



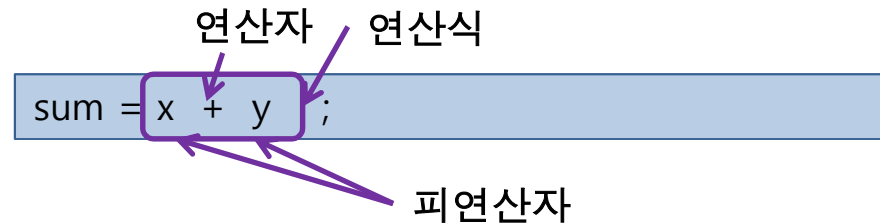
한림대학교 SW중심대학

생 초보를 위한 자바 프로그래밍

3장. 연산자

■ 연산이란?

- 데이터를 처리하여 결과를 산출하는 것



- 연산자(Operations)

- 연산에 사용되는 표시나 기호(+, -, *, /, %, =, ...)

- 피연산자(Operand): 연산 대상이 되는 데이터

- 연산식(Expressions)

- 연산자와 피연산자를 이용하여 연산의 과정을 기술한 것

연산자와 연산식

■ 연산자의 종류

연산자 종류	연산자	피연산자 수	산출값 타입	기능 설명
산술	+, -, *, /, %	이항	숫자	사칙연산 및 나머지 계산
부호	+, -	단항	숫자	음수와 양수의 부호
문자열	+	이항	문자열	두 문자열을 연결
대입	=, +=, -=, *=, /=, %= &=, ^=, =, <<=, >>=, >>>=	이항	다양	우변의 값을 좌변의 변수에 대입
증감	++, --	단항	숫자	1 만큼 증가/감소
비교	==, !=, >, <, >=, <=, instanceof	이항	boolean	값의 비교
논리	!, &, , &&,	단항 이항	boolean	논리적 NOT, AND, OR 연산
조건	(조건식) ? A : B	삼항	다양	조건식에 따라 A 또는 B 중 하나를 선택
비트	~, &, , ^	단항 이항	숫자 bloolean	비트 NOT, AND, OR, XOR 연산
쉬프트	>>, <<, >>>	이항	숫자	비트를 좌측/우측으로 밀어서 이동

연산의 방향과 우선 순위

■ 연산의 방향과 우선 순위

연산자	연산 방향	우선 순위
증감(++, --), 부호(+, -), 비트(~), 논리(!)	←	<div>높음</div> <div>↑</div> <div>↓</div> <div>낮음</div>
산술(*, /, %)	→	
산술(+, -)	→	
쉬프트(<<, >>, >>>)	→	
비교(<, >, <=, >=, instanceof)	→	
비교(==, !=)	→	
논리(&)	→	
논리(^)	→	
논리()	→	
논리(&&)	→	
논리()	→	
조건(?:)	→	
대입(=, +=, -=, *=, /=, %=, &=, ^=, =, <<=, >>=, >>>=)	←	

연산의 방향과 우선 순위

- 연산의 방향과 우선 순위
 - 연산자의 우선 순위에 따라 연산된다.

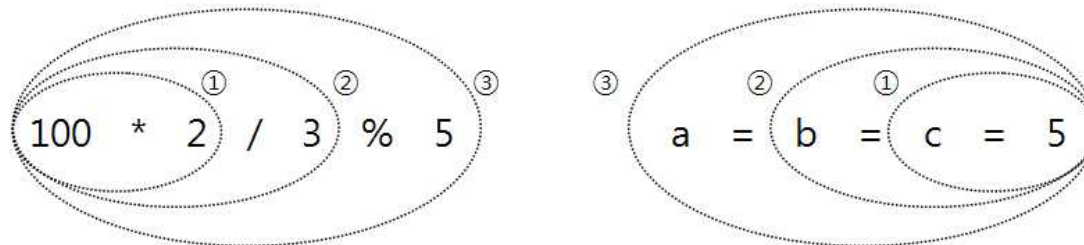
```
x > 0 && y < 0
```

- 다음과 같이 괄호를 이용 우선순위를 명시적으로 나타내는것을 권장

```
((x > 0) && (y < 0))
```

- 동일한 우선 순위의 연산자는 연산의 방향 존재

* , $/$, $\%$ 는 같은 우선 순위를 갖고 있다. 이들 연산자는 연산 방향이 왼쪽에서 오른쪽으로 수행된다. $100 * 2$ 가 제일 먼저 연산되어 200이 산출되고, 그 다음 $200 / 3$ 이 연산되어 66이 산출된다. 그 다음으로 $66 \% 5$ 가 연산되어 1이 나온다.



하지만 단항 연산자($++$, $--$, \sim , $!$), 부호 연산자($+$, $-$), 대입 연산자($=$, $+=$, $-=$, ...)는 오른쪽에서 왼쪽(\leftarrow)으로 연산된다. 예를 들어 다음 연산식을 보자.

