Praktikum 3

Operator dan Asignment

Tujuan

Mengenal dan memahami bentuk dan jenis operator yang ada di Java, serta dapat mengaplikasikannya dengan tepat dalam bahasa pemrograman Java.

Dasar Teori

Operator dapat diklasifikasikan menjadi 2 bentuk, yaitu unary operator dan binary operator. Unary operator adalah operator yang hanya melibatkan 1 operan. Sedangkan binary operator adalah operator yang melibatkan 2 operan. Java mempunyai berbagai macam jenis operator, dapat digolongkan menjadi operator aritmatika, increment decrement, bitwise, boolean, logika, shift (geser), penugasan, kombinasi dan kondisi.

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

Increment – decrement operator adalah operator yang berguna untuk menaikkan 1 nilai (increment) dan menurunkan 1 nilai (decrement). Berdasarkan urutan eksekusi penaikkan dan penurunan nilainya, increment-decrement operator ini dapat diklasifikasikan menjadi 2 macam, yaitu pre-increment/decrement dan post-increment/decrement. Yang termasuk increment-decrement operator ini sebagai berikut :

Increment-Decrement Operator	Keterangan
++	increment
--	decrement

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

Boolean operator (operator boolean) adalah operator yang mengharuskan operannya bertipe boolean (true atau false). 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. 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)

Logical operator (operator logika) 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:

Logical 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

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

Logical 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

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:

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

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 kondisional) 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.

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 ^

A. Unary Operator, Aritmatika Operator, Shift Operator, Comparison Operator

Percobaan 1:

Percobaan berikut ini menunjukkan *unary operator* untuk fungsi *increment* dan *decrement.*

```
public class IncrementDecrement {

    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 sebelum increment-decrement");
        System.out.println("a="+a+";b="+b);
    }
}
```

Percobaan 2:

Percobaan berikut ini menunjukkan *unary operator* untuk fungsi *complement.*

```
public class Complement {

    public static void main(String args[]){
        int i;
        i =~-7;//mengubah 1 bit biner ke 0s dan 0 bit ke 1s

        System.out.println("Hasil operasi ~ : "+i);
    }
}
```

Percobaan 3:

Percobaan berikut ini menunjukkan pemakaian *unary operator* untuk beberapa tipe data.

```
public class TestUnaryOperator {

    public static void main(String[] args){
        byte b;
        int i;
        long l = 0L;
        double d = 0D;

        System.out.println("!(false) = \t\t\t->"+ !(false) );
        System.out.println("!(true) = \t\t\t->"+ !(true) );

        b = -5;
        System.out.println("b = -5  \t\t\t ->"+b);

        b =(~5) +1;
        System.out.println("b = (~5)+1 \t\t\t -> " +b);
```

```
        i = -(-2147483648);
        System.out.println("i = -(-2147483648) \t\t ->" +i);

        i = (~i) +1;
        System.out.println("i = (~i) +1 \t\t\t -> " +i);

        l = -(-9223372036854775808L) ;
        System.out.println("l = -(-9223372036854775808L) \t -> " +1);

        l= (~1) +1;
        System.out.println("l = (~1) +1 \t\t\t -> " +1);

        d = -(15.63);
        System.out.println("d = -(15.63) \t\t\t -> " +d);

        d= -(-15.63);//nilai d berubah menjadi positif
        System.out.println("d= -(-15.63) \t\t\t -> " +d);
    }
}
```

Percobaan 4:

Percobaan berikut ini menunjukkan pemakaian *unary operator* untuk fungsi konversi atribut ke tipe data lainnya.

```
public class TestConversion {

     public static void main ( String[] args) {
        double d = 2.12345D;
        float f = 150.50F;
        long l = 15000L;
        int i = 55;
        char c = 20;
        short s = 1000;
        byte b = 126;

        System.out.println();
        System.out.println("Implicit Widening Conversions: ");
        System.out.println("------------------------------ ");
        System.out.println("byte to short : \t  -> " +(s = b));
        System.out.println("short to in : \t    -> " +(i = s));
        System.out.println("int to long : \t    -> " +(l = i));
        System.out.println("long to float : \t  -> " +(f = l));
        System.out.println("float to double :\t -> " +(d = f));
        System.out.println();
        System.out.println("Explicit Widening conversions: ");
        System.out.println("------------------------------ ");
        System.out.println("cast byte to char : \t  -> " + (char)b);
        System.out.println("cast short to char : \t -> " + (char)s);

        d = 150;
        System.out.println();
        System.out.println("Explicit Narrowing conversions: ");
        System.out.println("------------------------------ ");
        System.out.println("double to float: -> " +(f = (float)d));
        System.out.println("float to long :  ->" +(l = (long)f));
        System.out.println("long to int :    -> " +(i =(int) l));
        System.out.println("int to short :   -> " +(s= (short)i));
        System.out.println("short to byte :  -> " +(b = (byte)s));
    }
}
```

