PEMROGRAMAN BERORIENTASI OBJEK (OOP)

I. Dasar Pemrograman Java

1. Deklarasi Variabel

Bahasa pemrograman pada umumnya mengenal adanya variabel yang digunakan untuk menyimpan nilai atau data. Java dikenal dengan bahasa pemrograman yang bersifat *strongly typed* yang artinya diharuskan mendeklarasikan tipe data dari semua variabel dan apabila lupa atau salah mengikuti aturan pendeklarasian variabel maka akan mendapat *error* pada saat proses kompilasi.

1.1 Tipe Data

Java memiliki dua jenis tipe data yang dikategorikan menjadi dua yaitu tipe data primitif dan tipe data referensi.

- **Tipe Data Primitif**

  Macam – macam tipe data primitif diantaranya :

  - **Integer (Bilangan Bulat)**

    Integer merupakan tipe data numerik yang digunakan untuk mendefinisikan bilangan bulat. Tipe data numeric yang termasuk integer diantaranya :

Tipe	Deskripsi
Byte	-128 s/d +127 menempati 8 bits di memori
Short	-32768 s/d +32767 menempati 16 bits di memori
Int	-2147483648 s/d +2147483647 menempati 32 bits di memori
Long	-9223372036854775808 s/d +9223372036854775807 menempati 64 bits di memori

  - **Floating Point (Bilangan Pecahan)**

    Floating point digunakan untuk menangani bilangan decimal atau perhitungan yang lebih detail dibanding integer.

Tipe	Deskripsi
Float	-3.4×10^{8} s/d $+3.4 \times 10^{8}$
Double	-1.7×10^{308} s/d $+1.7 \times 10^{308}$

  - **Char**

    Char adalah karakter tunggal yang pendefinisiannya di awal dan akhir menggunakan tanda petik tunggal (‘). Tipe char mengikuti aturan Unicode,

sehingga bisa dapat menggunakan kode untuk kemudian diikuti bilangan dari 0 sampai 65535, tetapi yang biasa digunakan adalah bilangan heksadesimal dari 0000 sampai FFFF.

- **Boolean**

  Tipe data Boolean terdiri dari dua nilai saja, yaitu *true* dan *false*. Boolean sangat penting untuk mengevaluasi suatu kondisi.

➢ **Tipe Data Referensi**

Kelebihan pemrograman dengan orientasi objek adalah dapat mendefinisikan tipe data baru yang merupakan objek dari *class* tertentu. Tipe data ini digunakan untuk mereferensikan objek atau class tertentu, seperti String.

- **Variabel**

  Variabel merupakan *container* yang digunakan untuk menyimpan suatu nilai pada sebuah program dengan tipe tertentu. Untuk mendefinisikan variabel, suatu identifier dapat digunakan untuk menamai variabel tersebut.

- **Identifier**

  Identifier adalah kumpulan karakter yang dapat digunakan untuk menamai variabel, method, class, interface, dan package. Dalam pemrograman Java identifier bisa disebut sah apabila diawali dengan :

  - Huruf /abjad
  - Karakter Mata Uang
  - Underscore (_)

  Identifier dapat terdiri dari :

  - Huruf /abjad
  - Karakter Mata Uang
  - Underscore (_)

  Identifier tidak boleh mengandung @, spasi atau diawali dengan angka serta tidak boleh menggunakan *keyword* yang telah digunakan di pemrograman java. Selain karakter, Unicode juga dapat digunakan sebagai identifier.

