



Dasar Java (Konsol)

Modul Pembelajaran



September 2021

Daftar Isi

Persyaratan Sistem.....	ii
Bab 01. Dasar Java.....	1
A. Pengenalan Bahasa Java	1
B. Pengenalan <i>Integrated Development Environment</i> (IDE).....	4
C. Variabel.....	8
D. Tipe Data Variabel.....	10
E. <i>Naming Convention</i>	15
F. <i>Imports</i>	18
G. <i>Output</i>	20
H. <i>Input</i>	26
I. Operator	28
J. Seleksi	33
K. Repetisi	37
L. Latihan Soal	41
Bab 02. Array dan Pointer	45
A. Array Satu Dimensi.....	45
B. Array Banyak Dimensi	47
C. Latihan Soal	49

Persyaratan Sistem

Perangkat Lunak

- Eclipse IDE for Java Developers 2020-06
- Java Development Kit (JDK) 8

Bab 01. Dasar Java

A. Pengenalan Bahasa Java

Java merupakan sebuah bahasa pemrograman yang dikembangkan pertama kali oleh James Gosling saat bekerja di Sun Microsystems, yang sekarang merupakan anak perusahaan dari Oracle Corporation, dan dirilis pada tahun 1995. Bahasa pemrograman Java dirancang untuk menyelesaikan masalah yang ditemui dalam praktik pemrograman modern. Bahasa ini memiliki sintaks yang hampir mirip dengan bahasa pemrograman C dan C++, tetapi memiliki model objek yang lebih sederhana dan fasilitas level rendah (*low-level*) yang lebih sedikit.



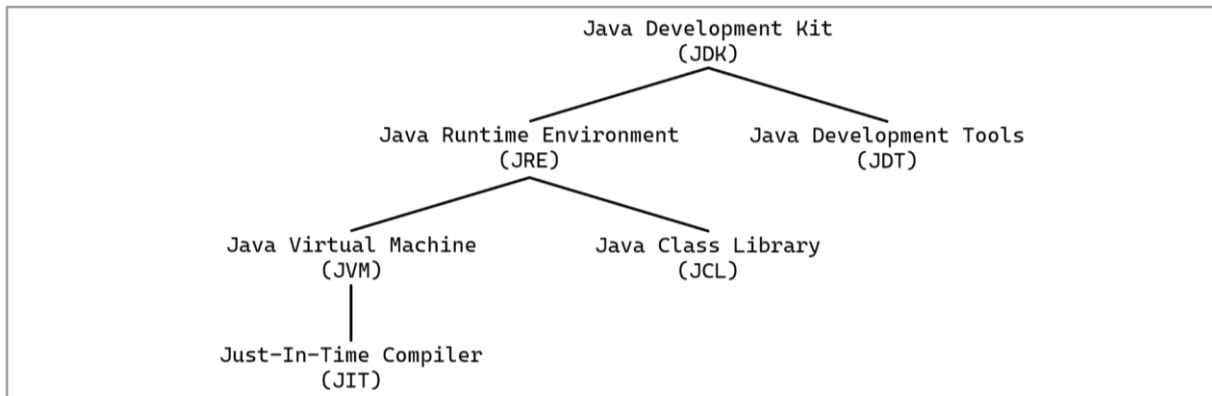
Gambar 1. Logo Bahasa Pemrograman Java

Berdasarkan *The Java Language: A White Paper*, terdapat serangkaian **kata kunci** (**keyword**) yang dapat digunakan untuk menggambarkan bahasa pemrograman Java.

1. **Sederhana** (*Simple*).
2. **Berorientasi pada Objek** (*Object-Oriented*).
3. **Terdistribusi** (*Distributed*).
4. **Kuat** (*Robust*).
5. **Aman** (*Secure*).
6. **Arsitektur yang Netral** (*Architecture Neutral*).
7. **Portabel** (*Portable*).
8. **Penerjemah** (*Interpreter*).
9. **Kinerja yang Tinggi** (*High Performance*).
10. **Beberapa Proses** (*Multithreaded*).
11. **Dinamis** (*Dynamic*).

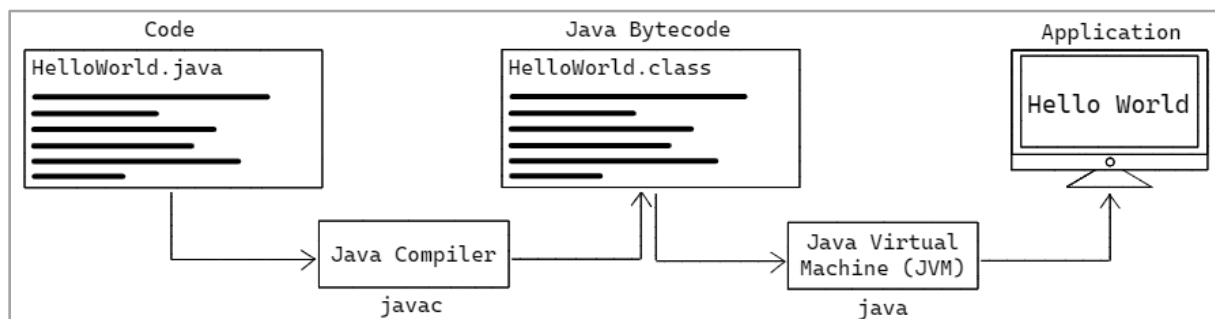
Ada beberapa **istilah** yang perlu diketahui berkaitan dengan cara kerja bahasa pemrograman Java. Berikut adalah istilah-istilah tersebut berserta dengan penjelasannya.

- a. **Java Development Kit** (JDK). JDK merupakan perangkat lunak yang digunakan untuk membuat aplikasi Java. Di dalamnya berisikan *Java Development Tools* (JDT) dan *Java Runtime Environment* (JRE).
- b. **Java Development Tools** (JDT). JDT merupakan sekumpulan alat yang digunakan untuk menunjang aplikasi Java, seperti *compiler* (javac), *intepreter* (java), *archiver* (jar), pembuat dokumentasi (javadoc), dan lain-lain.
- c. **Java Runtime Environment** (JRE). JRE merupakan sekumpulan alat yang digunakan untuk menjalankan aplikasi Java. Di dalamnya berisikan *Java Class Library* (JCL) dan *Java Virtual Machine* (JVM).
- d. **Java Class Library** (JCL). JCL merupakan sekumpulan *library* yang dimuat secara dinamis dan digunakan oleh *Java Virtual Machine* (JVM) saat aplikasi Java dijalankan. Karena bahasa pemrograman Java tidak bergantung pada sistem operasi komputer, Java sudah menyediakan *library* bawaan sendiri yang berisikan fungsi umum untuk sistem operasi. Contohnya adalah java.lang, java.io, java.math, dan lain-lain.
- e. **Java Virtual Machine** (JVM). JVM merupakan alat yang digunakan untuk menjalankan Java *bytecode* tanpa adanya ketergantungan dari sistem operasi tertentu. JVM juga dapat menjalankan bahasa pemrograman lainnya, seperti Jython, Groovy, dan Juby, selama kode tersebut dikompilasikan Java *bytecode*.
- f. **Just-In-Time Compiler** (JIT). JIT merupakan teknik yang digunakan oleh Java untuk meningkatkan performa kerja aplikasi saat mengompilasikan *bytecode* menjadi bahasa mesin saat aplikasi dijalankan.



Gambar 2. Komponen *Java Development Kit* (JDK)

Setelah mengetahui istilah-istilah tersebut, barulah kita dapat mempelajari **cara kerja** bahasa pemrograman Java yang dapat digambarkan seperti pada **Error! Reference source not found..** Pertama, **kode (file .java)** diubah menjadi **Java bytecode (file .class)** menggunakan **Java compiler (javac)**. Proses ini disebut juga sebagai proses kompilasi. Hasil dari proses tersebut akan dijalankan oleh **Java Virtual Machine (JVM)** dengan menggunakan **Java launcher tool (java)**. Pada saat dijalankan, JVM akan menerjemahkan **Java bytecode** menjadi **bahasa mesin** sehingga instruksi dapat dijalankan oleh komputer pengguna.



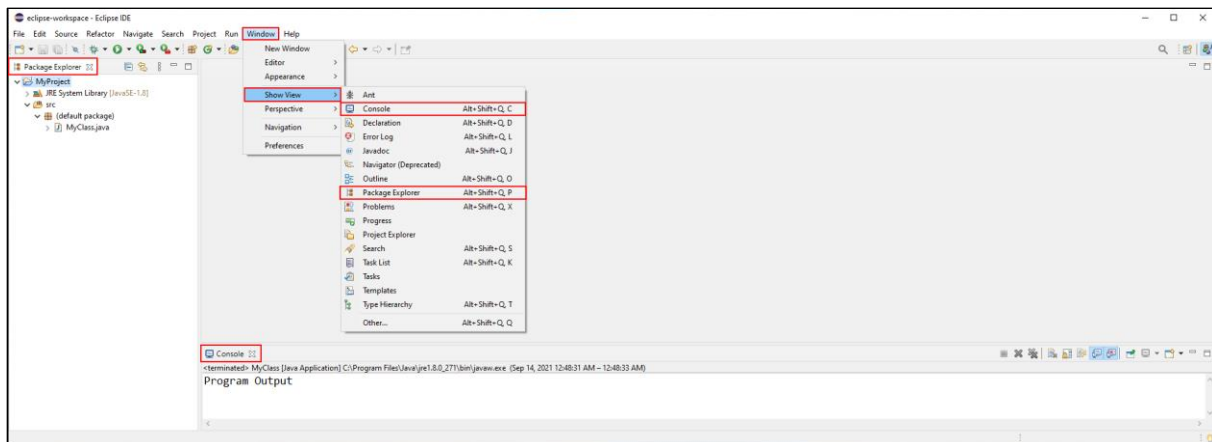
Gambar 3. Cara Kerja Bahasa Pemrograman Java

Sampai saat ini, Java selalu memperbaharui *Java Development Kit* (JDK) untuk menambahkan fitur yang tentunya mempermudah pengembang aplikasi untuk membuat aplikasi Java. JDK terbaru yang dimiliki oleh Java saat ini adalah versi JDK 17. Namun, untuk materi ini akan digunakan versi **JDK 8**.

B. Pengenalan *Integrated Development Environment* (IDE)

***Integrated Development Environment* (IDE)** merupakan suatu perangkat lunak yang menyediakan fasilitas bagi pengguna untuk membuat atau mengembangkan suatu program. Biasanya IDE terdiri dari *source code editor*, *build automation tools*, dan *debugger*. Untuk materi ini akan digunakan perangkat lunak **Eclipse IDE for Java Developers 2020-06**.

Apabila perangkat lunak Eclipse IDE tersebut sudah selesai terpasang di komputer, silakan buka IDE tersebut. Terdapat dua buah **window** yang akan sering digunakan, yaitu **package explorer** dan **console**. *Package explore* berguna untuk menampilkan struktur *project* Java yang dibuat. Sedangkan, *console* berguna sebagai tempat berinteraksi antara aplikasi dengan pengembang aplikasi, seperti melihat *output* yang diberikan oleh aplikasi atau memberikan *input* kepada aplikasi.



Gambar 4. Tampilan Dasar *Eclipse IDE for Java Developers 2020-06*

Catatan:

Apabila ada **window** yang tidak muncul pada Eclipse IDE, kita dapat membukanya dengan dua cara.

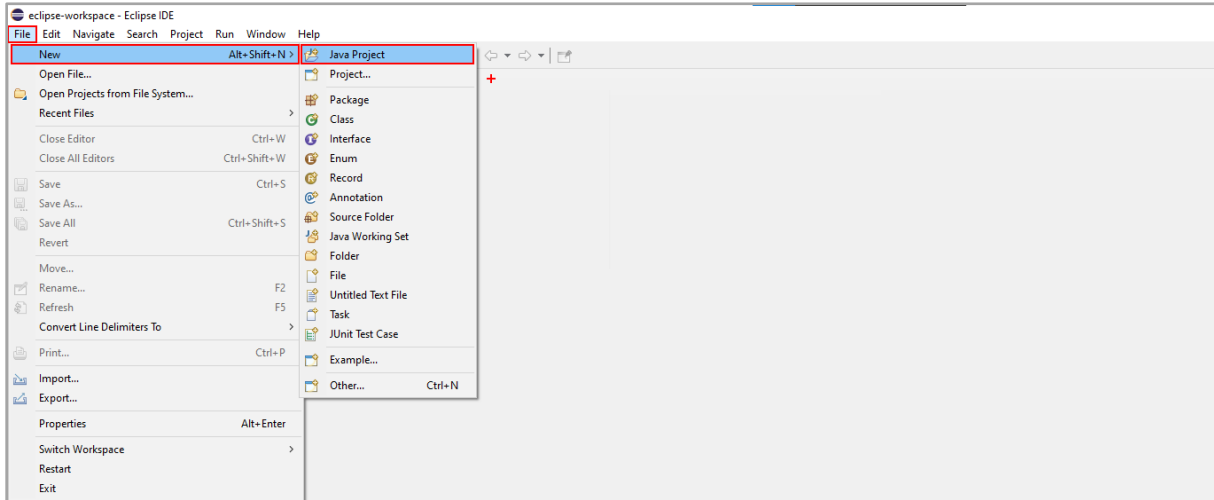
- Pilih **Window** → **Show View** → [Nama Window].
- Gunakan **shortcut** yang tertera pada opsi 1.

Misalkan: **console** menggunakan **shortcut Alt + Shift + Q, C**

package explorer menggunakan **shortcut Alt + Shift + Q, P**

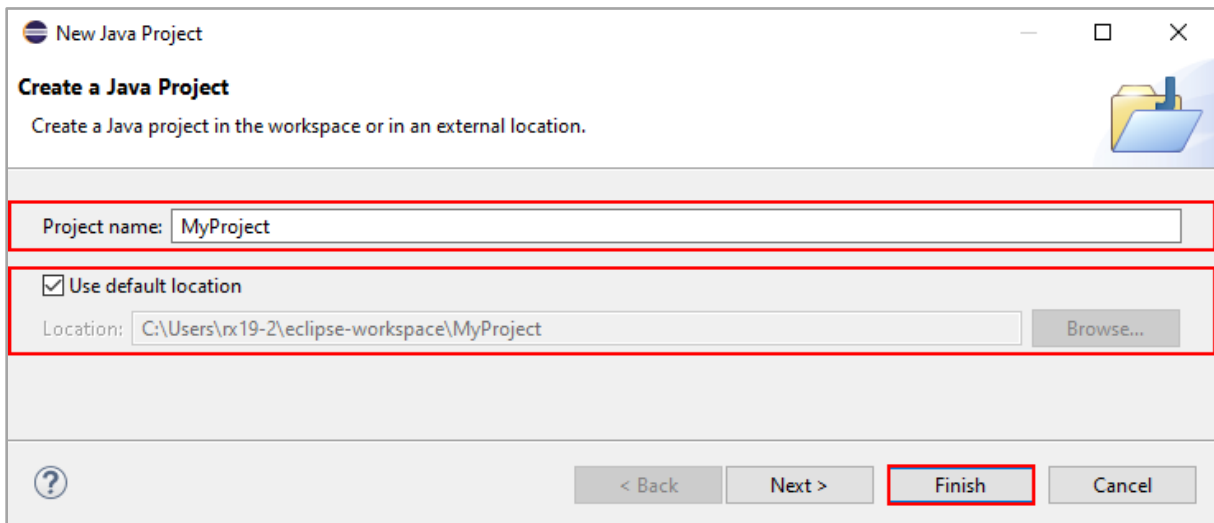
Sebelum masuk ke dalam materi, kita harus membuat *project* Java terlebih dahulu. Cara untuk **membuat *project* Java** pada Eclipse IDE adalah sebagai berikut.

a. Pilih **File** → **New** → **Java Project**.



Gambar 5. Membuat *Project* Java (a)

b. Pada *window New Java Project*, masukan **nama** dan **lokasi *project* Java** yang diinginkan. Apabila sudah, pilih tombol **Finish**.