산술 연산자

연산식			설명
피연산자	+	피연산자	덧셈 연산
피연산자	-	피연산자	뺄셈 연산
피연산자	*	피연산자	곱셈 연산
피연산자	/	피연산자	좌측 피연산자를 우측 피연산자로 나눗셈 연산
피연산자	%	피연산자	좌측 피연산자를 우측 피연산자로 나눈 나머지를 구하는 연산

```
int result = num % 3;
```

0, 1, 2 중의 한 값 num 을 3 으로 나눈 나머지

예제

```
public class ArithmeticOperatorExample {  
    public static void main(String[] args) {  
        int v1 = 5;  
        int v2 = 2;  
  
        int result1 = v1 + v2;  
        System.out.println("result1=" + result1);  
  
        int result2 = v1 - v2;  
        System.out.println("result2=" + result2);  
  
        int result3 = v1 * v2;  
        System.out.println("result3=" + result3);  
  
        int result4 = v1 / v2;  
        System.out.println("result4=" + result4);  
  
        int result5 = v1 % v2;  
        System.out.println("result5=" + result5);  
  
        double result6 = (double) v1 / v2;  
        System.out.println("result6=" + result6);  
    }  
}
```

- /와 % 응용

- 10의 자리와 1의 자리 분리

$69/10 = 6$	← 몫 6
$69\%10 = 9$	← 나머지 9

- x가 홀수인지 판단

```
int r = n % 2;      // r이 1이면 n은 홀수, 0이면 짝수
```

- n의 값이 3의 배수인지 확인

```
int s = n % 3;      // s가 0이면 n은 3의 배수
```


- 정수를 3으로 나눈 나머지를 출력

```
public class RemainderExample {  
  
    public static void main(String[] args) {  
        int num=6, x=3, remainder=0;  
        remainder = num%x;  
        System.out.println(remainder);  
    }  
  
}
```

- 정수형 변수에 70 이상의 값을 저장해서 받아서 분과 초를 출력하시오
- 정수형 변수에 값을 저장한후 2로 나눈후 나머지값을 출력하세요

- 문자열에서의 + 연산자 사용
 - 연산되는 문자열들을 결합 시킨다.

```
String f="app";  
String s="le";  
String res=f+s;
```

■ 문자열 연산자

- 피연산자 중 문자열이 있으면 문자열이 아닌 피연산자가 문자열로 변환된후 문자열로 결합

```
public class StringConcatExample {  
    public static void main(String[] args) {  
        String str1 = "JDK" + 6.0;  
        String str2 = str1 + " 특징";  
        System.out.println(str2);  
  
        String str3 = "JDK" + 3 + 3.0;  
        String str4 = 3 + 3.0 + "JDK";  
        System.out.println(str3);  
        System.out.println(str4);  
    }  
}
```

대입 연산자

■ 오른쪽 피연산자의 값을 좌측 피연산자인 변수에 저장

구분	연산식			설명
단순 대입 연산자	변수	=	피연산자	우측의 피연산자의 값을 변수에 저장
복합 대입 연산자	변수	+=	피연산자	우측의 피연산자의 값을 변수의 값과 더한 후에 다시 변수에 저장 (변수=변수+피연산자 와 동일)
	변수	-=	피연산자	우측의 피연산자의 값을 변수의 값에서 뺀 후에 다시 변수에 저장 (변수=변수-피연산자 와 동일)
	변수	*=	피연산자	우측의 피연산자의 값을 변수의 값과 곱한 후에 다시 변수에 저장 (변수=변수*피연산자 와 동일)
	변수	/=	피연산자	우측의 피연산자의 값으로 변수의 값을 나눈 후에 다시 변수에 저장 (변수=변수/피연산자 와 동일)
	변수	%=	피연산자	우측의 피연산자의 값으로 변수의 값을 나눈 후에 나머지를 변수에 저장 (변수=변수%피연산자 와 동일)
	변수	&=	피연산자	우측의 피연산자의 값과 변수의 값을 & 연산 후 결과를 변수에 저장 (변수=변수&피연산자 와 동일)
	변수	=	피연산자	우측의 피연산자의 값과 변수의 값을 연산 후 결과를 변수에 저장 (변수=변수 피연산자 와 동일)
	변수	^=	피연산자	우측의 피연산자의 값과 변수의 값을 ^ 연산 후 결과를 변수에 저장 (변수=변수^피연산자 와 동일)
	변수	<<=	피연산자	우측의 피연산자의 값과 변수의 값을 << 연산 후 결과를 변수에 저장 (변수=변수<<피연산자 와 동일)
	변수	>>=	피연산자	우측의 피연산자의 값과 변수의 값을 >> 연산 후 결과를 변수에 저장 (변수=변수>>피연산자 와 동일)
	변수	>>>=	피연산자	우측의 피연산자의 값과 변수의 값을 >>> 연산 후 결과를 변수에 저장 (변수=변수>>>피연산자 와 동일)

```
int sum=2;
sum+=10;
```


```
int sum=2;
sum=sum+10;
```

