Bab 4

Array

Tujuan

1. Memahami konsep array dalam pemrograman
2. Memahami konsep array 1 dimensi dan multidimensi
3. Mampu memanipulasi array

Array merupakan struktur data dasar dalam dunia komputer yang berfungsi untuk menyimpan banyak data dengan tipe yang sama dalam satu variabel. *Array* pada dasarnya mempunyai ukuran yang *fix* (tetap) sehingga tidak dapat diubah setelah dideklarasikan. Sebagian besar bahasa pemrograman memulai indeksnya dari 0, indeks digunakan untuk mengakses elemen *array*, jika terdapat *array* sepanjang ***n***, maka indeks terakhir dari *array* tersebut adaah ***n-1*.**

4.1 Deklarasi Array

Pendeklarasian *Array* sama seperti dengan pendeklarasian variabel,diawali dengan tipe data lalu diikuti nama variabel, bedanya, *array* ditandai dengan **[]** (kurung siku), ada beberapa cara dalam pendeklarasian *array*.

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

    // 1. tipeData[] namaVariabel;
    int[] arrayOfInt;

    // 2. tipeData namaVariabel[];
    char arrayOfChar[];

    // 3. tipeData[] namaVariabel = new tipeData[n];
    double[] arrayOfDouble = new double[10];

    // 3. tipeData namaVariabel[] = new tipeData[n];
    double arrayOfDouble[] = new double[10];
}
```

Drawing 19: Cara Pendeklarasian Array

4.2 Inisialisasi Array

Sama dengan deklarasi, inisialisasi *array* juga terdapat berbagai cara, *array* dapat dideklarasikan sekaligus diinisialisasikan atau dilakukan dengan terpisah. Inisialisasi *array* dapat ditandai dengan **{}** (kurung kurawal) atau dengan mengakses indeks *array*.

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

    // Cara Pertama
    int[] arrayOfInt = new int[]{1, 2, 3};

    // Cara Kedua
    char[] arrayOfChar = new char[4];
    arrayOfChar[0] = ‘h’;
    arrayOfChar[1] = ‘a’;
    arrayOfChar[2] = ‘l’;
    arrayOfChar[3] = ‘o’;

    // Cara Ketiga
    float[] arrayOfFloat = {1f, 2f, 3f};
}
```

Drawing 20: Berbagai Cara Inisialisasi Array

4.3 Mengakses Elemen Array

Array memiliki indeks, indeks dapat digunakan untuk mengakses elemen *array* dengan cara memanggil nama *array* diikuti dengan kurung siku dan indeks yang ingin diakses.

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

    int[] arr = {0, 4, 9};
    int n = arr[2];
    System.out.println(n)
}
```

Drawing 21: Mengakses Array Menggunakan Indeks

Cara di atas dapat digunakan jika ingin mengakses satu elemen *array*, namun ketika ingin mengakses beberapa elemen, akan lebih efisien jika menggunakan perulangan, selain menggunakan perulangan biasa, terdapat juga perulangan khusus untuk *array* yaitu ***for each loop***.

```
public static void main (String[] args) {
    char[] unhas = {‘U’, ‘N’, ’H’, ’A’, ’S’};
    // menggunakan perulangan biasa
    for (int i = 0; i < unhas.length; i++) {
         System.out.print(unhas[i]);
    }

    // menggunakan for each
    for (char e : unhas) {
         System.out.print(e);
    }
}
```

Drawing 22: Mengakses Array Menggunakan Perulangan

Fungsi ***.length*** pada program di atas adalah untuk mendapatkan panjang array, karena jika kita mengakses array di luar batas indeksnya maka akan terjadi *ArrayIndexOutOfBoundsException*. ***for each*** dapat dikatakan lebih aman terhadap *index out of bounds*, karena secara otomatis, banyaknya perulangan akan menyesuaikan dengan panjang array.

4.4 Multidimensional Array

Multidimensioal Array dapat diilustrasikan sebagai *array* dalam *array*, jadi setiap satu elemen *array* mengandung *array*, dimensi dari sebuah *array* dapat ditandai dengan banyakya **[]** (kurung siku). Untuk mengakses *array* multidimensi, maka dibutuhkan perulangan bersarang (*Nested Loop*) sebanyak jumlah dimensinya.

```
public static void main (String[] args) {
    int[][] eyes = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
    for (int i = 0; i < eyes.length; i++) {
         for (int j = 0; j < eyes[i].length; j++) {
              System.out.print(eyes[i][j] + “\t”);
         }
         System.out.println();
    }
}
```

Drawing 23: Nested Loop Untuk Mengakses Array 2 Dimensi

Dalam *Nested Loop* di atas, perulangan *j* akan diselesaikan terlebih dahulu, lalu perulangan *i* akan berlanjut, sehingga setiap kali perulangan *j* selesai, iterasi *i* akan bertambah satu dan *j* kembali ke nol.