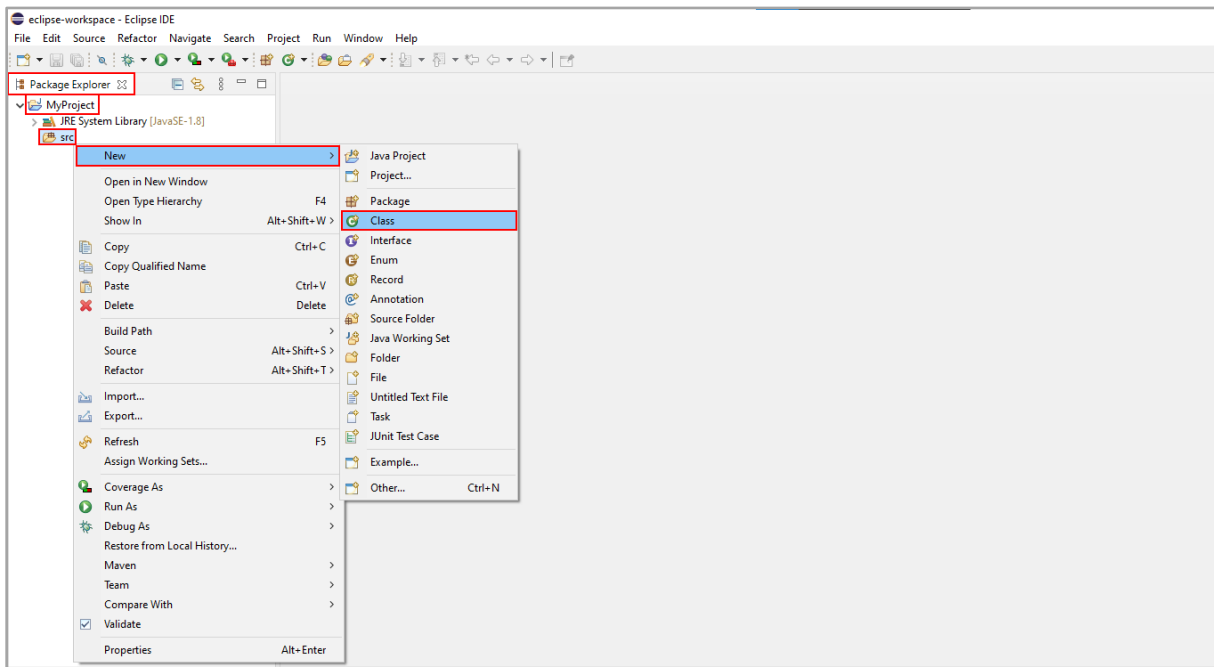
Gambar 6. Membuat *Project* Java (b)

Catatan:

Secara otomatis, Eclipse IDE menggunakan lokasi *workspace* yang dipilih di awal sebagai **lokasi *project***. Untuk mengubahnya, hapus centang pada **checkbox Use default location** dan klik tombol **Browse...** untuk memilih lokasi yang diinginkan.

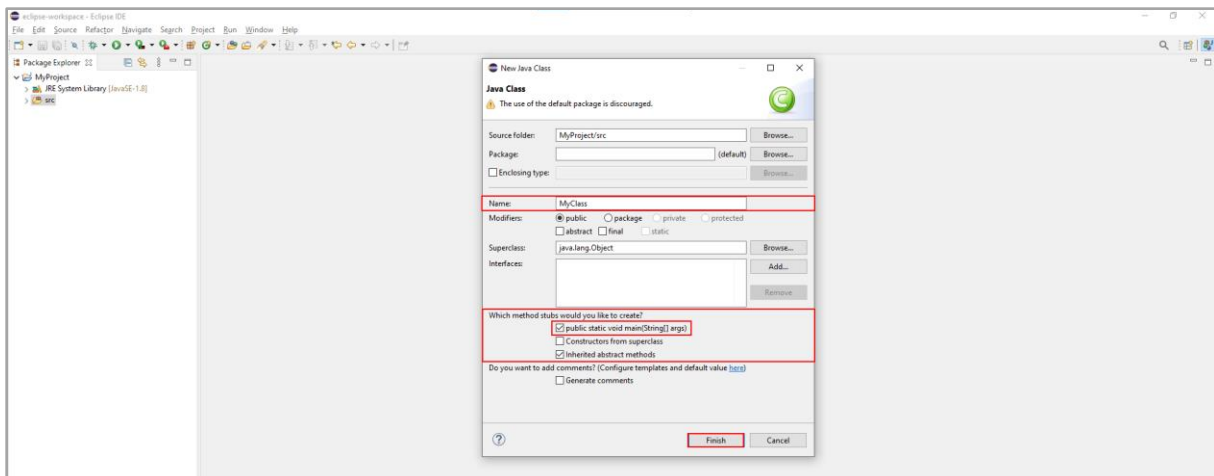
Apabila *project* Java selesai dibuat, kita akan membuat *file* Java sebagai tempat yang akan digunakan untuk menuliskan logika pemrograman. Cara **membuat *file* Java** pada Eclipse IDE adalah sebagai berikut.

- a. Pada **window *Package Explorer***, cari *package* bernama **src**. Klik kanan pada *package* tersebut, pilih **New → Class**.



Gambar 7. Membuat *File* Java (a)

- b. Pada **window *New Java Class***, masukan **nama *file* Java** yang diinginkan. Apabila sudah, pilih tombol **Finish**.



Gambar 8. Membuat *File* Java (b)

Catatan:

Pada bagian *Which method subs would you like to create?* disarankan untuk memberikan centang pada **checkbox public static void main (String[] args)** agar Eclipse IDE membuat *method main* secara otomatis.

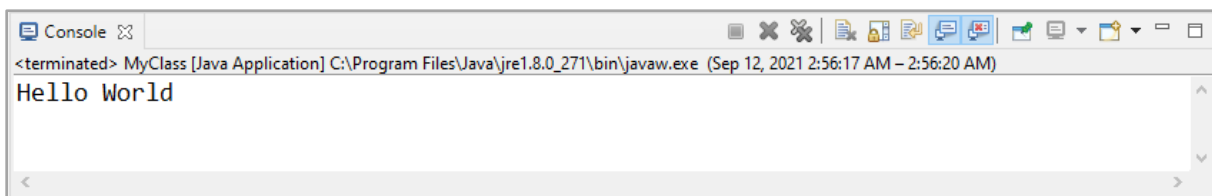
Apabila *file* Java telah selesai dibuat, kita akan mengecek apakah Java dapat berjalan dengan baik dengan **membuat aplikasi output sederhana**. Bukalah *file* Java yang telah dibuat sebelumnya, kemudian ikuti *code* seperti pada **Error! Reference source not found..** Apabila sudah, tekan tombol **Ctrl + F11** untuk **menjalankan kode** tersebut. Hasilnya akan terlihat seperti pada **Error! Reference source not found..**

```
1 public class MyClass {  
2  
3     public static void main(String[] args) {  
4         System.out.println("Hello World");  
5     }  
6  
7 }
```

scope 2 (mendefinisikan method main)

scope 1 (mendefinisikan kelas)

Kode 1. Program Output Sederhana



Hasil 1. Program Output Sederhana

Dari **Error! Reference source not found.** di atas, terdapat **dua scope** (yang berada di dalam kurung kurawal) utama, yaitu **mendefinisikan kelas** dan **mendefinisikan method main**. Mendefinisikan kelas (baris ke-1) diperlukan untuk memberi tahu *compiler* nama kelas yang akan dijalankan. Sedangkan, mendefinisikan *method main* (baris ke-3) diperlukan untuk memberi tahu bahwa *method* ini akan dijalankan pertama kali ketika aplikasi dijalankan. Kemudian, di dalam *method main* ada sebuah kode (baris ke-4) untuk melakukan *output* ke *window console*. Penjelasan lebih lanjut mengenai *output* akan dijelaskan pada materi selanjutnya.

C. Variabel

Dalam bahasa pemrograman, **variabel** merupakan tempat yang digunakan menampung sebuah nilai. Nilai dari sebuah variabel dapat diubah-ubah sesuai dengan kebutuhan. Variabel yang di deklarasikan harus memiliki tipe data tertentu. Untuk memberi nilai ke sebuah variabel, kita dapat menggunakan **tanda sama dengan (=)**.

```
Format Deklarasi           : <TipeData> <NamaVariabel>;  
Format Inisialisasi       : <NameVariabel> = <NilaiVariabel>;  
Format Deklarasi dan Inisialisasi : <TipeData> <NamaVariabel> = <NilaiVariabel>;
```

Format 1. Mendeklarasikan dan Menginisialisasikan Variabel

```
3 public static void main(String[] args) {  
4     int x;      // Mendeklarasikan variabel  
5     x = 10;     // Menginisialisasikan variable  
6  
7     int y = 20; // Mendeklarasikan dan menginisialisasikan variabel  
8  
9  
10 }
```

Kode 2. Mendeklarasikan dan Menginisialisasi Variabel

Catatan:

- Komentar baris tunggal diawali dengan tanda `"/"`.
Komentar baris banyak diawali dengan tanda `"/"` dan diakhiri dengan tanda `"*/"`.
- Pastikan untuk selalu meletakkan tanda `;` di setiap akhir perintah.

```
1 // Komentar baris tunggal  
2  
3 /*  
4  * Komentar  
5  * Baris  
6  * Banyak  
7  */
```

Kode 3. Penggunaan Komentar

Apabila kita ingin membuat sebuah **variabel bernilai konstan** (tidak dapat berubah-ubah), kita dapat menggunakan kata kunci (keyword) **final**. Kata kunci ini diletakan sebelum menuliskan tipe data dari variabel.

```
Format Deklarasi           : final <TipeData> <NamaVariabel>;  
Format Deklarasi dan Inisialisasi : final <TipeData> <NamaVariabel> = <NilaiVariabel>;
```

Format 2. Mendeklarasikan dan Menginisialisasikan Variabel Konstan

```
3 public static void main(String[] args) {  
4  
5     final double PI = 3.14159; // OK  
6     PI = 3.14;                // Error  
7  
8     final double e;  
9     e = 2.71828;              // OK  
10    e = 2.71;                 // Error  
11  
12 }
```

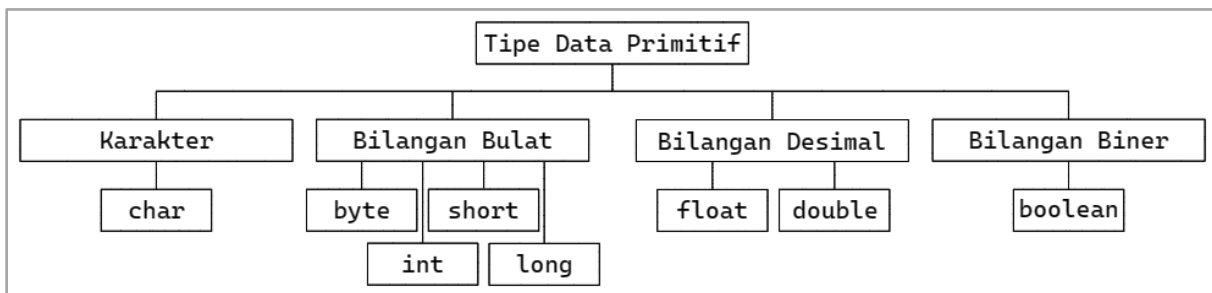
Kode 4. Mendeklarasikan dan Menginisialisasikan Variabel Konstan

Variabel PI merupakan variabel konstan dan sudah diinisialisasikan dengan nilai 3.14159 (baris ke-5). Jika nilai variabel PI diinisialisasikan ulang dengan nilai 3.14 (baris ke-6) maka akan terjadi error. Sedangkan, variabel e merupakan variabel konstan yang belum diinisialisasikan (baris ke-8). Jika nilai variabel e diinisialisasikan dengan nilai 2.71828 (baris ke-9) hal tersebut boleh saja, tetapi nilai tersebut tidak dapat diganti lagi karena akan terjadi error (baris ke-10).

D. Tipe Data Variabel

Pada materi sebelumnya, pernah disinggung mengenai tipe data pada variabel. Setiap variabel wajib memiliki **tipe data** untuk mendefinisikan nilai apa yang harus dimiliki oleh variabel tersebut. Terdapat dua jenis tipe data, yaitu tipe data primitif dan tipe data non-primitif.

Tipe data primitif adalah tipe data yang paling mendasar pada sebuah bahasa pemrograman. Dengan kata lain, tipe data jenis ini **sudah tersedia** secara langsung pada bahasa pemrograman Java. Tipe data primitif memiliki **ukuran memori yang tetap** dan **tidak memiliki method**. Berikut adalah *list* tipe data primitif yang dimiliki oleh Java.



Gambar 9. Jenis-Jenis Tipe Data Primitif

Tipe Data	Memori	Nilai
<i>boolean</i>	1 bytes	true atau false
<i>char</i> (*)	2 bytes	'\u0000' s/d '\uFFFF' (0 s/d 65,535)
<i>byte</i>	1 bytes	-128 s/d 127 (-2^7 s/d $2^7 - 1$)
<i>short</i>	2 bytes	-32,768 s/d 32,767 (-2^{15} s/d $2^{15} - 1$)
<i>int</i>	4 bytes	-2,147,483,648 s/d 2,147,483,647 (-2^{31} s/d $2^{31} - 1$)
<i>long</i>	8 bytes	-9,223,372,036,854,775,808 s/d 9,223,372,036,854,775,807 (-2^{63} s/d $2^{63} - 1$)
<i>float</i> (**)	4 bytes	$\pm 3.40282346638528860e+38$ ($\pm (2 - 2^{-23}) \times 2^{127}$)
<i>double</i> (**)	8 bytes	$\pm 1.79769313486231570e+308$ ($\pm (2 - 2^{-52}) \times 2^{1023}$)

(*) Menggunakan standart ISO *Unicode Character Set*.

(**) Menggunakan standart IEEE 754 *Floating Point*.

Pada bahasa pemrograman C dan C++, aplikasi sering kali harus ditulis dalam beberapa versi untuk mendukung komputer yang berbeda-beda. Hal ini dikarenakan ukuran memori tipe data primitif pada bahasa pemrograman tersebut tidak dijamin sama untuk tiap komputer. Misalnya, ukuran memori *int* pada satu komputer sebesar 16 *bits* (2 *bytes*), sedangkan ukuran memori *int* pada komputer lainnya sebesar 32 *bits* (4 *bytes*). Namun, pada bahasa pemrograman Java, ukuran memori *int* akan selalu 32 *bits* (4 *bytes*).

Tipe data non-primitif adalah tipe data yang tidak didefinisikan sendiri oleh bahasa pemrograman Java melainkan **didefinisikan sendiri** oleh pembuat aplikasi. Berbeda dengan tipe data primitif, tipe data non-primitif **dapat memiliki *method***. Sering kali disebut juga sebagai **tipe data referensi** dikarenakan tipe data ini tidak menyimpan nilainya secara langsung, tetapi menyimpan alamat memori dari nilai yang disimpan. Terdapat beberapa jenis tipe data non-primitif yang ada di Java, yaitu *String*, *Array*, *Class*, *Interface*, dan lainnya. Dari tipe data tersebut, yang akan kita bahas hanya *String*.

String merupakan tipe data yang digunakan untuk menyimpan **kumpulan dari beberapa karakter**. *String* tidak termasuk dalam tipe data primitif dikarenakan *String* memiliki *method* bawaan seperti *charAt*, *contains*, *endsWith*, dan lain-lain. Kelebihan yang dimiliki oleh *String* dibandingkan *array* dari *char* adalah *String* sudah memiliki karakter “\0” di bagian akhir. Terdapat dua cara untuk **membuat *String***, yaitu menggunakan **tanda kutip dua (“)** langsung atau dengan menggunakan kata kunci (*keyword*) ***new***.

```
3 public static void main(String[] args) {  
4     String str1 = "SLC"; // Cara ke-1 (menggunakan tanda kutip dua langsung)  
5     String str2 = new String("SLC"); // Cara ke-2 (menggunakan keyword new)  
6 }
```

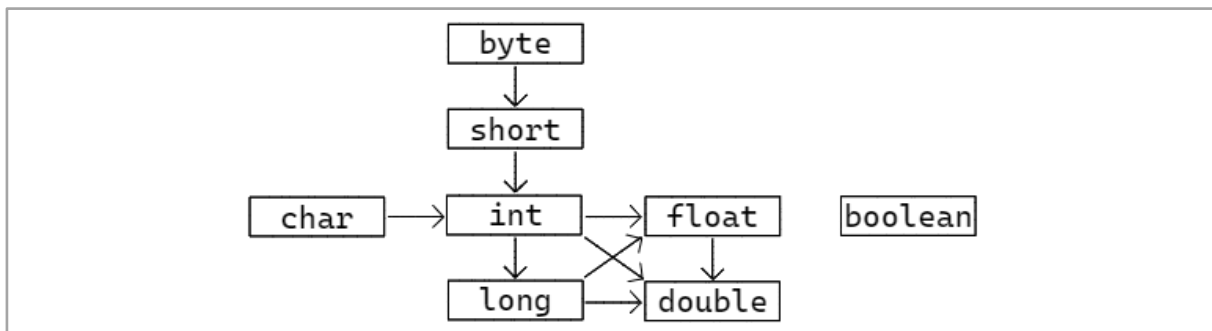
Kode 5. Mendeklarasikan dan Menginisialisasikan String

Bahasa pemrograman Java sangat ketat dalam membiarkan kita menginisialisasikan sebuah nilai ke dalam variabel dengan tipe data tertentu. Apabila terdapat sebuah variabel dengan tipe data *int*, kita tidak dapat memberikan variabel tersebut nilai dari tipe data lain, kecuali menggunakan *conversion* atau *type casting*.

