Operator

Operator adalah suatu tanda atau simbol yang dipakai untuk menyatakan suatu operasi atau manipulasi nilai. Contohnya adalah operasi penambahan direpresentasikan dengan tanda +, operasi pengurangan direpresentasikan dengan tanda -, dan sebagainya. Sedangkan operan adalah nilai yang dilibatkan oleh operator.

Bentuk operator
Berkenaan dengan banyaknya nilai (operan) yang dilibatkan oleh operator, maka operator dapat diklasifikasikan menjadi 2 bentuk, yaitu :

1. unary operator → membutuhkan 1 operan
2. binary operator → membutuhkan 2 operan
3. ternary operator → membutuhkan 3 operan

Unary operator
Unary operator (Operator unari) adalah operator yang melibatkan hanya satu operan dalam operasinya. Yang termasuk dalam operator unari ini adalah sebagai berikut :

Unary operator	Keterangan
+	Tanda plus
-	Tanda minus
++	Increment operator
--	Decrement operator
~	Complement operator
!	NOT operator

Contoh :

```
-5
```

yang berarti ada 1 unary operator yaitu – dan 1 buah operan yaitu 5.

Binary operator (Operator biner)
Binary operator (Operator biner) adalah operator yang melibatkan dua operan dalam operasinya. Yang termasuk dalam operator unari ini adalah semua operator yang tidak termasuk unary operator.

Contoh :

```
5 + 4
```

yang berarti ada 1 binary operator yaitu + dan 2 buah operan yaitu 5 dan 4.

Jenis operator
Java menyediakan berbagai macam jenis operator. Macam-macam operator ini mempunyai fungsi yang berbeda. Berbagai jenis operator ini dapat diklasifikan sebagai berikut :

- arithmatic operator (operator aritmatika)
- increment - decrement operator
- bitwise operator → komplemen + operasi logika pada biner
- boolean operator (operator boolean)
  → operator yang menghasilkan nilai Boolean true atau false; dlm C = operator logika
- relational operator (operator relasi)
- shift operator (operator geser)
- assignment operator (operator penugasan)

- combination operator (operator kombinasi)
- conditional operator (operator kondisi)

Arithmatic operator (operator aritmatika)
Arithmatic operator (operator aritmatika) adalah operator yang berfungsi untuk operasi aritmatika. Yang termasuk dalam arithmatic operator adalah sebagai berikut :

Arithmatic Operator	Keterangan
+	Operasi penambahan
-	Operasi pengurangan
*	Operasi perkalian
/	Operasi pembagian
%	Operasi modulus

Contoh program :

Berikut adalah program untuk mengitung nilai deskriminan D=b2 – 4ac dari suatu persamaan kuadrat ax2 + bx + c = 0

```
public class Deskriminan {
        public static void main(String args[]) {
                float a, b, c, d;

                a = 4.0F;
                b = 5.0F;
                c = 6.0F;
                d = b*b – 4*a*c;
                System.out.println(“Diskriminan = “ + d);
        }
}
```

Hasil eksekusi :

```
Diskriminan = -71.0
```

Increment - Decrement Operator
Increment – decrement operator adalah operator yang berguna untuk menaikkan 1 nilai (increment) dan menurunkan 1 nilai (decrement). Yang termasuk increment-decrement operator ini sebagai berikut :

Increment-Decrement Operator	Keterangan
++	increment
--	decrement

Contoh program :

```
public class IncDec {
        public static void main(String args[]) {
                int a=1, b=9;
                System.out.println(“Nilai sebelum increment-decrement”);
                System.out.println(“a = “ + a + “ ;  b = “ + b);
                a=++a;
                b=--b;
                System.out.println(“Nilai setelah increment-decrement”);
                System.out.println(“a = “ + a + “ ;  b = “ + b);
        }
}
```

Berdasarkan urutan eksekusi penaikkan dan penurunan nilainya, increment-decrement operator ini dapat diklasifikasikan menjadi 2 macam, yaitu :

- pre-increment/decrement
  Suatu nilai akan di-increment/decrement, baru nilai tersebut dipakai untuk perhitungan yang lain. Notasinya dituliskan dengan tanda increment/decrement operator yang diletakkan sebelum nilai/variabelnya
  Contoh :

```
int a, b=7;
a = ++b;
```

  Setelah baris kedua, a bernilai 8 dan b juga bernilai 8.

