Bab 7. Array

OBJEKTIF :

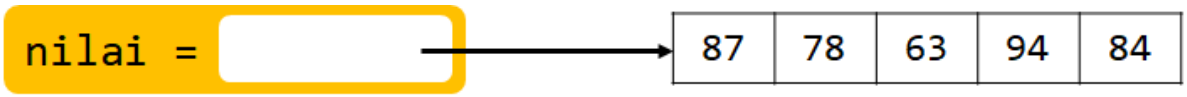
1. Mahasiswa mampu memahami mengenai materi Array pada Java.
2. Mahasiswa mampu memahami mengenai penggunaan Array pada program Java.
3. Mahasiswa mampu mensimulasikan penggunaan Array pada program Java untuk kejadian di dunia nyata.

7.1 Array

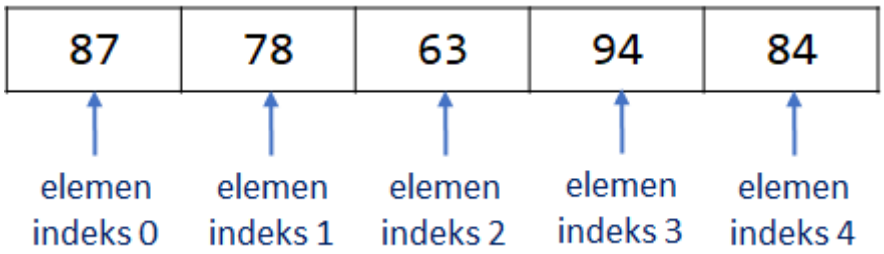
Array adalah sebuah object yang dapat menyimpan koleksi nilai-nilai bertipe sama. Kita menggunakan array untuk mengelompokkan data-data dalam sebuah variabel sehingga memudahkan kita untuk memanipulasi atau mengolah kelompok data-data tersebut. Sebagai contoh, misalkan kita membuat program yang mengolah nilai-nilai ujian dari lima mahasiswa dalam sebuah kelas. Jika kita memperlakukan masing-masing nilai ujian mahasiswa sebagai data tunggal, kita harus membuat lima variabel yang masing-masing variabelnya digunakan untuk menyimpan nilai masing-masing mahasiswa:

```
int nilai1 = 87;
int nilai2 = 78;
int nilai3 = 63;
int nilai4 = 94;
int nilai5 = 84;
```

Cara di atas tidaklah efisien. Kita dapat menuliskan program pengolahan koleksi data yang efisien dengan menggunakan array. Sebagai contoh, kita dapat menyimpan semua nilai mahasiswa menggunakan sebuah array dan mereferensikan array tersebut dengan sebuah variabel, seperti diilustrasikan pada gambar berikut:



Anggota-anggota data pada array disebut dengan elemen. Sama seperti pada `String`, setiap elemen dalam array mempunyai indeks yang menyatakan posisi elemen tersebut dalam array. Misalkan array seperti contoh di atas mempunyai indeks seperti terlihat pada gambar berikut:



Untuk menggunakan array, kita pertama harus mendeklarasikan sebuah variabel yang mereferensikan array seperti contoh berikut:

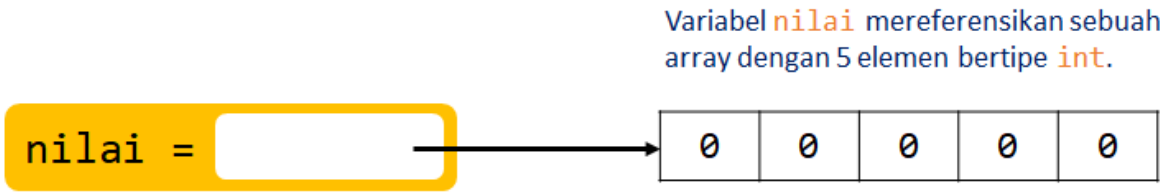
```
int[] nilai;
```

Statement di atas mendeklarasikan variabel `nilai` sebagai variabel referensi array yang semua elemen-elemennya bertipe `int`. Perhatikan bahwa statement di atas mirip dengan deklarasi dari variabel `int` biasa dengan perbedaan pada tanda kurung kotak setelah keyword `int`. Kurung kotak ini menandakan bahwa variabel ini adalah sebuah referensi ke array yang elemen-elemennya bertipe `int`.

Setelah kita mendeklarasikan variabel referensi array, langkah selanjutnya adalah membuat sebuah array dengan keyword `new` lalu menugaskan alamat array tersebut ke variabel `nilai`. Statement berikut mencontohkan langkah ini:

```
nilai = new int[5];
```

Angka 5 yang ditulis di dalam tanda kurung kotak menandakan banyaknya elemen yang akan disimpan oleh array tersebut. Angka ini sering disebut sebagai pendeklarasi ukuran array. Setelah statement di atas dieksekusi, `nilai` akan mereferensikan sebuah array yang menyimpan lima elemen, yang setiap elemennya bertipe `int`. Gambar berikut mengilustrasikan ini:



Perhatikan pada gambar semua elemen dalam array bernilai 0. Ini karena secara default, Java menginisialisasi nilai-nilai elemen array dengan nilai 0 saat array dibuat.

Kita juga dapat menuliskan deklarasi dan pembuatan array ini dalam satu baris seperti berikut:

```
int[] nilai = new int[5];
```

Array-array dengan elemen-elemen dari tipe data apapun dapat dideklarasikan. Berikut adalah contoh-contoh deklarasi array dan pembuatan array untuk array yang elemen-elemennya bertipe data selain `int`:

```
float[] temperatur = new float[100];
char[] himpHuruf = new char[41];
long[] unit = new long[50];
double[] ukuran = new double[1200];
```

Pendeklarasi ukuran array yang dituliskan dalam statement deklarasi haruslah berupa ekspresi yang menghasilkan integer non-negatif. Kita dapat menuliskan literal integer non-negatif seperti pada contoh-contoh sebelumnya, atau menuliskan sebuah variabel untuk menandakan banyaknya elemen array. Praktik yang sering dilakukan adalah dengan menggunakan konstanta sebagai pendeklarasi ukuran array, seperti contoh berikut:

```
final int BANYAK_ELEMEN = 6;
int[] bilBulat = new int[BANYAK_ELEMEN];
```

Hal penting yang perlu diingat mengenai ukuran array adalah setelah array dibuat, ukurannya tidak dapat diubah.

Mengakses Elemen Array

Setiap elemen dari array dapat diakses menggunakan notasi subscript. Sebagai contoh, untuk memberikan nilai 87 ke elemen dengan indeks 0 pada array `nilai`, kita menuliskan statement assignment dengan notasi subscript seperti berikut:

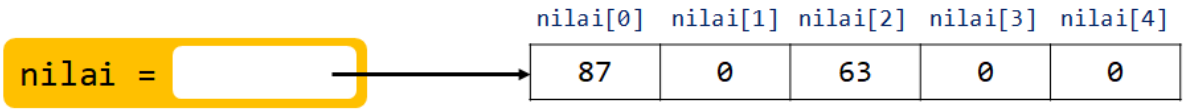
```
nilai[0] = 87;
```

`nilai[0]` adalah notasi subscript untuk mengakses elemen indeks 0 dari array `nilai`. Angka 0 yang kita tulis dalam tanda kurung adalah indeks dari elemen yang ingin diakses.

Jika kita ingin memberikan nilai 63 ke elemen dengan indeks 2 dari array `nilai` kita menuliskan statement berikut:

```
nilai[2] = 63;
```

Gambar berikut mengilustrasikan isi dari array `nilai` setelah statement-statement di atas dieksekusi:



Kita juga menggunakan notasi subscript untuk mendapatkan nilai elemen array pada indeks tertentu. Sebagai contoh, statement berikut mencetak nilai elemen dengan indeks 2 dari array `nilai`:

```
System.out.println(nilai[2]);
```

Program berikut mencontohkan penggunaan array:

Program (DemoArray.java)

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan array.
*/
public class DemoArray
{
    public static void main(String[] args)
    {
          final int BANYAK_NILAI = 3;                 // Banyaknya nilai untuk
disimpan
        int[] nilai = new int[BANYAK_NILAI];    // Array dari nilai

        // Buat object Scanner untuk mendapatkan input
        Scanner keyboard = new Scanner(System.in);

        System.out.println("Masukkan nilai ujian!");

        // Dapatkan nilai ujian 1
        System.out.print("Nilai ujian 1: ");
        nilai[0] = keyboard.nextInt();
```

```
        // Dapatkan nilai ujian 1
        System.out.print("Nilai ujian 2: ");
        nilai[1] = keyboard.nextInt();

        // Dapatkan nilai ujian 1
        System.out.print("Nilai ujian 3: ");
        nilai[2] = keyboard.nextInt();

        // Tampilkan nilai ujian yang dimasukkan pengguna
        System.out.println("Nilai ujian yang Anda masukkan: ");
        System.out.println("Ujian 1 = " + nilai[0]);
        System.out.println("Ujian 2 = " + nilai[1]);
        System.out.println("Ujian 3 = " + nilai[2]);
    }
}
```

Output Program (DemoArray.java)

```
Masukkan nilai ujian!
Nilai ujian 1: 96
Nilai ujian 2: 89
Nilai ujian 3: 82
Nilai ujian yang Anda masukkan:
Ujian 1 = 96
Ujian 2 = 89
Ujian 3 = 82
```

Angka indeks pada notasi subscript dapat disimpan dalam sebuah variabel `int`. Ini memungkinkan kita untuk menggunakan sebuah loop untuk mengunjungi setiap elemen dalam array dan melakukan operasi terhadap setiap elemen tersebut. Sebagai contoh, program berikut menyederhakan program `DemoArray.java` dengan menggunakan dua loop `for`: satu untuk menginput nilai-nilai ke array dan lainnya untuk menampilkan isi dari array.

Program (DemoArray2.java)

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan penggunaan
    loop for untuk memproses array.
*/
public class DemoArray2
{
    public static void main(String[] args)
    {
          final int BANYAK_NILAI = 3;                 // Banyaknya nilai untuk
disimpan
        int[] nilai = new int[BANYAK_NILAI];    // Array dari nilai

        // Buat object Scanner untuk mendapatkan input
        Scanner keyboard = new Scanner(System.in);

        System.out.println("Masukkan nilai ujian!");

        for (int index = 0; index < BANYAK_NILAI; index++)
        {
```

```
            System.out.print("Nilai ujian " + (index + 1) + ": ");
            nilai[index] = keyboard.nextInt();
        }

        System.out.println("Nilai ujian yang Anda masukkan: ");

        // Tampilkan nilai ujian yang dimasukkan pengguna
        for (int index = 0; index < BANYAK_NILAI; index++)
        {
            System.out.println("Ujian " + (index + 1) + " = " + nilai[index]);
        }
    }
}
```

Output Program (DemoArray2.java)

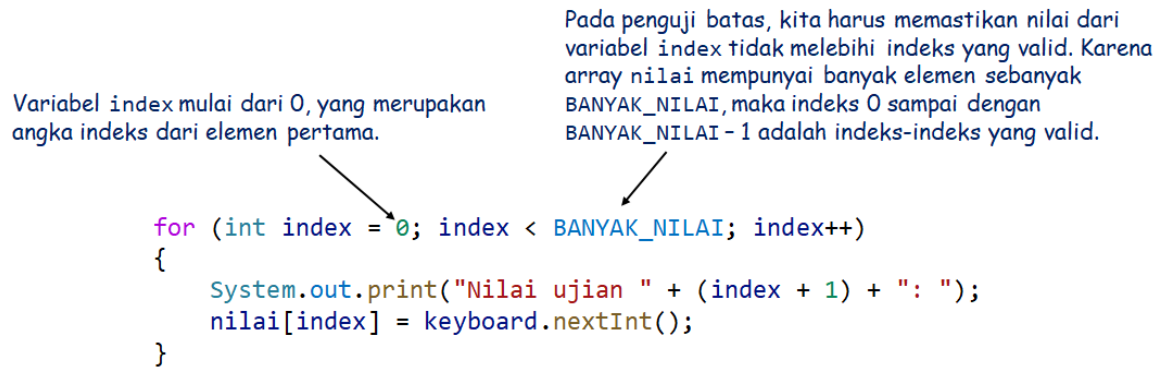
```
Masukkan nilai ujian!
Nilai ujian 1: 96
Nilai ujian 2: 89
Nilai ujian 3: 82
Nilai ujian yang Anda masukkan:
Ujian 1 = 96
Ujian 2 = 89
Ujian 3 = 82
```

Perhatikan pada loop pertama dalam program di atas pada baris 19 sampai dengan 23. Pada loop tersebut kita menggunakan variabel counter `index` yang digunakan pada baris 22:

```
nilai[index] = keyboard.nextInt();
```

Variabel `index` mulai dari 0. Saat iterasi pertama loop, input pengguna disimpan dalam `nilai[0]`. Lalu, `index` diinkrementasi, sehingga bernilai 1. Saat iterasi berikutnya, input pengguna disimpan dalam `nilai[1]`. Ini berlanjut sampai dengan semua elemen-elemen array diberikan nilai dari input pengguna.

