

# Pemrograman Berorientasi Objek 3 : Pernyataan, ekspresi, variabel, tipe data, dll

Royana Afwani

# Materi : Memahami sintaks dan grammar bahasa java

Pernyataan dan Ekspresi

Variabel dan Tipe Data

Operator

Pernyataan penentu keputusan

Pernyataan pengulangan proses (Looping)

Tambahan :

- I/O Stream sederhana
- Konversi String dan Bilangan (Number)
- Pernyataan pemindah proses



# 1. Pernyataan dan Ekspresi

# Pernyataan

- Perintah yang menyebabkan sesuatu terjadi dan merepresentasikan suatu aksi tunggal dalam program Java

Contoh: `int tahunProduksi;`

- Setiap pernyataan ditutup oleh karakter semicolon (;)
- Pernyataan dikelompokkan dengan tanda pembuka ({} dan penutup (}). Kelompok ini disebut blok atau blok pernyataan



# Ekspresi

- Pernyataan dapat menghasilkan suatu nilai. Nilai yang dihasilkan oleh pernyataan ini yang disebut dengan nilai balik (return value)
- Nilai balik bisa berupa bilangan, boolean, atau objek
- Method tanpa nilai balik biasanya menggunakan keyword void
- Contoh:     hasilBagi = a / b;



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

# Kata Kunci



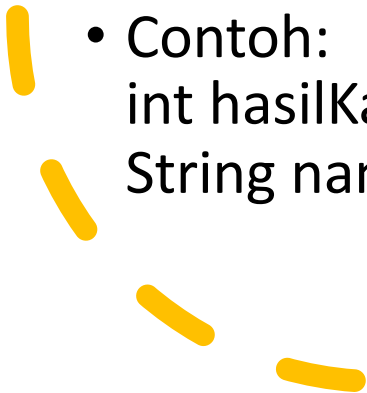


## 2. Variabel dan Tipe Data



# Variabel

- Lokasi di dalam memori komputer yang digunakan untuk menyimpan suatu informasi (nilai)
- Nilai variabel dapat diubah di pernyataan manapun di dalam program
- Mendeklarasikan variabel:  
tipe namaVariabel1 [, namaVariabel2]
- Contoh:  
int hasilKali;  
String namaSiswa, namaGuru, namaAdmin;





# Deklarasi Variabel

**Syntax**    *typeName variableName = value;*  
              or  
              *typeName variableName;*

## Example

The type specifies what can be done with values stored in this variable.

String greeting = "Hello, Dave!";

See the rules for and table of examples of valid names.

A variable declaration ends with a semicolon.

Supplying an initial value is optional, but it is usually a good idea.

Use a descriptive variable name.



# Memberi Nilai Variabel

- Menggunakan tanda sama dengan (=)
- Kebiasaan yang baik untuk memberi nilai awal (*initial value*) pada local variabel (mencegah *bug* pada program)
- Secara otomatis, Java akan memberi nilai awal pada instance variable
- Contoh:  
    int hasilTambah = 0;  
    boolean status = false;

# Memberi Nama Variabel

- Gunakan lowercase untuk variabel yang terdiri dari satu kata atau kata pertama
- Gunakan kapital untuk karakter pertama dari kata kedua, ketiga, dst
- Contoh:  
`int hasil;`  
`boolean statusMesinMobil;`  
`Button openFile;`

# Lingkup Variabel

Variabel dipanggil berdasarkan **lingkupnya**, dimulai dari blok yang paling kecil, kemudian blok di luar itu

1. **Local Variable**: digunakan di dalam method atau blok pernyataan yang lebih kecil dari itu
2. **Parameter**: variabel yg ada di dalam pernyataan (**argument**) method
3. **Instance Variable**: variabel yang memiliki nilai yang berbeda di setiap objek
4. **Class Variable**: variabel yang berlaku di suatu class dan seluruh instan dari class tersebut (objek). Ciri class variable adalah menggunakan keyword **static**
5. Variable static dipakai **bersama** oleh seluruh objek dari kelas yang memiliki variabel static tersebut.

<https://www.javatpoint.com/static-keyword-in-java>

```
public class Bilangan {  
    static int pencacah = 0;  
    int nilai;  
  
    public Bilangan(int nilai){  
        this.nilai = nilai;  
        pencacah++;  
    }  
  
    public void info(){  
        System.out.println("Nilai:" + nilai);  
        System.out.println("Pencacah:" + pencacah);  
        System.out.println("");  
    }  
}
```

```
public class BilanganBeraksi{  
    public static void main(String[] args){  
        Bilangan b1 = new Bilangan(50);  
        b1.info();  
  
        Bilangan b2 = new Bilangan(15);  
        b2.info();  
  
        Bilangan b3 = new Bilangan(30);  
        b3.info();  
    }  
}
```

## Bilangan.java

```
public class Bilangan {  
    static int pencacah = 0;  
    int nilai;  
  
    public Bilangan(int nilai){  
        this.nilai = nilai;  
        pencacah++;  
    }  
  
    public void info(){  
        System.out.println("Nilai:" + nilai);  
        System.out.println("Pencacah:" + pencacah);  
        System.out.println("");  
    }  
}
```

## BilanganBeraksi.java

```
public class BilanganBeraksi{  
    public static void main(String[] args){  
        Bilangan b1 = new Bilangan(50);  
        b1.info();  
  
        Bilangan b2 = new Bilangan(15);  
        b2.info();  
  
        Bilangan b3 = new Bilangan(30);  
        b3.info();  
    }  
}
```

# Hasil Eksekusi: static vs non-static

static

non-static

Nilai:50  
Pencacah:1

Nilai:15  
Pencacah:2

Nilai:30  
Pencacah:3

Nilai:50  
Pencacah:1

Nilai:15  
Pencacah:1

Nilai:30  
Pencacah:1

# Konvesi Pemrograman Java

Element	Convention	Example
Package	All letters in lowercase.	<code>com.company.customer</code>
Class	First letter of each word is capitalized.	<code>CustomerDriver</code>
Interface	First letter of each word is capitalized.	<code>Drawable</code>
Variable	First word is not capitalized but the subsequent words are capitalized	<code>grandTotal</code>
Method	First word is not capitalized but subsequent words are capitalized. Methods should be verbs.	<code>computePay</code>
Constant	Every letter is uppercase.	<code>LIMIT</code>

\* Konvesi lengkapnya ada di:

<http://www.oracle.com/technetwork/java/codeconv-138413.html>

# Modifier

- Modifier adalah **keyword yang diletakkan di depan** class, interface, variable (field) atau method
- Jenis Modifier:

## 1. Access Modifier:

<https://www.javatpoint.com/access-modifiers>

- Pengaturan pengaksesan dari variable dan method

## 2. Static Modifier:

- Membuat method dan variable menjadi milik class, bukan object
- Tidak perlu membuat object untuk penggunaan variable (field) dan method

## 3. Final Modifier:

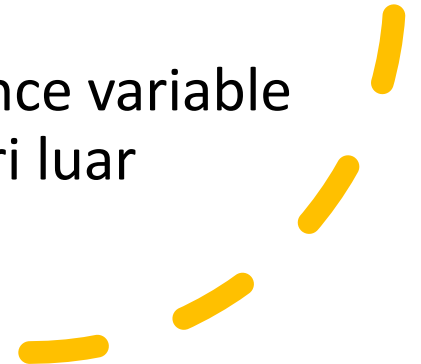
- Menyatakan bahwa sesuatu (class, method, variable) sudah final dan tidak dapat diubah

<https://www.javatpoint.com/final-keyword>



# Access Modifier

- Private: pengaksesan suatu instance variabel dan method hanya dapat dilakukan di dalam class ( tidak bisa dari luar class)
- Tanpa Tanda: pengaksesan suatu instance variabel dan method dapat dilakukan di dalam kelas dan kelas lain dalam satu paket
- Protected: pengaksesan suatu instance variabel dan method dapat dilakukan di dalam kelas, kelas lain dalam satu paket, dan sub class
- Public: pengaksesan suatu instance variable dan method dapat dilakukan dari luar (sembarang) kelas



# Access Modifier

Modifier	Dalam Class yang Sama	Dalam Package yang Sama	Dalam SubClass	Dalam Package Lain
private	✓			
tanpa tanda	✓	✓		
protected	✓	✓	✓	
public	✓	✓	✓	✓

**Syntax** `ClassName.methodName(parameters)`

**Example**

The class where the  
pow method is declared.

