mulai dari index ke-0  
            // jika dimulai dari index 0, maka j-1 akan menghasilkan -1 dan itu  
            // tidak diperbolehkan dalam array  
            while ((j > 0) && (min < arr[j - 1])) {  
                // nilai pada array ke-j akan selalu digantikan oleh nilai  
                // sebelumnya  
                // contoh : 3 4 2 1 (disini kita mengambil patokan 2) (ini hanya  
                // contoh) (j ada di index 2)  
  
                // menjadi: 3 4 4 1 (j adadi index 1)  
                // menjadi: 3 3 4 1 (j adadi index 0, perulangan selesai)  
                // lho, itu 2-nya hilang, kemana ?  
                arr[j] = arr[j - 1];  
                j--;  
            }  
            // masih ingat nilai i yang kita anggap terkecil ?  
            // saat menjadi patokan, otomatis 2 akan menjadi nilai terkecil juga  
            // sehingga menjadi : 2 3 4 1  
            arr[j] = min;  
        }  
        // coba tampilkan hasilnya  
        for (int i = 0; i < arr.length; i++) {  
            System.out.println(arr[i] + " ");  
        }  
    }  
}
```

3. Bubble Sort (Pengurutan dengan Pertukaran Harga)

Pada metode bubble sort, proses pengurutan dimulai dengan membandingkan elemen pertama untuk mendapatkan elemen terbesar. Lalu elemen tersebut ditempatkan pada posisi terakhir.

Cara pengurutan dalam Bubble Sort:

- Lakukan pengulangan (pass) pada array, tukar elemen yang bersebelahan jika diperlukan (perbandingan nilainya tidak sesuai); jika tidak ada pertukaran nilai maka array sudah terurut.
- Dalam pass pertama, temukan elemen dengan nilai tertinggi (maksimal) dalam array dan tukarkan dengan elemen di sebelah kanannya dan seterusnya sampai dengan mencapai posisinya di ujung array sebelah kanan.
- Kemudian dalam pass kedua, nilai tertinggi kedua dalam array akan ditempatkan dalam posisinya (di sebelah kiri elemen dengan nilai tertinggi/maksimal).
- Teruskan untuk pass ketiga dan seterusnya sampai pass n-1
- Jika sebelum pass n-1 sudah tidak ada pertukaran data maka pertukaran langsung dihentikan.

Contoh berikut ini menunjukkan cara kerja bubble sort pada deretan angka “5 1 4 8 2” yang diurutkan menaik (ascending):

5	1	4	8	2	{kondisi awal}
1	4	2	5	8	{iterasi ke-1}
1	2	4	5	8	{iterasi ke-2}
1	2	4	5	8	{iterasi ke-3, karena sudah tidak ada perpindahan elemen maka iterasi dihentikan}

Untuk detail setiap tahap dalam iterasi dapat dilihat pada proses di bawah ini,

Iterasi 1 (5 1 4 2 8) → (1 5 4 2 8)

 (1 5 4 2 8) → (1 4 5 2 8)

 (1 4 5 2 8) → (1 4 2 5 8)

	(1 4 2 5 8) → (1 4 2 5 8) {sudah terurut}
	{Karena sudah mencapai elemen data terakhir dalam daftar, algoritma melakukan iterasi kedua}
Iterasi	(1 4 2 5 8) → (1 4 2 5 8) {sudah terurut}
2	(1 4 2 5 8) → (1 2 4 5 8)
	(1 2 4 5 8) → (1 2 4 5 8) {sudah terurut}
	(1 2 4 5 8) → (1 2 4 5 8) {sudah terurut}
	(1 2 4 5 8) → (1 2 4 5 8) {sudah terurut}
	{Karena sudah mencapai elemen data terakhir dalam daftar, algoritma melakukan iterasi ketiga}
Iterasi	(1 2 4 5 8) → (1 2 4 5 8) {sudah terurut}
3	(1 2 4 5 8) → (1 2 4 5 8) {sudah terurut}
	(1 2 4 5 8) → (1 2 4 5 8) {sudah terurut}
	(1 2 4 5 8) → (1 2 4 5 8) {sudah terurut}
	{Karena sudah mencapai elemen data terakhir dalam daftar dan tidak ada pertukaran selama iterasi 3, algoritma dihentikan}

Contoh 6.5

Program Bubble Sort dalam Java

