BAB 4

Dasar-Dasar Pemrograman

4.1 Tujuan

Pada bagian ini, kita akan mendiskusikan mengenai bagian dasar pemrograman Java. Kita akan memulai dengan mencoba menjelaskan bagian dasar dari program Hello.java yang telah diperkenalkan pada bab sebelumnya. Kita juga akan mendiskusikan beberapa pedoman cara menulis script atau petunjuk penulisan kode dalam penulisan program yang lebih efektif dan mudah dibaca.

Pada akhir pembahasan, diharapkan pembaca dapat :

- Mengidentifikasi bagian dasar dari program Java
- Membedakan mana yang termasuk ke dalam Java literals, tipe data dasar, tipe variabel, pengidentifikasian dan operator
- Mengembangkan program Java sederhana menggunakan konsep yang dipelajari pada bab ini

4.2 Menganalisa program Java Pertama

Sekarang, kita akan mencoba untuk menganalisa program Java pertama :

```
public class Hello
{
    /**
     * My first java program
     */
    public static void main(String[] args) {
      //menampilkan string ”Hello world” pada layar

        System.out.println("Hello world!");
    }
}
```

Baris pertama kode :

```
public class Hello
```

menandakan nama class yaitu Hello. Dalam Java, semua kode seharusnya ditempatkan di dalam deklarasi class. kita melakukannya dengan menggunakan kata kunci class. Sebagai tambahan, class menggunakan *access specifier* **public**, yang mengindikasikan bahwa class kita mempunyai akses bebas ke class yang lain dari package yang lain pula (package

merupakan kumpulan class-class). Kita akan membahas lebih dalam mengenai package dan *access specifier* pada pembahasan selanjutnya.

Baris berikutnya yaitu yang terdiri atas kurung kurawal { menandakan awal blok. Pada kode ini, kita menempatkan kurung kurawal pada baris selanjutnya setelah deklarasi class, bagaimanapun, kita dapat juga meletakkan kurung kurawal ini setelah baris pertama dari kode yang kita tulis. Jadi, kita dapat menulis kode kita sebagai berikut :

```
public class Hello
{
```

atau

```
public class Hello {
```

Tiga baris selanjutnya menandakan adanya komentar Java. Komentar adalah sesuatu yang digunakan untuk mendokumentasikan setiap bagian dari kode yang ditulis. Komentar bukan merupakan bagian dari program itu sendiri, tetapi digunakan untuk tujuan dokumentasi. Komentar itu sendiri dapat ditambahkan pada kode yang Anda tulis sebagai petunjuk yang dapat membantu proses pembelajaran pemrograman yang baik.

```
/**
* My first java program
*/
```

Komentar dinyatakan dengan tanda “/*” dan “*/”. Segala sesuatu yang ada diantara tanda tersebut diabaikan oleh compiler Java, dan mereka hanya dianggap sebagai komentar.
Baris selanjutnya,

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

atau dapat juga ditulis sebagai berikut,

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

mengindikasikan nama suatu method dalam class **Hello** yang bertindak sebagai **method utama.** Method utama adalah titik awal dari suatu program Java. Semua program kecuali applet yang ditulis dalam bahasa Java dimulai dengan method utama. Yakinkan untuk mengikuti kaidah penulisan tanda yang benar.

Baris selanjutnya juga merupakan komentar,

```
//Menampilkan string "Hello world" pada layar
```

Sekarang kita mempelajari 2 cara untuk membuat komentar. Cara pertama adalah dengan menempatkan komentar dalam /* dan */, dan cara yang lain adalah dengan menuliskan tanda // pada awal komentar

Baris selanjutnya,

```
System.out.println("Hello world!");
```

menampilkan teks “Hello World!” pada layar. Perintah System.out.println(), menampilkan teks yang diapit oleh tanda *double pute* (“ ”) pada layar.

Dua baris terakhir yang terdiri atas dua kurung kurawal digunakan untuk menutup method utama dan masing-masing class secara berurutan.

Petunjuk Penulisan Program:

1. *Program Java yang Anda buat harus selalu diakhiri dengan ekstensi file **.java**.*
2. *Nama File seharusnya sesuai/sama dengan nama class public nya. Sebagai contoh, jika nama class public Anda adalah **Hello**, Anda harus menyimpan file tersebut dengan nama **Hello.java**.*
3. *Anda harus menulis komentar sebagai penjelasan pada kode yang Anda tulis, yaitu komentar yang berisi keterangan mengenai baris perintah pada class atau apa yang dijalankan oleh method yang Anda tulis tersebut.*

4.3. Komentar pada Java

Komentar adalah catatan yang ditulis pada kode dengan tujuan sebagai bahan dokumentasi. Teks tersebut bukan bagian dari program dan tidak mempengaruhi jalannya program.

