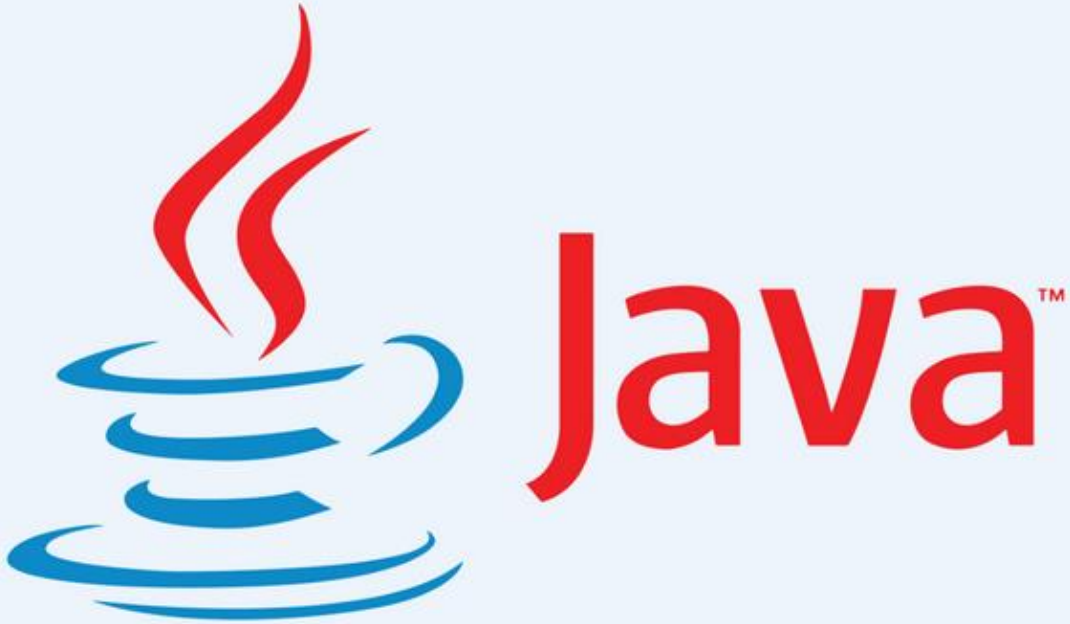
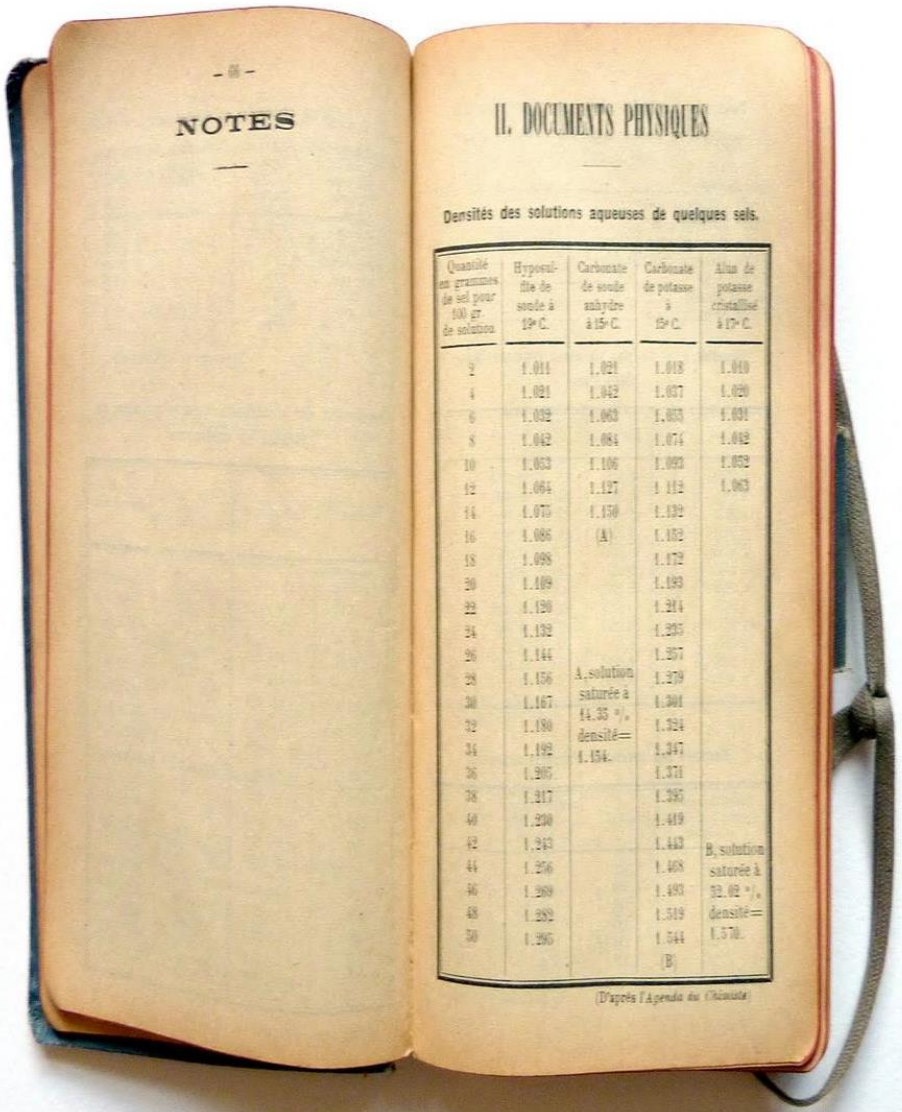
Pemrograman Berorientasi Objek



Pertemuan 2

Tutorial JAVA (Pemrograman Dasar)

Pemateri : Chrismikha Hardyanto S.Kom., M.Kom.



KONTEN PERKULIAHAN

- **Tipe Data Pada JAVA**
- **Variabel & Konstanta**
- **Operator pada JAVA**
- **Struktur Program Dasar (Runtunan, Percabangan, Perulangan)**
- **Prosedur & Fungsi (Method)**

Pemrograman Dasar Dengan JAVA

Praktikum 1:
Tipe Data Pada JAVA

Tipe Data

Pemrogramman pada JAVA pasti akan menggunakan suatu data didalamnya.

Ada **2 jenis tipe data** yang dikenal oleh JAVA, yaitu :

1. Tipe Data Primitif

2. Tipe Data References/Objek

TIPE DATA PRIMITIF

Merupakan **tipe data bawaan** yang disediakan oleh JAVA. beberapa tipe data yang dikenal oleh JAVA, antara lain :

- **Tipe Bilangan Bulat** (`byte`, `short`, `int`, **dan** `long`)
- **Tipe Bilangan Pecahan** (`float` **dan** `double`)
- **Tipe Text** (`char`)
- **Tipe Logika** (`boolean`)

TIPE DATA REFERENCES/OBJEK

1. Tipe Data ini didefinisikan ketika akan **menginstansiasikan** sebuah **kelas** menjadi **objek.**
2. Nilai awalnya adalan **null**
3. contoh instansiasi objek adalah
   Manusia Chris = new Manusia("Chris) - "Akan dibahas lebih lanjut "

[sumber : Buku Pemrograman Berorientasi Objek dengan JAVA – Adam MB**]**

Tipe Data Primitif

TIPE BILANGAN BULAT (NUMBER)

Tipe	Panjang	Range	Contoh
byte	8 bit	-2^7 to $2^7 - 1$ (-128 to 127)	2 -114 0b10 (biner)
short	16 bit	-2^{15} to $2^{15} - 1$ (-32,768 s.d 32,767)	2 -32699
int (Default)	32 bit	-2^{31} to $2^{31} - 1$ (-2,147,483,648 to 2,147,483,647)	2 147334778 123_456_678
long	64 bit	-2^{63} to $2^{63} - 1$ (-9,223,372,036854,775,808 to 9,223,372,036854,775,807)	2 -2036854775808L 1L

- Gunakan akhiran huruf "l" atau "L" jika ingin mengisi variable bertipe long.