`Math.pow(10, 3)`

All parameters of a static method  
are explicit parameters.

## Static Modifier (Method)

- Sama seperti static variabel, ketika method ditambahkan static modifier, maka method tersebut dikontrol oleh class, dan bukan oleh object lagi
- Pemanggilan method dapat dilakukan tanpa membuat object
- Static method biasanya digunakan pada method yang hanya melakukan perhitungan matematika

# Latihan: Static Method pada Matematika

1. Buat class MatematikaBaru dan MatematikaBaruBeraksi, yang berisi sama persis dengan class Matematika dan MatematikaBeraksi
2. Tambahkan static modifier untuk semua method, dan panggil method dari class MatematikaBaruBeraksi dengan tanpa membuat object



# Tipe Data

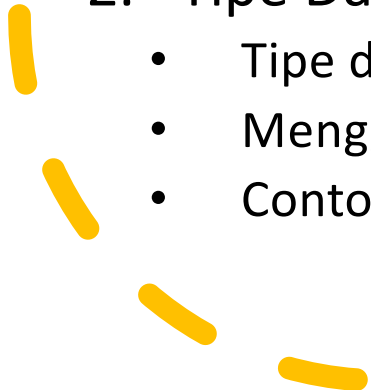
## 1. Tipe Data Primitif

- Tipe data yang merupakan kata kunci di Java (tertanam di compiler Java), sehingga pemrosesan jauh lebih cepat
- Menggunakan huruf kecil (lowercase)
- Contoh: int, double

## 2. Tipe Data Reference (Class)

- Tipe data berupa class yang ada di library Java (java.lang)
- Menggunakan huruf capital
- Contoh: String, Integer, Double

<https://www.javatpoint.com/reference-data-types-in-java>



# Tipe Data Primitif

1. byte
2. short
3. int
4. long
5. float
6. double
7. char
8. boolean

## Tipe Data Primitif:

- Secara umum jenis tipe data:
  1. bilangan bulat
  2. pecahan
  3. karakter
  4. boolean
- Tipe data hasil operasi matematika akan mengikuti tipe data dari operand

Tipe Data Primitif	Keterangan	Ukuran	Jangkauan
byte	Bilangan bulat	8 bit	-128 → 127
short	Bilangan bulat	16 bit	-32.768 → 32.767
int	Bilangan bulat	32 bit	-2.147.483.648 → 2.147.483.647
long	Bilangan bulat	64 bit	-9.223.372.036.854.775,808 → 9.223...807
float	Bilangan pecahan	32 bit (presisi 6-7 bit)	-3.4E38 → 3.4E38
double	Bilangan pecahan	64 bit (presisi 14-15 bit)	-1.7E308 → 1.7E308
char	Karakter (unicode)	16 bit	\u0000 → \uffff
boolean	Logika (true/false)		

# Karakter.java

```
public class Karakter{  
    public static void main(String[] args){  
  
        char karakter1=88, karakter2='X', karakter3='Y';  
  
        System.out.println("Karakter 1: " + karakter1);  
        System.out.println("Karakter 2: " + karakter2);  
        System.out.println("Karakter 3: " + karakter3);  
    }  
}
```



# Constant Variable (Konstanta)

- **Constant Variable** digunakan apabila kita ingin membuat nilai sebuah **variable tidak berubah** (tetap)
- Constant variable menggunakan keyword **final** di depan tipe data
- Biasanya digabungkan dengan keyword **static** bila dideklarasikan pada class
- Nama constant variable menggunakan **kapital**
- Contoh:

final float **PI** = 3.141592;

static final boolean **DEBUG** = false;

**Syntax**    Declared in a method:    `final typeName variableName = expression;`  
             Declared in a class:        `accessSpecifier static final typeName variableName = expression;`

**Example**

Declared in a method

`final double NICKEL_VALUE = 0.05;`

The `final` reserved word indicates that this value cannot be modified.

Use uppercase letters for constants.

`public static final double LITERS_PER_GALLON = 3.785;`

Declared in a class

## Constant Variable (Konstanta)

# Pengarah Tipe (Type-Casting)

- Pengarah Tipe (Type-Casting)

- Contoh:

```
double i = 10.56;
```

```
int paksa = (int) i;
```

Hasil → paksa = 10

- Casting tanpa menghilangkan nilai:

Tipe Sumber	Tipe Tujuan
byte	short, char, int, long, float, double
short	int, long, float, double
char	int, long, float, double
int	long, float, double
long	float, double
float	double

# Pembulatan (Math.round())

- **Math.round()** mengkonversi bilangan pecahan ke **bilangan bulat terdekat**

- Contoh:

```
int rounded = Math.round(balance);  
// if balance is 13.75, then rounded is set to 14
```

# LuasSegitiga.java

```
public class LuasSegitiga {  
    public static void main(String[] args) {  
        int alas= 3;  
        int tinggi = 7;  
  
        double luas = (double) (alas*tinggi)/2;  
  
        System.out.println("Luas Segitiga : " + luas);  
    }  
}
```

## Latihan: Hitung Luas Lingkaran

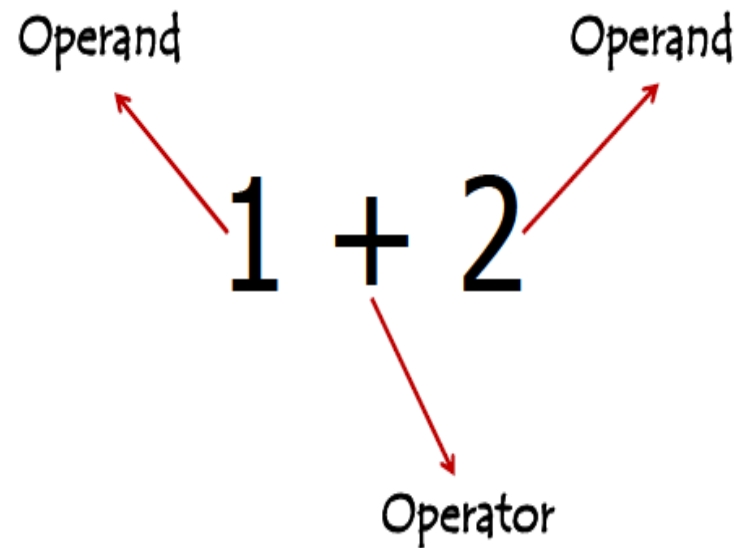
- Buat class Lingkaran yang mempunyai method menghitung luas lingkaran `hitungLuas(){ ... }`
  - Rumus luas lingkaran =  $\text{PI} * r * r$
  - PI adalah konstanta dengan nilai 3.141592
  - r adalah jari-jari lingkaran
- Buat class LingkaranBeraksi, yang menampilkan hasil perhitungan luas lingkaran dalam tiga bentuk bilangan: bilangan pecahan, bilangan bulat (type-casting) dan pembulatan (rounding). Beri nilai  $r = 11.78$



# 3. Operator

# Operator

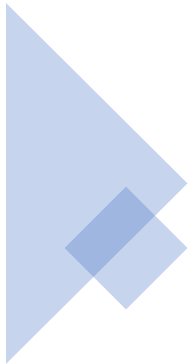
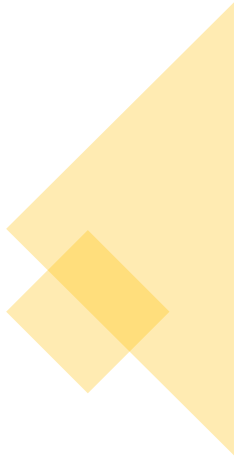
- Operator adalah simbol dan karakter khusus (matematika) yang digunakan dalam suatu ekspresi
- Contoh:
  - `int x = 3;`
  - `int y = x;`
  - `int z = x * y;`
  - `boolean status = true;`






# Jenis Operator Berdasar Operand

1. Operator Unary: operator yang melibatkan satu operand
2. Operator Binary: operator yang melibatkan dua operand
3. Operator Ternary: operator yang melibatkan tiga operand



A large orange shape on the left side of the slide, consisting of a rectangle with a quarter-circle cutout on its right side.