- post-increment/decrement
  Suatu nilai akan dipakai dulu, baru nilai tersebut di-increment/decrement. Notasinya dituliskan dengan tanda increment/decrement operator yang diletakkan setelah nilai/variabelnya.
  Contoh :

```
int a, b=7;
a = b++;
```

  Setelah baris kedua, a bernilai 7 dan b bernilai 8.

Bitwise operator

Bitwise operator adalah operator yang dipakai untuk operasi bit pada nilai operan. Yang termasuk bitwise operator ini adalah sebagai berikut :

Bitwise Operator	Keterangan
~	Operasi complement
&	Operasi AND
|	Operasi OR
^	Operasi XOR

Operator ~ (complement)

Operator ~ adalah operator yang berfungsi untuk melakukan operasi pembalikan bit pada nilai operan. Operasi ~ mempunyai ketentuan sebagai berikut :

Bit	Hasil ~
0	1
1	0

Contoh program :

```
public class Complement {
   public static void main(String args[]) {
      int i;
      i = ~7;
      System.out.println("Hasil operasi ~ : " + i);
   }
}
```

Hasil eksekusi :

```
Hasil operasi ~ = -8
```

Penjelasan program :

Susunan bit dari nilai 7 : 0000 0000 0000 0000 0000 0000 0000 0111
---------- ~
Bit hasil operasi ~ : 1111 1111 1111 1111 1111 1111 1111 1000 → -8

Catatan :
Bagaimana kita bisa membaca bahwa susunan representasi bit 1111 1111 1111 1111 1111 1111 1111 1000 adalah bernilai –8?. Caranya cukup mudah, yaitu sebagai berikut.

Bit yang paling kiri disebut dengan signed bit, yaitu bit yang menandakan positif atau negatifnya suatu nilai. Jika signed bit-nya bernilai 0, maka menandakan nilainya positif. Jika signed bit-nya bernilai 1, maka menandakan nilainya negatif. Susunan bit dalam contoh diatas bernilai 1, sehingga menandakan susunan bit tersebut bernilai negatif (-).

Susunan bit : 1111 1111 1111 1111 1111 1111 1111 1000
Lalu lakukan operasi complement terhadap seluruh bit, sehingga susunan bit akan berubah menjadi:

1111 1111 1111 1111 1111 1111 1111 1000
---------- ~
0000 0000 0000 0000 0000 0000 0000 0111

Kemudian tambahkanlah dengan nilai 1

0000 0000 0000 0000 0000 0000 0000 0111
1
---------- +
0000 0000 0000 0000 0000 0000 0000 1000 → 8

Karena signed bit pada susunan bit awal adalah 1, maka nilai akhirnya adalah –8.

Demikian juga sebaliknya jika kita ingin mengetahui nilai pada suatu susunan bit. Misalnya saja kita ingin mengetahui susunan bit dari nilai –4 (bertipe int). Caranya adalah sebagai berikut :
Nilai –4 menghasilkan bahwa signed bit dari susunan bit nantinya bernilai 1. Sekarang kita membuat susunan bit dari nilai 4.

0000 0000 0000 0000 0000 0000 0000 0100

Jika melakukan operasi penambahan dengan nilai 1 jika ingin mengetahui nilai dari susunan bit, maka sebaliknya kita harus melakukan operasi pengurangan dengan nilai 1 jika ingin mengetahui susunan bit dari suatu nilai.

0000 0000 0000 0000 0000 0000 0000 0100
1
---------- -
0000 0000 0000 0000 0000 0000 0000 0011

Kemudian kita lakukan operasi complement sebagai berikut

0000 0000 0000 0000 0000 0000 0000 0011
---------- ~
1111 1111 1111 1111 1111 1111 1111 1100

Karena nilai awalnya adalah –4, maka harus kita pastikan bahwa susunan bit tersebut mempunyai signed bit bernilai 1. Jika susunan bit terakhir masih mempunyai signed bit bernilai 0, maka kita harus menggantinya dengan nilai 1.
Jadi nilai –4 mempunyai representasi bit sebagai berikut :

1111 1111 1111 1111 1111 1111 1111 1100

Operator & (AND)
Operator & adalah operator yang berfungsi untuk melakukan operasi AND pada bit-bit nilai operan. Operasi AND mempunyai ketentuan sebagai berikut :

