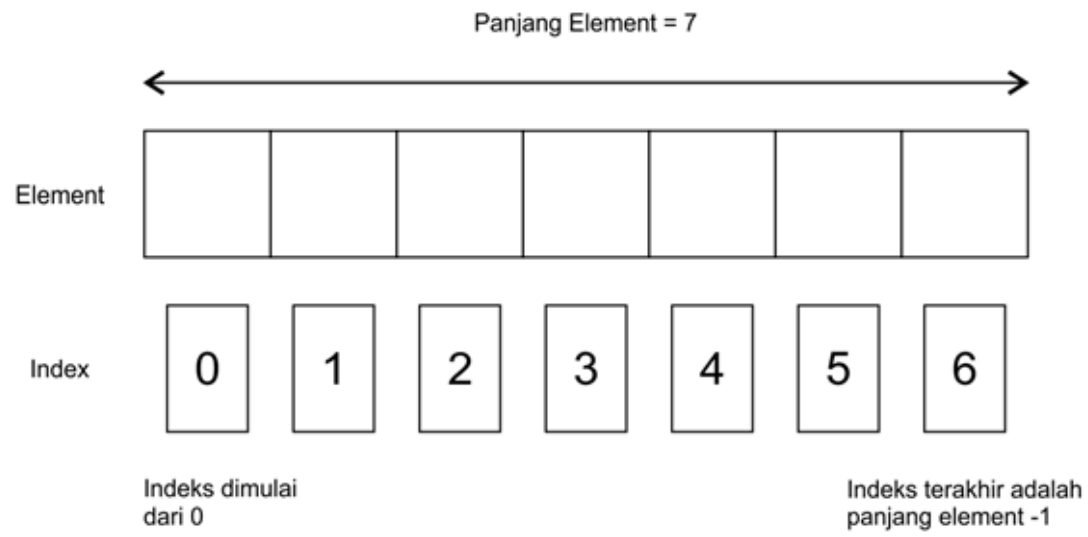
Array

Array adalah obyek yang bisa digunakan untuk menyimpan kumpulan data lebih dari satu dengan tipe sama.

Array memiliki jumlah data yang *fixed* (tetap).

Ilustrasi Array

Array bisa diilustrasikan dengan gambar berikut:



Angka di atas menunjukkan indeks dari array tersebut, yakni dimulai dari 0 sampai 6. Setiap kolom nilai bisa disebut sebagai elemen.

Deklarasi Array

Untuk mendeklarasikan suatu array kita bisa menuliskannya seperti berikut ini.

```
1. void cobaArray(){
2.    double[] arrA;
3.    arrA = new double[10];
4. }
```

Tanda `[]` menunjukkan bahwa obyek tersebut adalah sebuah array. Kemudian, setelah membuat suatu objek *array* yang memiliki tipe data double, kita harus mendefinisikan *length* (panjang) dari *array* tersebut. Untuk mendefinisikan panjangnya, kita harus menuliskan jumlahnya dalam nominal integer di dalam tanda `[]`.

Perlu Anda ketahui juga bahwa hanya terdapat 2 jenis penulisan array yang diperbolehkan di dalam Java, yakni:

```java
void cobaArray(){
    // Cara pertama
    double[] arrA = new double[10];

    // Cara kedua
    double arrB[] = new double[10];
}
```

Keduanya diperkenankan, akan tetapi cara pertama lebih lazim digunakan. Oleh karena itu pada materi ini kita akan menggunakan cara pertama dalam penulisannya.

Inisiasi Array

Untuk inisiasi suatu array, ada 2 cara yang bisa digunakan.

Pertama, menggunakan tanda `{}` (sepasang urung kurawal atau *braces/curly brackets*) dan memisahkan nilai antar elemen dengan tanda koma.

Codelab Inisiasi Array

Selanjutnya kita coba melakukan inisiasi Array agar kalian makin paham.