Percobaan 5:

Percobaan berikut ini menunjukkan pemakaian *arithmatic operator* pada beberapa tipe data.

```
public class TestArithmetic {

    public static void main (String[] args){
        System.out.println("Integer Division - results truncated :");
        System.out.println("---------------------------------------");
        System.out.println("\t 10.3/3 \t = " + (10/3));
        System.out.println("\t 10/-3  \t = " + (10/-3));
        System.out.println("\t -10/3  \t = " + (-10/3));

        System.out.println();
        System.out.println("Floating-point Division by 0:");
        System.out.println("-----------------------------");
        System.out.println("\t 10.34 /0 \t = " + (10.34/0));
        System.out.println("\t -10.34 /0\t = " + (-10.34 /0));
        System.out.println("\t 10.34/-0 \t = " + (10.34/-0));
        System.out.println("\t 0.0/0 \t    = " + (0.0/0));
        System.out.println("\t 0.0/ -0 \t  = " + (0.0/ -0) );

        System.out.println();
        System.out.println("Modulo operations : ");
        System.out.println("\t 5% 3\t   = " + (5%3) );
        System.out.println("\t -5%3 \t  = " + (-5%3));
        System.out.println("\t 5%-3 \t  = " + (5%-3));
        System.out.println("\t 5.0%3 \t = " + (5.0%3));
        System.out.println("\t 5.0%-3 \t= " + (5.0%-3));
        System.out.println("\t -5.0%3 \t= " + (-5.0%3));
        System.out.println("\t 5.0%0 \t = " + (5.0%0));
    }
}
```

Percobaan 6:

Percobaan berikut ini menunjukkan pemakaian *shift operator* pada tipe data integer.

```
public class ShiftInteger {

    public static void main (String args[]) {
        int x =7;
        System.out.println("x=" +x);

        System.out.println("Right shift n-bit: ");
        System.out.println("x>>1= " + (x>>1));
        System.out.println("x>>2= " + (x>>2));

        System.out.println("Left shift n-bit: ");
        System.out.println("x<<1= " + (x<<1));
        System.out.println("x<<2= " + (x<<2));

        System.out.println("Unsigned right shift n-bit: ");
        System.out.println("x>>>1=" + (x>>>1));
        System.out.println("x>>>2=" + (x>>>2));
    }
}
```

Percobaan 7:

Percobaan berikut ini menunjukkan pemakaian *comparison operator* (pembanding) pada fungsi relasional.

```
public class Relational {

    public static void main(String [] args){
        int x=5;
        int y=6;
        int z=5;
        float f0=0.0F;
        float f1=-0.0F;
        float f2=5.0F;

        System.out.println("Relational operators: ");
        System.out.println("---------------------");

        System.out.println("Less than: 5<6                      "+(x<y));
        System.out.println("Less than or equal to: 5<=5        "+(x<=z));
        System.out.println("Greater than: 5>6                   "+(x>y));
        System.out.println("Greater than or equal to: 5>=5     "+(x>=z));
        System.out.println();
        System.out.println("Less than: -0.0 < 0.0             "+(f1<f0));
        System.out.println("Less than or equal to: -0.0<=0.0 "+(f1<f0));
        System.out.println("Greater than: 5 > NaN (Not a Number)"
        +(x>(f0/f1)));
    }
}
```

Percobaan 8:

Percobaan berikut ini menunjukkan pemakaian *comparison operator* (pembanding) pada fungsi *equality* (persamaan).

```
public class Equality {

    public static void main(String[] args) {
        int x = 5;
        float f2 = 5.0f;
        int arr1[] = {1,2,3};
        int arr2[] = {4,5,6};
        int arr3[] = arr1;
        String s1 = "hello";
        String s2 = "hello";
        String s3 = s1;
        String s4 = new String("hello");

        System.out.println();
        System.out.println("Equality operator :");
        System.out.println("--------------------");
        System.out.println();
        System.out.println("Equals: 5== 5.0 \t          "+ (x==f2) );
        System.out.println("Not Equals: 5 != 5.0\t      "+ (x!=f2) );

        System.out.println("Equals: arr1 == arr2\t      "+ (arr1==arr2)+
"[different array object]");

        System.out.println("Equals: arr1 == arr3\t      "+
```

```
(arr1==arr3)+"[ref to same array objects]");

        System.out.println("Not Equals: arr1 != arr2 \t"+ (arr1!=arr2));
        System.out.println("Not Equals: arr1 != arr3 \t"+ (arr1!=arr3));

        System.out.println("Equals; s1 == s2 \t        "+ (s1 == s2) +
"[same literal]");

        System.out.println("Equals; s1 == s3 \t        "+ (s1 == s3) +
"[same object reference]");

        System.out.println("Equals: s1 == s4 \t        "+ (s1 == s4) +
"[s4 is new object]");
    }
}
```

