BAB 1

Dasar-Dasar Pemrograman

1.1 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* lebih efektif dan mudah dibaca.

Pada akhir pembelajaran ini, pelajar seharusnya 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 pembelajaran pada bab ini.
- Menganalisa *program java* pertama saya

1.2 Menganalisa program Java pertama saya

Sekarang, kita akan berusaha untuk menganalisa *program java* pertama anda :

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

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

Baris pertama kode :

```
public class Hello
```

mengindikasikan nama *class* yaitu *Hello*. Pada *java* semua kode seharusnya ditempatkan didalam deklarasi *class*. Kita melakukannya dengan menggunakan kata kunci *class*. Sebagai tambahan, *class* menggunakan akses khusus ***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 akses khusus pada pembahasan selanjutnya.

Baris berikutnya yaitu yang terdiri atas kurung kurawal { mengindikasikan 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 mengindikasikan adanya komentar dalam bahasa *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 diindikasikan oleh 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,

```
//prints the string "Hello world" on screen
```

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.

Pedoman 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.*

1.3. Komentar pada Java

Komentar adalah catatan yang ditulis pada kode dengan tujuan sebagai bahan dokumentasi. teks ini 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

1.3.1. Penulisan Komentar pada C++

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

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

1.3.2. Penulisan Komentar pada C

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, itu dapat menjangkau beberapa baris. Sebagai contoh,

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

1.3.3. Komentar Khusus javadoc

Komentar javadoc khusus digunakan untuk generalisasi dokumentasi HTML untuk *program java* anda. Anda dapat menciptakan komentar javadoc dengan memulai baris dengan /** dan mengakhirinya dengan */. Seperti Komentar C_style, ini dapat juga menjangkau beberapa baris. 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
*/
```

1.4. Pernyataan dalam Java dan Block

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

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

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

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

Pedoman Penulisan Program:

1. Pada saat pembuatan block, 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 block , seperti contoh berikut,

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

1.5. Java Identifier

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

Pendeklarasian *Java* adalah *case-sensitive*. Hal ini berarti bahwa pengidentifikasi : ***H****ello* tidak sama dengan ***h****ello*. Pengidentifikasi 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.

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

Pedoman Penulisan Program:

1. *Untuk pemberian nama dai 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 pengidentifikasi labih 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 pengidentifikasian seperti _read atau _write.*

1.6. Keyword dalam Java

Keyword adalah pengidentifikasi yang telah dipesan untuk didefinisikan sebelumnya oleh *java* untuk tujuan tertentu. Anda tidak dapat menggunakan *keyword* sebagai nama variabel anda, *class, method* dsb. Berikut ini adalah daftar dai 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.

1.7. 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*.

1.7.1. Literals Integer

literals Integer 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, menjdi **0xC**, dan dalam oktal, nilai tersebut ekuivalen dengan **014**.

Default tipe data untuk *integer literals* adalah ***int***. *Int* ditandai dengan ditampilkannya dalam 32-bit. 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.

1.7.2. 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”.

1.7.3. Boolean Literals

Boolean literals hanya memiliki dua nilai, *true* atau *false*.

1.7.4. 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 *symbol* dan karakter khusus dari bahasa lain.

Untuk menggunakan *character literal*, karakter tersebut di dalam tanda *single quote* (' ') (*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*.

1.7.5. String Literals

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

1.8. 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* dan *float (floating point)*.

1.8.1. 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***.

1.8.2. 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!”
```

1.8.3. Integral – byte, short, int & long

tipe data yang terintegrasi dalam *java* menggunakan tiga bentuk- yaitu desimal, oktal atau heksadesimal.
Sebagai contoh,

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

tipe-tipe terintegrasi memiliki *default* tipe data yaitu *int*. Anda dapat merubahnya ke bentuk *long* dengan menambahkan huruf l atau L. tipe data terintegrasi 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 $2^{63}-1$

Table 1: Integral types and their ranges

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

1.8.4. Floating Point – float dan double

Tipe *Floating point* memiliki *double* sebagai *default* tipe datanya. *Floating-point literal* terdiri atas 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 ekuivalen dengan 6.02E+23.tipe data *Floating-point* memiliki *range* sebagai berikut:

Float Length	***Name or Type***	***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

1.9. Variabel

Variabel adalah *item* yang digunakan data untuk menyimpan pernyataan *object*.