Implementasi pada JAVA:

```
public static void main(String[] args) {
    byte b;
    short s;
    int i;
    long l;
    b = 120;
    s = 32767;
    i = 1_000_000_000;
    l = 90000000000L;
    System.out.println("byte  : "+b);
    System.out.println("short : "+s);
    System.out.println("int:  "+i);
    System.out.println("long  : "+l);
}
```

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Tipe Data Primitif (2)

TIPE BILANGAN PECAHAN

Tipe	Panjang	Contoh
float	32 bit	99F -32745699.01F 4.2E6F (notasi untuk 4.2 * 10^6)
double (Default)	64 bit	-1111 2.1E12 99970132745699.999

- Gunakan akhiran "F" jika anda ingin mengisi nilai berupa float. Jika sebuah angka pecahan tidak diakhiri dengan "F" maka akan dianggap bertipe double.

Implementasi pada JAVA:

```
public class TipeData {
    public static void main(String[] args) {
        float f;
        double d;
        f = 1234567890.123456789F;
        d = 1_234_567_890.123456789;
        System.out.println("Float    : "+f);
        System.out.println("Double   : "+d);
    }
}
```

```
run:
Float    : 1.23456794E9
Double   : 1.2345678901234567E9
BUILD SUCCESSFUL (total time: 0 seconds)
```

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Tipe Data Primitif (3)

TIPE BILANGAN TEXT

- Satu-satunya tipe data primitif teks adalah `char`.
- Digunakan untuk sebuah karakter (16 bit).
- Contoh :

```
public char jenisKelamin= 'L';
```

Implementasi pada JAVA:

```
public class TipeData {
    public static void main(String[] args) {
        char c;
        char tab='\t';
        char newline='\n';
        c='A';
        System.out.println("Hasil"+newline+"Char: "+ tab+"berisi "+c);
    }
}
```

```
run:
Hasil
Char:   berisi A
BUILD SUCCESSFUL (total time: 0 seconds)
```

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Tipe Data Primitif (4)

TIPE BILANGAN BOOLEAN

- Tipe data untuk menampung data logika bisa menggunakan tipe `boolean`.
- Hanya dapat menampung nilai `true` atau `false`.

Implementasi pada JAVA:

```
public class TipeData {
    public static void main(String[] args) {
        int umur=19;
        boolean dewasa;
        dewasa = (umur>= 17);
        System.out.println("Status Dewasa: "+dewasa);
    }
}
```

```
run:
Status Dewasa: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Tipe Data String pada JAVA

KONSEP DASAR :

STRING pada JAVA **bukan** merupakan kelompok tipe data primitive. String disini sebenarnya merupakan sebuah **kelas java** dimana isi objeknya merupakan **kumpulan dari char atau suatu text**. (Bisa dikatakan String adalah implementasi Tipe Data Objek)

- **Cara mendeklarasikan String Pada JAVA (gunakan kata kunci "String"):**

```
String <nama_variable> = "nilai_string";
```

- **Contoh Implementasi pada JAVA :**

```
String namaDepan, namaBelakang;

namaDepan = "Chrismikha";
namaBelakang = "Hardyanto";
```

Contoh Program

- **Penggalan Source code**

```
//Deklarasi Variable untuk Tipe Data Number
byte iniByte = 127;
short iniShort = 32727;
int iniInt = 2147483647;
long iniLong = 100000000001;

//Deklarasi Variable untuk Tipe Data Desimal
float iniFloat = 3.14f;
double iniDouble = 3.14;

//Deklarasi Variable Untuk Tipe Data Text
char iniChar = 'C';
char iniChar2 = 'H';

//Dklarasi Variable untuk Tipe Data Boolean
boolean iniTrue = true;
boolean iniFalse = false;

//Deklarasi Variabel Untuk Tipe Data String
String namaDepan = "Chrismikha";
String namaBelakang = "Hardyanto";
String namaLengkap = namaDepan +" "+namaBelakang;
```

[souce code untuk menampilkan nilai]

```
//Menampilkan Nilai Pada Setiap Variabel
 System.out.println("Nilai Byte \t= " +iniByte );
 System.out.println("Nilai Short \t= " +iniShort);
 System.out.println("Nilai Interger \t= "+iniInt);
 System.out.println("Nilai Long \t= " +iniLong);
 System.out.println("Nilai Float \t= " +iniFloat);
 System.out.println("Nilai Double \t= "+iniDouble);
 System.out.println("Nilai Char \t= " +iniChar);
 System.out.println("Nilai Boolean Benar : "+iniTrue);
 System.out.println("Nilai Boolean True  : "+iniFalse);
```

```
run:
Nilai Byte      = 127
Nilai Short     = 32727
Nilai Interger  = 2147483647
Nilai Long      = 10000000000
Nilai Float     = 3.14
Nilai Double    = 3.14
Nilai Char      = C
Nilai Boolean Benar : true
Nilai Boolean True  : false
BUILD SUCCESSFUL (total time: 0 seconds)
```

Praktikum 2:
Variable & Konstanta

Variable Pada JAVA

- **Definisi : tempat** atau **kontainer** yang digunakan untuk **menyimpan suatu nilai(data)**.

- Java adalah bahasa bertipe **statis (static)**, sehingga sebuah variable hanya bisa digunakan untuk menyimpan nilai **dengan tipe data yang sama**, Tidak bisa **berubah-ubah (dinamis)** seperti di bahasa pemrograman PHP atau JavaScript

- **Cara mendeklarasikan variable Pada JAVA :**

```
<tipe_data> <nama_variable>;
```

- Untuk pendefinisian nama variable, ada beberapa Aturan dan kebiasaan dari programmer Java yang harus diperhatikan.

Aturan Penamaan Variabel

- **Aturan Penamaan :**
  - Nama variable harus diawali oleh huruf, garis_bawah (_) atau tanda dolar ($).
  - Tidak boleh memiliki spasi, atau tanda baca
  - Tidak boleh menggunakan keyword java, di bawah ini

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

* not used
** added in 1.2
*** added in 1.4
**** added in 5.0

Panduan lain dalam penulisan variable adalah :

- Awali nama variable dengan huruf kecil. Untuk kata ke dua dan selanjutnya awali dengan huruf besar. (contoh: `myVariable`).
- Pilihlan nama yang mengindikasikan isi variablenya. Jangan hanya x, y atau z.

[**sumber :** Modul PBL – Andri Heryandi,M.T]

Variable Pada JAVA (3)

Setelah Dideklarasikan. Variabel bisa diakses debgab memanggil nama variable nya pada program. Di JAVA sendiri ada 3 jenis cara pengaksesan variable berdasarkan lingkupnya (scope):

1. Variable Lokal

Variable lokal tercipta ketika dideklarasikan didalam sebuah method atau blok progam. Variable lokal akan hilang dari memori ketika method tersebut selesai dieksekusi.

2. Variable Instance

Variable intstance merupakan variable yang dideklarasikan dalam kelas atau scara global, diluar method atau blok.Ruang lingkupnya bisa diakse secara global sesuai akses modifier-nya . (konsep enkapsulsi – dibahas lebih lanjut nanti).

3. Variable Static

Variable Static dideklarasikan dengan kata kunci “static”. Sifatnya hampir mirip dengan variable instance namun variable ini dapat diakses langsung diluar kelas tampa harus kelas tersebut diinstansiasi terlebih dahulu

Contoh Program (2)

- **Penggalan Source code untuk contoh Variable**

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

      String nama;
      nama = "Chrismiha Hardyanto";

      System.out.println(nama);

      int umur = 28;
      String alamat = "Margahayu Raya";

      System.out.println(umur +" tahun");
      System.out.println("Alamat Saya adalah : " +alamat);
      System.out.println("");

      nama = "Fakhrian Fadlia"; /*Apabila Mengisi nilai baru pada
                                  variabel yang sama */
      System.out.println(nama);
  }
}
```