1.2 Mendklarasikan Variabel

Sintaks dasar :

[tipe data] [nama variabel]

Menuliskan tipe data dari variabel, contoh :

```
int bilangan;
char karakter;
float bildesimal;
boolean status;
```

{setelah dideklarasikan sesuai dengan tipe data, selanjutnya memberi nilai variabel tersebut dengan tanda =}

```
bilangan = 20;
karakter = 'k';
bildesimal = 22.2f;
status = true;
```

{membuat variabel menjadi konstanta sehingga tidak bisa diubah lagi nilainya dengan menambahkan keyword sebelum tipe data}

```
Final int a = 10;
Final float pajak = 15.5;
```

{membuat agar konstanta dapat diakses oleh kelas lain tanpa harus membuat objek terlebih dulu dilakukan penambahan modifier public dan keyword static}

```
Public static final a = 10;
```

2. Operator

Operator adalah simbol khusus yang menyajikan operasi khusus pada satu, dua, atau tiga operand dan kemudian mengembalikan hasilnya. Jenis operator antara lain :

- **Operator Aritmatika**

  Operator ini digunakan pada operasi-operasi aritmatika seperti penjumlahan, pengurangan, pembagian dan lain-lain, contoh :

Operator	Keterangan
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Modulus (sisa bagi)
++	Increment (menaikkan nilai dengan 1)
--	Decrement (menurunkan nilai dengan 1)

- **Operator Relasional**

  Untuk membandingkan 2 nilai (variabel) atau lebih digunakan operator relasional, dimana operator ini akan mengembalikan atau menghasilkan nilai *True* atau *False*, contoh :

Operator	Keterangan
==	Sama dengan
!=	Tidak sama dengan
>	Lebih besar
<	Lebih kecil
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan

- **Operator Kondisional**

  Operator ini menghasilkan nilai yang sama dengan operator relasional, hanya saja penggunaannya lebih pada operasi – operasi Boolean, contoh :

Operator	Keterangan
&&	Operasi AND
||	Operasi OR
^	Operasi XOR
!	Operasi NOT (Negasi)

Contoh Operasional Kondisional :

A	B	A&&B	A||B	A^B	!A
True	True	True	True	False	False
TRUE	False	False	True	True	False
False	True	False	True	True	True
False	False	False	False	False	True

- **Operator Shift dan Bitwise**

  Kedua operator ini digunakan untuk memanipulasi nilai dari bitnya, sehingga diperoleh nilai yang lain, contoh :

Operator	Keterangan
&	Operasi bitwise AND
|	Operasi bitwise OR
^	Operasi bitwise XOR
~	Operasi bitwise NOT
>>	Operasi shift right (geser ke kanan sebanyak n bit)
>>>	Operasi shift right zero fill
<<	Operasi shift left (geser ke kiri sebanyak n bit)

  Contoh Operasi Bitwise :

A	B	A&B	A|B	A^B	~A
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

- **Operasi Assignment**

  Operator assignment dalam java digunakan untuk memberikan suatu nilai ke sebuah variabel. Operator assignment hanya berupa '=', namun selain itu dalam Java beberapa shortcut assignment operator yang penting, contoh :

Operator	Contoh	Ekivalen dengan
+=	b+=a	b=b+a
-=	b-=a	b=b-a
*=	b*=a	b=b*a
/=	b/=a	b=b/a
%=	b%=a	b=b%a
&=	a&=b	a=a&b
|=	a|=b	a=a|b
^=	a^=b	a=a^b
<<=	a<<=b	a=a<<b
>>=	a>>=b	a=a>>b
>>>=	a>>>=b	a=a>>>b

Contoh 1 :

```
public class contoh1a {
public static void main (String[]args) {
int x, y, c;
x = 4&6;
y= 5>>>2;
c= (2+4)*6;
System.out.println(y);
System.out.println(c);
System.out.println(x);
}}
```

Contoh 2 :

```
public class Konversi{
public static void main(String args[]){
float m, cm, inci;
m = 30;
cm = m * 100;
inci = m * 100 / 2.54f;
System.out.println("Ukuran dalam CM = " + cm);
System.out.println("Ukuran dalam Inci = " + inci);
}}
```

3. Kondisional dan Pilihan

- **IF**

Statement if memungkinkan sebuah program untuk dapat memilih beberapa operasi untuk dieksekusi, berdasarkan beberapa pilihan. Terdapat tiga jenis statement If diantaranya :

- If

  Bentuk If adalah yang paling sederhana, mengandung suatu pernyataan tunggal yang dieksekusi jika ekspresi bersyarat adalah benar.