# Jenis Operator

1. Operator Aritmatika
  2. Operator Penugasan
  3. Operator Penggabungan
  4. Operator Increment dan Decrement
  5. Operator Bit
  6. Operator Pembandingan
  7. Operator Logika
- 
- Three short, curved yellow lines in the bottom right corner of the slide, arranged in a slightly upward-curving sequence.

## Operator Aritmatika

- Hasil operasi matematika akan mengikuti tipe data operand
- Operand bertipe int akan menghasilkan int

Operator	Meaning	Example
+	Addition	$3 + 4$
-	Subtraction	$5 - 7$
*	Multiplication	$5 * 5$
/	Division	$14 / 7$
%	Modulus	$20 \% 7$

Mathematical Expression	Java Expression	Comments
$\frac{x + y}{2}$	<code>(x + y) / 2</code>	The parentheses are required; <code>x + y / 2</code> computes $x + \frac{y}{2}$ .
$\frac{xy}{2}$	<code>x * y / 2</code>	Parentheses are not required; operators with the same precedence are evaluated left to right.
$\left(1 + \frac{r}{100}\right)^n$	<code>Math.pow(1 + r / 100, n)</code>	Complex formulas are “flattened” in Java.
$\sqrt{a^2 + b^2}$	<code>Math.sqrt(a * a + b * b)</code>	<code>a * a</code> is simpler than <code>Math.pow(a, 2)</code> .

Ekspresi Aritmatika

# Class Math dan Methodnya

Function	Returns
<code>Math.sqrt(x)</code>	square root
<code>Math.pow(x, y)</code>	power $x^y$
<code>Math.exp(x)</code>	$e^x$
<code>Math.log(x)</code>	natural log
<code>Math.sin(x)</code> , <code>Math.cos(x)</code> , <code>Math.tan(x)</code>	sine, cosine, tangent (x in radians)
<code>Math.round(x)</code>	closest integer to x
<code>Math.min(x, y)</code> , <code>Math.max(x, y)</code>	minimum, maximum

# Operator Penugasan

- Operator penugasan berguna untuk memberi nilai ke suatu variabel
- Operator penugasan menggunakan tanda sama dengan ( = )
- Operator penugasan digabungkan dengan operator aritmatika membentuk operator penugasan gabungan (compound assignment)



Operator  
Penugasan  
Gabungan

Expression	Meaning
$x += y$	$x = x + y$
$x -= y$	$x = x - y$
$x *= y$	$x = x * y$
$x /= y$	$x = x / y$

# Operator Penggabungan

- Operator + dapat digunakan untuk penggabungan String dan String maupun String dan Bilangan

- Contoh:

```
System.out.println("Saya adalah" + "Mahasiswa");
```

```
int mahasiswa = 30;
```

```
System.out.println("Jumlah Mahasiswa" + mahasiswa);
```



## Latihan: Tampilkan Nilai x, y,w dan z (latihan dirumah)

*Penambahan2.java*

```
int w, x, y, z;  
x = 5; w =5 ;  
y = 8 - x++;  
z = 8 - ++w;
```

*Pengurangan2.java*

```
int w,x, y, z;  
x = 5; w =5 ;  
y = 8 - x--;  
z = 8 - --w;
```

# Operator Bit




&	operasi AND untuk bit
	operasi OR untuk bit
^	operasi Ex OR untuk bit
~	operasi NOT untuk bit
<<	geser kiri (geser 1 bit = *2)
>>	geser kanan (geser 1 bit = /2)
>>>	geser kanan tak bertanda

## Operator Pembanding (Relasional)

Operator pembanding menggunakan ekspresi dengan nilai balik boolean (true or false)

Operator	Meaning	Example
==	Equal	x == 3
!=	Not equal	x != 3
<	Less than	x < 3
>	Greater than	x > 3
<=	Less than or equal to	x <= 3
>=	Greater than or equal to	x >= 3

**Table 1** Relational Operator Examples

Expression	Value	Comment
<code>3 &lt;= 4</code>	true	3 is less than 4; <= tests for “less than or equal”.
 <code>3 =&lt; 4</code>	<b>Error</b>	The “less than or equal” operator is <=, not =<, with the “less than” symbol first.
<code>3 &gt; 4</code>	false	> is the opposite of <=.
<code>4 &lt; 4</code>	false	The left-hand side must be strictly smaller than the right-hand side.
<code>4 &lt;= 4</code>	true	Both sides are equal; <= tests for “less than or equal”.
<code>3 == 5 - 2</code>	true	== tests for equality.
<code>3 != 5 - 1</code>	true	!= tests for inequality. It is true that 3 is not 5 - 1.
 <code>3 = 6 / 2</code>	<b>Error</b>	Use == to test for equality.
<code>1.0 / 3.0 == 0.333333333</code>	false	Although the values are very close to one another, they are not exactly equal. See Common Error 4.3.
 <code>"10" &gt; 5</code>	<b>Error</b>	You cannot compare a string to a number.
<code>"Tomato".substring(0, 3).equals("Tom")</code>	true	Always use the equals method to check whether two strings have the same contents.
<code>"Tomato".substring(0, 3) == ("Tom")</code>	false	Never use == to compare strings; it only checks whether the strings are stored in the same location. See Common Error 5.2 on page 180.
<code>"Tom".equalsIgnoreCase("TOM")</code>	true	Use the equalsIgnoreCase method if you don't want to distinguish between uppercase and lowercase letters.

# Contoh Operator Pembanding

# Membandingkan Bilangan Bulat

- The == denotes equality testing:

`a = 5; // Assign 5 to a`

`if (a == 5) ... // Test whether a equals 5`

- Relational operators have lower precedence than arithmetic operators:

`amount + fee <= balance`

Latihan:  
Tampilkan  
hasilBanding1  
dan 2 (latihan  
dirumah)

*Pembanding.java*

```
int age = 36;  
boolean hasilBanding1 = age < 25;  
boolean hasilBanding2 = age != 26;  
  
//Tampilkan hasilBanding1 dan hasilBanding2
```

# Membandingkan String

- Untuk membandingkan dua string, gunakan method **equals**:

`if (string1.equals(string2)) //Don't use == for strings!`

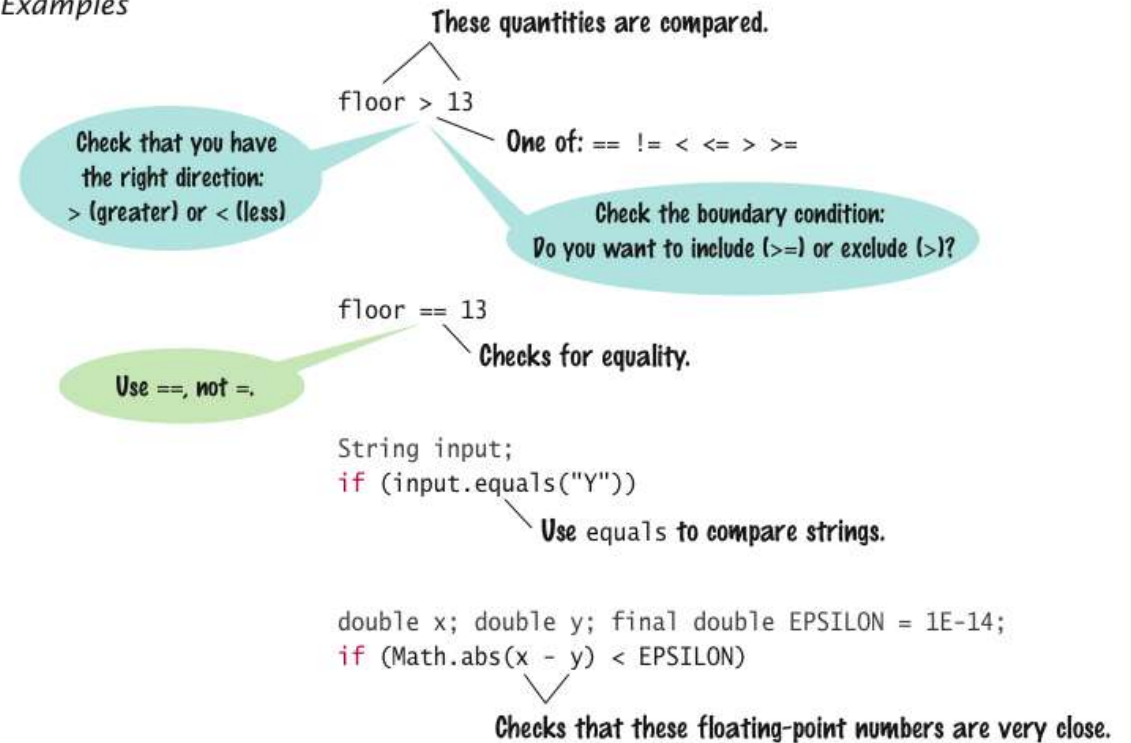
`if (string1 == string2) // Not useful`

- `==` membandingkan **identitas**
- `equals()` membandingkan kesamaan **content**
- Untuk membandingkan secara Case insensitive:

`if (string1.equalsIgnoreCase(string2))`

# Rangkuman Operator Pembanding

## Examples





A large orange shape on the left side of the slide, consisting of a rectangle with a quarter-circle cutout on its right side.

# Operator Logika

&&	operasi logika AND
	operasi logika OR
!	operasi logika NOT

Operator logika menggunakan ekspresi dengan nilai balik Boolean (true or false)

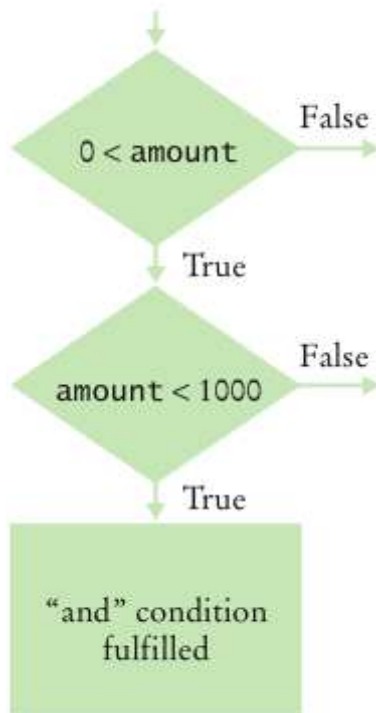
A yellow dashed line in the bottom right corner, consisting of several short, curved segments.

# Operator Logika

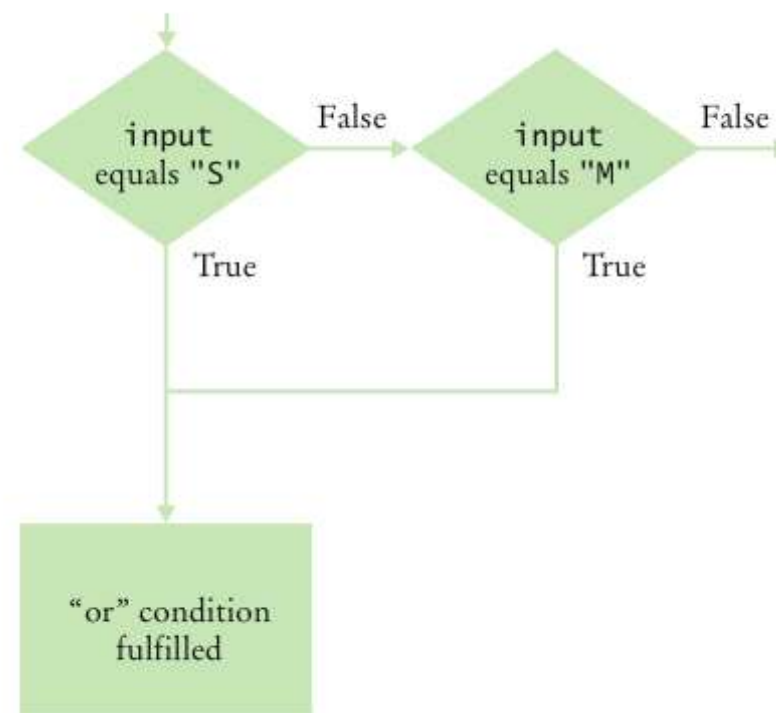
X	Y	X OR Y	X AND Y	!X
1	1	1	1	0
1	0	1	0	0
0	1	1	0	1
0	0	0	0	1

# Contoh Penggunaan && dan ||



`0 < amount && amount < 1000`



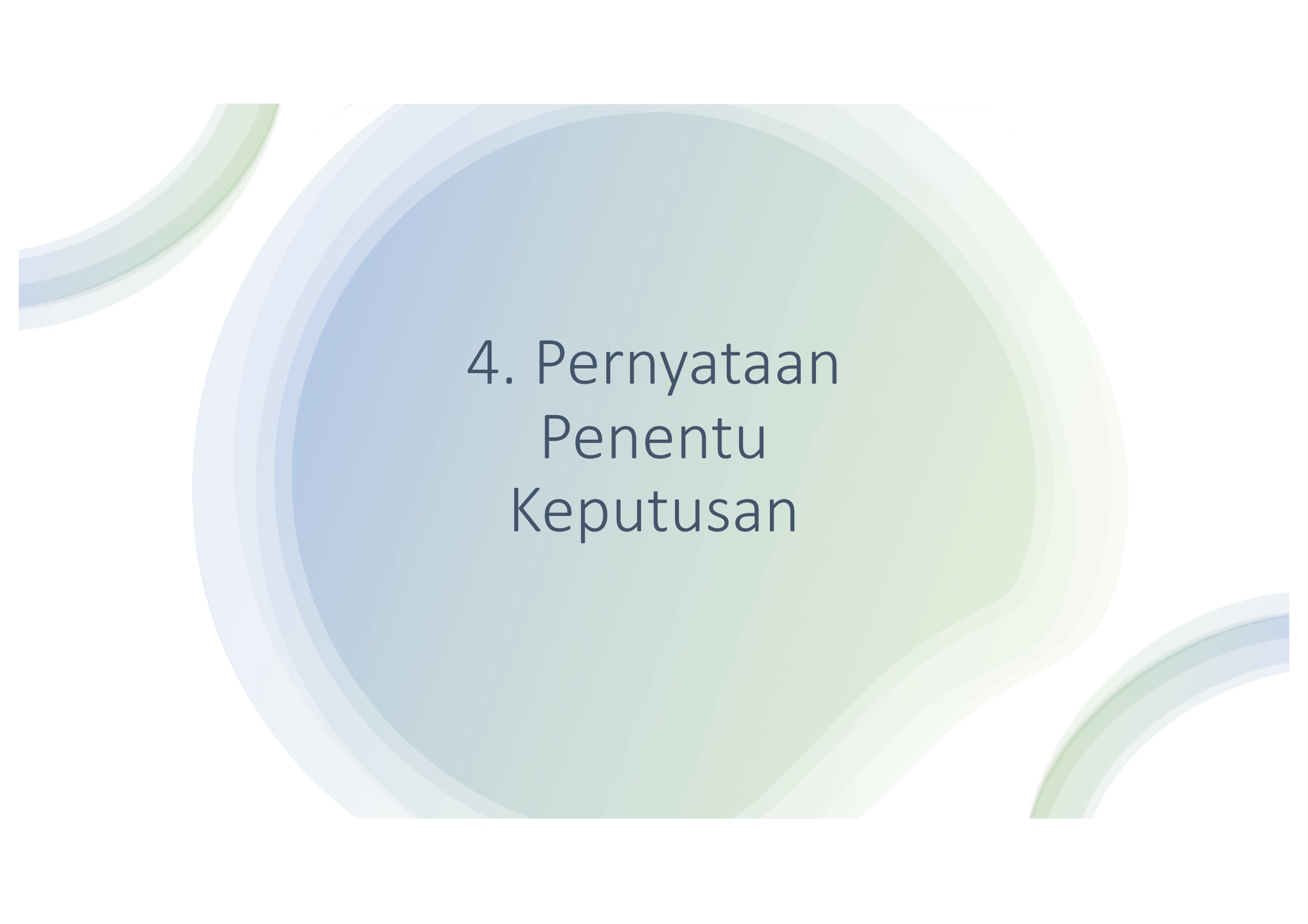
`input.equals("S") || input.equals("M")`



