Bab 2

Dasar Pemrograman Java

Kode yang ditulis dalam bahasa Java harus memenuhi kaidah-kaidah yang sudah ditentukan. Pada contoh ini, kita akan mempelajari bahasa yang digunakan untuk menuliskan bahasa Java, termasuk sintaks dan semantiknya.

2.1 Identifier dan Keyword

Dalam bahasa Java, penamaan elemen program harus mengikuti aturan. Penamaan identifier tidak boleh sama dengan keyword.

2.1.1 Identifier

Identifier adalah nama. Lebih lengkapnya, identifier merupakan nama yang digunakan untuk mengidentifikasi elemen program, misalnya nama variabel, nama konstanta, nama kelas, dan lain-lain.

Aturan identifier untuk Java adalah sebagai berikut:

- Identifier tidak boleh diambil dari keyword, atau true, false, null.
- Identifier boleh menggunakan huruf, angka 0-9, garis bawah (*underscore*), atau tanda dolar "$".
- Identifier harus dimulai dengan huruf, garis bawah, atau tanda Dollar "$".

Jika menyalahi ketentuan di atas, program Java tidak akan dapat dikompilasi karena error.

2.1.2 Keyword

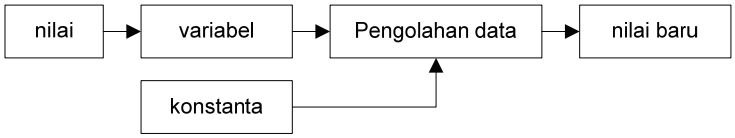
Keyword atau kata kunci merupakan kata yang mempunyai arti khusus sehingga tidak dapat digunakan sebagai *identifier*. Berikut daftar keyword dalam bahasa Java.

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

Daftar identifier di atas dapat berubah, seperti bertambah atau berkurang, tergantung perkembangan bahasa Java di masa depan.

2.2 Bekerja dengan Data

Program sering memanfaatkan konstanta ataupun variabel untuk pengolahan data. Setiap konstanta dan variabel dalam bahasa Java pasti mempunyai tipe data tertentu. Tipe data tersebut menentukan jenis dan nilai yang diperbolehkan untuk diberikan pada variabel dan konstanta tersebut.



2.2.1 Tipe Data Primitif

Semua nilai dalam Java adalah referensi terhadap objek. Namun, untuk fleksibilitas, Java tetap mempertahankan tipe data dengan nilai biasa atau bisa juga tipe data primitif (*primitive types*). Tipe data primitif adalah tipe data yang mempunyai nilai tertentu, bukan referensi class ataupun objek. Berdasarkan tipe datanya, data primitif dapat dikelompokkan menjadi empat:

- Tipe data integer atau bilangan bulat: byte, short, int, long.

- Tipe data float atau bilangan nyata: float, double.
- Tipe data char atau karakter: char.
- Tipe data Boolean: boolean.

Berikut keterangan besar nilai yang dapat diterapkan pada tipe-tipe data primitif di atas.

Tipe Data	Besar Storage	Nilai minimal	Nilai maksimal
byte	8 bit (1 byte)	–128	127
short	16 bit (2 byte)	–32768	32767
int	32 bit (4 byte)	–2147483648	2147483647
long	64 bit (8 byte)	–9223372036854775808	9223372036854775807
float	32 bit (4 byte)	±3.4E-38	±3.4E+38
double	64 bit (8 byte)	±1.7E-308	±1.7E+308
char	16 bit (2 byte)	\u0000	\uFFFF
boolean	1 bit		true atau false

Tipe Data Boolean

Boolean digunakan untuk menentukan suatu kondisi apakah benar (*true*) atau salah (*false*). Nilai boolean sering digunakan untuk mengatur alur program, terutama pada perulangan dan pencabangan.

```
boolean isLulus = true; // sudah lulus
boolean isCumlaude = false;    // tidak cumlaud
```

Tipe Data Char

Tipe karakter dalam Java mempunyai ukuran 16 bit atau setara dengan 2^{16}=65.536 kode. 256 kode pertama dalam tipe data char digunakan oleh karakter ASCII. Karakter ASCII merupakan karakter-karakter yang banyak digunakan dalam bahasa Inggris. Untuk mencari tahu informasi tentang karakter, Anda dapat membuka situs **www.unicode.org**.

Tipe data karakter harus didefinisikan menggunakan tanda petik satu (*apostrop*), misalnya:

```
char bs='B';
```

Dalam char terdapat escape character yang digunakan untuk menampilkan karakter khusus, seperti tab atau pergantian baris.

Escape Sequence	Keterangan
\b	Backspace
\t	Tab
\n	Linefeed
\f	Formfeed
\r	Carriage Return
\\	Backslash
\'	Single Quote
\"	Double Quote
ddd	Oktal, misal '\123' setara dengan huruf S
\u*dddd*	Karakter Unicod, misal '\u1234' setara dengan tanda tanya ?

Berikut kode untuk menampilkan tulisan: Aku berkata "Hidup Java!" pada command prompt.

```
System.out.println("Aku berkata \"Hidup Java! \"");
```

Tipe Data Integer

Integer merupakan bilangan bulat. Dalam bahasa Java terdapat empat buah tipe integer. Semuanya dapat bernilai negatif maupun positif: byte, short, int, dan long. Perbedaan keempat integer tersebut hanyalah pada ukurannya, yaitu mulai dari 8 bit, 16 bit, 32 bit, dan 64 bit. Semakin besar ukuran tipe data integer tersebut, semakin besar ukuran nilai yang dapat ditampung.

```
bit z = 1;
```

Tipe Data Floating Point

Floating point merupakan bilangan rasional. Dalam bahasa Java terdapat dua buah tipe data floating point: float dan double. Perbedaan keduanya hanya pada ukuran, yaitu masing-masing 32 bit dan 64 bit.

Penulisan bilangan floating point menggunakan tanda titik sebagai tanda desimal atau bisa juga menggunakan tanda eksponensial e atau E.

```
double a = 12.34; // 12,34
double b = .01;          // 0,01
double c = 1e-6          // 1x10^-6 atau 0,000006
double d = 5200000D      // 5,2x10^6
```

2.2.2 Variabel

Variabel digunakan untuk menyimpan data sehingga dapat diolah oleh program. Data yang disimpan dalam variabel bisa berupa referensi objek maupun tipe data primitif. Dalam bahasa Java, variabel harus dideklarasikan dengan menentukan nama variabel dan tipe data variabel itu sendiri. Sintaks yang digunakan:

Sintaks Deklarasi Variabel

```
[modifier] tipe identifier;
[modifier] tipe identifier1, identifier2, identifierxx;
```

Berikut contoh kode untuk deklarasi variabel.

```
float salary;
short x, y, z;
```

Sedangkan untuk memberikan nilai ke dalam variabel, digunakan operator assignment, berupa tanda sama dengan.

Sintaks Inisialisasi Variabel

```
[modifier] identifier = ekspresi;
[modifier] identifier = lateral;
```

Semua variabel dalam Java harus ditentukan tipenya, sedangkan nilai bisa ditentukan kemudian. Jika tipe data pada variabel tidak ditentukan, akan terjadi error dan program tidak dapat dikompilasi. Berikut beberapa contoh penggunaan variabel.

```
short b,
b = 2;
t = 1 + 2;               // Error, t belum dideklarasikan
```

Agar lebih sederhana, deklarasi dan inisialisasi variabel dapat dilakukan sekaligus dalam satu statement.

Sintaks Deklarasi dan Inisialisasi Variabel

```
[modifier] tipe identifier = ekspresi;
[modifier] tipe identifier1 = nilai1, identifier2 = nilai2;
```

Berikut contoh deklarasi sekaligus pemberian nilai pada variabel.

```
short b = 12;
short x = 12 + 10, c = 5;
char y = '1';
```

Sebelum digunakan untuk pengolahan data, variabel harus diberi nilai. Jika tidak, kompilasi akan gagal.

```
int angka;
System.out.println(angka);     // Error nilai angka tidak
diketahui
```

2.2.3 Konstanta

Konstanta digunakan untuk menyimpan data yang tidak akan kita ubah. Contoh konstanta adalah π. Dalam bahasa Java, konstanta dituliskan dengan menambahkan keyword **final** di depan tipe variabel. Sekali konstanta diberi nilai, maka nilai tersebut tidak dapat diubah lagi.

Sintaks Deklarasi dan Inisialisasi Konstanta

```
[modifier] final tipe identifier;
[modifier] final tipe identifier = ekspresi;
```

Deklarasi beberapa konstanta secara langsung juga dapat dilakukan.

Sintaks Deklarasi dan Inisialisasi Beberapa Konstanta

```
[modifier] final tipe identifier1, identifier2;
[modifier] final tipe identifier1 = nilai, identifier2 = nilai;
```

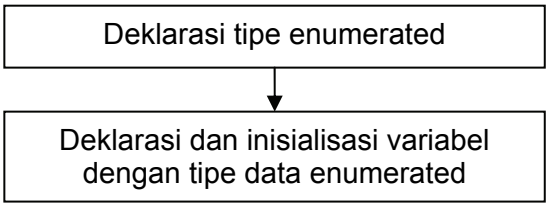
Berikut contoh kode program untuk konstanta.

```
final double PI = 3.14;
PI = 2;    // Error, kode tidak dapat dikompilasi!
final int Y;
Y = 11;    // OK
Y = 100;   // Error, x bernilai 12 dan tidak dapat diganti
```

Adanya konstanta membantu kita mengamankan agar program tetap berjalan dengan semestinya. Bayangkan jika kita menggunakan variabel π dan bukan konstanta. Jika nilai variabel π tidak sengaja diganti, hasil penghitungan program dapat menjadi salah semua.

2.2.4 Tipe Enumerated

Tipe enumerated (*enumerated type*) merupakan tipe data yang didefinisikan sendiri oleh programmer. Dengan demikian, diharapkan pengolahan data lebih fleksibel.



