[4] Array & LinkedList

Array

- ✧ Array merupakan tipe data yang dapat menampung lebih dari satu elemen yang bertipe sama.
- ✧ Jumlah tempat harus dipesan terlebih dahulu.
- ✧ Setiap tempat pada array diberi index.
- ✧ Elemen pada array dapat di akses dengan menggunakan index.

Array 1 Dimensi

Sebagai contoh kita mempunyai 5 buah data dengan nilai 1, 2, 3, 4, dan 5 dengan tipe data integer *(int)*. Kita bisa saja mendeklarasikan tiap datanya dalam 1 variabel, misalnya :

A =	1	B =	2	C =	3	D =	4	E =	5

Tapi deklarasi tersebut menjadi tidak efektif karena terlalu banyak variable yang digunakan. Kita masih dapat menjangkaunya jika hanya 5 buah.

Tetapi bagaimana jika ada 100 data? Kita harus mendeklarasikan 100 buah variable. Karena itulah kita bisa menggunakan array untuk menyimpan itu semua. 5 variabel tadi dapat dimasukkan ke dalam satu variable “nomor”:

Nomor =	1	2	3	4	5

Sintaks:

1. Deklarasi array

```
//cara pertama
int[] nomor;

//cara kedua
int nomor[];

//cara ketiga
Int[] nomor = new Int[5];
```

2. Pengisian array

```
//saat deklarasi
int nomor[5] = {1,2,3,4,5};

//satu-persatu
nomor[0] = 1;
nomor[1] = 2;
nomor[2] = 3;
nomor[3] = 4;
nomor[4] = 5;

//menggunakan perulangan
for(int i=0; i<5 ;i++){
  nomor[i] = i+1;
}
```

3. Pengaksesan Elemen Array

```
//satu elemen
System.out.println(nomor[0]);
System.out.println(nomor[1]);
System.out.println(nomor[2]);
System.out.println(nomor[3]);
System.out.println(nomor[4]);

//menggunakan perulangan
for(int i=0; i<5; i++){
  System.out.println(nomor[i]);
}
```

Array Multi Dimensi

Array multidimensi merupakan array yang dapat menampung nomor indeks lebih dari satu macam. Contoh analogi multi dimensi pada nomor terakhir NIM mahasiswa 181101[6110001]

NIM	6	1	1	0001
Arti	Fakultas	Jalur Masuk	Jenis Kelamin	Nomor Urut
indeks	3	2	1	0

Deklarasi array multi dimensi

```
int nomor[][]...[];
```

Array dua dimensi merupakan array multidimensi yang jumlah nomor indeksnya adalah 2 dan biasanya untuk membuat data dalam baris dan kolom. Contoh jika ingin membuat matriks 2x2 dengan 4 variabel

A 2 B 4

C 3 D 8

Kita dapat membuat 4 variabel tersebut ke dalam 1 variabel

Matriks =

2	4
3	8

Sintaks:

1. Deklarasi array 2 dimensi

```
//cara pertama
int matriks[][];

//cara kedua
int[][] matriks;

//cara ketiga
Int matriks[][] = new Int[][];
```

2. Pengisian array

```
//baris 1 kolom 1
matriks[0][0] = 2;

//baris 1 kolom 2
```

```
matriks[0][1] = 4

//baris 2 kolom 1
matriks[1][0] = 3

//baris 2 kolom 2
matriks[1][1] = 8
```

3. Pengaksesan Array

```
//baris 1 kolom 1
System.out.println(matriks[0][0]);

//baris 1 kolom 2
System.out.println(matriks[0][1]);

//baris 2 kolom 1
System.out.println(matriks[1][0]);

//baris 2 kolom 2
System.out.println(matriks[1][1]);
```

Latihan Array 2 Dimensi

a. Buatlah array 3x3 dengan nilai masing-masingnya sebagai berikut dan tampilkan ke layar!