  Sintaks dasar :

```
If (ekspresi_kondisional) {
statement1;
statement2;
… }
```

- **If, else**

  Untuk melakukan beberapa operasi yang berbeda jika salah satu ekspresi kondisional bernilai salah, maka digunakan statement else. Bentuk if-else memungkinkan dua alternatif operasi pemrosesan.

  Sintak dasar :

```
If (ekspresi_kondisional) {
statement1;
statement2;
…
}else {
statement1;
statement2;
… }
```

- **If, else if, else**

  Bentuk if, else if, else memungkinkan untuk tiga atau lebih alternative pemrosesan.

  Sintaks dasar :

```
If (ekspresi_kondisional) {
statement1;
statement2;
…
}else if (ekspresi_kondisional){
statement1;
statement2;
…
} else {
statement1;
statement2;
… }
```

  Contoh :

1. **If**

```
public class IfSatuPilihan{
public static void main(String args[]){
int bil;
bil=0;
if (bil==0)
System.out.println("Bilangan Nol");
}}
```

2. **If, else**

```
import java.util.Scanner;
public class IfDuaPilihan{
public static void main(String args[]){
Scanner masuk = new Scanner(System.in);
int bil;
System.out.print("Masukkan bilangan : ");
bil=masuk.nextInt();
if (bil==0)
System.out.println("Bilangan Nol");
else
System.out.println("Bilangan Bukan Nol");}}
```

3. **If, else if, else**

```
import java.util.Scanner
public class Buah{
public static void main(String args[]){Scanner masuk =
new Scanner(System.in);
int pil;
System.out.print("Masukkan pilihan : ");
pil = masuk.nextInt();
if (pil==1)
System.out.println("Buah Pepaya");
else if(pil==2)
System.out.println("Buah Stroberi");
else if(pil==3)
System.out.println("Buah Apel");
else
System.out.println("Bukan Buah deh");}}
```

- **Switch**

  Switch adalah pernyataan yang digunakan untuk menjalankann salah satu pernyataan dari beberapa kemungkinan statement untuk dieksekusi, berdasarkan nilai dari sebuah

ungkapan dan nilai penyeleksi. Setiap ungkapan diungkapkan dengan sebuah nilai integral konstan, seperti sebuah nilai dengan tipe byte, short, int atau char.

Sintaks dasar :

```
switch (ekspresi){
            case value1:
                        statement1;
                        statement2;
                        break;
            case value2:
                        statement1;
                        statement2;
                        break;
             … [
             default:]
                        statement1;
                        statement2;
}
```

Keterangan

Case	:menandai posisi kode dimana eksekusi dilaksanakan.
Value1,dst	:konstanta integer atau karakter ataupun ekspresi yang mengevaluasi keduanya.
Default	:berfungsi sama seperti else pada statement if.
Break	:dapat menghentikan perulangan walaupun kondisi untuk berhenti belum terpenuhi.
Continue	:dengan statement ini kita bisa melewatkan operasi yang dilakukan dalam iterasi sesuai dengan kondisi tertentu.

Contoh :

```
import java.util.Scanner;
public class CaseJurusan{
    public static void main(String args[]){
         Scanner masuk = new Scanner(System.in);
         int pil;
         System.out.print("Masukkan pilihan :S1 TT ");
         pil = masuk.nextInt();
         switch (pil) {
                  case 1:System.out.println("S1 TE");
                  break;
                  case 2:System.out.println("S1 SK");
```

```
                    break;
                    case 3:System.out.println("D3 TT");
                    break;
                    case 4:System.out.println("D3 IF");
                    break;
                    case 5:System.out.println("S1 TI");
                    break;
                    default:
                    System.out.println("Input salah!");
                    break;
                    }
        }
}
```

