

Java Web Programming 입문 03

(Java #03)

오늘의 키워드

- ▶ 주석 (comment)
 - Single line / Multi line
- ▶ 연산자 (operator)
 - 단항 연산자
 - 산술 연산자
 - 비교 연산자
 - 논리 연산자
 - 삼항 연산자
 - 대입 연산자
- ▶ 제어문 (control statement)
 - 조건문(conditional statement)/분기문 (branch statement)
 - if
 - switch ~ case
 - 반복문 (repetitive / iterative / loop statement)
 - for
 - while
 - do ~ while
 - break
 - continue



주석 (comment)

- ▶ No Compile

- ▶ Programmer

- ▶ Communication

- ▶ Explain

- ▶ Single line / Multi line



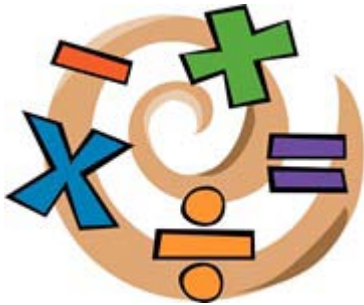
```
/*#####  
## 작성일 : 2014-01-12 ##  
## 작성자 : 홍길동 선임 ##  
## 연락처 : XX) XXX - XXXX (XX사 XX팀) ##  
## 작성내용 : 주석처리 예제 ##  
#####*/
```

```
// Single line comment1
```

```
/* Multi  
line  
comment  
*/
```

연산자 (operator)

- ▶ 데이터의 가공
- ▶ 데이터 타입에 따라



- ▶ 단항 연산자
 - 증감 연산자
 - ++, --
 - 부호 연산자
 - +, -
 - 논리부정 연산자
 - !
 - 비트 전환 연산자
 - ~
- ▶ 산술 연산자
 - 오칙 연산자
 - +, -, *, /, %
 - 쉬프트 연산자
 - <<, >>, >>>
- ▶ 비교 연산자
 - 대소비교 연산자
 - <, >, <=, >=
 - 등가비교 연산자
 - ==, !=
 - 비트 연산자
 - &, |, ^
- ▶ 논리 연산자
 - &&, ||
- ▶ 삼항 연산자
 - ?:
- ▶ 대입 연산자
 - =, +=, -=
 - *=, /=, %=, <<=, >>=, >>>=, &=, ^=, |=

연산자 (operator)

▶ 단항 연산자

◦ 증감 연산자 (중요)

- ++
- --

◦ 부호 연산자

- +
- -

◦ 논리부정 연산자 (중요)

- !

◦ 비트 전환 연산자 (pass)

- ~

①
y = x++;
②

The diagram shows the expression y = x++; An orange arrow labeled ① points from the x to the y, indicating the value of x is used for the assignment. Another orange arrow labeled ② points from the ++ to the x, indicating the increment operation is performed on x after the assignment.

②
y = ++x;
①

The diagram shows the expression y = ++x; An orange arrow labeled ② points from the ++ to the y, indicating the increment operation is performed on x first. Another orange arrow labeled ① points from the x to the y, indicating the incremented value of x is used for the assignment.

□ TRUE
□ FALSE

연산자 (operator)

증감 연산자

연산자	사용	설명
++	++ A	A의 값을 1증가시킨 후, A를 처리
	A ++	A를 처리한 후 A의 값을 1 증가시킴
--	-- A	A의 값을 1 감소시킨 후, A를 처리
	A --	A를 처리한 후, A의 값을 1 감소시킴

```
public class IncreaseDecrease{  
  
    public static void main(String[] args){  
  
        int a = 10;  
        System.out.println("현재 a의 값 : " + a);  
        System.out.println("[++a] 출력 : " + (++a));  
        System.out.println("현재 a의 값 : " + a);  
        System.out.println("[a++] 출력 : " + (a++));  
        System.out.println("현재 a의 값 : " + a);  
        System.out.println("[--a] 출력 : " + (--a));  
        System.out.println("현재 a의 값 : " + a);  
        System.out.println("[a--] 출력 : " + (a--));  
        System.out.println("현재 a의 값 : " + a);  
    }  
}
```

연산자 (operator)

▶ 산술 연산자

◦ 오직 연산자 (중요)

- +
- -
- *
- /
- %

◦ 쉬프트 연산자 (pass)

- <<
- >>
- >>>



연산자 (operator)

