BAB II

Program Pertama

2.1. Dua kriteria program yang benar

Sebagaimana di tulis dalam Bab I, untuk dapat membuat program yang baik maka Anda harus mengenali kosa kata, aturan sintaksis (grammar) dan makna atau arti (semantik) dari kata dan kalimat yang Anda susun. Program yang Anda tulis bisa saja secara sintaksis benar tetapi hasilnya keliru (secara semantik salah). Tentu saja hampir tidak mungkin membuat program yang secara semantik benar tetapi secara sintaktik salah. Semakin kompleks program yang Anda tulis maka program Anda tidak hanya harus benar secara sintaksis dan semantik tetapi juga harus efisien dan praktis. Maksudnya program Anda harus berjalan dengan cepat, mudah dibaca orang lain, serta mudah dimodifikasi.

2.2. Program Pertama

Perhatikan rangkaian perintah (program) berikut

```
// Program untuk menampilkan tulisan
// "Selamat Datang di USD" pada layar komputer

public class SelamatDatang{
        public static void main(String[] args) {
                System.out.println("Selamat Datang di USD");
        }
} // Akhir dari kelas SelamatDatang
```

Jika program tersebut di atas dieksekusi oleh *compiler Netbean* maka akan menghasilkan tulisan di layar

```
Selamat Datang di USD
```

tanpa ada tanda petiknya.

Tulisan *Selamat Datang di USD* dihasilkan oleh perintah

```
System.out.println("Selamat Datang di USD");
```

Perintah ini sebenarnya bukan perintah biasa dan bukan perintah sederhana karena berupa perintah panjang kata-katnya dipisahkan dengan titik yakni **System** dengan **S** besar lalu **titik** lalu kata **out** dengan **o** kecil lalu **titik** lagi dan kemudian kata **println** dan diikuti **kurang buka** lalu **tanda petik 2** kemudian kalimat **Selamat Datang di USD** lalu tanda **petik 2** lagi lalu **kurung tutup** dan diakhiri dengan **titik koma**.

Perlu Anda ketahui bahwa perintah tersebut harus Anda tulis seperti itu, baik urutannya maupun jenis huruf besar kecilnya. Perkecualian terjadi hanya pada rangkaian kata-kata yang terletak di dalam tanda petik 2. Anda boleh menuliskan apapun di antara dua tanda petik tersebut. Misalnya program di atas Anda ubah menjadi

```
// Program untuk menampilkan tulisan
// “Selamat Datang di USD” pada layar komputer

public class SelamatDatang {
        public static void main(String[] args) {
                System.out.println("Piye kabare, lak penak jamanku tho…?");
        }
} // Akhir dari kelas SelamatDatang
```

maka setelah dieksekusi akan menghasilkan tulisan berikut di layar komputer Anda

```
Piye kabare, lak penak jamanku tho…?
```

Anda tentu bisa menebak dengan mudah apabila ingin mencetak tulisan di layar komputer

```
Selamat Datang di USD
Piye kabare, lak penak jamanku tho…?
```

Maka program di atas harus Anda modifikasi sehingga menjadi

```
// Program untuk menampilkan tulisan
// kalimat tertentu pada layar komputer

public class SelamatDatang {

        public static void main(String[] args) {
                System.out.println("Selamat Datang di USD");
                System.out.println("Piye kabare, lak penak jamanku tho…?");
        }
} // Akhir dari kelas SelamatDatang
```

Dua baris pertama program di atas yakni:

```
// Program untuk menampilkan tulisan
// “Selamat Datang di USD” pada layar komputer
```

tidak akan berpengaruh apa-apa terhadap jalannya program. Semua tulisan yang diawali dengan dua garis miring (//) akan diabaikan oleh *compiler*. Meskipun diabaikan tetapi tulisan seperti ini sangat banyak manfaatnya. Semakin program yang anda buat semakin rumit maka akan lebih baik (untuk kepentingan Anda sendiri maupun orang lain) kalau Anda memberi komentar atau catatan di dalam teks program.