II. Array, Method dan Pengulangan

1. Array

- **Definisi Array**

Array adalah sebuah struktur data yang terdiri dari data yang bertipe sama. Ukuran array bersifat tetap, array akan mempunyai ukuran yang sama pada saat sekali dibuat. Array dalam Java adalah obyek, disebut juga sebagai tipe referensi. Sedangkan elemen dalam array Java bisa primitif atau referensi. Posisi dari array biasa disebut sebagai elemen. Elemen array dimulai dari 0 (nol). Penyebutan array diberikan dengan cara menyebutkan nama arraynya dan diikuti dengan indeksnya, dimana indeks dituliskan diantara tanda kurung siku. Contoh array dengan 10 elemen, dimana setiap elemennya bertipe integer, dengan nama A.

Nama	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
Isi larik	12	-56	23	45	-16	-2	85	41	15	20

- Dekalarasi dan Menciptakan Array

  Sebagai sebuah obyek, larik harus diciptakan dengan menggunakan kata cadang new. Deklarasi dan penciptaan variabel array sebagai berikut :

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

  array dideklarasi dan langsung diciptakan.

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

Array dideklarasikan, baru pada pernyataan berikutnya diciptakan.

Contoh :

```
import java.util.Scanner;
public class Array1
{
    public static void main (String args[]){
      Scanner masuk=new Scanner(System.in);
      float nilai[]=new float[5];
      System.out.println("masukkan 5 buah data nilai");
       for(int i=0;i<5;i++){
         System.out.print("Data ke"+(i+1)+": ");
         nilai[i]=masuk.nextFloat();
       }
      System.out.println("data nilai yang dimasukkan");
        for(int i=0;i<5;i++)
          System.out.println(nilai[i]);
    }
}
```

Hasil Output :

```
masukkan 5 buah data nilai
Data ke1: 2
Data ke2: 4
Data ke3: 5
Data ke4: 7
Data ke5: 9
data nilai yang dimasukkan
2.0
4.0
5.0
7.0
9.0
```

- **Array Multi Dimensi**

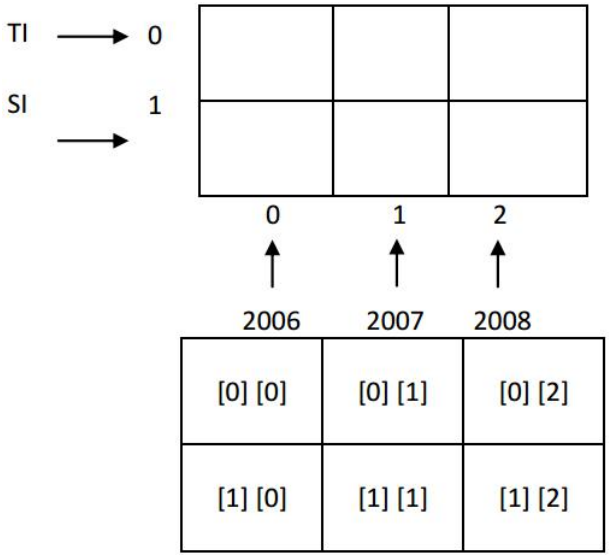
  Kita juga bisa membuat variabel array yang tipe elemennya adalah array. Dengan cara demikian, kita membuat array dua dimensi. Dengan array dua dimensi, maka kita mempunyai elemen yang berindeks tidak hanya satu, tetapi dua. Kita bisa membayangkan array dua dimensi tersebut seperti sebuah tabel yang berisi baris dan kolom. Penyebutan tabel selalu diikuti dengan penyebutan baris berapa dan kolom berapa.

