

[Document title]

[Document subtitle]

[Company name]

[Company address]

## MODUL 3 INSTRUKSI DASAR OOP

## Kegiatan Belajar 3

### Judul : INSTRUKSI DASAR OOP

### Indikator Keberhasilan

Setelah mempelajari materi pokok ini peserta dapat :

* Menerapkan instruksi dasar class
* Mengaplikasikan penggunaan operator

### Tujuan Pembelajaran

Setelah mempelajari materi pokok ini peserta dapat :

* Menerapkan instruksi dasar class
* Mengaplikasikan penggunaan operator

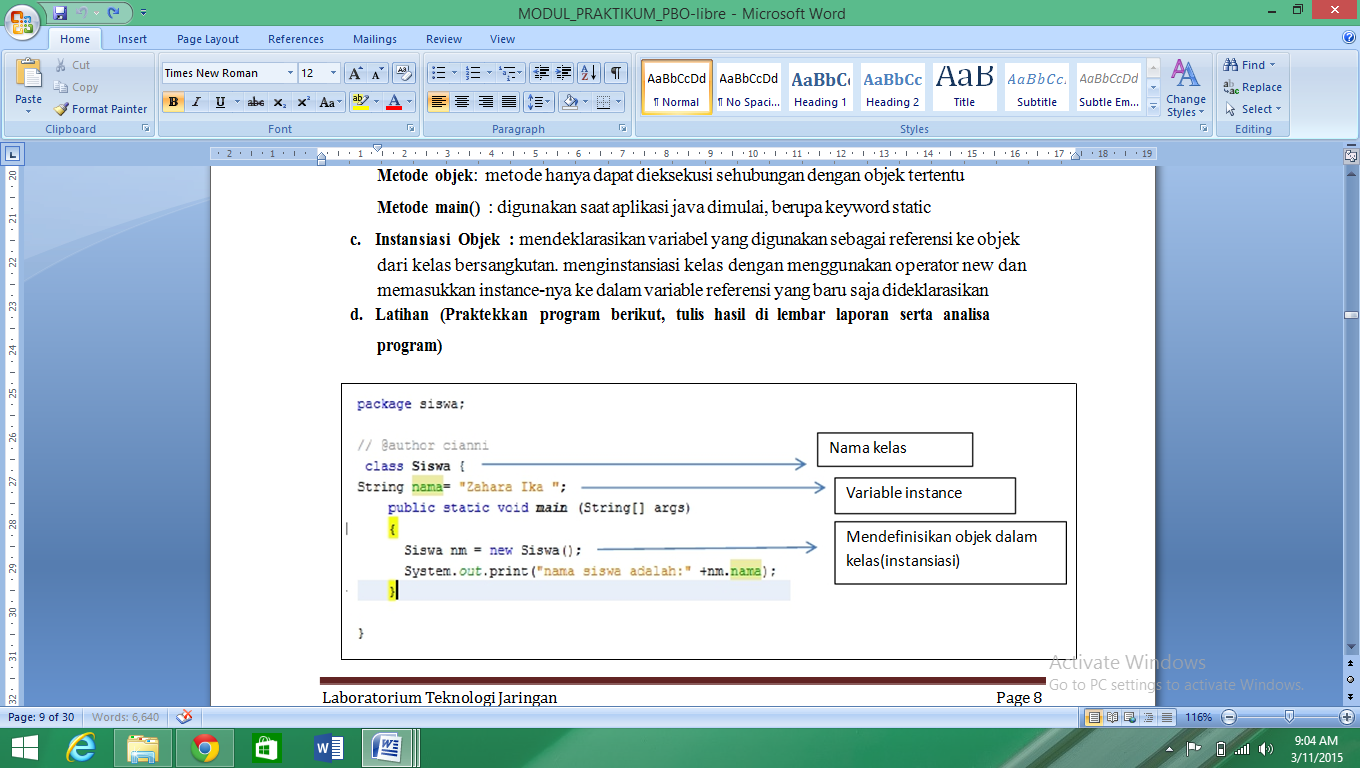
1. INSTRUKSI DASAR CLASS

**Class**

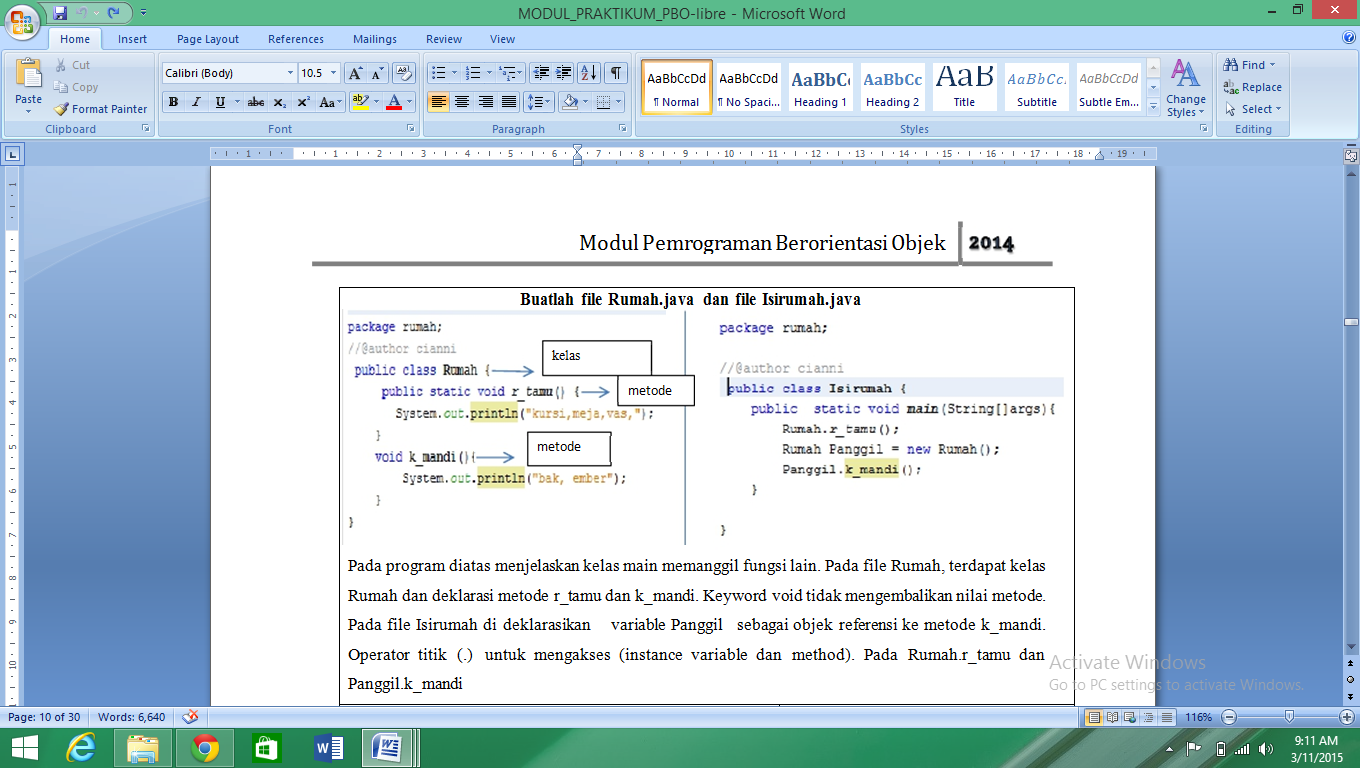
Class merupakan cetak biru (blue print) dari objek atau dengan kata lain sebuah Class menggambarkan ciri-ciri objek secara umum.

**Contoh penggunaan class pada Java**.

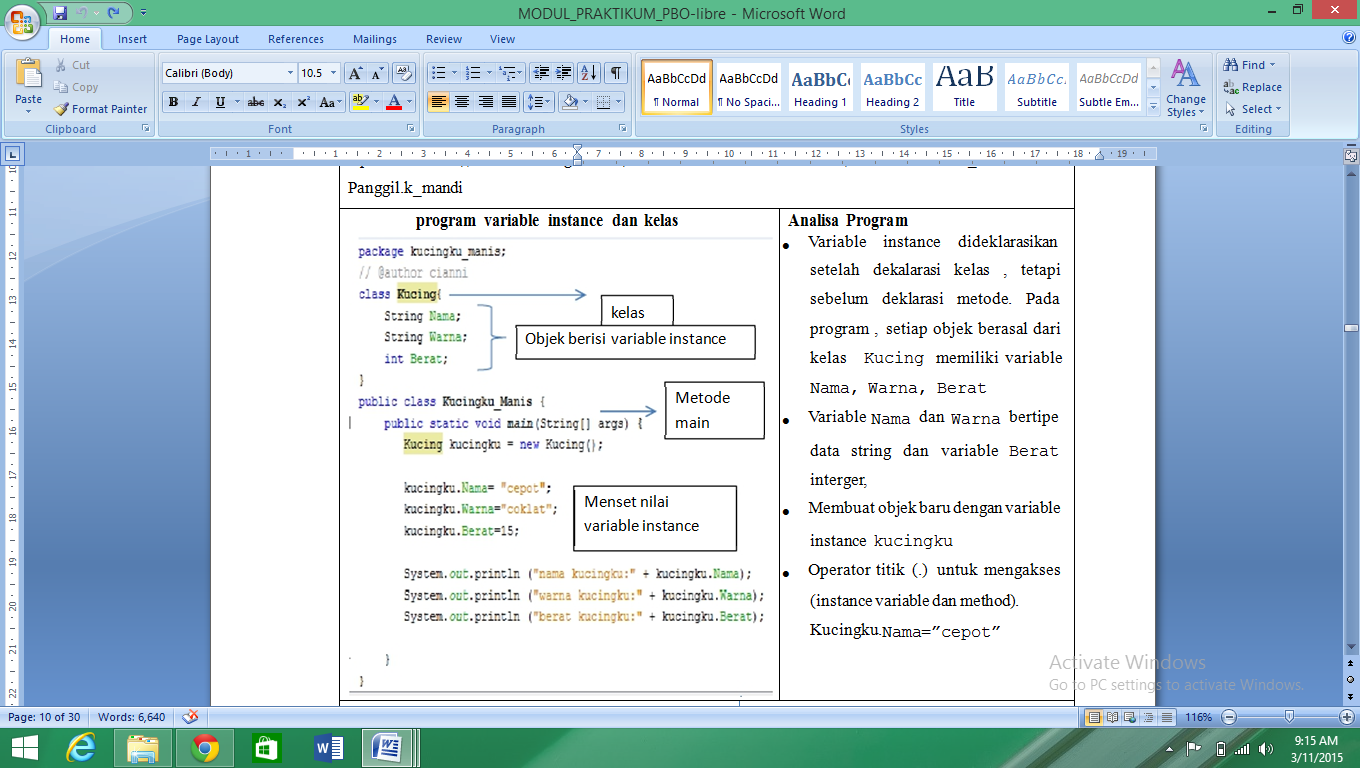
1. Pemberian nama kelas, variable instance, dan instansiasi objek terlihat pada source code berikut ini

