Pemrograman Berbasis Obyek

Operator dan Assignment

Oleh Politeknik Elektronika Negeri Surabaya

2017

Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Konten

- Unary operator
- Arithmetic operator
- Shift operator: <<, >>, dan >>>
- Comparison operator
- Bitwise operator: &, ^, dan |.
- Short – Circuit operator
- Conditional operator : ?
- Assignment operator
- Operator lain : new, instanceof
- Urutan pemrosesan

The Unary Operators

- Hanya membutuhkan satu operan.

1. Operator increment dan decrement : ++ dan - -
2. Operator unary plus dan minus : + dan –
3. Operator bitwise inversion : ~
4. Operator boolean complement : !
5. Cast : ()

The Unary Operators: ++ dan - -

Examples of Pre-Modify and Post-Modify with the Increment and Decrement Operators

Initial Value of x	Expression	Final Value of y	Final Value of x
5	y = x++	5	6
5	y = ++x	6	6
5	y = x--	5	4
5	y = --x	4	4

```
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);
        }
}
```

The Unary Operators: + dan -

1. X = -3;
2. Y = +3;
3. Z = -(Y+6);

The Unary Operators

- **The Bitwise Inversion Operator: ~**
  - converting all the 1 bits in a binary value to 0s and all the 0 bits to 1s.

  Example:

  `00001111 → 11110000`

- **The Boolean Complement Operator: *!***
  - inverts the value of a boolean expression.

  Example:

  `!true → false`
  `!false → true`

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

RUMUS INVERS

~a = -a - 1

Mencari nilai biner suatu bilangan negatif

- Cara:
  1. Tulis biner bilangan positifnya
  2. Dikurangi dengan 1
  2. ~(hasil)
- Misal: Bagaimana representasi biner untuk bilangan –5 ?

0000 …. 0000 0101 → 5

1

0000 …. 0000 0100

1111 …. 1111 1011 → -5

Mencari bilangan desimal dari bilangan biner negatif

- Cara:
  1. Lakukan negasi terhadap bilangan biner tersebut
  2. Ditambah dengan 1
- Misal : 1111 …. 1111 1011

1111 …. 1111 1011 → berapa?
0000 …. 0000 0100
1
0000 … 0000 0101 --> 5

The Unary Operators: cast → (type)

- Casting digunakan untuk melakukan konversi tipe secara eksplisit ke dalam type baru yang ada dalam tanda ().
- Akan dilakukan pengecekan tipe terlebih dahulu.
- Contoh:

```
int keliling = (int) (Math.PI * diameter);
```

The Unary Operators: cast → (type)

- Bisa diaplikasikan pada tipe obyek.

1. Vector v = new Vector();
2. v.add(“Hello”);
3. String s = (String) v.get(0);

The Arithmetic Operators

- **The Multiplication and Division Operators: * and /**
  - multiply or divide two integers, the result will be calculated using integer arithmetic in either int or long representation.
  - Issues:
    - Loses precision.

```
int x = 7;
int y = 4;
int result =  x/ y;
```

    - The result will be bigger than the maximum number (overflow)

```
byte x = 64;
byte y = 4;
byte result = x*y;
```

The Modulo Operator: %

- Adalah sisa pembagian
- Bisa diaplikasikan pada:
  - Bilangan integer
  - Bilangan floating - point

Example:

```
x = 7 % 4; //so x = 3
y = 7.6 % 2.9; //so y = 1.8

+ % + = +
- % - = -
+ % - = +
- % + = -
```

The Addition and Subtraction Operators: + and -

- Digunakan untuk melakukan operasi penambahan dan pengurangan.
- *Concatenation* → + → bisa juga digunakan untuk menggabungkan 2 string

Arithmetic Error Conditions

- Integer division by zero (ArithmeticException)
- Floating-point calculations represent out-of-range values using the IEEE 754 infinity, minus infinity, and Not a Number (NaN) values.
- Overflow

The Shift Operators:

- **Shift operator:**
  - << : left shift
  - >> : sign right shift
  - >>> : unsigned right shift
- **Fundamentals of Shifting**
  - moving the bit pattern left or right.
  - applied to arguments of integral types only.
- **Pada operator << dan >>>: Nilai bit yang baru adalah 0**
- **Pada operator >> : Nilai bit yang baru tergantung pada bit pada posisi terkiri yang akan digeser, jika nilainya :**
  - **1 → negatif, maka nilai baru adalah 1**
  - **0 → positif, maka nilai baru adalah 0**