Contoh :

Diberikan data kelulusan mahasiswa sebuah perguruan tinggi sebagai berikut:

Jurusan	2006	2007	2008
Teknik Informatika	110	125	135
Sistem Informasi	56	75	80

`int` data_lulus [2] [3]



Contoh:

```
public class ArrayDimensiDua
{
public static void main(String [] args)
{
  int [][] piksel = new int[2][3];
  // mengisi elemen tertentu
  piksel[0][0] = 70;
  piksel[0][1] = 18;
  piksel[0][2] = 45;
  piksel[1][0] = 75;
  piksel[1][1] = 66;
  piksel[1][2] = 89;
  //menampilkan elemen array
  int i,j;
  for(i=0;i<2;i++){
  for (j=0; j<3;j++)
  System.out.print(piksel[i][j] +" ");
  System.out.println("");
```

```
}}
}
```

Hasil Output:

```
70 18 45
75 6 89
```

- **String**

String adalah kelas yang menangani deretan karakter. Kelas ini mendukung sejumlah metode yang sangat berguna untuk memanipulasi string, misalnya untuk mengkonversikan setiap huruf kecil menjadi huruf besar atau sebaliknya, memperoleh jumlah karakter dan sebagainya. String sebenarnya merupakan *class* yang terdapat pada library Java. Kelas string memiliki banyak konstruktor seperti berikut:

Konstruktor	Keterangan
String()	Menciptakan obyek string yg berisi string kosong (jumlah karakter = 0)
String(char[]v)	Menciptakan obyek string yg berisi string yg berasal dari array yg dirujuk oleh v
String(String v)	Menciptakan obyek string yg isinya sama dengan obyek string argumennya

Metode dalam kelas string memperlihatkan sejumlah metode penting dalam kelas string, seperti :

- copyValueOf(char data[])
- copyValueOf(char data[], int offset, int jum)
- valueOf(boolean b)
- valueOf(double c)
- cocat(String s)
- length()
- trim()
- dan lain-lain

Kelas StringBuffer adalah kelas yg menyimpan string yang konstan, begitu obyek string telah diciptakan maka string tidak dapat diubah. Konstruktor kelas ini antara lain :

- StringBuffer() digunakan untuk menciptakan StringBuffer yang kosong

- StringBuffer(int n) digunakan untuk menciptakan StringBuffer dengan n karakter
- StringBuffer(String s) digunakan untuk menciptakan StringBuffer dengan string berupa s.

Contoh :

```
public class ContohString
{
public static void main(String args[])
{
    byte data[] = new byte[6];
    data[0] = 64;
    data[1] = 65;
    data[2] = 66;
    data[3] = 67;
    data[4] = 68;
    data[5] = 69;
    String s1 = "Selamat Pagi";
    String s2 = new String("Good Morning");
    String s3 = new String(data);
    String s4 = new String(data, 2, 3);
    System.out.println("s1 = " + s1);
    System.out.println("s2 = " + s2);
    System.out.println("s3 = " + s3);
    System.out.println("s4 = " + s4);
}
}
```

Hasil Output:

```
s1 = Selamat Pagi
s2 = Good Morning
s3 = @ABCDE
s4 = BCD
```

Pada program di atas, pernyataan seperti :

String s1 = "Selamat Pagi";

Sebenarnya identik dengan :

String s1 = new String("Selamat Pagi");

Pernyataan

String s3 = new String(data);

akan membuat string yang tersusun atas karakter-karakter yang nilainya sama seperti elemen-elemen pada array data, maka s3 berisi string @ABCDE adalah karakter @ = 64, A=65 dan seterusnya.

Pernyataan :

```
String s4 = new String(data, 2, 3);
```

Angka 3 menyatakan jumlah karakter yg menyusun string dan angka 2 menyatakan karakter pertama pada string, hasil diambil pd indeks ke-2 array.

2. Method

- **Method Tanpa Variabel**

  Method (atau dalam beberapa bahasa pemrograman sering disebut fungsi atau prosedur) adalah sub program yang membiarkan seorang programer untuk membagi program dengan membagi masalah ke dalam beberapa sub masalah yang bisa diselesaikan secara modular. Dengan cara demikian, maka pembuatan program bisa lebih dimanajemen. Kelas (*class*) adalah program java yang akan dieksekusi. Method ada di dalam kelas. Java mempunyai kumpulan kelas yang sudah dimiliki yang tersimpan di dalam paket-paket. Kumpulan kelas tersebut ada di dalam *Java Application Interface* (Java API) atau *Java class libraries* dan beberapa *libraries* lainnya.