Ada dua cara memberi komentar di dalam teks program. Pertama dengan mengawalinya memakai dua garis miring (//) seperti di atas dan yang kedua meletakkannya diantara garis miring dan bintang (/* .. */). Cara pertama digunakan pada setiap baris komentar, sedangkan cara yang kedua diletakkan di awal dan di akhir dari lebih dari satu baris komentar. Jadi komentar

```
// Program untuk menampilkan tulisan
// “Selamat Datang di USD” pada layar komputer
```

dapat juga ditulis dengan cara

```
/ * Program untuk menampilkan tulisan
  “Selamat Datang di USD” pada layar komputer */
```

Anda boleh meletakkan komentar seperti contoh di atas di manapun di dalam teks program sejauh tidak memenggal perintah yang sudah ditetapkan.

Selain komentar di atas, semua teks program **harus** mengikuti ketentuan penulisan (sintaksis) yang sudah ditetapkan oleh Java. Hal pertama yang harus Anda mengerti adalah bahwa semua program dalam Java harus masuk ke dalam apa yang disebut dengan **class.** Setiap class harus mempunyai nama maka program di atas masuk kedalam class yang bernama **SelamatDatang**. Nama class agak boleh bebas tetapi juga ada aturannya. Di depan kata class ada jenis atau tipe class yang untuk kasus di atas berjenis ***public*.** Kata *public* berarti umum atau terbuka sehingga *class* yang berjenis public dapat dipanggil/digunakan oleh *class* lain yang membutuhkan. Segala hal yang ada di dalam *class*, dapat berupa metode atau variabel/atribut harus diletakkan di antara kurung kurawal buka ({) dan ditutup dengan kurung kurawal tutup (}).

Secara umum class yang akan kita pakai memiliki sintaks :

```
// komentar program
/* bisa diletakkan pada
   lebih dari satu baris */

public class SelamatDatang  {  // Awal dari kelas SelamatDatang

        <variabel atau metode>

} // Akhir dari kelas SelamatDatang
```

Perlu kita ingat bahwa tidak setiap class akan menjadi bagian dari program. Hanya class yang memuat metode/subprogram yang bernama **main** yang akan dieksekusi oleh compiler. Class yang bernama SelamatDatang di atas memiliki satu metode yakni main(). Perintah lengkap metode *main* adalah

```
public static void main(String[] args) {
        System.out.println("Selamat Datang di USD");
}
```

Secara umum, metode main akan berbentuk seperti berikut:

```
public static void main(String[] args)
{
        < pernyataan/perintah>
}
```

Artinya di dalam metode main boleh memuat lebih dari satu perintah atau pernyataan dan semua pernyataan tersebut harus berada/terletak di antara kurung kurawal ({ .... }). Selain itu, setiap pernyataan juga harus diakhiri dengan titik koma (;). Kata *public static void* yang berada di depan kata main dan kata (String[] args) untuk sementara Anda tulis saja begitu apa adanya tanpa perlu mengerti apa maksudnya.

Akhirnya, sintaks sebuah class yang berisi sebuah metode main adalah sebagai berikut :

```
// komentar program
/* bisa diletakkan pada
   lebih dari satu baris */

public class SelamatDatang {

        public static void main(String[] args) {

                < pernyataan/perintah>

        } //akhir dari metode main

} // Akhir dari kelas SelamatDatang
```

2.3. Variabel dan Tipe/Jenis Dasarnya (Primitifnya)

Nama sangatlah penting dalam pemrograman. Oleh karena itu sajak terkenal dari Shakespeare yang berbunyi *"What's a name"* tidak berlaku dalam pemrograman. Semua hal di dalam program dikenali atau dirujuk memakai nama yang ditentukan sendiri oleh pemrogram. Untuk bisa membuat program dengan baik maka pemrogram harus menguasai aturan penulisan serta makna dari setiap nama yang dibuat atau diberikan.