Ketika kita menuliskan loop untuk mengakses sebuah array, kita harus memastikan variabel yang kita gunakan sebagai indeks untuk mengakses elemen-elemen dalam array mempunyai nilai berupa angka indeks yang valid. Perhatikan pada program di atas, loop `for` mulai dengan variabel `index` bernilai 0 dan berakhir ketika variabel `index` bernilai 2, sehingga saat loop berjalan, variabel `index` akan bernilai 0 sampai dengan 2. Angka 0 sampai dengan 2 ini merupakan angka-angka indeks yang valid untuk array `nilai`. Gambar berikut menjelaskan loop `for` pada program di atas:



Error Batas Array

Kita harus berhati-hati ketika kita mengakses elemen array di luar batas array. Java tidak memeriksa ini saat kompilasi namun jika di dalam program terdapat penggunaan indeks pada notasi subscript yang mencoba mengakses elemen yang tidak dimiliki array maka program akan menghasilkan error. Misalkan, statement berikut yang membuat sebuah array dengan 10 elemen:

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

Angka indeks yang valid yang dapat digunakan dalam notasi subscript untuk array `bilBulat` adalah 0 sampai dengan 9. Jika sebuah program mencoba menggunakan angka indeks 10, maka program tersebut akan menghasilkan error. Sebagai contoh, program berikut mendeklarasikan array dengan tiga elemen, namum mencoba menyimpan empat nilai dalam array tersebut:

Program (IndeksSalah.java)

```
/*
    Program ini menggunakan angka indeks yang tidak valid
    dalam sebuah array.
*/
public class IndeksSalah
{
    public static void main(String[] args)
    {
        int[] bilBulat = new int[3];

        System.out.println("Saya akan mencoba menyimpan empat " +
                           "angka ke array dengan tiga elemen.");

        for (int indeks = 0; indeks < 4; indeks++)
        {
            System.out.println("Memproses elemen " + indeks);
            bilBulat[indeks] = 10;
        }
    }
}
```

Output Program (IndeksSalah.java)

```
Saya akan mencoba menyimpan empat angka ke array dengan tiga elemen.
Memproses elemen 0
Memproses elemen 1
Memproses elemen 2
Memproses elemen 3
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
        at IndeksSalah.main(IndeksSalah.java:17)
```

Program di atas menghasilkan error ketika mencoba menetapkan nilai ke elemen indeks 3 dari array `bilBulat` yang hanya mempunyai tiga elemen.

Menginisialisasi Array

Kita dapat menginisialisasi elemen-elemen array ketika kita membuat array tersebut. Berikut adalah contoh inisialisasi array saat deklarasi:

```
int[] hari = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
```

Statement di atas mendeklarasikan variabel referensi `hari`, membuat sebuah array dalam memori, dan menyimpan nilai-nilai awal dalam tanda kurung kurawal ke array. Nilai-nilai elemen dalam array mempunyai urutan sesuai dengan yang tertulis pada nilai-nilai dalam tanda kurung kurawal. Pada contoh di atas, `hari[0]` akan menyimpan nilai 31, `hari[1]` akan menyimpan nilai 28, dan seterusnya. Perhatikan kita tidak menggunakan keyword `new` saat membuat array dengan inisialisasi array. Java akan secara otomatis membuat sebuah array dengan nilai-nilai yang dituliskan dalam tanda kurung kurawal. Ukuran array ditentukan secara otomatis berdasarkan banyaknya nilai-nilai yang dituliskan dalam tanda kurung kurawal.

Compiler Java menentukan ukuran dari array dari banyaknya nilai-nilai yang dituliskan dalam kurung kurawal. Karena terdapat 12 nilai dalam contoh inisialisasi array di atas, maka array tersebut akan mempunyai 12 elemen. Program berikut mendemonstrasikan inisialisasi array:

Program (InisialisasiArray.java)

```
/*
    Program berikut mendemonstrasikan pembuatan
    array dengan inisialisasi.
*/
public class InisialisasiArray
{
    public static void main(String[] args)
    {
        int[] hari = {31, 28, 31, 30, 31, 30,
                      31, 31, 30, 31, 30, 31};

        for (int index = 0; index < 12; index++)
        {
            System.out.println("Bulan " + (index + 1) +
                               " mempunyai " + hari[index] +
                               " hari.");
        }
    }
}
```

Output Program (InisialisasiArray.java)

```
Bulan 1 mempunyai 31 hari.
Bulan 2 mempunyai 28 hari.
Bulan 3 mempunyai 31 hari.
Bulan 4 mempunyai 30 hari.
Bulan 5 mempunyai 31 hari.
Bulan 6 mempunyai 30 hari.
Bulan 7 mempunyai 31 hari.
Bulan 8 mempunyai 31 hari.
Bulan 9 mempunyai 30 hari.
Bulan 10 mempunyai 31 hari.
Bulan 11 mempunyai 30 hari.
Bulan 12 mempunyai 31 hari.
```

Kita dapat menuliskan nilai inisialisasi dalam multi baris untuk memudahkan membacanya. Perhatikan pada contoh program di atas, kita menuliskan inisialisasi dari array `hari` dalam dua baris:

```
int[] hari = {31, 28, 31, 30, 31, 30,
              31, 31, 30, 31, 30, 31};
```

7.2 Memproses Array

Memproses elemen-elemen array tidak berbeda dari memproses variabel-variabel lainnya. Kita dapat memperlakukan elemen individu dari array sebagai variabel tunggal. Sebagai contoh, statement berikut mengalikan `jamKerja[3]` dengan variabel `honor`:

```
gaji = jamKerja[3] * honor;
```

Elemen array juga dapat digunakan dalam ekspresi Boolean dengan operator relasional. Sebagai contoh, statement `if` berikut menguji apakah `biaya[20]` kurang dari `biaya[0]`:

```
if (biaya[20] < biaya[0])
{
    // statement-statement.
}
```

Loop `while` berikut mengiterasi sepanjang `nilai[count]` tidak sama dengan 0:

```
while (nilai[count] != 0)
{
    // statement-statement.
}
```

Program berikut mencontohkan penggunaan elemen array dalam operasi aritmatika:

Program (Gaji.java)

```
import java.time.Period;
import java.util.Scanner;

public class Gaji
{
    public static void main(String[] args)
    {
        final int PEGAWAI = 5;      // Banyak pegawai
        double honor;               // Gaji per jam
        double gaji;                // Gaji sebulan

        // Buat sebuah array untuk menyimpan jam kerja pegawai
        int[] jamKerja = new int[PEGAWAI];

        // Buat sebuah object Scanner untuk input keyboard
        Scanner keyboard = new Scanner(System.in);

        // Dapatkan jam kerja untuk masing-masing pegawai
         System.out.println("Masukkan masing-masing jam kerja dari " + PEGAWAI +
" pegawai.");

        for (int index = 0; index < PEGAWAI; index++)
        {
            System.out.print("Jam kerja pegawai ke-" + (index + 1) + ": ");
            jamKerja[index] = keyboard.nextInt();
        }

        // Dapatkan honor per jam untuk semua pegawai
        System.out.print("Masukkan honor per jam untuk semua pegawai: ");
        honor = keyboard.nextDouble();

        // Hitung gaji masing-masing pegawai
        System.out.println("Berikut adalah gaji setiap pegawai:");
        for (int index = 0; index < PEGAWAI; index++)
        {
            gaji = jamKerja[index] * honor;
            System.out.printf("Pegawai %d: Rp. %,.2f\n", (index + 1), gaji);
        }
    }
}
```

Output Program (Gaji.java)

```
Masukkan masing-masing jam kerja dari 5 pegawai.
Jam kerja pegawai ke-1: 10
Jam kerja pegawai ke-2: 20
Jam kerja pegawai ke-3: 30
Jam kerja pegawai ke-4: 40
Jam kerja pegawai ke-5: 50
Masukkan honor per jam untuk semua pegawai: 100000
Berikut adalah gaji setiap pegawai:
Pegawai 1: Rp. 1,000,000.00
Pegawai 2: Rp. 2,000,000.00
Pegawai 3: Rp. 3,000,000.00
Pegawai 4: Rp. 4,000,000.00
Pegawai 5: Rp. 5,000,000.00
```

Panjang Array

Panjang array adalah banyaknya elemen dalam array. Kita dapat mendapatkan panjang dari suatu array dengan ekspresi berikut:

```
varRefArray.length
```

dimana `varRefArray` adalah variabel yang mereferensikan sebuah array. Perhatikan kita tidak menuliskan tanda kurung buka dan tutup setelah `length`.

Cara mendapatkan panjang dari array berbeda dengan cara mendapatkan panjang dari `String`. Pada `String`, kita memanggil method `length` dari object string: `str.length()`. Sedangkan pada object array, `length` bukanlah method dari object array namun merupakan field dari object array. Field adalah data yang dimiliki oleh object (kita akan membahas perbedaan field dan method nanti pada pembahasan class dan object). Untuk mengakses field dari object kita tidak menuliskan tanda kurung buka dan tutup setelah nama field. Sehingga kita mendapatkan panjang array dengan menuliskan: `arr.length` tanpa tanda kurung setelah `length`.

Sebagai contoh, misalkan sebuah array dibuat dengan statement berikut:

```
double[] temperatur = new double[25];
```

Kita dapat menuliskan statement berikut untuk menyimpan panjang dari array `temperatur` ke sebuah variabel:

```
int ukuran = temperatur.length;
```

Karena array `temperatur` mempunyai 25 elemen, maka setelah statement di atas dieksekusi variabel `ukuran` akan menyimpan nilai 25.

Ekspresi untuk mendapatkan panjang array berguna ketika kita memproses keseluruhan isi array dengan loop. Sebagai contoh loop berikut mencetak element-element array:

```
for (int i = 0; i < temperatur.length; i++)
{
    System.out.println(temperatur[i]);
}
```

Pada loop `for` di atas kita menuliskan penguji batas dengan ekspresi `i < temperatur.length` dan pengupdate dengan inkrementasi variabel counter `i`, sehingga loop `for` tersebut akan mengunjungi setiap elemen dari array dimulai dari elemen indeks 0 sampai dengan elemen terakhir (elemen dengan indeks `temperatur.length - 1`).

Loop `for` Enhanced

Loop `for` mempunyai bentuk khusus untuk digunakan dengan tipe data koleksi seperti array. Loop ini disebut loop `for` enhanced. Syntax dari loop `for` enhanced adalah seperti berikut:

```
for (tipeData variabelElemen : array)
{
    statement
    statement
    ...
}
```

Loop `for` enhanced ini diperuntukkan untuk mengiterasi setiap elemen dalam sebuah array. Setiap kali iterasi loop, nilai elemen array disalin ke sebuah variabel. Berikut adalah penjelasan dari syntax loop `for` enhanced:

- `tipeData variabelElemen` adalah deklarasi variabel. Variabel ini akan menerima nilai dari elemen-elemen array saat setiap iterasi loop. Saat iterasi pertama, variabel ini menerima nilai dari elemen pertama; saat iterasi kedua, variabel ini menerima nilai dari elemen kedua; dan seterusnya. Tipe data dari variabel ini harus sama dengan tipe data dari elemen-elemen array yang diiterasi.
- `array` adalah nama array yang ingin diiterasi dengan loop.
- `statement` adalah statement yang dieksekusi saat setiap iterasi loop.

Contoh kode yang menggunakan loop `for` enhanced ini adalah sebagai berikut:

```
int[] himpunanAngka = { 3, 6, 9 };

for (int angka : himpunanAngka)
{
    System.out.println(angka);
}
```

Kode di atas akan memberikan output:

```
3
6
9
```

Berikut adalah contoh kode lain yang menggunakan loop `for` enhanced untuk mengiterasi array dengan elemen-elemen bertipe `double`:

```
double[] nilaiUjian = {78.54, 87.5, 98.4};

for (double nilai : nilaiUjian)
{
    System.out.println(nilai);
}
```

Kode di atas akan memberikan output:

```
78.54
87.5
98.4
```

Perhatikan pada contoh kode di atas, di dalam tanda kurung setelah keyword `for`, kita menuliskan deklarasi `variabelElemen` bernama `nilai` dengan tipe data `double`.

Loop `for` enhanced memudahkan kita ketika kita memerlukan mengakses semua nilai-nilai yang disimpan dalam sebuah array dari elemen pertama ke elemen terakhir. Namun, perlu diperhatikan, loop `for` enhanced ini tidak menggantikan loop `for` biasa untuk memproses array. Tidak semua hal yang dapat dilakukan menggunakan loop `for` biasa dapat dilakukan dengan loop `for` enhanced. Kita tidak dapat menggunakan loop `for` enhanced untuk hal-hal berikut:

- jika kita perlu memodifikasi isi dari elemen-elemen array
- jika kita perlu mengakses elemen-elemen array secara mundur (dari elemen terakhir ke elemen pertama)
- jika kita perlu mengakses hanya sebagian elemen array dan tidak semua elemen array
- jika kita perlu memproses dua atau lebih array dalam loop
- jika kita ingin menggunakan notasi subscript ke elemen tertentu

Dalam hal-hal di atas, kita harus menggunakan loop `for` biasa.

