

# JAVA Ch. 3

Prof. Youngchul Jung

2012 – Winter class



주문식교육의 신실  
영진전문대학

# 연산자 (Operator)

## 연산자(Operator)란?

### ▶ 연산자(Operator)

- 어떠한 기능을 수행하는 기호(+, -, \*, / 등)

### ▶ 피연산자(Operand)

- 연산자의 작업 대상(변수, 상수, 리터럴, 수식)

`int` a = 2 + 3

# 연산자의 종류

▶ 단항 연산자 : + - (타입) ++ -- ~ !

산술 : + - \* / % << >> >>>

▶ 이항 연산자 비교 : > < >= <= == !=

논리 : && || & ^ |

▶ 삼항 연산자 : ? :

▶ 대입 연산자 : =

# 연산자 우선순위 (1)

종 류	연산방향	연 산 자	우 선 순 위
단항 연산자	←	++ -- + - ~ ! (타입)	높음
산술 연산자	→	* / %	
	→	+ -	
	→	<< >> >>>	
비교 연산자	→	< > <= >= instanceof	
	→	== !=	
논리 연산자	→	&	
	→	^	
	→		
	→	&&	
	→		
삼항 연산자	→	?:	낮음
대입 연산자	←	= *= /= %= += -= <<= >>= >>>= &= ^=  =	

[표3-1] 연산자의 종류와 우선순위

## 연산자 우선순위 (2)

- 괄호의 우선순위가 제일 높다.
- 산술 > 비교 > 논리 > 대입
- 단항 > 이항 > 삼항
- 연산자의 연산 진행방향은 왼쪽에서 오른쪽( $\rightarrow$ )이다.
  - 단, 단항, 대입 연산자만 오른쪽에서 왼쪽( $\leftarrow$ )이다.

$$3 * 4 * 5$$

$$x = y = 3$$



## 연산자 우선순위 (3)

- 상식적으로 생각하라. 우리는 이미 다 알고 있다!!

ex1)  $-x + 3$

단항 > 이항

ex2)  $x + 3 * y$

곱셈, 나눗셈 > 덧셈, 뺄셈

ex3)  $x + 3 > y - 2$

산술 > 비교

ex4)  $x > 3 \&\& x < 5$

비교 > 논리

ex5) `int result = x + y * 3;`

항상 대입은 맨 끝에

## 연산자 우선순위 (4)

- 그러나 몇 가지 주의해야 할 것이 있다
- $<<, >>, >>>$ 는 덧셈연산자보다 우선순위가 낮다

ex5)  $x << 2 + 1$        $x << (2 + 1)$  과 같다.

- $\parallel, |$  (OR)는  $\&\&, \&$  (AND)보다 우선순위가 낮다.

ex6)  $x < -1 \parallel x > 3 \&\& x < 5$

$x < -1 \parallel (x > 3 \&\& x < 5)$  와 같다.

# 단항연산자



## 증감연산자: **--**, **++**

- ▶ **증가연산자(++)** : 피연산자의 값을 1 증가시킨다.
- ▶ **감소연산자(--)** : 피연산자의 값을 1 감소시킨다.

```
int i = 5;
```

```
int j = 0;
```

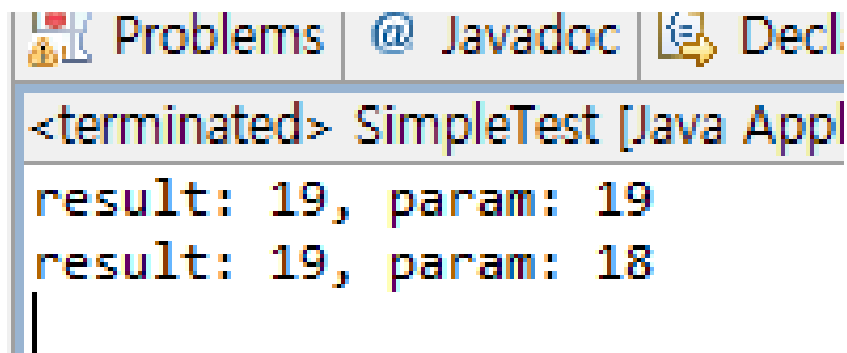
전위형	<code>j = ++i;</code>	<code>++i;</code> <code>j = i;</code>	값이 참조되기 전에 증가시킨다.
후위형	<code>j = i++;</code>	<code>j = i;</code> <code>i++;</code>	값이 참조된 후에 증가시킨다.