Percobaan 9:

Percobaan berikut ini menunjukkan pemakaian *comparison operator* (pembanding) pada fungsi *instance of* .

```
public class InstanceOf {

    public static void main(String[] args){
        int arr1[]={1,2,3};
        String[] arr= new String[5];
        String[] arr4=arr;
        Object myNull = null;
        Object o= new Object();

        System.out.println();
        System.out.println("InstanceOf operator");
        System.out.println("--------------------");
        System.out.println();
        System.out.println("arr instanceof String[] \t -> " +( arr
instanceof String[]));
        System.out.println("myNull instanceof Object\t -> " +( arr4
instanceof Object));
        System.out.println("arr1 instanceof int[] \t   -> " +( arr1
instanceof int[]));
    }
}
```

B. Bitwise Operator, Conditional Operator, Assignment Operator, Logical Operator

Percobaan 10:

Percobaan berikut ini menunjukkan pemakaian *bitwise operator* pada fungsi *AND, OR* dan *XOR* .

```
public class Bitwise {

    public static void main(String[] args) {
       int x=5,y=6;
       System.out.println("x=" +x);
       System.out.println("y=" +y);
       System.out.println();
```

```
        System.out.println("Bitwise operator AND,OR ,XOR ");
        System.out.println("x&y=" +(x&y));
        System.out.println("x|y=" +(x|y));
        System.out.println("x^y=" +(x^y));
    }
}
```

Percobaan 11:

Percobaan berikut ini menunjukkan pemakaian *logical operator* pada tipe data *boolean*.

```
public class Logical {

    public static void main(String[] args) {
        boolean a=true;
        boolean b=true;
        boolean c=false;
        boolean d=false;

        System.out.println();
        System.out.println("Logical operator:");
        System.out.println("-----------------");

        System.out.println();
        System.out.println("\t true & true  = \t"+(a&b));
        System.out.println("\t true & false = \t"+(a&c));
        System.out.println("\t true ^ false = \t"+(a^c));
        System.out.println("\t true ^ true  = \t"+(a^b));
        System.out.println("\t true | false = \t"+(a|b));
        System.out.println("\t true | false = \t"+(c|d));

        System.out.println();
        System.out.println("\t true && true = \t"+(a&&b));
        System.out.println("\t false && true= \t"+(c&&a));

        System.out.println();
        System.out.println("\t false||true  = \t"+(c||a));
        System.out.println("\t false||false = \t"+(c||d));
        System.out.println("\t true ||false = \t"+(a||d));
        System.out.println("\t true ||true  = \t"+(a||b));
    }
}
```

Percobaan 12:

Percobaan berikut ini menunjukkan pemakaian beberapa operator pada Short Circuit.

```
public class ShortCircuit {

    public static void main(String[] args) {
        int x=0;
        int y=0;
        boolean A=true;
        boolean B= false;

        System.out.println("Test with normal operator '&', value of x
increase.");

        if(A==B & x++ <0){}
```

```
        System.out.println("After test x ="+x+ "y="+y+"\n");
        x=0;
        y=0;

        System.out.println("test with short circuit operator '&&', value
of y does not change");

        if(A==B && y++ <0){}
        System.out.println("After test x ="+x+"y="+y);
    }
}
```

Percobaan 13:

Percobaan berikut ini menunjukkan pemakaian operator *"equal(…)"*

```
public class BooleanEquals {

    public static void main(String[] args){
        // Object equals()
        Class_A a =  new Class_A();
        Class_B b =  new Class_B();
        Class_A c =  new Class_A();
        Class_B d =  b;
        Class_A e =  null;

        System.out.println("a.equals(b) \t -> " + (a.equals(b) );
        System.out.println("a.equals(c) \t -> " + (a.equals(c) );
        System.out.println("b.equals(d) \t -> " + (b.equals(d) );
        System.out.println("a.equals(e) \t -> " + (a.equals(e) );

        // String equals()
        String s0 = "Hello";
        String s1 = new String("Hello");
        String s2 = s0;
        System.out.println("s0.equals(s1) \t -> " + (s0.equals(s1) );
        System.out.println("s0.equals(s2) \t -> " + (s0.equals(s2) );

        // Boolean equals()
        Boolean b0 = new Boolean(true);
        Boolean b1 = new Boolean(false);
        Boolean b2 = new Boolean(true);
        Boolean b3 = b1;
        System.out.println("b0.equals(b1) \t -> " + (b0.equals(b1) );
        System.out.println("b0.equals(b2) \t -> " + (b0.equals(b2) );
        System.out.println("b3.equals(b1) \t -> " + (b3.equals(b1) );
    }

}

class Class_A(){
    // null class
}

class Class_B() extends Class_A{
    // null class
}
```