Meminta Pengguna untuk Menentukan Ukuran Array

Salah satu hal yang umum dilakukan dalam program yang memproses array adalah meminta pengguna untuk menentukan ukuran array. Contoh program yang meminta pengguna untuk menentukan ukuran array adalah sebagai berikut:

Program (TampilkanNilaiUjian.java)

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan input pengguna untuk
    menentukan ukuran array.
*/
public class TampilkanNilaiUjian
{
    public static void main(String[] args)
    {
        int bykUjian;       // Banyak ujian
        int[] ujian;        // Array dari nilai ujian

        // Buat object Scanner untuk input keyboard
        Scanner keyboard = new Scanner(System.in);

        // Minta pengguna memasukkan banyak ujian
        System.out.print("Berapa banyak ujian yang ingin dimasukkan? ");
        bykUjian = keyboard.nextInt();

        // Buat sebuah array untuk menyimpan nilai-nilai ujian
        ujian = new int[bykUjian];
```

```
        // Dapatkan masing-masing nilai ujian
        for (int index = 0; index < ujian.length; index++)
        {
            System.out.print("Masukkan nilai ujian " +
                             (index + 1) + ": ");
            ujian[index] = keyboard.nextInt();
        }

        // Tampilkan nilai ujian
        System.out.println();
         System.out.println("Berikut adalah nilai-nilai ujian yang Anda masukkan:
");
        for (int index = 0; index < ujian.length; index++)
        {
            System.out.print(ujian[index] + " ");
        }
    }
}
```

Output Program (TampilkanNilaiUjian.java)

```
Berapa banyak ujian yang ingin dimasukkan? 5
Masukkan nilai ujian 1: 72
Masukkan nilai ujian 2: 85
Masukkan nilai ujian 3: 81
Masukkan nilai ujian 4: 94
Masukkan nilai ujian 5: 99

Berikut adalah nilai-nilai ujian yang Anda masukkan:
72 85 81 94 99
```

Program di atas meminta pengguna untuk menentukan ukuran dari array. Pada baris 22, statement berikut membuat sebuah array dengan menggunakan variabel `bykUjian` (yang menyimpan input pengguna) untuk menentukan ukuran array tersebut:

```
ujian = new int[bykUjian];
```

Lalu program menggunakan dua buah loop `for`. Loop `for` pertama, pada baris 25 sampai dengan 30, digunakan untuk meminta pengguna memasukkan setiap nilai ujian. Loop `for` kedua, pada baris 35 sampai dengan 38, digunakan untuk menampilkan semua nilai ujian. Kedua loop `for` ini menggunakan `length` untuk mengatur banyaknya iterasi, seperti berikut:

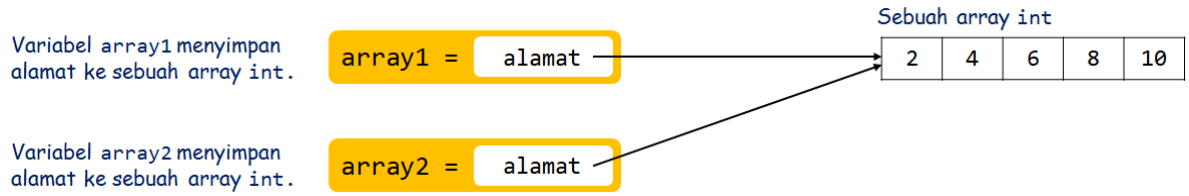
```
for (int index = 0; index < ujian)
```

Menyalin Array

Karena array adalah object, terdapat perbedaan antara array dan variabel yang mereferesikannya. Array dan variabel referensi adalah dua entitas berbeda. Ini penting untuk diingat ketika kita ingin menyalin isi dari satu array ke array lain. Mungkin yang terpikir untuk dilakukan pertama kali ketika kita ingin menyalin array adalah dengan menuliskan kode seperti berikut:

```
int[] array1 = { 2, 4, 6, 8, 10 };
int[] array2 = array1;    // Ini tidak menyalin array1
```

Statement pertama membuat sebuah array dan menugaskan alamatnya ke variabel `array1`. Statement kedua menugaskan `array1` ke `array2`. Statement kedua ini tidak membuat salinan dari array yang direferensikan oleh `array1`. Namun, statement tersebut menyalin alamat yang disimpan dalam variabel `array1`. Setelah statement kedua ini dieksekusi, kedua variabel `array1` dan `array2` akan mereferensikan array yang sama. Gambar berikut mengilustrasikan ini:



Program berikut mendemonstrasikan bahwa dua variabel dapat mereferensikan array yang sama:

Program (ArraySama.java)

```
/*
    Program ini mendemonstrasikan bahwa dua variabel
    dapat mereferensikan array yang sama.
*/
public class ArraySama
{
    public static void main(String[] args)
    {
        int[] array1 = { 2, 4, 6, 8, 10 };
        int[] array2 = array1;

        // Ubah salah satu elemen array menggunakan array1
        array1[0] = 200;

        // Ubah salah satu elemen array menggunakan array2
        array2[4] = 1000;

        // Tampilkan semua elemen menggunakan array1
        System.out.println("Isi dari array1: ");
        for (int nilai : array1)
        {
            System.out.print(nilai + " ");
        }
        System.out.println();

        // Tampilkan semua elemen menggunakan array2
        System.out.println("Isi dari array2: ");
        for (int nilai : array2)
        {
            System.out.print(nilai + " ");
        }
        System.out.println();
    }
}
```

Output Program (ArraySama.java)

```
Isi dari array1:
200 4 6 8 1000
Isi dari array2:
200 4 6 8 1000
```

Program di atas mengilustrasikan bahwa kita tidak dapat menyalin sebuah array hanya dengan menugaskan sebuah variabel referensi ke variabel referensi lainya. Untuk menyalin array, kita menggunakan sebuah loop, seperti berikut:

```
int[] arrayPertama = { 5, 10, 15, 20, 25};
int[] arrayKedua = new int[5];

for (int index = 0; index < arrayPertama.length; index++)
{
    arrayKedua[index] = arrayPertama[index];
}
```

Loop di atas menyalin setiap elemen dari `arrayPertama` ke elemen dengan indeks yang sama ke `arrayKedua`.

7.3 Menggunakan Array dengan Method

Memberikan Array sebagai Argument

Array dapat diberikan sebagai argument ke method. Sebagai contoh, method berikut menerima sebuah array `double` dan menghitung jumlah dari elemen-elemen dalam array tersebut:

```
public static double sum(double[] nilai)
{
    double jumlah = 0.0;

    for (double elemen : nilai)
    {
        jumlah = jumlah + elemen;
    }

    return jumlah;
}
```

Perhatikan pada header dari method `sum` di atas kita mendeklarasikan sebuah variabel parameter `nilai` dengan tipe data `double[]` yang berarti parameter `nilai` akan digunakan untuk mereferensikan sebuah array yang elemen-elemennya bertipe `double`.

Ketika kita memberikan sebuah array sebagai argument ke pemanggilan method, kita memberikan variabel yang mereferensikan array tersebut. Misalkan jika variabel `skor` mereferensikan sebuah array, pemanggilan method `sum` dituliskan seperti berikut:

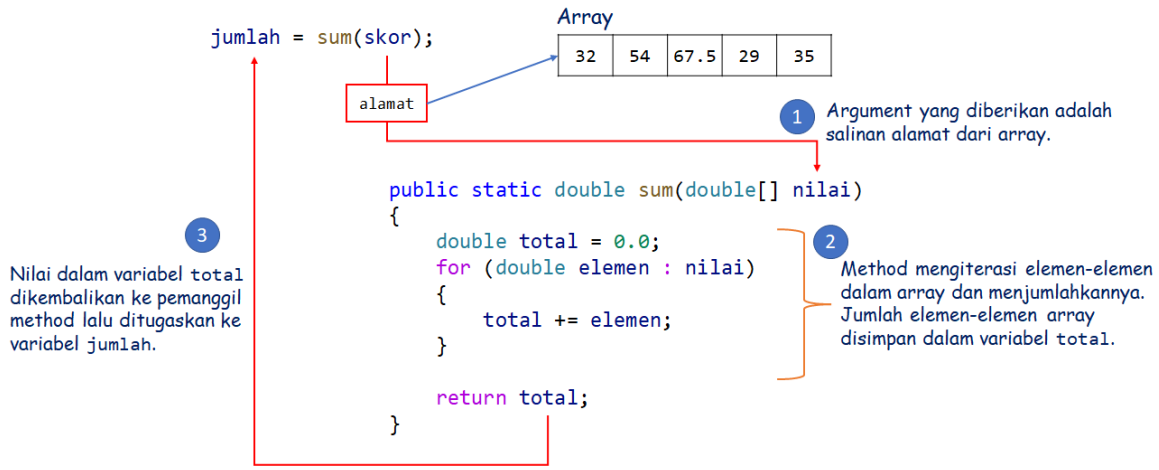
```
sum(skor)
```

Contoh penggunaan method `sum` di atas dapat dilihat pada kode berikut:

```
double[] skor = { 32, 54, 67.5, 29, 35 };
double jumlah;
jumlah = sum(skor);
```

Statement terakhir pada kode di atas, memanggil method `sum` dengan memberikan argument berupa array yang direferensikan oleh variabel `skor`. Setelah kode di atas selesai dieksekusi, variabel `jumlah` akan menyimpan nilai `217.5`.

Ketika sebuah array diberikan ke sebuah method, referensi (alamat memori) ke array tersebut yang disalin ke variabel parameter. Pada contoh pemanggilan method di atas, alamat memori array yang direferensikan oleh variabel `skor` disalin ke variabel parameter `nilai`. Gambar berikut mengilustrasikan pemanggilan method ini:



Program berikut mencontohkan penggunaan method `sum` di atas:

Program (PassArray.java)

```
import java.util.Scanner;

/*
    Program ini mendemostrasikan sebuah method yang menerima
    sebuah array sebagai argument.
*/
public class JumlahArray
{
    public static void main(String[] args)
    {
        double[] skor = new double[5];
        double jumlah;

        Scanner keyboard = new Scanner(System.in);

        for (int i = 0; i < 5; i++)
        {
            System.out.print("Masukkan angka ke-" + (i + 1) +": ");
            skor[i] = keyboard.nextDouble();
        }

        jumlah = sum(skor);
        System.out.println("Jumlah angka = " + jumlah);
    }

    /*
```

```
        Method ini menerima sebuah array tipe double
        dan mengembalikan jumlah dari elemen-elemen array
        yang diberikan
    */
    public static double sum(double[] nilai)
    {
        double total = 0.0;
        for (double elemen : nilai)
        {
            total += elemen;
        }

        return total;
    }
}
```

Output Program (PassArray.java)

```
Masukkan angka ke-1: 32
Masukkan angka ke-2: 54
Masukkan angka ke-3: 67.5
Masukkan angka ke-4: 29
Masukkan angka ke-5: 35
Jumlah angka = 217.5
```

Memodifikasi Elemen-elemen pada Array Argument

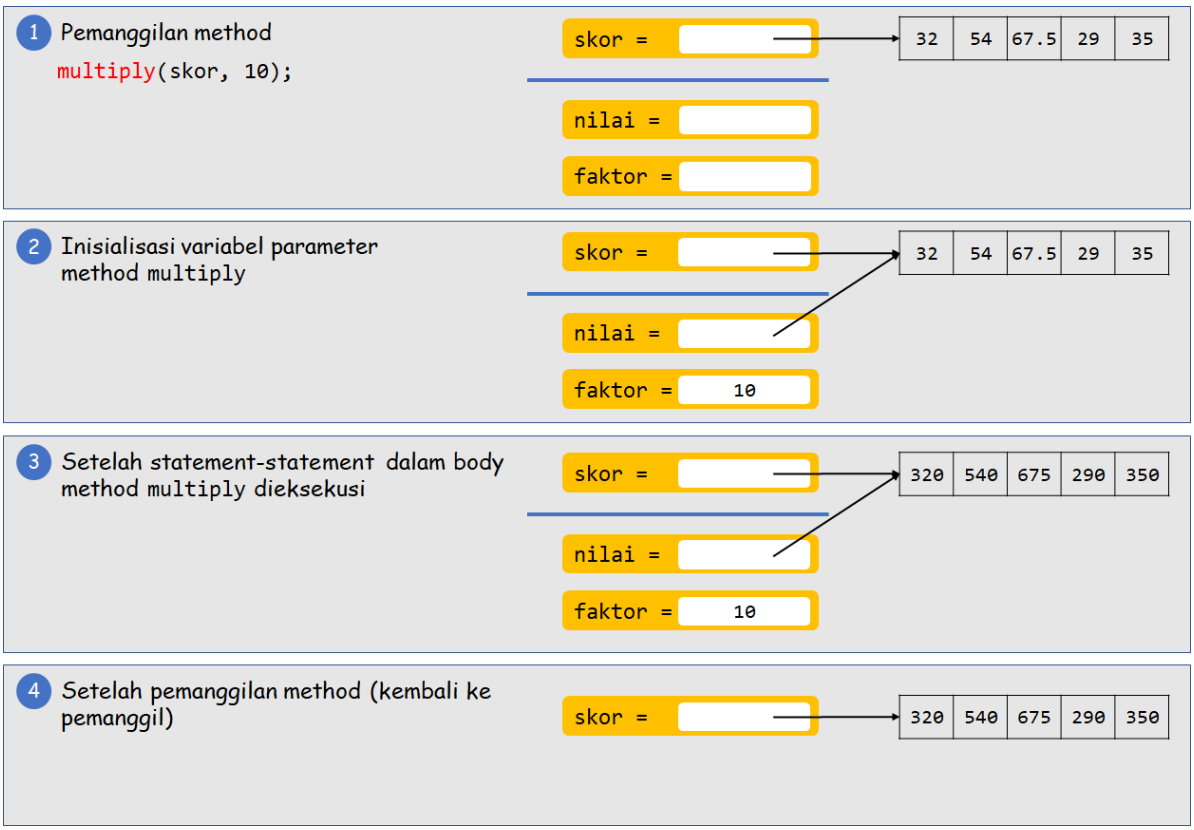
Karena ketika kita memberikan array sebagai argument ke method, alamat dari array tersebut yang diberikan, maka di dalam method kita mempunyai akses langsung ke array yang diberikan. Oleh karena ini, kita dapat mempunyai method yang memodifikasi array yang diberikan. Perhatikan definisi method berikut:

```
public static void multiply(double[] nilai, double faktor)
{
    for (int i = 0; i < nilai.length; i++)
    {
        nilai[i] = nilai[i] * faktor;
    }
}
```

Method `multiply` di atas menerima sebuah argument berupa array tipe `double` dan sebuah argument berupa nilai `double`. Method ini mengalikan semua elemen-elemen dalam array yang diberikan pada argument pertama dengan nilai yang diberikan pada argument kedua. Perhatikan pada header method kita menuliskan tipe return dengan `void`. Ini karena method `multiply` tidak mengembalikan array namun memodifikasi nilai-nilai elemen pada array yang diberikan. Gambar berikut mengilustrasikan nilai-nilai variabel saat pemanggilan method

```
multiply(skor, 10);
```

dimana `skor` adalah variabel referensi array bernilai `{32, 54, 67.5, 29, 35}`:



Penjelasan langkah-langkah pada gambar di atas:

1. Ketika method `multiply` dipanggil, variabel parameter `nilai` dan `faktor` dibuat.
2. Lalu, kedua variabel parameter diinisialisasi dengan argument-argument yang diberikan dalam pemanggilan method. Dalam contoh, variabel parameter `nilai` ditetapkan dengan alamat memori dari array yang direferensikan oleh `skor` dan variabel parameter `faktor` ditetapkan dengan nilai `10`. Perhatikan bahwa `nilai` dan `skor` mereferensikan array yang sama.
3. Statement-statement dalam body method `multiply`. Statement-statement ini mengalikan semua elemen array `nilai` dengan `10`. Sehingga setelah statement-statement dalam body method `multiply` selesai dieksekusi, array yang direferensikan oleh `skor` akan mempunyai nilai-nilai elemen yang telah berubah.
4. Setelah selesai mengeksekusi semua statement dalam method `multiply`, program kembali ke pemanggil method. Variabel parameter dihapus dari memori. Namun, variabel `skor` tetap mereferensikan array yang sama yang elemen-elemennya telah dimodifikasi.

Program berikut mendemonstrasikan penggunaan method `multiply` ini:

Program (ModifikasiArray.java)

```java
/*
    Program ini mendemonstrasikan method yang mengubah
    nilai-nilai elemen pada array yang diberikan sebagai argument.
*/
public class ModifikasiArray
{
    public static void main(String[] args)
    {
        double[] skor = { 32, 54, 67.5, 29, 35 };

        // Cetak nilai array sebelum modifikasi
        System.out.println("Array sebelum modifikasi: ");
        for (double elemen : skor)
        {
            System.out.print(elemen + " ");
        }
        System.out.println();

        // Modifikasi array skor dengan memanggil method multiply
        multiply(skor, 10);

        // Cetak array setelah modifikasi
        System.out.println("Array sesudah modifikasi: ");
        for (double elemen : skor)
        {
            System.out.print(elemen + " ");
        }
        System.out.println();
    }

    /*
        Method ini memodifikasi nilai-nilai elemen pada array yang diberikan
        sebagai argument pertama dengan mengalikan setiap nilai elemen dengan
        nilai yang diberikan sebagai argument kedua.
        Method ini tidak mengembalikan nilai.
    */
    public static void multiply(double[] nilai, double faktor)
    {
        for (int i = 0; i < nilai.length; i++)
        {
            nilai[i] = nilai[i] * faktor;
        }
    }
}
```

Output Program (ModifikasiArray.java)

```
Array sebelum modifikasi:
32.0 54.0 67.5 29.0 35.0
Array sesudah modifikasi:
320.0 540.0 675.0 290.0 350.0
```

Mengembalikan Array dari Method

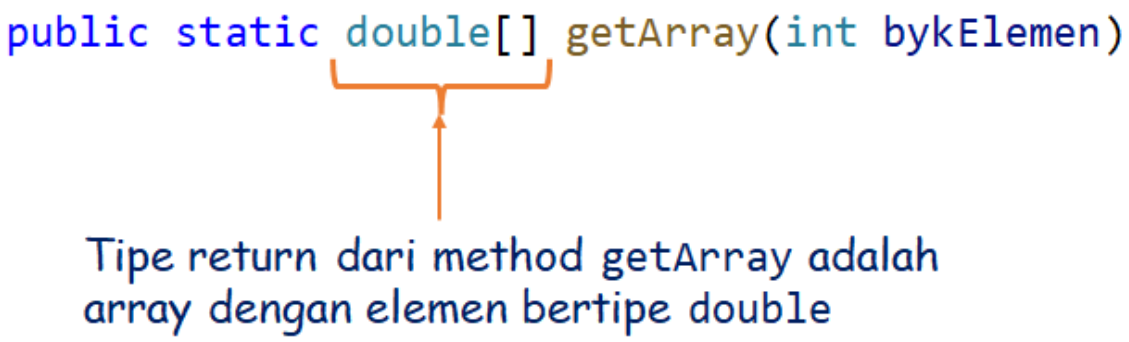
Method dapat mengembalikan sebuah array. Untuk melakukannya, tipe return dari method harus dideklarasikan sebagai tipe array. Sebagai contoh, perhatikan definisi method berikut:

```
public static double[] getArray(int bykElemen)
{
    double[] arrDouble = new double[bykElemen];

    for (int i = 0; i < bykElemen; i++)
    {
        arrDouble[i] = Math.random() * 100;
    }

    return arrDouble;
}
```

Method `getArray` di atas mengembalikan sebuah array `double`. Perhatikan tipe return pada method header. Kita menuliskan `double[]` sebagai tipe return. Ini menandakan bahwa method `getArray` mengembalikan sebuah array tipe `double`. Gambar berikut menjelaskan tipe return pada header dari method `getArray`:



Di dalam method `getArray`, sebuah array `double` dibuat, diinisialisasikan dengan sejumlah nilai-nilai, dan direferensikan dengan sebuah variabel `arrDouble`. Lalu, statement `return` mengembalikan variabel `arrDouble`. Nilai return dari method ini dapat disimpan di dalam variabel referensi yang cocok. Program berikut mendemonstrasikan penggunaan method yang mengembalikan array:

Program (ReturnArray.java)

```
/*
    Program ini mendemonstrasikan sebuah method yang mengembalikan
    referensi ke sebuah array.
*/
public class ReturnArray
{
    public static void main(String[] args)
    {
        double[] nilai;

        nilai = getArray(10);

        // Cetak array yang didapatkan dari pemanggilan
        // method getArray
        for (double elemen : nilai)
```

```
        {
            System.out.println(elemen);
        }
    }

    /*
        Method ini menggenerate angka random sebanyak nilai argument
        yang diberikan.
        Method ini mengembalikan sebuah array dengan nilai-nilai yang
        digenerate.
    */
    public static double[] getArray(int bykElemen)
    {
        double[] arrDouble = new double[bykElemen];

        for (int i = 0; i < bykElemen; i++)
        {
            // Isi elemen dengan angka random double antara 0 s.d 100
            arrDouble[i] = Math.random() * 100;
        }

        return arrDouble;
    }
}
```

Output Program (ReturnArray.java)

```
34.64134562616028
17.290490320170793
6.554726907339181
24.732679424753144
17.649710429216604
64.09052770734654
72.06801327302065
20.68065907252199
16.081507136131258
77.81296932530938
```

Statement pada baris 11:

```
nilai = getArray(10);
```

menugaskan array yang dikembalikan oleh method `getArray` ke variabel array `nilai`. Lalu loop `for` enhanced pada baris 15 sampai dengan 18 menampilkan nilai dari setiap elemen dalam array `nilai`.

7.4 Algoritma-algoritma Array yang Umum

Membandingkan Array

Pada bagian sebelumnya kita telah melihat bahwa kita tidak dapat menyalin sebuah array dengan hanya menugaskan variabel referensinya ke variabel referensi array lain. Lebih lanjut, kita tidak dapat menggunakan operator `==` untuk membandingkan dua variabel referensi array dan menentukan apakah kedua array adalah sama. Sebagai contoh, kode berikut terlihat membandingkan dua array, namun sebenarnya tidak:

```
int[] arrayPertama = { 5, 10, 15, 20, 25 };
int[] arrayKedua = { 5, 10, 15, 20, 25 };
if (arrayPertama == arrayKedua)    // Ini membandingkan array yang salah.
{
    System.out.println("Kedua array sama.");
}
else
{
    System.out.println("Kedua array tidak sama.");
}
```

Kode di atas akan memberikan output:

```
Kedua array tidak sama.
```

Ini terlihat aneh karena kita mengharapkan kode di atas memberikan output "Kedua array sama." karena `arrayPertama` dan `arrayKedua` mempunyai elemen-elemen yang sama persis. Namun yang sebenarnya terjadi, ketika kita menggunakan operator `==` dengan variabel referensi, termasuk variabel referensi array, operator tersebut membandingkan alamat memori yang disimpan variabel tersebut bukan isi dari object yang direferensikan oleh variabel tersebut. Karena `arrayPertama` dan `arrayKedua` mereferensikan array-array yang berbeda tempat dalam memori, mereka akan memiliki alamat memori yang berbeda. Oleh karena ini, ekspresi `boolean`, `arrayPertama == arrayKedua` akan menghasilkan `false` dan kode di atas akan melaporkan bahwa kedua array tidak sama.

Untuk membandingkan isi dari kedua array, kita harus membandingkan elemen-elemen dalam kedua array tersebut. Misalkan, perhatikan kode berikut:

```
int[] arrayPertama = { 2, 4, 6, 8, 10 };
int[] arrayKedua = { 2, 4, 6, 8, 10 };
boolean arraySama = true;        // Variabel flag
int index = 0;                   // Variabel pencacah loop

// Pertama cek apakah kedua array berukuran sama.
// Jika kedua ukuran array tidak sama, maka sudah pasti
// kedua array tidak sama.
if (arrayPertama.length != arrayKedua.length)
{
    arraySama = false;
}

// Kemudian cek apakah elemen-elemen dari kedua array menyimpan data yang sama.
while (arraySama && index < arrayPertama.length)
{
```

```
    if (arrayPertama[index] != arrayKedua[index])
    {
        arraySama = false;
    }
    index++;
}

if (arraySama)
{
    System.out.println("Kedua array sama.");
}
else
{
    System.out.println("Kedua array tidak sama.");
}
```

Kode di atas membandingkan apakah `arrayPertama` dan `arrayKedua` memiliki isi yang sama persis. Variabel `boolean`, `arraySama`, yang diinisialisasi dengan `true`, digunakan untuk menandakan apakah kedua array sama. Variabel lain, `index`, yang diinisialisasi ke 0, digunakan sebagai variabel kontrol loop.

Pertama-tama kode ini mencaritahu apakah kedua array memiliki panjang yang sama. Jika kedua array tidak mempunyai panjang sama, maka kedua array tersebut pasti tidak sama, sehingga variabel `arraySama` ditetapkan ke `false`. Lalu, loop `while` dimulai. Loop `while` ini dieksekusi selama `arraySama` bernilai `true` dan variabel kontrol loop, `index`, kurang dari `arrayPertama.length`. Setiap iterasinya, loop ini membandingkan nilai elemen dengan indeks `index` dari `arrayPertama` dan `arrayKedua`. Jika, kedua nilainya tidak sama, maka variabel `arraySama` ditetapkan ke `false`. Setelah loop selesai, sebuah statement `if` menguji nilai variabel `arraySama`. Jika nilai variabel tersebut `true`, maka kedua array adalah sama dan sebuah pesan yang menginformasikan ini ditampilkan. Sebaliknya, jika variabel `arraySama` bernilai `false`, maka kedua array tidaklah sama, dan sebuah pesan yang menginformasikan kedua array tidak sama ditampilkan.

Menjumlahan Nilai-nilai dalam Array Numerik

Untuk menjumlahkan nilai-nilai dalam sebuah array kita harus menggunakan sebuah loop dengan variabel akumulator. Loop menambahkan nilai dalam setiap elemen array ke akumulator. Sebagai contoh, misalkan statement berikut berada dalam sebuah program dan nilai-nilai telah disimpan dalam array `units`:

```
int[] units = new int[25];
```

Loop berikut menambahkan nilai-nilai setiap elemen dari array `units` ke variabel `total`. Ketika kode selesai, `total` akan berisi jumlah dari semua nilai elemen-elemen array `unit`.