Java mendukung tiga jenis komentar : C++ style komentar satu baris, C style beberapa baris, dan komentar javadoc khusus

4.3.1. Penulisan Komentar C++ Style

Komentar C++ style diawali dengan //. Semua teks setelah // dianggap sebagai komentar. Sebagi contoh,

```
// This is a C++ style or single line comments
```

4.3.2. Penulisan Komentar C Style

Komentar C-style atau juga disebut komentar beberapa baris diawali dengan /* dan diakhiri dengan */. Semua teks yang ada diantara dua tanda tersebut dianggap sebagai komentar. Tidak seperti komentar C++ style, komentar ini dapat menjangkau beberapa baris. Sebagai contoh,

```
/* this is an example of a
    C style or multiline comments */
```

4.3.3. Komentar Khusus javadoc

Komentar javadoc khusus digunakan untuk men-generate dokumentasi HTML untuk program Java Anda. Anda dapat menciptakan komentar javadoc dengan memulai baris dengan /** dan mengakhirinya dengan */. Seperti Komentar C_style, dapat juga menjangkau beberapa baris. Komentar ini juga dapat terdiri atas tag-tag untuk menambahkan lebih banyak informasi pada komentar Anda. Sebagai contoh,

```
/**
 This is an example of special java doc comments used for \n
 generating an html documentation. It uses tags like:
 @author Florence Balagtas
 @version 1.2
*/
```

4.4. Pernyataan dalam Java dan Blok

Pernyataan adalah satu atau lebih baris kode yang diakhiri dengan semicolon. Sebagai contoh untuk pernyataan tunggal adalah

```
System.out.println(“Hello world”);
```

Blok adalah satu atau lebih pernyataan yang terbentang antara kurung kurawal buka dan kurung kurawal tutup yaitu sekumpulan pernyataan sebagai satu unit kesatuan. Blok pernyataan dapat dikumpulkan akan tetapi tidak secara pasti mempunyai keterkaitan fungsi. Beberapa jumlah spasi kosong diijinkan terdapat didalamnya, sebagai contoh dari suatu blok adalah :

```
public static void main( String[] args ){
      System.out.println("Hello");
      System.out.println("world");
}
```

Petunjuk Penulisan Program:

1. Pada saat pembuatan blok, Anda dapat meletakkan kurung kurawal buka pada baris dengan pernyataan seperti contoh sebagai berikut ,

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

atau Anda dapat meletakkan kurung kurawal pada baris selanjutnya, seperti,

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

2. Anda harus memberi jarak (indent) pernyataan selanjutnya setelah awal dari blok , seperti contoh berikut,

```
public static void main( String[] args ){
      System.out.println("Hello");
      System.out.println("world");
}
```

4.5. Java Identifier

Java Identifier adalah suatu tanda yang mewakili nama-nama variabel, method, class, dsb. Contoh dari Identifier adalah : Hello, main, System, out.

Pendeklarasian Java adalah case-sensitive. Hal ini berarti bahwa Identifier : **H**ello tidak sama dengan **h**ello. Identifier harus dimulai dengan salah satu huruf, underscore "_", atau tanda dollar "$". Hurufnya dapat berupa huruf besar maupun huruf kecil. Karakter selanjutnya dapat menggunakan nomor 0 smpai 9.

Identifier tidak dapat menggunakan kata kunci dalam Java seperti class, public, void, dsb. Selanjutnya kita akan berdiskusi lebih banyak tentang kata kunci dalam Java.

Petunjuk Penulisan Program:

1. *Untuk pemberian nama dari class Java, diberikan huruf kapital untuk huruf pertama pada nama class. Untuk nama method dan variabel, huruf pertama dari kata harus dimulai dengan huruf kecil. Sebagi contoh:*
   ***T**his**I**s**A**n**E**xample**O**f**C**lass**N**ame*
   ***t**his**I**s**A**n**E**xample**O**f**M**ethod**N**ame*

2. *Pada kasus untuk identifier lebih dari satu kata, menggunakan huruf kapital untuk mengindikasikan awal dari kata kecuali kata pertama. Sebagai contoh, charArray, fileNumber, ClassName.*
3. *Hindari menggunakan underscores pada awal identifier seperti _read atau _write.*

4.6. Keyword dalam Java

Kata kunci adalah identifier yang telah dipesan untuk didefinisikan sebelumnya oleh Java untuk tujuan tertentu. Anda tidak dapat menggunakan keyword sebagai nama variabel, class, method Anda, dsb. Berikut ini adalah daftar dari kata kunci dalam Java (Java Keywords).

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

* not used
** added in 1.2
*** added in 1.4
**** added in 5.0

Gambar1: Java Key Word

Kita akan berdiskusi tentang semua arti dari masing-masing kata kunci dan bagaimana mereka digunakan dalam proses penulisan program Java.

Catatan: true, false, dan null bukan termasuk kata kunci akan tetapi mereka termasuk kata-kata khusus, jadi Anda tidak dapat menggunakan mereka sebagai nama variabel pada program Anda.

4.7. Java Literals

Literals adalah tanda bahwa tidak terjadi perubahan atau konstan. Macam-macam literals dalam Java adalah : *Integer Literals*, *Floating-Point Literals*, *Boolean Literals*, *Character Literals* dan *String Literals*.

4.7.1. Integer Literals

Integer literals dibedakan dalam beberapa format yang berbeda: **desimal** (berbasis 10), **heksadesimal** (berbasis 16), and **oktal** (berbasis 8). Dalam penggunaan tipe data integer pada program, kita harus mengikuti aturan penggunaan beberapa notasi khusus.

Untuk angka desimal, kita tidak memerlukan notasi khusus. Kita hanya menulis angka desimal seperti apa adanya. untuk angka heksadesimal, hal itu harus ditandai oleh “0x” atau “0X”. untuk oktal, ditandai oleh“0”.

Sebagai contoh, mewakili angka **12**. Penulisan dalam bentuk desimalnya adalah **12**, Sementara dalam heksadesimal, menjadi **0xC**, dan dalam oktal, nilai tersebut sama dengan **014**.

