

MODUL 3. VARIABLE, OPERATOR, KONSTANTA

3.1 Tujuan

3.2 Alat dan Bahan

3.3 Dasar Teori

3.3.1 Algoritma dan Variabel

Algoritma—apakah algoritma itu? Mudahnya, algoritma adalah sebuah rangkaian langkah-langkah yang tersusun, yang digunakan untuk memecahkan masalah atau melakukan proses tertentu. Algoritma bisa diimplementasikan melalui kode yang kita tulis. Jadi, saat kita memprogram, kita sebenarnya hanya melakukan implementasi dari algoritma yang kita pikirkan. Karena itulah, nama mata kuliah ini adalah implementasi algoritma!

Sekarang, mari kita masuk ke variabel. Apa sih variabel itu? Dalam pemrograman, variabel ini bisa kita ibaratkan sebagai wadah. Sebuah wadah bisa menampung sesuatu, yang mana sesuatu tersebut bisa berupa apa saja. Namun, tentu saja saat kita menaruh sesuatu ke dalam wadah, kita akan menaruh “sesuatu” tersebut ke dalam wadah yang sesuai, dengan tipe dari “sesuatu” tersebut. Kalau ingin menaruh buku, tentu kita akan menaruhnya di rak buku, bukan? Kalau ingin menaruh baju, kita akan menaruhnya di lemari baju. Hal yang sama juga berlaku pada variabel! Singkatnya, variabel adalah sebuah tempat penyimpanan data, dimana variabel berlaku sebagai wadah, dan data berlaku sebagai “sesuatu” yang kita simpan. Ingat, variabel bersifat sementara. Variabel akan tetap menyimpan data selama dia berada dalam sebuah “blok” atau tempat ia dibuat, atau selama program berjalan.

Sekarang, kita masuk ke teknisnya, yuk! Asik nih, mulai ngoding.

```
public class ProgramDira {  
    public static void main(String[] args) {  
        int angka = 10;  
    }  
}
```

Di sini, kalian bisa melihat sebuah potongan kode java yang tentu sudah tidak asing. Di tengah-tengahnya, ada potongan kode baru. Benar, “int angka = 10” yang kalian lihat ini merupakan sebuah variabel. Kalian bisa melihat bahwa sebuah variabel terdiri dari 4 bagian. Apa sajakah itu?

Bagian pertama, yaitu tanda “int”. Bagian ini menandakan tipe data, atau *data type*. Seperti yang udah dijelaskan, kita kalau ingin masukin sesuatu ke wadah, tentu harus ke wadah yang sesuai dengan tipe “sesuatu” tersebut, bukan? Nah, di sini, int ini berarti *integer* atau dalam Bahasa Indonesia, bilangan bulat. Berarti, dalam potongan kode ini, kita ingin memasukkan bilangan bulat ke dalam variabel ini. Tipe data nggak terbatas hanya bilangan bulat saja, namun masih ada banyak lagi!

Bagian kedua, yaitu “angka”. Ini disebut juga dengan nama variabel, atau *identifier*. Nama variabel ini bebas kalian tulis dengan apapun. Tapi ingat, ya! Variabel bagusya memiliki nama yang berarti, seperti misalnya jika kalian ingin menyimpan umur kalian dalam variabel, maka variabel tersebut bisa kalian namai “umur”. Jadinya terlihat mudah diingat, bukan?

Bagian ketiga, yaitu tanda “=”. Dalam pemrograman, tanda “=” atau sama dengan ini tidak digunakan seperti pada matematika. Tanda sama dengan ini merupakan sebuah *operator*, yang nantinya akan kalian pelajari di bab selanjutnya. Tanda sama dengan ini disebut juga dengan operator penugasan atau *assignment operator*, yang digunakan untuk menugaskan variabel untuk menyimpan sebuah nilai. Jadi, dalam potongan kode di atas, kita bisa menginterpretasikannya seperti ini: “Hey, variabel angka yang bertipe *integer*. Kamu harus menyimpan nilai 10 ya!”

Bagian keempat, yaitu angka “10”. Nah, angka “10” ini disebut juga dengan nilai atau *value*. Nilai ini adalah data yang disimpan oleh variabel. Dengan begini, nilai 10 akan diberikan kepada variabel angka. Mudah bukan?