```
int total = 0;        // Inisialisasi akumulator
for (int elemen : nilai)
{
    total += elemen;
}
```

Mendapatkan Rata-rata dari Nilai-nilai dalam Array Numerik

Langkah pertama untuk menghitung rata-rata dari semua nilai elemen dalam sebuah array adalah untuk menjumlahkan semua nilai elemen tersebut. Langkah kedua adalah membagi jumlah dari semua nilai elemen tersebut dengan banyaknya elemen dalam array. Misalkan, statement berikut terdapat dalam sebuah program dan nilai-nilali telah disimpan dalam array `skor`:

```
double[] skor = new double[10];
```

Kode berikut mengkalkulasi rata-rata dari semua nilai dalam array `skor`. Ketika kode ini selesai, nilai rata-rata akan disimpan dalam variabel `average`.

```
double total = 0;         // Inisialisasi akumulator
double average;           // Untuk menyimpan rata-rata
for (int elemen : nilai)
{
    total += elemen;
}
average = total / skor.length;
```

Perhatikan statement terakhir, yang membagi `total` dengan `skor.length`, tidak berada di dalam loop. Statement ini hanya dieksekusi sekali, setelah loop selesai dengan semua iterasinya.

Mencari Nilai Tertinggi dan Nilai Terendah dalam Array Numerik

Algoritma untuk mencari nilai tertinggi dan nilai terendah dalam sebuah array mirip dengan cara kita mencari nilai tertinggi atau nilai terendah dari input pengguna dalam loop (seperti yang telah kita bahas saat pembahasan algoritma loop yang umum). Kita memerlukan sebuah loop yang mengiterasi setiap elemen dari array dimulai dari elemen pertama hingga elemen terakhir. Untuk mencari nilai tertinggi, kita harus mempunyai sebuah variabel yang menjaga nilai tertinggi setiap iterasinya. Sedangkan untuk mencari nilai terendah, kita harus mempunyai sebuah variabel yang menjaga nilai terendah setiap iterasinya.

Misalkan statement berikut berada dalam program dan nilai-nilai telah disimpan dalam array `angka`:

```
int[] angka = new int[50];
```

Untuk mencari nilai tertinggi dalam array `angka` kita dapat menuliskan kode seperti berikut:

```
int tertinggi = angka[0];       // Inisialisasi variabel yang menjaga nilai
tertinggi
                                // dengan nilai elemen pertama dalam array
for (int index = 1; index < angka.length; index++)
{
    if (angka[index] > tertinggi)
    {
        tertinggi = angka[index];
    }
}
```

Kode di atas bekerja dengan pertama-tama menyalin nilai dari elemen pertama dari array `angka` ke variabel `tertinggi`. Lalu, sebuah loop `for` membandingkan semua elemen-elemen array berikutnya, dimulai dari index 1, dengan nilai pada variabel `tertinggi`. Setiap kali nilai elemen dalam array lebih besar dari nilai pada variabel `tertinggi`, nilai tersebut disalin ke variabel `tertinggi`. Ketika loop selesai, variabel `tertinggi` akan menyimpan nilai tertinggi dalam array.

Kode berikut mencari nilai terendah dalam array:

```
int terendah = angka[0];    // Inisialisasi variabel yang menjaga nilai terendah
                            // dengan nilai elemen pertama dalam array
for (int index = 1; index < angka.length; index++)
{
    if (angka[index] < terendah)
    {
        terendah = angka[index];
    }
}
```

Seperti dapat Anda lihat, kode untuk mencari terendah ini mirip dengan kode untuk mencari nilai tertinggi. Ketika loop selesai, variabel `terendah` akan berisi nilai terendah dalam array `angka`.

Pencarian Linear

Dalam membuat program yang bekerja dengan array, seringkali kita perlu melakukan pencarian posisi dari elemen spesifik dalam sebuah array sehingga kita dapat memodifikasi nilai elemen tersebut atau menghapusnya. Berikut adalah contoh kode yang mencari posisi dari elemen yang pertama kali ditemukan yang bernilai sama dengan 100 dalam sebuah array yang direferensikan oleh variabel `nilai`:

```
int nilaiDicari = 100;
int posisi = 0;
boolean ditemukan = false;
while (posisi < nilai.length && !ditemukan)
{
    if (nilai[posisi] == nilaiDicari)
    {
        ditemukan = true;
    }
    else
    {
        posisi++;
    }
}
if (ditemukan)
{
    System.out.println("Ditemukan pada posisi: " + posisi);
}
else
{
    System.out.println("Tidak ditemukan.");
}
```

Algoritma di atas disebut sebagai **pencarian linear** karena kita memeriksa setiap elemen dalam array secara berurutan dari elemen pertama hingga terakhir.

Array Terisi Sebagian

Terkadang kita perlu menyimpan rangkaian data dalam sebuah array, tetapi kita tidak mengetahui berapa banyak data yang akan disimpan. Dalam kondisi ini, kita harus menentukan banyak maksimum dari data yang dapat disimpan dan membuat array dengan panjang sebesar banyak maksimum data tersebut. Namun, terdapat hal lain yang perlu kita perhatikan, jika banyaknya data yang disimpan ternyata kurang dari banyaknya elemen array, maka array tersebut akan terisi sebagian. Ketika kita memproses array yang terisi sebagian, kita harus memproses hanya elemen-elemen yang berisi data yang valid.

Array terisi sebagian umumnya digunakan bersama dengan sebuah variabel integer yang menyimpan banyaknya data yang disimpan dalam array tersebut. Sebagai contoh, misalkan sebuah program menggunakan kode berikut untuk membuat sebuah array dengan 100 elemen, dan sebuah variabel `int` bernama `count`, yang digunakan untuk menyimpan banyaknya data yang disimpan dalam array:

```
final int MAKSIMUM = 100;
int[] array = new int[MAKSIMUM];
int count = 0;
```

Setiap kali kita menambahkan data ke dalam array, kita harus menginkrementasikan `count`. Kode berikut mendemonstrasikan ini:

```
int angka;
Scanner keyboard = new Scanner(System.in);
System.out.print("Masukkan sebuah angka atau -1 untuk keluar: ");
angka = keyboard.nextInt();
while (angka != -1 && count < array.length)
{
    array[count] = angka;
    count++;
    System.out.print("Masukkan sebuah angka atau -1 untuk keluar: ");
    angka = keyboard.nextInt();
}
```

Setiap iterasi dari loop sentinel ini memungkinkan pengguna untuk memasukkan sebuah angka yang disimpan dalam array `nilai`, atau memasukkan -1 untuk keluar. Variabel `count` digunakan sebagai indeks dari elemen array berikutnya yang tersedia, dan lalu diinkrementasi. Ketika pengguna memberikan input -1, atau ketika `count` mencapai ukuran dari array, loop berhenti. Kode berikut dapat digunakan untuk menampilkan nilai-nilai data yang valid di dalam array `nilai`:

```
for (int index = 0; index < count; index++)
{
    System.out.println(array[index]);
}
```

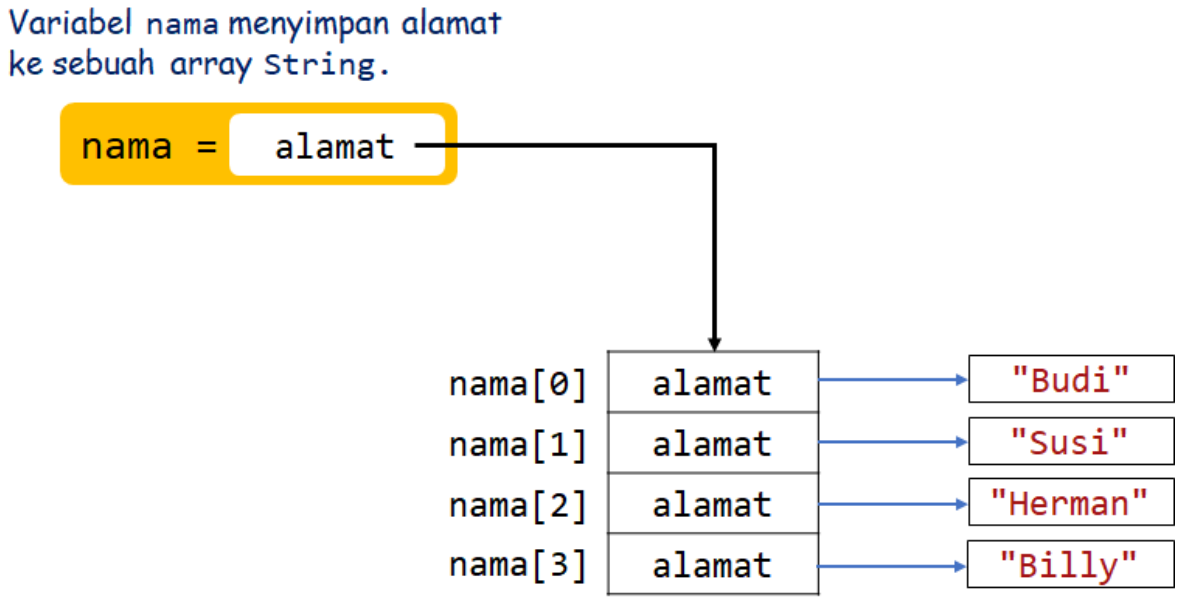
Perhatikan pada loop `for` di atas kita menggunakan variabel `count` sebagai batas iterasi loop.

7.5 Array String

Kita dapat membuat array yang elemen-elemennya berupa string. Berikut adalah contoh statement yang membuat sebuah array dari `String` yang diinisialisasi dengan nilai-nilai:

```
String[] nama = { "Budi", "Susi", "Herman", "Billy" };
```

Gambar berikut mengilustrasikan array yang dibuat dari statement di atas:



Pada array `String`, setiap elemen dari array tersebut adalah sebuah referensi ke object `String`. Elemen `nama[0]` mereferensikan sebuah object `String` yang berisi "Budi", elemen `nama[1]` mereferensikan sebuah object `String` yang berisi "Susi", dan selanjutnya. Program berikut mendemonstrasikan penggunaan sebuah array dari object `String`:

Program (BulanTahun.java)

```
/*
    Program ini mendemonstrasikan array dengan elemen-elemen
    berupa object String.
*/
public class BulanTahun
{
    public static void main(String[] args)
    {
        String[] bulan = { "Januari", "Februari", "Maret",
                           "April", "Mei", "Juni",
                           "Juli", "Agustus", "September",
                           "Oktober", "November", "Desember" };

        int[] hari = { 31, 28, 31, 30, 31, 30, 31,
                       31, 30, 31, 30, 31 };

        for (int index = 0; index < bulan.length; index++)
        {
            System.out.println(bulan[index] + " mempunyai " +
                               hari[index] + " hari.");
        }
    }
```

