Java Developer Fundamentals

2nd Module

Programming Basics

Blok Program Java

Blok program merupakan kumpulan dari statement dan ekspresi yang dibungkus menjadi satu.

Blok program selalu dibuka dengan kurung kurawal { dan ditutup dengan }.

Contoh blok program:

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

    System.out.println("Hello World");
    System.out.println("Hello Kode");

    // blok program if
    if( true ){
        System.out.println('True');
    }

    // blok program for
    for ( int i = 0; i<10; i++){
        System.out.println("Perulangan ke"+i);
    }

}
```

Deklarasi variabel

Sintak umum untuk mendeklarasikan variable

<Tipe data> <Nama variabel>;

Contoh : mendeklarasikan variabel id dengan tipe data int
int id;

Inisialisasi / Assignment Variabel

Sintak umum untuk inisialisasi variable

```
<Nama variabel> = value;
```

Contoh : memberikan nilai 10 pada variabel id

```
id = 10;
```

Inisialisasi / Assignment Variabel

Variable dapat pula diinisialisasi dengan expression

<Nama variabel> = expression;

Contoh : memberikan nilai yang didapakan dari perhitungan 5*2 pada variable id

id = 5*2;

Inisialisasi / Assignment Variabel

- Nama *variable* harus berada di sebelah kiri, sehingga syntax seperti 1 = x adalah salah.
- *Expression* merepresentasikan perhitungan yang melibatkan nilai, *variable* dan operator.
- *Variable* yang berada di sebelah kiri dapat juga digunakan pada *expression* di sebelah kanan. Contoh:

```
x = x + 1
```

- Pada contoh di atas, x + 1 diberikan kepada x. Jika nilai x adalah 1 sebelum *statement* di atas dieksekusi, maka x akan bernilai 2 setelah *statement* di eksekusi.

Penggabungan Deklarasi dan Assignment

Variable seringkali memiliki nilai awal. Dapati dilakukan deklarasi *variable* dan memberikannya nilai awal dalam satu langkah. Contoh:

```
int x = 1;
```

Atau

```
int x;

x = 1;
```

Data Type - numeric

Setiap jenis data memiliki rentang (domain) nilai tertentu. Kompiler mengalokasikan ruang pada memory untuk menyimpan *variable* berdasarkan pada jenis datanya. Java menyediakan beberapa jenis data primitive seperti *numeric*, *character*, dan *boolean*. Java mempunyai enam jenis data *numeric* yang terbagi menjadi empat jenis data integer dan dua jenis data *floating-point* dan alokasi memory-nya.

Nama	Rentang (domain)	Ukuran memory
byte	$-2^7(-128)$ *sampai* $2^7-1\ (127)$	8-bit signed
short	$-2^{15}(-32768)$ *sampai* $2^{15}-1\ (32767)$	16-bit signed
int	$-2^{31}(-2147483648)$ *sampai* $2^{31}-1\ (2147483647)$	32-bit signed
long	$-2^{63}(-9223372036854775808)$ *sampai* $2^{63}-1\ (9223372036854775808)$	64-bit signed
float	$-3.4E38$ *sampai* $3.4E38$ $(6\ to\ 7\ significat\ bit\ accuracy)$	32-bit IEEE 754
double	$-1.7E308$ *sampai* $1.7E308$ $(14\ to\ 15\ significat\ bit\ accuracy)$	64-bit IEEE 754

Data Type - character

Jenis data karakter, char, digunakan untuk merepresentasikan sebuah karakter. Sebuah nilai ditulis didalam single quotation (‘ ‘).

Contoh:

```
char a = ‘A’;

char num = ‘5’;
```

Data Type - character

Java juga mengijinkan untuk menggunkan ‘escape’ karakter. Escape karakter diawali dengan backslash karakter (\ ) diikuti oleh sebuah karakter, dimana escape karakter ini memiliki arti yang khusus bagi kompiler.

Deskripsi	Escape karakter
Backspace	\b
Tab	\t
Linefeed	\n
Carriage return	\r
Backslash	\\
Single Quote	\’
Double Quote	\”

Data Type - boolean

- Jenis data *Boolean* memiliki dua nilai yaitu: true dan false.
- Syntax memberikan data jenis *boolean* bernilai true pada *variable b*:

```
boolean b = true;
```

- Operator yang berhubungan dengan *boolean* adalah operator perbandingan dan operator *Boolean*. Operator perbandingan dikenal juga sebagai relational operators, yang dapat digunakan pada *expression* yang menghasilkan nilai *boolean*.

Data Type - object

- Pada Java juga juga terdapat data dengan tipe object seperti String, Array, Mobil, Sepeda, Kendaraan dan lain sebagainya. Hal ini akan dijelaskan lebih lanjut pada pertemuan-pertemuan berikutnya

Variable Casting

Pada banyak kasus kita sering terpaksa mengubah suatu variabel dengan tipe tertentu kepada tipe yang lain.

Mengubah tipe data atau tipe object pada suatu variabel inilah yang disebut dengan istilah Casting. Terdapat dua jenis casting pada Java;

- Implitic / Automatic Casting
- Explicit / Manual Casting

Casting Variable

AUTOMATIC CASTING / IMPLICIT CASTING

Automatic Casting / Implicit Casting digunakan untuk mengubah nilai dari tipe data kecil ke tipe data besar, seperti berikut ini:

- Dari tipe data **short** ke tipe data **int, long, float,** atau **double**
- Dari tipe data **int** ke dalam tipe data **long, float**, atau **double**
- Dari tipe data **char** ke dalam tipe data **int, long, float**, atau **double**
- Dari tipe data **float** ke dalam tipe data **double**
- Dari tipe data **long** atau ke dalam tipe data f**loat** atau **double**
- Dari tipe data **byte** ke tipe data **short, int, long, float**, atau **double**

Mari kita lihat contoh kasus berikut ini.

```
// Automatic casting
int suatuInteger = suatuShort;
```

Casting Variable

AUTOMATIC CASTING / IMPLICIT CASTING

Automatic casting terjadi apabila **tipe data lama sudah pasti juga merupakan tipe data baru**.

Sebagai contoh :

```
int i = 24234; long l = i; // semua int sudah pasti juga merupakan long
```

Karena int berada dalam range yang lebih sempit dibandingkan dengan long, maka otomatis **tipe data int juga merupakan tipe data long**. Karena kondisi ini kita hanya perlu melakukan automatic casting dalam kasus ini.

Casting Variable

Contoh Program Implicit Casting :

```
public class latihan_java {
  public static void main(String[] args){
    short data1 = 457;
    double double_data = data1; //short ke double
    char data2 = 'W';
    long long_data = data2;//char ke long
    int data3 = 456;
    float int_data = data3;// int ke float
    byte data4 = 127;
    long long_data2 = data4;//byte ke long
    float data5 = 565.3f;
    double double_data2 = data5;//float ke double
System.out.println("Short ke Double: "+ double_data);
System.out.println("Char ke Long: "+long_data);
System.out.println("Int ke Float: "+int_data);
System.out.println("Byte ke Long: "+long_data2);
System.out.println("Float ke Double: "+double_data2);
   }
}
```

Casting Variable

MANUAL CASTING / EXPLICIT CASTING

Explicit Casting / Manual Casting adalah kebalikan dari Implicit Casting yaitu untuk mengubah nilai dari tipe data besar ke tipe data kecil, seperti ini:
Dari tipe data **short** ke **byte** atau **char**
•Dari tipe data **double** ke **byte, short, char, int, long** atau **float**
•Dari tipe data **char** ke **byte** atau **short**
•Dari tipe data **long** ke **byte, short, char** atau **int**
•Dari tipe data **int** ke **byte, short**, atau **char**
•Dari tipe data **float** ke **byte, short, char, int** atau **long**
•Dari tipe data **byte** ke dalam tipe data **char**

Casting Variable

MANUAL CASTING / EXPLICIT CASTING

Manual casting dilakukan apabila **tipe data lama belum tentu juga merupakan tipe data baru**. Sebagai contoh :