Namun, perlu diingat, ada yang bernama *reserved keywords*, atau kata kunci yang sudah disediakan oleh Java, yang tidak bisa kita dijadikan sebagai nama variabel. Mereka ada di bawah ini.

abstract	double	int	strictfp
boolean	else	interface	super
break	extends	long	switch
byte	final	native	this
case	finally	new	throw
catch	float	package	throws
char	for	private	transient
class	goto	protected	try
const	if	public	void
continue	implements	return	volatile
default	import	short	while
do	instanceof	static	synchronized

Terakhir, variabel tidak harus selalu kita *assign* nilai secara langsung setiap kali kita membuatnya. Kita bisa hanya mendefinisikan saja, dan kemudian melakukan *assign* setelahnya.

```

public class ProgramDira {
    public static void main(String[] args) {
        int a = 10;
        int b;

        b = 5;
    }
}

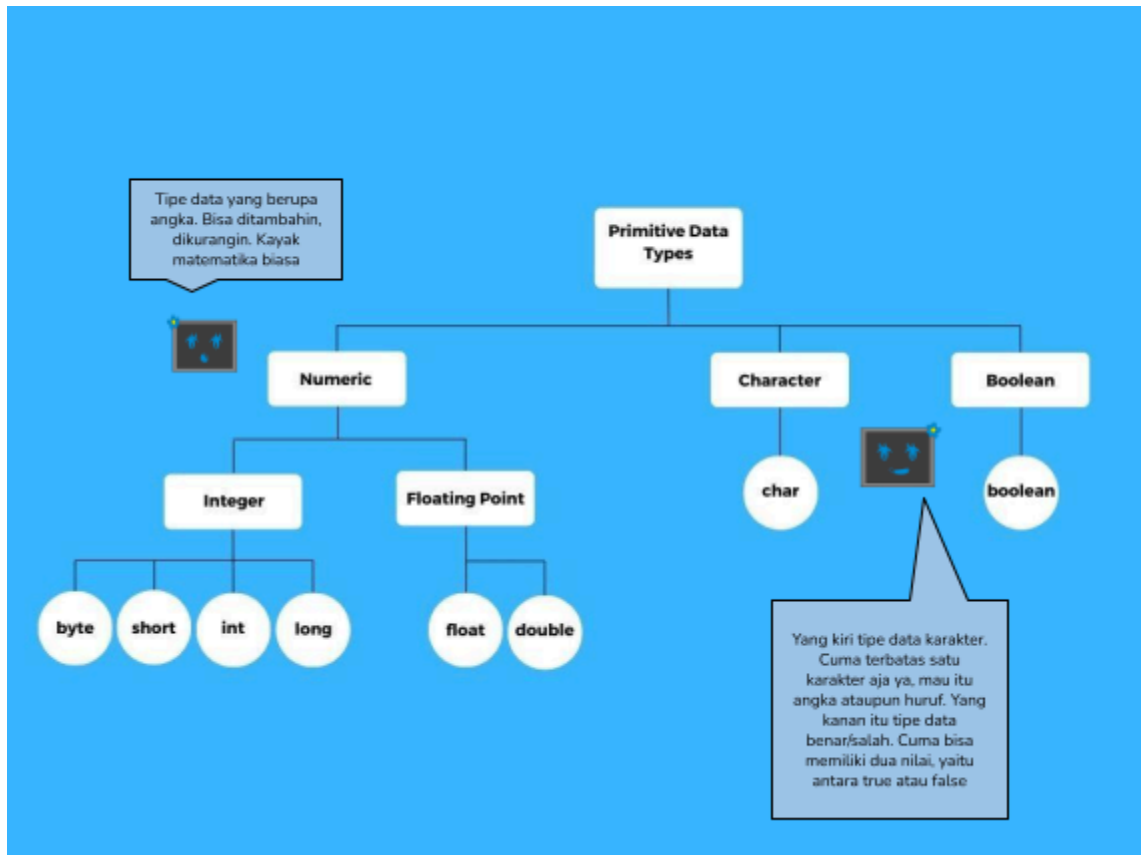
```

Di atas, kita melakukan *assignment* secara langsung pada a. Namun, kita tidak melakukan hal yang sama pada b. Yang kita lakukan pada b sebelum melakukan *assignment* disebut juga dengan deklarasi. Kita baru melakukan *assignment* setelah kita melakukan deklarasi. Hal seperti ini juga diperbolehkan, lho!