Menurut aturan sintaksis Java, nama segala sesuatu di sebuah program harus berupa rangkaian alphabet yang memenuhi 4 syarat berikut:

1) huruf besar atau kecil,
2) angka dan/atau garis bawah (_)
3) tidak boleh memuat spasi (jeda)
4) tidak boleh diawali dengan angka

Dua nama yang *bunyinya sama* tetapi tersusun dari jenis huruf besar atau huruf kecil yang berbeda menyatakan nama/hal yang berbeda. Berikut adalah nama-nama yang dapat diterima oleh Java

N m M16 asal_mhs rata_rata MYNAME muterTerus

Sedang nama-nama berikut tidak boleh digunakan alias ilegal dan menyebabkan *error*.

16Orang	(*karena diawali angka*)
Rata rata	(*karena memakai spasi*)
kelulusan%	(*karena memakai %)*
laki&perempuan20	(*karena memakai &)*

Nama-nama berikut dapat dipakai tetapi masing-masing merupakan nama yang berbeda sehingga merujuk ke hal/tempat yang berbeda

Rata_rata rata_rata rata_Rata rerata Rerata

Beberapa kata tidak boleh digunakan karena sudah dipakai sebagai kata bawaan oleh Java (*reserved words*). Kata-kata ini meliputi: **class, public, static, if, else, for, while**, dll. Supaya Anda mudah membuat nama maka buatlah nama yang tidak memakai kosa kata bahasa Inggris maka pasti dijamin tidak bertabrakan dengan nama yang sudah dipakai oleh Java.

Selain ketentuan di atas, ada tradisi dalam komunitas pemrogram Java dalam pemberian/pembuatan nama di Java. Tradisi tersebut adalah

1) Nama class diawali dengan huruf besar, misalnya Mahasiswa, Karyawan, Dosen, dll
2) Nama variabel (dijelaskan di bawah) diawali dengan huruf kecil, misalnya rata2, panjang, indeks
3) Pemrogram Java jarang memakai garis bawah dan sebagai gantinya menggunakan huruf besar di awal kata, misalnya rataRata, bungaTahunan, indeksPrestasi
4) Beberapa variabel khusus biasa ditulis dengan garis bawah, misalnya _BUNGADEPOSITO

Terlepas dari tradisi di atas, yang paling penting adalah, berilah nama yang bermakna, konsisten, sederhana dan tidak terlalu panjang. Nama-nama yang benar di atas cukup baik digunakan tetapi nama-nama berikut sebaiknya dihindari, seperti panjangSisiSegitigaABC, superXtremk234Xysd.

2.3.1. Variabel

Variabel adalah nama tempat di memori komputer. Memori komputer adalah tempat di mana data disimpan sementara selama program berjalan. Pemrogram bisa memesan tempat di memori dengan cara menyebutkan tipe/jenis data yang akan disimpan lalu menuliskan nama tempat yang dipesan. Pemrogram tidak perlu memikirkan di mana tepatnya tempat yang dipesan tersebut di memori. Prinsip yang akan dipakai oleh compiler adalah **dua nama variabel yang berbeda merujuk ke dua tempat yang berbeda.**

2.3.2. Tipe Data Dasar

Java mempunyai 8 tipe/jenis data dasar yakni *byte, short, int, long, float, double, char,* dan *boolean.* Empat tipe yang pertama yakni *byte, short, int, long* semuanya terkait dengan bilangan bulat seperti 22, 245, 1025, dst. Perbedaanya terletak kepada besarnya bilangan bulat yang bisa disimpan. Sedang jenis data *float* dan *double* keduanya terkait dengan tempat untuk bilangan real atau pecahan. Perbedaanya hanya pada kapasitas atau besarnya data yang bisa disimpan. Tipe data *char* dipakai untuk merujuk ke data yang berupa huruf seperti 'A', 'z', '*', dll. Akhirnya tipe data *boolean* dipakai untuk merujuk ke data yang mempunyai nilai kebenaran yakni 'true' atau 'false' saja.