****

1. Penggunaan kelas dan pemanggilan fungsi pada suatu main class.

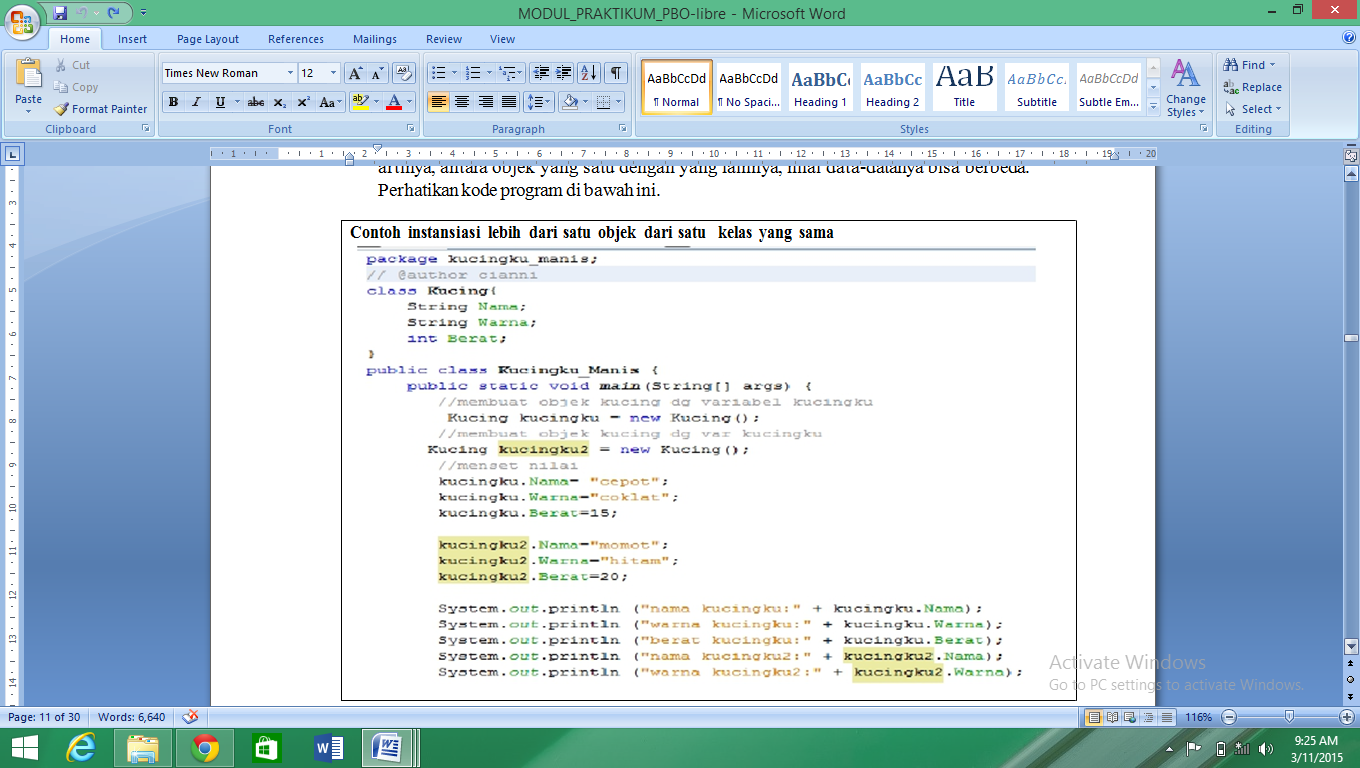
****

1. Penggunaan kelas dan pemanggilan fungsi pada suatu main class.

****

Setiap objek atau instance dari suatu kelas akan memiliki salinan data sendiri-sendiri. Ini artinya, antara objek yang satu dengan yang lainnya, nilai data-datanya bisa berbeda.

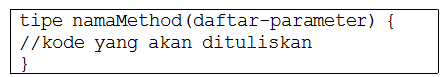
Perhatikan kode program di bawah ini.

****

PENGGUNAAN VARIABEL OLEH METHOD

Konsep Method

Dalam Java, kelas berisi kumpulan data dan method, yang selanjutnya akan saling Bekerjasama dalam melakukan tugas tugas spesifik tertentu sesuai dengan perilaku objek yang dimodelkan . Bentuk umum dari pembuatan method di dalam kelas.