▶ 산술 연산자

연산자	사용	설명
+	A + B	A값과 B 값을 더한다.
-	A - B	A값에 B값을 뺀다.
*	A * B	A값과 B값을 곱한다.
/	A / B	A값에 B값을 나눈다.
%	A % B	A값을 B값으로 나눈 나머지를 구한다.

```
public class Arithmetic{  
  
    public static void main(String[] args){  
  
        int a = 10;  
        int b = 3;  
  
        System.out.println("a의 값 : " + a);  
        System.out.println("b의 값 : " + b);  
  
        System.out.println(a + " + " + b + " = " + (a + b));  
        System.out.println(a + " - " + b + " = " + (a - b));  
        System.out.println(a + " * " + b + " = " + (a * b));  
        System.out.println(a + " / " + b + " = " + (a / b));  
        System.out.println(a + " % " + b + " = " + (a % b));  
  
    }  
}
```


연산자 (operator)

▶ 비교 연산자

◦ 대소비교 연산자 (중요)

- $<$
- $>$
- $<=$
- $>=$

◦ 등가비교 연산자 (중요)

- $==$
- $!=$

◦ 비트 연산자 (pass)

- $\&$
- $|$
- \wedge



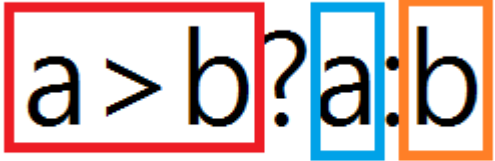
연산자 (operator)

▶ 비교 연산자

연산자	사용	설명
==	A == B	A의 값과 B의 값이 같은 경우 true, 다를 경우 false
!=	A != B	A의 값과 B의 값이 다를 경우 true, 같을 경우 false
>	A > B	A의 값이 B의 값보다 클 경우 true, 그렇지 않을 경우 false
>=	A >= B	A의 값이 B의 값보다 크거나 같으면 true, 그렇지 않으면 false
<	A < B	A의 값이 B의 값보다 작을 경우 true, 그렇지 않을 경우 false
<=	A <= B	A의 값이 B의 값보다 작거나 같으면 true, 그렇지 않으면 false

```
public class Compare{  
  
    public static void main(String[] args){  
  
        int a = 10;  
        int b = 3;  
        System.out.println("a : " + a + ", b : " + b);  
        System.out.println("a > b : " + (a > b));  
        System.out.println("a >= b : " + (a >= b));  
        System.out.println("a < b : " + (a < b));  
        System.out.println("a <= b : " + (a <= b));  
        System.out.println("a == b : " + (a == b));  
        System.out.println("a != b : " + (a != b));  
    }  
}
```

연산자 (operator)


비교문 TRUE FALSE

▶ 논리 연산자 (중요)

- &&
- ||

▶ 삼항 연산자 (좀 할 줄 아는 것처럼 보인다)

- (조건식) ? 식1 : 식2
- `result = ((x > 0) ? x : -x);`
- 행동 = ((니가 쓸래) ? 지갑을꺼낸다 : 화장실에간다);
- if 문으로 바꿔 쓸 수 있다.

▶ 대입 연산자 (중요)

- = (항상 쓴다)
- +=, -=, *=, /=, %= (꽤 쓴다)
- <<=, >>=, >>>=, &=, ^=, |= (pass)

연산자 (operator)

▶ 논리 연산자

연산자	사용	설명
&&	A && B	A의 값과 B의 값이 모두 true 일 경우 true, 그 외에는 false 반환
	A B	A의 값과 B의 값이 모두 false 일 경우 false, 그 외에는 true 반환
!	! A	A의 값이 true일 경우 false, false일 경우 true 반환

&& 연산자	true	false	연산자	true	false
true	true	false	true	true	true
false	false	false	false	true	false

```
public class Logical{  
  
    public static void main(String[] args){  
  
        int num1 = 10;  
        int num2 = 5;  
        System.out.println("num1 : " + num1 + ", num2 : " + num2);  
  
        boolean bool1 = num1 > num2;  
        boolean bool2 = num1 < num2;  
        boolean bool3 = num1 == num2;  
        boolean bool4 = num1 != num2;  
        System.out.println("bool1 : " + bool1);  
        System.out.println("bool2 : " + bool2);  
        System.out.println("bool3 : " + bool3);  
        System.out.println("bool4 : " + bool4);  
  
        System.out.println("bool1 && bool2 : " + (bool1 && bool2));  
        System.out.println("bool1 || bool2 : " + (bool1 || bool2));  
        System.out.println("bool1 && bool4 : " + (bool1 && bool4));  
        System.out.println("bool2 || bool3 : " + (bool2 || bool3));  
  
    }  
}
```