1. Buatlah proyek baru dengan nama Array dengan nama package `com.dicoding.javafundamental.array` di dalamnya:

2. Buatlah sebuah kelas baru dengan nama InisiasiArray, kemudian tambahkan kode berikut:

```java
package com.dicoding.javafundamental.array;

public class InisiasiArray {
    public static void main(String[] args) {
        int[] arrInt = new int[]{1, 2, 3, 4, 5, 6};

        System.out.println(arrInt[0]);
        System.out.println(arrInt[1]);
        System.out.println(arrInt[2]);
        System.out.println(arrInt[3]);
        System.out.println(arrInt[4]);
        System.out.println(arrInt[5]);
    }
}
```

Cara di atas merupakan salah satu contoh untuk melakukan insialisasi array.

3. Selanjutnya jalankanlah kode di atas pada IDE yang kalian gunakan. Bila sukses, seharusnya Console akan menampilkan output seperti ini.

```
1
2
3
4
5
6
```

4. Selanjutnya cara kedua dengan melakukan inisiasi array per elemen. Ubahlah kode di dalam kelas `InisiasiArray` menjadi seperti ini:

```
1.  package com.dicoding.javafundamental.array;
2.
3.  public class InisiasiArray {
4.      public static void main(String[] args) {
5.          int[] arrInt = new int[6];
6.          arrInt[0] = 1;
7.          arrInt[1] = 2;
8.          arrInt[2] = 3;
9.          arrInt[3] = 4;
10.         arrInt[4] = 5;
11.         arrInt[5] = 6;
12.
13.         System.out.println(arrInt[0]);
14.         System.out.println(arrInt[1]);
15.         System.out.println(arrInt[2]);
```

5. Jalankanlah kembali kode di atas bila sukses, seharusnya Console akan menampilkan output seperti ini:

```
1
2
3
4
5
6
```

6. Selain itu, cara pertama juga kita sederhanakan dengan menghilangkan kata `new` + tipe data-nya, menjadi seperti ini:

```
1.  package com.dicoding.javafundamental.array;
2.
3.  public class InisiasiArray {
4.      public static void main(String[] args) {
5.          int[] arrInt = {1, 2, 3, 4, 5, 6};
6.
7.          System.out.println(arrInt[0]);
8.          System.out.println(arrInt[1]);
9.          System.out.println(arrInt[2]);
10.         System.out.println(arrInt[3]);
11.         System.out.println(arrInt[4]);
12.         System.out.println(arrInt[5]);
13.     }
14. }
```

7. Jalankanlah kembali kode di atas maka tampilan output akan jadi seperti ini:

```
1
2
3
4
5
6
```

Bedah Kode Kegunaan Array

Lalu apa kegunaan utama dari suatu *array*? Seperti yang sudah dijelaskan sebelumnya, *array* berupa kumpulan data. Ini berarti bahwa satu objek *array* dapat menyimpan beberapa data sekaligus.

Jika dibandingkan dengan variabel *primitif* maka untuk membuat 6 buah *variabel integer* kita bisa menuliskannya seperti berikut:

```java
int int1 = 1;
int int2 = 2;
int int3 = 3;
int int4 = 3;
int int5 = 5;
int int6 = 6;
```

Penulisan kode di atas bisa kita konversi dengan menggunakan sebuah array yang memiliki 6 elemen, seperti ini:

```java
int[] arrInt = new int[6];
```

Bayangkan jika program kita mengharuskan penulisan obyek sebanyak 1 juta? Tentu kita dapat menghemat banyak energi dalam menulis baris kode dengan array!.

Namun sebenarnya proses konversi kode di atas belum selesai karena kita belum memberikan nilai pada setiap elemennya. Mari masuk ke bab selanjutnya.