Variabel memiliki tipe data dan nama. tipe data mengindikasikan tipe dari nilai yang dapat dibentuk oleh variabel itu sendiri. nama variabel harus mengikuti aturan untuk pengidentifikasian.

1.9.1. 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 a 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. *Hal ini selalu baik untuk menginisialisasi variabel yang anda buat seperti anda mendeklarasikannya.*
2. *Gunakan nama yang bersifat menggambarkan object untuk variabel yang anda buat., jika anda ingin mempunyai variabel yang terdiri atas nilai pelajar, beri nama dengan nama nilai 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;

1.9.2. 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
```

1.9.3. 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!
```

1.9.4. Reference Variables vs. primitif Variables

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

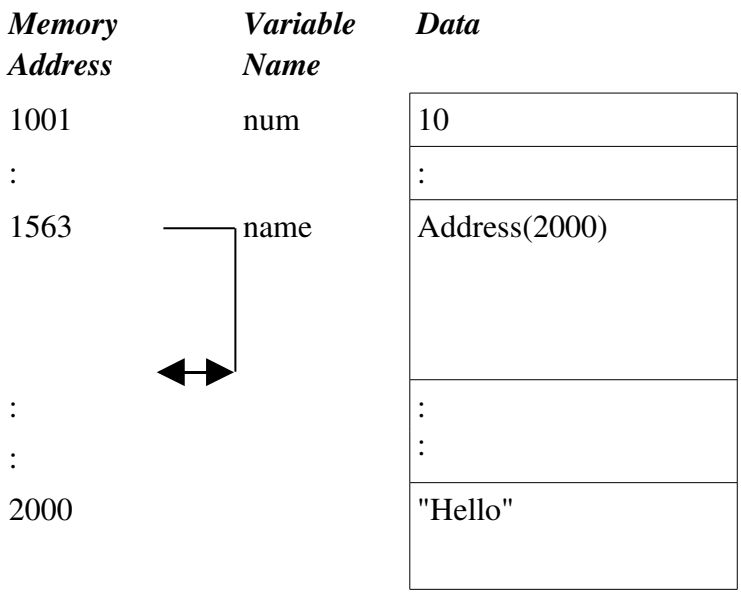
Variabel primitif adalah variabel dengan tipe data primitif. Mereka menyimpan data dalam lokasi memori yang sebenarnya dimana variabel tersebut berada.

Variabel *Reference* adalah variabel yang menyimpan alamat dalam lokasi memori. Yang menunjuk ke lokasi memori dimana data sebenarnya berada. Ketika anda mendeklarasi variabel pada *class* tertentu, anda sebenarnya mendeklarasikan *reference* variable dalam bentuk *object* dalam *class*nya tersebut.

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

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

Apabila, 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 primitif variabel num, datanya berada dalam lokasi dimana variabel berada. Untuk *reference* variabel *name*, variabel hanya menunjuk alamat dimana data tersebut benar-benar ada.

1.10 Operators

Dalam *Java*, ada beberapa tipe operator. Ada operator arithmatika, operator relasi, operator logika, dan operator kondisi. Operator ini mengikuti macam-macam prioritas yang pasti jadi *compiler*nya akan tahu yang mana operator untuk dijalankan lebih dulu dalam kasus beberapa operator yang dipakai bersama-sama dalam satu pernyataan.