**Table 3** Boolean Operators

Expression	Value	Comment
<code>0 &lt; 200 &amp;&amp; 200 &lt; 100</code>	false	Only the first condition is true.
<code>0 &lt; 200    200 &lt; 100</code>	true	The first condition is true.
<code>0 &lt; 200    100 &lt; 200</code>	true	The <code>  </code> is not a test for “either-or”. If both conditions are true, the result is true.
 <code>0 &lt; 100 &lt; 200</code>	Syntax error	<b>Error:</b> The expression <code>0 &lt; 100</code> is true, which cannot be compared against 200.
 <code>0 &lt; x    x &lt; 100</code>	true	<b>Error:</b> This condition is always true. The programmer probably intended <code>0 &lt; x &amp;&amp; x &lt; 100</code> . (See Common Error 5.5).
<code>0 &lt; x &amp;&amp; x &lt; 100    x == -1</code>	<code>(0 &lt; x &amp;&amp; x &lt; 100)    x == -1</code>	The <code>&amp;&amp;</code> operator binds more strongly than the <code>  </code> operator.
<code>!(0 &lt; 200)</code>	false	<code>0 &lt; 200</code> is true, therefore its negation is false.
<code>frozen == true</code>	frozen	There is no need to compare a Boolean variable with true.
<code>frozen == false</code>	!frozen	It is clearer to use <code>!</code> than to compare with false.

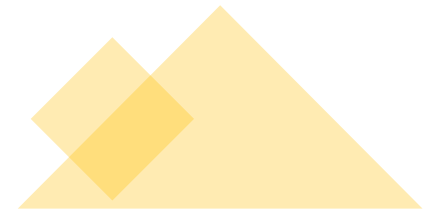
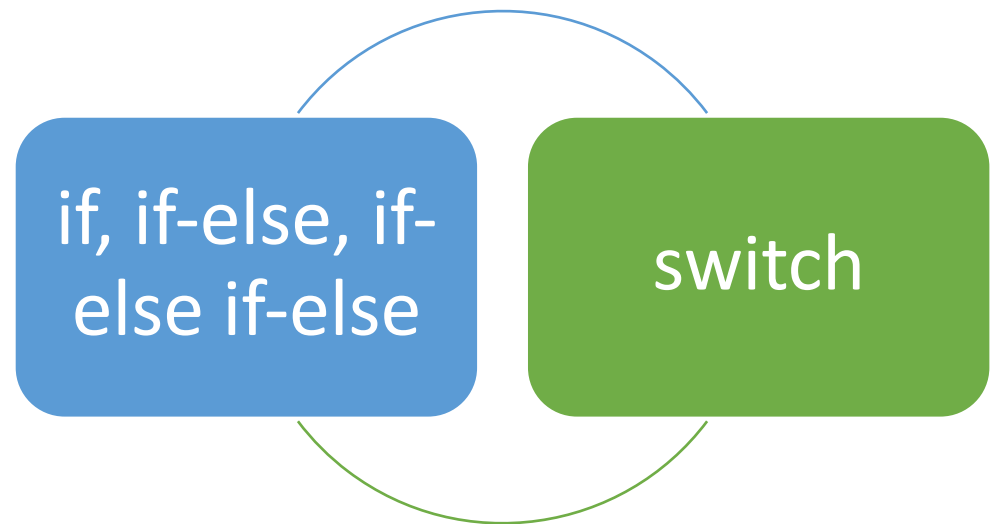
# Contoh Penggunaan Operator Logika



## 4. Pernyataan Penentu Keputusan



## Pernyataan Penentu Keputusan





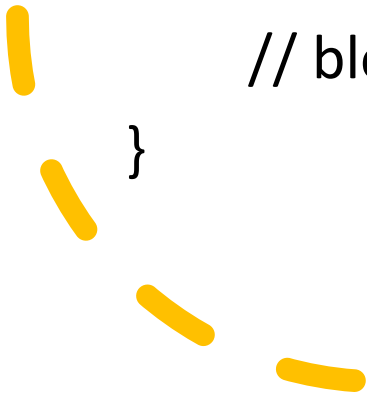
# if

- Pernyataan kondisi yang digunakan untuk pengambilan keputusan terhadap dua buah kemungkinan
- if bisa berdiri sendiri atau dengan menggunakan else
- Bentuk:

```
if(kondisi){
```

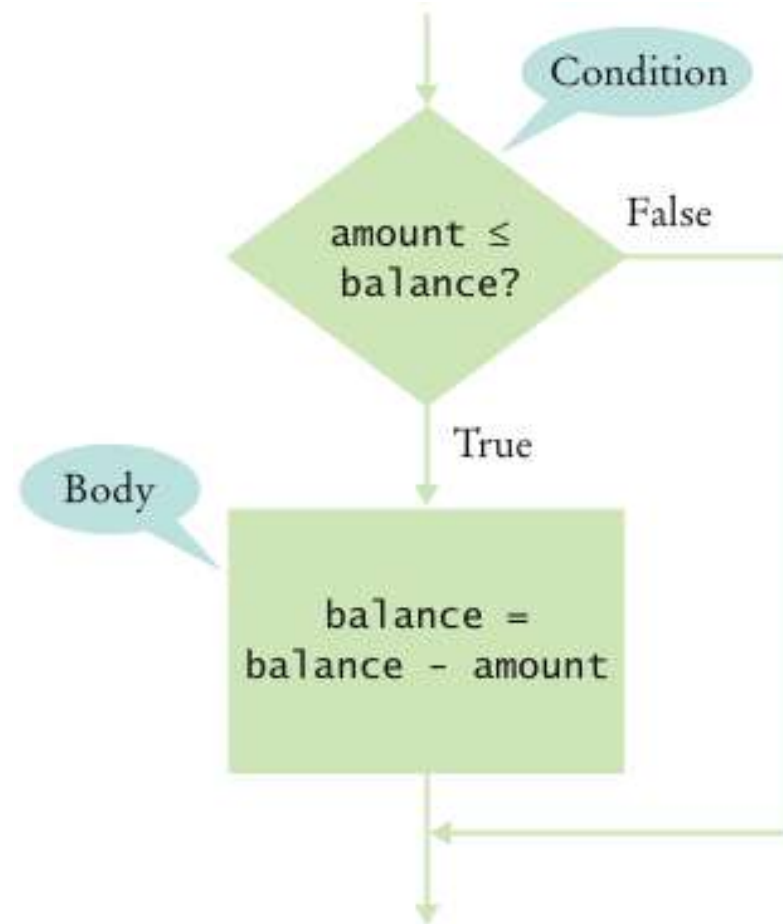
```
    // blok pernyataan yang dijalankan, bila kondisi benar
```

```
}
```



# Contoh if

```
if (amount <= balance) balance  
= balance - amount;
```






A large orange shape on the left side of the slide, consisting of a rectangle with a quarter-circle cutout on its right side.

PernyataanIF.java

```
public class PernyataanIF{  
    public static void main(String[] args){  
        int diskon =0, totalBelanja = 500000;  
  
        if(totalBelanja >= 100000){  
            diskon = totalBelanja/10;  
        }  
        System.out.println("Diskon = " + diskon);  
    }  
}
```

A yellow dashed line in the bottom right corner, consisting of several short, curved segments.