[Output Program]

```
run:
Chrismiha Hardyanto
27 tahun
Alamat Saya adalah : Margahayu Raya

Fakhrian Fadlia
BUILD SUCCESSFUL (total time: 0 seconds)
```

Konstanta Pada JAVA

Selain variable, terdapat juga **konstanta** pada JAVA. Konstanta digunakan untuk kebutuhan menyimpan nilai yang **tetap** atau **tidak pernah berubah**.

❑ **Bentuk Umum Mendeklarasikan Konstanta :**

```
final <tipe_data> <NAMA_KONSTANTA> = <nilai_konstanta>;
```

❑ **Contoh :**

```
public static void main(String[] args) {
    final double PHI = 3.14;    //Mendeklarasikan  konstanta PHI

}
```

[catatan : nama konstanta umumnya menggunakan UPPERCASE]

Praktikum 3:
Operator pada JAVA

Operator Pada JAVA

JAVA memiliki beberapa jenis **Operator** yang dapat digunakan untuk melakukan beberapa bentuk operasi tertentu. Berikut adalah beberapa operator umum yang sering digunakan pada JAVA :

- Operator Aritmatika
- Operator Relasional
- Operator Logika
- Operator Penugasan

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Operator Aritmatika

Operator aritmatika digunakan untuk melakukan operasi perhitungan/matematika

Beberapa contoh **Operator Aritmatika** yang dikenal oleh JAVA :

- + Penambahan
- - Pengurangan
- * Perkalian
- / Pembagian
- % Modulus (sisa hasil bagi)
- ++ Penambahan 1
- -- Pengurangan 1

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Operator Aritmatika (2)

Contoh Implementasi Program untuk **Operator Aritmatika** pada JAVA :

```
public static void main(String[] args) {
    int a = 10;
    int b = 20;
    int c = 25;
    int d = 25;

    System.out.println("a + b = " + (a + b));
    System.out.println("a -b = " + (a - b) );
    System.out.println("a * b = " + (a * b) );
    System.out.println("b / a = " + (b / a) );
    System.out.println("b % a = " + (b % a) );
    System.out.println("c % a = " + (c % a) );
    System.out.println("a++   = " +  (a++) );
    System.out.println("b--= " +  (a--) );
    System.out.println("d++   = " +  (d++) );
    System.out.println("++d   = " +  (++d) );

}
```

```
run:
a + b = 30
a -b = -10
a * b = 200
b / a = 2
b % a = 0
c % a = 5
a++   = 10
b--= 11
d++   = 25
++d   = 27
BUILD SUCCESSFUL (total time: 0 seconds)
```

Operator Relasional

Operator relasional digunakan untuk melakukan operasi perbandingan dari 2 nilai atau lebih. Nilai yang dihasilkan dari operator relasional adalah **Boolean**.

Beberapa contoh **Operator relasional** yang dikenal oleh JAVA :

- == Perbandingan sama dengan
- > Perbandingan lebih besar dari
- < Perbandingan lebih kecil dari
- >= Perbandingan lebih besar atau sama dengan dari
- <= Perbandingan lebih kecil atau sama denan dari
- != Perbandingan tidak sama dengan

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Operator Relasional (2)

Contoh Implementasi **Operator Relasional** pada JAVA :

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

    int a = 10;
    int b = 20;

    System.out.println("a == b = " + (a == b) );
    System.out.println("a != b = " + (a != b) );
    System.out.println("a > b  = " + (a > b) );
    System.out.println("a < b  = " + (a < b) );
    System.out.println("b >= a = " + (b >= a) );
    System.out.println("b <= a = " + (b <= a) );
}
```

```
run:
a == b = false
a != b = true
a > b  = false
a < b  = true
b >= a = true
b <= a = false
BUILD SUCCESSFUL (total time: 0 seconds)
```

Operator Logika

Operator logika digunakan untuk melakukan operasi logika pada 2 buah nilai atau lebih . Nilai yang dihasilkan dari operator relasional adalah **Boolean**.

Beberapa contoh **Operator logika** yang dikenal oleh JAVA :

- && : Operasi logika AND
- || : Operasi logika OR
- ! : Operasi logika NOT

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Operator Logika (2)

Contoh Implementasi **Operator Logika** pada JAVA :

```
public static void main(String[] args) {
    boolean a = true;
    boolean b = false;

    System.out.println("a && b     = " + (a&&b));
    System.out.println("a || b     = " + (a||b) );
    System.out.println("!a         = " + !a);
    System.out.println("!(a && b) = " + !(a && b));

}
```

```
run:
a && b     = false
a || b     = true
!a         = false
!(a && b) = true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Operator Penugasan

Operator penugasan digunakan untuk melakukan operasi pengisian suatu data/nilai (seperti memasukan nilai kedalam variable).

Beberapa contoh **Operator penugasan** yang dikenal oleh JAVA :

- = : Pengisian nilai
- +=, -=, *=, /=, %= : Operasi aritmatika dengan nilai tertentu **[Augmented Assignments]**

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Operator Penugasan (2)

Contoh Implementasi **Operator Logika** pada JAVA :

```
public static void main(String[] args) {
    int a = 10;
    int b = 20;
    int c = 0;

    c = a + b;
    System.out.println("c = a + b = " + c );
    c += a ;
    System.out.println("c += a  = " + c );
    c -= a ;
    System.out.println("c -= a = " + c );
    c *= a ;
    System.out.println("c *= a = " + c );
    a = 10;c = 15;c /= a ;
    System.out.println("c /= a = " + c );

}
```

```
run:
c = a + b = 30
c += a  = 40
c -= a = 30
c *= a = 300
c /= a = 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Praktikum 4:
Struktur Dasar Runtunan

Struktur Runtunan

Struktur runtunan adalah proses algoritma yang dilakukan **secara beruntun(linear) dari langah ke-1 sampai langkah ke-N**. Tiap barisnya dikerjakan satu-persatu tampa ada percabangan atau perulangan didalamnya.

[Merupakan struktur yang selalu ada didalam program]

Contoh Runtunan Pada JAVA

Implementasi **Struktur Runtunan** pada JAVA : "**Menghitung Nilai Akhir Mahasiswa"**

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

        String nama, kelas;
        int nilaiTugas, nilaiUts, nilaiUas;
        double nilaiAkhir;

        nama = "Chrismikha Hardyanto";
        kelas = "IF6";
        nilaiTugas = 80;
        nilaiUts = 75;
        nilaiUas = 85;

        //Bobot = Tugas 30%, UTS 30%, UAS 40%
        nilaiAkhir = (0.3*nilaiTugas)+(0.3*nilaiUts)+(0.4*nilaiUas);
        System.out.println("Nama Mahasiswa \t\t= " +nama);
        System.out.println("Kelas Mahasiswa \t= " +kelas);
        System.out.println("Tampil Nilai Akhir \t= " +nilaiAkhir);

    }
}
```