Penamaan variabel juga sebaiknya menggunakan **camelCase** atau cara penamaan dimana kita menggunakan huruf kecil semua, namun khusus saat ada kata baru, kita baru menggunakan huruf kapital. Contohnya adalah **variabelSaya** dan **iniVariabelMilikSaya**.

3.3.1.1 Tipe data Primitif

Sekarang kita akan melihat tipe data-tipe data lain yang tersedia di Java. Kita akan mulai dari tipe data primitif. Loh, apa sih yang dimaksud dengan tipe data primitif? Apa tipe datanya tinggal di zaman batu? Hahaha, tentu saja tidak. Tipe data primitif ini bisa kita bilang sebagai tipe data yang mendasari semuanya, atau tipe data yang paling mendasar. Ibaratnya, tipe data primitif ini seperti batu bata rumah, yang nanti bisa kita buat untuk membangun dinding dari gabungan-gabungan batu bata. Apa saja tipe data primitif? Mari kita lihat!



Kalian akan bertemu dengan tipe-tipe data di atas dengan jauh lebih sering nantinya! Dan cara memakainya mirip, kok dengan kode yang ada di atas. Beda di nilai yang akan dimasukkan saja! Untuk lebih detailnya, di bawah ini ada kode yang memakai semua jenis tipe data primitif.

```

public class ProgramDira {
    public static void main(String[] args) {
        // Tipe data byte
        // Menyimpan angka dari -128 sampai 127
        byte iniByte = 100;

        // Tipe data short
        // Menyimpan angka dari -32768 sampai 32767
        short iniShort = 1000;

        // Tipe data int
        // Menyimpan angka dari -2147483648 sampai 2147483647
        int iniInt = 1000000;

        // Tipe data long
        // Menyimpan angka dari -9223372036854775808 sampai 9223372036854775807
        long iniLong = 10000000000L;

        // Tipe data float
        // Menyimpan angka desimal dengan 6-7 digit di belakang koma
        float iniFloat = 10.10f;

        // Tipe data double
        // Menyimpan angka desimal dengan 15 digit di belakang koma
        double iniDouble = 10.5135135135;

        // Tipe data char
        // Menyimpan karakter
        char iniChar = 'A';

        // Tipe data boolean
        // Menyimpan nilai benar atau salah
        boolean booleanBenar = true;
        boolean booleanSalah = false;
    }
}

```

3.3.1.2 Tipe data objek

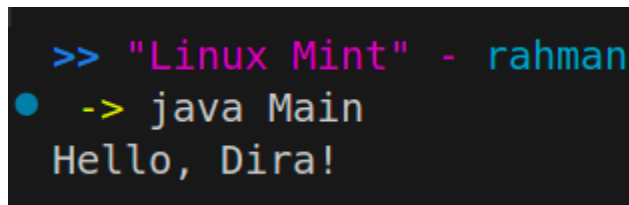
Kita sudah mengenal tipe data primitif, yaitu tipe data dasar. Namun, ada lagi nih tipe data yang bukan primitif, yaitu tipe data objek. Kalian sudah mempelajari tentang objek sebelumnya, kan? Nah, sebenarnya tipe-tipe data di atas memiliki tipe data objek yang mewakili diri mereka, loh! Tapi itu akan dibahas di bab-bab selanjutnya. Di sini, kita akan membahas tipe data objek yang paling sering digunakan: String.