Default tipe data untuk integer literals adalah **int**. Int adalah signed 32-bit value. Pada kasus-kasus tertentu Anda dapat berharap untuk memaksa integer literal untuk menjadi tipe data **long** dengan menambahkan karakter “l” or “L”. tipe data long ditandai oleh ditampilkannya data dalam 64-bit. Kita akan membahas mengenai tipe data pada kesempatan selanjutnya.

4.7.2. Floating-Point Literals

Floating point literals mewakili bentuk desimal dengan bagian yang terpisah. Sebagai contoh adalah 3.1415. *Floating point literals* dapat dinyatakan dalam notasi standard atau scientific. Sebagai contoh, 583.45 dinyatakan dalam notasi standard, Sementara 5.8345e2 dinyatakan dalam notasi scientific.

Default *Floating point literals* mempunyai tipe data **double** yang dinyatakan dalam 64-bit. Untuk menggunakan ketelitian yang lebih kecil (32-bit) **float**, hanya dengan menambahkan karakter “f” atau “F”.

4.7.3. Boolean Literals

Boolean literals hanya memiliki dua nilai, true atau false.

4.7.4. Character Literals

Character Literals diwakili oleh karakter single Unicode. Karakter Unicode adalah 16-bit character set yang menggantikan 8-bit ASCII character set. Unicode memungkinkan pengunaan simbol dan karakter khusus dari bahasa lain.

Untuk menggunakan *character literals*, karakter tersebut di dalam tanda *single pute* (' ') (single quote delimiters). Sebagai contoh huruf a, diwakili sebagai **‘a’**.

Untuk menggunakan karakter khusus seperti karakter baris baru, *backslash* digunakan diikuti dengan karakter kode. Sebagai contoh, ‘\n’ untuk karakter baris baru atau ganti baris, ‘\r’ untuk menyatakan nilai balik (carriage return), ‘\b’ untuk backspace.

4.7.5. String Literals

String literals mewakili beberapa karakter dan dinyatakan dalam tanda double pute(“ ”)(double quotes). Sebagai contoh *string literal* adalah, **“Hello World”.**

4.8. Tipe Data Primitif

Bahasa pemrograman Java mendefinisikan delapan tipe data primitif. Mereka diantaranya adalah boolean (untuk bentuk logika), char (untuk bentuk tekstual), byte, short, int, long (integral), double and float (floating point).

4.8.1. logika - boolean

Tipe data boolean diwakili oleh dua pernyataan : true dan false. Sebagai contoh adalah,

```
boolean result = true;
```

Contoh yang ditunjukkan diatas, mendeklarasikan variabel yang dinamai **result** sebagai tipe data **boolean** dan memberinya nilai **true**.

4.8.2. teksual – char

Tipe data character (char), diwakili oleh karakter single Unicode. Tipe data ini harus memiliki ciri berada dalam tanda *single quotes*(' '). Sebagai contoh,

```
'a'   //Huruf a
'\t'  //A tab
```

Untuk menampilkan karakter khusus seperti ' (single quotes) atau " (double quotes), menggunakan karakter escape \. Sebagai contoh,

```
'\''  //untuk single quotes
'\"'  //untuk double quotes
```

Meskipun String bukan merupakan tipe data primitif (namun merupakan suatu Class),kita akan memperkenalkan mengenai pada bagian ini. String mewakili tipe data yang terdiri atas beberapa karakter. Mereka tidak **termasuk tipe data primitif, melainkan suatu class.** Mereka memiliki literal yang terdapat diantara tanda double quotes("").

Sebagai contoh,

```
String message="Hello world!"
```

4.8.3. Integral – byte, short, int & long

Tipe data integral dalam Java menggunakan tiga bentuk- yaitu desimal, oktal atau heksadesimal. Contohnya,

```
2     //nilai desimal 2
077   //angka 0 pada awal pernyataan mengindikasikan nilai oktal
0xBACC      //karakter 0x mengindikasikan nilai heksadesimal
```

Tipe-tipe integral memiliki default tipe data yaitu int. Anda dapat merubahnya ke bentuk long dengan menambahkan huruf l atau L. Tipe data integral memiliki range sebagai berikut:

Integer Length	***Name or Type***	***Range***
8 bits	byte	-2^7 to 2^7-1
16 bits	short	-2^{15} to $2^{15}-1$
32 bits	int	-2^{31} to $2^{31}-1$
64 bits	long	-2^{63} to $2^{63}-1$

Tabel 1: Tipe-tipe integral dan range-nya

Petunjuk Penulisan Program:
Dalam mendefinisikan suatu nilai long, lowercase L tidak dianjurkan karena sangat sulit untuk membedakan dari digit 1.

4.8.4. Floating Point – float dan double

Tipe Floating point memiliki double sebagai default tipe datanya. Floating-point literal termasuk salah satunya desimal point atau salah satu dari pilihan berikut ini,

```
E or e //(add exponential value)
F or f //(float)
D or d //(double)
```

Contohnya adalah,

```
3.14              //nilai floating-point sederhana (a double)
6.02E23           //A nilai floating-point yang besar
2.718F       //A nilai float size sederhana
123.4E+306D  //A nilai double yang besar dengan nilai redundant D
```

Pada contoh yang ditunjukkan diatas, 23 setelah E pada contoh kedua bernilai positif. Contoh tersebut sama dengan 6.02E+23. Tipe data Floating-point memiliki range sebagai berikut:

Panjang Float	***Nama atau Tipe***	***Range***
32 bits	float	-2^{31} to $2^{31}-1$
64 bits	double	-2^{63} to $2^{63}-1$

Table 2: Tipe Floating point dan range nya

4.9. Variabel

Variabel adalah item yang digunakan data untuk menyimpan pernyataan objek.