Bit 1	Bit 2	Hasil &
0	0	0
0	1	0
1	0	0
1	1	1

Contoh program :

```
public class And {
   public static void main(String args[]) {
      int i;
      i = 6 & 13;
      System.out.println("Hasil operasi & = " + i);
   }
}
```

Hasil eksekusi :

```
Hasil operasi &  = 4
```

Penjelasan program :

Susunan bit dari nilai 6 : 0000 0000 0000 0000 0000 0000 0000 0110
Susunan bit dari nilai 13 : 0000 0000 0000 0000 0000 0000 0000 1101
---------- &
Bit hasil operasi & : 0000 0000 0000 0000 0000 0000 0000 0100 → 4

Operator | (OR)
Operator | adalah operator yang berfungsi untuk melakukan operasi OR pada bit-bit nilai operan. Operasi OR mempunyai ketentuan sebagai berikut :

Bit 1	Bit 2	Hasil |
0	0	0
0	1	1
1	0	1
1	1	1

Contoh program :

```
public class Or {
   public static void main(String args[]) {
      int i;
      i = 5 | 9;
      System.out.println("Hasil operasi 5 | 9 = " + i);
   }
}
```

Hasil eksekusi : Hasil operasi 5 | 9 = 13

Penjelasan program :

Susunan bit dari nilai 5 : 0000 0000 0000 0000 0000 0000 0000 0101
Susunan bit dari nilai 9 : 0000 0000 0000 0000 0000 0000 0000 1001
---------- |
Bit hasil operasi | : 0000 0000 0000 0000 0000 0000 0000 1101 → 13

Operator ^ (XOR)
Operator ^ adalah operator yang berfungsi untuk melakukan operasi XOR pada bit-bit nilai operan. Operasi XOR mempunyai ketentuan sebagai berikut :

Bit 1	Bit 2	Hasil ^
0	0	0
0	1	1
1	0	1
1	1	0

Contoh program :

```
public class Xor {
   public static void main(String args[]) {
      int i;
      i = 5 ^ 9;
      System.out.println("Hasil operasi 5 ^ 9 = " + i);
   }
}
```

Hasil eksekusi : Hasil operasi 5 ^ 9 = 12

Penjelasan program :

Susunan bit dari nilai 5 : 0000 0000 0000 0000 0000 0000 0000 0101
Susunan bit dari nilai 9 : 0000 0000 0000 0000 0000 0000 0000 1001
---------- ^
Bit hasil operasi ^ : 0000 0000 0000 0000 0000 0000 0000 1100 → 12

Relational operator (operator relasi) **→ true atau false**
→ dibaca : apakah

Relational operator (operator relasi) adalah operator yang sering dipakai untuk operasi perbandingan dan selalu menghasilkan suatu nilai bertipe boolean (true atau false). Yang termasuk logical operator adalah sebagai berikut :

Relational Operator	Keterangan
= =	Operasi perbandingan sama dengan
!=	Operasi perbandingan tidak sama dengan
>	Operasi perbandingan lebih besar
>=	Operasi perbandingan lebih besar sama dengan
<	Operasi perbandingan lebih kecil
<=	Operasi perbandingan lebih kecil sama dengan

Contoh program :

```
public class Equal {
        public static void main(String args[]) {
                int a=6;

                if ((a % 2) != 0)
                        System.out.println("Bilangan ganjil");
                else
                        System.out.println("Bilangan genap");
        }
}
```

Boolean operator (operator boolean)

Boolean operator (operator boolean) adalah operator yang mengharuskan operannya bertipe boolean (true atau false). Yang termasuk boolean operator adalah sebagai berikut :

Logical Operator	Keterangan
!	Operasi negasi (NOT)
&	Operasi AND
|	Operasi OR
^	Operasi XOR
&&	Operasi AND (short circuit)
||	Operasi OR (short circuit)

Operator !, &, | dan ^ mempunyai implementasi yang sama sebagaimana ketika ia menjadi bitwise operator. Hanya saja di logical operator, operan yang dilibatkan disini harus bertipe boolean, yang hanya mempunyai nilai true atau false.