3.3.1.2.1 String

Apa itu String? String adalah tipe data yang bisa kita gunakan untuk menyimpan banyak karakter. Singkat kata, yang kita simpan bisa berupa huruf, angka, kata, kalimat, hingga paragraf, loh! Jadi, misal kalian ingin menyimpan nama kalian dalam sebuah variabel, kalian bisa menggunakan String ini.

```
public class Main {  
    public static void main(String[] args) {  
        String name = "Dira";  
  
        System.out.println("Hello, " + name + "!");  
    }  
}
```

Dan saat program ini dijalankan, ia akan memunculkan output berikut:



```
>> "Linux Mint" - rahman  
● -> java Main  
Hello, Dira!
```

Lalu mengapa String ini disebut dengan tipe data objek? Nah, itu karena String ini sebenarnya merupakan sebuah kelas yang memiliki atribut yang bisa menyimpan banyak karakter. Atribut tersebut memiliki tipe data “array”, yang akan kita bahas di kemudian bab. Array yang berada dalam kelas String tersebut akan menyimpan setiap karakter yang ada dalam nilai yang kita berikan, yang dalam kasus ini adalah namaku, “Dira”. Keren, kan? String ini memiliki banyak *method* yang bisa digunakan untuk berbagai macam hal. Namun karena terlalu banyak, jadi tidak mungkin dibahas satu per satu di sini. Di bab-bab selanjutnya, kita akan memakai beberapa method milik String. Namun bila kalian ingin melihat method-method tersebut dan mempelajarinya sendiri, link berikut memiliki informasi yang dibutuhkan: https://www.w3schools.com/java/java_ref_string.asp.

3.3.2 Operator

Kalian tentu pernah melihat dan mempelajari tanda-tanda seperti “+” dan “-” di matematika, bukan? Nah, mereka disebut juga dengan operator. Operator digunakan untuk melakukan operasi tertentu. Dalam bahasa pemrograman, operator ini terdiri dari beberapa jenis.

3.3.2.1 Operator Aritmatika

Operator Aritmatika, atau *Arithmetic Operator* adalah operator yang digunakan untuk operasi matematika. Ada tambah, kurang, bagi, kali, dan sebagainya, yang tentunya sudah kalian kenal. Kalian bisa melihat penggunaannya pada kode berikut ini:

```
public class Main {  
    public static void main(String[] args) {  
        int a = 20;  
        int b = 5;  
  
        // Pertambahan  
        int pertambahan = a + b;  
        System.out.println("Hasil pertambahan: " + pertambahan);  
  
        // Pengurangan  
        int pengurangan = a - b;  
        System.out.println("Hasil pengurangan: " + pengurangan);  
  
        // Perkalian  
        int perkalian = a * b;  
        System.out.println("Hasil perkalian: " + perkalian);  
  
        // Pembagian  
        int pembagian = a / b;  
        System.out.println("Hasil pembagian: " + pembagian);  
  
        // Sisa bagi (modulus)  
        int modulus = a % b;  
        System.out.println("Hasil modulus: " + modulus);  
    }  
}
```

Dan output yang dihasilkan adalah:

```
• -> java Main
    Hasil pertambahan: 25
    Hasil pengurangan: 15
    Hasil perkalian: 100
    Hasil pembagian: 4
    Hasil modulus: 0
```

Mudah, kan? Mungkin ada satu operator yang agak asing bagi kalian, yaitu operator sisa bagi atau modulus. Nah, operator ini akan mengembalikan nilai sisa dari pembagian yang dilakukan. Contohnya, kalian tidak bisa membagi habis 7 dengan 3, bukan? Yang akan habis dibagi dengan 3, yang paling dekat dengan 7 adalah 6. 6 dibagi 3 hasilnya adalah 2, dan tidak ada nilai yang tersisa. Karena itu, bila kalian membagi 7 dengan 3, maka nilai yang dapat dibagi yang paling mendekati adalah 6. Dan 7 dikurangi dengan 6, hasilnya adalah 1. Tersisa 1 nilai yang tidak bisa dibagi. Oleh karena itu, modulus dari 7 dibagi 3 adalah 1.

Biasanya, modulus ini digunakan untuk mengecek apakah suatu bilangan ganjil atau genap. Ingat, kan? Bilangan ganjil itu yang tidak habis dibagi 2 sedangkan bilangan genap adalah bilangan yang habis dibagi 2.

```
int ganjil = 5 % 2 // Hasilnya 1 (tidak habis);
int genap = 4 % 2 // Hasilnya 0 (habis);
```

Oh iya, operator pertambahan (+) bisa juga digunakan untuk melakukan *string concatenation* atau penggabungan String, seperti yang sudah kalian lihat di fungsi `System.out.println` pada gambar-gambar di atas! Kita bisa menambahkan satu potongan string dengan potongan string atau nilai lainnya!