Tahap penggunaan tipe data enumerated

Tipe enumerated digunakan melalui dua tahap berikut:

1. **Deklarasi tipe data enumerated itu sendiri**. Pada tahap ini, ditentukan nama tipe data enumerated. Selain itu, ditentukan juga daftar nilai-nilai yang dapat diberikan ke dalam tipe data baru ini.
2. **Penggunaan tipe data enumerated**. Dilakukan dengan deklarasi dan inisialisasi variabel atau konstanta dengan tipe data enumerated.

Pada prakteknya, enumeration sering digunakan untuk membuat daftar nilai-nilai dalam jumlah terbatas. Berikut sintaks yang digunakan untuk mendefinisikan tipe enumerated. Perhatikan bahwa di bagian akhir **tidak** diakhiri dengan tanda titik koma ";".

Sintaks Deklarasi Tipe Enumerated

```
enum nama_tipe_enumerated {nilai1, nilai2, nilai3, nilaix}
```

Berikut contoh deklarasi tipe enumerated:

```
public enum Hari
 {SENIN, SELASA, RABU, KAMIS, JUMAT, SABTU, MINGGU}
public enum JK {LAKI, PEREMPUAN}
```

Sedangkan untuk mendeklarasikan variabel dengan tipe enumerated, dapat dilakukan dengan contoh berikut:

```
Hari hariUpacara = Hari.SENIN;
Hari hariLibur;
hariLibur = Hari.SENIN;
```

2.3 Statement

Kode program dapat dilihat sebagai kumpulan instruksi yang harus dikerjakan oleh komputer. Setiap instruksi ini disebut juga dengan statement. Setiap statement dalam bahasa Java diakhiri dengan tanda titik koma.

Beberapa statement dapat dikelompokkan ke dalam grup atau blok yang berbeda. Caranya, digunakan tanda kurung kurawal untuk mengawali dan mengakhiri grup.

2.3.1 Operator

Operator digunakan untuk melakukan pengolahan data, biasanya melibatkan konstanta dan variabel yang telah kita buat.

Jenis Operator	Simbol
Increment/decrement	++, –
Unary operators	+, -, ~, !, (cast)
Perkalian/pembagian/modulus	*, /, %
Penambahan/pengurangan	+, -
Operator shift	<<, >>, >>>
Perbandingan	<, <=, >, >=,= =, !=
Bitwise	AND, OR, XOR &, |, ^
Kondisional	AND, OR &&, ||
Operator Ternary	? :
Operator penugasan (*assignment*)	=

Operator Assignment

Assignment atau penugasan dilakukan menggunakan operator “=”. Sintaks yang digunakan adalah:

Sintaks Assignment

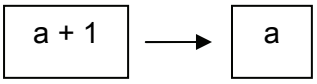
```
identifier = ekspresi;
identifier = literal;
```

Meskipun mirip, operasi assignment berbeda dengan operasi "sama dengan" dalam Matematika. Berikut contohnya:

```
a = a + 1;
```

Statement di atas bukan berarti a sama dengan a ditambah 1. Dalam Matematika, hal itu tidak mungkin.

Statement tersebut berarti: tambahkan nilai a dengan angka 1 lalu jadikan nilai tersebut menjadi nilai a yang baru.



Literal dan Ekspresi

Literal digunakan untuk memberi nilai pada suatu variabel ataupun konstanta. Literal tidak melibatkan operator.

Literal	Tipe data
2	int
0x1b	int (heksadesimal). Diawali dengan 0x
032	int (oktal). Diawali dengan 0
25.0D	double
true	boolean

Ekspresi melibatkan operator yang digunakan pada variabel, konstanta, ataupun nilai; tanpa menggunakan tanda sama dengan. Jadi, suatu ekspresi mempunyai nilai tertentu. Contoh ekspresi:

Ekspresi	Nilai
20 + 5	7
(10 * 2) + b	20 + b
"Hallo ini" + "adalah saya"	"Hallo ini adalah saya"

Operator Logika Boolean

Operator boolean digunakan untuk mengolah dan menghasilkan nilai true/false.

Nama	Arti	Simbol	Deskripsi
And	dan	&, &&	Bernilai benar hanya jika kedua ekspresi yang dievaluasi benar semua
Or	atau	|, ||	Bernilai benar jika salah satu ekspresi benar
Not	tidak	!	Bernilai benar jika ekspresi salah, dan salah jika ekspresi benar
XOR	XOR	^	Bernilai benar jika kedua ekspresi berbeda nilainya (satu benar dan satu salah)

Berikut hasil ketiga operator Boolean di atas.

A	B	A & B
true	true	true
true	false	false
false	true	false
false	false	false

A	B	A | B
true	true	true
true	false	true
false	true	true
false	false	false

A	!A
true	false
false	true

Tanda && dan & serta || dan | memiliki arti berbeda. & dan | akan tetap mengevaluasi kondisi apa pun yang terjadi. && akan menghentikan evaluasi kondisi jika salah satu ekspresi bernilai salah. Jika salah satu kondisi ekspresi salah, dipastikan hasil operasi logika bernilai salah. Sedangkan || akan menghentikan evaluasi kondisi jika salah satu ekspresi bernilai benar. Jika salah satu kondisi ekspresi benar, dipastikan hasil operasi logika bernilai benar.

Mari kita buat contoh kode program untuk melakukan operasi boolean dan menampilkan hasilnya ke layar monitor.

Listing program

```
class OperatorLogika{
 public static void main(String[] args) {
  boolean B = true;
  boolean S= false;
  System.out.println("Operator OR");
  System.out.println("Benar || Benar : " +(B || B));
  System.out.println("Benar || Salah : " +(B || S));
  System.out.println("Salah || Benar : " +(S || B));
  System.out.println("Salah || Salah : " +(S || S));
  System.out.println("Operator AND");
  System.out.println("Benar && Benar : " +(B && B));
  System.out.println("Benar && Salah : " +(B && S));
  System.out.println("Salah && Benar : " +(S && B));
  System.out.println("Salah && Salah : " +(S && S));

  System.out.println("Operator NOT");
  System.out.println("Tidak Benar: " +!B);
  System.out.println("Tidak Salah: " +!S);
  }
}
```

Jika dijalankan, kode listing di atas akan mengeluarkan output sebagai berikut.

```
C:\java\praktek\23 boolean>java OperatorLogika
Operator OR
Benar || Benar : true
Benar || Salah : true
Salah || Benar : true
Salah || Salah : false
Operator AND
Benar && Benar : true
Benar && Salah : false
Salah && Benar : false
Salah && Salah : false
Operator NOT
Tidak Benar: false
Tidak Salah: true
```

Berikut contoh lain kode program yang menggunakan logika boolean.

```
boolean isBonusGaji = (gaji < 500000) & (prestasi > 50);
 // Jika gaji kurang dari lima ratus ribu dan poin prestasi
 // lebih dari lima puluh, maka karyawan mendapat bonus gaji.
isSenior = !(lamakerja < 10);
 // Jika lama kerja tidak kurang dari 10 tahun, berarti
 // karyawan senior
```

Operator Numerik

Operator numerik digunakan untuk melakukan penghitungan.

Operator	Simbol
Perkalian	*
Pembagian	/
Penjumlahan	+
Pengurangan	-
Sisa(modulus)	%

Berikut contoh kode yang menggunakan operator numerik untuk melakukan penghitungan.

```
int a = 1 + 1;          // a bernilai 2
float b = 3 x 1.2;      // b bernilai 3,6
double c = 25 % 3;
// c bernilai 1. Didapat dari sisa pembagian.
// 25/3 = 24+1 = (8x3) + 1.
```

Penggunaan tanda kurung dapat digunakan untuk mengatur operasi, mana dulu yang akan dilakukan.

```
int f = (20 – 4) * (2 – 6) / 4;
// f = 16*(-4)/4 = -64/4 = -16
```

Operator Shorthand (*Compound, op*=)

Operator ini digunakan dengan menyingkat penulisan. Selain melakukan perhitungan, operator shorthand juga bertugas melakukan assignment.

Operator	Nama operator	Contoh	Ekuivalen
+=	Addition assignment	i += 8	i = i + 8
-=	Subtraction assignment	f -= 8.0	f = f - 8.0
*=	Multiplication assignment	i*= 8	i = i * 8
/=	Division assignment	i /= 8	i = i / 8
%=	Remainder assignment	i %= 8	i = i % 8

Contoh penggunaan operator shorthand:

```
bagi += bagi + 1;       // bagi = bagi + 1;
result /= (x – y)/(x + y);
// Nilainya setara dengan result = result/(x - y/(x + y));
```

Increment dan Decrement

Operator	Nama	Deskripsi
++var	preincrement	Menambah nilai pada variabel dengan satu, lalu menggunakannya dalam operasi.
var++	postincrement	Menggunakan nilai variabel pada operasi yang ada, lalu menambahkannya dengan satu.
–var	predecrement	Mengurangi nilai pada variabel dengan satu, lalu menggunakannya dalam operasi.
var–	postdecrement	Menggunakan nilai variabel pada operasi yang ada, lalu menguranginya dengan satu.

Contoh penggunaan increment dan decrement dalam operasi numerik:

```
int x, y, z; // deklarasi variabel x, y, dan z
x = 12;      // x diberi nilai 12
y = x++;     // nilai x dimasukkan ke dalam x, kemudian nilai x
             // ditambahkan dengan satu. Jadi, y=12 // dan x=13
z = ++x;     // x ditambahkan satu menjadi 14 dan dimasukkan ke
             // dalam variabel z. Jadi z=14 dan x=14
y = x-- + 2  // nilai y sama dengan x ditambah 2, lalu x
             // dikurangi 1. Jadi y=16 dan x=13.
```

Operator Perbandingan

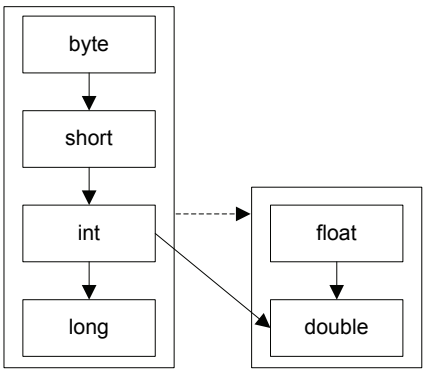
Operator perbandingan digunakan untuk membandingkan antara nilai yang satu dengan nilai lainnya. Operator ini sering digunakan pada pencabangan dan perulangan untuk mengubah alur program.