```
}
```

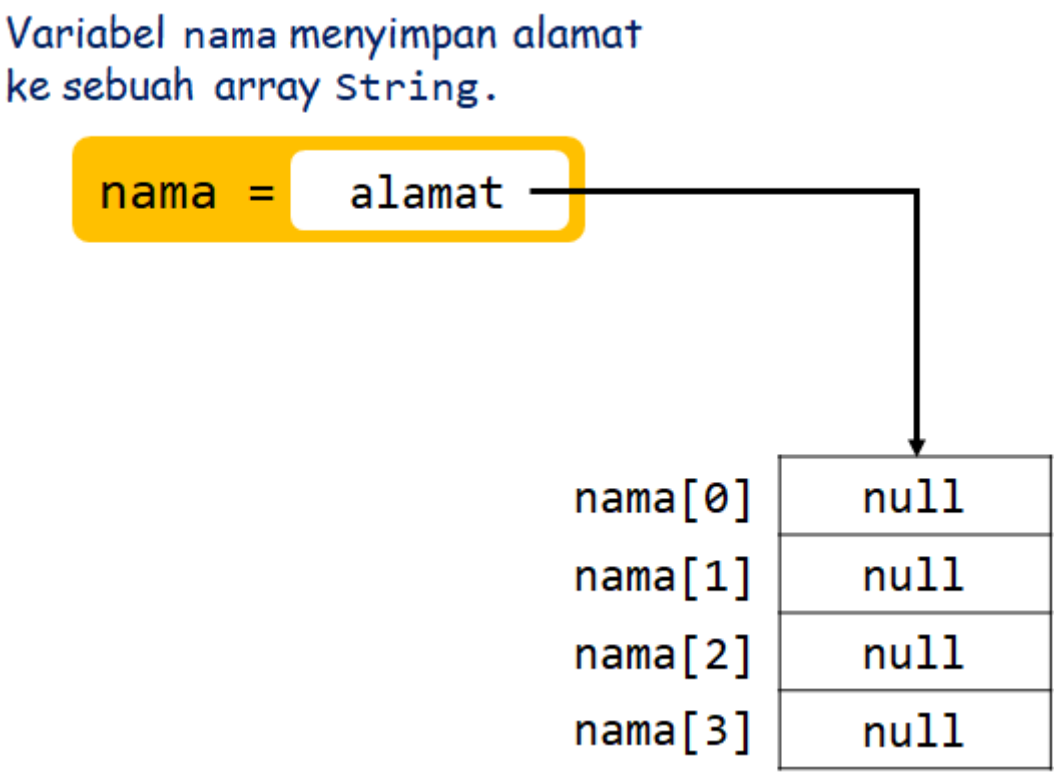
Output Program (BulanTahun.java)

```
Januari mempunyai 31 hari.
Februari mempunyai 28 hari.
Maret mempunyai 31 hari.
April mempunyai 30 hari.
Mei mempunyai 31 hari.
Juni mempunyai 30 hari.
Juli mempunyai 31 hari.
Agustus mempunyai 31 hari.
September mempunyai 30 hari.
Oktober mempunyai 31 hari.
November mempunyai 30 hari.
Desember mempunyai 31 hari.
```

Seperti pada array bertipe data primitif, inisialisasi dengan nilai-nilai membuat sebuah array dari object `String` dibuat dalam memori. Jika kita tidak menuliskan inisialisasi dengan nilai-nilai, kita harus menggunakan keyword `new` untuk membuat array, seperti berikut:

```
final int UKURAN = 4;
String[] nama = new String[UKURAN];
```

Statement kedua dalam kode di atas membuat sebuah array dari empat referensi ke object `String`, seperti yang dapat dilihat pada gambar berikut:



Perhatikan pada gambar di atas bahwa array yang terbentuk adalah array dari empat referensi `String` yang tidak terinisialisasi yang ditandakan nilai elemen-elemen dari array tersebut sama dengan `null`.

Ketika kita membuat sebuah array dari object `String` yang tidak terinisialisasi, kita harus menugaskan sebuah nilai ke setiap elemen dalam array. Berikut adalah contohnya:

```
final int UKURAN = 4;
String[] nama = new String[UKURAN];
nama[0] = "Budi";
nama[1] = "Susi";
nama[2] = "Herman";
nama[3] = "Billy";
```

Setelah statement-statement di atas dieksekusi, setiap elemen dari array `nama` akan mereferensikan sebuah object `String`.

Memanggil Method `String` dari Elemen Array

Pada topik 3 kita melah mempelajari bahwa object `String` memiliki sejumlah method. Sebagai contoh, method `charAt` digunakan untuk mendapatkan karakter pada indeks tertentu dari object `String`. Karena setiap elemen dari array `String` adalah sebuah object `String`, kita dapat menggunakan elemen dari array tersebut dengan method-method dari object `String`. Sebagai contoh, statement berikut menggunakan elemen 0 dari array `nama` untuk memanggil method `charAt()`:

```
System.out.println(nama[0].charAt(0));
```

Statement di atas akan mencetak karakter pertama dari `String` yang direferensikan oleh `nama[0]`.

Program berikut mendemonstrasikan pemanggilan method `String` pada elemen-elemen dari array `String`:

Program (PanjangNama.java)

```
import java.util.Scanner;

/* Program ini mendemonstrasikan pemanggilan method String
    pada elemen-elemen dari array String.
*/
public class PanjangNama
{
    public static void main(String[] args)
    {
        String[] nama = new String[4];

        Scanner keyboard = new Scanner(System.in);

        // Minta nama 1 s.d 4
        for (int i = 0; i < nama.length; i++)
        {
            System.out.print("Masukkan nama ke-" + (i + 1) + ": ");
            nama[i] = keyboard.nextLine();
        }

        // Cari panjang masing-masing nama
        for (int i = 0; i < nama.length; i++)
        {
            System.out.println("Panjang nama ke-" + (i + 1) + " " +
                               nama[i] + " adalah " + nama[i].length());
        }
```

```
    }
}
```

Output Program (PanjangNama.java)

```
Masukkan nama ke-1: Budi
Masukkan nama ke-2: Susi
Masukkan nama ke-3: Herman
Masukkan nama ke-4: Billy
Panjang nama ke-1 Budi adalah 4
Panjang nama ke-2 Susi adalah 4
Panjang nama ke-3 Herman adalah 6
Panjang nama ke-4 Billy adalah 5
```

Perhatikan pada loop `for` terakhir pada program di atas:

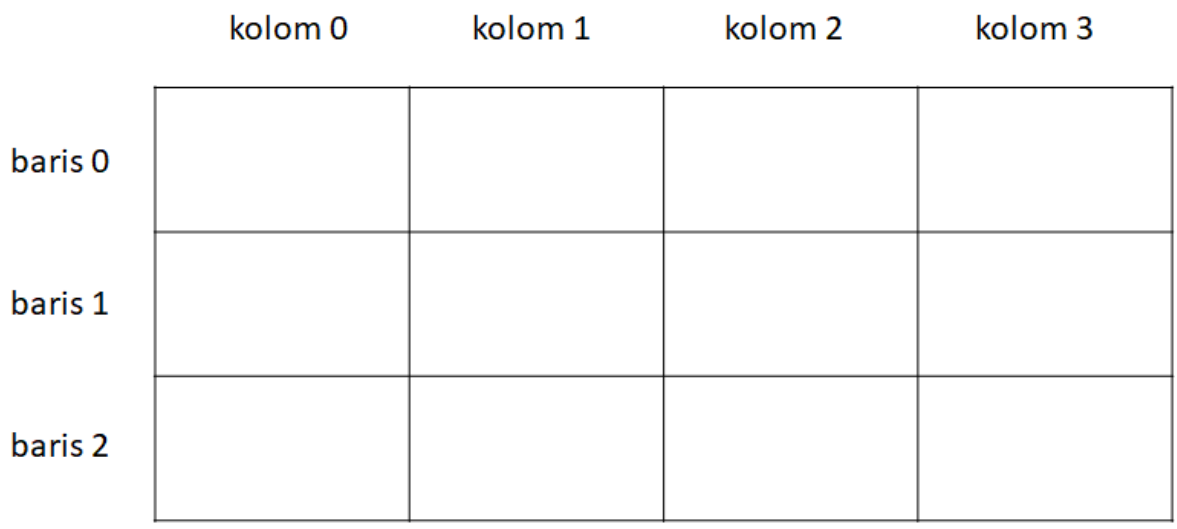
```
for (int i = 0; i < nama.length; i++)
{
    System.out.println("Panjang nama ke-" + (i + 1) + " " +
                       nama[i] + " adalah " + nama[i].length());
}
```

Pada header dari loop `for` kita menggunakan `nama.length` untuk mendapatkan panjang array `nama` sebagai batas atas dari iterasi. Lalu pada body dari loop, kita menggunakan `nama[i].length()` untuk mendapatkan panjang `String` yang direferensikan oleh elemen `nama[i]`.

7.6 Array Multi Dimensi

Array Dua Dimensi

Array yang sejauh ini telah kita pelajari adalah array satu-dimensi. Disebut array satu-dimensi karena array tersebut hanya menyimpan satu himpunan data. Kita dapat mempunyai array dengan lebih dari satu dimensi. Misalkan, kita dapat mempunyai array dua dimensi yang menyimpan lebih dari satu himpunan data. Array dua dimensi adalah array yang elemennya berupa array juga. Array dua dimensi dapat dibayangkan sebagai sebuah tabel dengan baris dan kolom seperti berikut:

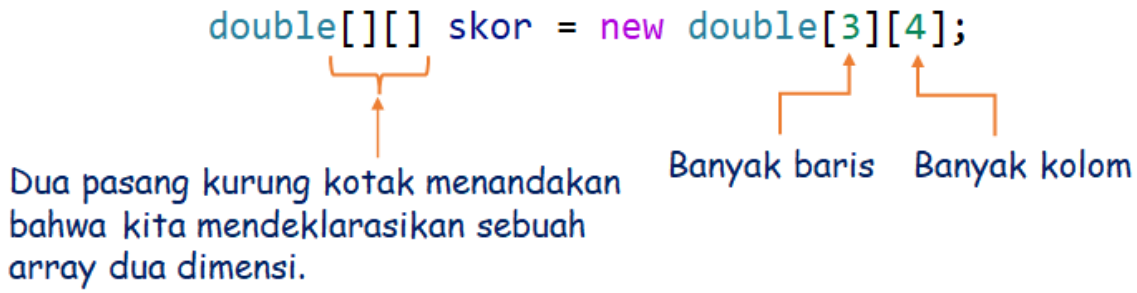


Array dua dimensi pada gambar di atas mempunyai tiga baris (dinomori 0 sampai dengan 2) dan empat kolom (dinomori 0 sampai dengan 3). Terdapat total 12 elemen dalam array tersebut.

Untuk mendeklarasikan sebuah array dua dimensi, dua pasang kurung kotak dan dua pendeklarasi ukuran array dibutuhkan: satu untuk jumlah baris dan lainnya untuk jumlah kolom. Sebagai contoh, statement berikut adalah contoh deklarasi dari array dua dimensi dengan tiga baris dan empat kolom:

```
double[][] skor = new double[3][4];
```

Dua pasang tanda kurung setelah tipe data `double` menandakan bahwa variabel `skor` akan mereferensikan array dua dimensi. Angka 3 dan 4 adalah pendeklarasi ukuran. Angka dalam kurung kotak pertama menyatakan banyak baris dan angka dalam kurung kotak kedua menandakan banyak kolom. Ini diilustrasikan pada gambar berikut:



Elemen-elemen dalam array dua dimensi mempunyai dua indeks: indeks pertama untuk barisnya dan indeks kedua untuk kolomnya. Pada array `skor`, elemen-elemen pada baris 0 direferensikan seperti berikut:

```
skor[0][0]
skor[0][1]
skor[0][2]
skor[0][3]
```

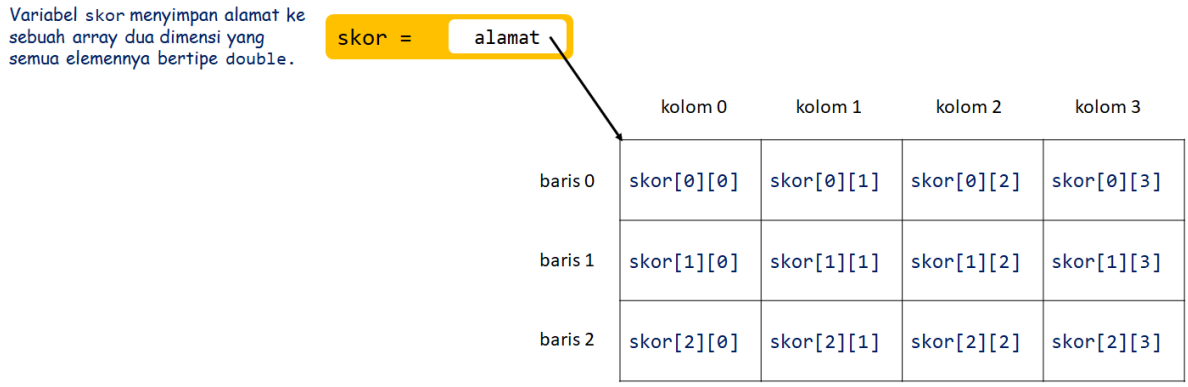
Elemen-elemen dalam baris 1 mempunyai indeks-indeks seperti berikut:

```
skor[1][0]
skor[1][1]
skor[1][2]
skor[1][3]
```

Dan elemen-elemen dalam baris 2 mempunyai indeks-indeks seperti berikut:

```
skor[2][0]
skor[2][1]
skor[2][2]
skor[2][3]
```

Gambar berikut mengilustrasikan indeks-indeks untuk setiap elemen dalam array dua dimensi `skor`:



Untuk mengakses data dalam array dua dimensi, kita harus menuliskan kedua indeks dalam dua tanda kurung kotak. Sebagai contoh, statement berikut menyimpan nilai 95 ke `skor[2][1]`:

```
skor[2][1] = 95;
```

Untuk mengakses semua elemen-elemen dalam array dua dimensi, kita menggunakan loop tersarang. Sebagai contoh, kode berikut menampilkan prompt ke pengguna untuk memasukkan sebuah skor untuk setiap elemen dalam array:

```
final int BARIS = 3;
final int KOLOM = 4;
double[][] skor = new double[BARIS][KOLOM];
for (int baris = 0; baris < BARIS; baris++)
{
    for (int kolom = 0; kolom < KOLOM; kolom++)
    {
        System.out.print("Masukkan skor: ");
        skor[baris][kolom] = keyboard.nextDouble();
    }
}
```

Kode berikut menampilkan semua elemen dalam array `skor`:

```
for (int baris = 0; baris < BARIS; baris++)
{
    for (int kolom = 0; kolom < KOLOM; kolom++)
    {
        System.out.println(skor[baris][kolom]);
    }
}
```

Menginisialisasi Array Dua Dimensi

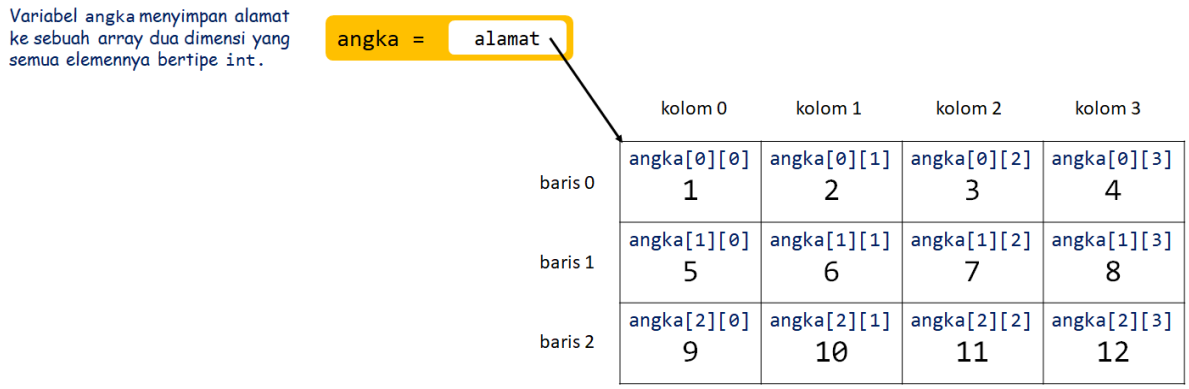
Ketika kita menginisialisasi array dua dimensi, kita menuliskan nilai-nilai inisialisasi dari setiap baris dalam kurung kurawal sendiri. Berikut adalah contoh inisialisasi array dua dimensi:

```
int[][] angka = { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} };
```

Seperti pada inisialisasi array satu dimensi, kita tidak menggunakan keyword `new` ketika kita membuat array dengan insialisasi nilai-nilai. Java akan secara otomatis membuat array dan mengisi elemen-elemennya dengan nilai-nilai inisialisasi. Pada statement di atas, nilai inisialisasi untuk baris 0 adalah {1, 2, 3, 4}, nilai inisialisasi untuk baris 1 adalah {5, 6, 7, 8}, dan nilai inisialisasi untuk baris 2 adalah {9, 10, 11, 12}. Kita juga dapat menuliskan inisialiasi di atas seperti berikut untuk memudahkan dalam membacanya:

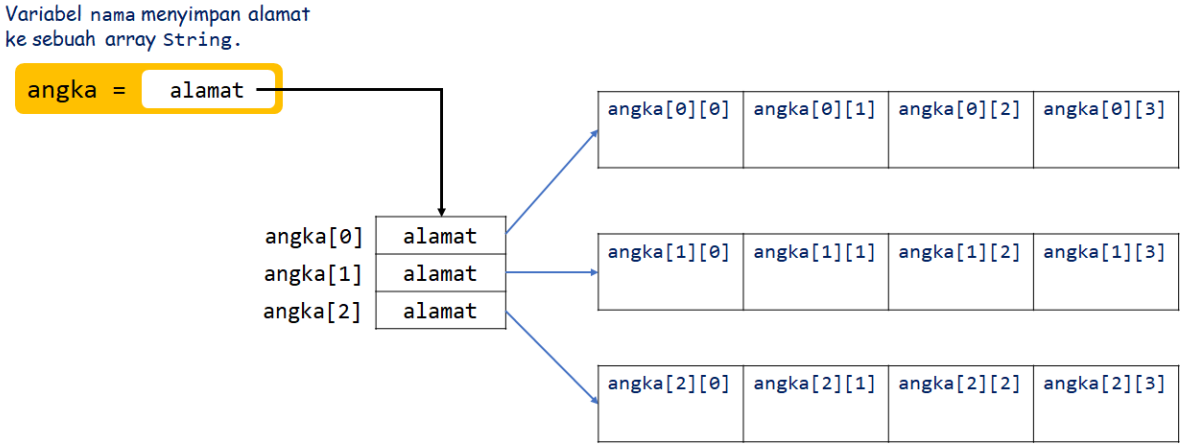
```
int[][] angka = { {1, 2, 3, 4},
                  {5, 6, 7, 8},
                  {9, 10, 11, 12} };
```

Gambar berikut mengilustrasikan nilai-nilai yang ditugaskan ke array `angka` setelah inisialisasi di atas:



Panjang dari Array Dua Dimensi

Seperti yang telah disebutkan di awal, array dua dimensi adalah array yang masing-masing elemennya berupa array juga. Selain menggambarkan array dua dimensi sebagai tabel, kita juga dapat menggambarkan array dua dimensi seperti gambar berikut:



Pada gambar di atas, dapat dilihat, variabel `angka` mereferensikan sebuah array satu dimensi dengan empat elemen yang dapat diakses dengan `angka[0]`, `angka[1]`, dan `angka[2]`. Masing-masing elemen dari array tersebut juga mereferensikan array satu dimensi dengan empat elemen. Elemen `angka[0]` mereferensikan sebuah array satu dimensi dengan empat elemen yang masing-masing elemennya dapat diakses dengan `angka[0][0]`, `angka[0][1]`, `angka[0][2]`, dan `angka[0][3]`. Pola yang sama juga berlaku untuk elemen `angka[1]` dan `angka[2]`.

Sehingga, pada array dua dimensi dengan tiga baris dan empat kolom, kita akan mempunyai total empat array satu dimensi. Setiap array ini mempunyai field `length` sendiri. Kita dapat menggunakan field `length` ini untuk mendapatkan banyak baris dan banyak kolom dari array dua dimensi. Misalkan, untuk mendapatkan banyak baris dari array dua dimensi yang

direferensikan oleh variabel `angka` kita dapat menuliskan ekspresi seperti berikut:

```
angka.length
```

Ekspresi di atas akan mengembalikan banyaknya elemen dari array satu dimensi yang menyimpan referensi-referensi ke array satu dimensi

Sedangkan untuk mendapatkan banyak kolom dari array dua dimensi yang direferensikan oleh variabel `angka`, kita dapat mengakses field `length` dari salah satu array yang berisi elemen-elemen kolom dari suatu baris. Sebagai contoh, kita dapat mengakses field `length` dari array yang berisi elemen-elemen kolom pada baris 0 untuk mendapatkan banyak kolom dari array dua dimensi:

```
angka[0].length
```

Program berikut mendemonstrasikan penggunaan field `length` untuk mendapatkan banyak baris dan kolom dari array dua dimensi:

Program (Panjang.java)

```
/*
    Program ini menggunakan field length dari array 2D
    untuk mendapatkan banyak baris dan banyak kolom dari
    setiap baris.
*/
public class Panjang
{
    public static void main(String[] args)
    {
        // Inisialisasi array 2D dengan 3 baris
        // dan 4 kolom.
        int[][] angka = { { 1, 2, 3, 4},
                          { 5, 6, 7, 8},
                          { 9, 10, 11, 12} };

        // Tampilkan banyak baris
        System.out.println("Banyak baris adalah " + angka.length);

        // Tampilkan banyak kolom pada setiap baris
        for (int index = 0; index < angka.length; index++)
        {
            System.out.println("Banyak kolom dari baris " + index +
                               " adalah " + angka[index].length);
        }
    }
}
```

Output Program (Panjang.java)

```
Banyak baris adalah 3
Banyak kolom dari baris 0 adalah 4
Banyak kolom dari baris 1 adalah 4
Banyak kolom dari baris 2 adalah 4
```

Menampilkan Semua Elemen dalam Array Dua Dimensi

Seperti yang telah kita lihat pada contoh program sebelumnya, sebuah loop tersarang dapat digunakan untuk menampilkan semua elemen-elmeen dari array dua dimensi. Sebagai contoh, kode berikut membuat array `angka` dengan tiga baris dan empat kolom, dan lalu menampilkan semua elemen-elemen dalam array:

```
int[][] angka = { { 1, 2, 3, 4 },
                  { 5, 6, 7, 8 },
                  { 9, 10, 11, 12} };

for (int baris = 0; baris < 3; baris++)
{
    for (int kolom = 0; kolom < 4; kolom++)
    {
        System.out.println(angka[baris][kolom]);
    }
}
```

Kode di atas hanya bisa digunakan pada array dengan tiga baris dan empat kolom. Pendekatan yang lebih baik adalah menggunakan `length` dari array sebagai batas atas indeks pada ekspresi penguji batas. Berikut adalah loop yang dimodifikasi yang dapat digunakan oleh berbagai ukuran array dua dimensi:

```
for (int baris = 0; baris < angka.length; baris++)
{
    for (int kolom = 0; kolom < angka[baris].length; kolom++)
    {
        System.out.println(angka[baris][kolom]);
    }
}
```

Perhatikan pada kode di atas. Pada loop luar kita menuliskan header loop seperti berikut:

```
for (int baris = 0; baris < angka.length; baris++)
```

Loop ini mengendalikan indeks untuk baris dari array `angka`. Karena `angka.length` menyimpan banyaknya baris dalam array, kita menggunakannya sebagai batas atas dari indeks baris.

Pada loop dalam, kita menuliskan header loop seperti berikut:

```
for (int kolom = 0; kolom < angka[baris].length; kolom++)
```

Loop ini mengendalikan indeks untuk kolom dari array `angka`. Karena setiap field `length` dari array yang menyimpan elemen-elemen kolom pada suatu baris akan berisi banyaknya kolom pada baris tersebut, kita menggunakannya sebagai batas atas dari indeks kolom.

Dengan menggunakan field `length` sebagai batas atas pada loop tersarang yang memproses array dua dimensi, kita dapat mempunyai kode yang dapat bekerja pada array dua dimensi dengan banyak baris dan banyak kolom berapapun.

Menjumlahkan Semua Elemen dari Array Dua Dimensi

Untuk menjumlahkan elemen-elemen dalam array dua dimensi, kita dapat menggunakan loop tersarang yang menjumlahkan setiap elemen ke sebuah akumulator. Kode berikut mencontohkannya:

```
int[][] angka = { { 1,  2,  3,  4 },
                  { 5,  6,  7,  8 },
                  { 9, 10, 11, 12} };

int total = 0;        // Akumulator, diinisialisasi ke 0
// Jumlahkan elemen-elemen array
for (int baris = 0; baris < angka.length; baris++)
{
    for (int kolom = 0; kolom < angka[baris].length; kolom++)
    {
        total += angka[baris][kolom];
    }
}

// Tampilkan jumlah elemen-elemen
System.out.println("Total elemen adalah " + total);
```

Menjumlahkan Baris-baris dari Array Dua Dimensi

Terkadang kita ingin menghitung jumlah dari setiap baris dalam array dua dimensi. Sebagai contoh, misalkan dalam sebuah program yang mengelola nilai-nilai ujian mahasiswa dalam sebuah kelas, kita menggunakan sebuah array dua dimensi untuk menyimpan tiga nilai ujian dari setiap mahasiswa. Program tersebut menghitung rata-rata tiga nilai ujian dari setiap mahasiswa. Untuk menghitung rata-rata dari setiap mahasiswa, kita perlu menjumlahkan nilai-nilai dari setiap baris terlebih dahulu. Kode berikut mencontohkannya:

```
double[][] nilaiUjian = { { 76.5 , 96.4, 89.25 },
                          { 69.5 , 91.5, 87.4  },
                          { 67.4 , 65.8, 82.34 },
                          { 98.5 , 99.5, 100   },
                          { 89.56, 92.5, 97.56 } };

double total;        // akumulator
for (int baris = 0; baris < nilaiUjian.length; baris++)
{
    // Tetapkan akumulator ke 0
    total = 0.0;

    // Jumlahkan setiap baris
    for (int kolom = 0; kolom < nilaiUjian[baris].length; kolom++)
    {
        total += nilaiUjian[baris][kolom];
    }