4	8	1
5	5	2
0	5	3

Jawaban

```
int matriks[3][3];

//untuk baris pertama
matriks[0][0] = 4;
matriks[0][1] = 8;
matriks[0][2] = 1;
//untuk baris kedua
matriks[1][0] = 5;
matriks[1][1] = 5;
matriks[1][2] = 3;
//untuk baris ketiga
matriks[2][0] = 0;
matriks[2][1] = 5;
matriks[2][2] = 3;

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

b. Buatlah array 3x4 dengan nilai dimulai dari 1 sampai 11 dan tampilkan ke layar!

1	2	3	4
5	6	7	8
9	10	11	12

Jawaban :

```
//deklarasi array
int matriks[3][4];

//menginisialisasikan nilai matriks
int nilai = 1;
for(int i=0; i<3 ;i++){
  for(int j=0;j<4;j++){
    matriks[i][j] = nilai;
    nilai++;
  }
}

//mencetak matriks
for(int i=0;i<3;i++){
  for(int j=0;j<4;j++){
    System.out.print(matriks[i][j]);
    System.out.print("  ");
  }
  System.out.println("");
}
```

Mengenal Java Collections Framework

Collection = container
Collection digunakan untuk menyimpan, mengambil, dan memanipulasi data.

Salah satu collection paling sederhana adalah Array. Namun dalam Collection Framework, Java menawarkan berbagai bentuk lain dari Collection.

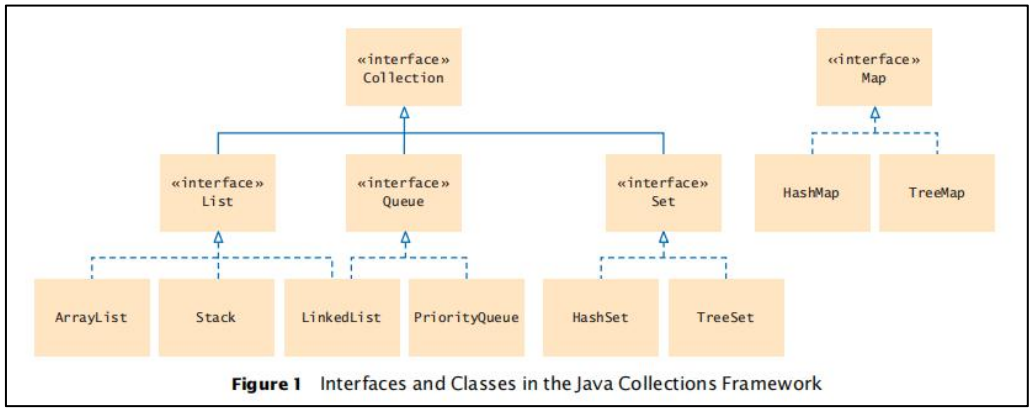

Figure 1 Interfaces and Classes in the Java Collections Framework

Semua class yang berhunbungan dengan Collection tergabung dalam Java Collection Framework dan terdapat dalam package java.util, punya 2 interface utama : Collection dan Map.

- ✧ List adalah collection yang mengingat urutan dari elemennya sendiri
- ✧ Pada queue, kita hanya bisa menambah item baru pada ujung yang satu (tail/ekor) dan menghapusnya dari ujung yang lain (kepala/head). Seperti konsep antrian
- ✧ Stack mengingat urutan dari elemennya sendiri, namun hanya bisa menambah dan menghapus elemen pada satu ujung saja yaitu top (bagian atas)
- ✧ Set adalah unordered collection, cara pengelompokkanya seperti himupan, setiap elemennya harus unik
- ✧ Map menyimpan asosiasi/keterkaitan key dan value pada objek

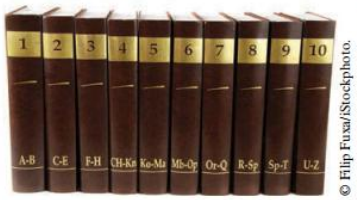

Figure 2 A List of Books

Figure 3 A Set of Books

Figure 4 A Stack of Books

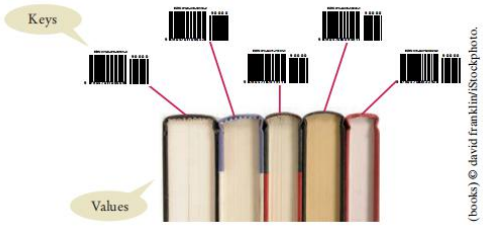

Figure 5
A Map from Bar Codes to Books

Perbedaan Array dan Collection Framework

✧ Jumlah tempat harus dipesan terlebih dahulu. ✧ Setiap tempat pada array diberi index. ✧ Objek pada array dapat di akses dengan menggunakan index.	✧ Jumlah dinamis. ✧ Tidak menggunakan index ✧ Class collection memiliki method yang digunakanuntuk mengakses data

Linked List