****

**Pengembalian nilai**

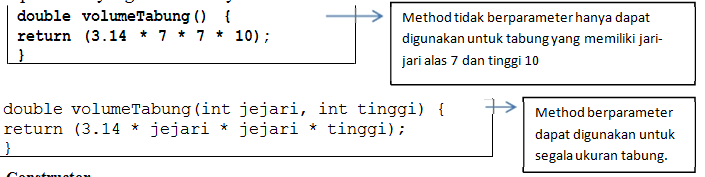
Java method terbagi menjadi dua: void dan non-void. Method void adalah method yang tidak mengembalikan nilai, sedangkan method non-void adalah method yang mengembalikan nilai.

Jika method yang kita buat ditujukan untuk mengembalikan suatu nilai tertentu, maka didalam method tersebut harus terdapat statemen return, yang diikuti dengan nilai yang akan dikembalikan, seperti berikut:

**Untitled2.png**

**Metode berparameter**

Pada kenyataannya sebagian besar method yang ditulis dalam program memiliki satu atau beberapa parameter. Dengan adanya parameter, sebuah method dapat bersifat dinamis dan general. Artinya, method tersebut dapat mengembalikan nilai yang beragam sesuai dengan nilai parameter yang dilewatkannya.



**Constructor**

Constructor adalah method khusus yang didefinisikan di dalam kelas dan akan dipanggil secara otomatis tiap kali terjadi instansiasi objek. Constructor itu sendiri berfungsi untuk melakukan inisialisasi nilai terhadap data-data yang terdapat pada kelas yang bersangkutan. Jika kita tidak mendefinisikan constructor pada kelas yang kita buat, maka secara otomatis Java akan membuatkannya untuk kita. Constructor semacam ini dinamakan dengan default constructor

**Keyword This**

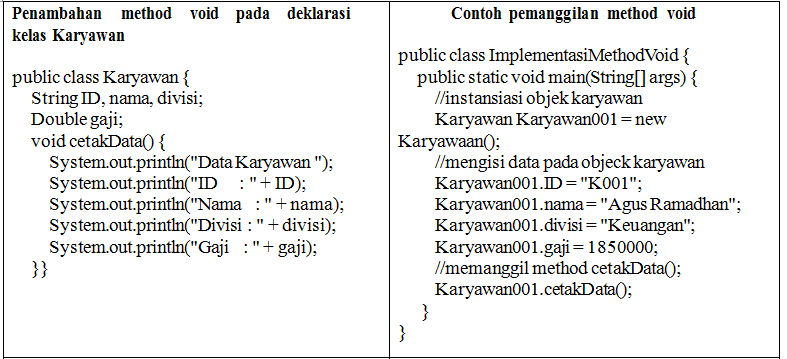
Java menyediakan kata kunci this. this merupakan referensi ke objek yang sedang aktif. this digunakan di dalam method untuk mewakili nama kelas bersangkutan. Overload terhadap method dan constructor Parameter dalam suatu method dikatakan berbeda dari method yang lainnya apabila:

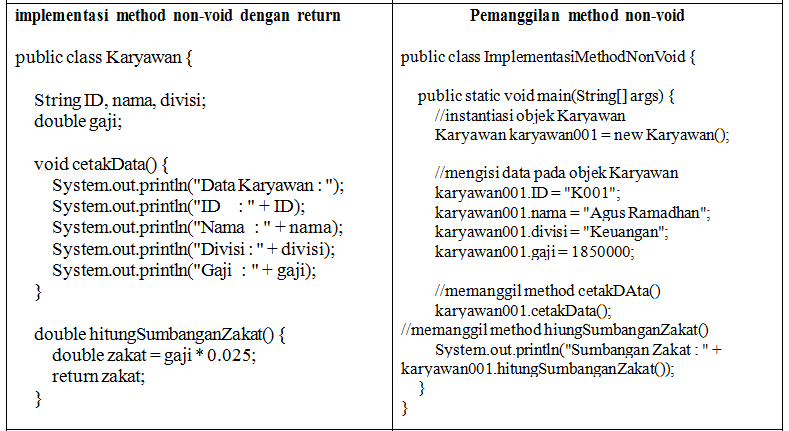
• Jumlahnya berbeda, meskipun tipe datanya sama

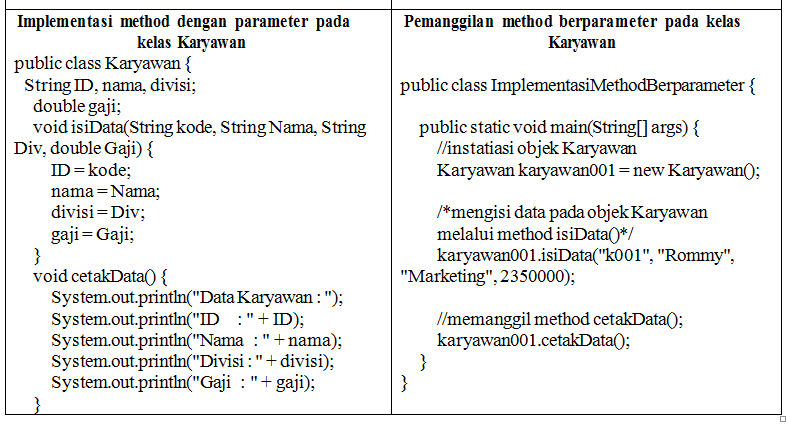
• Tipe datanya berbeda, meskipun jumlahnya samaProse pendefinisian method dengan nama sama ini disebut dengan overload.