"Seluruh perintah dijalankan secara berurutan (atas – bawah)"

```
run:
Nama Mahasiswa          = Chrismikha Hardyanto
Kelas Mahasiswa         = IF6
Tampil Nilai Akhir      = 80.5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Praktikum 5:
Struktur Dasar Percabangan

Struktur Percabangan

Struktur percabangan memungkinkan program kita bisa mengesekusi **kode program tertentu** ketika suatu **kondisi terpenuhi.**

Ada 3 bentuk umum dari struktur percabangan yang dikenal oleh JAVA, yaitu :

1. **If Statement**
2. **Switch statement**
3. **Ternary Operator**

[sumber : Modul PBL – Andri Heryandi,M.T**]**

IF Statement

- Dalam JAVA, **if** adalah salah satu kata kunci yang digunakan untuk membuat **struktur percabangan** didalam program

- Percabangan if akan mengesekusi **kode program tertentu** ketika suatu **kondisi terpenuhi (TRUE)**. Begitu juga sebaliknya, Jika kondisi **tidak terpenuhi (FALSE)** maka tidak akan dieksekusi.

- Jenis – jenis if statement pada JAVA :

1. **if statement untuk 1 kemungkinan aksi**
2. **if statement untuk 2 kemungkinan aksi**
3. **If statement untuk banyak kemungkinan aksi**

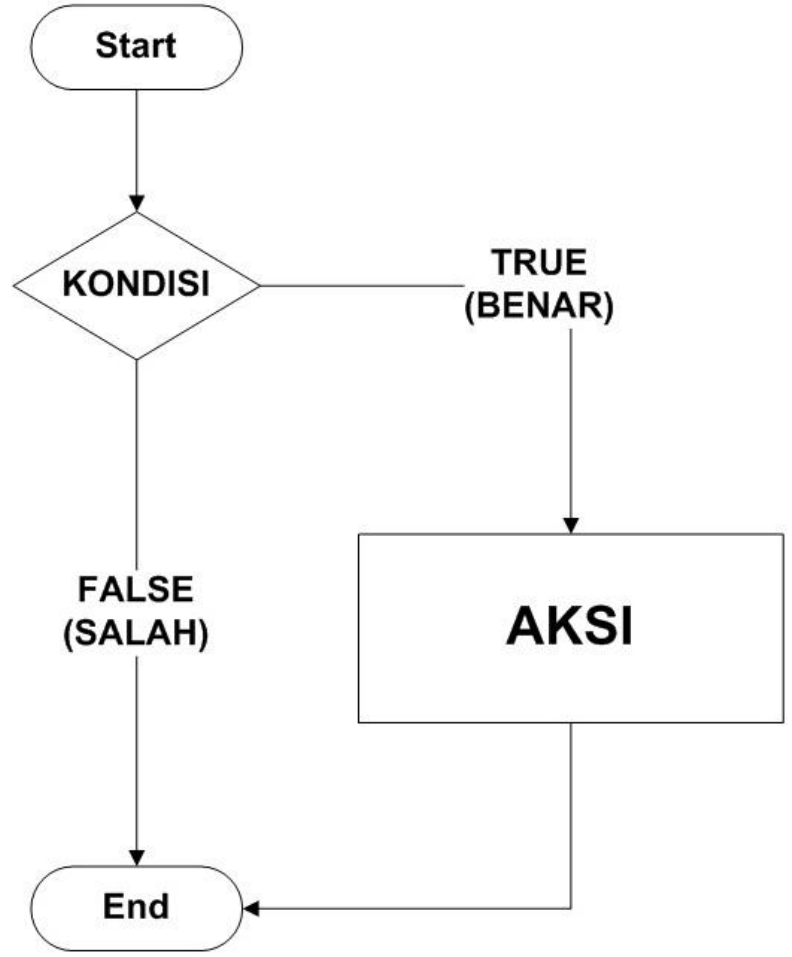
IF Statement (1 Aksi)

- Bentuk umum **sintaknya** adalah :

```
if (kondisi)
    baris perintah;
```

Jika aksi yang akan dieksekusi **terdapat lebih dari 1 baris perintah**, maka badan percabangan harus dibuat didalam block program / diapit dengan **kurung kurawal** (**{}**).

```
if (kondisi) {
    baris perintah-1;
    baris perintah-2;
    baris perintah ke-n;
}
```



Contoh if (1 Aksi) Pada JAVA

Implementasi **if statement dengan 1 aksi** pada JAVA

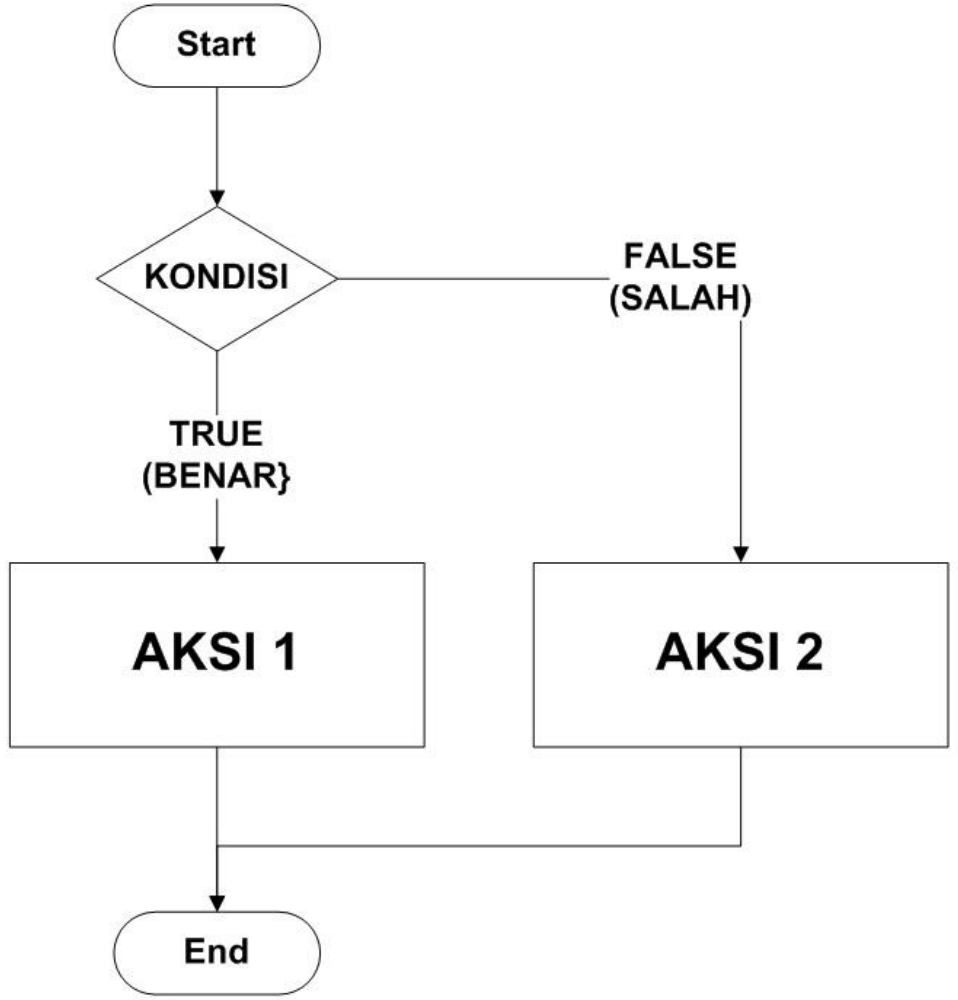
```
public static void main(String[] args) {