Akses Array

Pada latihan sebelumnya Anda sudah mengetahi bagaimana cara mengakses elemen dari suatu array, yaitu dengan menggunakan tanda `[]` (tanda kurung siku atau *square backets*) dan indeksnya. Perhatikan kode di bawah ini untuk akses array.

```java
int[] arrInt = {1, 2, 3, 4, 5, 6};

System.out.println(arrInt[0]);
System.out.println(arrInt[1]);
System.out.println(arrInt[2]);
System.out.println(arrInt[3]);
System.out.println(arrInt[4]);
System.out.println(arrInt[5]);
```

Output-nya:

```
1
2
3
4
5
6
```

Proses Array

Sebelumnya telah dijelaskan mengenai beberapa *poin* penting mengenai *array*, tapi yang tidak kalah penting adalah hubungan antara *array* dengan *looping*. Kenapa looping? Kembali ke pertanyaan bagaimana jika *array*-nya memiliki 1 juta panjang elemen, maka kita harus menggunakan looping untuk melakukan proses elemennya.

Jika kita ingin menampilkan nilai array yang memiliki panjang 1 juta dengan menggunakan akses, tiap indeksnya akan kurang lebih seperti ini.

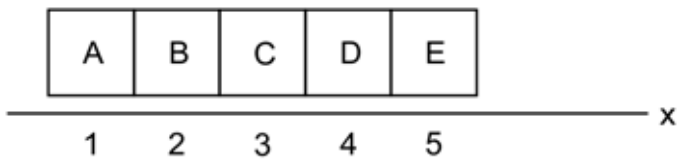
```
1.  public class LoopingArray {
2.      public static void main(String[] args) {
3.          int[] arrInt = new int[]{1, 2, 3, 4, 5, ..., 999998, 999999, 1000000};
4.          System.out.println(arrInt[0]);
5.          System.out.println(arrInt[1]);
6.          System.out.println(arrInt[2]);
7.          System.out.println(arrInt[3]);
8.
9.          ...
10.
11.         System.out.println(arrInt[999998]);
12.         System.out.println(arrInt[999999]);
13.     }
14. }
```

Jika kita konversi kode di atas dengan menggunakan *looping*, kita bisa mendapatkan panjang *array* dengan mengakses *atribut* `length`. Kodenya akan menjadi seperti ini.

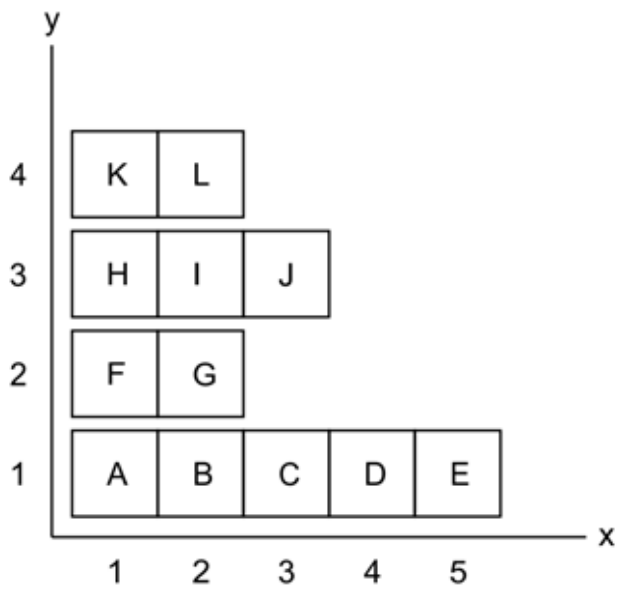
```
1.  public class LoopingArray {
2.      public static void main(String[] args) {
3.          int[] arrInt = new int[1000000];
4.
5.          // Inisiasi dan menampilkan arrInt dari elemen ke 1 sampai 1000000
6.          for (int x = 0; x < arrInt.length; x++) {
7.              arrInt[x] = x + 1;
8.              System.out.println(arrInt[x]);
9.          }
10.     }
11. }
```

Multi Dimensional Array

Seperti dengan namanya "*multi dimensional*" berarti array bisa memiliki lebih dari 1 dimensi. Pada penjelasan sebelumnya kita baru hanya menggunakan *array* yang memiliki 1 dimensi. Jika diilustrasikan dengan dimensi maka 1 dimensi *array* baru hanya menggunakan sumbu x, seperti ini.