  Format Method Secara Umum

```
tipe_return-value
nama_method(parameter1,parameter2,…,parameterN)
{
   deklarasi dan pernyataan;
}
```

Elemen yang diperlukan dari deklarasi method adalah tipe kembalian method, nama, kurung buka dan tutup () dan isi method yang diawali dan diakhiri dengan kurung kurawal buka dan tutup {}. Secara umum, deklarasi method mempunyai 6 komponen, yaitu:

1. Modifier - seperti public, private, dan yang lainnya.
2. Tipe kembalian (*return type*)—tipe data dari nilai yang dikembalikan oleh method, atau void jika method tidak mempunyai nilai kembalian.
3. Nama method—aturan untuk penamaan field diterapkan untuk nama method.
4. Daftar parameter – pemisah antar parameter input adalah koma, diawali oleh tipe datanya, yang diletakkan diantara kurung (…daftar parameter….). Jika tidak ada parameter, harus menggunakan kurung buka tutup saja ().
5. Daftar exception
6. Isi method, diletakkan di antara kurung kurawal buka dan tutup {} — kode-kode method, termasuk deklarasi variabel lokal ada di sini.

Contoh:

```
public class Fungsi2
{
    public static void kalimat(){
       System.out.println("Di dalam method kalimat");
    }

    public static void main(String args[]){
       kalimat();
       System.out.println("Di dalam main");
       kalimat();
    }
}
```

Hasil Output :

```
Di dalam method kalimat
Di dalam main
Di dalam method kalimat
```

- **Method dengan Variabel**

  Method (atau dalam beberapa bahasa pemrograman sering disebut fungsi atau prosedur) adalah sub program yang membiarkan seorang programer untuk membagi program dengan membagi masalah ke dalam beberapa sub masalah yang bisa diselesaikan secara modular. Dengan cara demikian, maka pembuatan program bisa lebih dimanajemen.

Contoh:

```
public class FungsiParameter{
   public static int jumlah(int a){
      return a;
   }
   public static void main(String args[]){
     System.out.println("Hasil pemanggilan method jumlah
     ");
     System.out.println(jumlah(5));
   }
}
```

Hasil Output:

```
Hasil pemanggilan method jumlah
```

```
5
Press any key to continue . . .
```

Parameter pada baris kedua disebut sebagai parameter formal, dan pada baris ke 8 disebut parameter aktual. Ada 2 buah parameter yaitu:

- parameter formal adalah parameter yang tertulis dalam definisi method
- Parameter aktual parameter yang berada pada inputan langsung pada saat penggunaan method tersebut.

Parameter bisa lebih dari satu dengan dipisahkan tanda koma,. Yang perlu diperhatikan pada saat pemanggilan method adalah jumlah, urutan dan tipe parameter aktual harus sesuai dengan jumlah urutan dan tipe parameter formal.

Pemberian Variabel Dalam Method

Ada dua tipe data variable passing pada method, yaitu pass-by-value dan pass-by-reference.

- **Pass-by-value**
  Ketika pass-by-value terjadi, method membuat sebuah salinan dari nilai variable yang dikirimkan ke mthod. Walaupun demikian method tidak dapat secara langsung memodifikasi nilai variable pengirimnya meskipun parameter salinannya sudah dimodifikasi nilainya di dalam method.
- **Pass-by-reference**
  Ketika sebuah pass-by-reference terjadi, alamat memori dari nilai pada sebuah variable dilewatkan pada saat pemanggilan method. Ini tidak seperti pada pass-by-value, method dapat memodifikasi variable asli dengan menggunakan alamat memori tersebut, meskipun berbeda nama variable yang digunakan dalam method dengan variable aslinya, kedua variable ini menunjukkan lokasi dari data yang sama.