Percobaan 14:

Percobaan berikut ini menunjukkan pemakaian *conditional operator* untuk membandingkan bilangan ganjil/genap dengan identifier *"isEven"*.

```
public class Conditional01 {

    public static void main(String[] args) {
        int x=0;
        boolean isEven=false;
        System.out.println("x="+x);
        x=isEven?4:7;
        System.out.println("x="+x);
    }
}
```

Percobaan 15:

Percobaan berikut ini menunjukkan pemakaian *conditional operator* dengan fungsi relasi.

```
public class Conditional02 {

    public static void main(String[] args) {
        int nilai=55;
        boolean lulus;
        lulus=(nilai>=60)?true:false;
        System.out.println("Anda lulus?"+lulus);
    }
}
```

Percobaan 16:

Percobaan berikut ini menunjukkan pemakaian *ternary operator*

```
public class Ternary {

    public static void main(String[] args) {
        System.out.println();
        System.out.println("true ? op2:op3\t\t->"+(true?"op2":"op3"));
        System.out.println("false ? op2:op3\t\t->"+(false?"op2":"op3"));
        System.out.println();

        byte bR;
        short sR;
        byte b0=10;
        short s0=10;
        byte i0=10;
        bR=true? b0:(byte)s0;
        System.out.println("byte=true ? b0:s0     \t\t->"+s0);
        bR=true? b0:i0;
        System.out.println("byte=true ? b0:i0     \t\t->"+bR);
        sR=true? s0:1000;
        System.out.println("short=true ? s0:1000  \t\t->"+sR);
        sR=false? s0:1000;
        System.out.println("short=false ? s0:1000 \t\t->"+sR);
    }
}
```

Percobaan 17:

Percobaan berikut ini menunjukkan pemakaian *bitwise operator* dalam contoh pengunaan variable *flag*. Kemudian gambarkan operasi bit yang terjadi pada tiap barisnya. (Perubahan variabel *flag* pada tiap baris)

```
public class BitwiseDemo {

    static final int VISIBLE =1;
    static final int DRAGGABLE =2;
    static final int SELECTABLE =4;
    static final int EDITABLE =8;

    public static void main(String[] args) {
        int flags =0;

        flags = flags | VISIBLE;
        flags = flags | DRAGGABLE;

        if ((flags & VISIBLE)==VISIBLE){
            if((flags & DRAGGABLE)==DRAGGABLE){
                System.out.println("Flags are Visible and Draggable");
            }
        }

        flags = flags | EDITABLE;

        if((flags &EDITABLE)==EDITABLE){
            System.out.println("Flags are now also Editable");
        }
    }
}
```

Percobaan 18:

Percobaan berikut ini menunjukkan pemakaian *precedence operator* dan contoh pengunaan tanda kurung "(…)" untuk menunjukkan operator mana yang dieksekusi terlebih dahulu.

```
public class Precedence {

    public static void main(String[] args){
        System.out.println("5+3*2 = \t"+(5+3*2));
        System.out.println("(5+3)*2 = \t"+((5+3)*2));
        int a;
        int b=5;
        int c=1;
        a=b=c;
        System.out.println("5-2+1 = \t"+(5-2+1));
        System.out.println("5-2+1 = \t"+ a );

        int d=10;
        int [] f = new int[2];
        int i=0;
        f[i]= d-- + d*2 - ++d + i++;
        System.out.println("i:"+i+" f[0]"+f[0]+" f[1]:"+f[1]);
    }
}
```

Tugas

1. Menampilkan representasi biner dari bilangan desimal bertipe int

Buatlah suatu program untuk menampilkan susunan bit dari suatu bilangan desimal. Nilai bilangan input yang dimasukkan adalah bertipe **int**.

Contoh tampilan:

```
Masukkan nilai desimal: 13
Susunan bit dari 13 adalah 00000000000000000000000000001101
Masukkan nilai desimal: 612
Susunan bit dari 13 adalah 00000000000000000000001001100100
Masukkan nilai desimal: -1
Susunan bit dari 13 adalah 11111111111111111111111111111111
Masukkan nilai desimal: -13
Susunan bit dari 13 adalah 11111111111111111111111111110011
```

2. Menampilkan representasi biner dari bilangan desimal bertipe byte

Buatlah suatu program seperti diatas untuk menampilkan susunan bit dari suatu bilangan desimal, akan tetapi nilai bilangan input yang dimasukkan adalah bertipe **byte**.

3. Mencari representasi biner dari suatu bilangan

Tuliskan representasi bit dari nilai –19? Jelaskan.

4. Menganalisa pergeseran bit dari operasi shift

Jelaskan apa yang terjadi pada potongan program berikut ini:

```
byte a=-1;
a=(byte) (a >>> 2);
```