Selain kedelapan tipe data dasar di atas, pemrogram dapat membuat tipe/jenis data yang lebih kompleks tetapi akan dibicarakan ketika sudah membahas tentang obyek. Namun demikian ada tipe data yang tidak dasar karena sudah berupa obyek tetapi penting dikenali sejak awal yakni tipe data *String*. Tipe data ini merujuk ke data yang berupa rangkaian huruf yang akhirnya membentuk kata atau kalimat. Data dengan tipe String disajikan dengan meletakkannya di dalam dua tanda petik ganda, seperti "Cerdas dan Humanis", "Kartu Hasil Studi", "Pajak kendaraan tahun 2013", dll.

Tabel berikut menyajikan jangkauan data yang bisa disimpan ke dalam variabel dengan tipe/jenis data di atas

Tabel 2.1. Tipe data dasar beserta jangkauan nilai-nilainya

Type	Size in bits	Values	Standard
boolean		true or false	
[*Note:* A boolean's representation is specific to the Java Virtual Machine on each platform.]			
char	16	'\u0000' to '\uFFFF' (0 to 65535)	(ISO Unicode character set)
byte	8	128 to +127 (2^7 to 2^7 1)	
short	16	32,768 to +32,767 (2^{15} to 2^{15} 1)	
int	32	2,147,483,648 to +2,147,483,647 (2^{31} to 2^{31} 1)	
long	64	9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 (2^{63} to 2^{63} 1)	
float	32	*Negative range:* 3.4028234663852886E+38 to 1.40129846432481707e45 *Positive range:* 1.40129846432481707e45 to 3.4028234663852886E+38	(IEEE 754 floating point)
double	64	*Negative range:* 1.7976931348623157E+308 to 4.94065645841246544e324 *Positive range:* 4.94065645841246544e324 to 1.7976931348623157E+308	(IEEE 754 floating point)

2.3.3. Menggunakan Variabel

Pemesanan tempat di memori memakai nama variabel tertentu (biasa disebut dengan deklarasi variabel), dilakukan dengan menulis perintah memakai pola/format berikut:

<nama-tipe-data> <nama satu atau lebih variabel yang dipisah dng koma>;

Contoh:

```
int jumlahMahasiswa;
double modal2013;
int jumDosen, jumPegawai, jumKontrak;
double rerataIPS, rerataIPK;
char nilaiFinal;
boolean sudahLulus;
String namaOrtu, alamatOrtu;
```

Merupakan sebuah praktek penulisan program yang baik apabila deklarasi variabel juga kita sertai keterangan sehingga mudah dipahami maksudnya, seperti contoh berikut:

```
int jumlahMahasiswa;              //jumlah mahasiswa aktif
double modal2013;                 //modal awal di tahun 2013
double rerataIPS, rerataIPK;      //rata-rata indeks prestasi semester dan kumulatif
char nilaiFinal;                  //nilai matakuliah final
boolean sudahLulus;               //status kelulusan mahasiswa
String namaOrtu, alamatOrtu;      //nama dan alamat orang tua
```

Setelah tempat di memori dipesan dengan perintah seperti di atas, maka variabel tersebut bisa diisi dengan data. Ada banyak cara mengisi data tetapi untuk sementara data diisikan lewat teks program. Pengisian data ke variabel tertentu dilakukan dengan perintah yang memakai pola/format berikut:

```
nama-variabel = data-yang-sejenis/setipe-dengan-nama-variabel-tersebut;
```