Variabel memiliki **tipe data** dan **nama**. Tipe data menandakan tipe nilai yang dapat dibentuk oleh variabel itu sendiri. **Nama variabel** harus mengikuti aturan untuk identifier.

4.9.1. Deklarasi dan Inisialisasi Variabel

Untuk deklarasi variabel adalah sebagai berikut,

```
<data tipe> <name> [=initial value];
```

Catatan: Nilainya berada diantara < > adalah nilai yang disyaratkan, sementara nilai dalam tanda [] bersifat optional.

Berikut ini adalah contoh program yang mendeklarasikan dan menginisialisasi beberapa variabel,

```
public class VariableSamples
{
    public static void main( String[] args ){
        //deklarasi tipe data dengan nama variable
        // result dan tipe data boolean
        boolean  result;

        //deklarasi tipe data dengan nama variabel
        // option dan tipe data char
        char     option;
        option = 'C';  //menandai 'C' sebagai option

        //deklarasi tipe data dengan nama variabel
        //grade, double tipe data dan telah  di inisialisasi
        //to 0.0
        double   grade = 0.0;
    }
}
```

Petunjuk Penulisan Program:

1. *Sangat baik untuk menginisialisasi variabel yang Anda buat seperti Anda mendeklarasikannya.*
2. *Gunakan nama yang bersifat menggambarkan deskriptif untuk variabel yang Anda buat, jika Anda ingin mempunyai variabel yang terdiri atas nilai siswa, beri nama dengan nama* ***grade*** *dan jangan hanya beberapa huruf random yang Anda pilih.*
3. *Deklarasikan satu variabel tiap baris kode. Sebagai contoh , deklarasi variabel adalah sebagai berikut,*

```
double exam=0;
double quiz=10;
```

double grade = 0;

Bentuk yang lebih disukai ketika melakukan deklarasi adalah,

double exam=0, quiz=10, grade=0;

4.9.2. Menampilkan Data Variabel

Untuk mengeluarkan nilai dari variabel yang diinginkan, kita dapat menggunakan perintah sebagai berikut,

```
System.out.println()
System.out.print()
```

Berikut ini adalah contoh program,

```
public class OutputVariable
{
    public static void main( String[] args ){
        int value = 10;
        char x;
        x = 'A';

        System.out.println( value );
        System.out.println( "The value of x=" + x );
    }
}
```

Program tersebut akan mengeluarkan teks berikut pada layar,

```
10
The value of x=A
```

4.9.3. System.out.println() vs. System.out.print()

Apa yang membedakan diantara perintah System.out.println() and System.out.print()? Yang pertama menambahkan baris baru pada akhir data untuk dikeluarkan, sementara selanjutnya tidak.

Perhatikan pernyataan tersebut,

```
System.out.print("Hello ");
System.out.print("world!");
```

Pernyataan tersebut akan menghasilkan output berikut ini pada layar,

```
Hello world!
```

Sekarang perthatikan pernyataan berikut,

```
System.out.println("Hello ");
System.out.println("world!");
```

Pernyataan ini akan menghasilkan output sebagai berikut pada layar,

```
Hello
world!
```

4.9.4. Variabel Reference dan Variabel Primitif

Sekarang kita akan membedakan dua tipe variabel yang dimiliki oleh program Java. Ada **variabel reference** dan **variabel primitif** .

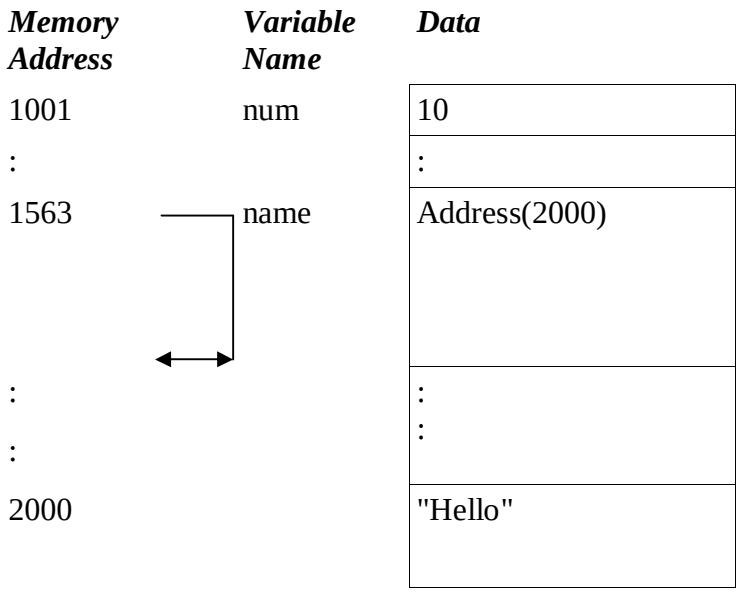
Variabel primitif adalah variabel dengan tipe data primitif. Mereka menyimpan data dalam lokasi memori yang sebenarnya dimana variabel tersebut berada.