# if-else

- if-else mengatur pernyataan yang dijalankan sewaktu kondisi bernilai benar atau salah
- Bentuk:

```
if(kondisi){  
    // blok pernyataan yang dijalankan, bila kondisi benar  
} else{  
    // blok pernyataan yang dijalankan, bila kondisi salah  
}
```

# PernyataanIFELSE.java

```
public class PernyataanIFELSE{  
    public static void main(String[] args){  
        int diskon =0, totalBelanja = 500000;  
  
        if(totalBelanja >= 100000){  
            diskon = totalBelanja/10;  
        } else{  
            diskon = 0;  
        }  
        System.out.println("Diskon = " + diskon);  
    }  
}
```

# if-else if-else

- Mengatur pernyataan yang dijalankan sewaktu kondisi berupa pilihan
- Bentuk:

```
if(kondisiA){  
    // pernyataan yang dijalankan, bila kondisiA benar  
}else if(kondisiB){  
    // pernyataan yang dijalankan, bila kondisiB benar  
}else if(kondisiC){  
    // pernyataan yang dijalankan, bila kondisiC benar  
}else{  
    // pernyataan yang dijalankan untuk kondisi selain itu  
}
```

# PernyataanIFELSEIF.java

```
public class PernyataanIFELSEIF{  
    public static void main(String[] args) {  
        int skorUjian= 86; char nilai;  
        if (skorUjian >= 90) {  
            nilai = 'A';  
        } else if (skorUjian >= 80) {  
            nilai = 'B';  
        } else if (skorUjian >= 70) {  
            nilai = 'C';  
        } else {  
            nilai = 'D';  
        }  
        System.out.println("Nilai = " + nilai);  
    }  
}
```

}

# switch

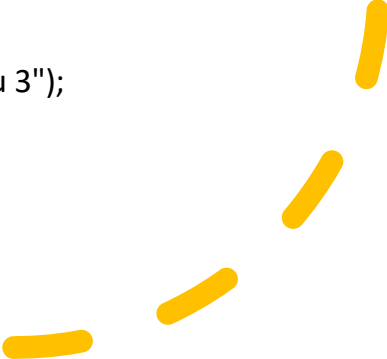
- Switch digunakan untuk melakukan **tindakan berbeda** terhadap sejumlah kemungkinan nilai
- Bentuk:

```
switch(ekspresi){  
    case nilaiSatu:  
        Pernyataan 1  
        break;  
    case nilaiDua:  
        Pernyataan2  
        break;  
    ...  
    default: PernyataanN;  
}
```

A large orange shape on the left side of the slide, consisting of a rectangle with a quarter-circle cutout on its right side.

PernyataanSWITCH1.java


```
public class PernyataanSWITCH1{  
    public static void main(String[] args){  
        int pilihan = 3;  
        switch(pilihan){  
            case 1:  
                System.out.println("Soto Ayam");  
                break;  
            case 2:  
                System.out.println("Gule Kambing");  
                break;  
            case 3:  
                System.out.println("Nasi Goreng");  
                break;  
            default:  
                System.out.println("Silakan Pilih 1, 2 atau 3");  
        }  
    }  
}
```

A yellow shape on the bottom right of the slide, consisting of several short, curved line segments arranged in an arc.

A large orange shape on the left side of the slide, consisting of a rectangle with a quarter-circle cutout on its right side.

PernyataanSWITCH2.java

```
public class PernyataanSWITCH2{  
    public static void main(String[] args){  
        int pilihan = 3;  
        switch(pilihan){  
            case 1:  
            case 2:  
            case 3:  
            case 4:  
            case 5:  
                System.out.println("Hari Kerja");  
                break;  
            case 6:  
            case 7:  
                System.out.println("Hari Libur");  
                break;  
            default:  
                System.out.println("Silakan Pilih Hari");  
        }  
    }  
}
```

A yellow shape on the bottom right of the slide, consisting of several curved, parallel lines that form a stylized, abstract shape.



## Latihan: Menentukan Jumlah Hari

- Buat program (dengan SWITCH) untuk menghitung **berapa jumlah hari pada suatu bulan dan tahun** yang ditunjuk
- Bulan dan tahun dimasukkan dengan **input dari keyboard (class Scanner)**
- Filter semua pilihan supaya **mengeluarkan error** untuk pilihan di luar yang kita tentukan (termasuk non digit)
- Tampilkan hasilnya dengan:

Masukkan tahun: **1900**

Masukan bulan: **2**

Jumlah hari pada tahun **1900** bulan **2** adalah **28** hari




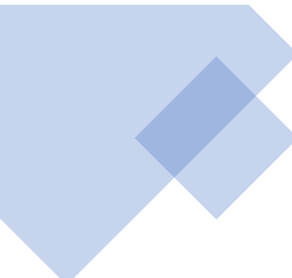
# Syarat Tahun Kabisat

1. Tahun yang habis dibagi 400


OR


2. Tahun yang habis dibagi 4 AND tidak habis dibagi 100

---



```
...va x Joko.java x Budi.java x Budi.java x CelciusToFahrenheitGUI.java x AppJ
Source Design
139 switch (bulan) {
140     case 1:
141     case 3:
142     case 5:
143     case 7:
144     case 8:
145     case 10:
146     case 12:
147         hari = 31;
148         break;
149     case 4:
150     case 6:
151     case 9:
152     case 11:
153         hari = 30;
154         break;
155     case 2:
156         if ( ((tahun % 4 == 0) && !(tahun % 100 == 0))
157             || (tahun % 400 == 0) )
158             hari = 29;
159         else
160             hari = 28;
161         break;
162     default:
163         System.out.println("Invalid month.");
164         break;
155:20 INS
```





## 5. Pernyataan Pengulangan Proses (*Loop*)

# Pernyataan Pengulangan Proses

**for**

**while**

**do-while**



for

- for sering disebut *for loop*, karena digunakan untuk proses looping atau pengulangan

- Bentuk:

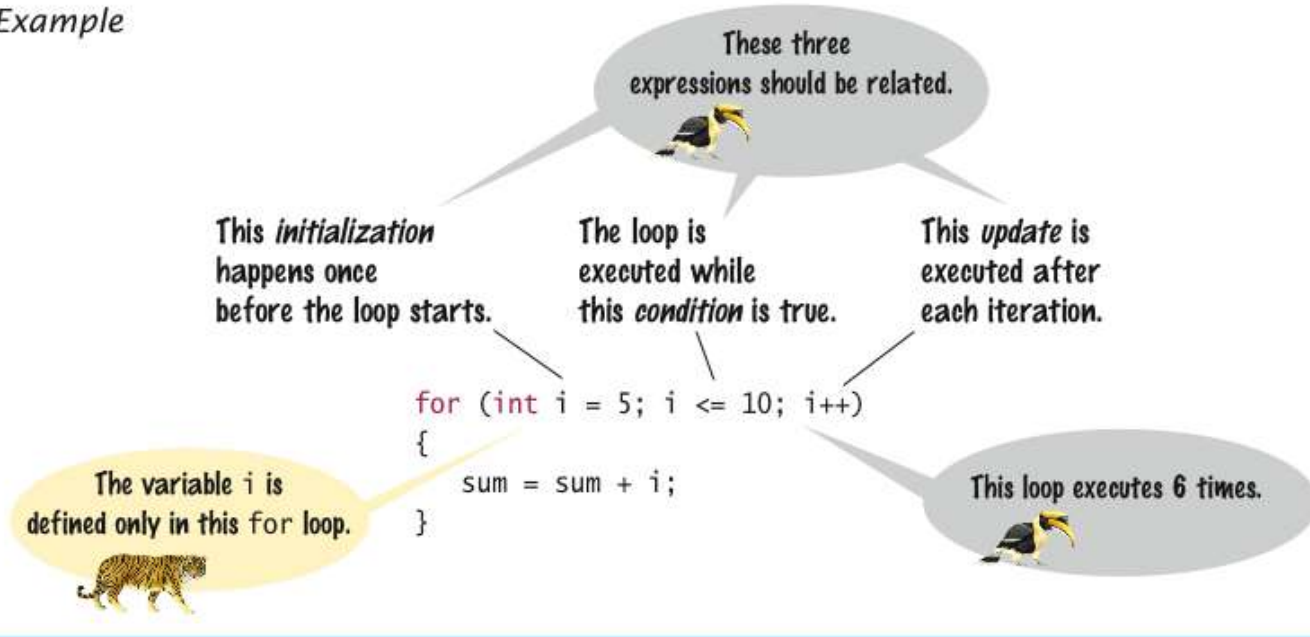
```
for (inisialisasi; kondisi;  
    penaikan_penurunan){  
    pernyataan  
}
```



# for

**Syntax**    `for (initialization; condition; update)  
                  statement`


**Example**



A large orange shape on the left side of the slide, consisting of a rectangle with a quarter-circle cutout from its top-right corner.