Operator && (short circuit)

Operator && adalah operator yang berfungsi untuk melakukan operasi AND pada bit-bit nilai operan yang bertipe boolean. Operator && mempunyai ketentuan sebagaimana ketentuan operator &. Hanya saja perbedaannya terletak pada saat operasi AND sudah dapat menghasilkan suatu nilai dari operan pertama, maka operasi AND dengan && tidak lagi melihat operan kedua, sedangkan operasi AND dengan & masih melihat operan kedua. Oleh karena itu operator && disebut juga dengan short circuit boolean operator. Sehingga operator && mempunyai ketentuan sebagai berikut:

Operan 1	Operan 2	Hasil &&
false	Tidak dilihat	false
false	Tidak dilihat	false
true	0	false
true	1	true

Berikut adalah 2 contoh program sederhana untuk membedakan pemakaian operator & dan operator &&.

```
public class BooleanAnd {
   public static void main(String args[]) {
      int a=5, b=7;
      if ((a<2) & (b++<10))
         b+=2;
      System.out.println(b);
   }
}
```

Hasil eksekusi :

```
8
```

```
public class ShortCircuitBooleanAnd {
   public static void main(String args[]) {
      int a=5, b=7;
      if ((a<2) && (b++<10))
         b+=2;
      System.out.println(b);
   }
}
```

Hasil eksekusi :

```
7
```

Penjelasan program :

Pada class BooleanAnd, operasi perbandingan menggunakan operator &, kedua operan dilibatkan. Oleh karena itu operan yang kedua dilihat dan disana terdapat post-increment. Setelah operan kedua dibandingkan, nilai b di-increment satu, sehingga nilai b menjadi 8.

Sedangkan pada class ShortCircuitBooleanAnd, operasi perbandingan menggunakan operator &&. Saat mengecek nilai operan pertama, didapatkan nilainya false. Karena nilai operan pertama false, maka hasil dari operasi perbandingan pasti bernilai false, bagaimanapun nilai yang ada pada operan kedua. Sehingga operan kedua dalam operasi perbandingan tersebut tidak dilihat. Oleh karena itu, b tetap bernilai 7.

Operator || (short circuit)

Operator || adalah operator yang berfungsi untuk melakukan operasi OR pada bit-bit nilai operan yang bertipe boolean. Operator || mempunyai ketentuan sebagaimana ketentuan operator |. Perbedaannya sama sebagaimana perbedaan antara operator & dan &&, yaitu terletak pada saat operasi OR sudah dapat menghasilkan suatu nilai dari operan pertama, maka operasi OR dengan || tidak lagi melihat operan kedua, sedangkan operasi OR dengan | masih melihat operan kedua. Oleh karena itu operator || disebut juga dengan short circuit boolean operator. Sehingga operator || mempunyai ketentuan sebagai berikut:

Operan 1	**Operan 2**	**Hasil ||**
false	false	false
false	true	true
true	Tidak dilihat	true
true	Tidak dilihat	true

Berikut adalah 2 contoh program sederhana utk membedakan pemakaian operator | dan operator ||.

```
public class BooleanOr {
   public static void main(String args[]) {
      int a=5, b=7;
      if ((a>2) | (b++<10))
         b+=2;
      System.out.println(b);
   }
}
```

Hasil eksekusi :

```
10
```

```
public class ShortCircuitBooleanOr {
   public static void main(String args[]) {
      int a=5, b=7;
      if ((a>2) || (b++<10))
         b+=2;
      System.out.println(b);
   }
}
```

Hasil eksekusi :

```
9
```

Shift operator (operator geser)
Shift operator (operator geser) adalah operator yang berfungsi untuk menggeser susunan bit pada suatu nilai. Yang termasuk dalam shift operator ini adalah sebagai berikut :

Shift Operator	Keterangan
>>	right shift
>>>	unsigned right shift
<<	left shift

Operator >>

Operator ini berfungsi untuk menggeser susunan bit pada suatu nilai ke arah kanan dimana signed bit (bit yang terletak di sebelah paling kiri) akan diisi dengan nilai signed bit yang sebelumnya. Karena signed bit sesudah pergeseran diisi dengan signed bit yang sebelumnya, maka suatu nilai yang positif, maka setelah pergeseran nilai tersebut tetap positif. Demikian juga suatu nilai yang negatif, setelah pergeseran niali tersebut tetap negatif.