Sesuai dengan namanya, **conversion** atau **type casting** merupakan operator yang digunakan untuk **mengubah nilai dari satu tipe data ke tipe data lainnya**. Berdasarkan urutannya, *type casting* dapat dibagi menjadi dua jenis, yaitu *implicit type casting* dan *explicit type casting*.

a. **Implicit Type Casting**

Implicit type casting digunakan untuk **mengubah nilai dari tipe data yang ukurannya lebih kecil menjadi tipe data yang ukurannya lebih besar**. *Type casting* jenis ini disebut juga **type casting otomatis**. Hal ini dikarenakan pengembang aplikasi tidak perlu menambahkan kata kunci (*keyword*) apa pun untuk mengubah tipe data. Bahasa pemrograman Java secara langsung dapat mengubah tipe data tersebut.



Gambar 10. Diagram *Implicit Type Casting*

```
3 public static void main(String[] args) {
4     byte b = 100;
5
6     short s = b;    // 100
7     int i = b;      // 100
8     long l = b;     // 100
9     float f = b;    // 100.0
10    double d = b;   // 100.0
11 }
```

Kode 6. *Implicit Type Casting* dari Tipe Data byte

```
3 public static void main(String[] args) {
4     char c = 'A';
5
6     int i = c;      // 65
7     long l = c;     // 65
8     float f = c;    // 65.0
9     double d = c;   // 65.0
10 }
```

Kode 7. *Implicit Type Casting* dari Tipe Data char

Perhatikan **Error! Reference source not found.** di atas. Pada gambar tersebut, terlihat sebuah variabel bernama *c* dengan tipe data *char* memiliki nilai 'A'. Ketika dilakukan *type casting*, nilainya berubah menjadi sebuah angka. Hal ini dikarenakan nilai setiap karakter disimpan oleh memori dalam bentuk angka berdasarkan **American Standard Code for Information Interchange (ASCII)**, yaitu skema pengkodean karakter berdasarkan urutan alfabet bahasa inggris. Berikut adalah *list* karakter yang ada di dalam ASCII.

Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char
0	0000 0000	00	[NUL]	32	0010 0000	20	space	64	0100 0000	40	@	96	0110 0000	60	`
1	0000 0001	01	[SOH]	33	0010 0001	21	!	65	0100 0001	41	A	97	0110 0001	61	a
2	0000 0010	02	[STX]	34	0010 0010	22	"	66	0100 0010	42	B	98	0110 0010	62	b
3	0000 0011	03	[ETX]	35	0010 0011	23	#	67	0100 0011	43	C	99	0110 0011	63	c
4	0000 0100	04	[EOT]	36	0010 0100	24	\$	68	0100 0100	44	D	100	0110 0100	64	d
5	0000 0101	05	[ENQ]	37	0010 0101	25	%	69	0100 0101	45	E	101	0110 0101	65	e
6	0000 0110	06	[ACK]	38	0010 0110	26	&	70	0100 0110	46	F	102	0110 0110	66	f
7	0000 0111	07	[BEL]	39	0010 0111	27	'	71	0100 0111	47	G	103	0110 0111	67	g
8	0000 1000	08	[BS]	40	0010 1000	28	(72	0100 1000	48	H	104	0110 1000	68	h
9	0000 1001	09	[TAB]	41	0010 1001	29)	73	0100 1001	49	I	105	0110 1001	69	i
10	0000 1010	0A	[LF]	42	0010 1010	2A	*	74	0100 1010	4A	J	106	0110 1010	6A	j
11	0000 1011	0B	[VT]	43	0010 1011	2B	+	75	0100 1011	4B	K	107	0110 1011	6B	k
12	0000 1100	0C	[FF]	44	0010 1100	2C	,	76	0100 1100	4C	L	108	0110 1100	6C	l
13	0000 1101	0D	[CR]	45	0010 1101	2D	-	77	0100 1101	4D	M	109	0110 1101	6D	m
14	0000 1110	0E	[SO]	46	0010 1110	2E	.	78	0100 1110	4E	N	110	0110 1110	6E	n
15	0000 1111	0F	[SI]	47	0010 1111	2F	/	79	0100 1111	4F	O	111	0110 1111	6F	o
16	0001 0000	10	[DLE]	48	0011 0000	30	0	80	0101 0000	50	P	112	0111 0000	70	p
17	0001 0001	11	[DC1]	49	0011 0001	31	1	81	0101 0001	51	Q	113	0111 0001	71	q
18	0001 0010	12	[DC2]	50	0011 0010	32	2	82	0101 0010	52	R	114	0111 0010	72	r
19	0001 0011	13	[DC3]	51	0011 0011	33	3	83	0101 0011	53	S	115	0111 0011	73	s
20	0001 0100	14	[DC4]	52	0011 0100	34	4	84	0101 0100	54	T	116	0111 0100	74	t
21	0001 0101	15	[NAK]	53	0011 0101	35	5	85	0101 0101	55	U	117	0111 0101	75	u
22	0001 0110	16	[SYN]	54	0011 0110	36	6	86	0101 0110	56	V	118	0111 0110	76	v
23	0001 0111	17	[ETB]	55	0011 0111	37	7	87	0101 0111	57	W	119	0111 0111	77	w
24	0001 1000	18	[CAN]	56	0011 1000	38	8	88	0101 1000	58	X	120	0111 1000	78	x
25	0001 1001	19	[EM]	57	0011 1001	39	9	89	0101 1001	59	Y	121	0111 1001	79	y
26	0001 1010	1A	[SUB]	58	0011 1010	3A	:	90	0101 1010	5A	Z	122	0111 1010	7A	z
27	0001 1011	1B	[ESC]	59	0011 1011	3B	;	91	0101 1011	5B	[123	0111 1011	7B	{
28	0001 1100	1C	[FS]	60	0011 1100	3C	<	92	0101 1100	5C	\	124	0111 1100	7C	
29	0001 1101	1D	[GS]	61	0011 1101	3D	=	93	0101 1101	5D]	125	0111 1101	7D	}
30	0001 1110	1E	[RS]	62	0011 1110	3E	>	94	0101 1110	5E	^	126	0111 1110	7E	~
31	0001 1111	1F	[US]	63	0011 1111	3F	?	95	0101 1111	5F	_	127	0111 1111	7F	[DEL]

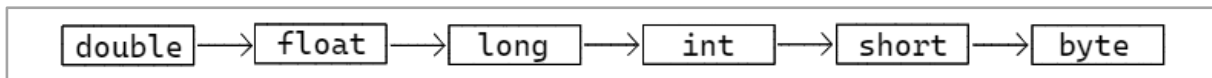
Gambar 11. American Standard Code for Information Interchange (ASCII)

b. *Explicit Type Casting*

Explicit type casting digunakan untuk **mengubah nilai dari tipe data yang ukurannya lebih besar menjadi tipe data yang ukurannya lebih kecil**. Eksplisit di sini diartikan bahwa *compiler* tidak dapat secara otomatis mengubah tipe data, sehingga harus **dilakukan secara manual** oleh pengembang aplikasi. Berikut adalah sintaks untuk melakukan *explicit type casting*.

Format : `<TipeData> <NamaVariabel> = (<TipeData>) <NilaiVariabel>;`

Format 3. *Explicit Type Casting*



Gambar 12. Diagram *Explicit Type Casting*

```
1 public class MyClass {  
2  
3     public static void main(String[] args) {  
4         double d = 50.55;  
5  
6         float f = (float) d; // 50.55  
7         long l = (long) d; // 50  
8         int i = (int) d; // 50  
9         short s = (short) d; // 50  
10        byte b = (byte) d; // 50  
11    }  
12  
13 }
```

Kode 8. *Explicit Type Casting* dari Tipe Data double

Catatan:

Ketika melakukan ***explicit type casting***, nilai dalam sebuah variabel dapat menjadi **tidak akurat** dikarenakan tipe data sebelumnya memiliki memori yang lebih besar dibandingkan setelah tipe datanya diubah

E. Naming Convention

Naming Convention merupakan **standar** yang dianjurkan untuk diikuti terkait dengan **pemberian nama *identifier*** pada suatu bahasa pemrograman. Standar ini diperlukan karena setiap pengembang aplikasi memiliki gaya dan pendekatan yang berbeda-beda untuk membuat nama *identifier*.

Dengan adanya standar ini diharapkan kode yang dibuat menjadi lebih mudah dibaca dan dipahami oleh pengembang aplikasi lainnya. Dengan demikian, semakin sedikit waktu yang dibutuhkan pengembang aplikasi untuk mencari tahu maksud dari kode tersebut dan semakin banyak waktu dapat digunakan untuk menambahkan atau memodifikasi kode tersebut. Berikut adalah standar tersebut dikelompokkan sesuai jenisnya.

a. *Identifier*

- Usahakan nama *identifier* harus **mendesripsikan sesuatu** agar mudah dipahami.
- Karakter yang dapat digunakan oleh *identifier* terdiri atas **huruf, angka, simbol garis bawah (_), dan simbol dolar (\$)**.
- Nama *identifier* hanya boleh **dimulai** dengan **huruf, simbol garis bawah (_), dan simbol dolar (\$)**. Nama *identifier* tidak dapat dimulai dengan angka.
- Nama *identifier* **tidak** dapat menggunakan **kata kunci (*keyword*)** yang sudah ditentukan Java, **true, false**, atau **null**.
- Nama *identifier* bersifat **case sensitif**.
- Disarankan untuk tidak menggunakan simbol dolar (\$) sebagai nama *identifier* walaupun memungkinkan. Simbol ini biasanya digunakan pada kode yang dihasilkan secara otomatis.

```
1 public class MyClass {  
2     String customerName_1;           // Poin 2  
3     short _myAge;                     // Poin 3  
4     long 3student;                   // Poin 3 (Error)  
5     double false, true, null;        // Poin 4 (Error)  
6     int averagePrice, AveragePrice;  // Poin 6  
7 }
```

Kode 9. Naming Convention untuk Identifier

b. **Kelas (Class)**

- Nama kelas biasanya ditulis menggunakan **pascal case**.
- Usahakan nama kelas **menggunakan kata benda** karena biasanya kelas merepresentasikan sesuatu yang nyata.

```
1 class Customer {}  
2 class PrivateAccount {}  
3 class Animal {}
```

Kode 10. *Naming Convention* untuk Kelas

c. **Interface**

- Nama *interface* biasanya ditulis menggunakan **pascal case**.
- Usahakan nama *interface* **menggunakan kata kerja** karena biasanya *interface* merepresentasikan sesuatu yang dapat dilakukan sebuah kelas.
- Biasanya beberapa pengembang aplikasi membedakan *interface* dengan memberi awalan "I".

```
1 interface Enumerable {}; // interface IEnumerable {};  
2 interface Compareable {}; // interface ICompareable{};  
3 interface Moveable {}; // interface IMoveable {};
```

Kode 11. *Naming Convention* untuk *Interface*

d. **Method**

- Nama *method* biasanya ditulis menggunakan **camel case**.
- Usahakan nama *method* **menggunakan kata kerja** karena biasanya *method* merepresentasikan algoritma dari sebuah program.

```
1 public class MyClass {  
2     void calculateTax() {}  
3     void saveLocalTransaction() {}  
4     void refund() {}  
5 }
```

Kode 12. *Naming Convention* untuk *Method*

e. **Variabel** (*Variable*)

- Nama variabel biasanya ditulis menggunakan ***camel case***.
- Usahakan nama variabel **merepresentasikan nilai dari variabel** tersebut.

```
1 public class MyClass {  
2     int totalTransaction = 3;  
3     String username = "slc";  
4     double averageScore = 95.67;  
5 }
```

Kode 13. *Naming Convention* untuk Variabel

f. **Variable Konstan** (*Constant*)

- Nama variabel konstan biasanya ditulis menggunakan ***screaming snake case***.
- Usahakan nama variable konstan **merepresentasikan nilai dari variabel** tersebut.

```
1 public class MyClass {  
2     final int MAX_CUSTOMER = 50;  
3     final String USER_TABLE_NAME = "users";  
4     final double DEFAULT_GRAVITY = 9.8;  
5 }
```

Kode 14. *Naming Convention* untuk Variabel Konstan

Catatan:

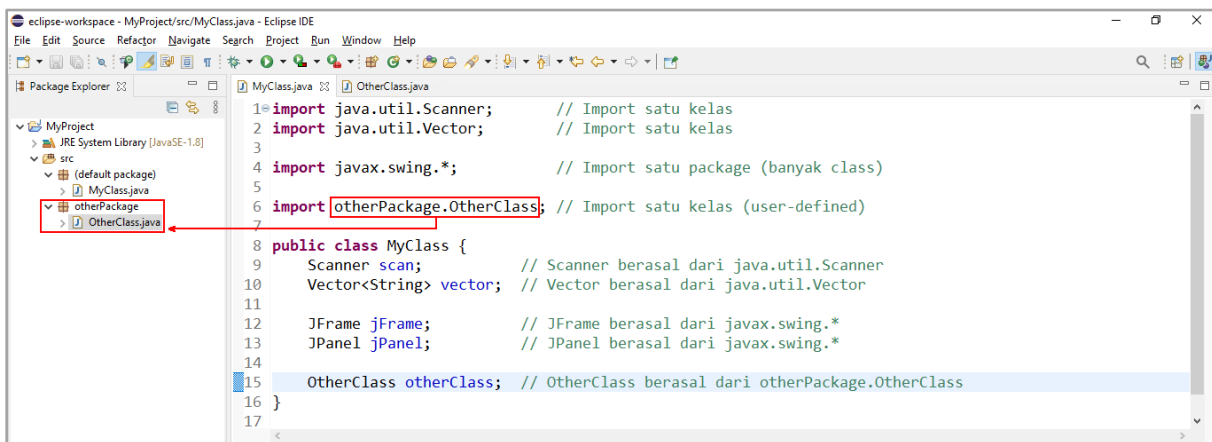
- ***Identifier*** merupakan **nama** yang digunakan untuk mengidentifikasi sesuatu. *Identifier* dapat berupa nama kelas, *interface*, *method*, variabel, dan variabel konstan.
- ***Pascal case*** adalah cara penulisan *identifier* **tanpa menggunakan spasi** dan setiap kata selalu **diawali dengan huruf besar**.
- ***Camel case*** adalah cara penulisan *identifier* **tanpa menggunakan spasi** dan **kata pertama diawali dengan huruf kecil** dan kata selanjutnya ditulis dengan huruf besar.
- ***Screaming snake case*** adalah cara penulisan *identifier* di mana **spasi diganti dengan simbol garis bawah (_)** dan setiap huruf ditulis dalam huruf besar.

F. Imports

Ketika lingkup sebuah *project* sudah besar, pengembang aplikasi tidak mungkin menuliskan semua logika pemrograman pada sebuah *file*. Hal ini dikarenakan *file* akan menjadi terlalu panjang dan lebih sulit untuk dipahami oleh pengembang aplikasi lainnya. Terlebih lagi, biasanya bahasa pemrograman Java terikat kuat dengan konsep **Object-Oriented Programming (OOP)** yang membutuhkan banyak *file* kelas. Oleh karena itu, Java telah menyediakan kata kunci (*keyword*) untuk memberitahu *compiler* apabila *file* kelas tidak dapat ditemukan dalam *library java.lang.package*, yaitu **import**.