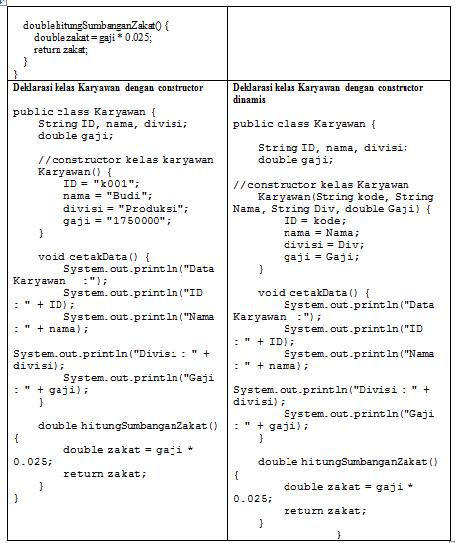
proses overload diaplikasikan ke dalam constructor suatu kelas. Hal ini disebabkan karena sebenarnya constructor juga adalah sebuah method yang mengembalikan tipe kelas (dirinya sendiri).

Latihan Penerapan Class pada JAVA

****

****

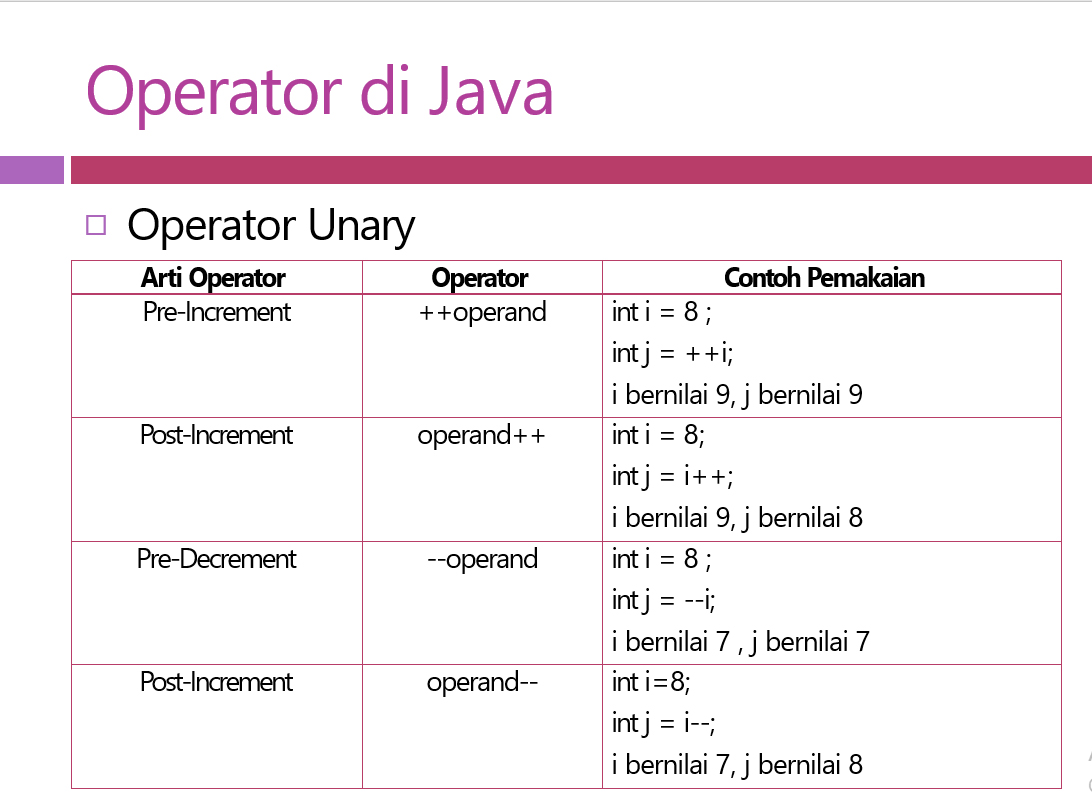


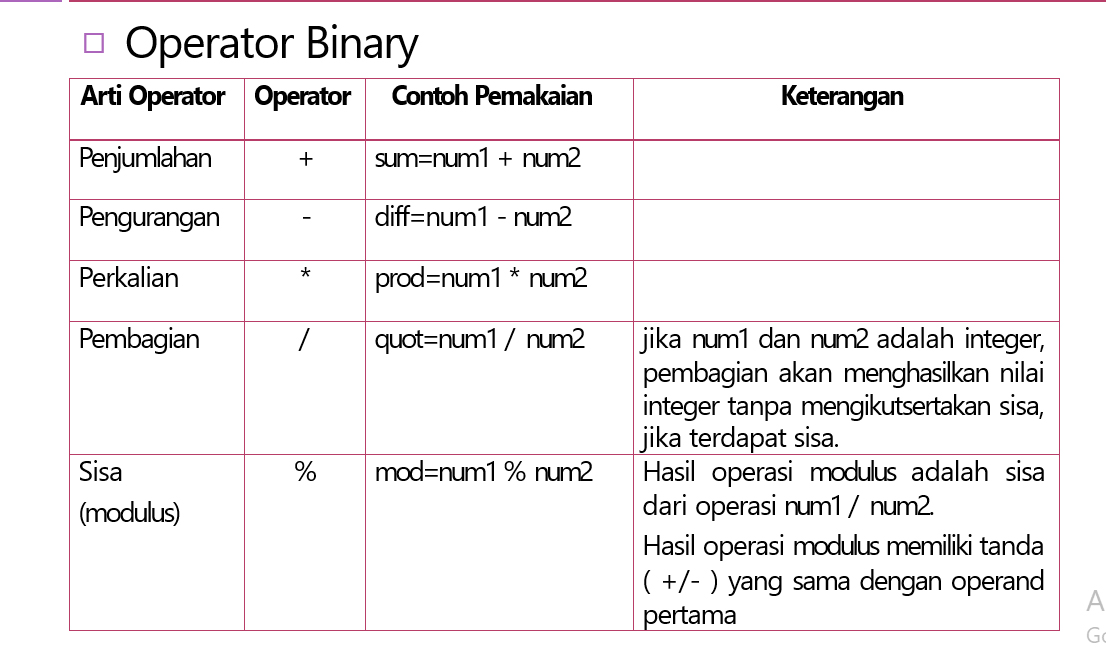
****

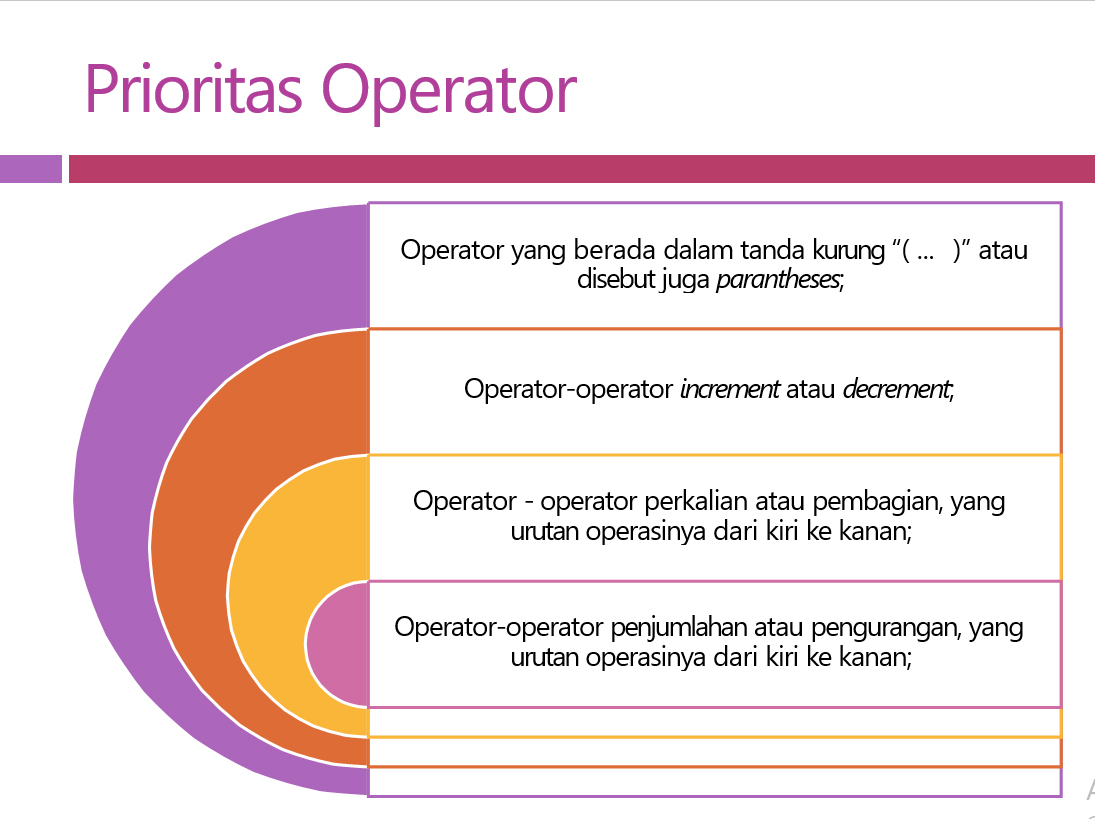
1. Penggunaan Operator

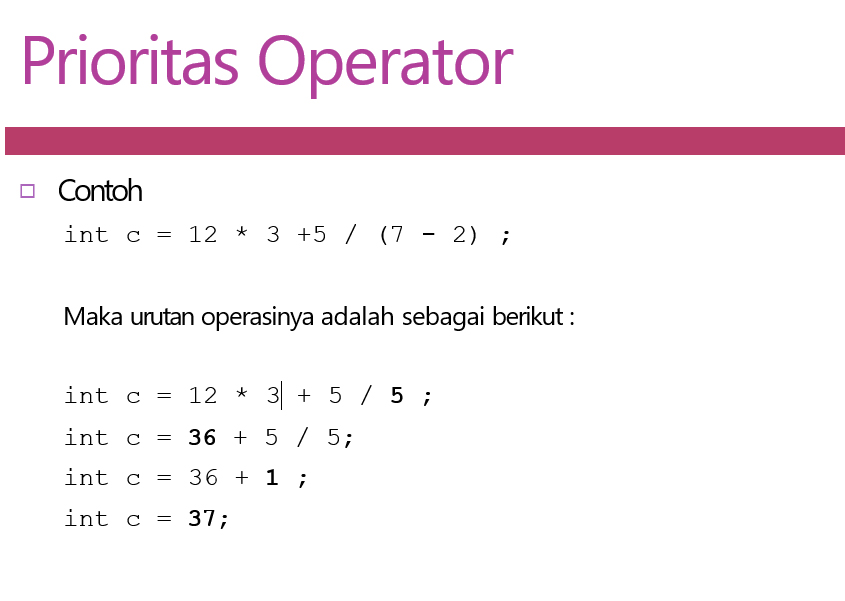
Beberapa contoh operator di JAVA antara lain yaitu:

1. Operator arithmatika (unary dan binary)
2. Operator pembanding
3. Operator Logika









**OPERATOR LOGIKA**

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

**Logika && (Logika AND)**

Hasil Operand Logika && (AND) akan bernilai **true** jika kedua operand bernilai **true** dan akan bernilai **false** jika ada salah satu operand yang bernilai **false**. Untuk lebih jelasnya coba perhatikan tabel kebenaran berikut ini.

|  |  |  |