Contoh program :

```
public class RightShift {
   public static void main(String args[]) {
      int i=7;
      i=i >> 2;
      System.out.println(i);
   }
}
```

Hasil eksekusi :

```
1
```

Penjelasan program :

Susunan bit 7 : 0000 0000 0000 0000 0000 0000 0000 0111
Geser ke kanan 2 kali : 0000 0000 0000 0000 0000 0000 0000 0001 → 1

Operator >>>

Operator ini berfungsi untuk menggeser susunan bit pada suatu nilai ke arah kanan dimana signed bit (bit yang terletak di sebelah paling kiri) akan diisi dengan bit 0. Karena signed bit sesudah pergeseran diisi dengan bit 0, maka suatu nilai apapun hasil operasi pergeseran selalu positif. Oleh karena itu operator ini dinamakan unsigned right shift

Contoh program :

```
public class UnsignedRightShift {
   public static void main(String args[]) {
      int i = -1;
      i = i >>> 30;
      System.out.println(i);
   }
}
```

Hasil eksekusi :

```
3
```

Penjelasan program :

Susunan bit -1 : 1111 1111 1111 1111 1111 1111 1111 1111
Geser ke kanan 30 kali : 0000 0000 0000 0000 0000 0000 0000 0011 → 3

Operator <<

Operator ini berfungsi untuk menggeser susunan bit pada suatu nilai ke arah kiri dimana bit yang terletak di sebelah paling kanan akan diisi dengan bit 0.

Contoh program :

```
publicœlass LeftShift {
   public static void main(String args[]) {
      int i=3;
      i=i << 2;
      System.out.println(i);
   }
}
```

Hasil eksekusi :

```
12
```

Penjelasan program :

Susunan bit 3 : 0000 0000 0000 0000 0000 0000 0000 0011
Geser ke kanan 2 kali : 0000 0000 0000 0000 0000 0000 0000 1100 → 12

Assigment Operator (operator penugasan)

Assigment operator (operator penugasan) adalah operator ini berfungsi untuk menyalin suatu nilai yang terletak di sebelah kanannya ke dalam suatu variabel yang terletak di sebelah kirinya. Assigment operator ini hanya mempunyai 1 macam operator, yaitu =.

Contoh program :

```
int a;
a=7+5:
```

Pada baris kedua diberikan assigment operator, yang bertugas menyalin nilai di sebelah kanannya (7+5), ke dalam variabel a yang terletak di sebelah kirinya, sehingga sesudah baris kedua dieksekusi, variabel a akan bernilai 12.

Combination operator (operator kombinasi)

Combination operator (operator kombinasi) adalah operator yang terdiri dari gabungan 2 operator. Biasanya combination operator ini dipakai untuk mempersingkat waktu penulisan program. Yang termasuk operator combination ini adalah :

Combination Operator	Keterangan
+=	Gabungan dari operator = dan +
-=	Gabungan dari operator = dan -
*=	Gabungan dari operator = dan *
/=	Gabungan dari operator = dan /
%=	Gabungan dari operator = dan %
>>=	Gabungan dari operator = dan >>
>>>=	Gabungan dari operator = dan >>>
<<=	Gabungan dari operator = dan <<
&=	Gabungan dari operator = dan &
|=	Gabungan dari operator = dan |
^=	Gabungan dari operator = dan ^

Contoh :

Contoh	Keterangan
x += 5	x = x + 5
x *= 2	x = x * 2
x >>= 7	x = x >> 7

Conditional operator (operator konditional)

Conditional operator (operator konditional) adalah operator yang dipakai untuk operasi kondisi (persyaratan), sama sebagaimana if-then-else dan hanya berlaku untuk pernyataan tunggal. Operator ini mengembalikan suatu nilai yang benar sesuai dengan kondisi yang diberikan. Conditional operator (operator konditional) ini hanya ada 1 macam, yaitu ? disertai dengan tanda : (titik dua). Jika kondisi persyaratan yang terletak di sebelah kiri tanda ? bernilai benar, maka pernyataan yang berada di sebelah kiri tanda : yang akan diambil. Demikian juga sebaliknya, jika kondisi persyaratan bernilai salah, maka pernyataan yang berada di sebelah kanan tanda : yang akan diambil.

Contoh program :

```
public class ConditionalOp {
   public static void main(String args[]) {
      int nilai=55;
      boolean lulus;

      lulus=(nilai>=60) ? true : false;
      System.out.println("Anda lulus? " + lulus);
   }
}
```

Penjelasan program :

Karena kondisi persyaratan (nilai>=60) bernilai salah, maka false yang akan diambil dan diberikan ke variabel lulus, sehingga variabel lulus bernilai false.