Format : `import <PackagePath>;`

Format 4. *Import*



Kode 15. Penggunaan *Import*

Pertama, dideklarasikan variabel dengan tipe data **kelas Scanner** (baris ke-9), sebuah kelas yang disediakan Java untuk melakukan *input* ke aplikasi. Kelas tersebut di-*import* dari **package java.util.Scanner**. Kedua, dideklarasikan sebuah variabel dengan tipe data **kelas Vector** (baris ke-10), sebuah kelas yang disediakan Java untuk menyimpan banyak data seperti array. Kelas tersebut di-*import* dari **package java.util.Vector**.

Ketiga, dideklarasikan sebuah variabel dengan tipe data **kelas JFrame dan JPanel** (baris ke-12 s/d 13). Kelas tersebut di-*import* dari ***package javax.swing.****, sebuah *package* yang berisikan *library* yang diperlukan untuk *Graphical User Interface* (GUI). **Simbol bintang (*)** pada *import* berguna untuk memberitahu *compiler* untuk **meng-import semua library** yang ada di dalamnya. Cara ini **kurang disarankan** apabila kelas yang digunakan hanya beberapa saja, karena akan berdampak pada performa aplikasi.

Keempat, dideklarasikan sebuah variabel dengan tipe data kelas *OtherClass* (baris ke-15). Kelas tersebut di-*import* dari ***package otherPackage.OtherClass***, sebuah kelas yang dideklarasikan sendiri oleh pengembang aplikasi. Kata kunci *import* dapat juga berlaku untuk ***package* atau kelas yang dibuat sendiri** oleh pengembang aplikasi.

G. Output

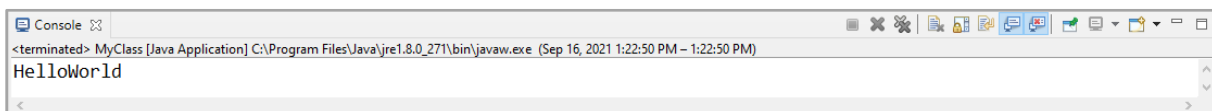
Dalam bahasa pemrograman Java, pengembang aplikasi tidak perlu membuat *method* untuk melakukan *output* secara manual karena *method* tersebut telah disediakan. Ada tiga buah *method* yang dapat digunakan, yaitu ***System.out.print()***, ***System.out.println()***, dan ***System.out.printf()***.

a. *System.out.print()*

Method ini berguna untuk melakukan **perintah *output* tanpa diakhiri dengan baris baru**. Terdapat satu parameter yang dibutuhkan oleh *method* ini, yaitu variabel atau nilai dari tipe data primitif yang akan di tampilkan ke *window console*.

```
3 public static void main(String[] args) {  
4     System.out.print("Hello");  
5     System.out.print("World");  
6 }
```

Kode 16. *Output* Menggunakan *System.out.print()*



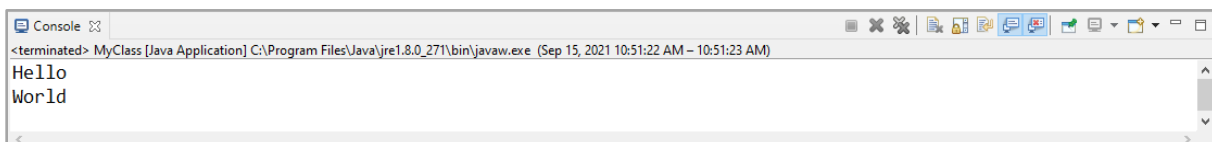
Hasil 2. *Output* Menggunakan *System.out.println()*

b. *System.out.println()*

Method ini berguna untuk melakukan **perintah *output* diakhiri dengan baris baru**. Terdapat satu parameter yang dibutuhkan oleh *method* ini, yaitu variabel atau nilai dari tipe data primitif yang akan di tampilkan ke *window console*.

```
3 public static void main(String[] args) {  
4     System.out.println("Hello");  
5     System.out.println("World");  
6 }
```

Kode 17. *Output* Menggunakan *System.out.println()*



Hasil 3. *Output* Menggunakan *System.out.println()*

c. *System.out.printf()*

Method ini berguna untuk **melakukan perintah *output* menggunakan format tertentu**. Terdapat dua parameter yang dibutuhkan oleh *method* ini, yaitu format *output* dan argumen yang dibutuhkan oleh format tersebut. Secara umum, sintaks untuk **format *output*** dapat dituliskan seperti berikut.

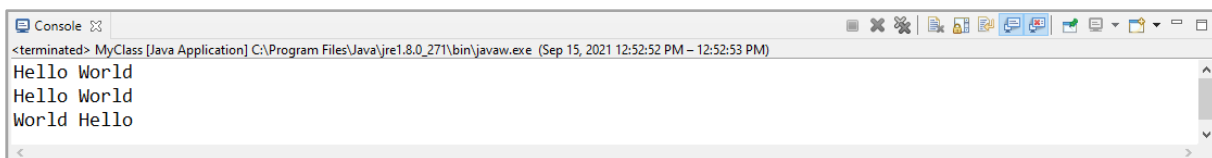
Format : %[Index\$][Flag][Width][.Presisi]<TipeData>

Format 5. *Output* Menggunakan *System.out.printf()*

Index merupakan opsi format **tidak wajib** yang digunakan untuk **menentukan posisi** dari sebuah argumen. Opsi format ini harus diakhiri dengan **simbol dolar (\$)**. Apabila kita tidak menggunakan opsi format ini, Java akan secara otomatis memasang argumen ke format *output* secara berurutan.

```
3 public static void main(String[] args) {  
4     System.out.printf("%s %s\n", "Hello", "World");           // Tanpa menggunakan format index  
5     System.out.printf("%1$s %2$s\n", "Hello", "World");       // Menggunakan opsi format index  
6     System.out.printf("%2$s %1$s\n", "Hello", "World");       // Menggunakan opsi format index  
7 }
```

Kode 18. *Output* Menggunakan *System.out.printf()* dengan Opsi *Index*



```
<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 12:52:52 PM - 12:52:53 PM)  
Hello World  
Hello World  
World Hello
```

Hasil 4. *Output* Menggunakan *System.out.printf()* dengan Opsi *Index*

Catatan:

- **Simbol persen (%)** akan ditandai sebagai **awal dari *output* dengan format**. Apabila ingin **mengeluarkan *output* simbol persen (%)**, gunakan **%%**.
- Pastikan **tidak ada karakter spasi** di dalam sebuah format.
- Untuk **membuat baris baru** dapat menggunakan **%n**.

Flag merupakan opsi format **tidak wajib** yang digunakan untuk **mengganti format output**. Berikut adalah jenis-jenis *flag* yang dapat digunakan.

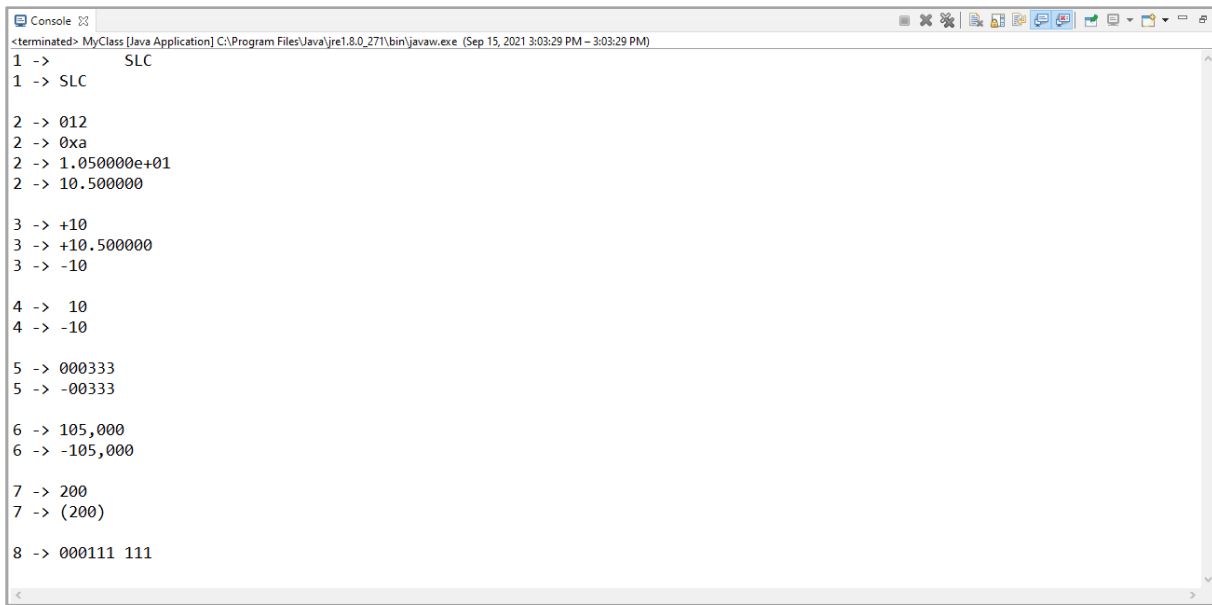
Flag	Deskripsi								
'.'	Membuat format teks menjadi rata kiri. Apabila tidak ada <i>flag</i> ini, format teks menjadi rata kanan.								
#	Mengubah format teks sesuai dengan tipe datanya. Penjelasan untuk masing-masing tipe data dapat dilihat pada tabel di bawah ini.								
	<table><tr><td>%o</td><td>Bilangan oktal ditambahi awalan "0".</td></tr><tr><td>%x, %X</td><td>Bilangan heksadesimal ditambahi awalan "0x" atau "0X".</td></tr><tr><td>%e, %E</td><td>Bilangan desimal akan diformat menjadi notasi eksponensial.</td></tr><tr><td>%f</td><td>Bilangan desimal akan diformat menjadi notasi desimal.</td></tr></table>	%o	Bilangan oktal ditambahi awalan "0".	%x, %X	Bilangan heksadesimal ditambahi awalan "0x" atau "0X".	%e, %E	Bilangan desimal akan diformat menjadi notasi eksponensial.	%f	Bilangan desimal akan diformat menjadi notasi desimal.
	%o	Bilangan oktal ditambahi awalan "0".							
	%x, %X	Bilangan heksadesimal ditambahi awalan "0x" atau "0X".							
	%e, %E	Bilangan desimal akan diformat menjadi notasi eksponensial.							
%f	Bilangan desimal akan diformat menjadi notasi desimal.								
'+'	Menambahkan tanda positif (+) di awal untuk bilangan desimal positif.								
' '	Menambahkan spasi di awal untuk bilangan desimal positif.								
'0'	Menambahkan angka nol (0) sebanyak opsi panjang di awal untuk bilangan bulat.								
','	Menambahkan separator simbol koma untuk bilangan bulat.								
('	Mengubah tanda negatif menjadi tanda kurung untuk bilangan bulat negatif.								
'<'	Mengikuti nilai argumen yang sebelumnya.								

```

3 public static void main(String[] args) {
4     System.out.printf("1 -> %10s\n", "SLC"); // Tanpa menggunakan flag.
5     System.out.printf("1 -> %-10s\n", "SLC"); // Menggunakan flag "-".
6
7     System.out.printf("2 -> %#o\n", 10); // Menggunakan flag "#" untuk "%o"
8     System.out.printf("2 -> %#x\n", 10); // Menggunakan flag "#" untuk "%x" dan "%X"
9     System.out.printf("2 -> %e\n", 10.5); // Menggunakan flag "#" untuk "%e" dan "%E"
10    System.out.printf("2 -> %f\n", 10.5); // Menggunakan flag "#" untuk "%f"
11
12    System.out.printf("3 -> %d\n", 10); // Menggunakan flag "+" untuk bilangan positif
13    System.out.printf("3 -> %f\n", 10.5); // Menggunakan flag "+" untuk bilangan positif decimal
14    System.out.printf("3 -> %d\n", -10); // Menggunakan flag "+" untuk bilangan negatif
15
16    System.out.printf("4 -> % d\n", 10); // Menggunakan flag " " untuk bilangan positif
17    System.out.printf("4 -> % d\n", -10); // Menggunakan flag " " untuk bilangan negatif
18
19    System.out.printf("5 -> %06d\n", 333); // Menggunakan flag "0" untuk bilangan positif
20    System.out.printf("5 -> %06d\n", -333); // Menggunakan flag "0" untuk bilangan negatif
21
22    System.out.printf("6 -> %,d\n", 105000); // Menggunakan flag "," untuk bilangan positif
23    System.out.printf("6 -> %,d\n", -105000); // Menggunakan flag "," untuk bilangan negatif
24
25    System.out.printf("7 -> %(d\n", 200); // Menggunakan flag "(" untuk bilangan positif
26    System.out.printf("7 -> %(d\n", -200); // Menggunakan flag "(" untuk bilangan negatif
27
28    System.out.printf("8 -> %06d <d", 111); // Menggunakan flag "<"
29 }

```

Kode 19. Output Menggunakan System.out.printf() dengan Opsi Flag



```
<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 3:03:29 PM - 3:03:29 PM)
1 -> SLC
1 -> SLC

2 -> 012
2 -> 0xa
2 -> 1.050000e+01
2 -> 10.500000

3 -> +10
3 -> +10.500000
3 -> -10

4 -> 10
4 -> -10

5 -> 000333
5 -> -00333

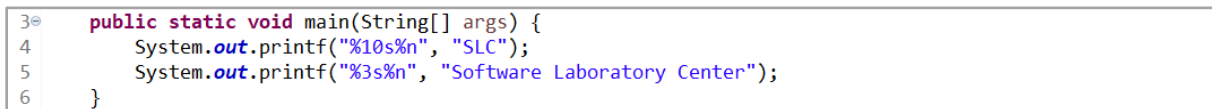
6 -> 105,000
6 -> -105,000

7 -> 200
7 -> (200)

8 -> 000111 111
```

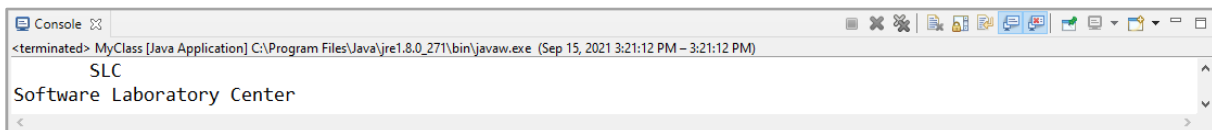
Hasil 5. Output Menggunakan *System.out.printf()* dengan Opsi *Flag*

Width merupakan opsi format **tidak wajib** yang digunakan untuk **memberikan jumlah karakter minimum** yang akan di *output*. Jika jumlah karakter tidak mencapai panjang yang ditentukan, sisanya akan diisi dengan spasi. Sedangkan, jika jumlah karakter melebihi panjang yang ditentukan, karakter tidak akan dipotong.



```
3 public static void main(String[] args) {
4     System.out.printf("%10s\n", "SLC");
5     System.out.printf("%3s\n", "Software Laboratory Center");
6 }
```

Kode 20. Output Menggunakan *System.out.printf()* dengan Opsi *Width*



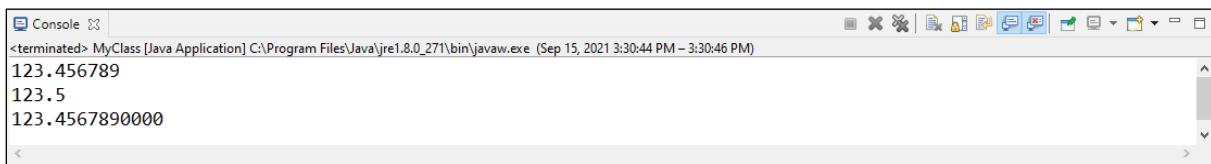
```
<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 3:21:12 PM - 3:21:12 PM)
SLC
Software Laboratory Center
```

Hasil 6. Output Menggunakan *System.out.printf()* dengan Opsi *Width*

Presisi merupakan opsi format **tidak wajib** yang digunakan untuk menentukan **jumlah angka di belakang koma**. Opsi ini berlaku untuk **tipe data desimal** (*float* dan *double*). Apabila opsi ini tidak diberikan, bilangan desimal akan memiliki *output* 6 angka di belakang koma.

```
3 public static void main(String[] args) {
4     System.out.printf("%f\n", 123.4567890);
5     System.out.printf("%.1f\n", 123.4567890);
6     System.out.printf("%.10f\n", 123.4567890);
7 }
```

Kode 21. *Output* Menggunakan *System.out.printf()* dengan Opsi Presisi



```
<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 3:30:44 PM - 3:30:46 PM)
123.456789
123.5
123.4567890000
```

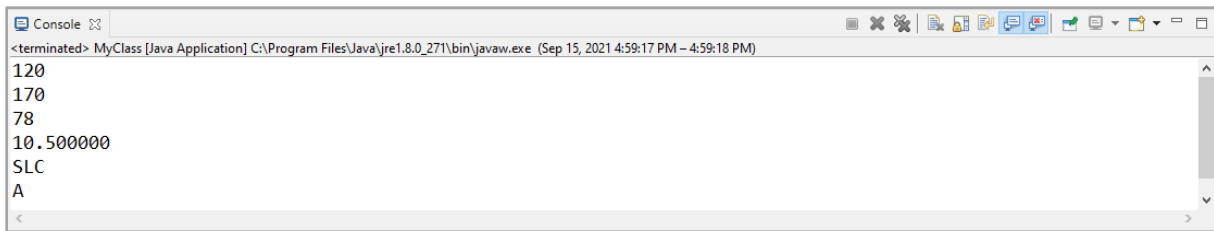
Hasil 7. *Output* Menggunakan *System.out.println()* dengan Opsi Presisi

Tipe data merupakan opsi format yang **wajib** digunakan untuk **memberitahu tipe data** yang akan di *output*. Untuk pilihan yang dapat digunakan, dapat dilihat pada tabel di bawah ini.