```
long l = 92233720368547750001;
int i = (int) l; // semua long belum tentu merupakan int
```

- Karena long berada dalam range yang lebih lebar dibandingkan dengan int, maka otomatis **tipe data long belum tentu juga merupakan tipe data int**. Karena kondisi ini kita perlu melakukan manual casting dalam kasus ini.
- Yang perlu diingat **manual casting dapat menyebabkan suatu nilai menjadi kehilangan presisi**. Seperti contoh diatas, angka 92233720368547750000 tidak mungkin masuk kedalam range int, maka variabel i tidak akan mempunyai nilai yang sama dengan 92233720368547750000.

Casting Variable

Contoh Program Explisit Casting :

```
public class latihan_java {
  public static void main(String[] args){
  short data1 = 3;
  Char char_data = (char) data1; //short ke char
  long data2 = 246447;
  byte byte_data = (byte) data2;//long ke byte
  int data3 = 34;
  char char_data2 = (char) data3;// int ke char
  char data4 = 1;
  short short_data2 = (short) data4;//char ke short
  double data5 = 345.3;
  float float_data2 = (float) data5;//double ke float
  System.out.println("Short ke Char: "+ char_data);
System.out.println("Long ke Byte: "+byte_data);
System.out.println("Int ke Char: "+char_data2);
System.out.println("Char ke Short: "+short_data2);
System.out.println("Double ke Float: "+float_data2);
   }
}
```

Perbedaan Konversi dan Casting pada Java

Konversi adalah pemberian nilai kepada variabel yang berbeda tipe datanya. Dalam konversi melibatkan widening conversation yaitu tipe data tujuan harus mempunyai range yang lebih besar daripada tipe data aslinya. Misalnya dari byte ke short, short ke long, atau int ke double. Konversi tidak dapat dilakukan pada tipe data boolean.

```
public class Promotion{
          public static void main (String[] args){
                    float f = 3.324243532f;
                    double d = 3.23442323445242;
                    System.out.println("Default value f = "+f);
                    System.out.println("Default value d = "+d);

                    d = f;

                    System.out.println("Default value f (long)    = "+f);
                    System.out.println("Default value d (float)   = "+d);
          }
}
```

Variable Passing by value vs Passing by Reference

- Pada passing by value, suatu nilai yang ditampung oleh variabel langsung disimpan pada memory komputer. Jika variabel tersebut dipanggil oleh suatu proses, akan diciptakan salinan yang berisi nilai yang sama dengan nilai yang tersimpan pada memory, namun tidak berhubungan satu sama lain. Sehingga jika pemanggil melakukan modifikasi terhadap nilai yang didapatkan, nilai yang tersimpan di memory komputer tidak berubah. Pada Java, proses ini terjadi pada variabel dengan tipe primitive (numeric, char, boolean).
- Pada passing by reverence, variable hanya tidak menyimpan nilai secara langsung, melainkan menyimpan alamat memory komputer di mana nilai tersebut sebenarnya disimpan. Jika variabel tersebut dipanggil oleh suatu proses, akan dikirimkan alamat dari tempat tersimpannya nilai tersebut. Sehingga, jika pemanggil melakukan modifikasi terhadap isi variabel, nilai yang tersimpan di memory akan ikut berubah. Pada Java, proses ini terjadi pada variabel yang menampung objek.

Variable Passing by value vs Passing by Reference

```
int umur = 81;
int duplikatUmur = umur;
duplikatUmur = 1308;

System.out.println(umur);
System.out.println(duplikatUmur);
```

Passing by Value

```
Rectangle kotak = new Rectangle(0, 0, 20, 30);
Rectangle duplikatKotak = kotak;
duplikatKotak = setSize(80,90);

System.out.println(kotak.getSize());
System.out.println(duplikatKotak.getSize());
```

Passing by Reference

RECTANGLE
X = 0
Y = 0
Height = 80
Width = 80

Sel memori no 5e265ba4

81	81	Sel memori no 5e265ba4	Sel memori no 5e265ba4
umur	duplikatUmur	kotak	duplikatKotak
81	1308	Sel memori no 5e265ba4	Sel memori no 5e265ba4
umur	duplikatUmur	kotak	duplikatKotak