| --- | --- | --- |
| **---- OPERAND1 ----** | **---- OPERAND2 ----** | **---- HASIL ----** |
| **true** | **true** | **true** |
| **true** | **false** | **false** |
| **false** | **true** | **false** |
| **false** | **false** | **false** |

Dan berikut ini merupakan contoh sederhana kode program yang menggunakan Logika && (AND).

public class operator\_logika {

public static void main(String[] args){

int operand1 = 48;

int operand2 = 70;

/\*\*

\* Demonstrasi1 penggunaan Logika AND, hasilnya akan true

\* karena kedua operand tersebut bernilai true

\*/

boolean test1 = operand1 > 20 && operand2 <= 100;

System.out.println(test1);

/\*\*

\* Demonstrasi2 penggunaan Logika AND, hasilnya akan false

\* karena ada salah satu operand yang bernilai false

\*/

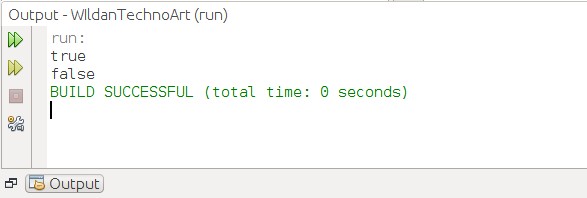
boolean test2 = operand1 < 20 && operand2 <= 100;

System.out.println(test2);

}

}

Hasilnya akan seperti berikut ini.