3. Pengulangan

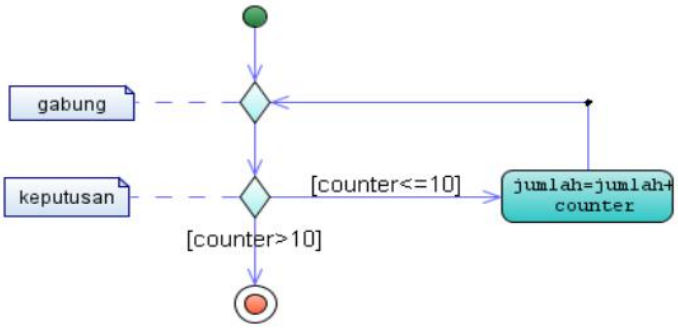
- **Pengulangan dengan while**
  Pernyataan ini berguna untuk memproses suatu pernyataan atau beberapa pernyataan beberapa kali. Selama ungkapan bernilai benar, pernyataan akan selalu dikerjakan.
  Bentuknya :

```
while (ungkapan)
Pernyataan;
```

Keterangan :

- Bagian pernyataan akan diekseskusi selama ungkapan dalam *while* bernilai benar.
- Pengujian terhadap ungkapan pada *while* dilakukan sebelum bagian pernyataan.
- Kemungkinan pernyataan pada *while* tidak dijalankan sama sekali, jika ketemu kondisi yang pertama kali bernilai salah.

Activity diagramnya adalah seperti berikut :



Contoh:

```
import java.util.Scanner;
public class UlangWhile1
{
    public static void main(String args[]){
       Scanner masuk = new Scanner(System.in);
       int bil;
       bil=1;
       while (bil<=5) {
          System.out.println(bil);
          bil++;
       }
    }
}
```

Hasil Output:

1
2
3
4
5

- **Pengulangan dengan do-while**

Seperti halnya perulangan dengan while, perulangan dengan do ... while ini juga digunakan untuk mengerjakan sebuah atau sekelompok pernyataan berulang-ulang. Bedanya dengan while adalah pernyataan do ... while akan mengecek kondisi di belakang, sementara while cek kondisi ada di depan.

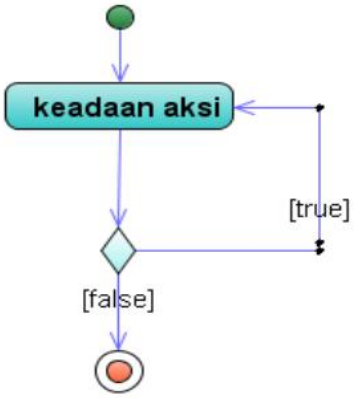
Bentuknya :

```
do
{
      pernyataan1;
      pernyataan2;
      ...........
      pernyataan_N;
}
while (ungkapan)
```

Keterangan :

- Bagian pernyataan1 hingga pernyataanN dijalankan secara berulang sampai ungkapan bernilai salah.
- Pengujian ungkapan dilakukan setelah bagian pernyataan, maka pada pernyataan *do ... while* minimal akan dijalankan sekali, karena begitu masuk ke blok perulangan, tidak ada cek kondisi tetapi langsung mengerjakan pernyataan.

Activity diagramnya :



Contoh penggunaan didalam program :

```
public class UlangDo2
{
    public static void main(String args[]){
      int c;
      double f;
      System.out.println("----------------------");
      System.out.println("CELCIUS FAHREINHEIT");
```

```
       System.out.println("----------------------");
       c=1;
       do
       {
         f=1.8 * c + 32;
         System.out.println("Celcius:"+c+"
         Fahreinhet:+f);
         c++;
        } while (c<=10);
        System.out.println("----------------------");
      }
}
```