1.10.1 Operator Aritmatika

Berikut ini adalah dasar operator arithmatika yang dapat digunakan untuk membuat suatu *program java*,

Operator	***Use***	***Description***
+	op1 + op2	Adds op1 and op2
*	op1 * op2	Multiplies op1 by op2
/	op1 / op2	Divides op1 by op2
%	op1 % op2	Computes the remainder of dividing op1 by op2
-	op1 - op2	Subtracts op2 from op1

Table 3: Operator Arithmatika 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 *(a single aritmatika operation)*, hasilnya berupa *floating point*. *Integer* adalah *converter* secara *implisit* ke bentuk angka *floating-point* sebelum operasi berperan mengambil tempat.

1.10.2. 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 dah mengurangi nilai yang tersimpan dalm bentuk variabel angka terhadap nilai 1.

Sebagai contoh, pernyataan,

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

pernyataan tersebut ekuivalen dengan,

```
count++;
```

Operator	***Use***	***Description***
++	op++	Increments op by 1; evaluates to the value of op before it was incremented
++	++op	Increments op by 1; evaluates to the value of op after it was incremented
--	op--	Decrements op by 1; evaluates to the value of op before it was decremented
--	--op	Decrements op by 1; evaluates to the value of op after it was decremented

Table 4: operator Increment dan Decrement

Operator *increment* dan *decrement* dapat ditempatkan sebelum atau sesudah *operand*.

Ketika digunakan sebelum *operand*, akan menyebabkan variabel di*increment* atau di*decrement* oleh 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
```

Pedoman Penulisan Program:
Selalu jaga pernyataan yang mengandung operator increment dan decrement untuk dipahami secara mudah dan sederhana.

1.10.3 Operator Relasi

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

Operator	***Use***	***Description***
>	op1 > op2	op1 is greater than op2
>=	op1 >= op2	op1 is greater than or equal to op2
<	op1 < op2	op1 is less than op2
<=	op1 <= op2	op1 is less than or equal to op2
==	op1 == op2	op1 and op2 are equal
!=	op1 != op2	op1 and op2 are not equal

Table 5: Operator Relasi

Berikut ini adalah contoh program yang menggunakan operator Relasi,

```
public class RelasiDemo
 {
     public static void main(String[] args) {
         //a few numbers
         int i = 37;
         int j = 42;
         int k = 42;
         System.out.println("Variable values...");
         System.out.println("    i = " + i);
         System.out.println("    j = " + j);
         System.out.println("    k = " + k);

    //lebih besar dari
         System.out.println("Greater than...");
         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("Greater than or equal to...");
         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("Less than...");
         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("Less than or equal to...");
         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("Equal to...");
         System.out.println("    i == j = " + (i == j)); //false
         System.out.println("    k == j = " + (k == j)); //true

    //tidak sama dengan
         System.out.println("Not equal to...");
```

```
        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
```

1.10.4 Operator logika

Operator logika memiliki satu atau lebih *operand boolean* yang menghasilkan nilai *boolean*. Ada 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.

1.10.4.1 && (logika AND) dan & (boolean logika AND)

Berikut ini adalah tabel kebenaran untuk && dan &,

x1	***x2***	***Result***
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

Table 6: Truth table for & and &&

Perbedaan dasar antara operator && dan & adalah bahwa && men*supports* ***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*.

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

Berikut ini adalah tabel kebenaran untuk || dan |,

x1	***x2***	***Result***
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Table 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*.

1.10.4.3 ^ (boolean logika ExclusiveOR)

Berikut ini adalah tabel kebenaran untuk ^,

x1	***x2***	***Result***
TRUE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Table 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
```

1.10.4.4 ! (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	***Result***
TRUE	FALSE
FALSE	TRUE