Memakai variabel dengan tipe di atas, berikut contoh pengisian data ke memori yang diijinkan.

```
jumlahMahasiswa = 200;
modal2013 = 150000000; //150 juta rupiah
rerataIPS = 2.56;
rerataIPK = 2.45;
nilaiFinal = 'B';
sudahLulus = true;
namaOrtu ="Susilo Harmoko";
alamatOrtu ="Jln. Merdeka Selatan No. 15 Jakarta Pusat";
```

Setelah variabel berisi data maka variabel tersebut bisa dimanipulasi (dikenai perhitungan) sesuai dengan kepentingan pemrogram. Operasi dasar untuk memanipulasi variabel hanya bisa dilakukan kalau operasi tersebut 'masuk akal'. Maka tidak diijinkan oleh Java untuk mengalikan variabel **nilaiFinal** di atas dengan bilangan bulat **10** karena tidak bermakna dan tidak masuk akal. Tetapi mengalikan variabel **rerataIPS** dengan bilangan **2** tetap diijinkan meskipun 2 adalah bilangan bulat dan **rerataIPS** berisi bilangan real/pecahan. Dan bisa ditebak hasilnya akan berupa

bilangan real. Hasil manipulasi suatu variabel kemudian dapat disimpan ke variabel yang lain asal sudah dipesan terlebih dahulu (dideklarasikan).

Berikut adalah program sederhana yang menghitung luas segitiga siku-siku berdasarkan panjang sisi yang saling tegak lurus. Rumus luas segitiga ini adalah 0.5 x alas x tinggi.

```
/**
* Program sederhana menampilkan sisi segitiga siku-siku
* .dan menghitung luasnya
*/
public class LuasSegitiga {
    public static void main(String[] args) {
        double alas;        // deklarasi variabel untuk alas segitiga
        double tinggi;      // deklarasi variabel untuk tinggi segitiga
        double  luas;       // deklarasi variabel untuk luas segitiga
        alas  = 20;
        tinggi = 30;
        luas = 0.5 * alas * tinggi ; // kali (x) ditulis * (bintang) di Java dan
                                     // koma (,) ditulis titik (.)
        // Mulai menampilkan hasil hitungan di layar komputer dan tanda
        // tambah (+) artinya untuk menyambung.
        System.out.println("Segitiga dengan alas  :"+ alas);
        System.out.println("dan tingginya : "+ tinggi);
        System.out.println("mempunyai luas = "+ luas);

    } // akhir dari main()
} // akhir dari class LuasSegitiga
```

Jika dijalankan maka hasilnya akan seperti di bawah ini

```
Segitiga dengan alas : 20.0
dan tingginya : 30.0
mempunyai luas = 300.0
```

Dari program di atas maka yang belum di bahas adalah pemakaian tanda * untuk perkalian dan penggunaan perintah System.out.println(). Selain itu perhatikan bahwa setiap perintah diakhiri dengan titik koma (;) dan program di atas terdiri atas dua (2) blok program yakni blok class dan blok main(). Setiap blok di Java harus diletakkan di antara dua (2) kurung kurawal ({…. }).

Tidak semua operator (lambang untuk operasi matematika) dituliskan sama persis antara Matematika dan Java. Perkalian di Matematika yang dilambangkan dengan x harus ditulis dengan *. Mengapa ? Supaya *compiler* bisa dengan mudah membedakan perkalian dengan variabel yang bernama x. Itulah sebabnya nama variabel tidak boleh diawali dengan angka supaya *compiler* cepat dan mudah memutuskan ketika pertama menjumpai angka maka lambang seterusnya adalah bagian dari bilangan yang harus dibaca. Secara umum operasi Matematika di Java memakai lambang berikut:

Tabel 2.2. Penulisan operator di Java

Operasi	Matematika	Java	Contoh
Perkalian	x	*	luas = 0.5 * alas * tinggi;
Pembagian	/ atau :	/	nilai = (uts1 + uts2 + uas) / 3;
Penjumlahan	+	+	rerata = (x1 + x2) /2 ;
Pengurangan	-	-	beratIdeal = tinggi – 100;
Urutan	( )	( )	jarak = (x1 – (x3 – x4) * 2);

*) Operator lain seperti pangkat, akar dan sebagainya akan dikenalkan kemudian.