Operator	Deskripsi	Contoh
>	Lebih besar	z > 1
>=	Lebih besar atau sama dengan	z >= 3
<	Lebih kecil	z < 4
<=	Lebih kecil atau sama dengan	x <= 1
==	Sama dengan	y == 2
!=	Tidak sama dengan	y != 0

Konversi Tipe Data

Konversi tipe data sering dilakukan jika terdapat lebih dari satu macam tipe data dalam satu ekspresi atau statement.

Jika tidak ada perintah untuk mengubah tipe data, konversi akan dilakukan secara implisit dan otomatis oleh program Java. Konversi implisit hanya dapat dilakukan dengan mengubah tipe data ukuran kecil ke tipe data dengan ukuran yang lebih besar. Misalnya dari tipe byte menjadi short.



Konversi tipe data dan presisi

Tanda panah dengan garis putus-putus berarti ada kemungkinan perubahan nilai karena tidak presisi. Misalnya dari tipe long menjadi tipe float. Long mempunyai rentang nilai yang lebih panjang daripada float, karena itu bisa muncul kesalahan penghitungan.

Pengubahan mengikuti tipe data yang paling luas rentang nilainya. Jika salah satu variabel adalah double, maka seluruh data akan diubah menjadi double. Jika tidak ada double dan adanya float, maka seluruh data akan menjadi float. Terus demikian hingga long, int, dan terakhir short.

Berikut contoh dilakukannya konversi secara otomatis.

```
int x = 1
int y = 2;
z = x / y; // z bernilai 0 karena terjadi pembulatan hasil
x = 1/2.0 /* x bernilai 0.5. Karena 2.0 adalah double. Maka 1
diubah menjadi double sehingga hasil x menjadi ½. */
```

Konversi otomatis tidak dapat mengubah data ke tipe yang berukuran lebih kecil.

```
int i = 12;
byte z = i;       // Error, Kode tidak dapat dikompilasi
double xx = 10.75;
int yy = xx;      // Error, Kode tidak dapat dikompilasi
```

Konversi tipe data menjadi berukuran lebih kecil dapat dilakukan melalui casting. Perlu diperhatikan bahwa ada kemungkinan terjadi perubahan nilai karena disesuaikan dengan tipe data yang baru.

Sintaks yang digunakan adalah menggunakan tanda kurung pada tipe data yang diinginkan.

Sintaks Casting

```
(tipe_data) ekspresi;
```

Contoh melakukan casting:

```
float f = (float)(10.1+12.2); // Mengubah nilai menjadi float
int i = (int)f;               // Mengubah nilai dari variabel f
                              // menjadi int. Nilai f tetap
```

Scope

Scope digunakan untuk membatasi sampai sejauh mana nilai suatu konstanta maupun variabel masih berlaku. Di luar scope tersebut, nilai yang telah didefinisikan menjadi tidak berlaku lagi. Scope ditentukan oleh tanda kurung kurawal {}.

Listing program

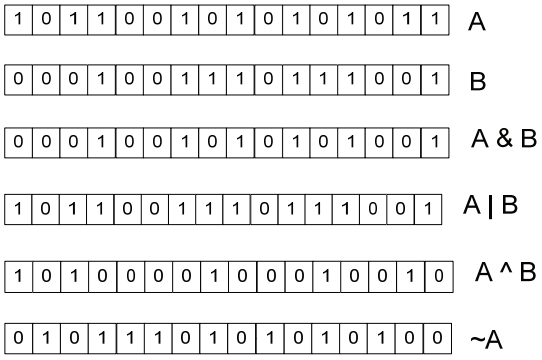
```
public class ContohScopeApp {
 static int x;
 public static void main(String[] args){
  x = 5;
  System.out.println("x dalam main = " + x);
  myMethod();
 }
 public static void myMethod(){
  int y = 10;
  if (y == x + 5){
   int z = 15;
   System.out.println("myMethod: z = " + z);
  }
  System.out.println("myMethod: x = " + x);
  System.out.println("myMethod: y = " + y);
 }
}
```

```
D:\Java\Jadi\CD JAVA\Latihan\Bab2>java ContohScopeApp
x dalam main = 5
myMethod: z = 15
myMethod: x = 5
myMethod: y = 10
```

Operator Bitwise

Operator bitwise digunakan untuk melakukan operasi pada tingkat digital. Terdapat empat jenis operator bitwise.

Simbol	Operator	Keterangan
&	AND	Nilai menjadi 1 jika variabelnya 1 semua, selain itu 0
|	OR	Nilai menjadi 0 jika variabelnya 0 semua, selain itu 1
^	Exclusive OR	Nilai menjadi 0 jika dua variabel sama, selain itu 1
~	Complement	Mengubah 1 menjadi 0 dan 0 menjadi 1

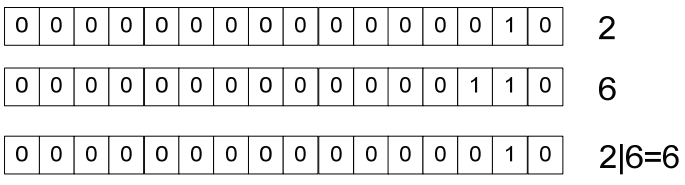



```
int x = 2 | 6     // x bernilai 6.
// Dalam basis dua, 10 | 110 = 110 atau = 6 dalam basis 10
```

Dalam bilangan digital, dua dituliskan sebagai 10 dan enam sebagai 110.

$2 = (2^1 x1) + (2^0 x0) = 2 + 0 = 2$

$6 = (2^2 x1) + (2^1 x1) + (2^0 x0) = 4 + 2 + 0 = 6$



Operasi bitwise untuk 2|6

2.3.2 Komentar, Huruf, dan Whitespace

Komentar

Komentar pada program dipakai untuk memudahkan programmer membaca kode program. Dalam bahasa Java, komentar dapat dituliskan setelah tanda dua buah garis miring //, disebut juga dengan *line comment*. Selain itu komentar dapat diapit di antara tanda /* dan */, disebut juga dengan *paragraph comment*. Berikut beberapa contoh:

```
// Komentar ini hanya berfungsi untuk satu baris saja.
/* Ini adalah komentar yang bebas digunakan untuk beberapa baris
namun harus diakhiri dengan tanda */
```

Javadoc

Dalam Java terdapat komentar yang juga dipakai untuk dokumentasi program. Sering disebut juga dengan Javadoc.

```
/** Ini adalah komentar yang menjadi dokumentasi program Java
melalui Javadoc. */
```

Javadoc akan di-generate oleh kompiler menjadi file dokumen dalam format HTML. Javadoc banyak dipakai untuk memberikan keterangan mengenai kode program.

```
/**
 * Class Karyawan, mewakili karyawan yang ada
 * @see Departemen
 * @author Rachmad Hakim S.
 * @version 1.00, Nov 2008
 */
```

Tag Javadoc

Tag dapat digunakan pada komentar untuk membuat dokumentasi dalam format yang standar. Tag pada komentar ditandai dengan karakter "@" diikuti dengan kata kunci. Misalnya untuk tag nama penulis kode, digunakan tag @author.

Tag	Keterangan
@author	Digunakan untuk menentukan nama pembuat kode program.
@deprecated	Dipakai untuk dokumentasi library pada class maupun method yang sudah digantikan dan sebaiknya tidak dipakai lagi dalam aplikasi baru.

	Deprecated menandakan bahwa method atau class yang dipakai merupakan method atau class yang sudah kuno.
@exception	Digunakan untuk mendokumentasikan kesalahan (*exception*) yang bisa muncul pada situasi tertentu.
{@link}	Men-generate *link* (tautan) ke bagian lain pada dokumentasi dalam dokumen yang sama. Tag ini dapat digunakan untuk memasukkan link ke dalam class atau method lain.
@param	Menampilkan informasi tentang parameter yang digunakan pada method.
@return	Digunakan untuk mendokumentasikan nilai yang dihasilkan dari suatu method.
@see	Digunakan untuk menentukan referensi silang (*cross-references*) ke bagian lain pada dokumentasi, misalnya pada class, method, atau alamat internet (URL) tertentu.
@throws	Hampir mirip dengan @exception.
@version	Menampilkan informasi tentang versi kode program.

Huruf

Java menerapkan pemrograman *case sensitive*. Dengan demikian, nama yang ditulis dengan huruf besar dan huruf kecil mempunyai arti berbeda. Sebagai contoh, if dan If dianggap dua hal yang berbeda.

Whitespace

Whitepsace merupakan karakter kosong yang dapat berupa tab, spasi, atau Enter. *Whitespace* sering digunakan programmer untuk merapikan kode program. *Whitespace* di luar karakter dan string akan diabaikan dan tidak diolah oleh kompiler.

2.4 Alur Program

Alur jalannya program dapat berubah karena salah satu sebab berikut:

- Pemanggilan metode (*invoke a method*). Dengan terjadinya kasus ini, alur program akan berubah dan mengalir ke metode yang dipanggil.
- Pencabangan (*decision making*). Terjadi ketika program dihadapkan pada pilihan dan harus memilih salah satu. Pencabangan dalam bahasa Java diaplikasikan melalui if/else dan switch.

- Perulangan (*repetition*). Perulangan terjadi ketika ada perintah yang harus dikerjakan sampai beberapa kali. Perulangan dalam bahasa Java dapat dilakukan melalui for, while, dan do/while.