    // Tampilkan total dari baris
    System.out.println("Total dari baris " + baris +
                       " adalah " + total);
}
```

Perhatikan bahwa variabel `total`, yang digunakan sebagai akumulator, ditetapkan ke 0 sebelum loop dalam dieksekusi. Ini karena loop dalam menjumlahkan elemen-elemen kolom dari suatu baris dan menyimpannya jumlahnya ke `total`. Sehingga, variabel `total` harus ditetapkan ke 0 sebelum setiap iterasi dari loop dalam.

Menjumlahkan Kolom-kolom dari Array Dua Dimensi

Terkadang kita juga ingin menghitung jumlah dari setiap kolom dari array dua dimensi. Sebagai contoh, pada sebuah program yang mengelola nilai-nilai ujian mahasiswa dalam sebuah kelas, kita menggunakan sebuah array dua dimensi untuk menyimpan tiga nilai ujian dari setiap mahasiswa. Kita ingin menghitung rata-rata dari setiap ujian. Untuk melakukannya, kita harus menjumlahkan semua elemen-elemen dalam setiap kolom. Kode berikut mencontohkannya:

```
double[][] nilaiUjian = { { 76.5 , 96.4, 89.25 },
                          { 69.5 , 91.5, 87.4  },
                          { 67.4 , 65.8, 82.34 },
                          { 98.5 , 99.5, 100   },
                          { 89.56, 92.5, 97.56 } };

double total;         // akumulator
for (int kolom = 0; kolom < nilaiUjian[0].length; kolom++)
{
    // Tetapkan akumulator ke 0.
    total = 0;

    // Jumlahkan kolom.
    for (int baris = 0; baris < nilaiUjian.length; baris++)
    {
        total += nilaiUjian[baris][kolom];
    }

    // Tampilkan total kolom
    System.out.println("Total kolom " + kolom +
                       " adalah " + total);
}
```

Memberikan Array Dua Dimensi ke Method

Untuk membuat sebuah method menerima array dua dimensi sebagai argument, kita harus mendeklarasikan parameternya sebagai referensi ke array dua dimensi. Berikut adalah contoh dari header method yang menerima array dua dimensi sebagai argument:

```
public static void cetakTabel(int[][] array)
```

Perhatikan pada header method `cetakTabel` di atas. Variabel parameter dari method, yang bernama `array`, dideklarasikan sebagai referensi ke array dua dimensi bertipe `int`.

Program berikut mendemonstrasikan penggunaan method-method yang menerima array dua dimensi:

Program (OlahNilai.java)

```
import java.util.Scanner;

/*
    Program ini mendemonstrasikan penggunaan method-method
    untuk mengolah data pada array dua dimensi.
*/
public class OlahNilai
{
    public static void main(String[] args)
    {
        double[][] nilaiUjian;
        int bykMahasiswa;
        int bykUjian;

        Scanner keyboard = new Scanner(System.in);

        // Prompt pengguna untuk memasukkan banyak mahasiswa dalam
        // satu kelas.
        System.out.print("Masukkan banyak mahasiswa: ");
        bykMahasiswa = keyboard.nextInt();

        // Prompt pengguna untuk memasukkan banyak ujian
        System.out.print("Masukkan banyak ujian: ");
        bykUjian = keyboard.nextInt();

        // Buat array 2-D dengan banyak baris ebanyak banyak mahasiswa
        // dan banyak kolom sebanyak banyak ujian
        nilaiUjian = new double[bykMahasiswa][bykUjian];

           // Prompt pengguna untuk memasukkan nilai-nilai ujian untuk setiap
mahasiswa
        for (int baris = 0; baris < nilaiUjian.length; baris++)
        {
            System.out.println("Masukkan nilai ujian untuk mahasiswa ke-" +
                               (baris + 1) + ".");
            for (int kolom = 0; kolom < nilaiUjian[baris].length; kolom++)
            {
                System.out.print("Masukkan nilai ujian ke-" +
                                 (kolom + 1) + ": ");
                nilaiUjian[baris][kolom] = keyboard.nextDouble();
            }
        }

        // Cetak rata-rata mahasiswa
        System.out.println();
        cetakRataRataMahasiswa(nilaiUjian);

        // Cetak rata-rata masing-masing ujian
        System.out.println();
        cetakRataRataUjian(nilaiUjian);
    }

    /*
        Method ini mencetak nilai rata-rata mahasiswa
        dengan menghitung rata-rata dari nilai baris dari argument array
```

```
        dua dimensi yang diberikan.
        Method ini menerima sebuah array 2D.
        Method ini tidak mengembalikan nilai.
    */
    public static void cetakRataRataMahasiswa(double[][] ujian)
    {
        double total;   // Akumulator setiap baris

        System.out.println("-----------------------------------");
        System.out.println("     Rata-rata nilai mahasiswa    ");
        System.out.println("-----------------------------------");
        for (int baris = 0; baris < ujian.length; baris++)
        {
            total = 0.0; // Tetapkan akumulator 0 setiap mulai iterasi kolom
            for (int kolom = 0; kolom < ujian[baris].length; kolom++)
            {
                total += ujian[baris][kolom];
            }
            // Tampilkan rata-rata per mahasiswa
            System.out.printf("Rata-rata mahasiswa ke-%d = %.2f\n", (baris + 1),
                              (total / ujian[baris].length));
        }
    }

    /*
        Method ini mencetak nilai rata-rata dari ujian
        dengan menghitung rata-rata dari nilai kolom dari argument array
        dua dimensi yang diberikan.
        Method ini menerima sebuah array 2D.
        Method ini tidak mengembalikan nilai.
    */
    public static void cetakRataRataUjian(double[][] ujian)
    {
        double total;   // Akumulator setiap kolom
        System.out.println("-----------------------------------");
        System.out.println("     Rata-rata nilai ujian        ");
        System.out.println("-----------------------------------");
        for (int kolom = 0; kolom < ujian[0].length; kolom++)
        {
            total = 0.0;
            for (int baris = 0; baris < ujian.length; baris++)
            {
                total += ujian[baris][kolom];
            }
            // Tampilkan rata-rata setiap ujian
            System.out.printf("Rata-rata ujian ke-%d = %.2f\n", (kolom + 1),
                              (total / ujian.length));
        }
    }


}
```

Output Program (OlahNilai.java)

```
Masukkan banyak mahasiswa: 5
Masukkan banyak ujian: 3
Masukkan nilai ujian untuk mahasiswa ke-1.
Masukkan nilai ujian ke-1: 98.5
Masukkan nilai ujian ke-2: 87.6
Masukkan nilai ujian ke-3: 76.5
Masukkan nilai ujian untuk mahasiswa ke-2.
Masukkan nilai ujian ke-1: 89.3
Masukkan nilai ujian ke-2: 87.5
Masukkan nilai ujian ke-3: 95.6
Masukkan nilai ujian untuk mahasiswa ke-3.
Masukkan nilai ujian ke-1: 89.4
Masukkan nilai ujian ke-2: 97.5
Masukkan nilai ujian ke-3: 75.0
Masukkan nilai ujian untuk mahasiswa ke-4.
Masukkan nilai ujian ke-1: 68.5
Masukkan nilai ujian ke-2: 98.5
Masukkan nilai ujian ke-3: 100
Masukkan nilai ujian untuk mahasiswa ke-5.
Masukkan nilai ujian ke-1: 74.5
Masukkan nilai ujian ke-2: 95.5
Masukkan nilai ujian ke-3: 97.3

---------------------------------
     Rata-rata nilai mahasiswa
---------------------------------
Rata-rata mahasiswa ke-1 = 87.53
Rata-rata mahasiswa ke-2 = 90.80
Rata-rata mahasiswa ke-3 = 87.30
Rata-rata mahasiswa ke-4 = 89.00
Rata-rata mahasiswa ke-5 = 89.10

---------------------------------
     Rata-rata nilai ujian
---------------------------------
Rata-rata ujian ke-1 = 84.04
Rata-rata ujian ke-2 = 93.32
Rata-rata ujian ke-3 = 88.88
```

Array dengan Dimensi Tiga atau Lebih

Java tidak membatasi banyaknya dimensi yang dapat dimiliki oleh array. Kita dapat membuat sebuah array dengan dimensi berapapun. Misalkan, contoh berikut adalah deklarasi dari array tiga dimensi:

```
double[][][] kursi = new double[3][3][4];
```

Array tiga dimensi di atas dapat dibayangkan sebagai tiga tabel dengan jumlah baris 3 dan jumlah kolom 4. Salah satu cara untuk menggambarkan sebuah array tiga dimensi adalah dengan menganggapnya sebagai halaman-halaman dari array dua dimensi. Misalkan array tiga dimensi `kursi` dapat diilustrasikan seperti gambar berikut:

Array dengan dimensi lebih dari tiga sulit untuk divisualisasikan, tetapi dapat berguna pada beberapa persoalan pemrograman. Sebagai contoh, dalam sebuah gudang pabrik terdapat beberapa boks-boks berisi barang inventaris yang ditumpuk dalam palet-palet kayu pada rak-rak, sebuah array empat dimensi dapat digunakan untuk menyimpan nomor inventaris dari barang inventaris tersebut. Empat indeks dari array empat dimensi dapat digunakan untuk merpresentasikan berikut: indeks dimensi pertama untuk nomor boks, indeks dimensi kedua untuk nomor baris rak, indeks dimensi ketiga untuk nomor kolom rak, dan indeks dimensi keempat untuk nomor palet.

7.7 Command Line Argument

Ketika kita menjalankan sebuah program Java dari command line, kita dapat menuliskan argument-argument yang diberikan ke method `main` dalam program tersebut.

Setiap program yang telah kita lihat dan tulis menggunakan method `main` static dengan header seperti berikut:

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

Di dalam tanda kurung pada header di atas, adalah deklarasi dari sebuah parameter bernama `args` yang mereferensikan sebuah array `String`. Ini berarti method `main` dapat menerima sebuah argument berupa array `String`. Array `String` yang diberikan ke parameter `args` pada method `main` ini berasal dari command line ketika kita menjalankan program Java dari method `main` tersebut. Sebagai contoh, perhatikan program berikut:

Program (CommandLine.java)

```
/*
    Program ini menampilkan argument-argument
    yang diberikan dari command line.
*/
public class CommandLine
{
    public static void main(String[] args)
    {
        for (int index = 0; index < args.length; index++)
        {
            System.out.println(args[index]);
        }
    }
}
```

Jika program di atas dikompilasi, lalu dijalankan dengan perintah berikut:

```
java CommandLine Bagaimana ini bekerja?
```

maka program tersebut akan menghasilkan output seperti berikut:

```
Bagaimana
ini
bekerja?
```

Teks-teks yang kita tulis pada command line, yang dipisahkan oleh spasi, dan setelah nama class dianggap sebagai satu atau lebih argument yang diberikan ke method `main`. Dalam contoh di atas, tiga argument diberikan ke parameter `args`. Teks "Bagaimana" diberikan ke `args[0]`, "ini" diberikan ke `args[1]`, dan "bekerja?" diberikan ke `args[2]`. Loop `for` pada program di atas digunakan untuk menampilkan elemen-elemen dari parameter `args`.

Method-method untuk Konversi `String` ke Tipe Primitif Numerik

Karena kita menuliskan parameter dari fungsi `main` sebagai array dari `String`, setiap argument-argument yang dituliskan pada command line akan selalu diartikan sebagai sebuah `String` meskipun kita menuliskan argument command line dengan angka-angka. Java menyediakan method-method yang dapat digunakan untuk mengkonversi `String` menjadi tipe numerik sehingga kita dapat melakukan operasi aritmatika terhadap argument-argument command line. Tabel berikut mendaftar method-method untuk mengkoversi `String` ke angka:

Method	Gunakan Method ini untuk	Contoh Kode
`Byte.parseByte`	Konversi string ke tipe `byte`	`byte num = Byte.parseByte(str)`
`Double.parseDouble`	Konversi string ke tipe `double`	`double num = Double.parseDouble(str)`
`Float.parseFloat`	Konversi string ke tipe `float`	`float num = Float.parseFloat(str)`
`Integer.parseInt`	Konversi string ke tipe `int`	`int num = Integer.parseInt(str)`
`Long.parseLong`	Konversi string ke tipe `long`	`long num = Long.parseLong(str)`
`Short.parseShort`	Konversi string ke tipe `short`	`short num = Short.parseShort(str)`

Program berikut mencontohkan penggunaaan method `Integer.parseInt` untuk mengkonversi argument command line menjadi tipe `int`:

Program (Tambah.java)

```
public class Tambah
{
    public static void main(String[] args)
    {
        int a, b;
        a = Integer.parseInt(args[0]);
        b = Integer.parseInt(args[1]);

        System.out.println(a + b);
    }
}
```

Program di atas mengkonversi argument indeks 0 dan indeks 1 dari parameter `args` ke `int` lalu menugaskan ke variabel `a` dan variabel `b` (sesuai urutannya). Lalu menampilkan hasil penjumlahan keduanya. Perintah command line berikut mencontohkan cara menjalankan program `Tambah.java` beserta output dari program:

```
C:\> java Tambah 45 98
143
```

REFERENSI

[1] Horstmann, Cay S. 2012. *Big Java: Late Objects, 1st Edition*. United States of America: John Wiley & Sons, Inc.

[2] Gaddis, Tony. 2016. *Starting Out with Java: From Control Structures through Objects (6th Edition)*. Boston: Pearson.