Operator - Conditional(Logic), Bit

Conditional(Logic) Operator

&&	논리 AND, A && B A와 B 모두 true면 true 반환	→
||	논리 OR, A || B A와 B 둘중 하나라도 true면 true반환	→
!	논리 NOT, !A A가 true이면 false, A가 false이면 true 반환	←

```
class LogicOp {
	public static void main(String[] args) {
	int num1 = 10, num2 = 20;
	boolean result1 = (num1==10 && num2==20);
	boolean result2 = (num1<=12 || num2>=30);

	System.out.println("num1==10 그리고 num2==20 : "+result1); // result1 : true
	System.out.println("num1<=12 또는 num2>=30 : "+result2);   // result2 : true

	if(!(num1==num2)){
	System.out.println("num1과 num2는 같지 않다.");
	}
	else
	System.out.println("num1과 num2는 같다.");
	}
}
```

Conditional(Logic) Operator의 SCE (Short Circuit Evaluation)

```
class SCE {
	public static void main(String[] args) {
	int num1 = 0, num2 = 0;
	boolean result;

	result = (num1+=10)<0 && (num2+=10)>0;
	// false && true

	System.out.println("result="+result);
	System.out.println("num1="+num1+", num2="+num2); // num1=10, num2=0

	result = (num1+=10)>0 || (num2+=10)>0;
	// true && true

	System.out.println("result="+result);
	System.out.println("num1="+num1+", num2="+num2); // num1=20, num2=0
	}
}
```

- SCE Case1

(num1+=10)<0	&&	(num2+=10)>0
이 부분이 false이면		이 부분은 볼 것 없음**(연산진행X)**

- SCE Case2

(num1+=10)>0	||	(num2+=10)>0
이 부분이 true이면		이 부분은 볼 것 없음**(연산진행X)**

Bit Operator

&	비트단위 AND n1 & n2	→
|	비트단위 OR n1 | n2	→
^	비트단위 XOR n1 ^ n2	→
~	비트단위 NOT, 피연산자의 모든비트 반전 ~n1	←

- 13 & 7 = 5

```
00001101
   &
00000111
-----------
00000101
```

진리표

비트AND	두 비트 모두 1이면 1
비트OR	두 비트 중 하나라도 1이면 1
비트XOR	두 비트가 서로 다를때만 1
비트NOT	0은 1, 1은 0으로 반전

Bit Shift Operator

<<	• 피연산자의 비트 열을 왼쪽으로 이동 • 이동에 따른 공간은 0으로 채움 • n<<2 n의 비트 열을 두칸 왼쪽으로 이동시킨 결과 반환	→
>>	• 피연산자의 비트 열을 오른쪽으로 이동 • 이동에 따른 공간은 양수이면 0, 음수이면 1으로 채움(MSB 유지) • n>>2 → n의 비트 열을 두칸 오른쪽으로 이동시킨 결과 반환	→
>>>	• 피연산자의 비트 열을 오른쪽으로 이동 • 이동에 따른 공간은 0으로 채움(MSB 무시) • n>>2	→

	→ n의 비트 열을 두칸 오른쪽으로 이동시킨 결과 반환	

- 비트 쉬프트 연산자의 특징
  - 왼쪽으로 한칸 이동시마다 2의 배수의 곱 만큼 증가 (X2)
  - 오른쪽으로 한칸 이동시마다 2의 배수의 나눗셈 만큼 감소 (½)