2.4.1 Logika Boolean

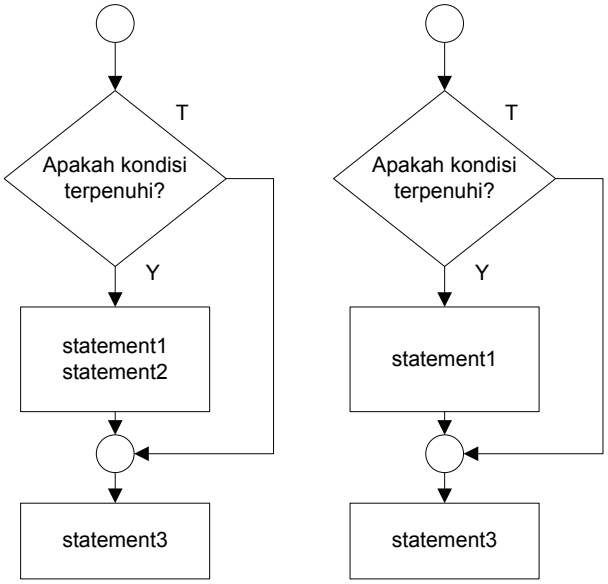
Pencabangan dilakukan menggunakan bantuan logika Boolean. Variabel tipe Boolean hanya mempunyai dua kemungkinan nilai: true (benar) atau false (salah). Dalam alur program, nilai Boolean digunakan setelah mengevaluasi kondisi. Evaluasi kondisi dilakukan menggunakan operator perbandingan dan operator Boolean.

2.4.2 Pencabangan

Komputer sering kali harus memutuskan yang berarti memilih satu dari beberapa aksi, tergantung dari kondisi yang ada. Inilah yang disebut dengan pencabangan.

If

If merupakan perintah pencabangan yang paling sederhana. If membuat statement atau kumpulan statement dijalankan, jika kondisi bernilai benar (true).



Flowchart pencabangan dengan if

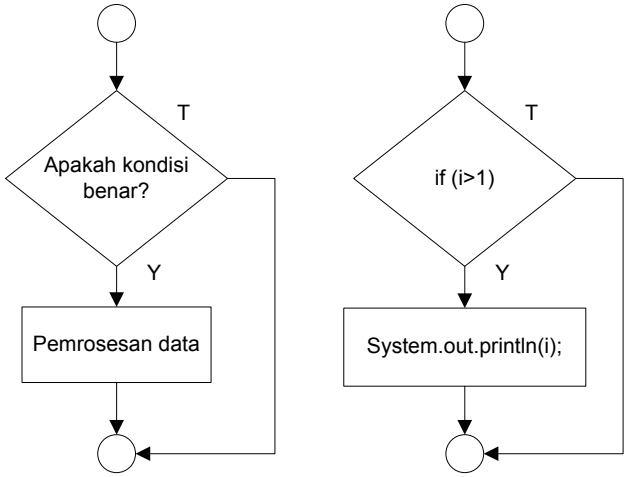
Sintaks If	

`if (condition) {`
`  statement(s);`
`}`
`statement;`
	`if (condition)`
	`  statement;`

Berikut contoh listing program yang digunakan untuk menampilkan nilai pada variabel i. Jika bernilai lebih besar daripada satu, nilai i akan ditampilkan. Jika i bernilai satu atau kurang, maka nilai i tidak akan ditampilkan.

```
int i = 5;
if (i > 1){
 System.out.println(i);
}
```

Program di atas berfungsi untuk menampilkan nilai i, jika nilai i lebih besar daripada satu. Karena nilai i diset sama dengan lima, maka angka 5 akan ditampilkan pada komputer.

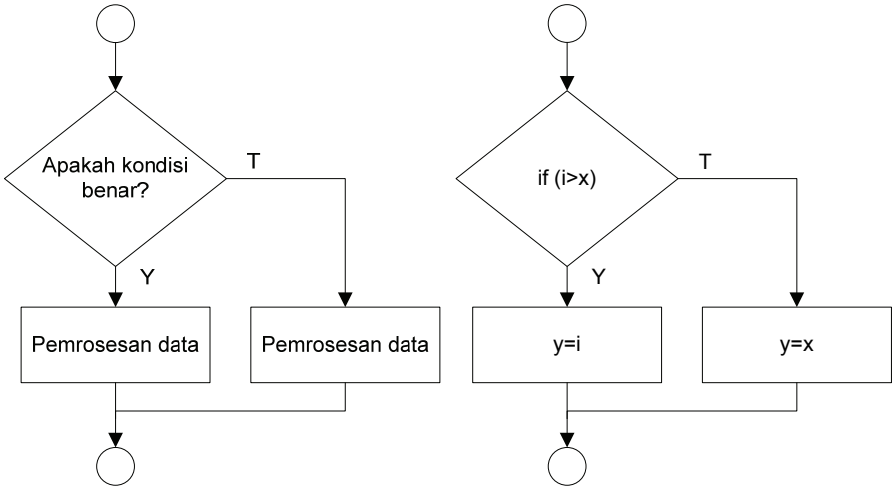


If Else

Pencabangan dengan if else membuat statement pertama dijalankan jika kondisinya benar. Namun, jika kondisi bernilai salah, statement setelah else akan dijalankan. Jika statement yang dipakai banyak, digunakan kurung kurawal. Dengan demikian, kumpulan statement di dalam blok kurung kurawal akan dianggap satu statement.

Sintaks If Else	Sintaks If Else {}

`if (condition)`
`  statement;`
`else`
`  statement;`
	`if (condition){`
	`  statement;`
	`}`
	`else {`
	`  statement;`
	`}`



Berikut contoh kode yang menggunakan pencabangan if-else.

```
int i = 5, x = 6, y;
if (i > x)
    y = i;
else
    y = x;
```

Karena i < x, kode program di atas akan menjalankan perintah y = i. Statement yang akan dijalankan adalah pencabangan else sehingga nilai variabel y = x = 6.

Nested If Else

If Else Nested terjadi jika ada pencabangan di dalam pencabangan. Maksudnya, terdapat if else di dalam if atau else yang lebih tinggi.

Sintaks If Else Nested

```
if (condition)
       if (condition)
              statement;
       else
              statement;
else
       if (condition)
              statement;
       else
              statement;
```

```
if (condition){
       if (condition)
              statement;
       else
              statement;
}
else {
       if (condition)
              statement;
       else
              statement;
}
```

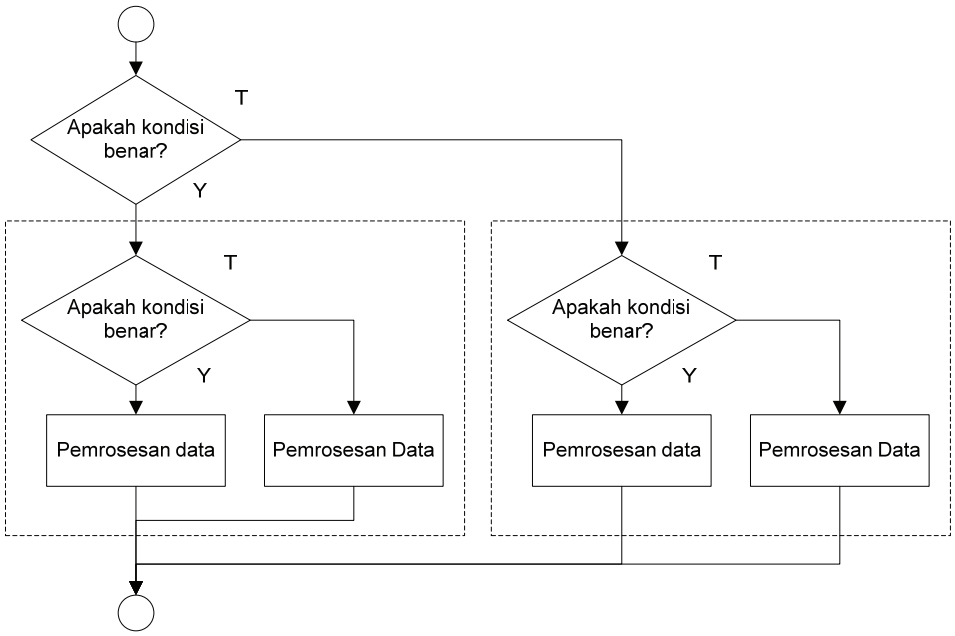
Agar tidak membingungkan, disarankan untuk menggunakan tanda kurung kurawal pada nested if else.

Jika yang Anda gunakan adalah nested if saja, tanpa else, maka gunakan tanda kurawal agar tidak membingungkan. Namun, jika tetap tidak ingin memakai tanda kurung kurawal, gunakan else diakhiri dengan titik koma ";".

Sintaks Variasi If Else Nested

```
if (condition)
       if (condition)
              statement;
       else;

else
       if (condition)
              statement;
       else
              statement;
```

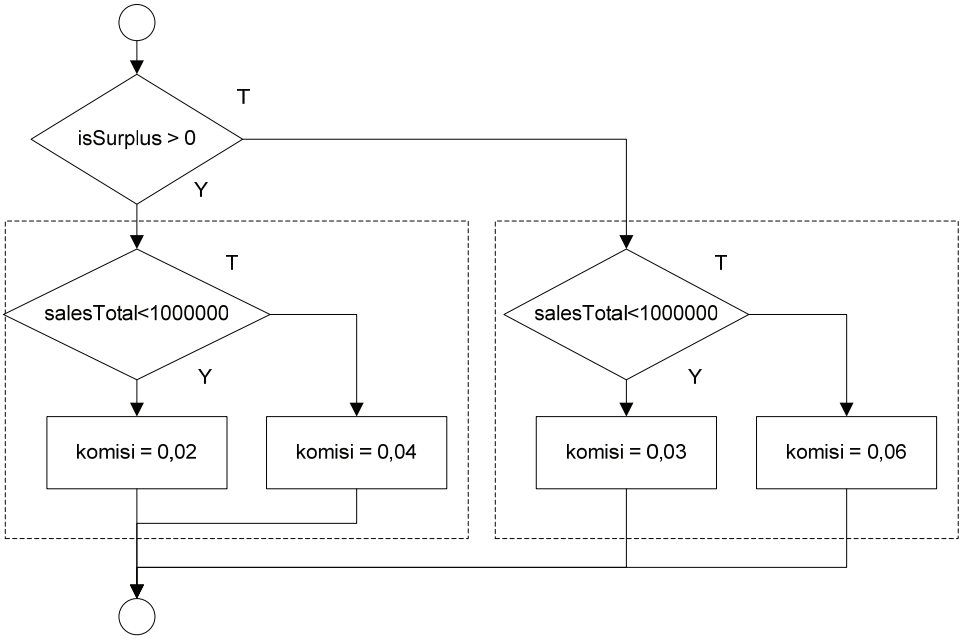
```
if (condition){
       if (condition)
              statement;

}
else {
       if (condition)
              statement;
       else
              statement;
}
```



Flowchart Nested If Else

Berikut contoh listing program dengan nested if else.

```
if (isSurplus == 0)
 if (salesTotal < 10000000)
  komisi = 0.02;
 else
  komisi = 0.04;
else
 if (salesTotal < 10000000)
  komisi = 0.03;
 else
  komisi = 0.06;
```



Else If

Else if sering digunakan pada pencabangan dengan kondisi pilihan lebih dari dua sekaligus.