Hasil Output:

```
----------------------
CELCIUS FAHREINHEIT
----------------------
Celcius : 1 Fahreinhet : 33.8
Celcius : 2 Fahreinhet : 35.6
Celcius : 3 Fahreinhet : 37.4
Celcius : 4 Fahreinhet : 39.2
Celcius : 5 Fahreinhet : 41.0
Celcius : 6 Fahreinhet : 42.8
Celcius : 7 Fahreinhet : 44.6
Celcius : 8 Fahreinhet : 46.4
Celcius : 9 Fahreinhet : 48.2
Celcius : 10 Fahreinhet : 50.0
----------------------------
Press any key to continue . . .
```

- **Pengulangan dengan for**

  Sama seperti pernyataan perulangan while dan do...while, pernyataan for juga digunakan untuk mengerjakan pernyataan atau sekelompok pernyataan secara berulang. Bedanya adalah dengan pernyataan for perulangan akan dikerjakan dalam hitungan yang sudah pasti, sementara while dan do...while tidak.

  Bentuknya :

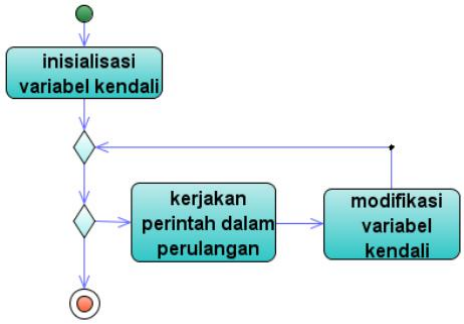
```
for (ungkapan1;ungkapan2;ungkapan3)
Pernyataan;
```

  Keterangan :

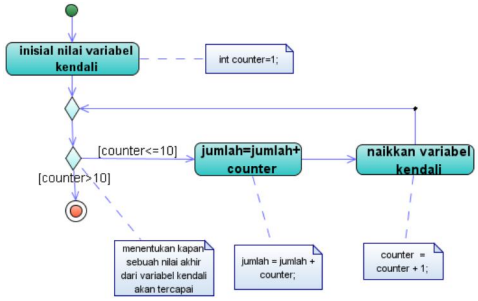
  - ungkapan1 merupakan pernyataan inisialisasi
  - ungkapan2 sebagai kondisi yang menentukan pengulangan terhadap pernyataan atau tidak

- ungkapan3 digunakan sebagai pengatur variabel yang digunakan didalam ungkapan1

Activity diagram pengulangan dengan for :



Contoh activity diagram pengulangan dengan for :



Contoh:

```
public class UlangFor
{
public static void main (String args[])
{
   int bil;
   for (bil=1;bil<=5;bil++)
   System.out.println(bil);
}
}
```

Contoh Program mencakup array, method, dan pengulangan :

```
package matrik;
import java.util.Scanner;
public class Main {
   public void kalimat(){
       System.out.print("Isi matriks adalah : ");
    }
    public int hitungluas(int p,int l){
      int luas;
      luas=p*l;
      return luas;
```

```
    }
    public int hitungvolume(int p,int l,int t){
      int volume;
      volume=p*l*t;
      return volume;
    }
    public static void main(String[] args) {
      int p,l,t;
      int data[];
         Scanner masuk=new Scanner(System.in);
         System.out.print("masukkan panjang : ");
         p=masuk.nextInt();
         System.out.print("masukkan lebar : ");
         l=masuk.nextInt();
         System.out.print("masukkan tinggi : ");
         t=masuk.nextInt();
         data=new int[3];
         Main saya=new Main();
           data[0]=saya.hitungluas(p,l);
           data[1]=saya.hitungvolume(p,l,t);
           data[2]=10;
           int bil=0;
         while (bil<=2) {
             saya.kalimat();
             System.out.println(data[bil]);
             bil=bil+1;
         }
     }
}
```

Hasil Output program:

```
masukkan panjang : 2
masukkan lebar : 3
masukkan tinggi : 4
Isi matriks adalah : 6
Isi matriks adalah : 24
Isi matriks adalah : 10
```