Variabel Reference adalah variabel yang menyimpan alamat dalam lokasi memori. Yang menunjuk ke lokasi memori dimana data sebenarnya berada. Ketika Anda mendeklarasikan variabel pada class tertentu, Anda sebenarnya mendeklarasikan reference variable dalam bentuk objek dalam classnya tersebut.

Sebagai contoh, Apabila kita mempunyai dua variabel dengan tipe data int dan String.

```
int num = 10;
String name = "Hello"
```

Dimisalkan ilustrasi yang ditunjukkan dibawah ini adalah memori yang ada pada komputer Anda, dimana Anda memiliki alamat dari setiap sel memorinya, nama variabel dan datanya terbentuk sebagai berikut.



Seperti yang dapat Anda lihat, untuk variable primitif num, datanya berada dalam lokasi dimana variabel berada. Untuk reference variable name, variabel hanya menunjuk alamat dimana data tersebut benar-benar ada.

4.10 Operator

Dalam Java, ada beberapa tipe operator. Ada operator aritmatika, operator relasi, operator logika, dan operator kondisi. Operator ini mengikuti bermacam-macam prioritas yang pasti sehingga compilernya akan tahu yang mana operator untuk dijalankan lebih dulu dalam kasus beberapa operator yang dipakai bersama-sama dalam satu pernyataan.

4.10.1 Operator Aritmatika

Berikut ini adalah dasar operator aritmatika yang dapat digunakan untuk membuat suatu program Java,

Operator	***Penggunaan***	***Keterangan***
+	op1 + op2	Menambahkan op1 dengan op2
*	op1 * op2	Mengalikan op1 dengan op2
/	op1 / op2	Membagi op1 dengan op2
%	op1 % op2	Menghitung sisa dari pembagian op1 dengan op2
-	op1 - op2	Mengurangkan op2 dari op1

Tabel 3: Operator Aritmatika dan fungsi-fungsinya

Berikut ini adalah contoh program dalam penggunaan operator-operator ini :

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

        //sedikit angka
        int i = 37;
        int j = 42;
        double x = 27.475;
        double y = 7.22;
        System.out.println("Variable values...");
        System.out.println("    i = " + i);
        System.out.println("    j = " + j);
        System.out.println("    x = " + x);
        System.out.println("    y = " + y); //penjumlahan angka
        System.out.println("Adding...");
        System.out.println("    i + j = " + (i + j));
        System.out.println("    x + y = " + (x + y));

        //pengurangan angka
        System.out.println("Subtracting...");
        System.out.println("    i - j = " + (i - j));
        System.out.println("    x - y = " + (x - y));

        //perkalian angka
        System.out.println("Multiplying...");
        System.out.println("    i * j = " + (i * j));
        System.out.println("    x * y = " + (x * y));
       //pembagian angka
        System.out.println("Dividing...");
        System.out.println("    i / j = " + (i / j));
        System.out.println("    x / y = " + (x / y));

        //menghitung hasil modulus dari pembagian
        System.out.println("Computing the remainder...");
        System.out.println("    i % j = " + (i % j));
        System.out.println("    x % y = " + (x % y));

        //tipe penggabungan
        System.out.println("Mixing tipes...");
        System.out.println("    j + y = " + (j + y));
        System.out.println("    i * x = " + (i * x));
    }
}
```

Berikut ini adalah output program,

```
Variable values...
    i = 37
    j = 42
```

```
    x = 27.475
    y = 7.22
    i + j = 79
Adding...
    x + y = 34.695
Subtracting...
    i - j = -5
    x - y = 20.255
Multiplying...
    i * j = 1554
    x * y = 198.37
Dividing...
    i / j = 0
    x / y = 3.8054
Computing the remainder...
    i % j = 37
    x % y = 5.815
Mixing tipes...
    j + y = 49.22
    i * x = 1016.58
```

Catatan: Ketika integer dan floating-point number digunakan sebagai operand untuk operasi aritmatika tunggal, hasilnya berupa floating point. Integer adalah converter secara implisit ke bentuk angka floating-point sebelum operasi berperan mengambil tempat.

4.10.2. Operator Increment dan Decrement

Dari sisi operator dasar aritmatika, Java juga terdiri atas operator *unary increment* (++) dan operator *unary decrement* (--). operator increment dan decrement menambah dan mengurangi nilai yang tersimpan dalam bentuk variabel angka terhadap nilai 1.

Sebagai contoh, pernyataan,

```
count = count + 1;      //increment nilai count dengan nilai 1
```

pernyataan tersebut sama dengan,

```
count++;
```

Operator	***Penggunaan***	***Keterangan***
++	op++	Menambahkan nilai 1 pada op; mengevaluasi nilai op sebelum diincrementasi/ditambahkan
++	++op	Menambahkan nilai 1 pada op; mengevaluasi nilai op setelah diincrementasi/ditambahkan
--	op--	Mengurangkan nilai 1 pada op; mengevaluasi nilai op sebelum didecrementasi/dikurangkan
--	--op	Mengurangkan nilai 1 pada op; mengevaluasi nilai op setelah didecrementasi/dikurangkan

Tabel 4: operator Increment dan Decrement

Operator increment dan decrement dapat ditempatkan sebelum atau sesudah operand.

Ketika digunakan sebelum operand, akan menyebabkan variabel diincrement atau didecrement dengan nilai 1, dan kemudian nilai baru digunakan dalam pernyataan dimana dia ditambahkan. Sebagai contoh,