Sintaks Else If

```
if (condition){
       statement(s);
}
else if (condition)
       statement;
else if (condition){
       statement(s);
}
else
       statement4;
```

Terdapat empat pilihan:

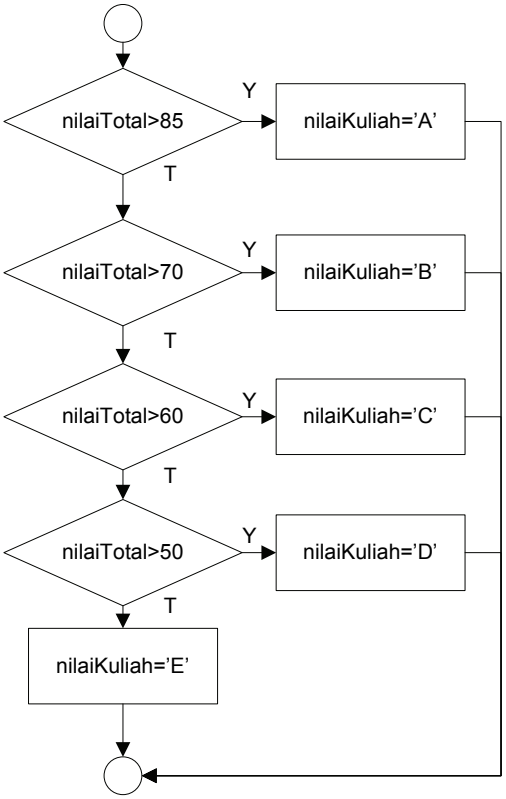
- if
- else if
- else if
- else

Berikut contoh penggunaan logika else if untuk menentukan nilai mata kuliah.

Jumlah Nilai	Nilai Mata Kuliah
85 – 100	A
70 – 85	B
60 - 70	C
50 – 60	D
0 - 50	E

Kode program untuk menerapkan aturan nilai di atas dapat dituliskan sebagai berikut:

```
if (nilaiTotal >= 85)
    nilaikuliah = 'A';
else if (nilaiTotal >= 70)
    nilaikuliah = 'B';
else if (nilaiTotal >= 60)
    nilaikuliah = 'C';
else if (nilaiTotal >= 50)
    nilaikuliah = 'D';
else
    nilaikuliah = 'E';
```



Ternary

Digunakan untuk menuliskan perintah pencabangan dalam sintaks yang singkat. Operator ternary melibatkan tiga buah operator dan cocok untuk kondisi pencabangan yang sederhana.

Sintaks Ternary

```
identifier = condition ? value1 : value2;
```

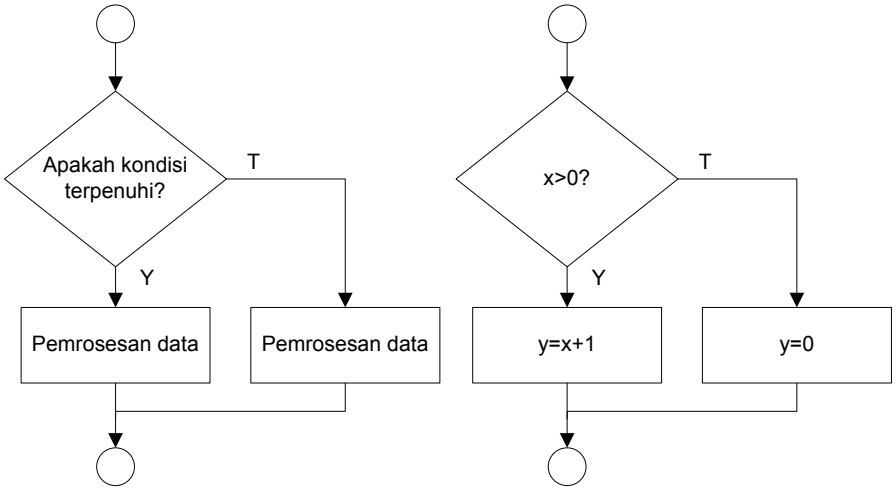
Contoh kode program dengan ternary.

```
y = x > 0 ? x+1 : 0;
```

Kode di atas mempunyai kesamaan dengan perintah kode berikut:

```
if (x > 0)
 y = x+1;
else
 y = 0;
```

Jadi, jika x > 0, maka y = x + 1. Jika x <= 0, maka y = 0.



Perhatikan bahwa kondisi pencabangan di atas langsung menggunakan nilai (*value*), bukan statement. Dengan kata lain, pemrosesan data hanya digunakan untuk menentukan nilai variabel y. Jadi, berbeda dengan perintah if yang menggunakan statement.

Switch

Jika pada pencabangan terdapat pilihan yang banyak, perintah switch lebih dianjurkan daripada menggunakan if. Switch hampir mirip seperti else if. Switch lebih sering dipakai pada beberapa pilihan menggunakan satu variabel sebagai penentu pencabangan.

Sintaks Switch

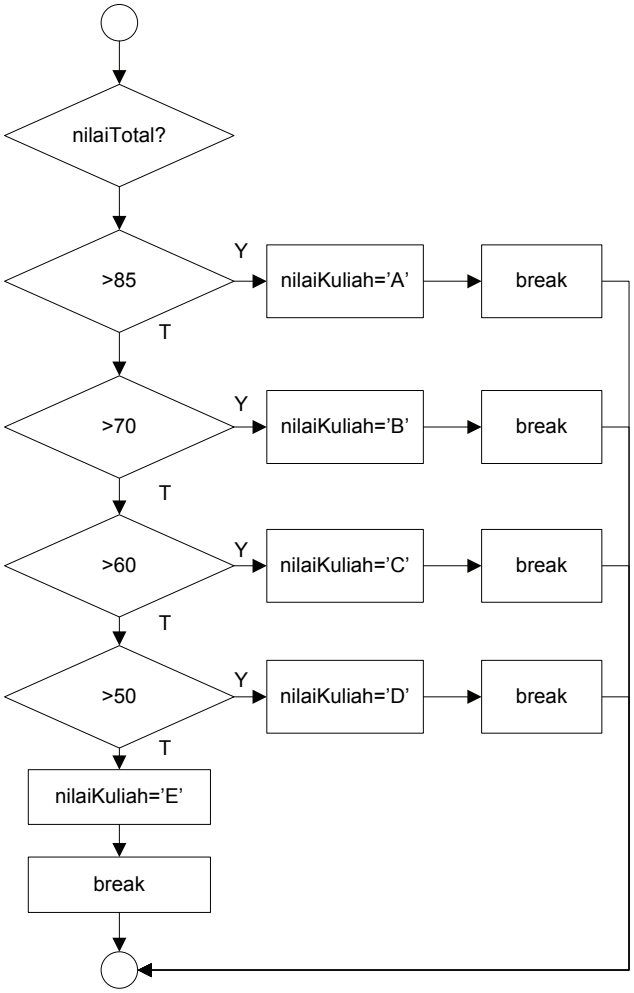
```
switch (identifier) {
case value1:
       statements;
       break;
case value2:
       statements;
       break;
default:
       statements;
       break;
}
```

Terdapat tiga pilihan tergantung pada nilai variabel identifier:

- Value1
- Value2
- Default

Berikut contoh kode yang menggunakan switch untuk menentukan nilai mata kuliah.

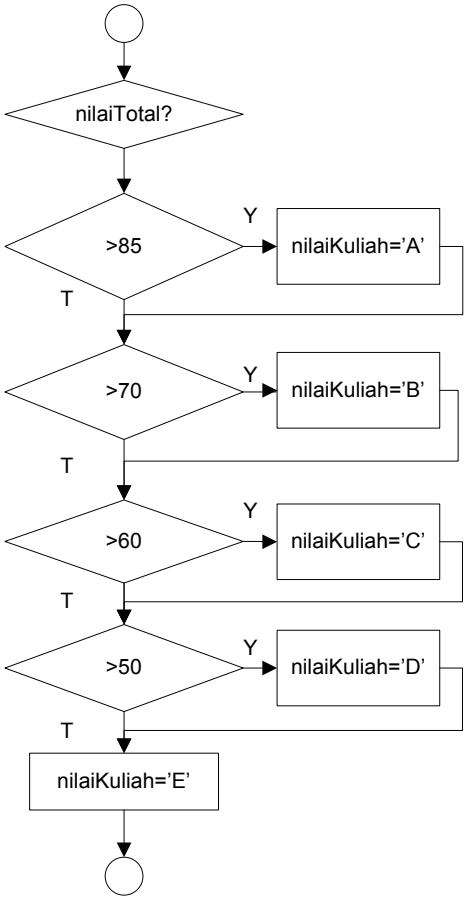
```
switch (nilaiTotal){
 case >= 85:
  nilaiKuliah='A';
  break;
 case >= 70:
  nilaikuliah = 'B';
  break;
 case >= 60:
  nilaikuliah = 'C';
  break;
 case >= 50:
  nilaikuliah = 'D';
  break;
 default:
  nilaikuliah = 'E';
  break;
}
```



Alur program dengan switch dan break

Perintah break dapat dipakai pada pencabangan dan perulangan. Dalam pencabangan, perintah break berguna untuk melompati sisa pencabangan. Jika salah satu kondisi sudah terpenuhi, kondisi-kondisi lain dalam pencabangan tidak akan dievaluasi lagi.