Untuk 2 dimensi array kita bisa menambahkan sumbu y, dengan ilustrasi seperti berikut.



Penulisan kode untuk 2 dimensi array adalah seperti ini.

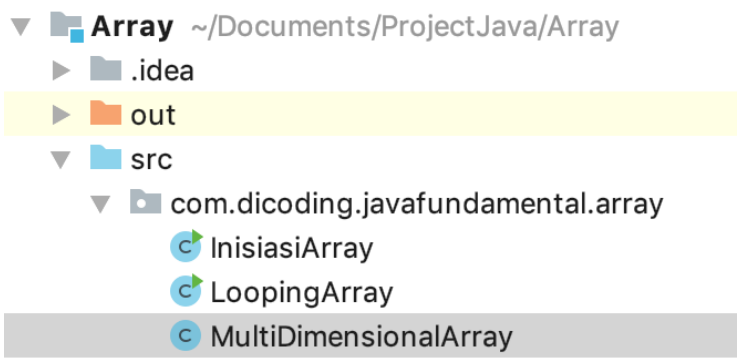
```
1. void cobaArray2D() {
2.     int[][] arrInt = new int[][];
3. }
```

Perhatikan bahwa ada 2 tanda `[]` (bracket) yaitu `[][]`. Tanda `[]` pertama adalah dimensi pertama dan tanda `[]` kedua adalah dimensi kedua. Sama juga dengan dimensi yang lebih dari itu, misalnya 3 dimensi, maka ada 3 tanda `[]` yaitu `[][][]`.

Kemudian untuk panjang elemen, pada dimensi kedua tiap panjang elemen-nya pun bisa bervariasi. Misalnya kita ingin membuat array 2 dimensi, panjang elemen dimensi 1 nya adalah 2, kemudian panjang elemen pada dimensi keduanya adalah 2 dan 3 secara berurutan. Penasaran? Langsung kita praktikan!

Codelab Multi Dimensional Array

1. Bukalah kembali proyek Array dan buatlah kelas baru dengan nama `MultiDimensionalArray`.



2. Masukkan kode berikut ke dalam kelas `MultiDimensionalArray`:

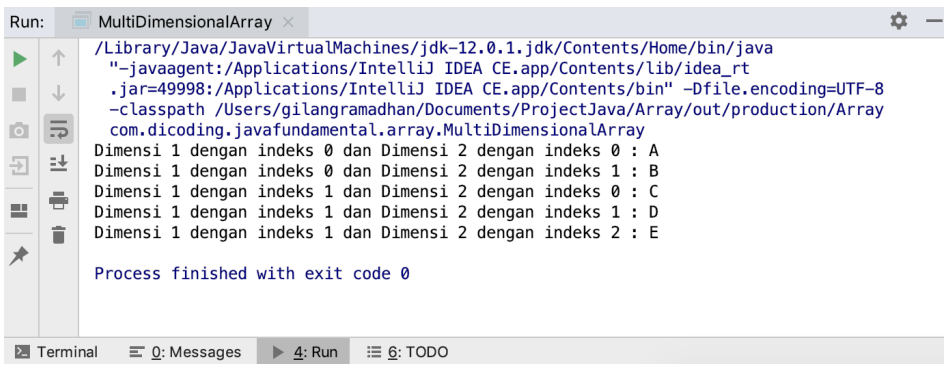
```java
package com.dicoding.javafundamental.array;

public class MultiDimensionalArray {
    public static void main(String[] args) {
        char[][] arrChar = new char[2][];
        arrChar[0] = new char[2];
        arrChar[1] = new char[3];
    }
}
```