```
int i = 10,
int j = 3;
int k = 0;

k = ++j + i; //akan menghasilkan k = 4+10 = 14
```

Ketika operator increment dan decrement ditempatkan setelah operand, nilai variabel yang lama akan digunakan lebih dulu dioperasikan lebih dulu terhadap pernyataan dimana dia ditambahkan. Sebagai contoh,

```
int i = 10,
int j = 3;
int k = 0;

k = j++ + i; //akan menghasilkan k = 3+10 = 13
```

Petunjuk Penulisan Program:
Selalu membuat pernyataan yang mengandung operator increment dan decrement untuk tetap dipahami secara mudah dan sederhana.

4.10.3 Operator Relasi

Operator Relasi membandingkan dua nilai dan menentukan keterhubungan diantara nilai-nilai tersebut. Hasil keluarannya berupa **nilai boolean** yaitu true atau false.

Operator	***Penggunaan***	***Keterangan***
>	op1 > op2	op1 lebih besar dari op2
>=	op1 >= op2	op1 lebih besar dari atau sama dengan op2
<	op1 < op2	op1 kurang dari op2
<=	op1 <= op2	op1 kurang dari atau sama dengan op2
==	op1 == op2	op1 sama dengan op2
!=	op1 != op2	op1 tidak sama dengan op2

Table 5: Operator Relasi

Berikut ini adalah contoh program yang menggunakan operator Relasi,

```
public class RelasiDemo
 {
     public static void main(String[] args) {
     //beberapa nilai
         int i = 37;
         int j = 42;
         int k = 42;
         System.out.println("Nilai variabel...");
         System.out.println("    i = " + i);
         System.out.println("    j = " + j);
         System.out.println("    k = " + k);

    //lebih besar dari
         System.out.println("Lebih besar dari...");
         System.out.println("    i > j = " + (i > j)); //false
         System.out.println("    j > i = " + (j > i)); //true
         System.out.println("    k > j = " + (k > j)); //false

    //lebih besar atau sama dengan
         System.out.println("Lebih besar dari atau sama dengan...");
         System.out.println("    i >= j = " + (i >= j)); //false
         System.out.println("    j >= i = " + (j >= i)); //true
         System.out.println("    k >= j = " + (k >= j)); //true

    //lebih kecil dari
         System.out.println("Lebih kecil dari...");
         System.out.println("    i < j = " + (i < j)); //true
         System.out.println("    j < i = " + (j < i)); //false
         System.out.println("    k < j = " + (k < j)); //false
    //lebih kecil atau sama dengan
         System.out.println("Lebih kecil dari atau sama dengan...");
         System.out.println("    i <= j = " + (i <= j)); //true
         System.out.println("    j <= i = " + (j <= i)); //false
         System.out.println("    k <= j = " + (k <= j)); //true

    //sama dengan
         System.out.println("Sama dengan...");
         System.out.println("    i == j = " + (i == j)); //false
         System.out.println("    k == j = " + (k == j)); //true

    //tidak sama dengan
         System.out.println("Tidak sama dengan...");
         System.out.println("    i != j = " + (i != j)); //true
         System.out.println("    k != j = " + (k != j)); //false

     }
 }
```

Berikut adalah hasil keluaran dari program ini :

```
Nilai variabel...
    i = 37
    j = 42
    k = 42
Lebih besar dari...
    i > j = false
    j > i = true
    k > j = false
Lebih besar dari atau sama dengan...
    i >= j = false
    j >= i = true
    k >= j = true
Lebih kecil dari...
    i < j = true
    j < i = false
    k < j = false
Lebih kecil dari atau sama dengan...
    i <= j = true
    j <= i = false
    k <= j = true
Sama dengan...
    i == j = false
    k == j = true
Tidak sama dengan...
    i != j = true
    k != j = false
```

4.10.4 Operator logika

Operator logika memiliki satu atau lebih operand boolean yang menghasilkan nilai boolean. Terdapat enam operator logika yaitu: && (logika AND), & (boolean logika AND), || (logika OR), | (boolean logika inclusive OR), ^ (boolean logika exclusive OR), dan ! (logika NOT).

Pernyataan dasar untuk operasi logika adalah,

```
x1 op x2
```

Dimana x1, x2 dapat menjadi pernyataan boolean. Variabel atau konstanta, dan op adalah salah satu dari operator &&, &, ||, | atau ^. Tabel kebenaran yang akan ditunjukkan selanjutnya, merupakan kesimpulan dari hasil dari setiap operasi untuk semua kombinasi yang mungkin dari x1 dan x2.

4.10.4.1 && (logika AND) dan & (boolean logika AND)

Berikut ini adalah tabel kebenaran untuk && dan &,

x1	*x2*	*Hasil*
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

Tabel 6: Tabel Kebenaran untuk & dan &&

Perbedaan dasar antara operator && dan & adalah bahwa && mensupports **short-circuit evaluations** (atau evaluasi perbagian), sementara operator & tidak. Apa arti dari pernyataan tersebut?

Diberikan suatu pernyataan,

```
exp1 && exp2
```

&& akan mengevaluasi pernyataan exp1, dan segera mengembalikan nilai false dan menyatakan bahwa exp1 bernilai false. Jika exp1 bernilai false, operator tidak akan pernah mengevaluasi exp2 karena hasil operasi operator akan menjadi false tanpa memperhatikan nilai dari exp2. Sebaliknya, operator & selalu mengevaluasi kedua nilai dari exp1 dan exp2 sebelum mengembalikan suatu nilai jawaban.

Berikut ini adalah suatu contoh source code yang menggunakan logika dan boolean AND,

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

            int   i     = 0;
            int   j     = 10;
            boolean test= false;

            //demonstrasi &&
            test = (i > 10) && (j++ > 9);
            System.out.println(i);
            System.out.println(j);
            System.out.println(test);

            //demonstrasi &
            test = (i > 10) & (j++ > 9);
            System.out.println(i);
            System.out.println(j);
            System.out.println(test);
      }
}
```

The output of the program is,

```
0
10
false
0
11
false
```

Catatan, bahwa j++ pada baris yang mengandung operator && tidak dievaluasi sejak pernyataan pertama (i>10) yaitu telah bernilai sama dengan false.