Jika nilai Total = 71, maka program hanya mengetes kondisi dua kali saja, yaitu >85 dan >70. Tanpa adanya statement break, maka semua kondisi akan dievaluasi terus, tidak peduli apakah sudah menemukan kondisi benar atau tidak.



Alur program pada switch tanpa break

Berikut contoh penggunaan pencabangan switch dengan tipe data enumerated.

```
enum JenisKelamin { LAKI, PEREMPUAN } // Deklarasi
JenisKelamin jk = JenisKelamin.LAKI;  // Inisialisasi
switch(jk) {
 case LAKI:
  System.out.println("Jenis kelamin: laki-laki");
  break;
 case PEREMPUAN:
  System.out.println("Jenis kelamin: perempuan");
  break;
}
```

2.4.3 Perulangan

Ada beberapa perintah yang digunakan untuk menerapkan perulangan dalam Java, antara lain:

- While loop
- Do while loop
- For loop

While loop

Perulangan while loop dimulai dengan mengevaluasi kondisi apakah benar atau tidak. Jika benar, statement akan dijalankan dan program kembali lagi melakukan evaluasi kondisi apakah benar atau tidak. Demikian seterusnya hingga kondisi bernilai salah.

Jika salah, statement akan dilewati dan evaluasi kondisi tidak akan dilakukan lagi. Dengan kata lain, perintah pada statements akan terus dieksekusi selama kondisi pada condition bernilai benar (true).

Sintaks While Loop

```
while (condition) {
       statement(s);
}
```

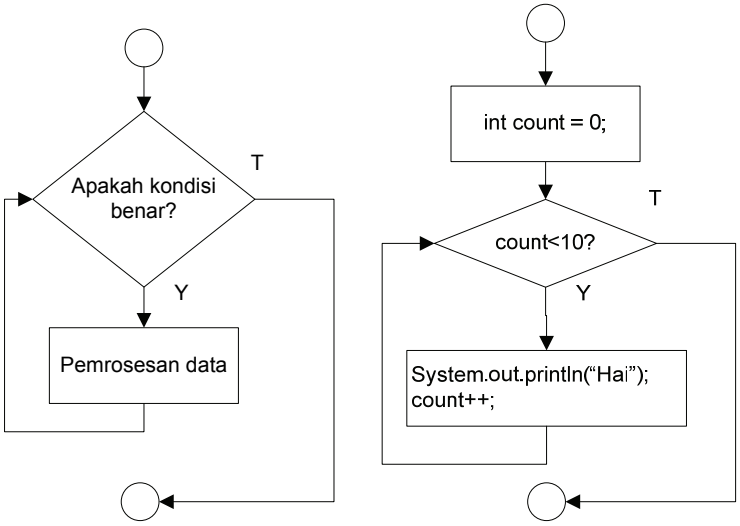
Berikut contoh iterasi menggunakan perintah while.

```
int count = 0;
while (count < 10) {
 System.out.println("Hallo Program Java!");
 count++;
}
```

Keterangan contoh kode program dengan While Loop.

- Variabel count mempunyai nilai awal 0.
- Perintah untuk menampilkan tulisan "**Hallo Program Java**" akan terus dieksekusi selama nilai count kurang dari 10.
- Setelah menampilkan tulisan "**Hallo Program Java**", nilai pada variabel count ditambah dengan satu.

- Dengan demikian, tulisan "**Hallo Program Java**" akan ditampilkan sebanyak 10 kali. Mulai dari saat nilai count=0 hingga nilai count=9.



Kesalahan penghitungan dapat menimbulkan infinite loop di mana perulangan terjadi terus dan tidak bisa berhenti. Kasus ini terjadi jika kita salah menuliskan kode sehingga kondisi pada while loop selalu benar. Untuk menghentikan infinite loop, tutup program aplikasi dan perbaiki kode agar tidak melakukan infinite loop.

Do While Loop

Sintaks Do While Loop

```
do {
        statement(s);
} while (condition);
```

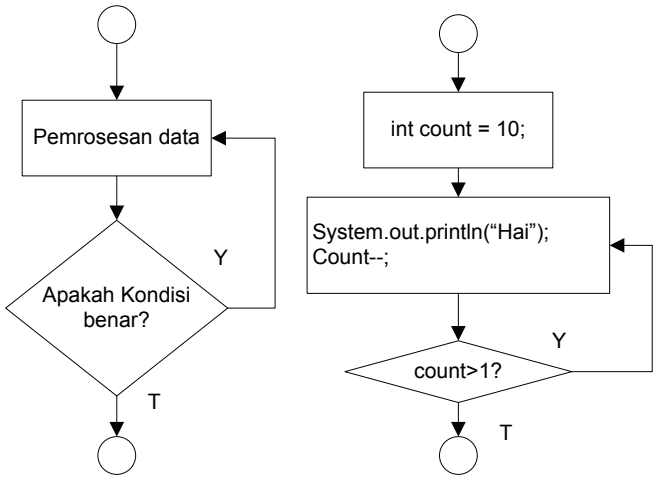
Perulangan dengan do while loop hampir sama dengan while loop. Bedanya adalah perintah ini pasti menjalankan statement minimal satu kali. Jadi, meskipun kondisi tidak memenuhi, perintah dalam statement tetap dijalankan satu kali.

Berikut contoh kode program untuk do while loop.

```
int count = 10;
do {
```

```
    System.out.println("Hai");
    count--;
} while (count > 1);
```

Dari perintah di atas, tulisan "**Hai**" akan muncul sebanyak sembilan kali. Mulai dari saat nilai variabel count=10 hingga menjadi 2.



For Loop

Sintaks For Loop

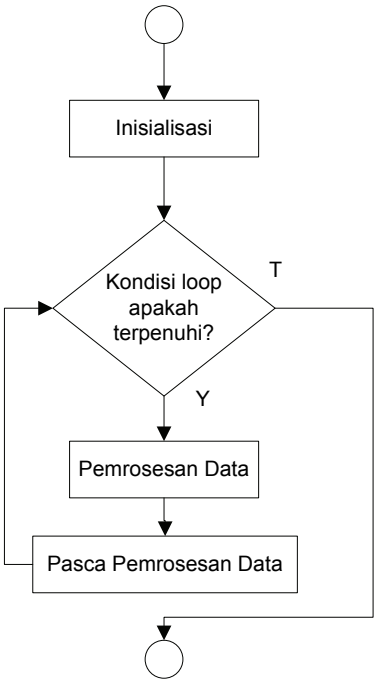
```
for (initial; condition; post-iteration) {
      statement(s);
}
```

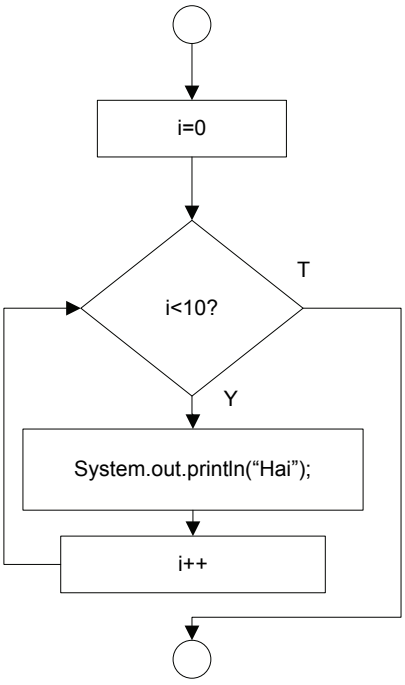
Berikut keterangan tentang sintaks for di atas:

- **Initial**: Nilai awal (inisialisasi) pada variabel yang digunakan untuk looping
- **Condition**: Kondisi yang akan dicek untuk menentukan apakah looping akan dijalankan atau tidak. Jika kondisi bernilai benar, looping dilanjutkan. Namun, jika tidak, looping akan berhenti.
- **Post-iteration**: ekspresi yang dijalankan di akhir setiap statement.
- **Statement(s)**: merupakan perintah dalam for loop body.

```
for (int i=0;i<10;i++){
 System.out.println("Hai");
}
```

Kode program di atas akan menyebabkan tulisan "Hai" muncul sebanyak sepuluh kali. Dimulai saat i = 0 hingga i = 9.





For Loop Berbasis Koleksi

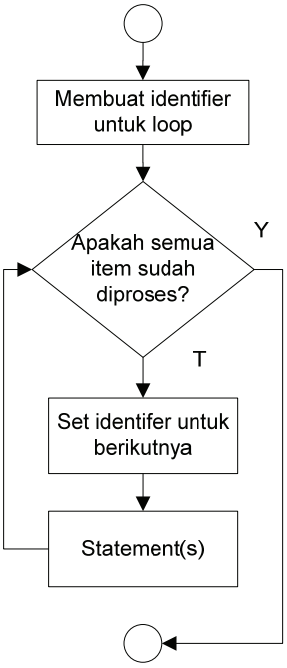
Sintaks For Berbasis Koleksi

```
for (tipe identifier : ekspresi) {
       statement(s);
}
```

- **Type**: tipe data pada variabel yang dipakai untuk perulangan.
- **Identifier**: nama variabel yang digunakan untuk perulangan.
- **Iterable_expression**: berisi seluruh nilai dalam koleksi yang akan digunakan untuk iterasi. Menentukan jumlah iterasi yang akan dilakukan.