**Logika | | (Logika OR)**

Hasil Operand Logika || (Logika OR) akan bernilai **true** jika kedua operand bernilai **true**atau ada salah satu operand yang bernilai **true** dan akan bernilai **false** jika kedua operand bernilai **false**. Coba perhatikan tabel kebenaran berikut ini.

|  |  |  |
| --- | --- | --- |
| **---- OPERAND1 ----** | **---- OPERAND2 ----** | **---- HASIL ----** |
| **true** | **true** | **true** |
| **true** | **false** | **true** |
| **false** | **true** | **true** |
| **false** | **false** | **false** |

Perhatikan contoh program berikut ini.

public class operator\_logika {

public static void main(String[] args){

int operand1 = 80;

int operand2 = 120;

//Demonstrasi1 || (Logika OR)

boolean test1 = (operand1 == 80) || (operand2 < 10);

System.out.println(test1);

//Demonstrasi2 || (Logika OR)

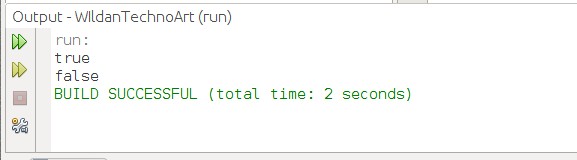
boolean test2 = (operand1 > 100) || (operand2 >= 200);

System.out.println(test2);

}

}

Jalankan maka hasilnya akan seperti ini.