4.10.4.2 || (logika OR) dan | (boolean logika inclusive OR)

Berikut ini adalah tabel kebenaran untuk || dan |,

x1	*x2*	*Hasil*
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Tabel 7: Table Kebenaran untuk | dan ||

Perbedaan dasar antara operator || dan | adalah bahwa || mendukung short-circuit evaluations (atau proses evaluasi sebagian), sementara | tidak. Apa maksud dari pernyataan tersebut?

diberikan suatu pernyataan,

```
exp1 || exp2
```

|| akan mengevaluasi pernyataan exp1, dan segera mengembalikan nilai true dan menyatakan bahwa exp1 bernilai true. Jika exp1 bernilai true, operator tidak akan pernah mengevaluasi exp2 karena hasil dari operasi operator akan bernilai true tanpa memperhatikan nilai dari exp2. Sebaliknya,operator | selalu mengevaluasi kedua nilai dari exp1 and exp2 sebelum mengembalikan suatu jawaban suatu nilai.

Berikut ini sebuah contoh source code yang menggunakan operator logika dan boolean OR,

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

            int   i     = 0;
            int   j     = 10;
            boolean test= false;

            //demonstrasi ||
            test = (i < 10) || (j++ > 9);
            System.out.println(i);
            System.out.println(j);
            System.out.println(test);

            //demonstrasi |
            test = (i < 10) | (j++ > 9);
            System.out.println(i);
            System.out.println(j);
            System.out.println(test);
      }
}
```

Hasil keluaran dari program ini adalah,

```
0
10
true
0
11
true
```

Catatan, bahwa j++ pada baris yang terdiri atas operator || tidak dievaluasi sejak pernyataan pertama (i<10) yaitu telah bernilai sama dengan true.

4.10.4.3 ^ (boolean logika ExclusiveOR)

Berikut ini adalah tabel kebenaran untuk ^,

x1	*x2*	*Hasil*
TRUE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Tabel 8: Tabel kebenaran untuk ^

Hasil operasi operator exclusive OR adalah TRUE, jika dan hanya jika satu operand bernilai TRUE dan yang lain bernilai False. Catatan jika kedua operand harus selalu dievaluasi untuk menjumlahkan hasil dari suatu exclusive OR.

Berikut ini adalah contoh source code yang menggunakan operator logika exclusive OR,

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

            boolean val1 = true;
            boolean val2 = true;
            System.out.println(val1 ^ val2);

            val1 = false;
            val2 = true;
            System.out.println(val1 ^ val2);

            val1 = false;
            val2 = false;
            System.out.println(val1 ^ val2);

            val1 = true;
            val2 = false;
            System.out.println(val1 ^ val2);
      }
}
```

Hasil keluaran program tersebut adalah,

```
false
true
false
true
```

4.10.4.4 ! (logika NOT)

Logika NOT digunakan dalam satu argumen, dimana argumen tersebut dapat menjadi suatu pernyataan, variabel atau konstanta. Berikut ini adalah tabel kebenaran untuk operator not!,

x1	***Hasil***
TRUE	FALSE
FALSE	TRUE

Tabel 9: Tabel Kebenaran untuk !

Berikut ini adalah contoh source code yang menggunakan operator logika NOT,

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

            boolean val1 = true;
            boolean val2 = false;
            System.out.println(!val1);
            System.out.println(!val2);
      }
}
```

Hasil keluaran program adalah sebagai berikut,

```
false
true
```

4.10.5 Operator Kondisi(?:)

Operator kondisi **?:** adalah operator ternary. Berarti bahwa operator ini membawa tiga argumen yang membentuk suatu ekspresi bersyarat. Struktur pernyataan yang menggunakan operator kondisi adalah,

```
exp1?exp2:exp3
```

Dimana nilai exp1 adalah suatu pernyataan boolean yang memiliki hasil yang salah satunya harus berupa nilai true atau false.

Jika exp1 bernilai true, exp2 merupakan hasil operasi. Jika bernilai false, kemudian exp3 merupakan hasil operasinya.

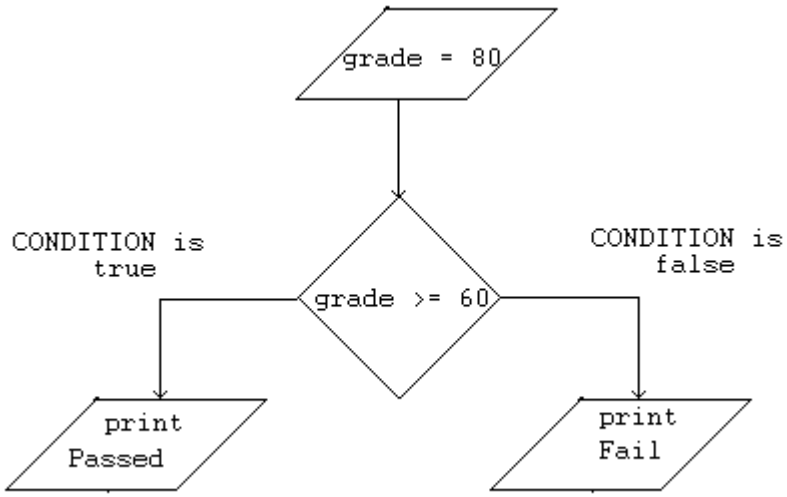
Sebagai contoh, diberikan code sebagai berikut,

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

            String      status = "";
            int   grade = 80;

            //mendapatkan status pelajar
            status = (grade >= 60)?"Passed":"Fail";

            //print status
            System.out.println( status );
      }
}
```