3. Selanjutntya tambahkan kode berikut untuk menambahkan elemen di tiap indeksnya:

```java
package com.dicoding.javafundamental.array;

public class MultiDimensionalArray {
    public static void main(String[] args) {
        char[][] arrChar = new char[2][];
        arrChar[0] = new char[2];
        arrChar[1] = new char[3];

        // Dimensi 1 yang indeksnya 0 memiliki panjang elemen 2
        arrChar[0][0] = 'A';
        arrChar[0][1] = 'B';

        // Dimensi 1 yang indeksnya 1 memiliki panjang elemen 3
        arrChar[1][0] = 'C';
        arrChar[1][1] = 'D';
```

4. Jalankan kode di atas maka hasilnya akan jadi seperti ini:

```
Dimensi 1 dengan indeks 0 dan Dimensi 2 dengan indeks 0 : A
Dimensi 1 dengan indeks 0 dan Dimensi 2 dengan indeks 1 : B
Dimensi 1 dengan indeks 1 dan Dimensi 2 dengan indeks 0 : C
Dimensi 1 dengan indeks 1 dan Dimensi 2 dengan indeks 1 : D
Dimensi 1 dengan indeks 1 dan Dimensi 2 dengan indeks 2 : E
```

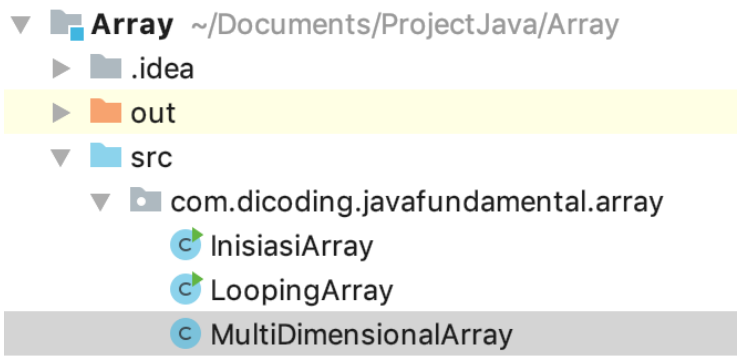


IndexOutOfBounds

Terakhir, selama menggunakan array Anda perlu berhati-hati. Jika tidak, Anda akan sering menemui eror `indexOutOfBounds`. Ini adalah eror yang disebabkan oleh percobaan saat melakukan akses indeks yang tidak dimiliki oleh array. Misalnya kita melakukan akses pada indeks 4 sedangkan array tersebut hanya memiliki 4 elemen. Ingatlah bahwa indeks dimulai dari 0 dan indeks terakhir adalah panjang element - 1. Agar Anda semakin paham, mari kita coba praktikan!

Codelab IndexOutOfBounds

1. Bukalah kembali proyek Array dan buatlah kelas baru dengan nama `IndexOutOfBound`.



2. Masukkan kode berikut ke dalam kelas `IndexOutOfBound`:

```
1.  package com.dicoding.javafundamental.array;
2.
3.  public class IndexOutOfBound {
4.      public static void main(String[] args) {
5.          int[] arrA = {1, 2, 3, 4};
6.          System.out.println(arrA[0]);
7.          System.out.println(arrA[1]);
8.          System.out.println(arrA[2]);
9.
10.         // Akses indeks ke 4
11.         System.out.println(arrA[4]);
12.     }
13. }
```

3. Jalankan kode di atas maka hasilnya akan jadi seperti ini:

```
1
2
3
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 4
at com.dicoding.javafundamental.array.IndexOutOfBound.main(IndexOutOfBound.java:11)
```

Bedah Code IndexOutOfBounds

Pada kode di atas, kita melakukan perulangan dengan mengeluarkan seluruh isi array. Metode ini dapat kita sebut sebagai iterasi. Cara di atas kita tidak perlu menginisalisasi angka awal dan akhir pada sebuah perulangan. Menggunakan cara ini sangat direkomendasikan.

Selesai sudah materi tentang array yang akan sangat sering kita gunakan di dalam pemrograman. Pastikan Anda telah menguasainya dengan baik!

‹ Sebelumnya

Selanjutnya ›