연산자 (operator)

▶ 대입 연산자

연산자	사용	동일 표현	설명
=	A = B		A에 B의 값을 대입
+=	A += B	A = A + B	A값에 B값을 더한 결과값을 A에 대입
-=	A -= B	A = A - B	A값에 B값을 뺀 결과값을 A에 대입
*=	A *= B	A = A * B	A값에 B값을 곱한 결과값을 A에 대입
/=	A /= B	A = A / B	A값에 B값을 나눈 결과값을 A에 대입
%=	A %= B	A = A % B	A값에 B값을 나눈 나머지 결과값을 A에 대입

```
public class Substitution{  
  
    public static void main(String[] args){  
  
        int a = 10;  
        int b = 3;  
        System.out.println("a : " + a + ", b : " + b);  
        a = b;  
        System.out.println("a = b 수행, " + "현재 a의 값 : " + a);  
        a += b;  
        System.out.println("a += b 수행, " + "현재 a의 값 : " + a);  
        a -= b;  
        System.out.println("a -= b 수행, " + "현재 a의 값 : " + a);  
        a *= b;  
        System.out.println("a *= b 수행, " + "현재 a의 값 : " + a);  
        a /= b;  
        System.out.println("a /= b 수행, " + "현재 a의 값 : " + a);  
        a %= b;  
        System.out.println("a %= b 수행, " + "현재 a의 값 : " + a);  
  
    }  
}
```

연산자 (operator)

우선순위

Prio- rity	Opera- tor	Name	Asso- ciativity	Example
1	++	Increment	r	x++ ++x
	--	Decrement	r	x-- --x
	+	Unary plus	r	+x
	-	Unary minus	r	-x
	!	Logical complement	r	!isOpen
	~	Bitwise complement	r	~i
2	(type)	Cast	r	i = (int) x

2	*	Multiplication	l	x * 2
	/	Division	l	x / 2
	%	Remainder	l	x % 2

3	+	Binary plus	l	x + 2 " " + x + i
	-	Binary minus	l	x - i

4	<<	Shift left	l	i << 2
	>>	Shift right	l	-i >> 2
	>>>	Shift right ignore sign	l	-i >>> 2

5	>	greater than	l	i > x
	<	less than	l	i < x
	>=	greater equal	l	i >= x
	<=	less equal	l	i <= x
	instanceof	Type check	l	s instanceof String

연산자 (operator)

Priority	Operator	Name	Associativity	Example
6	<code>==</code>	Equals	1	<code>i == j</code> <code>s == ""</code>
	<code>!=</code>	Not equal	1	<code>i != j</code> <code>s != null</code>
7	<code>&</code>	Bitwise and	1	<code>i & j</code>
8	<code>^</code>	Exclusive or	1	<code>i ^ 5</code>
9	<code> </code>	Bitwise or	1	<code>i j</code>
10	<code>&&</code>	Logical and	1	<code>isOpen && false</code>
11	<code> </code>	Logical or	1	<code>isOpen false</code>
12	<code>? :</code>	Conditional	r	<code>i < 0 ? -1 : 1</code>

13	<code>=</code>	Assignment	r	<code>j = i</code> <code>o = s;</code>
	<code>+=</code>	Plus assignment	r	<code>j += x</code>
	<code>-=</code>	Minus assignment	r	<code>j -= x</code>
	<code>*=</code>	Multiplication assign.	r	<code>j *= x</code>
	<code>/=</code>	Division assign.	r	<code>j /= x</code>
	<code>&=</code>	Bitwise and assign.	r	<code>j &= i</code>
	<code> =</code>	Bitwise or assign.	r	<code>j = i</code>
	<code>^=</code>	Exclusive or assign.	r	<code>j ^= i</code>
	<code>%=</code>	Remainder assign.	r	<code>j %= i</code>
	<code><<=</code>	Shift left assign.	r	<code>j <<= i</code>
14	<code>>>=</code>	Shift right assign.	r	<code>j >>= i</code>
	<code>>>>=</code>	Shift right i.s. assign.	r	<code>j >>>= i</code>