결과값 :sum =12

```
public class AssignmentOperatorExample {  
    public static void main(String[] args) {  
        int result = 0;  
        result += 10;  
        System.out.println("result=" + result);  
        result -= 5;  
        System.out.println("result=" + result);  
        result *= 3;  
        System.out.println("result=" + result);  
        result /= 5;  
        System.out.println("result=" + result);  
        result %= 3;  
        System.out.println("result=" + result);  
    }  
}
```

- 다음 복합대입연산자를 이용 수식의 결과를 출력하시오
 - 복합대입연산자 : /=, %=
 - int 타입
 - 변수명 : i, 값 : 10 -> 연산자의 좌측에 위치
 - 변수명 : j, 값 : 2 -> 연산자의 우측에 위치

- ++, --
 - 변수의 값을 1증가 시키거나 (++) 1 감소 (--) 시키는 연산자
 - 변수의 앞, 뒤에 모두 가능
- 사용경우

<pre>int i=0 ; i++;</pre>	<pre>int i=0 ; ++i ;</pre>		<pre>int i=0 ; i=i+1;</pre>
-------------------------------	--------------------------------	--	---------------------------------


```
public class IncreaseDecreaseOperatorExample {  
    public static void main(String[] args) {  
        int x = 10;  
        int y = 10;  
        int z=0;  
  
        System.out.println("-----");  
        x++;  
        System.out.println("x++=" + x);  
        ++x;  
        System.out.println("++x=" + x);  
  
        System.out.println("-----");  
        y--;  
        System.out.println("y--=" + y);  
        --y;  
        System.out.println("--y=" + y);  
  
    }  
}
```

비교 연산자

- ==, !=, <, >, <=, >=
 - boolean 타입인 true/false 산출

구분	연산식			설명
동등 비교	피연산자	==	피연산자	두 피 연산자의 값이 같은지를 검사
	피연산자	!=	피연산자	두 피 연산자의 값이 다른지를 검사
크기 비교	피연산자	>	피연산자	피 연산자 1 이 큰지를 검사
	피연산자	>=	피연산자	피 연산자 1 이 크거나 같은지를 검사
	피연산자	<	피연산자	피 연산자 1 이 작은지를 검사
	피연산자	<=	피연산자	피 연산자 1 이 작거나 같은지를 검사

- 흐름 제어문인 조건문(if), 반복문(for, while)에서 주로 이용
 - 실행 흐름을 제어할 때 사용
- 문자열 비교시 equals()

- 문자열 비교 :
 - 비교연산자 사용시 오류 가능성 있음
 - 문자열의 비교시에는 equals()를 이용한다.
 - 형식
 - 기준 문자열 변수.equals(비교할 문자열변수);

```
String strVar1 = "자바 ";  
String strVar2 = "java";  
System.out.println( strVar1.equals(strVar2));
```

```
public class CompareOperatorExample1 {  
    public static void main(String[] args) {  
        int num1 = 10;  
        int num2 = 10;  
        boolean result1 = (num1 == num2);  
        boolean result2 = (num1 != num2);  
        boolean result3 = (num1 <= num2);  
        System.out.println("result1=" + result1);  
        System.out.println("result2=" + result2);  
        System.out.println("result3=" + result3);  
  
        String strVar1 = " 자바 ";  
        String strVar2 = " 자바";  
        System.out.println( strVar1.equals(strVar2));  
    }  
}
```

- 변수 a,b를 ==, >, < 를 이용하여 boolean 값을 출력하시오

타입	변수명	초기값	결과값	
int	a	10		
int	b	5		

- 다음의 두변수가 같은지 결과를 출력하시오

타입	변수명	초기값	결과값	
String	a	apple		
String	b	aplpe		

논리 연산자

- **&&, ||, &, |, ^, !**
- **피연산자는 boolean 타입만 사용 가능**

구분	연산식		결과	설명
AND (논리곱)	true	&& 또는 &	true	피 연산자 모두가 true 일 경우에만 연산 결과는 true
	true		false	
	false		true	
	false		false	
OR (논리합)	true	 또는 	true	피 연산자 중 하나만 true 이면 연산 결과는 true
	true		false	
	false		true	
	false		false	
XOR (배타적 논리합)	true	^	true	피 연산자가 하나는 true 이고 다른 하나가 false 일 경우에만 연산 결과는 true
	true		false	
	false		true	
	false		false	
NOT (논리부정)		!	true	피 연산자의 논리값을 바꿈
			false	

```
public class CompareOperatorExample1 {
    public static void main(String[] args) {
        int num=11;
        System.out.println(( (num>=10) && (num<=122) ));
    }
}
```