Tipe Data	Input	Deskripsi
%d	Signed Int	Bilangan integer positif dan negatif.
%u	Unsigned Int	Bilangan integer positif.
%o	Unsigned Int	Bilangan oktal.
%x, %X	Unsigned Int	Bilangan heksadesimal (huruf kecil dan huruf besar).
%f	Float	Bilangan desimal.
%s	String	<i>String</i> .
%c	Char	Karakter.
%p	Object	Identitas objek dalam bentuk heksadesimal.

```
3 public static void main(String[] args) {
4     System.out.printf("%d\n", 120);
5     System.out.printf("%o\n", 120);
6     System.out.printf("%x\n", 120);
7     System.out.printf("%f\n", 10.5);
8     System.out.printf("%s\n", "SLC");
9     System.out.printf("%c\n", 'A');
10 }
```

Kode 22. *Output* Menggunakan *System.out.printf()* dengan Opsi Tipe Data



```
<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 4:59:17 PM - 4:59:18 PM)
120
170
78
10.500000
SLC
A
```

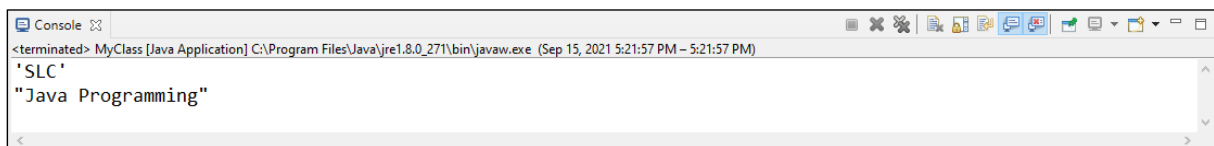
Hasil 8. Output Menggunakan *System.out.printf()* dengan Opsi Tipe Data

Karakter yang didahului oleh **simbol garis miring terbalik (\)** disebut juga ***escape sequence***. Berikut adalah *escape sequence* yang dimiliki oleh Java.

Escape	Deskripsi
\t	Membuat simbol tab secara vertikal.
\b	Membuat simbol backspace.
\n	Membuat baris baru.
\'	Membuat simbol kutip satu (').
\"	Membuat simbol kutip dua (").
\\	Membuat simbol garis miring terbalik (\).

```
3 public static void main(String[] args) {
4     System.out.printf("\'SLC'\n");
5     System.out.printf("\"Java Programming\"");
6 }
```

Kode 23. Penggunaan *Escape Sequence*



```
<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 5:21:57 PM - 5:21:57 PM)
'SLC'
"Java Programming"
```

Hasil 9. Penggunaan *Escape Sequence*

H. Input

Sama seperti dengan operasi *output*, bahasa pemrograman Java telah menyediakan sebuah kelas yang digunakan untuk menerima *input* yang pengguna aplikasi berikan ke program. Kelas tersebut bernama **Scanner**. Secara umum, sintaks yang digunakan untuk membuat objek Scanner adalah sebagai berikut.

```
Format : Scanner <NamaVariabel> = new Scanner(System.in);
```

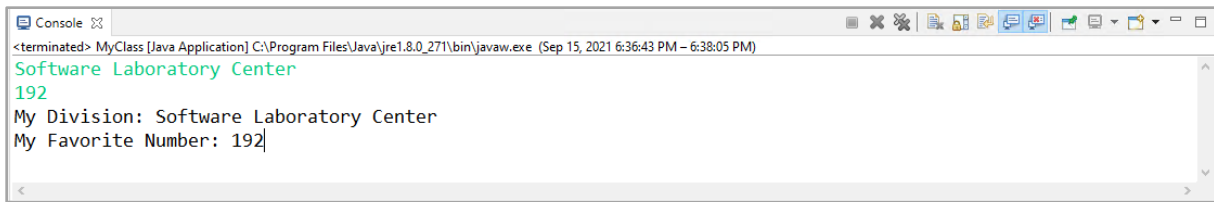
Format 6. Input Menggunakan Scanner

Ketika objek Scanner telah dibuat, kita dapat menggunakan *method-method* yang telah disediakan oleh kelas tersebut. Berikut adalah beberapa **method** yang bisa digunakan berdasarkan tipe datanya.

- nextByte()**. *Method* ini digunakan untuk mendapatkan *input* dalam tipe data **byte**.
- nextShort()**. *Method* ini digunakan untuk mendapatkan *input* dalam tipe data **short**.
- nextInt()**. *Method* ini digunakan untuk mendapatkan *input* dalam tipe data **int**.
- nextLong()**. *Method* ini digunakan untuk mendapatkan *input* dalam tipe data **long**.
- nextFloat()**. *Method* ini digunakan untuk mendapatkan *input* dalam tipe data **float**.
- nextDouble()**. *Method* ini digunakan untuk mendapatkan *input* dalam tipe data **double**.
- nextLine()**. *Method* ini digunakan untuk mendapatkan *input* sampai dengan **enter** dalam tipe data **String**.

```
1 import java.util.Scanner;
2
3 public class MyClass {
4
5     public static void main(String[] args) {
6         Scanner scan = new Scanner(System.in);
7
8         String division = scan.nextLine();
9         int favoriteNumber = scan.nextInt();
10        scan.nextLine();
11
12        System.out.println("My Division: " + division);
13        System.out.println("My Favorite Number: " + favoriteNumber);
14    }
15
16 }
```

Kode 24. Input Menggunakan Scanner



```
<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 6:36:43 PM - 6:38:05 PM)
Software Laboratory Center
192
My Division: Software Laboratory Center
My Favorite Number: 192
```

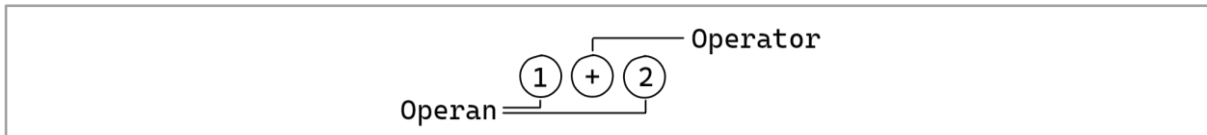
Hasil 10. *Input Menggunakan Scanner*

Catatan:

- Untuk menggunakan kelas **Scanner**, jangan lupa untuk melakukan **import kelas** dari **java.util.Scanner**.
- Setelah menggunakan **method input** yang berhubungan dengan **angka** (nextByte, nextShort, nextInt, nextLong, nextFloat, nextDouble), jangan lupa untuk **menambahkan method nextLine**. Hal ini dikarenakan **input** yang berhubungan dengan angka tidak menghilangkan karakter **enter** sehingga perlu dihilangkan dengan menggunakan nextLine.
- Untuk **menggabungkan dua buah output**, dapat digunakan **tanda tambah (+)** seperti yang dicontohkan pada Hasil 10.

I. Operator

Operator adalah **simbol** yang digunakan untuk mendapatkan hasil dari proses operasi. **Operan** adalah **nilai** yang digunakan dalam proses operasi. Sebagai contoh, apabila terdapat operasi $1 + 2$, maka operator-nya adalah tanda $+$, sedangkan operan-nya adalah angka 1 dan 2.



Gambar 13. Operator dan Operan

Berdasarkan **jenisnya**, operator dapat dibagi menjadi empat jenis, yaitu **operator aritmatika**, **operator *bitwise***, **operator perbandingan**, dan **operator logika**.

a. Operator Aritmatika

Operator aritmatika merupakan **operator dasar** yang biasanya digunakan untuk mengkalkulasikan masalah matematika. Tabel di bawah ini adalah operator dasar yang dimiliki oleh bahasa pemrograman Java.

Operator	Shorthand	Nama	Deskripsi
+	+=	Penjumlahan	Menambahkan dua buah nilai operan.
-	-=	Pengurangan	Mengurangi nilai operan pertama dengan operan kedua.
*	*=	Perkalian	Mengalikan dua buah nilai operan.
/	/=	Pembagian	Membagi nilai operan pertama dengan operan kedua.
%	%=	Modulus	Hasil dari pembagian dua buah operan.
++		<i>Increment</i>	Menaikan nilai operan sebanyak satu.
--		<i>Decrement</i>	Menurunkan nilai operan sebanyak satu.

Operator aritmatika ***increment*** terbagi menjadi dua jenis, yaitu ***pre-increment*** dan ***post-increment***. ***Pre-increment*** adalah jenis operator dimana nilai operan **ditambahkan satu sebelum** menjalankan sebuah proses. Sedangkan, ***post-increment*** adalah jenis operator dimana nilai operan **ditambahkan satu setelah** menjalankan sebuah proses.

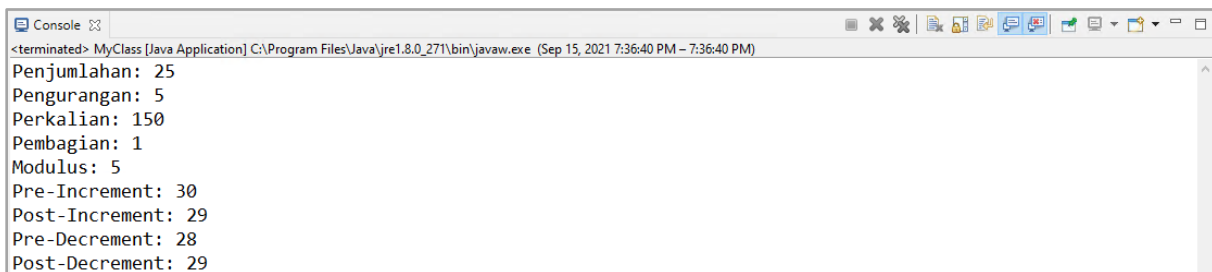
Sama seperti operator aritmatika *increment*, operator aritmatika **decrement** juga terbagi menjadi dua jenis, yaitu **pre-decrement** dan **post-decrement**. **Pre-decrement** adalah jenis operator dimana nilai operan **dikurangi satu sebelum** menjalankan sebuah proses. Sedangkan, **post-decrement** adalah jenis operasi dimana nilai operan **dikurangi satu setelah** menjalankan sebuah proses.

```

3 public static void main(String[] args) {
4     int a = 15, b = 10, c = 29;
5
6     System.out.println("Penjumlahan: " + (a + b));
7     System.out.println("Pengurangan: " + (a - b));
8     System.out.println("Perkalian: " + (a * b));
9     System.out.println("Pembagian: " + (a / b));
10    System.out.println("Modulus: " + (a % b));
11
12    c = 29;
13    System.out.println("Pre-Increment: " + (++c));
14    c = 29;
15    System.out.println("Post-Increment: " + (c++));
16    c = 29;
17    System.out.println("Pre-Decrement: " + (--c));
18    c = 29;
19    System.out.println("Post-Decrement: " + (c--));
20 }

```

Kode 25. Operator Aritmatika



```

<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 7:36:40 PM - 7:36:40 PM)
Penjumlahan: 25
Pengurangan: 5
Perkalian: 150
Pembagian: 1
Modulus: 5
Pre-Increment: 30
Post-Increment: 29
Pre-Decrement: 28
Post-Decrement: 29

```

Hasil 11. Operator Aritmatika

Catatan:

Operator **shorthand** merupakan **gabungan** antara **operator aritmatika** atau **operator bitwise** dengan **operator assignment**. Tujuannya adalah mempersingkat pengembang aplikasi dalam membuat kode.

b. Operator Bitwise

Dalam bahasa pemrograman Java, setiap bilangan bulat memiliki bentuk biner.

Bentuk biner terdiri dari **dua buah angka**, yaitu **0** dan **1**. Misalkan, bentuk biner dari

bilangan bulat enam adalah 0110. Untuk cara menghitungnya perhatikan **Error!**
Reference source not found..

$$\begin{array}{cccc}
 \textcircled{0} & \textcircled{1} & \textcircled{1} & \textcircled{0} \\
 2^3 & 2^2 & 2^1 & 2^0 \\
 8 & 4 & 2 & 1 \\
 = (0 \times 8) + (1 \times 4) + (1 \times 2) + (0 \times 1) \\
 = 0 + 4 + 2 + 0 \\
 = 6
 \end{array}$$

Gambar 14. Menghitung Bilangan Biner ke Bilangan Bulat

Untuk membantu memanipulasi bentuk biner dari sebuah bilangan bulat dibuatlah **operator bitwise**. Tabel di bawah ini adalah operator *bitwise* yang dimiliki oleh bahasa pemrograman Java.

Operator	Shorthand	Nama	Deskripsi
&	&=	AND	Apabila bit pada kedua operan bernilai 1, maka hasilnya 1. Apabila bit pada kedua operan ada yang bernilai 0, maka hasilnya 0.
	=	OR	Apabila bit pada kedua operan ada yang bernilai 1, maka hasilnya 1. Apabila bit pada kedua operan bernilai 0, maka hasilnya 0.
^	^=	XOR	Apabila bit pada kedua operan bernilai berlawanan, maka hasilnya 1. Apabila bit pada kedua operan bernilai sama, maka hasilnya 0.
~	~=	NOT	Apabila bit pada sebuah operan bernilai 1, maka hasilnya 0. Apabila bit pada sebuah operan bernilai 0, maka hasilnya 1.
<<	<<=	Left Shift	Menggeser bit ke kiri sesuai dengan jumlah yang diberikan.
>>	>>=	Right Shift	Menggeser bit ke kanan sesuai dengan jumlah yang diberikan.

<p>Operasi AND</p> <pre> a = 01001011 b = 10011101 ----- & a & b = 00001001 </pre>	<p>Operasi OR</p> <pre> a = 01001011 b = 10011101 ----- a b = 11011111 </pre>	<p>Operasi XOR</p> <pre> a = 01001011 b = 10011101 ----- ^ a ^ b = 11010110 </pre>
<p>Operasi NOT</p> <pre> c = 00101101 ----- ~ ~c = 11010010 </pre>	<p>Operasi Left Shift</p> <pre> c = 10101101 ----- << 2 c = 1010110100 </pre>	<p>Operasi Right Shift</p> <pre> c = 10101101 ----- >> 2 c = 101011 </pre>

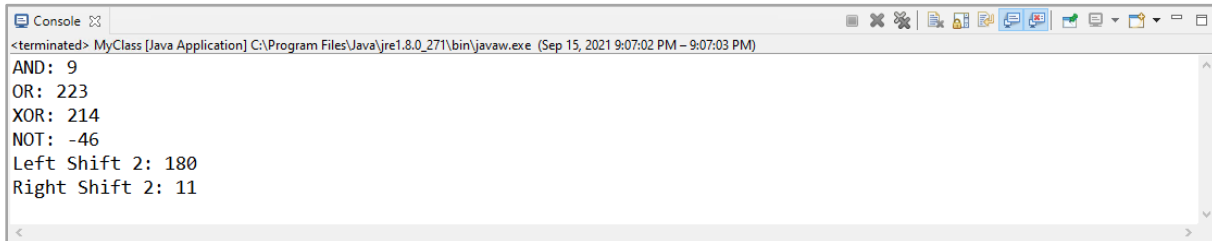
Gambar 15. Simulasi Operator Bitwise

```

3 public static void main(String[] args) {
4     int a = 75, b = 157, c = 45;
5
6     System.out.println("AND: " + (a & b));
7     System.out.println("OR: " + (a | b));
8     System.out.println("XOR: " + (a ^ b));
9     System.out.println("NOT: " + (~c));
10    System.out.println("Left Shift 2: " + (c << 2));
11    System.out.println("Right Shift 2: " + (c >> 2));
12 }

```

Kode 26. Operator *Bitwise*



```

<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 9:07:02 PM - 9:07:03 PM)
AND: 9
OR: 223
XOR: 214
NOT: -46
Left Shift 2: 180
Right Shift 2: 11

```

Hasil 12. Operator *Bitwise*

c. Operator Perbandingan

Operator perbandingan merupakan operator yang digunakan untuk **membandingkan dua buah operan**. Biasanya operator ini digunakan pada seleksi atau repetisi sebagai kondisi. Tabel di bawah ini adalah operator perbandingan yang dimiliki oleh bahasa pemrograman Java.