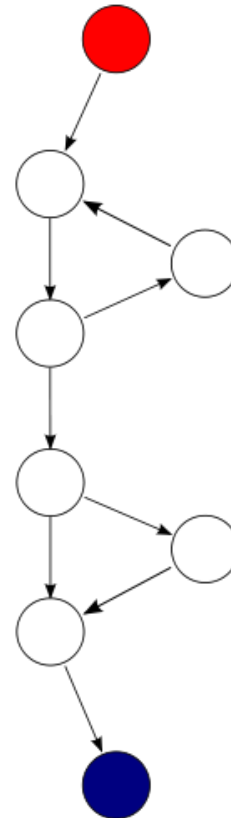
연산자 (operator)

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



제어문 (control statement)

- ▶ Flow Control
- ▶ Not Top-Down
- ▶ Condition
- ▶ TRUE/FALSE
- ▶ Repetition
- ▶ Real Programming



분기문 (branch statement)

▶ if

- popular
- every branch statement



```
before [if]

if ( condition ){
    "statement"
}

after [if]
```

```
before [if]

if ( condition ){
    "statement"
}else{
    "statement"
}

after [if]
```

```
before [if]

if ( condition ){
    "statement"
}else if ( condition1 ){
    "statement"
}else if ( condition 2){
    "statement"
    .
    .
    .
}else{
    "statement"
}

after [if]
```

조건문

▶ if 문

```
if (조건식1) {  
    // 조건식1의 연산결과가 true 일 때 수행될 문장  
}  
else if (조건식2) {  
    // 조건식2의 연산결과가 true 일 때 수행될 문장  
}  
else if (조건식3) {  
    // 조건식3의 연산결과가 true 일 때 수행될 문장  
}  
else {  
    // 위의 어느 조건식도 만족하지 않을 때 수행될 문장  
}
```

조건문

▶ if 문

```
if (조건식1) {  
    // 조건식1의 연산결과가 true 일 때 수행  
  
    if (조건식2) {  
        // 조건식1과 조건식2가 모두 true일 때 수행  
    } else {  
        // 조건식1이 true, 조건식2는 false일 때 수행  
    }  
  
} else {  
    // 위의 어느 조건식도 만족하지 않을 때 수행  
}
```

분기문 (branch statement)

- ▶ switch ~ case
 - One Variable, Various Case
 - Convert If statement



```
before [switch ~ case]

switch ( variable ){
case [case1] :
    "statement1"
    break;
case [case2] :
    "statement2"
    break;
.
.
.
default :
    "statemtntDefault"
    break;
}

after [switch ~ case]
```

조건문

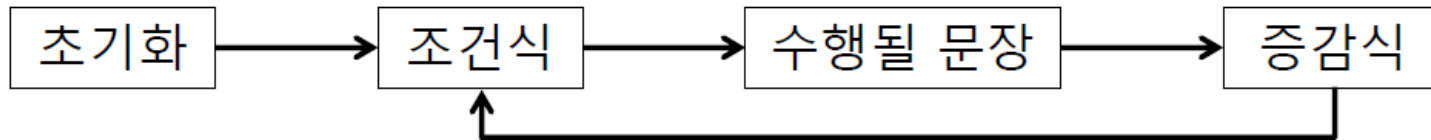
▶ switch 문

```
switch (조건식) {  
  
    case 값1 :  
        // 조건식의 결과가 값1과 같을 경우 수행  
        // ...  
        break;  
  
    case 값2 :  
        // 조건식의 결과가 값2와 같을 경우 수행  
        // ...  
        break;  
  
    // ... (중략)  
  
    default :  
        // 조건식의 결과와 일치하는 case문이 없을 때 수행  
        // ...  
}
```

반복문 (iterative statement)

▶ for

```
for (초기화 ; 조건식 ; 증감식){  
    // 조건식이 true일 때 수행  
}
```



```
for ( 초기화 ; 조건식 ; 증감식){  
    "statement"  
}
```

```
public static void main(String[] args){  
    int sum = 0;  
  
    for ( int i=1 ; i <= 10 ; i++ ){  
        sum += i;    // sum = sum + i  
    }  
    System.out.println( i-1 + " 까지의 합: " + sum);  
}
```



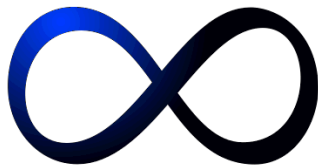
반복문 (iterative statement)