Listing program

```
public class CollectionForLoop {
 enum Season { spring, summer, fall, winter }
 public static void main(String[] args) {
  for(Season season : Season.values()) {
   System.out.println(" The season is now " + season);
  }
 }
}
```



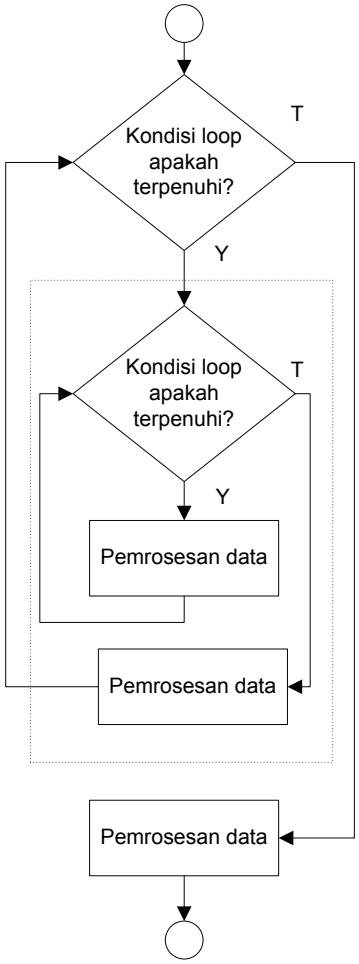

```
C:\java\praktek\2 for>java ForLoopKoleksi
 The season is now spring
 The season is now summer
 The season is now fall
 The season is now winter
```

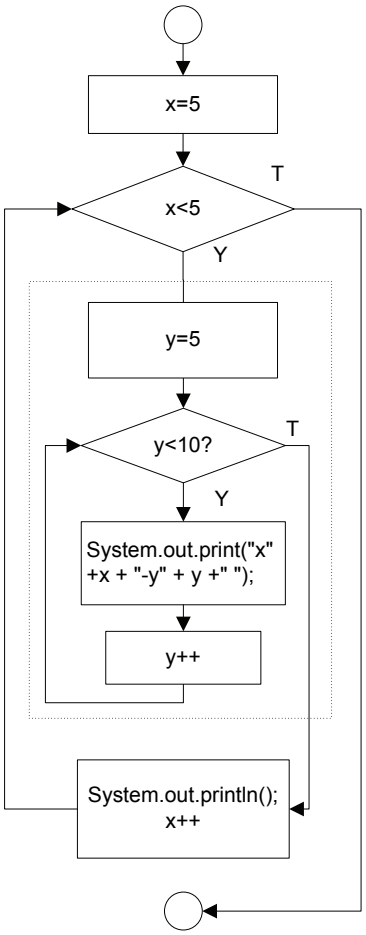
For loop untuk menampilkan isi koleksi

Nested Loop

Nested loop terjadi ketika terdapat loop di dalam loop. Dengan demikian, jumlah perulangan yang terjadi lebih banyak; yaitu perkalian antara loop yang di dalam dengan loop yang di luar.

Nested loop banyak dipakai untuk melakukan perulangan bertingkat, misalnya untuk menampilkan daftar perkalian bilangan, faktorial, dan lain-lain.



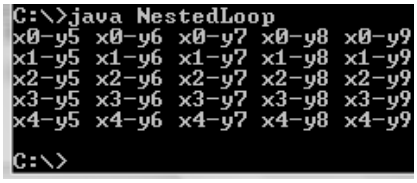


Pada contoh ini, kode program ini, akan dibuat program yang menampilkan urutan looping. Terlihat bahwa loop dengan variabel y terletak di dalam, sedangkan loop dengan variabel x terletak di luar.

Listing program

```
public class NestedLoop{
 public static void main(String[] args){
  for(int x = 0; x < 5; x++){
   for (int y = 5; y < 10; y++)
    System.out.print("x"+x + "-y" + y +" ");
   System.out.println();
  }
 }
}
```

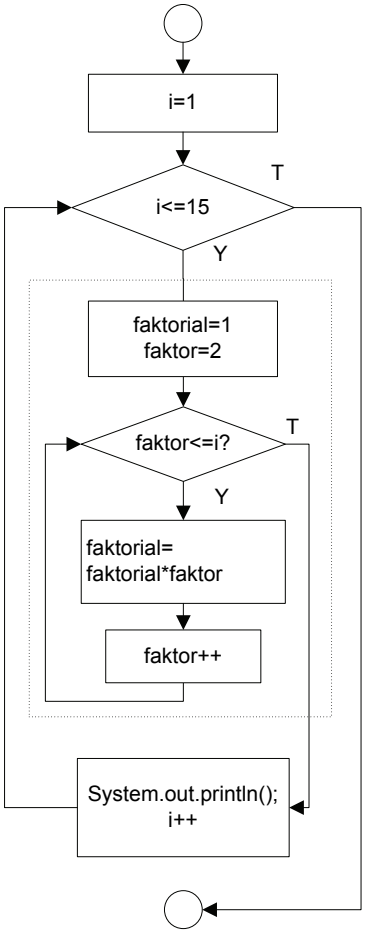
Program di atas menyebabkan perulangan sebanyak x * y = 5 x 5 = 25 kali.



Menghitung Faktorial dengan Nested Loop

Listing program

```
public class Faktorial {
 public static void main(String[] args) {
  long bilangan = 15L;  // Nilai bilangan maksimal
  long faktorial = 1L;  // Variabel untuk menyimpan faktorial
   // Looping untuk mencari faktorial pada bilangan
  for (long i = 1L; i <= bilangan; i++) {
   faktorial = 1L; // inisialisasi faktorial
   // Menghitung nilai faktorial
   for (long faktor = 2; faktor <= i; faktor++) {
    faktorial *= faktor;
   }
   System.out.println("Faktorial dari "+i + " adalah "
    + faktorial);
  }
 }
}
```




```
C:\>javac Faktorial.java

C:\>java Faktorial
Faktorial dari 1 adalah 1
Faktorial dari 2 adalah 2
Faktorial dari 3 adalah 6
Faktorial dari 4 adalah 24
Faktorial dari 5 adalah 120
Faktorial dari 6 adalah 720
Faktorial dari 7 adalah 5040
Faktorial dari 8 adalah 40320
Faktorial dari 9 adalah 362880
Faktorial dari 10 adalah 3628800
Faktorial dari 11 adalah 39916800
Faktorial dari 12 adalah 479001600
Faktorial dari 13 adalah 6227020800
Faktorial dari 14 adalah 87178291200
Faktorial dari 15 adalah 1307674368000
```

Menghitung dan menampilkan faktorial dari angka 1 hingga 15

2.4.4 Branching Statement

Branching statement digunakan untuk menginterupsi proses yang sedang berjalan. Terdapat tiga jenis branching dalam Java:

- break
- continue
- return

Selain berpengaruh terhadap perulangan dan pencabangan, branching statement dapat digunakan bersama label.

Label merupakan baris yang ditandai dengan kata tertentu. Gunanya untuk menandai baris program sehingga Anda dapat kembali ke baris tersebut sewaktu-waktu.

Sintaks Label

```
identifier:
```

Break dan Continue

Break dapat digunakan pada pencabangan maupun perulangan. Jika statement break dieksekusi, pencabangan atau perulangan akan dihentikan. Break berguna untuk menghentikan perulangan. Jika terjadi break, perulangan akan berhenti dan alur program keluar dari loop yang sedang dijalankan.

Sedangkan jika terjadi continue, statement akan dilewati dan alur program kembali ke evaluasi kondisi.

Listing program

```
public class DemoBreak{
 public static void main(String[] args){
  for (int nilai = 0; nilai < 20; nilai += 2){
   if (nilai == 10)
   break;
   System.out.print(nilai + " ");
  }
  System.out.println();
 }
}
```

Program di atas akan menampilkan angka dari 0 2 4 6 8. Pada saat nilai = 10, terjadi break sehingga perulangan dihentikan.

```
C:\java\praktek\2 for>java DemoBreak
0 2 4 6 8
```

Berbeda dengan break, continue menyebabkan program dihentikan namun tidak keluar dari perulangan (loop). Continue menyebabkan program dilanjutkan dengan iterasi berikutnya.

Listing program

```
public class DemoContinue{
 public static void main(String[] args){
  for (int nilai = 0; nilai <20; nilai += 2){
   if (nilai == 10)
    continue;
  System.out.print(nilai + " ");
  }
  System.out.println();
 }
}
```

Pada saat nilai = 10, perulangan dihentikan dan kembali memproses iterasi berikutnya. Jadi, perulangan berlanjut lagi dengan nilai = 12 hingga selesai. Akibatnya, program di atas akan menampilkan nilai 0 2 4 6 8 14 16 18.

```
C:\java\praktek\244 DemoBreak>java DemoBreak
0 2 4 6 8

C:\java\praktek\244 DemoBreak>java DemoContinue
0 2 4 6 8 12 14 16 18
```

Break dan Label

Jika digunakan bersama label, break berguna untuk menginterupsi proses. Program dilanjutkan dengan melewati perulangan atau pencabangan yang ada setelah label.

Listing program

```
public class DemoBreakLabel{
 public static void main(String[] args){
  ganti:
  for(int i = 1; i < 100; i++){
   int j = 1;
   System.out.println("Loop tingkat pertama. i= " + i);
   while(true){
    System.out.println("Loop tingkat kedua. j=" + j);
    if(j++ == 2){
     break ganti;
    }
   }
  }
  System.out.println("Keluar dari loop");
 }
}
```

Break pada kode di atas menyebabkan alur program berhenti dan dicari label dengan nama **ganti**. Diketahui setelah label **ganti**, terdapat perintah for loop. Dengan demikian, for loop ini dilewati dan program langsung melompat ke statement untuk menampilkan kalimat "**Keluar dari loop**".

```
C:\java\praktek\244 DemoBreak>javac DemoBreakLabel.java

C:\java\praktek\244 DemoBreak>java DemoBreakLabel
Loop tingkat pertama. i= 1
Loop tingkat kedua. j=1
Loop tingkat kedua. j=2
Keluar dari loop
```

Continue dan Label

Jika digunakan bersama label, continue berguna untuk menginterupsi proses kembali ke dalam label. Setelah itu, iterasi berikutnya akan dijalankan.