Hasil keluaran dari program ini akan menjadi,

```
Passed
```

Berikut ini adalah flowchart yang menggambarkan bagaimana operator ?: bekerja,



Gambar 2: Flowchart

Berikut ini adalah program lain yang menggunakan operator ?: ,

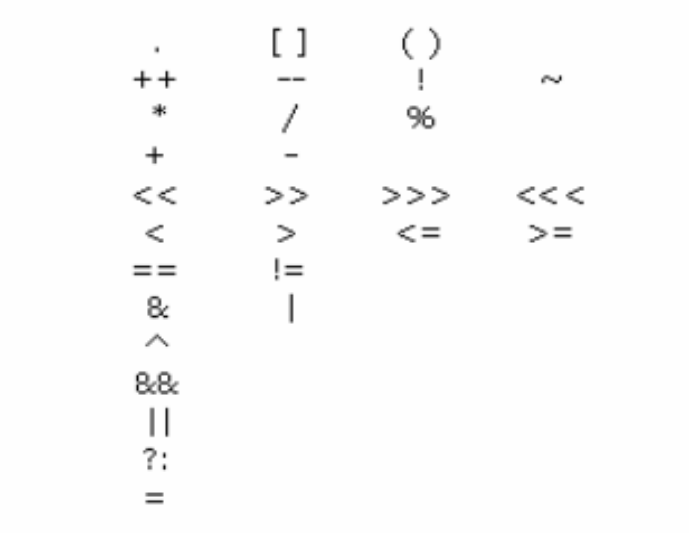
```
class ConditionalOperator
{
      public static void main( String[] args ){

            int   score = 0;
            char  answer = 'a';

            score = (answer == 'a') ? 10 : 0;
            System.out.println("Score = " + score );
      }
}
```

Hasil keluaran program adalah,

```
Score = 10
```

4.10.6 Operator Precedence

Operator precedence didefinisikan sebagai perintah yang dilakukan compiler ketika melakukan evaluasi terhadap operator, untuk mengajukan perintah dengan hasil yang tidak ambigu/ hasil yag jelas.



Gambar3: Operator Precedence

Diberikan pernyataan yang membingungkan,

```
6%2*5+4/2+88-10
```

Kita dapat menuliskan kembali pernyataan diatas dan menambahkan beberapa tanda kurung terhadap operator precedence,

```
((6%2)*5)+(4/2)+88-10;
```

Petunjuk Penulisan Program:

Untuk menghindari kebingungan dalam evaluasi operasi matematika, buatlah pernyataan sesederhana mungkin dan gunakan bantuan tanda kurung.

4.11 Latihan

4.11.1 Mendeklarasikan dan mencetak variabel

Diberikan tabel dibawah ini, deklarasikan variabel yang terdapat didalamnya dengan tipe data yang sesuai dan berikan nilai inisialisasi. Tampilkan hasil outputnya yaitu nama variabel dan nilainya.

Nama Variabel	*Tipe Data*	*Nilai Awal*
number	integer	10
letter	character	a
result	boolean	true
str	String	hello

Berikut ini merupakan tampilan yang diharapkan sebagai hasil eksekusi program,

```
Number = 10
letter = a
result = true
str = hello
```

4.11.2. Mendapatkan nilai rata-rata dari tiga angka

Buatlah program yang menghasilkan output nilai rata-rata dari tiga angka. Nilai dari masing-masing tiga angka tersebut adalah 10, 20 dan 45. Tampilan Output yang diharapkan adalah,

```
number 1 = 10
number 2 = 20
number 3 = 45
Rata-rata = 25
```

4.11.3. Menampilkan nilai terbesar

Diberikan tiga angka, tuliskan program yang menghasilkan output angka dengan nilai terbesar diantara tiga angka tersebut. Gunakan operator kondisi ?: yang telah kita pelajari sebelumnya (**PETUNJUK**: Anda akan perlu menggunakan dua set operator **?:** untuk memecahkan permasalahan ini). Sebagai contoh, diberikan angka 10, 23 dan 5, Program Anda akan menghasilkan output,

```
number 1 = 10
number 2 = 23
number 3 = 5
Nilai tertingginya adalah angka = 23
```

4.11.4. Operator precedence

Diberikan pernyataan berikut ini, tulis kembali soal tersebut dengan menambahkan tanda kurung pada urutan sesuai dengan bagaimana pernyataan tersebut akan dievaluasi.

1. a / b ^ c ^ d – e + f – g * h + i
2. 3 * 10 *2 / 15 – 2 + 4 ^ 2 ^ 2
3. r ^ s * t / u – v + w ^ x – y+ +