    String nama, kelas;
    int nilaiTugas, nilaiUts, nilaiUas;
    double nilaiAkhir;

    nama = "Chrismikha Hardyanto";
    kelas = "IF6";
    nilaiTugas = 80;
    nilaiUts = 75;
    nilaiUas = 85;

    //Bobot = Tugas 30%, UTS 30%, UAS 40%
    nilaiAkhir = (0.3*nilaiTugas)+(0.3*nilaiUts)+(0.4*nilaiUas);
    System.out.println("Nama Mahasiswa \t\t= " +nama);
    System.out.println("Kelas Mahasiswa \t= " +kelas);
    System.out.println("Tampil Nilai Akhir \t= " +nilaiAkhir);

    //Menambahkan percabangan untuk mengecek kelulusan mahasiswa
    if (nilaiAkhir >=45){
        String keterangan = "Lulus";
        System.out.println(nama +" Dinyatakan " +keterangan);
    }

}
```

```
run:
Nama Mahasiswa          = Chrismikha Hardyanto
Kelas Mahasiswa         = IF6
Tampil Nilai Akhir      = 80.5
Chrismikha Hardyanto Dinyatakan Lulus
BUILD SUCCESSFUL (total time: 0 seconds)
```

IF Statement (2 Aksi)

- Block if akan dieksekusi ketika kondisi bernilai **benar (true)**
- Terkadang kita ingin melakukan eksekusi kode program tertentu jika kondisi dari if **bernilai salah (false)**. Hal ini bisa dilakukan dengan menggunakan **kata kunci else** pada if statement
- Bentuk umum **sintaknya** adalah :

```
if (kondisi) {
    badan percabangan aksi ke-1;
} else {
    badan percabangan aksi ke-1;
}
```



Contoh if (2 Aksi) Pada JAVA

Implementasi **statement if dengan 2 aksi** pada JAVA

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

    String nama, kelas;
    int nilaiTugas, nilaiUts, nilaiUas;
    double nilaiAkhir;

    nama = "Chrismikha Hardyanto";
    kelas = "IF6";
    nilaiTugas = 80;
    nilaiUts = 0;
    nilaiUas = 0;

    //Bobot = Tugas 30%, UTS 30%, UAS 40%
    nilaiAkhir = (0.3*nilaiTugas)+(0.3*nilaiUts)+(0.4*nilaiUas);
    System.out.println("Nama Mahasiswa \t\t= " +nama);
    System.out.println("Kelas Mahasiswa \t= " +kelas);
    System.out.println("Tampil Nilai Akhir \t= " +nilaiAkhir);
```

```
    //Menambahkan percabangan untuk mengecek kelulusan mahasiswa
    if (nilaiAkhir >=45){
        String keterangan = "Lulus";
        System.out.println(nama +" Dinyatakan " +keterangan);
    } else {
        System.out.println("Silakan dicoba lagi Tahun Depan");
    }

}
```

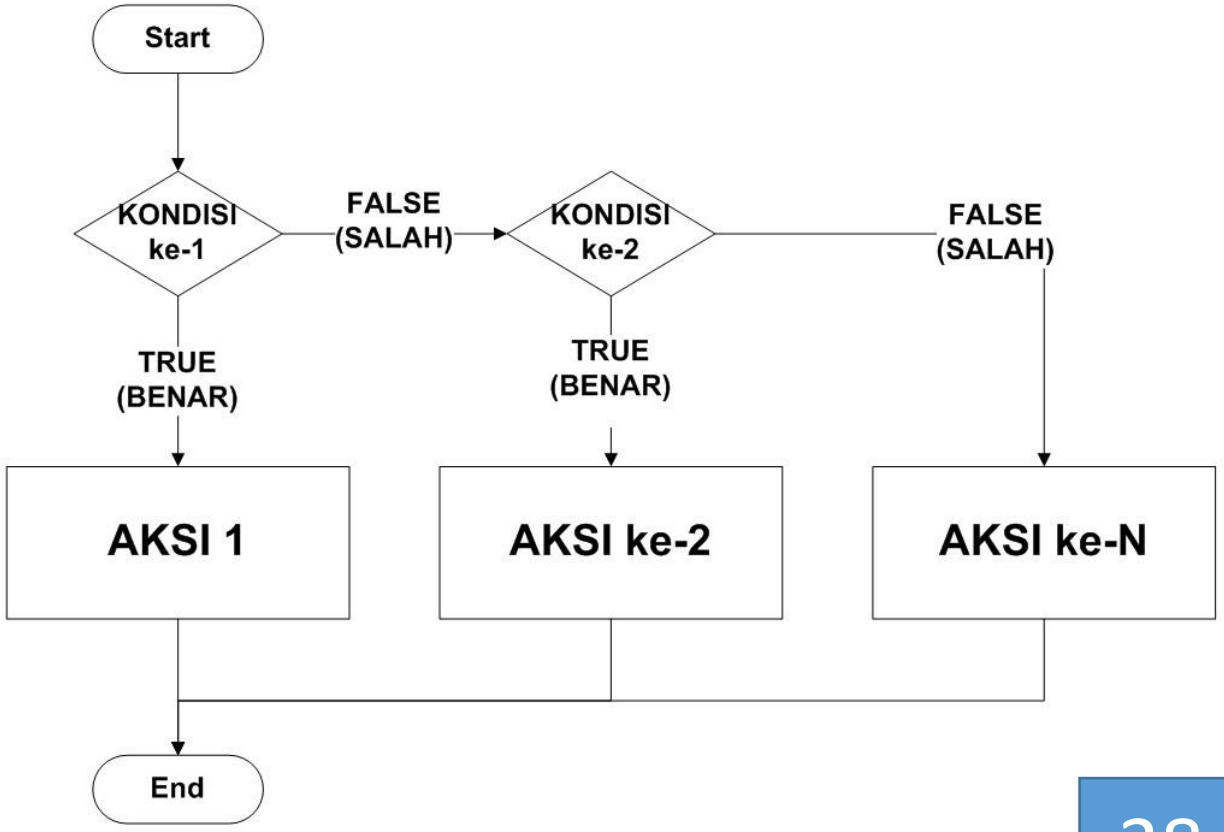
```
run:
Nama Mahasiswa          = Chrismikha Hardyanto
Kelas Mahasiswa         = IF6
Tampil Nilai Akhir      = 24.0
Silakan dicoba lagi Tahun Depan
BUILD SUCCESSFUL (total time: 0 seconds)
```

IF Statement (N Aksi)

- Terkadang kita ingin membuat kondisi dengan **banyak aksi (>2)** yang dapat dieksekusi oleh program. Untuk kasus seperti ini, Di JAVA kita bisa menggunakan **ekspresi else if didalam if statement**

- Bentuk umum **sintaknya** adalah :

```
if (kondisi aksi ke-1) {
    badan percabangan aksi ke-1;
} else if (kondisi aksi ke-2) {
    badan percabangan aksi ke-2;
} else if (kondisi aksi ke-n) {
    badan percabangan aksi ke-b;
} else {
    badan percabangan jika seluruh
    kondisi bernilai salah;
}
```



Contoh if (N Aksi) Pada JAVA

Implementasi **if statement dengan N aksi** pada JAVA

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

    String nama, kelas;
    int nilaiTugas, nilaiUts, nilaiUas;
    double nilaiAkhir;
    char indeks;

    nama = "Chrismikha Hardyanto";
    kelas = "IF6";
    nilaiTugas = 80;
    nilaiUts = 75;
    nilaiUas = 85;