Table 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
```

1.10.5 Operator Kondisi(?:)

operator kondisi **?:** adalah operator *ternary*. Hal ini berarti bahwa operator ini digunakan dalam tiga bentuk pernyataan *conditional argumen* yang digunakan bersama-sama. 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.

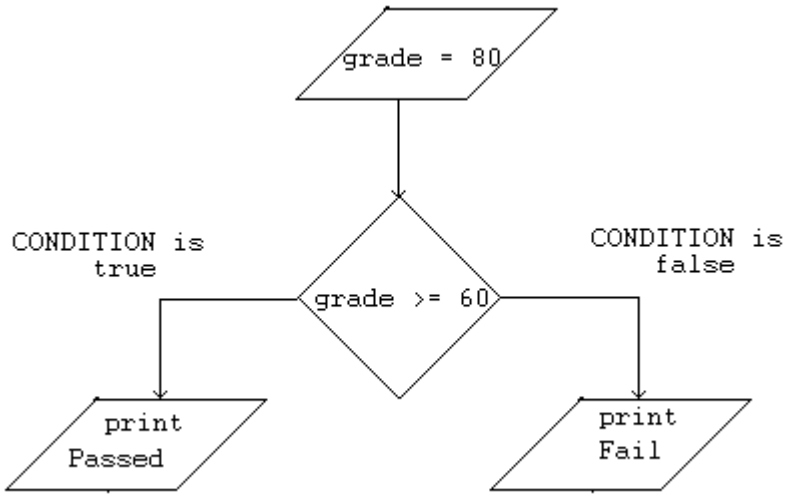
Sebagai contoh, diberikan *code* sebagai berikut,

```
public class kondisiOperator
{
          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,



Gambar2: Flowchart

Berikut ini adalah *program* lain yang menggunakan operator ?: ,

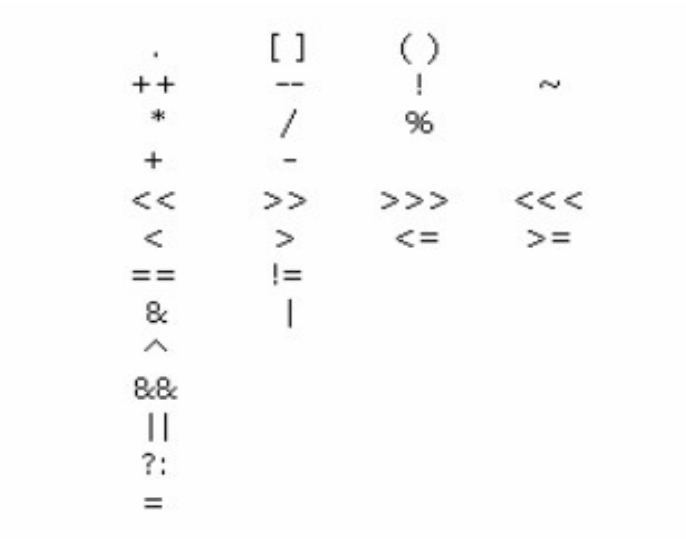
```
class kondisiOperator
{
        public static void main( String[] args ){

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

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

Hasil keluaran *program* adalah,

```
Score = 10
```

1.10.6 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;
```

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

1.11 Latihan

1.11.1 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.

Variable name	*Data tipe*	*Initial value*
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
```

1.11.2. Mendapatkan nilai rata-rata dari tiga angka

Buatlah *program* yang menghasilkan *output* nilai rata-ratadari 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
Average is = 25
```

1.11.3. 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 (***HINT***: 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
```

1.11.4. 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++

BAB 2

Mendapatkan Input dari Keyboard

2.1 Tujuan

Kita sudah mempelajari konsep mendasar pada *Java* dan menulis beberapa *program* sederhana. Sekarang kita akan mencoba membuat *program* kita lebih interaktif dengan menggunakan *input* dari *keyboard*. Pada bab ini, kita akan mempelajari dua cara memberikan *input*, yang pertama adalah menggunakan kelas *BufferedReader* dan melalui *GUI (Graphical User Interface)* dengan menggunakan kelas *JOptionPane*.

Pada akhir bab ini, para siswa diharapkan mampu:

- Membuat *program Java* yang *interaktif* yang bisa mendapatkan *input* dari *keyboard*
- Menggunakan kelas *BufferedReader* untuk mendapatkan *input* dari *keyboard* melalui layar *console*
- Menggunakan kelas *JOptionPane* untuk mendapatkan *input* dari *keyboard* melalui *GUI*

2.2 Menggunakan BufferedReader untuk mendapatkan input

Pada bagian ini, kita akan menggunakan kelas *BufferedReader* yang berada di *java.io package* untuk mendapatkan *input* dari *keyboard*.
Berikut ini adalah langkah-langkah yang diperlukan untuk mendapatkan *input* dari *keyboard*:

1. Tambahkan di bagian paling atas *code* anda:

```
import java.io.*;
```

2. Tambahkan *statement* ini:

```
BufferedReader dataIn = new BufferedReader(new InputStreamReader( System.in) );
```

3. Deklarasikan variabel *String temporer* untuk mendapatkan *input*, dan gunakan fungsi *readLine()* untuk mendapatkan *input* dari *keyboard*. Anda harus mengetikkannya di dalam *blok try-catch*:

```
try{
        String temp = dataIn.readLine();
}
catch( IOException e ){
        System.out.println("Error in getting input");
}
```

Berikut ini adalah *source code* selengkapnya:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;


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

                                        BufferedReader dataIn
= new BufferedReader(new
            InputStreamReader( System.in) );

                                        String name = "";



            System.out.print("Please Enter Your Name:");

                                        try{
                                         name =
dataIn.readLine();
                                        }catch( IOException
e ){

            System.out.println("Error!");
                                        }


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

Berikutnya akan dijelaskan setiap baris dari *code*:

Statement,

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
```

menjelaskan bahwa kita akan menggunakan kelas ***BufferedReader***, ***InputStreamReader*** dan ***IOException*** yang berada di ***java.io package***. *Java Application Programming Interface (API)* sudah berisi ratusan kelas yang bisa digunakan untuk *program* anda. Kelas-kelas tersebut dikumpulkan ke dalam ***packages*.**

Packages memiliki kelas yang mempunyai fungsi yang saling berhubungan. Seperti pada contoh di atas, ***java.io package*** mengandung kelas-kelas yang memungkinkan *program* untuk melakukan *input* dan *output* data. *Statement* diatas juga dapat ditulis,

```
import java.io.*;
```

yang akan mengeluarkan semua kelas yang berada pada paket, dan selanjutnya kita bisa menggunakan kelas-kelas tersebut pada *program* kita.

Dua *statement* selanjutnya,

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

kita sudah mempelajari pada bab sebelumnya. Statement ini menyatakan bahwa kita mendeklarasikan sebuah *class* bernama *GetInputFromKeyboard* dan kita mendeklarasikan *main method*.

Pada statement,

```
BufferedReader dataIn = new BufferedReader(new

            InputStreamReader( System.in) );
```

kita mendeklarasikan sebuah variabel bernama ***dataIn*** dengan tipe kelas ***BufferedReader***. Jangan mengkhawatirkan tentang maksud dari *syntax* saat ini. Kita akan menjelaskannya pada akhir pembahasan.

Sekarang, kita akan mendeklarasikan variabel *String* dengan *identifier name*,

```
String name = "";
```

Statement diatas merupakan tempat untuk menyimpan *input* dari *user*. Variabel name diinisialisasi sebagai *String* kosong "". Sebaiknya kita selalu menginisialisasi sebuah variabel setelah kita mendeklarasikannya.

Baris berikutnya adalah memberikan *output string* pada layar menanyakan nama *user*.

```
System.out.print("Please Enter Your Name:");
```

Sekarang, *block* di bawah ini merupakan *try-catch block*,

```
try{
            name = dataIn.readLine();
}catch( IOException e ){
            System.out.println("Error!");
}
```

Pada baris ini menjelaskan bahwa kemungkinan terjadi *error* pada *statement*

```
name = dataIn.readLine();
```

akan ditangkap. Kita akan membahas tentang exception handling pada bab selanjutnya dari pembahasan ini, tetapi untuk sekarang, anda cukup mencatat bahwa anda perlu menambahkan kode ini untuk menggunakan *readLine() method* dari *BufferedReader* untuk mendapatkan *input* dari *user*.