```

public class BubbleSort {
    public static void main(String[] args) {
        int arr[] = { 4, 3, 2, 5 };
        for (int i = 1; i < arr.length; i++) {
            // arr.length-1 berguna agar nilai
            // pada index yang telah dipastikan benar nilainya (dibelakang)
            // tidak berubah
            for (int j = 0; j < arr.length - i; j++) {
                // pengkondisian
                // jika tidak sesuai, maka nilainya akan ditukarkan
                if (arr[j + 1] < arr[j]) {
                    int tmp = arr[j + 1];
                    arr[j + 1] = arr[j];
                    arr[j] = tmp;
                }
            }
        }
        // menampilkan data
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i] + " ");
        }
    }
}

```



Latihan Soal :

1. a. Apa yang dimaksud dengan array?
 - Apa perbedaan array satu dimensi dan array dua dimensi?
 - Bagaimana cara penulisan array satu dimensi dan array dua dimensi pada Java?
- b. Carilah algoritma searching yang belum ada di modul!
Misalnya Pencarian pohon (*tree search*), Pencarian grafik(*graph search*)
- c. Carilah algoritma sorting yang belum ada di modul!
Misalnya Merge Sort, Quick Sort
2. Jelaskan perbedaan metode sequential search dan metode binary search !
3. Pengurutan genap dan ganjil
Brust menginginkan mengurutkan semua angka yang telah dimasukkan tetapi cara pengurutannya berbeda dari biasanya. Dia ingin angka ganjil yang diurutkan terlebih dahulu baru kemudian angka genap. Sebagai anak Sistem Informasi yang mengerti pemrograman bantulah dia.
*kalau ada angka 0 anggap saja termasuk angka genap.
Tampilan yang diinginkan Brust:
Input:
Jumlah angka yang dimasukkan: 7
Angka ke-1:1
Angka ke-2:6
Angka ke-3:9
Angka ke-4:2
Angka ke-5:3
Angka ke-6:2

Angka ke-7:4

Output:

Pengurutan Brust : 1 3 9 2 2 4 6

4. Jacky ingin mencari sebuah angka yang telah disimpan dalam sebuah array, tetapi karena dia tidak kuliah di ITT dia merasa sangat kesulitan. Untuk itu bantulah dia dengan cara membuat program yang dapat **mengurutkan** semua angka yang telah diinputkan secara *descending* kemudian **mencari** sebuah angka dengan metode binary search.

Tampilan yang diinginkan Jacky

Input:

Jumlah Angka yang disimpan = 6

Angka ke-1 = 9

Angka ke-2 = 7

Angka ke-3 = 1

Angka ke-4 = 14

Angka ke-5 = 3

Angka ke-6 = 2

Output:

Angka yang telah diurutkan:

1 2 3 7 9 14

Input:

Angka yang dicari = 7

Output = ditemukan pada array ke-3

Input:

Angka yang dicari = 10

Output = Angka tidak ditemukan

5. Ada sebuah deretan abjad yang tidak berurutan, seperti *z,g,q,t,o,z,a,g,k,p,o,l,f,s,q,g,w,g,d* karena jumlahnya yang sangat banyak seorang anak merasa kesulitan untuk mengurutkannya. Sebagai anak sistem informasi anda diminta untuk membuat sebuah program yang bisa mengurutkan abjad tersebut secara *ascending*.

Input :

Jumlah abjad yang diurutkan = 8

Note:::

- Ingat, angka yang dicari dengan metode binary search harus sudah dalam keadaan terurut

Abjad ke-1 = w

Abjad ke-2 = t

Abjad ke-3 = j

Abjad ke-4 = a

Abjad ke-5 = z

Abjad ke-6 = l

Abjad ke-5 = q

Abjad ke-6 = h

Output:

Abjad setelah diurutkan : a,h,j,l,q,t,w,z

TUJUAN PRAKTIKUM :

1. Memahami konsep method pada java.
2. Mengerti esensi penggunaan method dalam Java.
3. Memahami bentuk umum method.

A. Class

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

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

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.

B. 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();
```

C. 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.

Adapun perbedaan *local* dan *instance variable* adalah :

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

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

2. *Local variable* dideklarasikan di dalam method.

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

D. Method

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

Berikut adalah karakteristik dari method :

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

Deklarasi sebuah method

Method terdiri atas dua bagian yakni :

1. Method declaration
2. Method Body

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

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

Contoh :

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

E. Modifier pada method

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

- Public
Atribut ini menunjukan bahwa fungsi/method dapat diakses oleh kelas lain.
- Private

Atribut ini menunjukkan bahwa fungsi atau method tidak dapat diakses oleh kelas lain

- Protected

Atribut ini menunjukkan bahwa fungsi atau method bisa diakses oleh kelas lain dalam satu paket dan hanya kelas lain yang merupakan subclass nya pada paket yang berbeda.

- Tanpa modifier

Atribut ini menunjukkan bahwa method dapat diakses oleh kelas lain dalam paket yang sama.

F. Method tanpa nilai balikan

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

```
class Kotak{
    double panjang;
    double lebar;
    double tinggi;
    //mendefinisikan method void (tidak mengembalikan nilai)
    void cetakVolume(){
        System.out.println("Volume kotak = " +(panjang*lebar*tinggi));
    }
}

class DemoMethod1{
    public static void main(String[] args){
        Kotak k1, k2, k3;
        //instansiasi objek
        k1=new Kotak();
        k2=new Kotak();
        k3=new Kotak();

        //mengisi data untuk objek k1
        k1.panjang=4;
        k1.lebar=3;
        k1.tinggi=2;
        //mengisi data untuk objek k2
        k2.panjang=6;
        k2.lebar=5;
        k2.tinggi=4;
        //mengisi data untuk objek k3
```

```

        k3.panjang=8;
        k3.lebar=7;
        k3.tinggi=6;
        //memanggil method cetakVolume() untuk masing-masing
//objek
        k1.cetakVolume();
        k2.cetakVolume();
        k3.cetakVolume();
    }
}

//Output :
//Volume kotak = 24.0
//Volume kotak = 120.0
//Volume kotak = 336.0