    //Bobot = Tugas 30%, UTS 30%, UAS 40%
    nilaiAkhir = (0.3*nilaiTugas)+(0.3*nilaiUts)+(0.4*nilaiUas);
    System.out.println("Nama Mahasiswa \t\t= " +nama);
    System.out.println("Kelas Mahasiswa \t= " +kelas);
    System.out.println("Tampil Nilai Akhir \t= " +nilaiAkhir);
    System.out.println("");
    //Menambahkan percabangan untuk mengecek kelulusan mahasiswa
    if (nilaiAkhir >=45){
        String keterangan = "Lulus";
        System.out.println(nama +" Dinyatakan " +keterangan);
    } else {
        System.out.println("Silakan dicoba lagi Tahun Depan");
    }

    //menambahkan percabangan untuk menentukan indeks nilai
    if((nilaiAkhir >=80) && (nilaiAkhir <=100)){
        indeks = 'A';
        System.out.println("Indeks Nilai Anda : " +indeks);
    } else if ((nilaiAkhir >=65) && (nilaiAkhir < 80)){
        indeks = 'B';
        System.out.println("Indeks Nilai Anda : " +indeks);
    } else if ((nilaiAkhir >=45) && (nilaiAkhir < 65)){
         indeks = 'C';
        System.out.println("Indeks Nilai Anda : " +indeks);
    } else if ((nilaiAkhir >=30) && (nilaiAkhir < 45)){
        indeks = 'D';
        System.out.println("Indeks Nilai Anda : " +indeks);
    } else {
        indeks = 'E';
        System.out.println("Indeks Nilai Anda : " +indeks);
    }
}
```

Contoh if (N Aksi) Pada JAVA

Implementasi **if statement dengan N aksi** pada JAVA

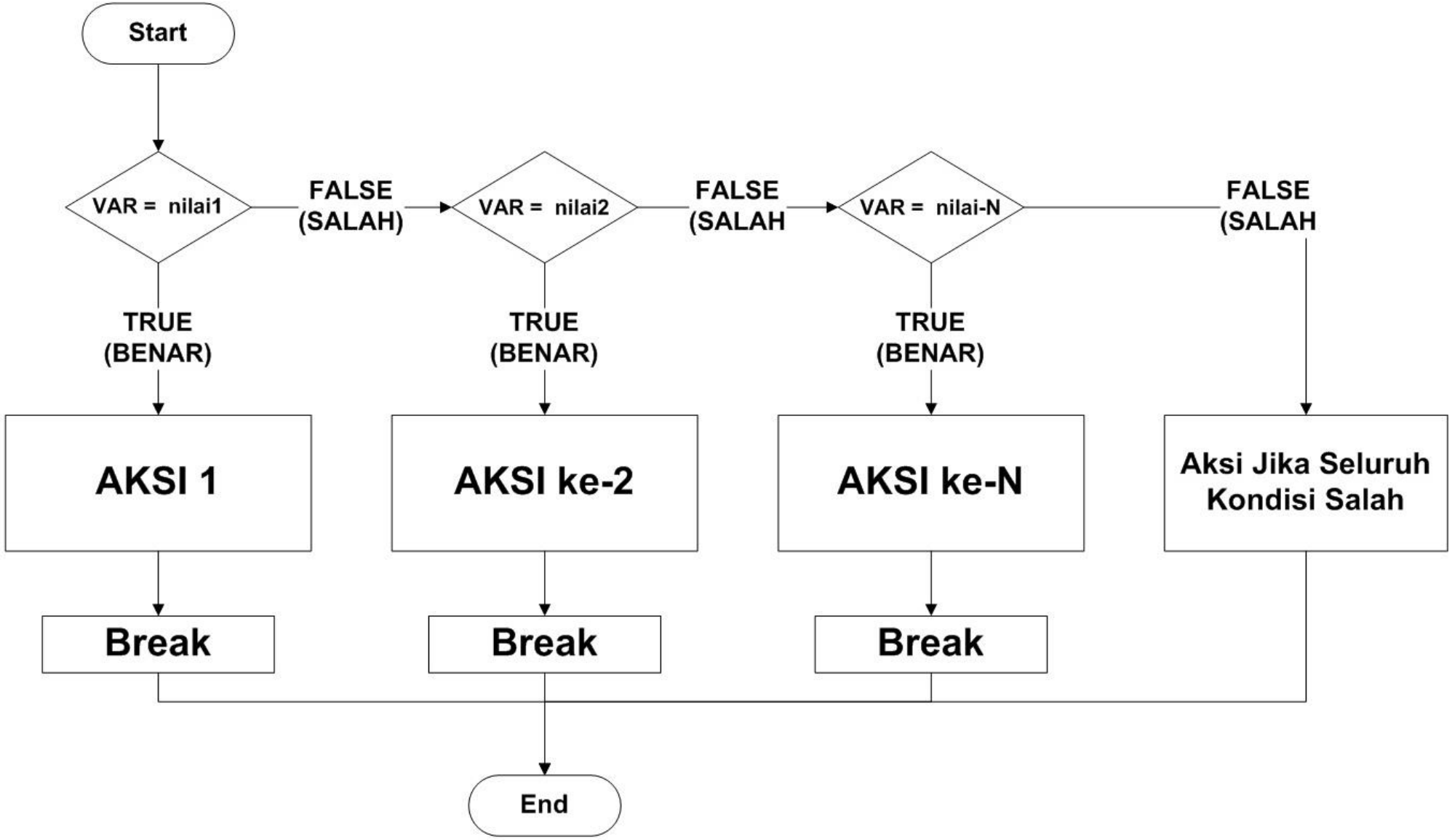
```
run:
Nama Mahasiswa            = Chrismikha Hardyanto
Kelas Mahasiswa           = IF6
Tampil Nilai Akhir        = 80.5

Chrismikha Hardyanto Dinyatakan Lulus
Indeks Nilai Anda : A
BUILD SUCCESSFUL (total time: 0 seconds)
```

Switch Statement

- **switch** dapat digunakan sebagai **alternatif** untuk membuat percabangan pada JAVA. Fungsinya sama dengan if namun penulisan perintahnya **lebih sederhana**.
- Terkadang kita hanya butuh menggunakan kondisi sederhana saja didalam if statement, seperti hanya menggunakan perbandingan == . (**kondisi di switch hanya untuk perbandingan ==)**
- **Sintak** dari pernyataan **switch-case** adalah sebagai berikut :

```
switch(kondisi){
        case nilai1 : Aksi1; break;
        case nilai2 : Aksi2; break;
        case nilai3 : Aksi3; break;
        default : Aksi_default;
}
```

Switch Statement



Contoh Switch Pada JAVA

Implementasi **switch statement dengan N aksi** pada JAVA

```
//Menambahkan prcabangan untuk menentukan keterangan dari indeks
switch (indeks){
    case 'A':   System.out.println("Anda Lulus Dengan Sangat Baik");
                break;
    case 'B':   System.out.println("Anda Lulus Dengan Baik");
                break;
    case 'C':   System.out.println("Anda Lulus Dengan Cukup");
                break;
    case 'D':   System.out.println("Anda Lulus Dengan Kurang Cukup");
                break;
    default: System.out.println("Anda Tidak Lulus");
}
```

```
Nama Mahasiswa          = Chrismikha Hardyanto
Kelas Mahasiswa         = IF6
Tampil Nilai Akhir      = 80.5

Chrismikha Hardyanto Dinyatakan Lulus
Indeks Nilai Anda : A
Anda Lulus Dengan Sangat Baik
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ternary Operator (?:)

- Didalam JAVA, **Ternary Operator (?:)** bisa digunakan sebagai cari lain untuk **membuat percabangan.** Ternary Operator adalah **operator sederhana** dari **if statement**.