Perintah **System.out.println("Segitiga dengan alas :"+ alas);** akan menampilkan tulisan **Segitiga dengan alas :** kemudian diikuti dengan penulisan isi dari variabel **alas.** Perhatikan dengan jeli bahwa apa yang ada di dalam tanda petik dicetak apa adanya oleh komputer sedang jika merujuk ke variabel maka yang dicetak isi variabel tersebut dan bukannya nama variabel tersebut. Oleh karena itu perintah di atas tidak akan mencetak tulisan **alas** di layar komputer. Perhatikan juga bahwa di perintah tersebut ada tanda plus (+) tetapi fungsinya tidak untuk menjumlah melainkan menggandeng tulisan **Segitiga dengan alas :** dengan isi variabel **alas.** Karena isi variabel alas adalah 20 maka yang akhirnya tercetak adalah:

Segitiga dengan alas : 20.0

Perintah **System.out.println("Segitiga dengan alas :"+ alas);** juga menyebabkan perintah pencetakan berikutnya akan ditampilkan di baris sesudahnya. Hal ini karena perintah tersebut memakai **println().** Kalau kita pakai perintah **print()** maka perintah ini tidak mengakibatkan ganti baris seperti sewaktu menekan Enter dalam MSWords.

2.4. Praktikum

1. Tentukan benar salahnya penulisan variabel di Java berikut

   a. tidakMungkin
   b. 20Bagian
   c. serba Bisa
   d. xFaktor
   e. duakali
   f. LimapuluhPersen
   g. complicated
   h. discount20Persen
   i. discount30%
   j. janganBelagu

Untuk nomor 2 - 5, sebutkan kegunaan atau yang dikerjakan pada setiap baris program.

2. Program kali

```
public class HitungKali1 {
   public static void main(String[] args) {
      int a,b;
      int hasil;
      a=3;
      b=4;
      hasil=a*b;
   }
}
```

3. Program kali dan menampilkan hasil :

```
public class HitungKali2 {
   public static void main(String[] args) {
     int a,b;
     int hasil;

     a=3;
     b=4;
     hasil=a*b;
      System.out.println(hasil);
   }
}
```

4. Program kali dan menampilkan hasil dengan layout :

```
 public class HitungKali3 {
    public static void main(String[] args) {
      int a,b;
      int hasil;

      a=3;
      b=4;
      hasil=a*b;

       System.out.println("Hasil dari perkalian 3 dan 4 adalah :"+hasil);
    }
 }
```

5. Program kali dan menampilkan hasil dengan layout :

```
public class HitungKali4 {
    public static void main(String[] args) {
        int a,b;
        int hasil;
        a=3;
        b=4;
        hasil=a*b;
         System.out.println("Hasil dari perkalian "+a+" dan "+b+" adalah :"+hasil);
    }
}
```

6. Program hitung kali dengan perintah print :

Untuk nomor 7-, temukan kesalahan dalam program atau kejanggalan dalam program :

7. Program salahA :

```
public class SalahA {
    public static void main(String[] args) {
        float a;
        a='d';
        System.out.println(a);
    }
}
```

8. Program salahB :

```
public class SalahB {
    public static void main(String[] args)
        int x,y,b;
        a=3
        x=4;
```

```
        y=ax+b;
        System.out.println(“Hasil dari persamaan y adalah :”+y);
    }
}
```

9. Program salahC :

```
public class SalahC {
        int hasilPangkat, nilai;
        nilai=3;
        hasilpangkat=nilai * nilai ;
        System.out.print(“Hasil dari pangkat adalah :”+hasilPangkat);
    }
```