Selanjutnya *statement*,

```
name = dataIn.readLine();
```

method diatas memanggil *dataIn.readLine()*, mendapatkan *input* dari *user* dan memberikan sebuah nilai *String*. Nilai ini akan disimpan ke dalam variabel name, yang akan kita gunakan pada *statement* akhir untuk menyambut *user*,

```
System.out.println("Hello " + name + "!");
```

2.3 Menggunakan JOptionPane untuk mendapatkan input

Cara lain untuk mendapatkan *input* dari *user* adalah dengan menggunakan kelas *JoptionPane* yang didapatkan dari *javax.swing package*. *JoptionPane* membuat kemudahan dengan memunculkan *dialog box* standar yang memberikan kepada *user* sebuah nilai atau menginformasikan sesuatu.

Berikan kode berikut ini,

```
import javax.swing.JOptionPane;

public class GetInputFromKeyboard
{

            public static void main( String[] args ){
                                        String name = "";
                                        name =
JoptionPane.showInputDialog("Please enter your
                                                name");

                                        String msg = "Hello "
+ name + "!";


            JOptionPane.showMessageDialog(null, msg);

            }
}
```

Akan menghasilkan *output*,

Statement pertama,

```
import javax.swing.JOptionPane;
```

Menjelaskan bahwa kita meng*import* kelas *JoptionPane* dari *javax.swing package*.

Bisa juga ditulis,

```
import javax.swing.*;
```

statement selanjutnya,

```
name = JOptionPane.showInputDialog("Please enter your name");
```

membuat sebuah *JOptionPane input dialog*, yang akan menampilkan *dialog* dengan sebuah pesan, sebuah *textfield* dan tombol OK seperti pada gambar. Hasil dari dialog tersebut adalah *String* dan disimpan ke dalam variabel *name*.

Sekarang kita membuat pesan selamat datang, yang akan disimpan ke dalam variabel msg,

```
String msg = "Hello " + name + "!";
```

Baris selanjutnya adalah menampilkan sebuah dialog yang memilki sebuah pesan dan tombol OK,

```
JOptionPane.showMessageDialog(null, msg);
```

2.4 Latihan

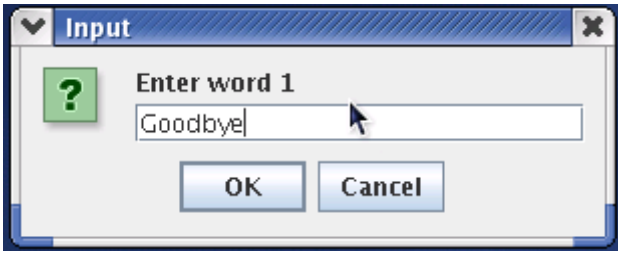
2.4.1 Kata Terakhir (versi BufferedReader)

Menggunakan *BufferedReader*, tanyakan tiga kata dari *user* dan tampilkan *output* dari *input user* tersebut ke layar. Contoh,

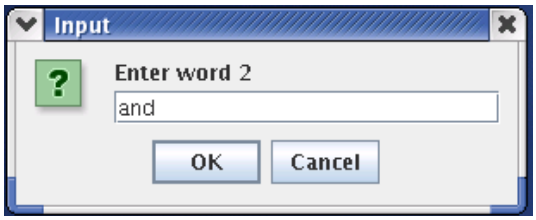
```
Enter word1:Goodbye
Enter word2:and
Enter word3:Hello

Goodbye and Hello
```

2.4.2 Kata Terakhir (versi JOptionPane)

Menggunakan *JOptionPane*, tanyakan tiga kata dari *user* dan tampilkan *output* dari *input user* tersebut ke layar. Contoh



Gambar 1: Input Pertama



Gambar 2: Input Kedua

Gambar 3: Menampilkan Pesan