MODUL 2
Array, Method, dan Pengulangan

A. TUJUAN

- Mahasiswa memahami tentang pengertian array serta dapat membuat program dengan menggunakan array.
- Mahasiswa memahami tentang pengertian sub program dan dapat membuat sub program sederhana.
- Mahasiswa dapat menyelesaikan permasalahan dengan menggunakan perulangan, baik rekursif, for, do..while dan while.

B. PERANGKAT

- NetBeans IDE 7.2

C. DASAR TEORI

1. Array / Larik

1.1 Definisi Array / Larik

Larik adalah sebuah struktur data yang terdiri dari data yang bertipe sama. Ukuran larik bersifat tetap, larik akan mempunyai ukuran yang sama pada saat sekali dibuat. Larik dalam Java adalah obyek, disebut juga sebagai tipe referensi. Sedangkan elemen dalam larik Java bisa primitif atau referensi. Posisi dari larik biasa disebut sebagai elemen. Elemen larik dimulai dari 0 (nol). Penyebutan larik diberikan dengan cara menyebutkan nama lariknya dan diikuti dengan indeksnya, dimana indeks dituliskan diantara tanda kurung siku. Gambar 1. memperlihatkan gambaran larik dengan 10 elemen, dimana setiap elemennya bertipe integer, dengan nama A.

Tabel 2.1 Contoh Array

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

1.2 Deklarasi dan Menciptakan Array / Larik

Sebagai sebuah obyek, larik harus diciptakan dengan menggunakan kata cadang new. Deklarasi dan penciptaan variabel larik gambar 1 adalah sebagai berikut.

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

larik dideklarasikan dan langsung diciptakan . Atau

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

larik dideklarasikan, baru pada pernyataan berikutnya larik diciptakan.

Contoh:

```
import java.util.Scanner;
public class Larik1
{
    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
```

1.3 Array Multi Dimensi (N-Dimensi)

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

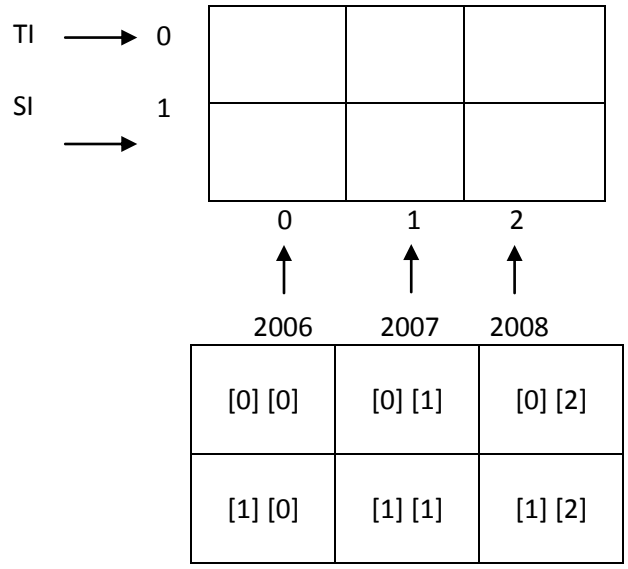
Contoh :

Diberikan data kelulusan mahasiswa sebuah perguruan tinggi sebagai berikut.

Tabel 2.2 Contoh Array Multi Dimensi

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 66 89
```

1.4 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 tabel berikut:

Tabel 2.3 Konstruktor String

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

2.1 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 lain yang akan kita pelajari kemudian.
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 tetapi kesepakatannya adalah sedikit berbeda.
4. Daftar parameter – pemisah antar parameter input adalah koma, diawali oleh tipe datanya, yang diletakkan diantara tkita kurung (…daftar parameter….). Jika tidak ada parameter, harus menggunakan kurung buka tutup saja ( ).
5. Daftar exception—tidak akan masuk dalam pembahasan di sini
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
```

2.2 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

3.1 Pengulangan dengan while

Pernyataan ini berguna untuk memproses suatu pernyataan atau beberapa pernyataan beberapa kali. Selama ungkapan bernilai benar, pernyataan akan selalu dikerjakan.

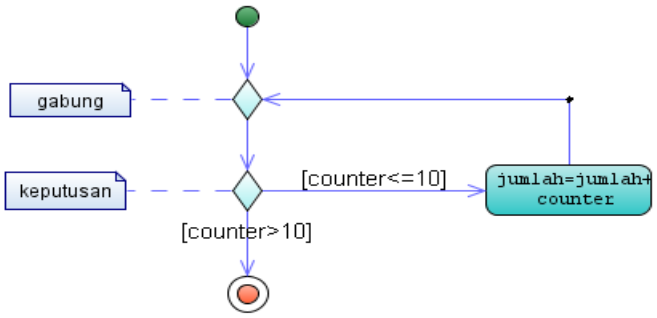
Bentuknya :

```
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 gambar berikut :



Gambar 2.1 Pengulangan dengan while

Catatan :
Pernyataan perulangan dengan while akan selalu dikerjakan jika ungkapan selalu benar. Oleh karena itu, kita harus membuat kondisi suatu saat ungkapan bernilai salah agar perulangan berakhir.

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
```

3.2 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.