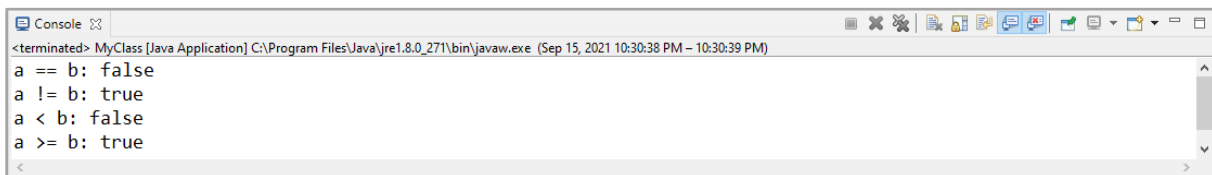
<i>Operator</i>	<i>Deskripsi</i>
<code>==</code>	Apabila kedua nilai operan sama, maka bernilai <i>true</i> . Apabila nilai operan tidak sama, maka bernilai <i>false</i> .
<code>!=</code>	Apabila kedua nilai operan sama, maka bernilai <i>false</i> . Apabila nilai operan tidak sama, maka bernilai <i>true</i> .
<code>></code>	Apabila operan kiri lebih besar dari operan kanan, maka bernilai <i>true</i> . Apabila operan kiri lebih kecil dari operan kanan, maka bernilai <i>false</i> .
<code><</code>	Apabila operan kiri lebih besar dari operan kanan, maka bernilai <i>false</i> . Apabila operan kiri lebih kecil dari operan kanan, maka bernilai <i>true</i> .
<code>>=</code>	Apabila operan kiri lebih besar atau sama dengan dari operan kanan, maka bernilai <i>true</i> . Apabila operan kiri lebih kecil dari operan dari kanan, maka bernilai <i>false</i> .
<code><=</code>	Apabila operan kiri lebih besar atau sama dengan dari operan dari kanan, maka bernilai <i>false</i> . Apabila operan kiri lebih kecil dari operan dari kanan, maka bernilai <i>true</i> .

```

3 public static void main(String[] args) {
4     int a = 75, b = 57;
5
6     System.out.println("a == b: " + (a == b));
7     System.out.println("a != b: " + (a != b));
8     System.out.println("a < b: " + (a < b));
9     System.out.println("a >= b: " + (a >= b));
10 }

```

Kode 27. Operator Perbandingan



```

<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 10:30:38 PM - 10:30:39 PM)
a == b: false
a != b: true
a < b: false
a >= b: true

```

Hasil 13. Operator Perbandingan

d. Operator Logika

Operator logika merupakan operator yang digunakan untuk **menggabungkan dua atau lebih operator perbandingan**. Biasanya operator ini digunakan pada seleksi atau repetisi sebagai penggabung dua atau lebih kondisi. Tabel di bawah ini adalah operator logika yang dimiliki oleh bahasa pemrograman Java.

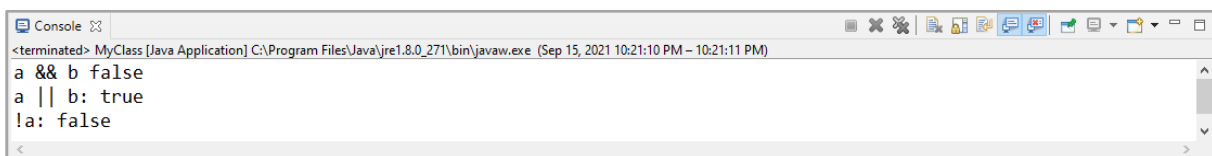
Operator	Deskripsi
&&	Apabila kedua kondisi benar, maka nilainya <i>true</i> . Apabila salah satu dari kedua kondisi tersebut salah, maka nilainya <i>false</i> .
	Apabila salah satu dari kedua kondisi tersebut benar, maka nilainya <i>true</i> . Apabila kedua kondisi salah, maka nilainya <i>false</i> .
!	Apabila sebuah kondisi bernilai <i>true</i> , maka nilainya <i>false</i> . Apabila sebuah kondisi bernilai <i>false</i> , maka nilainya <i>true</i> .

```

3 public static void main(String[] args) {
4     boolean a = true, b = false;
5
6     System.out.println("a && b " + (a && b));
7     System.out.println("a || b: " + (a || b));
8     System.out.println("!a: " + (!a));
9 }

```

Kode 28. Operator Logika



```

<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 15, 2021 10:21:10 PM - 10:21:11 PM)
a && b false
a || b: true
!a: false

```

Hasil 14. Operator Logika

J. Seleksi

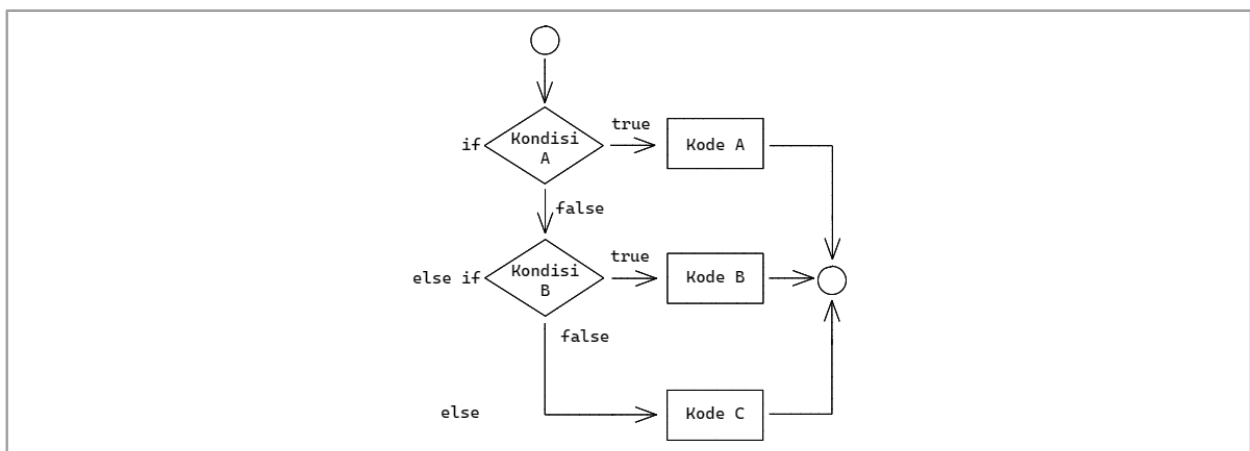
Dalam bahasa pemrograman, **struktur kendali seleksi** merupakan sintaks yang memungkinkan program dapat memiliki beberapa pilihan. Apabila pilihan yang dicek benar, maka kode yang ada di dalam *scope* tersebut akan dijalankan. Ada dua jenis seleksi yang ada di Java, yaitu ***if ... else ...*** dan ***switch ... case***

a. *if ... else ...*

Seleksi *if ... else ...* dapat dibagi menjadi **tiga bagian utama**, yaitu ***if***, ***else if***, dan ***else***. Bagian pertama (*if*) akan mengecek apakah kondisi yang diberikan terpenuhi atau tidak. Apabila terpenuhi, maka kode akan dijalankan dan seleksi telah selesai. Apabila tidak terpenuhi, akan dilanjutkan ke bagian kedua (*else if*). Sama seperti bagian pertama, hanya bedanya ketika kondisi tidak terpenuhi akan dilanjutkan ke bagian terakhir (*else*). Ketika mencapai bagian ini artinya semua kondisi yang diberikan di atas tidak terpenuhi sehingga kode yang ada di dalam *else* akan langsung dijalankan.

```
Format : if ( <Kondisi-1> ) { <Kode-1> }  
        else if ( <Kondisi-2> ) { <Kode-2> }  
        else { <Kode-3> }
```

Format 7. Seleksi Menggunakan *if ... else ...*



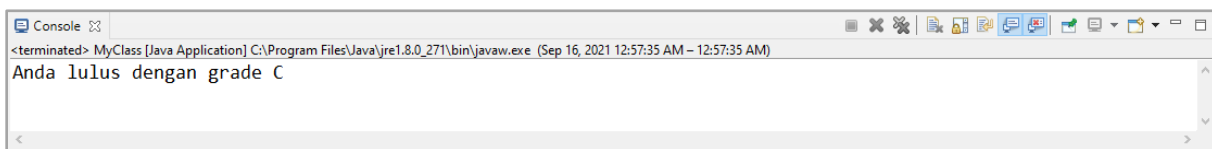
Gambar 16. Diagram Alur Seleksi Menggunakan *if ... else ...*

```

3 public static void main(String[] args) {
4     int myScore = 87;
5
6     if (myScore >= 95 && myScore <= 100) {
7         System.out.println("Anda lulus dengan grade A");
8     } else if (myScore >= 90 && myScore < 95) {
9         System.out.println("Anda lulus dengan grade B");
10    } else if (myScore >= 85 && myScore < 90) {
11        System.out.println("Anda lulus dengan grade C");
12    } else if (myScore >= 80 && myScore < 85) {
13        System.out.println("Anda lulus dengan grade D");
14    } else {
15        System.out.println("Anda tidak lulus");
16    }
17 }

```

Kode 29. Seleksi Menggunakan *if ... else ...*



Hasil 15. Seleksi Menggunakan *if ... else ...*

Dideklarasikan sebuah variabel bernama *myScore* dengan nilai 87. Nilai tersebut akan dicek ke dalam kondisi yang pertama, yaitu apakah 87 lebih besar atau sama dengan 95 dan lebih kecil atau sama dengan 100. Jawabannya tidak karena 87 lebih kecil dibandingkan 95. Oleh karena itu, kode yang ada dalamnya tidak dijalankan dan beralih ke kondisi yang kedua, yaitu apakah 87 lebih besar atau sama dengan 90 dan lebih kecil dari 95. Jawabannya tidak karena 87 lebih kecil dibandingkan 90. Program akan mengecek kondisi selanjutnya, yaitu apakah 87 lebih besar atau sama dengan 80 dan lebih kecil dari 90. Jawabannya iya. Oleh karena program akan menjalankan kode yang ada di dalamnya, yaitu mengeluarkan *output* berisikan “Anda lulus dengan *grade C*” dan kondisi selanjutnya tidak akan dicek.

b. **switch ... case ...**

Seleksi *switch ... case ...* digunakan untuk **memberikan beberapa kondisi** terhadap **satu buah variabel saja**. Seleksi jenis ini tidak seluas cakupan seleksi jenis sebelumnya. Tipe data yang diperbolehkan untuk seleksi ini hanya **tipe data primitif yang berukuran kecil** (*byte*, *char*, *short*, dan *int*). Jadi, tipe data primitif yang berukuran besar (*long*, *float*, dan *double*) tidak dapat digunakan.

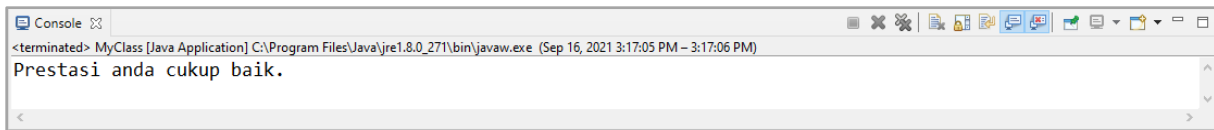
Pada seleksi *switch...case...*, terdapat **tiga kata kunci (keyword)** yang akan sering digunakan, yaitu ***case***, ***break***, dan ***default***. Kata kunci *case* digunakan untuk memberikan kondisi dari variabel yang ditentukan pada kata kunci *switch*. Kata kunci *break* digunakan untuk menandakan bahwa setelah mengeksekusi kode di dalam sebuah kondisi, maka kode yang ada di bawahnya tidak akan dijalankan lagi. Kata kunci *default* digunakan untuk memberikan kode yang dijalankan apabila semua kondisi yang diberikan tidak terpenuhi.

```
Format : switch ( <NamaVariabel> ) {  
        case <Kondisi-1>: { <Kode-1>; break; }  
        [ case <Kondisi-2>: { <Kode-2>; break; } ]  
        ...  
        [ case <Kondisi-n>: { <Kode-n>; break; } ]  
        [ default: { <Kode>; break; } ]  
    }
```

Gambar 17. Seleksi Menggunakan *switch .. case ...*

```
3 public static void main(String[] args) {  
4     char myGrade = 'B';  
5  
6     switch (myGrade) {  
7     case 'A': {  
8         System.out.println("Prestasi anda sangat baik.");  
9         break;  
10    }  
11    case 'B': {  
12        System.out.println("Prestasi anda cukup baik.");  
13        break;  
14    }  
15    case 'C': {  
16        System.out.println("Prestasi perlu ditingkatkan.");  
17        break;  
18    }  
19    default: {  
20        System.out.println("Grade yang dimasukan tidak diproses.");  
21        break;  
22    }  
23 }  
24 }
```

Kode 30. Seleksi Menggunakan *switch ... case ...*



Hasil 16. Seleksi Menggunakan *switch ... case ...*

Dideklarasikan sebuah variabel bernama *myGrade* dengan nilai 'B'. Nilai tersebut akan dicek ke dalam kondisi yang pertama, yaitu apakah 'B' sama dengan 'A'. Jawabannya tidak. Oleh karena itu, kode yang ada dalamnya tidak dijalankan dan beralih ke kondisi yang kedua, yaitu apakah 'B' sama dengan 'B'. Jawabannya iya. Oleh karena program akan menjalankan kode yang ada di dalamnya, yaitu mengeluarkan *output* "Prestasi Anda cukup baik." dan kondisi selanjutnya tidak akan dicek.

K. Repetisi

Dalam bahasa pemrograman, **struktur kendali repetisi** merupakan sintaks yang memungkinkan program untuk **mengulangi suatu kode** sampai dengan kondisi yang diberikan. Terdapat tiga jenis repetisi yang ada di dalam Java, yaitu ***for***, ***while***, and ***do ... while***.

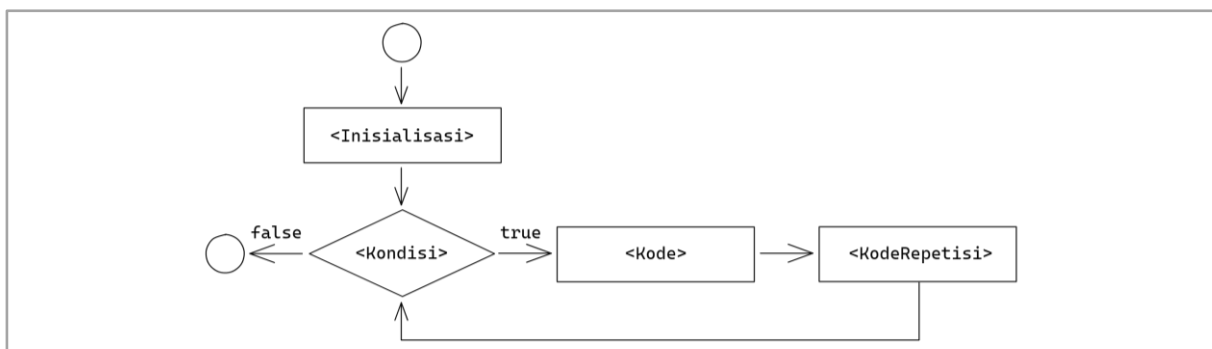
a. *for*

Repetisi *for* merupakan struktur kendali perulangan yang menerima **tiga buah parameter**, yaitu **inisialisasi variabel**, **kondisi perulangan**, dan **kode repetisi** yang dilakukan setelah satu kali repetisi selesai dilakukan. Biasanya kode ini merupakan *increment* atau *decrement* dari variabel yang diinisialisasikan. Ketiga parameter ini bersifat *optional*.