Selain itu, operator aritmatika meliputi juga operator inkremen dan dekremen atau *increment* dan *decrement operators* di mana kita bisa menambahkan 1 atau mengurangi 1 dari sebuah nilai tipe data numerik. Namun, ada dua tipe inkremen dan dekremen, yaitu:

- Pre-Increment/Decrement
- Post-Increment/Decrement

Post-increment/decrement adalah operator yang menambahkan 1 ke nilai variabel, tetapi melakukannya **setelah** nilai variabel digunakan dalam ekspresi. Post-decrement juga sama, namun mengurangi 1 dari nilai variabel.

```
int a = 5;
```

Rizza Mandasari, Rahman Hakim

Ilustrasi oleh Nazwa Mursyidan Baldan


```
int b = a++; // Di sini, a bernilai 6 dan b bernilai 5
int c = 5;
int d = c--; // Di sini, c bernilai 4 dan d bernilai 5
```

Dengan post-increment/decrement, nilai inkremen atau dekremen yang di-assign kepada variabel lain tidak akan berubah karena variabel yang di inkremen/dekremen baru akan diproses setelah nilai diberikan kepada variabel lain tersebut. Jadi pada kode di atas, b akan menyimpan 5 terlebih dahulu, kemudian baru a akan ditambahkan 1 menjadi 6. Hal yang sama berlaku pada dekremen.

Pre-increment adalah operator yang menambahkan 1 ke nilai variabel, tetapi melakukannya sebelum nilai variabel digunakan dalam ekspresi.

```
int a = 5;
int b = ++a; // Di sini, a bernilai 6 dan b bernilai 6
int c = 5;
int d = --c; // Di sini, c bernilai 4 dan d bernilai 4
```

Tidak seperti post-increment/decrement, nilai akan langsung berubah setelah digunakan. Karena itu, sekarang b langsung bernilai sama dengan a ketika ++a di-assign kepada b. Begitu pula dengan c dan d.

3.3.2.2 Operator Penugasan

Operator Penugasan, atau *Assignment Operator* adalah operator yang pernah dibahas di bab sebelumnya. Ia merupakan operator “=” yang digunakan untuk menyuruh variabel menyimpan sebuah nilai. Bisa kita pikirkan sebagai: “Kamu, variabel x yang bertipe integer. Kamu ditugaskan untuk menyimpan nilai 10 ya!”.

3.3.2.3 Operator Perbandingan

Operator Perbandingan, atau *Comparison Operator* adalah operator yang digunakan untuk membandingkan sesuatu dengan sesuatu yang lainnya. Ingat, operator perbandingan hanya akan mengembalikan boolean saja! Ia disebut juga sebagai operator logika atau *logic operator* karena operator inilah yang akan digunakan untuk menjalankan logika-logika kita. Ia hanya akan mengembalikan antara “true” (benar) atau “false” (salah).

Mari kita membuat sebuah program sederhana dengan 3 variabel. Beri saja nama “a”, “b”, dan “c” yang masing-masing memiliki nilai tersendiri. Kemudian, kita akan bandingkan satu sama lain dan kita akan melihat output yang dihasilkan dari perbandingan-perbandingan tersebut.

```

public class Main {
    public static void main(String[] args) {
        // Operator Perbandingan
        int a = 10;
        int b = 10;
        int c = 5;

        // Operator sama dengan (==)
        System.out.println("Apakah a sama dengan b? : " + (a == b));
        System.out.println("Apakah a sama dengan c? : " + (a == c));

        // Operator tidak sama dengan (!=)
        System.out.println("Apakah a tidak sama dengan b? : " + (a != b));
        System.out.println("Apakah a tidak sama dengan c? : " + (a != c));

        // Operator lebih besar dari (>)
        System.out.println("Apakah a lebih besar dari b? : " + (a > b));
        System.out.println("Apakah a lebih besar dari c? : " + (a > c));

        // Operator lebih kecil dari (<)
        System.out.println("Apakah a lebih kecil dari b? : " + (a < b));
        System.out.println("Apakah a lebih kecil dari c? : " + (a < c));

        // Operator lebih besar sama dengan (>=)
        System.out.println("Apakah a lebih besar sama dengan b? : " + (a >= b));
        System.out.println("Apakah a lebih besar sama dengan c? : " + (a >= c));

        // Operator lebih kecil sama dengan (<=)
        System.out.println("Apakah a lebih kecil sama dengan b? : " + (a <= b));
        System.out.println("Apakah a lebih kecil sama dengan c? : " + (a <= c));
    }
}