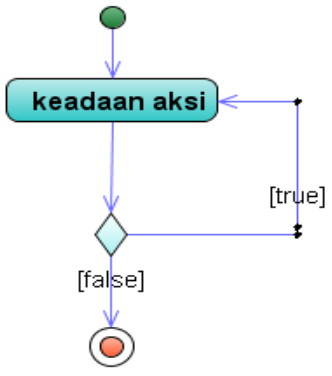
Bentuknya :

```
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 :



Gambar 2.2 Pengulangan dengan do-while

Contoh:

Buatlah program mencetak konversi suhu dari celcius ke fahreinhet mulai dari 1 sampai 10 dengan membuat tabel.

```
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 . . .
```

3.3 Pengulangan dengan for

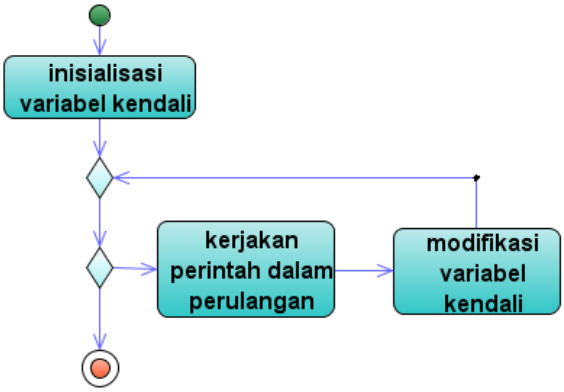
Sama seperti pernyataan perulangan while dan do...while, pernyataan for juga digunakan untuk mengerjakan pernyataan atau sekelompok pernyataan secara berulang. Bedanya adalah dengan pernyataan for perulangan akan dikerjakan dalam hitungan yang sudah pasti, sementara while dan do...while tidak.

Bentuknya :

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

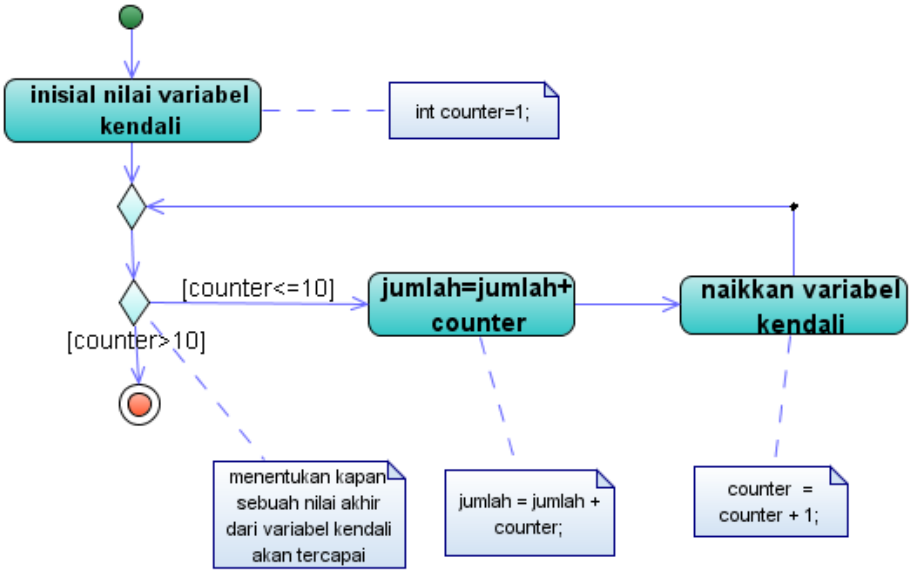
Keterangan :

- ungkapan1 merupakan pernyataan inisialisasi
- ungkapan2 sebagai kondisi yang menentukan pengulangan terhadap pernyataan atau tidak
- ungkapan3 digunakan sebagai pengatur variabel yang digunakan didalam ungkapan1



Gambar 2.3 Activity Diagram untuk perulangan dengan FOR

Contoh activity diagram untuk perulangan dengan for:



Gambar 2.4 Contoh Activity Diagram untuk perulangan dengan FOR

Contoh:

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

Maka akan ditampilkan hasil Output seperti berikut:

```
1
2
3
4
5
```

4. Rekursi

Relasi perulangan adalah persamaan-persamaan untuk menentukan satu atau lebih urutan-urutan secara rekursif. Beberapa relasi perulangan tertentu dapat "diselesaikan" untuk mendapatkan definisi bukan-rekursif.

Penggunaan rekursi dalam suatu algoritma memiliki kelebihan dan kekurangan. Kelebihan utamanya adalah biasanya kesederhanaan. Kekurangan utamanya adalah terkadang algoritma tersebut membutuhkan memori yang sangat banyak jika kedalaman rekursi sangat besar. Rekursif dapat mendefinikan barisan, fungsi dan himpunan.

Langkah-langkah untuk mendefinisikan secara rekursif:

1. Langkah basis : Tentukan anggota awalnya.
2. Langkah rekursif : Bentuk aturan untuk membuat anggota baru dari anggota yang telah ada.

Dalam bahasa pemrograman, rekursif adalah proses dimana fungsi memangil dirinya sendiri. Berikut ini contoh pemrograman factorial dengan rekursif.

```
public class Factorial{
   public static void main (String args[]){
         Factorial f = new Factorial();
         System.out.print("8! =  ");
         System.out.print(f.HitungFactorial(8));
   }

   public int HitungFactorial(int x){
         if(x==1){
```

```
                return 1;
          }
          else {
                return x * HitungFactorial(x-1);
          }
    }
}
```

Contoh Program Modul 2 yang 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;
             }
    }
}
```

keluaran program di atas:

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