PernyataanFOR.java

```
public class PernyataanFOR {  
    public static void main(String[] args){  
        for(int i=1; i<11; i++){  
            System.out.println(i);  
        }  
    }  
}
```

A yellow shape in the bottom right corner, composed of several short, curved line segments arranged to form a larger, abstract curve.



# PernyataanFORArray.java

```
public class PernyataanFORArray{  
    public static void main(String[] args){  
        int[] numbers = {1,2,3,4,5,6,7,8,9,10};  
  
        for (int i : numbers) {  
            System.out.println(i);  
        }  
    }  
}
```

## Nested Loop - Program Pembuat Segitiga

```
public class Segitiga {  
    private int lebar;  
  
    public Segitiga(int lebar){  
        this.lebar = lebar;  
    }  
  
    public String gambarSegitiga(){  
        String r = "";  
        for (int i = 1; i <= lebar; i++){  
            for (int j = 1; j <= i; j++){  
                r = r + "[";  
                r = r + "\n";  
            }  
            return r;  
        }  
    }  
}
```

Segitiga.java

```
public class SegitigaBeraksi {  
    public static void main(String[] args) {  
        Segitiga kecil = new Segitiga(5);  
        System.out.println(kecil.gambarSegitiga());  
  
        Segitiga besar = new Segitiga(15);  
        System.out.println(besar.gambarSegitiga());  
    }  
}
```

SegitigaBeraksi.java

A large orange shape on the left side of the slide, consisting of a rectangle with a quarter-circle cutout on its right side.

# while

- while digunakan untuk melakukan proses pengulangan suatu blok pernyataan selama kondisinya bernilai true

- Bentuk:

```
while (kondisi) {  
    pernyataan  
}
```



# PernyataanWHILE.java

```
class PernyataanWHILE {  
    public static void main(String[] args){  
        int i = 1;  
        while (i < 11) {  
            System.out.println(i);  
            i++;  
        }  
    }  
}
```

# Latihan: Tampilkan Bilangan Genap

Tampilkan bilangan genap antara 1 sampai 20 dengan menggunakan:

1. Pernyataan FOR
2. Pernyataan WHILE

# do...while

- do ... while digunakan untuk melakukan proses pengulangan suatu blok pernyataan selama kondisinya bernilai true
- Pernyataan dalam blok paling tidak dieksekusi satu kali
- Bentuk:

```
do {  
    pernyataan  
} while (kondisi);
```

# PernyataanDOWHILE.java

```
class PernyataanDOWHILE {  
    public static void main(String[] args){  
        int i = 1;  
        do {  
            System.out.println(i);  
            i++;  
        } while (i <= 10);  
    }  
}
```



# Tambahan

## 6.1. I/O Stream Sederhana



# Stream Standard

1. **System.in**: menangani pembacaan dari keyboard (**standard input**)
2. **System.out**: mengirimkan keluaran ke layar (**standard output**)
3. **System.err**: mengirimkan kesalahan (**standard error**)

# Review : Method

**Syntax**    *object.methodName(parameters)*

**Example**

The method is  
invoked on this object.

This is the  
name of the method.

This parameter is  
passed to the method.

System.out.println("Hello")

Parameters are enclosed in parentheses.  
Multiple parameters are separated by commas.

# Membaca Input dari Keyboard

Menggunakan class **Scanner** (java.util.Scanner) dengan method sebagai berikut:

1. **nextInt()**: untuk menerima tipe data integer
2. **nextShort()**: untuk menerima tipe data short
3. **nextLong()**: untuk menerima tipe data long
4. **nextDouble()**: untuk menerima tipe data double
5. **nextFloat()**: untuk menerima tipe data float
6. **nextLine()**: untuk menerima tipe data string
7. **nextBoolean()**: untuk menerima tipe data boolean

# SalamKenal.java

```
import java.util.Scanner;

public class SalamKenal {
    public static void main( String[] args ){
        Scanner masukan = new Scanner(System.in);

        System.out.print("Masukkan Nama Anda: ");
        String nama = masukan.nextLine();

        System.out.println("Halo, Salam Kenal sdr " + nama + "!");
    }
}
```

# Perkalian.java

```
public class Perkalian{  
    public static void main(String[] args){  
        Scanner input = new Scanner(System.in);  
        System.out.print("Masukkan bilangan pertama: ");  
        int bilangan1 = input.nextInt();  
  
        System.out.print("Masukkan bilangan kedua: ");  
        int bilangan2 = input.nextInt();  
  
        System.out.print("Hasil perkalian: " +(bilangan1 * bilangan2));  
    }  
}
```

# Latihan

Ubah class LuasSegitiga dengan nilai alas dan tinggi dimasukkan oleh user lewat prompt (gunakan class Scanner)

```
public class LuasSegitiga {  
    public static void main(String[] args) {  
        double alas= 17; double tinggi = 11;  
  
        double luas = (alas*tinggi)/2;  
        System.out.println("Luas Segitiga : " + luas);  
    }  
}
```

# Tampilan Program

Program Penghitung Luas Segitiga

Masukkan Alas = 13

Masukkan Tinggi = 24

Jadi, Luas Segitiga adalah =

# Argument untuk Menerima Input

```
public class LuasSegitigaArgs {  
    public static void main(String[] args) {  
  
        double alas= Double.parseDouble(args[0]);  
        double tinggi = Double.parseDouble(args[1]);  
  
        double luas = (alas*tinggi)/2;  
  
        System.out.println("Luas Segitiga : " + luas);  
  
    }  
}
```



# Tampilan Program

%java LuasSegitigaArgs 2 18

Luas Segitiga: 18



```
C:\> Command Prompt

D:\>java LuasSegitigaArgs 2 18
Luas Segitiga : 18.0

D:\>
```

A screenshot of a Windows Command Prompt window. The title bar is blue and says "C:\> Command Prompt". The main area is black with white text. It shows the command "D:\>java LuasSegitigaArgs 2 18" being entered, followed by the output "Luas Segitiga : 18.0". The prompt "D:\>" is shown again on the next line.