Cara kerja dari repetisi *for* adalah dengan menginisialisasikan variabel yang akan digunakan dalam repetisi tersebut. Kemudian, kondisi tersebut akan dicek apakah kondisi tersebut terpenuhi atau tidak. Apabila terpenuhi, kode di dalam repetisi akan dijalankan. Setelah itu, akan dijalankan kode yang ada di parameter ketiga dari repetisi *for*. Proses kembali lagi ke kondisi tersebut sampai kondisi tidak terpenuhi.

```
Format : for ( <Inisialisasi>; <Kondisi>; <KodeRepetisi> ) {  
        <Kode>  
    }
```

Format 8. Repetisi Menggunakan *for*



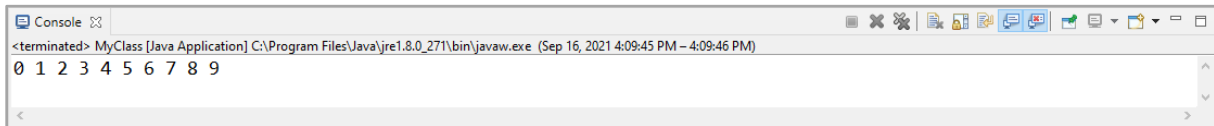
Gambar 18. Diagram Alur Repetisi Menggunakan *for*


```

3 public static void main(String[] args) {
4     for (int i = 0 ; i < 10 ; i++) {
5         System.out.print(i + " ");
6     }
7 }

```

Kode 31. Repetisi Menggunakan *for*



```

<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 16, 2021 4:09:45 PM - 4:09:46 PM)
0 1 2 3 4 5 6 7 8 9

```

Hasil 17. Repetisi Menggunakan *for*

b. *while*

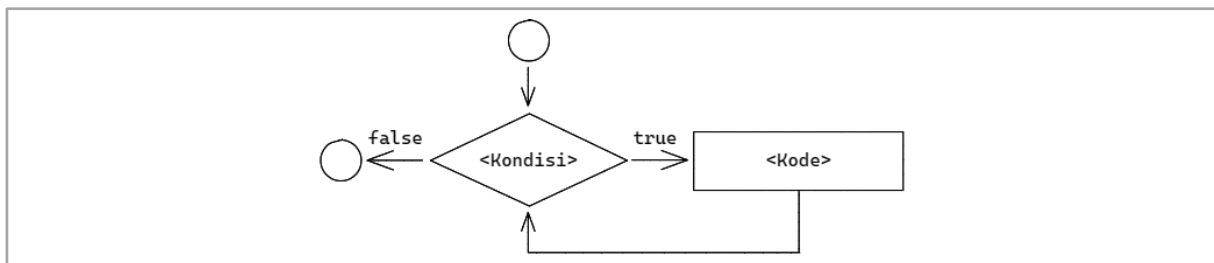
Fungsi dari repetisi *while* sama seperti dengan *for*, yaitu untuk mengulang suatu kode sampai kondisi tidak terpenuhi. Hanya saja **cara kerjanya lebih simpel** dibandingkan dengan repetisi *for*. Repetisi *while* **hanya mengecek apakah kondisi yang diberikan terpenuhi atau tidak**. Apabila terpenuhi, maka kode yang ada di dalam repetisi akan dijalankan sampai kondisi tidak terpenuhi lagi.

```

Format    : while ( <Kondisi> ) {
              <Kode>
            }

```

Format 9. Repetisi Menggunakan *while*



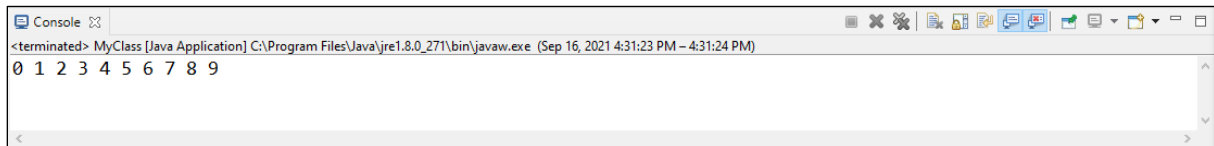
Gambar 19. Diagram Alur Repetisi Menggunakan *while*

```

3 public static void main(String[] args) {
4     int i = 0, j = 0;
5
6     while (i < 10) {
7         System.out.print(i++ + " ");
8     }
9     System.out.println();
10
11    while (j < 0) {
12        System.out.print(j++ + " ");
13    }
14 }

```

Kode 32. Repetisi Menggunakan *while*



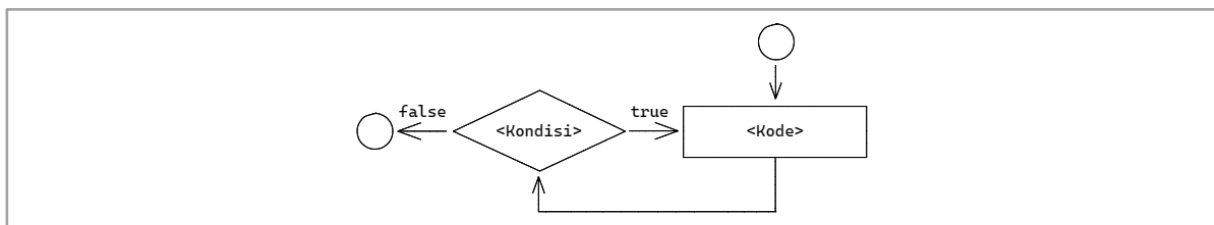
Hasil 18. Repetisi Menggunakan *while*

c. *do ... while*

Fungsi dari repetisi *do ... while* hampir sama seperti dengan *while*. Pada repetisi *while* kode bisa saja tidak dijalankan sama sekali. Namun, pada repetisi *do ... while* **kode akan dijalankan minimal satu kali**. Hal tersebut dikarenakan cara kerja *do ... while* yang menjalankan kode di dalam repetisi terlebih dahulu kemudian baru kondisinya dicek apakah memenuhi atau tidak.

Format : do {
 <Kode>
 } while (<Kondisi>);

Format 10. Repetisi Menggunakan *do ... while*



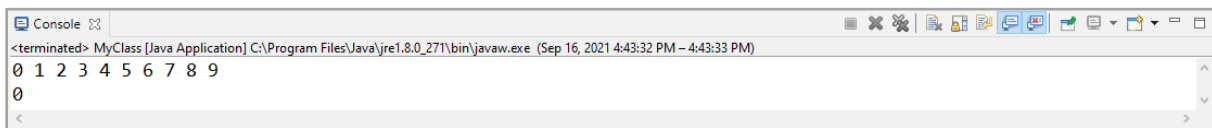
Gambar 20. Diagram Alur Menggunakan *do ... while*

```

3 public static void main(String[] args) {
4     int i = 0, j = 0;
5
6     do {
7         System.out.print(i++ + " ");
8     } while (i < 10);
9     System.out.println();
10
11    do {
12        System.out.print(j++ + " ");
13    } while (j < 0);
14 }

```

Kode 33. Repetisi Menggunakan *do ... while*



```

<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 16, 2021 4:43:32 PM - 4:43:33 PM)
0 1 2 3 4 5 6 7 8 9
0

```

Hasil 19. Repetisi Menggunakan *do ... while*

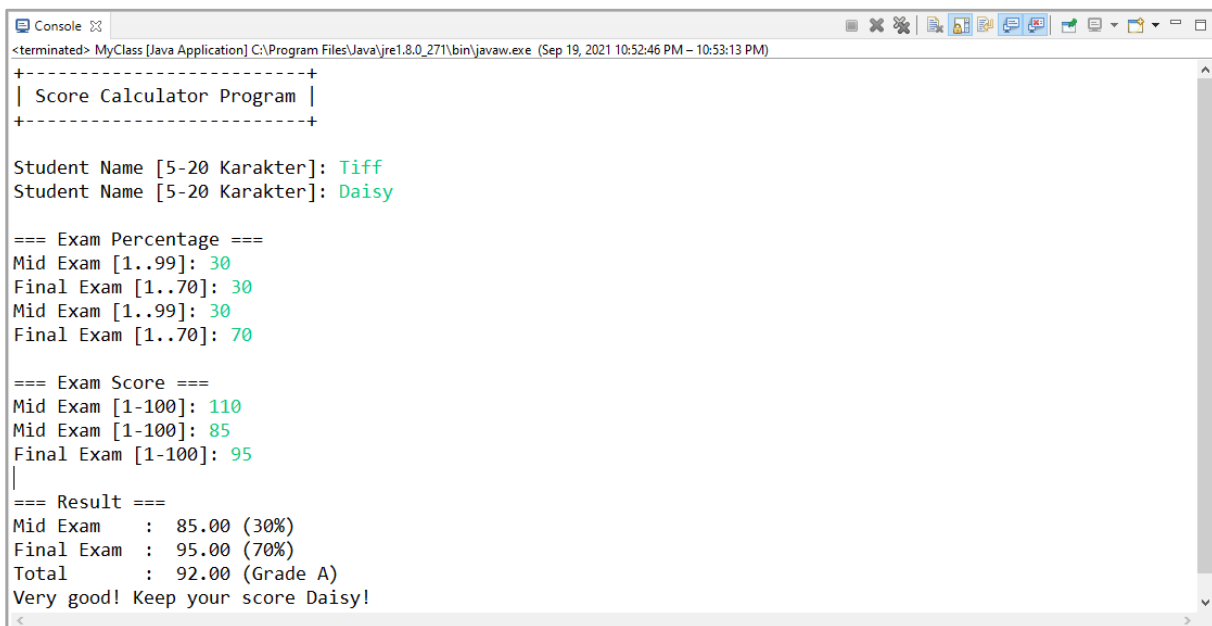
Pada repetisi **while yang kedua** (Kode 32. Repetisi Menggunakan *while* Kode 32), program **tidak menghasilkan output** apa pun dikarenakan kondisi yang diberikan tidak terpenuhi. Sedangkan, pada repetisi **do ... while yang kedua** (Kode 33), program **menghasilkan output 0** karena dijalankan minimal satu kali.

L. Latihan Soal

Buatlah sebuah program untuk **menghitung nilai mahasiswa** dengan kriteria sebagai berikut.

- Buatlah **input** dengan menggunakan kelas **Scanner** dari *java.util.Scanner*.
- Program akan dimulai dengan **meminta beberapa input** dan **melakukan beberapa validasi**.
 - **Input** pertama adalah **nama mahasiswa** antara **5 sampai 20 karakter** dengan tipe data **String**.
 - **Input** kedua adalah **persentase ujian tengah semester** antara **1 sampai 99** dengan tipe data **integer**.
 - **Input** ketiga adalah **persentase ujian akhir semester** antara **1 sampai selisih dari 100 dikurangi persentase ujian tengah semester** dengan tipe **integer**.
 - Apabila **jumlah persentase ujian tidak bernilai 100**, maka program akan **meminta ulang input** kedua dan ketiga.
 - **Input** keempat adalah **nilai ujian tengah semester** antara **1 sampai 100** dengan tipe data **double**.
 - **Input** kelima adalah **nilai ujian akhir semester** antara **1 sampai 100** dengan tipe data **double**.
- Program akan **menghitung nilai akhir mahasiswa** dengan **mengalikan persentase ujian dengan nilai ujiannya**.
- Program akan **menentukan grade mahasiswa** berdasarkan nilai akhir mahasiswa dengan kondisi:
 - Apabila **nilai akhir diatas 85**, maka mahasiswa mendapat **grade A**.
 - Apabila **nilai akhir diatas 75 dan dibawah 85**, maka mahasiswa mendapat **grade B**.
 - Apabila **nilai akhir diatas 65 dan dibawah 75**, maka mahasiswa mendapat **grade C**.
 - Apabila **nilai akhir diatas 55 dan dibawah 65**, maka mahasiswa mendapat **grade D**.
 - Apabila **nilai akhir dibawah 55**, maka mahasiswa mendapat **grade E**.

- Program akan **menentukan komentar mahasiswa** berdasarkan grade dengan kondisi:
 - Apabila **grade A**, maka mahasiswa mendapatkan **komentar “Very good! Keep your score”**.
 - Apabila **grade B**, maka mahasiswa mendapatkan **komentar “Pretty good! Increase your score”**.
 - Apabila **grade C**, maka mahasiswa mendapatkan **komentar “Quite less! You have to learn more”**.
 - Apabila **grade D**, maka mahasiswa mendapatkan **komentar “It's okay! You have to be more active”**.
 - Apabila **grade E**, maka mahasiswa mendapatkan **komentar “You have to learn more and be more active”**.
- Program akan **menampilkan nilai akhir, grade, dan komentar mahasiswa**.



```

<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 19, 2021 10:52:46 PM – 10:53:13 PM)
+-----+
| Score Calculator Program |
+-----+

Student Name [5-20 Karakter]: Tiff
Student Name [5-20 Karakter]: Daisy

=== Exam Percentage ===
Mid Exam [1..99]: 30
Final Exam [1..70]: 30
Mid Exam [1..99]: 30
Final Exam [1..70]: 70

=== Exam Score ===
Mid Exam [1-100]: 110
Mid Exam [1-100]: 85
Final Exam [1-100]: 95
|

=== Result ===
Mid Exam   : 85.00 (30%)
Final Exam : 95.00 (70%)
Total      : 92.00 (Grade A)
Very good! Keep your score Daisy!
  
```

Hasil 20. Program untuk Menghitung Nilai Mahasiswa

Berikut adalah **jawaban** untuk latihan soal diatas.

a. **Langkah 1** – Membuat Kelas *Scanner*.

```
7 Scanner scan = new Scanner(System.in);
```

Kode 34. Program untuk Menghitung Nilai Mahasiswa (Step 1)

b. **Langkah 2.1** – Mendapatkan input nama mahasiswa.

```
13 String name = "";
14 do {
15     System.out.print("Student Name [5-20 Karakter]: ");
16     name = scan.nextLine();
17 } while (name.length() < 5 || name.length() > 20);
```

Kode 35. Program untuk Menghitung Nilai Mahasiswa (Step 2.1)

c. **Langkah 2.2** – Mendapatkan input presentase nilai ujian mahasiwa.

```
19 System.out.println("\n=== Exam Percentage ===");
20 int midPercentage = 0, finalPercentage = 0;
21 do {
22     do {
23         System.out.printf("Mid Exam [1..99]: ");
24         midPercentage = scan.nextInt();
25         scan.nextLine();
26     } while (midPercentage < 1 || midPercentage > 99);
27
28     do {
29         System.out.printf("Final Exam [1..%d]: ", (100 - midPercentage));
30         finalPercentage = scan.nextInt();
31         scan.nextLine();
32     } while ((finalPercentage < 1 || finalPercentage > 100));
33 } while (midPercentage + finalPercentage != 100);
```

Kode 36. Program untuk Menghitung Nilai Mahasiswa (Step 2.2)

d. **Langkah 2.3** – Mendapatkan input nilai ujian mahasiswa.

```
35 System.out.println("\n=== Exam Score ===");
36 double midScore = 0;
37 do {
38     System.out.print("Mid Exam [1-100]: ");
39     midScore = scan.nextDouble();
40     scan.nextLine();
41 } while (midScore < 1 || midScore > 100);
42
43 double finalScore = 0;
44 do {
45     System.out.print("Final Exam [1-100]: ");
46     finalScore = scan.nextDouble();
47     scan.nextLine();
48 } while (finalScore < 1 || finalScore > 100);
```

Kode 37. Program untuk Menghitung Nilai Mahasiswa (Step 2.3)

e. **Langkah 3** – Menghitung nilai akhir mahasiswa.

```
50 double totalScore = ((midPercentage * midScore) + (finalPercentage * finalScore)) / 100;
```

Kode 38. Program untuk Menghitung Nilai Mahasiswa (Step 3)

f. **Langkah 4** – Menentukan *grade* mahasiswa.

```
52 char grade = 'E';
53 if (totalScore >= 85) {
54     grade = 'A';
55 } else if (totalScore >= 75 && totalScore < 85) {
56     grade = 'B';
57 } else if (totalScore >= 65 && totalScore < 75) {
58     grade = 'C';
59 } else if (totalScore >= 55 && totalScore < 65) {
60     grade = 'D';
61 }
```

Kode 39. Program untuk Menghitung Nilai Mahasiswa (Step 4)

g. **Langkah 5** – Menentukan komentar mahasiswa.

```
63 String comment = "";
64 switch(grade) {
65     case 'A': {
66         comment = "Very good! Keep your score";
67         break;
68     }
69     case 'B': {
70         comment = "Pretty good! Increase your score";
71         break;
72     }
73     case 'C': {
74         comment = "Quite less! You have to learn more";
75         break;
76     }
77     case 'D': {
78         comment = "It's okay! You have to be more active";
79         break;
80     }
81     case 'E': {
82         comment = "You have to learn more and be more active";
83         break;
84     }
85 }
```

Kode 40. Program untuk Menghitung Nilai Mahasiswa (Step 5)

h. **Langkah 6** – Menampilkan nilai akhir, grade, dan komentar mahasiswa

```
87 System.out.println("\n=== Result ===");
88 System.out.printf("Mid Exam    : %.2f (%d%%)\n", midScore, midPercentage);
89 System.out.printf("Final Exam   : %.2f (%d%%)\n", finalScore, finalPercentage);
90 System.out.printf("Total       : %.2f (Grade %c)\n", totalScore, grade);
91 System.out.printf("%s %s!\n", comment, name);
```

Kode 41. Program untuk Menghitung Nilai Mahasiswa (Step 6)

Bab 02. Array dan Pointer

A. Array Satu Dimensi

Variabel hanya dapat digunakan untuk menyimpan sebuah nilai. Sehingga, apabila kita ingin menyimpan lebih dari satu nilai, kita membutuhkan beberapa variabel. Oleh karena itu, Java menyediakan fitur bernama **array** sehingga kita dapat **menyimpan beberapa nilai dengan tipe data yang sama**. Secara teori, *array* merupakan sekumpulan alamat memori yang berurutan dimana tiap alamat memori menyimpan sebuah nilai. Semua tipe data, baik tipe data primitif ataupun tipe data non-primitif, dapat dibuat menjadi *array*. Untuk **mengakses nilainya**, kita dapat menggunakan **index**.