- Bentuk umum dari **sintak** nya adalah sebagai berikut :

```
(Kondisi) ? pernyataan1 : pernyataan2 ;
```

Baris perintah di atas dapat dibaca sebagai "Jika **kondisi(ekspresi logika)** bernilai **true/benar**, maka operator ini akan mengembalikan **nilai pernyataan1** dan jika **kondisi(ekspresi logika)** bernilai **false/salah** maka akan operator mengembalikan nilai **pernyataan2**

Contoh Ternary Operator Pada JAVA

Implementasi **ternary operator dengan N aksi** pada JAVA

```
//Menambahkan percabangan untuk mengecek kelulusan mahasiswa
/*
if (nilaiAkhir >=45){
    String keterangan = "Lulus";
    System.out.println(nama +" Dinyatakan " +keterangan);
} else {
    System.out.println("Silakan dicoba lagi Tahun Depan");
} */
//Alternatif percabangan if dengan ternary operator
String keterangan;
keterangan = nilaiAkhir >=45 ? "Lulus":"Mencoba lagi Tahun Depan";
System.out.println(nama +" Dinyatakan " +keterangan);
```

```
Nama Mahasiswa          = Chrismikha Hardyanto
Kelas Mahasiswa         = IF6
Tampil Nilai Akhir      = 80.5

Chrismikha Hardyanto Dinyatakan Lulus
```

Dapat dilihat dengan ternary operator, penulisan statement if sebelumnya menjadi lebih sederhana

Praktikum 6:
Struktur Dasar Perulangan

Struktur Perulangan

Struktur perulangan memungkinkan program kita bisa mengesekusi **kode program yang sama berulang kali** selama **kondisi terpenuhi.**

Ada 3 bentuk statement perulangan yang dikenal oleh JAVA, yaitu :

1. Perulangan FOR

2. Perulangan WHILE

3. Perulangan DO WHILE

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Perulangan FOR

- **for** adalah salah satu **kata kunci** untuk membuat perulangan. Memungkinkan blok kode yang terdapat didalam badan for akan **selalu diulangi** selama **kondisi for terpenuhi**

- **contoh** sintaks dari perulangan for adalah sebagai berikut :

```
for (init_statement; kondisi_for; post_statement)
{
        badan perulangan;
}
```

- **Keterangan :**

Init_statement = nilai/counter awal dari perulangan. Hanya dieksekusi 1 kali diawal perulangan

Kondisi_for = menentukan berapa kali perulangan dilakukan. Kondisi akan dicek setiap iterasi

Post_statement = menambah/mengurangi nilai perulangan sebanyak 1 setiap iterasi perulangan

Contoh perulangan FOR

Implementasi **perulangan for** pada JAVA

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

    //Coba Perulangan dengan statement for
    System.out.println("Coba Perulangan Java dengan for statement");

    String keterangan1;
    keterangan1 = "Saya Sedang Belajar perulangan for pada JAVA";
    for(int i = 1;i <= 10; i++){

        System.out.println(keterangan1 + " ke-" +i);

    }

}
```

```
run:
Coba Perulangan Java dengan for statement
Saya Sedang Belajar perulangan for pada JAVA ke-1
Saya Sedang Belajar perulangan for pada JAVA ke-2
Saya Sedang Belajar perulangan for pada JAVA ke-3
Saya Sedang Belajar perulangan for pada JAVA ke-4
Saya Sedang Belajar perulangan for pada JAVA ke-5
Saya Sedang Belajar perulangan for pada JAVA ke-6
Saya Sedang Belajar perulangan for pada JAVA ke-7
Saya Sedang Belajar perulangan for pada JAVA ke-8
Saya Sedang Belajar perulangan for pada JAVA ke-9
Saya Sedang Belajar perulangan for pada JAVA ke-10
BUILD SUCCESSFUL (total time: 0 seconds)
```

Perulangan WHILE

- **Perulangan while** akan mengulang seluruh perintah yang terdapat didalam badan perulangan selama **kondisi perulangan (ekspresi boolean) bernilai true**. Ketika kondisi perulangan bernilai **false**, maka eksekusi program akan melanjutkan ke baris perintah **setelah badan while.**

- Di perulangan while, hanya terdapat **kondisi perulangan** tampa ada init statement dan post statement

- **Sintak** perulangan while adalah sebagai berikut :

```
while (kondisi_perulangan)
{
        badan perulangan;
}
```

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Contoh perulangan WHILE

Implementasi **perulangan while** pada JAVA

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

     //Coba Perulangan dengan statement while
    System.out.println("Coba Perulangan Java dengan while statement");

    String keterangan2;
    keterangan2 = "Saya Sedang Belajar perulangan while pada JAVA";
    int counter = 1;
    while (counter <= 10) {

        System.out.println(keterangan2 + " ke-" +counter);
        counter++;
    }

}
```

```
run:
Coba Perulangan Java dengan while statement
Saya Sedang Belajar perulangan while pada JAVA ke-1
Saya Sedang Belajar perulangan while pada JAVA ke-2
Saya Sedang Belajar perulangan while pada JAVA ke-3
Saya Sedang Belajar perulangan while pada JAVA ke-4
Saya Sedang Belajar perulangan while pada JAVA ke-5
Saya Sedang Belajar perulangan while pada JAVA ke-6
Saya Sedang Belajar perulangan while pada JAVA ke-7
Saya Sedang Belajar perulangan while pada JAVA ke-8
Saya Sedang Belajar perulangan while pada JAVA ke-9
Saya Sedang Belajar perulangan while pada JAVA ke-10
BUILD SUCCESSFUL (total time: 0 seconds)
```

Perulangan DO..WHILE

- **Perulangan do while** akan mengulang seluruh perintah yang terdapat didalam badan perulangan selama **kondisi perulangan (ekspresi boolean) bernilai true**. **(sama seperti while)**
- Perbedaan do while dengan while terletak pada **posisi pemeriksaan kondisi perulangan-nya**. Kondisi pada do while diletakan **setelah badan perulangan** sehingga di**PASTIKAN** badan perulangan akan dieksekusi **1 kali**.
- **Sintak** perulangan do while adalah sebagai berikut :

```
do {
        badan perulangan;
}
while (kondisi_perulangan);
```

[sumber : Modul PBL – Andri Heryandi,M.T**]**

Contoh perulangan DO..WHILE

Implementasi **perulangan do..while** pada JAVA

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

     //Coba Perulangan dengan statement while
    System.out.println("Coba Perulangan Java dengan do..while statement");

    String keterangan3;
    keterangan3= "Saya Sedang Belajar perulangan do..while pada JAVA";
    int counter2 = 1;
    do{

       System.out.println(keterangan3 + " ke-" +counter2);
       counter2++;

    }while(counter2 <= 10);
}
```

```
run:
Coba Perulangan Java dengan do..while statement
Saya Sedang Belajar perulangan do..while pada JAVA ke-1
Saya Sedang Belajar perulangan do..while pada JAVA ke-2
Saya Sedang Belajar perulangan do..while pada JAVA ke-3
Saya Sedang Belajar perulangan do..while pada JAVA ke-4
Saya Sedang Belajar perulangan do..while pada JAVA ke-5
Saya Sedang Belajar perulangan do..while pada JAVA ke-6
Saya Sedang Belajar perulangan do..while pada JAVA ke-7
Saya Sedang Belajar perulangan do..while pada JAVA ke-8
Saya Sedang Belajar perulangan do..while pada JAVA ke-9
Saya Sedang Belajar perulangan do..while pada JAVA ke-10
BUILD SUCCESSFUL (total time: 0 seconds)
```

Perintah Break & Continue

- **Perintah break** digunakan jika Anda ingin **keluar dari badan perulangan.** **(stop perulangan)**

- **Perintah continue** digunakan jika anda ingin **mengabaikan perintah** didalam badan perulangan dan **melanjutkan ke iterasi perulangan selanjutnya**. **(skip perulangan)**

- **Perintah** break & continue ditulis **didalam badan perulangan**.

Contoh break pada perulangan

Implementasi kata kunci **break** pada perulangan JAVA