# Argument untuk Menerima Input (Rev)

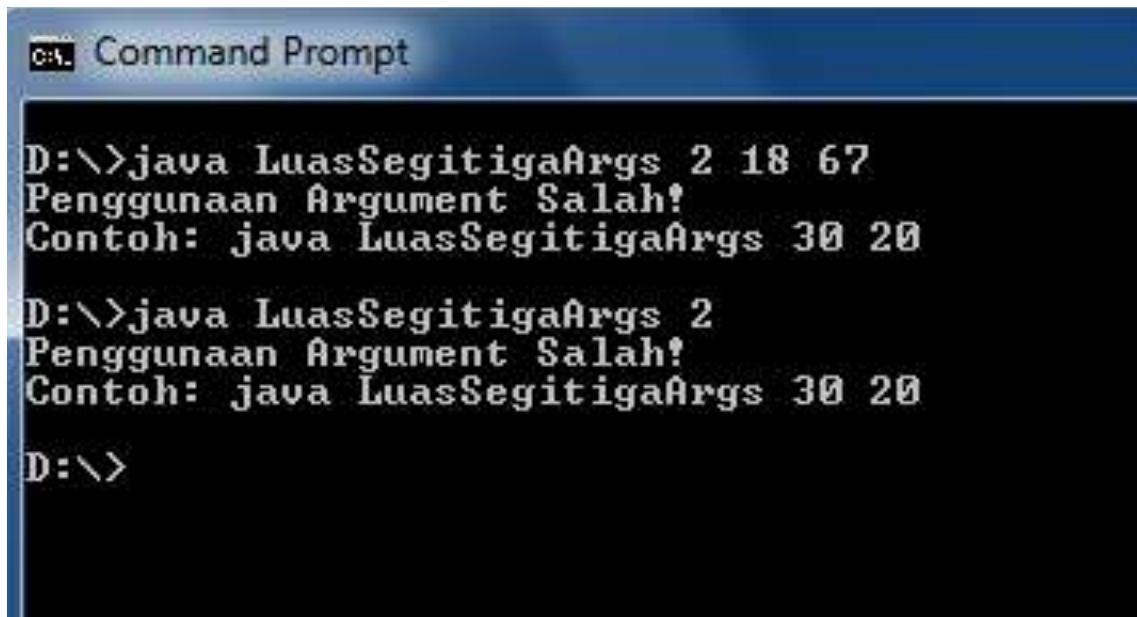
```
public class LuasSegitigaArgs {  
    public static void main(String[] args) {  
  
        if (args.length == 2){  
            double alas = Double.parseDouble(args[0]);  
            double tinggi = Double.parseDouble(args[1]);  
            double luas = (alas*tinggi)/2;  
            System.out.println("Luas Segitiga : " + luas);  
        }else{  
            System.out.println("Penggunaan Argument Salah!");  
            System.out.println("Contoh: java LuasSegitigaArgs 30 20");  
        }  
    }  
}
```

# Tampilan Program

%java LuasSegitigaArgs 2 18 67

Penggunaan Argument Salah!

Contoh: java LuasSegitigaArgs 30 20



```
Command Prompt

D:\>java LuasSegitigaArgs 2 18 67
Penggunaan Argument Salah!
Contoh: java LuasSegitigaArgs 30 20

D:\>java LuasSegitigaArgs 2
Penggunaan Argument Salah!
Contoh: java LuasSegitigaArgs 30 20

D:\>
```



# Tambahan

## 6.2. Konversi String dan Bilangan (Number)

# Konversi String ke Bilangan

```
String myString;
```

```
double myDbl = Double.parseDouble(myString);
```

```
Integer.parseInt(myString);
```

```
Float.parseFloat(myString);
```

# Argument untuk Menerima Input

```
public class LuasSegitigaArgs {  
    public static void main(String[] args) {  
  
        double alas= Double.parseDouble(args[0]);  
        double tinggi = Double.parseDouble(args[1]);  
  
        double luas = (alas*tinggi)/2;  
  
        System.out.println("Luas Segitiga : " + luas);  
  
    }  
}
```

# Konversi Bilangan ke String

```
double myDouble;
```

```
int myInteger;
```

```
float myFloat;
```

```
String myString = Double.toString(myDouble);
```

```
Integer.toString(myInteger);
```

```
Float.toString(myFloat);
```

# Substring

```
String greeting = "Hello, World!";
```

```
String sub = greeting.substring(0, 5); // sub is "Hello"
```

- Supply start and “past the end” position
- First position is at 0

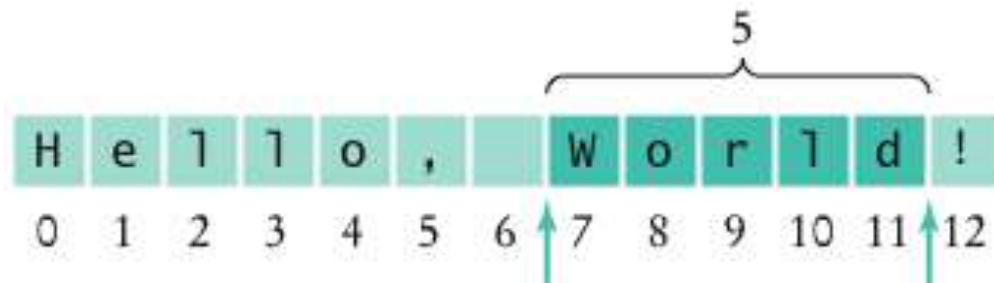
H	e	l	l	o	,		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11	12



# Substring

`String sub2 = greeting.substring(7, 12); // sub2 is "World"`

- Substring length is “past the end” - start



# Latihan

1. Buat class InputNama yang meminta kita memasukkan nama lengkap dan panjang karakter nama depan kita
2. Sebagai hasilnya class InputNama akan menampilkan nama depan kita
3. Gunakan method substring() untuk class tersebut

Masukkan nama anda: **Marina Dimana**

Berapa panjang karakter nama depan anda: **6**

Jadi nama depan adalah Marina



# Tambahan

## 6.3. Pernyataan Pemindah Proses (Jump)

A large orange shape on the left side of the slide, consisting of a rectangle with a quarter-circle cutout on its right side.

# Pernyataan Pemindah Proses

1. `return`
2. `break`
3. `continue`



# return

- Digunakan untuk **keluar dari method**
- **return** memiliki dua bentuk:
  1. **mengembalikan nilai** (sesuai dengan tipe data)
  2. **tidak mengembalikan nilai** (untuk void)
- Contoh:

```
int perkalian(int x, int y){  
    return x * y;  
}  
  
void perkalian(int x, int y){  
    hasilKali = x * y;  
    return;  
}
```

# break

- Pernyataan break digunakan untuk keluar dari suatu pengulangan (loop)
- Penggunaan break bisa berbentuk tanpa label atau berlabel

# PernyataanBREAK.java

```
public class PernyataanBREAK {  
    public static void main(String[] args){  
        for(int i=1; i<11; i++){  
            if(i==5) break;  
            System.out.println(i);  
        }  
    }  
}
```

# Pernyataan BREAK LABEL.java

```
public class PernyataanBREAK {  
    public static void main(String[] args){  
        selesai:  
        for(int i=1; i<11; i++){  
            if(i==5) break selesai;  
            System.out.println(i);  
        }  
    }  
}
```



# continue

- Digunakan untuk melanjutkan eksekusi ke suatu pengulangan (loop)
- Bisa berbentuk tanpa label atau berlabel
- Bentuk code persis sama dengan break, baik untuk yang tanpa label atau berlabel

# Latihan: Input Data pada Matematika

1. Modifikasi program **Matematika** dan **MatematikaBeraksi** yang sebelumnya sudah dibuat
2. Semua method di class Matematika menggunakan **return value int**
3. Input data adalah lewat **prompt (class Scanner)**
4. Buat **menu pilihan** untuk fitur pertambahan, pengurangan, pembagian atau perkalian
5. Filter semua pilihan supaya **mengeluarkan error** untuk pilihan di luar yang kita tentukan

# Latihan: Tampilan Matematika

## Menu Aplikasi Matematika:

1. Pertambahan    2. Pengurangan    3. Perkalian    4. Pembagian

Pilih Menu = 1

Masukkan Angka Pertama = 3

Masukkan Angka Kedua = 23

Hasil **Pertambahan** antara 3 dan 23 adalah 26

# Latihan: Menentukan Jumlah Hari (Rev)

- Buat program (dengan SWITCH) untuk menghitung **berapa jumlah hari pada suatu bulan dan tahun** yang ditunjuk
- Bulan dan tahun dimasukkan dengan **input dari keyboard (class Scanner)**
- Filter semua pilihan supaya **mengeluarkan error** untuk pilihan di luar yang kita tentukan (termasuk non digit)
- Buat jadi dua class: **JumlahHari2** dan **JumlahHari2Beraksi**
- Pada class JumlahHari2, buat method hitungHari, yang memiliki dua parameter  
**hitungHari(tahun, bulan)**
- Pada class JumlahHariBeraksi, letakkan main method dengan desain tampilan sama dengan program JumlahHari sebelumnya
- Tampilkan hasilnya dengan:  
Masukkan tahun: 1900  
Masukan bulan: 2  
Jumlah hari pada tahun 1900 bulan 2 adalah 28 hari