# 증감연산자 예제 (1)

```
public class SimpleTest {  
    public static void main(String[] args) {  
        int varTest = 10;  
  
        varTest++;  
        System.out.println(varTest );  
  
        ++varTest;  
        System.out.println(varTest );  
  
        System.out.println(varTest++ );  
        System.out.println(++varTest );  
    }  
}
```

## 증감연산자 예제 (2)

```
public class SimpleTest {  
    public static void main(String[] args) {  
        int param    = 20;  
        int result    = 0;  
  
        result = -- param;  
        System.out.println("result: " + result + ", param: " + param);  
  
        result = param--;  
        System.out.println("result: " + result + ", param: " + param );  
    }  
}
```



```
<terminated> SimpleTest [Java Appl  
result: 19, param: 19  
result: 19, param: 18  
|
```

# 부호연산자(+,-)와 논리부정연산자(!)

- 부호연산자(+,-)
  - '+'는 피연산자에 1을 곱하고
  - '-'는 피연산자에 -1을 곱한다.
- 논리부정연산자(!)
  - true는 false로, false는 true로
  - 피연산자가 boolean일 때만 사용가능

`int i = -10;`

`boolean power = false;`

`i = +i;`

`power = !power;`

`i = -i;`

`power = !power;`

# 부호연산자 예제

```
public class SimpleTest {  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
  
        // 키보드로부터 정수 입력  
        int inputValue = scan.nextInt();  
  
        System.out.println("입력값은 " + inputValue);  
  
        // 음수 입력시 양수 변환  
        if( inputValue < 0 )  
        {  
            inputValue = -inputValue;  
  
            System.out.println("음수입니다. 변환  
        }  
    }  
}
```

Problems @ Javadoc C  
<terminated> SimpleTest [Java A  
-5  
입력값은 -5  
음수입니다. 변환된 값은 : 5

## 비트전환연산자: ~

- 정수를 2진수로 표현했을 때, 1을 0으로 0은 1로 바꾼다.
  - 정수형, Char 형에만 사용가능.

2진수	10진수								
<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr></table>	0	0	0	0	1	0	1	0	10
0	0	0	0	1	0	1	0		
<table><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	1	1	1	1	0	1	0	1	-11
1	1	1	1	0	1	0	1		
<table><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	1	1	1	1	0	1	0	1	-11
1	1	1	1	0	1	0	1		
<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>	0	0	0	0	0	0	0	1	+) 1
0	0	0	0	0	0	0	1		
<table><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	1	1	1	1	0	1	1	0	-10
1	1	1	1	0	1	1	0		

[표3-5] 음수를 2진수로 표현하는 방법



## ~ 연산자를 이용한 ??? 구현 ^\_^

```
public class SimpleTest {  
    public static void main(String[] args) {  
        int firstValue    = 10;  
        int secondValue   = 5;  
  
        secondValue = ~secondValue;  
  
        secondValue++;  
  
        System.out.println(firstValue + secondValue);  
    }  
}
```

# 이항연산자

# 이항연산자의 특징(1)

- 이항연산자는 연산을 수행하기 전에 피연산자의 타입을 일치시킨다!!
- int보다 크기가 작은 타입은 int로 변환한다.
  - ( byte, char, short → int )
- 피연산자 중 표현범위가 큰 타입으로 형변환 한다.

byte + short → int + int → int

char + int → int + int → int

float + int → float + float → float

long + float → float + float → float

float + double → double + double → double

## 이항연산자의 특징(2)

byte a = 10;

byte + byte → int + int → int

byte b = 20;

byte c = a + b;

```
public class Simpletest {
```

```
    public static void main(String[] args) {
```

```
        byte a = 10;
```

```
        byte b = 20;
```

```
        byte c = a + b;
```

```
        System.out.println(c);
```

```
    }
```

```
}
```

byte c = (byte)a + b;

byte c = (byte)(a + b);

# 이항연산자 예제(1)

```
public class SimpleTest {  
    public static void main(String[] args) {  
        byte firstValue    = 10;  
        byte secondValue   = 5;  
        byte result         = 0;  
  
        result = firstValue + secondValue;  
  
        System.out.println(result);  
    }  
}
```