The basic mechanisms of shifting

Original data			192			
in binary		00000000	00000000	00000000	11000000	
Shifted left 1 bit	0	00000000	00000000	00000001	1000000?	
Shifted right 1 bit		?0000000	00000000	00000000	01100000	0
Shifted left 4 bits	0000	00000000	00000000	00001100	0000????	
Original data			–192			
in binary		11111111	11111111	11111111	01000000	
Shifted left 1 bit	1	11111111	11111111	11111110	1000000?	
Shifted right 1 bit		?1111111	11111111	11111111	00100000	0

Operator >>

Signed right shift of positive and negative numbers

Original data	192			
in binary	00000000	00000000	00000000	11000000
Shifted right 1 bit	00000000	00000000	00000000	01100000
Shifted right 7 bits	00000000	00000000	00000000	00000001
Original data	–192			
in binary	11111111	11111111	11111111	01000000
Shifted right 1 bit	11111111	11111111	11111111	10100000
Shifted right 7 bits	11111111	11111111	11111111	11111110

Shifting positive and negative numbers right

Original data		**192**		
in binary	00000000	00000000	00000000	11000000
Shifted right 1 bit = 96 = 192 / 2	00000000	00000000	00000000	01100000
Shifted right 4 bits = 12 = 192 / 16 = 192 / 2^4	00000000	00000000	00000000	00001100
Original data		**–192**		
in binary	11111111	11111111	11111111	01000000
Shifted right 1 bit = –96 = –192 / 2	11111111	11111111	11111111	10100000
Shifted right 4 bits = –12 = –192 / 16 = –192 / 2^4	11111111	11111111	11111111	11110100

Operator >>>

Unsigned right shift of a byte

Calculation for –64 >>> 4.

Original data (–64 decimal)			11000000	
Promote to `int` gives:	11111111	11111111	11111111	11000000
Shift right unsigned 4 bits gives:	00001111	11111111	11111111	11111100
Truncate to byte gives:				11111100
Expected result was:				00001100

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

Hasil eksekusi :

```
1
```

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

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

Hasil eksekusi :

```
3
```

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

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

Hasil eksekusi :

```
12
```

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

The Comparison Operators

- Menghasilkan boolean result.
- **Yang termasuk comparison operator:**
  - **Ordinal comparison: <, <=, >, >=**
  - **The *instanceof* Operator**

    Tests the class of an object at runtime.
  - **The Equality Comparison Operators: == and *!=***

    Test for equality and inequality, respectively, returning a boolean value.

Ordinal comparison

```
int p = 9;
int q = 65;
int r = 12;
float f = 9.0f;
char c = 'A';

Berikut ini akan menghasilkan true:
p < q
f < q
f <= c
c > r
c >= q
```

Operator instanceof

- Operator instance of digunakan untuk mengecek class suatu obyek.
- Pengecekan dilakukan pada saat runtime.

```
import java.awt.*;
class CompareTest {
  public static void main(String [] args) {
       Button b = new Button(“Exit”);
       boolean compare1 = b instanceof Button;
       boolean compare2 = b instanceof Component;
       System.out.println(“Is b a Button?” + compare1)
       System.out.println(“Is b a Component?” + compare2)
  }
}
```

Operator instanceof

- Hasil:

```
Is b a Button? true

Is b a Component? true
```

- Argumen sebelah kiri adalah object reference expression.
- Argumen sebelah kanan adalah class, interface, atau array

Equality operators

- Equality can be tested with the operators equals and not equals:
  == → equals
  != → not equals
- There are four different types of entities that can be tested:
  - Numbers
  - Characters
  - Boolean primitives
  - Reference variables to object

Equality for Primitives

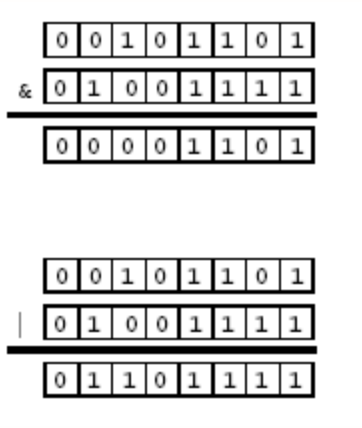
```
class ComparePrimitives{
  public static void main(String [] args) {
    System.out.println('a' =='a');
    System.out.println('a' =='b');
    System.out.println(5 != 6);
    System.out.println(5.0 == 5L);
    System.out.println(true==false);
  }
}
```