Listing program

```
public class DemoContinueLabel{
 public static void main(String[] args){
  wow:
  for(int i = 0; i < 3; i++){
   System.out.println("Loop tingkat pertama. i=" + i);
   for(int j = 0; j < 10; j++){
    System.out.println("Loop tingkat kedua. j=" + j);
    if(j == (i * 2)){
     continue wow;
    }
    }
   }
  System.out.println("Keluar dari loop");
 }
}
```

Program akan kembali ke loop pertama jika nilai j = i x 2. Akibatnya, akan terjadi continue pada saat:

- i = 0 dan j =0
- i = 1 dan j = 2
- i = 2 dan j =4

Program berhenti pada saat i = 3.

```
C:\java\praktek\244 DemoBreak>javac DemoContinueLabel.java

C:\java\praktek\244 DemoBreak>java DemoContinueLabel
Loop tingkat pertama. i=0
Loop tingkat kedua. j=0
Loop tingkat pertama. i=1
Loop tingkat kedua. j=0
Loop tingkat kedua. j=1
Loop tingkat kedua. j=2
Loop tingkat pertama. i=2
Loop tingkat kedua. j=0
Loop tingkat kedua. j=1
Loop tingkat kedua. j=2
Loop tingkat kedua. j=3
Loop tingkat kedua. j=4
Keluar dari loop
```

Return

Return dibutuhkan untuk keluar dari method. Dengan dijalankannya perintah return, method telah selesai dijalankan. Jika kita tidak menuliskan perintah return, kompiler Java akan menambahkannya pada saat kompilasi.

Return tidak selalu mengembalikan nilai. Pada method dengan modifier void, perintah return dapat ditambahkan sebagai berikut:

```
public void ucapan(){
 System.out.println("Selamat Datang");
 return;
}
```

Sedangkan pada method yang mengembalikan nilai, tipe data pada return harus sama dengan tipe data keluaran pada deklarasi method.

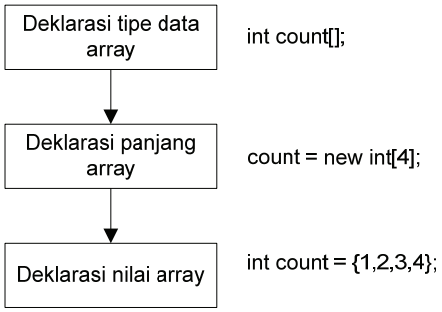
```
public String ucapan(){
 return  "Selamat Datang";
}
```

2.5 Array

Kita sudah mempelajari bahwa setiap variabel dengan tipe data primitif hanya mempunyai satu nilai saja. Jika programmer bekerja dengan banyak nilai, tentu akan kesulitan jika harus menentukan nilai tersebut satu per satu.

Dalam kasus inilah array sangat berguna. Dengan menggunakan array, variabel dengan satu nama dapat digunakan untuk beberapa nilai

sekaligus. Penggunaan array dapat diilustrasikan melalui tahap-tahap berikut.



Tahap menggunakan array pada tipe data primitif

2.5.1 Deklarasi Array

Sintaks Deklarasi Array

```
tipe[] identifier;
tipe identifier[];
```

Sintaks Penentuan Panjang Array

```
identifier = new tipe[jumlah];
```

Sintaks Deklarasi Nilai Array

```
tipe[]identifier = {nilai1, nilai2, nilai3};
```

Berikut contoh statement untuk membuat array.

```
int [] prima;
prima = new int[7];
int[] prima = {2, 3, 5, 7, 11, 13, 17};
 // Array dengan tujuh elemen, terdiri atas bilangan prima
```

Index array dimulai dari 0 hingga panjang – 1. Jika array mempunyai panjang 7, berarti index dimulai dari 0 hingga 6.

0	1	2	3	4	5	6	Index array
2	3	5	7	11	13	17	Nilai array

Pasangan index dan isi dalam array

Deklarasi array dan panjangnya juga dapat langsung dituliskan dalam satu statement.

Sintaks Deklarasi Array dan Panjangnya

```
tipe identifier = new tipe[jumlah];
```

```
int prima2[] = new int[10];
 // Deklarasi array dengan panjang 10
```

Selain dengan cara manual, deklarasi nilai array dapat juga dilakukan secara otomatis menggunakan loop. Cara ini cocok dipakai untuk nilai yang mempunyai pola. Berikut contohnya.

```
double[] data = new double[100];
 // Deklarasi array dengan panjang 100
for(int i = 0 ; i<data.length ; i++) {
 data[i] = i x 10;
}
```

Dengan kode di atas, array pada data akan berisi nilai dari 0 hingga 99 x 10 = 990.

Jika nilai array tidak didefinisikan, kompiler akan memberikan nilai default. Nilai default tergantung dari tipe data array tersebut.

- Numerik, nilai default 0
- Char, nilai default '\u0000'
- Boolean, nilai default false

Setelah ditentukan, jumlah index dalam array tidak dapat diganti lagi. Jumlah index dalam array dapat diketahui menggunakan perintah length, misalnya data.length seperti pada contoh sebelumnya.

Nilai yang disimpan pada array dapat diakses menggunakan index-nya. Berikut contoh kode untuk menampilkan nilai pada array data dengan index lima.

```
System.out.println("Data pada index array kelima="+ data[5]);
```

Kode di atas akan menampilkan tulisan berikut:

Data pada index array kelima=50

2.5.2 Array pada Method Main

Method main yang digunakan untuk mengawali program Java menerima masukan array String. Jika Anda perhatikan, method main sering dituliskan sebagai berikut:

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

}
```

Artinya, method main menerima parameter yang dapat digunakan untuk pengolahan data. Tipe data dalam parameter tersebut adalah String dengan array satu dimensi.

Berikut contoh program yang memanfaatkan parameter pada method main. Parameter ini akan dibaca sebagai tipe data String lalu diubah menjadi Integer. Setelah itu, nilai Integer akan dijumlahkan semua dan hasilnya akan ditampilkan.

Listing program

```
public class DemoArgs{
 public static void main(String[] angka){
  if(angka.length < 2){
   System.out.println("Gunakan minimal dua angka");
   System.exit(1);
  }
  int jumlah = 0;
  for(int i = 0; i < angka.length; i++){
   jumlah += Integer.parseInt(angka[i]);
  }
  System.out.println("Hasil penjumlahan= " + jumlah);
 }
}
```

```
C:\java\praktek\251 Array>java DemoArgs
Gunakan minimal dua angka

C:\java\praktek\251 Array>java DemoArgs 10 7
Hasil penjumlahan= 17

C:\java\praktek\251 Array>java DemoArgs 1000 798 56 4510
Hasil penjumlahan= 6364

C:\java\praktek\251 Array>_
```

Menggunakan parameter command prompt untuk diolah

2.5.3 Array Multidimensi

Array multidimensi terjadi jika terdapat array di dalam array (*arrays of array*). Array multidimensi ditandai dengan beberapa jenis index.

Berikut contoh kode array multidimensi.

Listing program

```
public class Kali {
 public static void main(String[] args) {
  int kali[][] = new int[5][10];
  for (int i = 0; i<kali.length; i++){
   for(int j = 0 ; j<kali[i].length ; j++) {
    kali[i][j] = (i +1) * (j + 1);
    System.out.print(kali[i][j] +"\t" );
   }
  }
 }
}
```

Perhatikan bahwa kode di atas menggunakan perintah length untuk menentukan jumlah array.

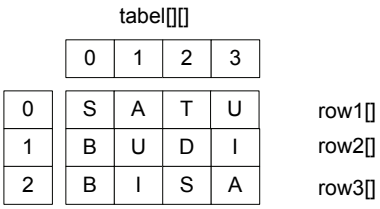
```
C:\>javac Kali.java

C:\>java Kali
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30
4       8       12      16      20      24      28      32      36      40
5       10      15      20      25      30      35      40      45      50

C:\>_
```

index **j**

index i	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10
1	2	4	6	8	10	12	14	16	18	20
2	3	6	9	12	15	18	21	24	27	30
3	4	8	12	16	20	24	28	32	36	40
4	5	10	15	20	25	30	35	40	45	50

Kita dapat melihat array multidimensi sebagai array dengan data array di dalamnya.

```
char[] row1 = {'S', 'A', 'T', 'U'};
char[] row2 = {'B', 'U', 'D', 'I'};
char[] row3 = {'B', 'I', 'S', 'A'};
char[] [] tabel = {row1, row2, row3};
```



2.6 Latihan Membuat Aplikasi Sederhana

Sebelumnya kita sudah membahas pembuatan aplikasi sederhana yang menampilkan teks. Pada contoh ini, kita akan membuat program yang menerima variabel untuk diolah.

Aplikasi Penghitung Volume Tabung

Aplikasi ini menggunakan command prompt tanpa GUI. Program dijalankan dengan masukan berupa jari-jari tabung dan tingginya.

Untuk memudahkan cara berpikir dalam membuat program, kita tidak akan membuatnya langsung. Tetapi, dibuat dulu sebagai komentar yang menggambarkan langkah-langkah pembuatan program tersebut.

```
publis class HitungVolumeTabung {
    // 1. Baca masukan berupa jari-jari dan tinggi tabung
    // 2. Hitung volume tabung
    // 3. Tampilkan besar volume tabung
}
```

Setelah langkah yang dituliskan sudah lengkap, maka langkah selanjutnya baru menuliskan kode programnya. Jangan lupa untuk menuliskan method main yang digunakan sebagai permulaan program Java.

Listing program

```
public class HitungVolumeTabung {
 public static void main(String[] args) {
  System.out.println("Masukkan jari-jari diikuti tinggi tabung");
  double jari;
  double tinggi;
  // 1. Baca masukan berupa jari-jari dan tinggi tabung
  jari = Double.parseDouble(args[0]);
  tinggi = Double.parseDouble(args[1]);
  // 2. Hitung volume tabung
  double volume = 0.5 * 3.14 * jari * jari * tinggi;
  // 3. Tampilkan besar volume tabung
  System.out.println("Besar volume tabung dengan jari-jari "
```

```
    +args[0]+" dan tinggi "+args[1]+" adalah "+ volume);
  }
 }
```

```
C:\java\praktek\2 for>java HitungVolumeTabung 2 10
Masukkan jari-jari diikuti tinggi tabung
Besar volume tabung dengan jari-jari 2 dan tinggi 10 adalah 62.8
```