**Logika | (boolean logika inclusive OR)**

Perbedaan dasar antara logika | | (OR) dan | (inclisive OR) adalah bahwa logika  | | (OR) mendukung **short-circuit** evaluations (proses evaluasi sebagian), sementara | tidak. Contoh penulisan kode programnya seperti berikut ini.

public class operator\_logika {

public static void main(String[] args){

int operand1 = 80;

int operand2 = 120;

//Demonstrasi || (Logika OR)

boolean test1 = (operand1 < 200) || (operand2++ != 400);

System.out.println(operand1);

System.out.println(operand2);

System.out.println(test1);

//Demonstrasi | (boolean Logika Inclusive OR)

boolean test2 = (operand1 < 200) | (operand2++ != 400);

System.out.println(operand1);

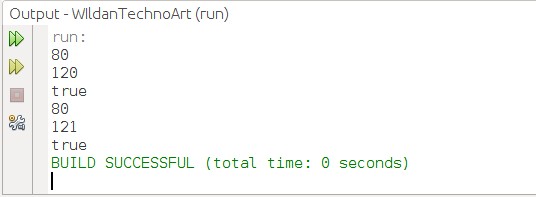
System.out.println(operand2);

System.out.println(test2);

}

}

Hasilnya:



Coba perhatikan, pada Logika | | (OR), variable **operand2++** tidak  akan di evaluasi karena pada operand1 sudah bernilai **true** jadi program tidak akan memperdulikan sesuatu yang terjadi pada operand2 tapi jika operand1 bernilai **false** maka program akan mengevaluasi operand2.

Pada logika | (boolean logika inclusive OR) operand2 akan tetap di evaluasi walaupun pada operand1 bernilai **true**.

**Logika ^ (boolean logika exclusive OR)**

Pada logika ^ (exclusive OR), operand akan bernilai **true** jika nilai boolean pada kedua operand tidak sama dan akan bernilai **false** jika nilai boolean pada kedua operand sama. Berikut ini merupakan tabel kebenaran untuk logika ^ (exclusive OR).

|  |  |  |
| --- | --- | --- |
| **---- OPERAND1 ----** | **---- OPERAND2 ----** | **---- HASIL ----** |
| **true** | **true** | **false** |
| **true** | **false** | **true** |
| **false** | **true** | **true** |
| **false** | **false** | **false** |

Dan berikut ini merupakan contoh program yang menggunakan logika ^ (exclusive OR).

public class operator\_logika {

public static void main(String[] args){

int operand1 = 77;

int operand2 = 30;

//Demonstrasi1 ^ (boolean logika exclusive OR)

boolean test1 = (operand1 > 100) ^ (operand2 != operand1);

System.out.println(test1);

//Demonstrasi2 ^ (boolean logika exclusive OR)

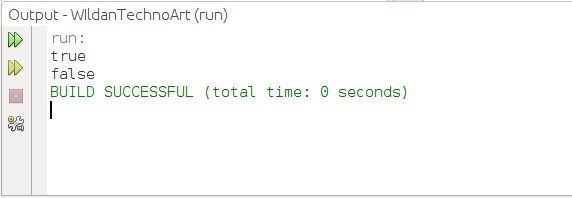
boolean test2 = (operand1 < 10) ^ (operand2 > 40);

System.out.println(test2);

}

}

Outputnya akan seperti berikut ini.



**Logika ! (Logika NOT)**

Logika ! (NOT) digunakan hanya untuk satu Operand saya, logika NOT akan menghasilkan **true** jika nilai pada boolean berisi **false** dan akan bernilai **false** jika nilai pada boolean berisi **true**, singkatnya operator ini merupakan kebalikan dari **true** menjadi **false** dan **false** menjadi **true**. Tebel kebenaran untuk logika NOT seperti ini.

|  |  |
| --- | --- |
| **---- OPERAND ----** | **---- HASIL ----** |
| **true** | **false** |
| **false** | **true** |

Untuk contoh programnya seperti berikut ini.

public class operator\_logika {

public static void main(String[] args){

int operand = 200;

//Demonstrasi1 ! (logika NOT)

boolean test1 = (operand > 100);

System.out.println(test1);

//Demonstrasi2 ! (logika NOT)

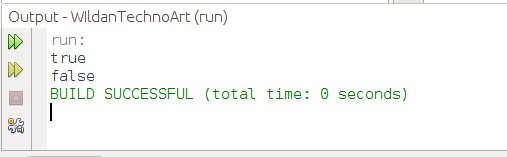
boolean test2 = (operand > 100);

System.out.println(!test2);

}

}

Hasilnya akan seperti ini.



# **DAFTAR PUSTAKA**

* Dimitrios Kalemis, The Fundamental of Object Oriented Programming, 2013, Prentice Hall, USA
* [Permanand Mohan](http://www.amazon.com/s/ref=dp_byline_sr_book_1?ie=UTF8&field-author=Permanand+Mohan&search-alias=books&text=Permanand+Mohan&sort=relevancerank), Fundamentals of Object-Oriented Programming in Java Paperback , 2013, OOPBook, USA
* Gandharba Swain, Object-Oriented Analysis and Design Through Unified Modeling Language, 2010, University Science Press, New Delhi
* Tim Bodreau, NetBeans: The Definitive Guide Paperback , 2002,O Reilly Inc, Canada
* [Ying Bai](http://www.amazon.com/Ying-Bai/e/B001H6MK22/ref=dp_byline_cont_book_1), Practical Database Programming with Java Paperback , 2011, Wiley Inc, USA