## 이항연산자의 특징(3)

```
int a = 1000000; // 1,000,000
```

```
int b = 2000000; // 2,000,000
```

```
long c = a * b; // c는 2,000,000,000,000 ?
```

```
<terminated> SimpleTest [Java Ap  
-1454759936
```

**int \* int → int**

```
long c = (long)a * b; // c는 2,000,000,000,000
```

**long \* int → long \* long → long**



## 이항연산자의 특징(4)

**long** a = 1000000 \* 1000000;      // a는 -727,379,968

**long** b = 1000000 \* 1000000L;      // b는 1,000,000,000,000

**int** c = 1000000 \* 1000000 / 1000000;      // c는 -727

**int** d = 1000000 / 1000000 \* 1000000;      // d는 1,000,000

# 이항연산자의 특징(5)

```
public class SimpleTest {  
  
    public static void main(String[] args) {  
  
        char c1 = 'a';  
        char c2 = c1 + 1;  
        char c3 = ++c1;  
  
        int i = 'B' - 'A';  
        int j = '2' - '0';  
  
        System.out.println(c1);  
        System.out.println(c2);  
        System.out.println(c3);  
        System.out.println(i);  
        System.out.println(j);  
    }  
}
```

문자	코드
...	...
0	48
1	49
2	50
...	...
A	65
B	66
C	67
...	...
a	97
b	98
c	99
...	...

## 나머지 연산자: %

- 나누기한 나머지를 반환한다.
- 홀수, 짝수 등 배수검사에 주로 사용.

```
int share = 10 / 8;
```

```
int remain = 10 % 8;
```

$10 \% 8 \rightarrow 2$

$10 \% -8 \rightarrow 2$

$-10 \% 8 \rightarrow -2$

$-10 \% -8 \rightarrow -2$

# 나머지 연산자 예제

```
public class SimpleTest {  
    public static void main(String[] args) {
```

```
(int count = 1 ; count <= 20 ; count++)  
    if(count%5 == 0)  
        System.out.println("5의 배수: " + count);  
  
    if(count%2 == 0)  
        System.out.println(count + ": 짝수");  
    else  
        System.out.println(count + ": 홀수");  
}
```

Problems @ Ja  
<terminated> Simpl  
1: 홀수  
2: 짝수  
3: 홀수  
4: 짝수  
5의 배수: 5  
5: 홀수  
6: 짝수  
7: 홀수  
8: 짝수  
9: 홀수  
5의 배수: 10

## 쉬프트연산자: <<, >>, >>>

- $2^n$ 으로 곱하거나 나눈 결과를 반환한다.
- 곱셈, 나눗셈보다 빠르다

$x \ll n$  은  $x * 2^n$ 과 같다.

$x \gg n$  은  $x / 2^n$ 과 같다.

$8 \ll 2$  는  $8 * 2^2$ 과 같다.

$8 \gg 2$  는  $8 / 2^2$ 과 같다.

$1 \ll 0$

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

$1 \ll 1$

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

$1 \ll 2$

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

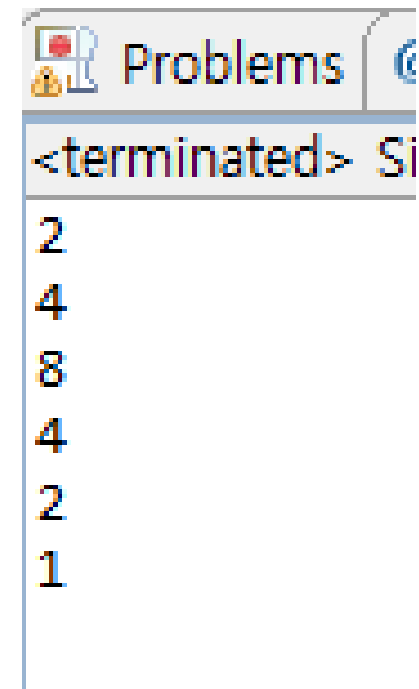
# 쉬프트연산 예제 (1)

```
public class SimpleTest {  
  
    public static void main(String[] args) {  
  
        System.out.println("----- << 연산 -----");  
        System.out.println( (1<<0));  
        System.out.println( (1<<1));  
        System.out.println( (1<<2));  
  
        System.out.println("----- >> 연산 -----");  
        System.out.println( (8>>1));  
        System.out.println( (8>>2));  
        System.out.println( (8>>3));  
  
    }  
}
```