▶ while

- while condition is true...

```
while ( 조건식 ){  
    "statement"  
}
```

- infinite loop warning

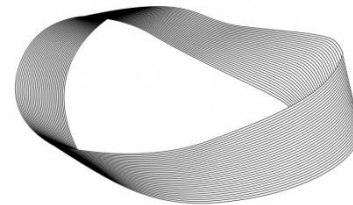


반복문

▶ while 문

```
while (조건식){  
    // 조건식의 연산결과가 true일 때 수행  
}
```

```
int i = 0;  
while ( i >= 0 ) {  
    i = 10;  
    System.out.println(i--);  
}
```



반복문 (iterative statement)

▶ do ~ while

- First, execute!
- Then condition check

```
do{  
    "statement"  
}  
while ( 조건식 );
```

JUST DO IT.

반복문

▶ do-while 문 (잘 안쓴다)

```
do {  
    // 일단 한번 수행 후,  
    // 조건식의 결과가 true일때 수행  
} while ();
```



반복문 (iterative statement)

- ▶ break

- Unconditioned escape

```
Loop_Statement{  
    Special_Condition{  
        break;  
    }  
    "statement"  
}
```

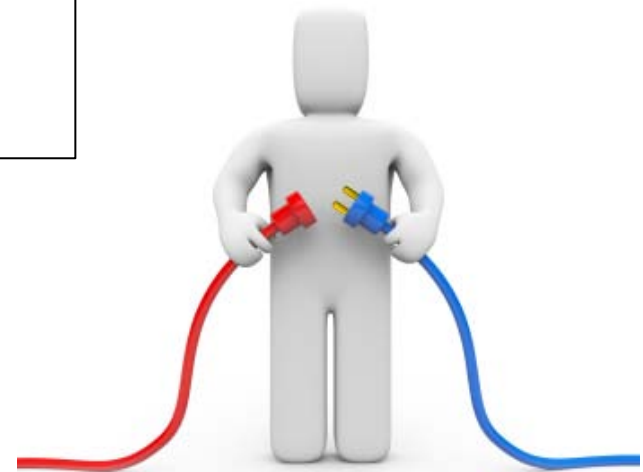


반복문 (iterative statement)

▶ continue

- Not this time
- Go on

```
Loop_Statement{  
    Special_Condition{  
        continue;  
    }  
    "statement"  
}
```



반복문

▶ break

```
public static void main(String[] args){
    int sum = 0;
    int i = 0;

    while (true){
        if ( sum > 100 )
            break;
        i++;
        sum += i;
    }

    System.out.println("i=" + i);
    System.out.println("sum=" + sum);
}
```



BREAK



CONTINUE

▶ continue 문

```
public static void main(String[] args{
    for ( int i = 0 ; i <= 10 ; i++ ) {
        if (i%3 == 0)
            continue;
        System.out.println(i);
    }
}
```

오늘 숙제

▶ Thinking

```
int idx = 0;
int sum = 0;

for ( idx=0 ; idx < 10 ; idx++ ){
    if ( idx % 2 == 0 )
        continue;
    if ( idx == 6 )
        break;
    sum += idx;
}
```

- 45? 25? 9? 4?



오늘 숙제

- ▶ 연산자의 의미와 용도를 다양하게 생각해보자
- ▶ 조건문과 반복문의 중첩 및 조합을 해보자
- ▶ 변수, 연산자, 조건문, 반복문 복습
- ▶ 반복문을 쓰지 않고 구구단 2단을 출력
 - `Int a = 2...`
- ▶ `for`문과 `while` 문을 사용하여 2단을 출력
- ▶ `for`문과 `while` 문을 사용하여 2단부터 9단까지 출력



오늘 숙제

▶ For / While Version

2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18

2 X 1 = 2
2 X 3 = 6
2 X 5 = 10
2 X 7 = 14
2 X 9 = 18

2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
·
·
·

3 X 1 = 3
3 X 2 = 6
·
·
·

9 X 8 = 72
9 X 9 = 81

2 X 2 = 4

2 X 4 = 8

2 X 6 = 12

6까지만 곱하고 끝!

2 X 1 = 2

2 X 2 = 4

2 X 3 = 6

2 X 4 = 8

2단 중 5곱할때 출력안해요~

2 X 6 = 12

2 X 7 = 14

2 X 8 = 16

2 X 9 = 18