```

G. Method dengan nilai balikan

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

```

class Kotak{
    double panjang;
    double lebar;
    double tinggi;
    //mendefinisikan method yang mengembalikan tipe double
    double hitungVolume(){
        //menghitung volume
        double vol = panjang*lebar*tinggi;
        //mengembalikan nilai
        return vol;
    }
}

class DemoMethod2{
    public static void main(String[] args){
        Kotak k1, k2, k3;
        //instansiasi objek
        k1=new Kotak();
        k2=new Kotak();
        k3=new Kotak();
        //mengisi data untuk objek k1

```

```

        k1.panjang=4;
        k1.lebar=3;
        k1.tinggi=2;
        //mengisi data untuk objek k2
        k2.panjang=6;
        k2.lebar=5;
        k2.tinggi=4;
        //mengisi data untuk objek k3
        k3.panjang=8;
        k3.lebar=7;
        k3.tinggi=6;
        System.out.println("Volume k1 = "+k1.hitungVolume());
        System.out.println("Volume k2 = "+k2.hitungVolume());
        System.out.println("Volume k3 = "+k3.hitungVolume());
    }
}
//Output :
//Volume k1 = 24.0
//Volume k2 = 120.0
//Volume k3 = 336.0

```

H. 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;
}

```

Disini, variabel panjang dan lebar disebut parameter.

```

Luas1 = luasPersegiPanjang(10, 5);

```

Adapun pada statemen diatas, nilai 10 dan 5 disebut argumen.

Sekarang, kita akan mengimplementasikan konsep di atas ke dalam kelas Kotak. Di sini data panjang, lebar, dan tinggi akan kita isikan melalui sebuah method. Berikut ini kode program yang dapat digunakan untuk melakukan hal tersebut.

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

    //mendefinisikan method dengan parameter
    void isiData(double p, double l, double t){
        panjang = p;
        lebar = l;
        tinggi = t;
    }
    double hitungVolume(){
        return(panjang*lebar*tinggi);
    }
}

class DemoMethod3{
    public static void main(String[] args){
        Kotak k;
        //instansiasi objek
        k = new Kotak();

        //memanggil method isiData()
        k.isiData(4,3,2);

        System.out.println("Volume kotak = " + k.hitungVolume());
    }
}

Output :
Volume kotak = 24.0
```

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

Contoh penggunaan return:

```
package cobaCoba;
```

```

import java.util.Scanner;

class balok {
    int p, l, t;
    int volume( int p, int l, int t)
    {
        return (p*l*t);
    }

    public static void main(String args[]) {
        Scanner masuk = new Scanner(System.in);
        //fungsi untuk menginputkan suatu nilai
        System.out.print("Panjang = "); int a=masuk.nextInt();
        System.out.print("Lebar = "); int b=masuk.nextInt();
        System.out.print("Tinggi = "); int c=masuk.nextInt();

        balok coba = new balok();
        System.out.print("\nVolume balok = "+ coba.volume(a,b,c));
    }
}

//Out :
//Panjang = 2
//Lebar = 3
//Tinggi = 2
//Volume balok = 12

```

I. Method Static

Sebuah method static dapat diakses tanpa harus melakukan instantiasi terlebih dahulu.

Pemanggilan method static dilakukan dengan format :

```
Nama_kelas.nama_method();
```

Nama_kelas diberikan bila method tersebut dipanggil dari kelas yang berbeda..

Contoh :

```
class MyStatic {
    static void Luas() {
        int p = 10; int l = 2 ;
        System.out.println("Luas = "+ p*l );
    }
    public static void main (String [] args )
    {
        Luas(); // pemanggilan method luas
    }
}

////////////////////////////////////
////////////////////////////////////

public class persegi {
    static int hitungluas(int p, int l){
        return p*l;
    }
    public static void main(String[] args) {
        int z=0;
        z=hitungluas(3,2);
        System.out.println(z);
    }
}
```


Herdianto (RDE)	herdi.16@gmail.com
Hafidh Rashemi R (HRR)	section_another@hotmail.com
Septian Hari (SHA)	septianhari@gmail.com
Bambang Dwi Asmoro (BAM)	pioner.ultimate@gmail.com
Moh Fahrur Rizqon (FHR)	f4_lung@yahoo.co.id
Rani Auliya Syafrudin (RAS)	rani.syafrudin@yahoo.com
Kholifatul Ummah (KHU)	ifakholifa@outlook.com
Komang Aditya Respa Putra (RPA)	rezpa.snk@gmail.com
Yusril Maulidan Raji (YRL)	yusrilmaulidanraji@gmail.com
Ainu Faisal Pambudy (NOE)	ainu@outlook.com
Renantia Indriani (RNT)	renantia.indriani@gmail.com
Fathimah Muthi Luthfiyah (FML)	muthi.rikaka@gmail.com