## 쉬프트연산 예제 (2)

```
public class SimpleTest {  
    public static void main(String[] args) {  
        int i = 1;  
  
        // << 연산자  
        i = i << 1;  
        System.out.println(i);  
  
        i = i << 1;  
        System.out.println(i);  
  
        i = i << 1;  
        System.out.println(i);  
  
        // >> 연산자  
        i = i >> 1;  
        System.out.println(i);  
  
        i = i >> 1;  
        System.out.println(i);  
  
        i = i >> 1;  
        System.out.println(i);  
    }  
}
```



## 비교연산자: > < >= <= == !=

- 피연산자를 같은 타입으로 변환한 후에 비교한다.
  - 결과 값은 true 또는 false이다.
- 기본형(boolean제외)과 참조형에 사용할 수 있으나
  - 참조형에는 ==와 !=만 사용할 수 있다.

수 식	연 산 결 과
$x > y$	x가 y보다 클 때 true, 그 외에는 false
$x < y$	x가 y보다 작을 때 true, 그 외에는 false
$x \geq y$	x가 y보다 크거나 같을 때 true, 그 외에는 false
$x \leq y$	x가 y보다 작거나 같을 때 true, 그 외에는 false
$x == y$	x와 y가 같을 때 true, 그 외에는 false
$x != y$	x와 y가 다를 때 true, 그 외에는 false

[표3-11] 비교연산자의 연산결과

# 비교연산자: > < >= <= == !=

'A' < 'B'

→ 65 < 66

→ true

'0' == 0

→ 48 == 0

→ false

'A' != 65

→ 65 != 65

→ false

10.0d == 10.0f

→ 10.0d == 10.0d → true

0.1d == 0.1f

→ 0.1d == 0.1d

→ false

double d = (double)0.1f;

System.out.println(d); // 0.10000000149011612

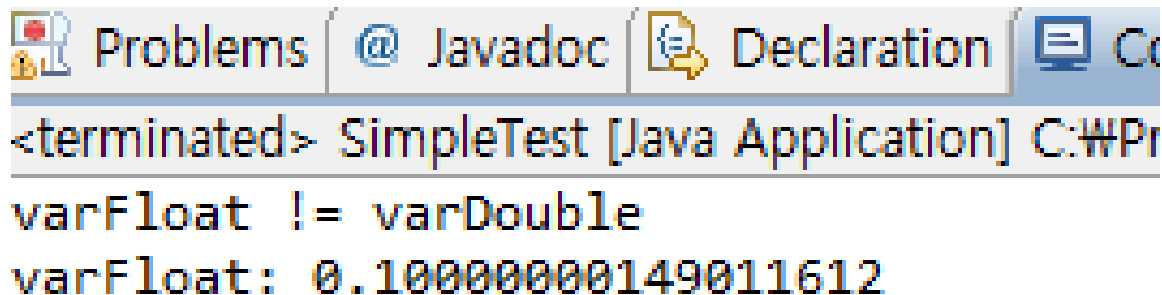
(float)0.1d == 0.1f

→ 0.1f == 0.1f → true

문자	코드
...	...
0	48
1	49
2	50
...	...
A	65
B	66
C	67
...	...
a	97
b	98
c	99
...	...

# 비교연산자 예제

```
public class SimpleTest {  
    public static void main(String[] args) {  
        float    varFloat    = 0.1f;  
        double   varDouble   = 0.1d;  
  
        if(varFloat == varDouble)  
            System.out.println("varFloat == varDouble");  
        else  
        {  
            System.out.println("varFloat != varDouble" + "\nvarFloat: " + (double)varFloat);  
        }  
    }  
}
```



Problems | @ Javadoc | Declaration | Console

<terminated> SimpleTest [Java Application] C:\#Pr  
varFloat != varDouble  
varFloat: 0.10000000149011612

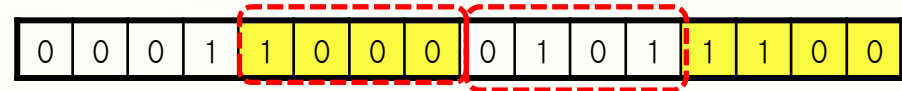
# 비트연산자: & | ^

- 피연산자를 비트단위로 연산한다.
- 실수형(float, double)을 제외한 모든 기본형에 사용가능
  - OR연산자 ( | ) : 피연산자 중 어느 한 쪽이 1이면 1이다.
  - AND연산자 ( & ) : 피연산자 양 쪽 모두 1이면 1이다.
  - XOR연산자 ( ^ ) : 피연산자가 서로 다를 때 1이다.