```
//Coba perulangan dengan kata kunci break
System.out.println("Mencoba kata kunci break didalam perulangan");
String keterangan4 = "Saya Sedang Belajar Perulangan";

int counter4 = 1;
while(true){

    System.out.println(keterangan4 +" ke- " +counter4);
    counter4++;

    if(counter4 > 10){
        break;
    }
}
```

```
Mencoba kata kunci break didalam perulangan
Saya Sedang Belajar Perulangan ke- 1
Saya Sedang Belajar Perulangan ke- 2
Saya Sedang Belajar Perulangan ke- 3
Saya Sedang Belajar Perulangan ke- 4
Saya Sedang Belajar Perulangan ke- 5
Saya Sedang Belajar Perulangan ke- 6
Saya Sedang Belajar Perulangan ke- 7
Saya Sedang Belajar Perulangan ke- 8
Saya Sedang Belajar Perulangan ke- 9
Saya Sedang Belajar Perulangan ke- 10
BUILD SUCCESSFUL (total time: 0 seconds)
```

*Dapat dilihat walaupun kondisi perulangan selalu **TRUE** , namun jika didalam badan perulangan ketemu **kata kunci break** maka perulangan **langsung terhenti**

Contoh continue pada perulangan

Implementasi kata kunci **continue** pada perulangan JAVA

```
//Coba perulangan dengan kata kunci continue
System.out.println("Nenampilkan 20 bilangan ganjil pertama");

for(int bilangan = 1; bilangan <=20; bilangan++){
    //Buatlah kondisi untuk mengecek apakah a adalab bilangan genap
    if(bilangan % 2 == 0){
      continue;
    }

    System.out.println(bilangan);
}
```

```
Nenampilkan 20 bilangan ganjil pertama
1
3
5
7
9
11
13
15
17
19
BUILD SUCCESSFUL (total time: 0 seconds)
```

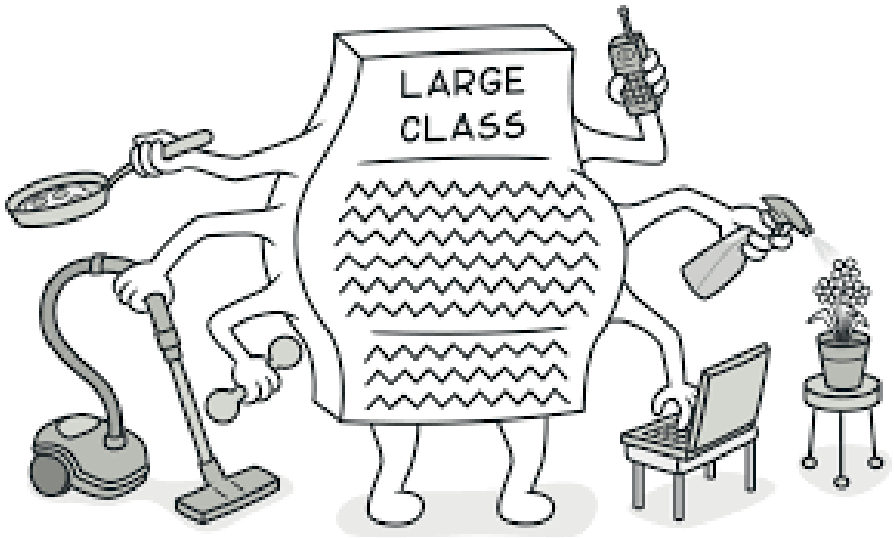
*Dapat dilihat walaupun perulangan diatas kondisi nya adalah melakukan **perulangan 20 kali**, namun tidak semua nilai pada bilangan dicetak ke layar (**hanya bilangan ganjil saja**)

Praktikum 7:
Prosedur & Fungsi

Prosedur & Fungsi pada JAVA

Di bahasa pemrograman lain, **method** juga disebut sebagai **fungsi (function)** atau **prosedur (procedure)**. Manfaatnya sama yaitu membantu kita untuk **membagi - bagi program** menjadi blok - blok sub program yang lebih kecil.

Method adalah **block kode program** yang akan **berjalan** saat kita **panggil** dan tidak akan pernah dieksekusi apabila tidak dipanggil.

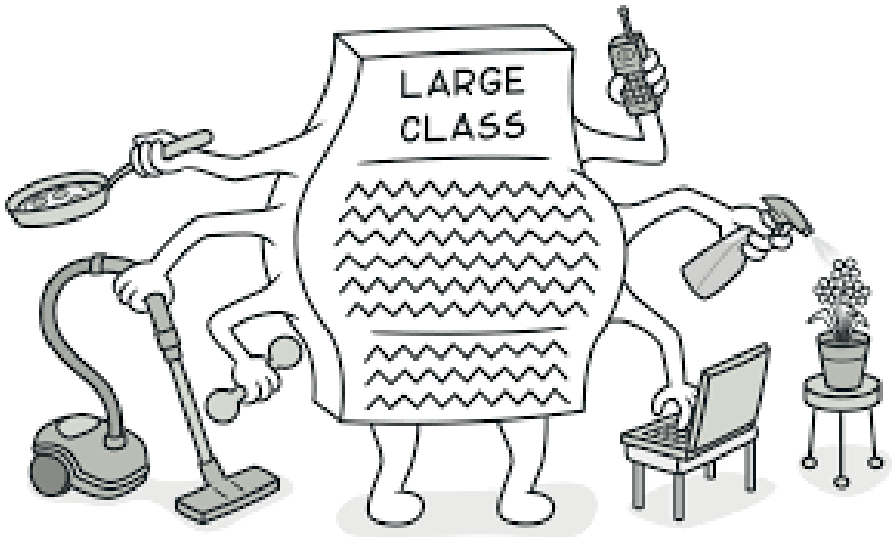


Mendefinisikan Method

Untuk membuat method sederhana di JAVA, Kita bisa menggunakan **kata kunci void**, lalu diikuti dengan **nama method** ditambah **tanda kurung** dan diakhiri dengan membuat **block untuk badan methodnya (kurung kurawal)**

Contoh **Sintaks** untuk membuat method pada JAVA

```
void  namaMethod(parameter)
{
      block badan method;
}
```



Contoh Method Pada JAVA

Implementasi kata kunci **continue** pada perulangan JAVA

```
public class Method {

    public static void main(String[] args){

        cetakText();
    }


    static void cetakText(){
        for (int i = 1; i <= 5; i++){

            System.out.println("Hello World "+i);
        }
    }
}
```

Memanggil method

pendefinisian method

```
run:
Hello World 1
Hello World 2
Hello World 3
Hello World 4
Hello World 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

*untuk contoh diatas ada tambahan **kata kunci static**. Static digunakan agar method bisa dipanggil didalam method main (karena sifatnya juga static). Akan dibahas lebih lanjut di materi berikutnya

Setelah memahami konsep dasar pemrograman pada JAVA, mari kita berlatih

*Silakan buka modul praktikum pertemuan 2

Terima Kasih