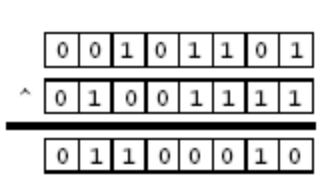
Equality for Reference Variables

```
import java.awt.Button;
class CompareReference {
   public static void main(String [] args) {
         Button a = new Button(“Exit”);
         Button b = new Button(“Exit”);
         Button c = a;
         System.out.println(a==b);
         System.out.println(a==c);
   }
}
```

The Bitwise Operators: &, |, and ^

- Provide logical AND, OR and XOR operations on integral data types.





```
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
```

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

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

Hasil eksekusi :

```
Hasil operasi | = 13
```

```
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
```

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

Hasil operasi ^ = 3

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

Binary Operators: &, |, and ^

- AND, OR and XOR operations on logical data types.
- Semua operan akan dieksekusi.
- Operator &
    - True & True = True
    - True & False = False
    - False & True = False
    - False & False = False
- Operator |
    - True & True = True
    - True & False = True
    - False & True = True
    - False & False = False
- Operator ^
    - True & True = False
    - True & False = True
    - False & True = True
    - False & False = False

The Short-Circuit Logical Operators

- Operators && and ||
- Applicable only to boolean values and not integral types.
- For an AND operation, if one operand is false, the result is false, without regard to the other operand.
- For an OR operation, if one operand is true, the result is true, without regard to the other operand.

•Jadi, untuk nilai boolean x:

- false && X = false
- true || X = true

```
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
```

```
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
```

The Conditional Operator: *?:*

- known as the *ternary* operator
- takes three operands
- code simple conditions (if/else) into a single expression.

- Example:

```
a = x ? b : c;
```

- Aturan:
  - Tipe data b, c dan a sebaiknya sama. Jika tidak sama? Terjadi promosi
  - Tipe ekspresi x harus boolean
  - Contoh nilai x → (6>7)
  - Jika ekspresi x benar maka akan menghasilkan b
  - Jika ekspresi x salah maka akan menghasilkan c

```
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);
    }
}
```

The Assignment Operators

- set the value of a variable or expression to a new value.
- Example:

```
1. byte x = 2;
2. x += 3;
3. a = b = c = 0; //legal.
```

Operators Precedence

Operators in Java, in Descending Order of Precedence

Category	Operators
Unary	++ -- + - ! ~ ()
Arithmetic	* / %
	+ -
Shift	<< >> >>>
Comparison	< <= > >= instanceof
	== !=

Operators Precedence (cont.)

Operators in Java, in Descending Order of Precedence *(continued)*

Category	Operators
Bitwise	& ^ |
Short-circuit	&& ||
Conditional	?:
Assignment	= "op="

Evaluation Order

```
1. int [] a = { 4, 4 };
2. int b = 1;
3. a[b] = b = 0;
```

Note: untuk assignment berlaku aturan asosiatif → dari kanan ke kiri.

1. a[b] → a[1]
2. b = 0
3. a[1] = 0

Tugas

- Buatlah makalah yang berisi tentang berbagai macam operator dengan disertai contoh penggunaan dan outputnya

1. Oracle Java Documentation, The Java™ Tutorials, https://docs.oracle.com/javase/tutorial/, Copyright © 1995, Oracle 2015.
2. Tita Karlita, Yuliana Setrowati, Rizky Yuniar Hakkun, Pemrograman Berorientasi Obyek, PENS-2012
3. Sun Java Programming, Sun Educational Services, Student Guide, Sun Microsystems, 2001.
4. John R. Hubbard, Programming With Java, McGraw-Hill, ISBN: 0-07-142040-1, 2004.
5. Patrick Niemeyer, Jonathan Knudsen, Learning Java, O'reilly, CA, ISBN: 1565927184, 2000.
6. Philip Heller, Simon Roberts, Complete Java 2 Certification Study Guide, Third Edition, Sybex, San Francisco, London, ISBN: 0-7821-4419-5, 2002.
7. Herbert Schildt, The Complete Reference, Java™ Seventh Edition, Mc Graw Hill, Osborne, ISBN: 978-0-07-163177-8, 2007

bridge to the future