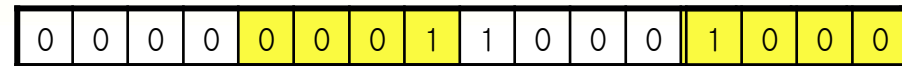
x	y	x   y	x & y	x ^ y
1	1	1	1	0
1	0	1	0	1
0	1	1	0	1
0	0	0	0	0

# 비트연산자: & | ^

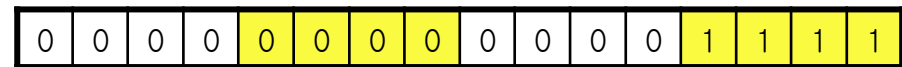
0x185C



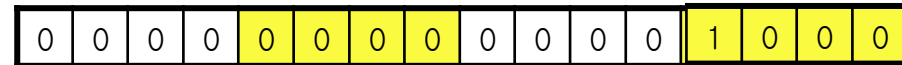
0x185C >> 4 → 0x0185



0x000F



0x0185 & 0x000F → 0x0005



**0x185C >> 4 & 0x000F → 0x0005**

**0x185C >> 8 & 0x000F → 0x0008**



# 비트연산자 예제

```
public class SimpleTest {  
  
    public static void main(String[] args) {  
  
        // 2 byte - 1011 0111 0000 0000  
        int status = 0xB700;  
        int mask = 0x00ff;  
  
        status = status >> 12;  
  
        System.out.println(status & mask);  
    }  
}
```

## 논리연산자: && ||

- 피연산자가 반드시 boolean이어야 하며 연산결과도 boolean이다.
  - &&가 || 보다 우선순위가 높다. 같이 사용되는 경우 괄호를 사용하자
- ▶ OR 연산자 ( || ) : 피연산자 중 어느 한 쪽이 true이면 true이다.
- ▶ AND 연산자 (&&) : 피연산자 양 쪽 모두 true이면 true이다.

x	y	x    y	x && y
true	true	true	true
true	false	true	false
false	true	true	false
false	false	false	false

# 논리연산자: && ||

```
int i = 7;
```

```
i > 3 && i < 5
```

```
i > 3 || i < 0
```

```
char x = 'j';
```

```
x >= 'a' && x <= 'z'
```

```
(x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z')
```

# 삼항연산자, 대입연산자

## 삼항연산자: ? :

- 조건식의 연산결과가 true이면 '식1'의 결과를 반환하고 false이면 '식2'의 결과를 반환한다.

(조건식) ? 식1 : 식2

```
int x = -10;
```

```
int absX = x >= 0 ? x : -x;
```

```
int score = 50;
```

```
char grade = score >= 90 ? 'A' : (score >= 80 ? 'B' : 'C');
```

```
if(x >= 0) {  
    absX = x;  
} else {  
    absX = -x;  
}
```

# 삼항연산자 예제

```
public class SimpleTest {  
    public static void main(String[] args) {  
        Random rand = new Random();  
        String strResult = "";  
        int selectedValue = rand.nextInt() % 10;  
        strResult = selectedValue % 2 == 0 ? "짝수" : "홀수";  
        System.out.println("selectedValue: " + selectedValue + ", " + strResult);  
    }  
}
```

## 대입연산자: = op=

- 오른쪽 피연산자의 값을 왼쪽 피연산자에 저장한다.
- 단, 왼쪽 피연산자는 상수가 아니어야 한다.

```
int i = 0;
```

```
i = i + 3;
```

```
final int MAX = 3;
```

```
MAX = 10; // 에러
```

op=	=
i += 3;	i = i + 3;
i -= 3;	i = i - 3;
i *= 3;	i = i * 3;
i /= 3;	i = i / 3;
i %= 3;	i = i % 3;
i <<= 3;	i = i << 3;
i >>= 3;	i = i >> 3;
i >>>= 3;	i = i >>> 3;
i &= 3;	i = i & 3;
i ^= 3;	i = i ^ 3;
i  = 3;	i = i   3;
i *= 10 + j;	i = i * (10+j);

대학