```

Dan output yang akan dihasilkan adalah:

```
• -> java Main
Apakah a sama dengan b? : true
Apakah a sama dengan c? : false
Apakah a tidak sama dengan b? : false
Apakah a tidak sama dengan c? : true
Apakah a lebih besar dari b? : false
Apakah a lebih besar dari c? : true
Apakah a lebih kecil dari b? : false
Apakah a lebih kecil dari c? : false
Apakah a lebih besar sama dengan b? : true
Apakah a lebih besar sama dengan c? : true
Apakah a lebih kecil sama dengan b? : true
Apakah a lebih kecil sama dengan c? : false
```

3.3.2.4 Operator Logika

Operator logika ini merupakan operator yang mirip dengan gerbang logika. Beberapa di antaranya yang paling sering digunakan adalah AND, OR, dan NOT.

```
public class Main {
    public static void main(String[] args) {
        // Operator logika
        boolean benar = true;
        boolean salah = false;

        // OR
        System.out.println("Benar atau salah: " + (benar || salah));
        System.out.println("Salah atau salah: " + (salah || salah));
        System.out.println("Benar atau benar: " + (benar || benar));

        // AND
        System.out.println("Benar dan salah: " + (benar && salah));
        System.out.println("Salah dan salah: " + (salah && salah));
        System.out.println("Benar dan benar: " + (benar && benar));

        // NOT
        System.out.println("Tidak benar: " + !benar);
        System.out.println("Tidak salah: " + !salah);
    }
}
```

Rizsa Mandasari, Rahman Hakim

Ilustrasi oleh Nazwa Mursyidan Baldan

```
}  
}
```

```
>> "Linux Mint" - rahman @ furude-PC in ~/Projects/Java/Permulaan  
● -> java Main  
Benar atau salah: true  
Salah atau salah: false  
Benar atau benar: true  
Benar dan salah: false  
Salah dan salah: false  
Benar dan benar: true  
Tidak benar: false  
Tidak salah: true
```

3.3.2 Konstanta

Konstanta merupakan variabel juga. Namun bedanya, dia tidak bisa di-assign ulang. Sekali kita membuat konstanta, ia akan **konstan**, atau tidak akan berubah-ubah. Secara normal, kita bisa mengubah nilai variabel seperti menambahkan, atau melakukan assignment ulang.

```
int a = 10;  
a = 5;  
a += 1;
```

Dan nilai variabel akan berubah-ubah! Namun, tidak dengan konstanta. Saat membuat sebuah konstanta, kita biasanya menggunakan huruf kapital untuk semua huruf, dan ditaruh di luar blok program utama karena dia akan tetap sama selama program berjalan!

```
public class ProgramDira {  
  
    static final double PI = 3.14;  
  
    public static void main(String[] args) {
```

Contohnya adalah variabel PI di atas. Dalam matematika, pi tidak mungkin berubah, bukan? Pasti akan selalu 3.14. Karena itu, kita memakai konstanta untuk pi. Pembuatan konstanta menggunakan *keyword* final, yang artinya variabel tersebut sudah final dan tidak bisa diapa-apain lagi, hanya bisa dipakai. Penggunaan static di atas itu untuk menunjukkan bahwa variabel PI tersebut tidak ada hubungannya dengan kelas kita. Ia bersifat statis. Bisa kita anggap “independen” lah, haha!

Ada banyak sekali tipe data dan *operator* yang bisa kita gunakan, bukan? Oleh karena itu, sebaiknya kalian banyak latihan dan belajar mandiri, ya! Soalnya dunia pemrograman itu memang luas!

Kalau begitu, sampai bertemu di bab selanjutnya!