- ✧ Linked list adalah struktur data linear yang digunakan untuk mengumpulkan susunan objek yang memungkinkan penambahan dan penghapusan elemen di tengah susunan secara efisien
- ✧ Linked list berisi kumpulan node yang tersusun secara sekuensial. Node adalah objek yang menyimpan data dari sebuah elemen beserta penunjuk ke node disebelahnya.

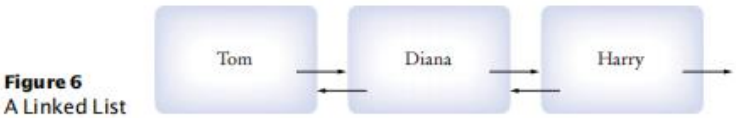

Figure 6
A Linked List

When you insert a new node into a linked list, only the neighboring node references need to be updated (see Figure 7).

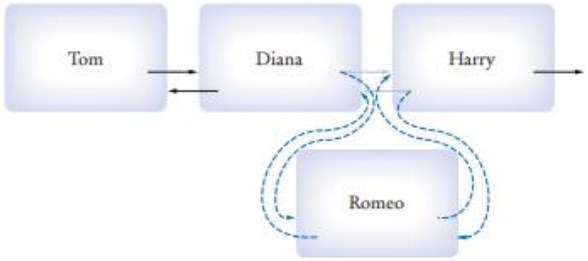

Figure 7 Inserting a Node into a Linked List

Java Librarymenyediakan implementasi linked list dalam class LinkedList, berikut ini contoh sintaks yang dapat langsung kita gunakan:

LinkedList<String> list = new LinkedList<>();	Sebuah list kosong
list.addLast("Harry");	Menambah sebuah elemen pada bagian akhir list
list.addFirst("Diana");	Menambah sebuah elemen pada awal list. Sekarang isi list adalah [Diana, Harry]
list.getFirst();	Get elemen yang tersimpan di bagian awal list: “Diana”
List.getLast();	Get elemen yang tersimpan di bagian akhir list: “Harry”
String removed = list.removeFirst();	Hapus elemen pertama dari list, sehingga tersisa [Harry].

Contoh program:

```
import java.util.*;

public class Test

{

    public static void main(String args[])

    {

        // Creating object of class linked list

        LinkedList<String> object = new LinkedList<String>();
```

```
// Adding elements to the linked list
object.add("A");
object.add("B");
object.addLast("C");
object.addFirst("D");
object.add(2, "E");
object.add("F");
object.add("G");
System.out.println("Linked list : " + object);

// Removing elements from the linked list
object.remove("B");
object.remove(3);
object.removeFirst();
object.removeLast();
System.out.println("Linked list after deletion: " + object);

// Finding elements in the linked list
boolean status = object.contains("E");

if(status)
    System.out.println("List contains the element 'E' ");
else
    System.out.println("List doesn't contain the element 'E'");
```

```
        // Number of elements in the linked list
        int size = object.size();
        System.out.println("Size of linked list = " + size);

        // Get and set elements from linked list
        Object element = object.get(2);
        System.out.println("Element returned by get() : " + element);
        object.set(2, "Y");
        System.out.println("Linked list after change : " + object);
    }
}
```