Biasanya, kita hanya menggunakan *array* satu dimensi atau dua dimensi. Namun apabila terdapat kasus khusus, tidak menutup kemungkinan untuk menggunakan *array* lebih dari dua dimensi. Secara umum **cara mendeklarasikan dan menginisialisasikan array satu dimensi** adalah sebagai berikut.

```
Format Deklarasi           : <TipeData> <NamaVariabel>[];  
                           : <TipeData>[] <NamaVariabel>;  
  
Format Inisialisasi        : <NamaVariabel> = new <TipeData>[<Ukuran>];  
  
Format Deklarasi dan Inisialisasi : <TipeData> <NamaVariabel>[] = new <TipeData>[<Ukuran>;  
    (Kosong) <TipeData>[] <NamaVariabel> = new <TipeData>[<Ukuran>;  
  
Format Deklarasi dan Inisialisasi : <TipeData> <NamaVariabel>[] = { <Nilai-1>, ... , <Nilai-n> };  
    (Dengan Nilai) <TipeData>[] <NamaVariabel> = { <Nilai-1>, ... , <Nilai-n> };  
  
Format Akses              : <NamaVariabel>[<Index>]
```

Format 11. Mendeklarasikan dan Menginisialisasikan Array Satu Dimensi

Catatan:

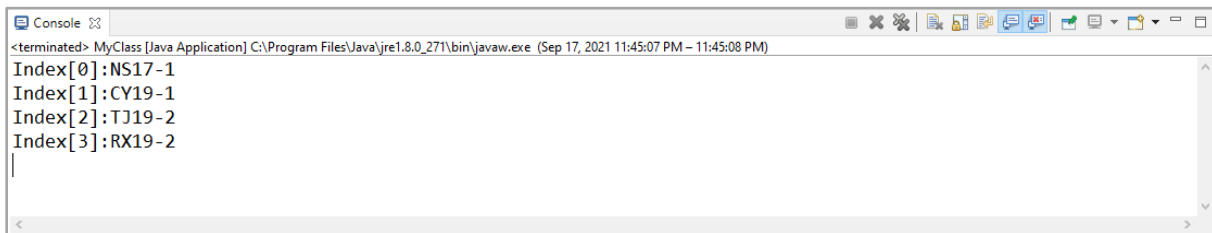
- Kita **tidak bisa menginisialisasikan nilai konstan** apabila variabel **dideklarasikan secara terpisah**.
- **Index** pada *array* **dimulai dari nol (0)**. Jadi, apabila ingin mengakses nilai ke-6, maka gunakan *index* ke-5.


```

3 public static void main(String[] args) {
4     // Mendeklarasikan array
5     int scores[];           // Cara ke-1
6     char[] alphabets;       // Cara ke-2
7
8     // Menginisialisasikan array
9     scores = new int[5];
10    alphabets = new char[26];
11
12    // Mendeklarasikan dan menginisialisasikan array
13    double multipliers[] = new double[25];           // Kosong
14    String[] assistants = {"NS17-1", "CY19-1", "TJ19-2", "RX19-2"}; // Dengan nilai
15
16    // Mengakses array
17    for (int i = 0 ; i < assistants.length ; i++) {
18        System.out.println("Index[" + i + "]: " + assistants[i]);
19    }
20 }

```

Kode 42. Mendeklarasikan dan Menginisialisasikan Array Satu Dimensi



```

<terminated> MyClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 17, 2021 11:45:07 PM - 11:45:08 PM)
Index[0]:NS17-1
Index[1]:CY19-1
Index[2]:TJ19-2
Index[3]:RX19-2

```

Hasil 21. Mendeklarasikan dan Menginisialisasikan Array Satu Dimensi

Catatan:

Salah satu cara **mendapatkan panjang sebuah array** adalah dengan menggunakan **method length**. Untuk cara menggunakannya dapat dilihat pada Kode 42.

B. Array Banyak Dimensi

Array dapat juga dibuat dalam bentuk banyak dimensi. Namun, pada bagian ini hanya akan dicontohkan mengenai **array dua dimensi**. Biasanya *array* jenis ini digunakan untuk **menghitung matriks, menyimpan map**, dan lain-lain. Secara umum, **cara mendeklarasikan dan menginisialisasikan *array* dua dimensi** adalah sebagai berikut.

```
Format Deklarasi           : <TipeData> <NamaVariabel>[][];  
                           : <TipeData>[][] <NamaVariabel>;  
  
Format Inisialisasi       : <NamaVariabel> = new <TipeData>[<Ukuran-1>][<Ukuran-2>];  
  
Format Deklarasi dan Inisialisasi : <TipeData> <NamaVariabel>[][] = new <TipeData>[<Ukuran-1>][<Ukuran-2>];  
                                (Kosong) <TipeData>[][] <NamaVariabel> = new <TipeData>[<Ukuran-1>][<Ukuran-2>];  
  
Format Deklarasi dan Inisialisasi : <TipeData> <NamaVariabel>[][] = {  
                                (Dengan Nilai)                                { <Nilai-1>, ... , <Nilai-n> },  
                                                                { <Nilai-1>, ... , <Nilai-n> },  
                                                                };  
                                : <TipeData>[][] <NamaVariabel> = {  
                                                                { <Nilai-1>, ... , <Nilai-n> },  
                                                                { <Nilai-1>, ... , <Nilai-n> },  
                                                                };  
  
Format Akses              : <NamaVariabel>[<Index-1>][<Index-2>]
```

Format 12. Mendeklarasikan dan Menginisialisasikan Array Dua Dimensi

```
3 public static void main(String[] args) {  
4     // Mendeklarasikan array  
5     char maps[][];           // Cara ke-1  
6     double[][] matrixs;     // Cara ke-2  
7  
8     // Menginisialisasikan array  
9     maps = new char[5][100];  
10    matrixs = new double[10][10];  
11  
12    // Mendeklarasikan dan menginisialisasikan array  
13    String albums[][] = new String[25][10];           // Kosong  
14    double[][] grades = {                             // Dengan nilai  
15        { 98.7, 81.6, 90.3 },  
16        { 82.3, 98.1, 78.4 },  
17        { 78.3, 86.1, 89.4 },  
18    };  
19  
20    // Mengakses array  
21    for (int i = 0; i < grades.length; i++) {  
22        for (int j = 0; j < grades[i].length; j++) {  
23            System.out.println("Index[" + i + "][" + j + "]: " + grades[i][j]);  
24        }  
25    }  
26 }
```

Kode 43. Mendeklarasikan dan Menginisialisasikan Array Dua Dimensi



The screenshot shows a Java console window titled "Console" with a tab for "MyClass [Java Application]". The console output displays the values of a 2D array at each index, formatted as "Index[i][j]:value". The values are as follows:

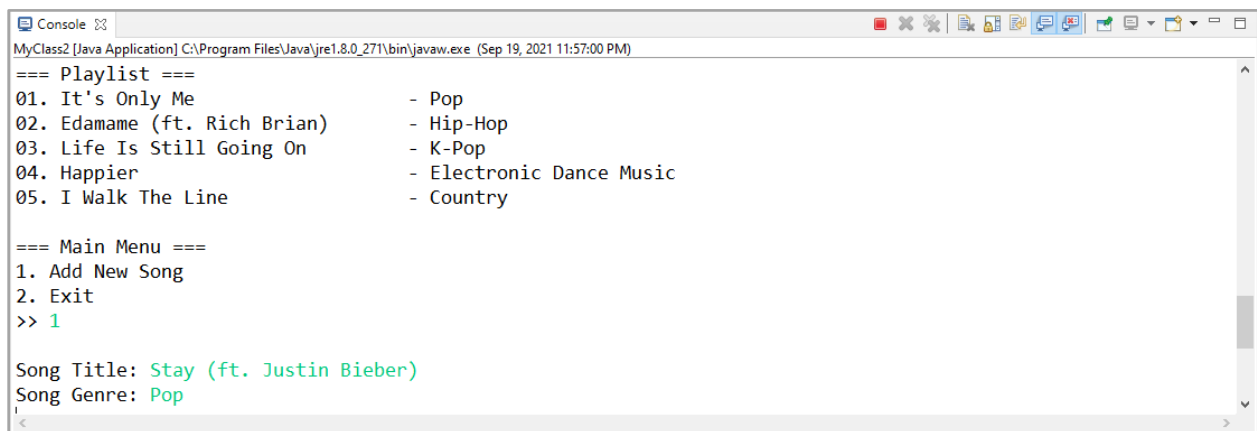
Index	Value
Index[0][0]	98.7
Index[0][1]	81.6
Index[0][2]	90.3
Index[1][0]	82.3
Index[1][1]	98.1
Index[1][2]	78.4
Index[2][0]	78.3
Index[2][1]	86.1
Index[2][2]	89.4

Hasil 22. Mendeklarasikan dan Menginisialisasikan *Array* Dua Dimensi

C. Latihan Soal

Buatlah sebuah program untuk **kumpulan lagu** dengan kriteria sebagai berikut.

- Buatlah **dua buah array** masing-masing untuk menampung **judul lagu** dan **genre lagu** dengan tipe data **String** sebanyak **50 data**.
- Program **menampilkan semua lagu** yang ada di dalam *array*.
- Buatlah sebuah **menu** yang terdiri atas **dua pilihan**, yaitu **menambahkan lagu (Add New Song)** dan **keluar (Exit)**.
 - Apabila pengguna memilih **menu pertama**, maka program akan **meminta input judul dan genre lagu** dengan tipe data **String**. Setelah itu, program akan **memasukan data lagu** yang sudah di input ke dalam *array*.
 - Apabila pengguna memilih **menu kedua**, maka program akan **berhenti**.



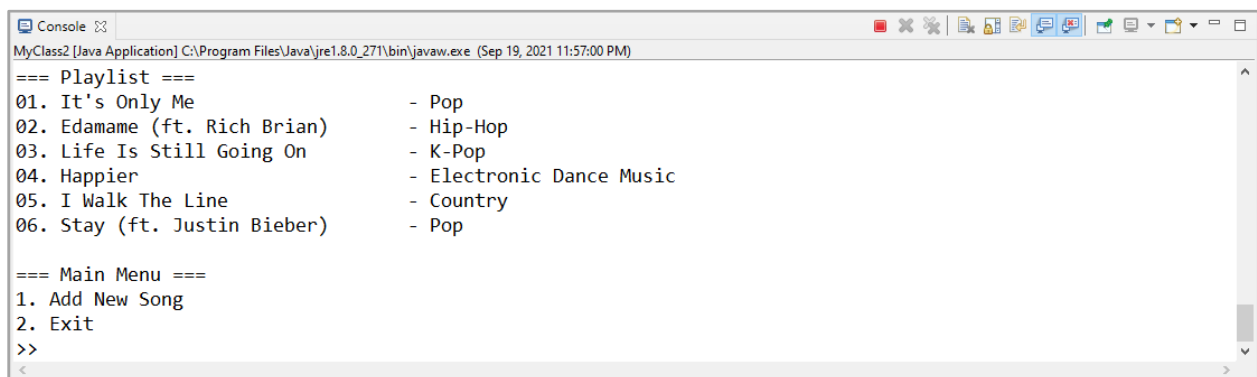
```
Console
MyClass2 [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 19, 2021 11:57:00 PM)

=== Playlist ===
01. It's Only Me           - Pop
02. Edamame (ft. Rich Brian) - Hip-Hop
03. Life Is Still Going On  - K-Pop
04. Happier                - Electronic Dance Music
05. I Walk The Line         - Country

=== Main Menu ===
1. Add New Song
2. Exit
>> 1

Song Title: Stay (ft. Justin Bieber)
Song Genre: Pop
```

Hasil 23. Program Kumpulan Lagu (Sebelum Menambahkan Lagu Baru)



```
Console
MyClass2 [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Sep 19, 2021 11:57:00 PM)

=== Playlist ===
01. It's Only Me           - Pop
02. Edamame (ft. Rich Brian) - Hip-Hop
03. Life Is Still Going On  - K-Pop
04. Happier                - Electronic Dance Music
05. I Walk The Line         - Country
06. Stay (ft. Justin Bieber) - Pop

=== Main Menu ===
1. Add New Song
2. Exit
>>
```

Hasil 24. Program Kumpulan Lagu (Setelah Menambahkan Lagu Baru)

Berikut adalah **jawaban** untuk latihan soal diatas.

a. **Langkah 1** – Membuat *array*.

```
8      String songTitles[] = new String[50];
9      String songGenres[] = new String[50];
10     int songIndex = 0;
```

Kode 44. Program Kumpulan Lagu (Step 1)

b. **Langkah 2** – Menampilkan semua lagu.

```
14     System.out.println("=== Playlist ===");
15     for (int i = 0 ; i < songIndex ; i++) {
16         System.out.printf("%02d. %-30s - %s\n", (i + 1), songTitles[i], songGenres[i]);
17     }
18     System.out.println("");
```

Kode 45. Program Kumpulan Lagu (Step 2)

c. **Langkah 3.1** – Menampilkan dan mendapatkan *input* menu.

```
12     int menu = 0;
13     do {
14         System.out.println("=== Main Menu ===");
15         System.out.println("1. Add New Song");
16         System.out.println("2. Exit");
17         System.out.printf(">> ");
18         menu = scan.nextInt();
19         scan.nextLine();
20     } while (menu != 2);
```

Kode 46. Program Kumpulan Lagu (Step 3.1)

d. **Langkah 3.2** – Menu pertama.

```
28     if (menu == 1) {
29         System.out.printf("Song Title: ");
30         String songTitle = scan.nextLine();
31
32         System.out.printf("Song Genre: ");
33         String songGenre = scan.nextLine();
34
35         songTitles[songIndex] = songTitle;
36         songGenres[songIndex] = songGenre;
37         songIndex++;
38     }
```

Kode 47. Program Kumpulan Lagu (Step 3.2)

Catatan:

- Untuk **langkah 2** diletakan **di dalam repetisi *do ... while*** sebelum menu yang ada pada langkah 3.
- Untuk **langkah 3.2** diletakan **di dalam repetisi *do ... while*** setelah menu yang ada pada langkah 3.
- Untuk **menu kedua**, yaitu Exit (keluar) **tidak dibuat** karena sudah menjadi **kondisi untuk repetisi *do ... while***. Program akan mengulang kode selama pengguna tidak memasukan *input* dua.