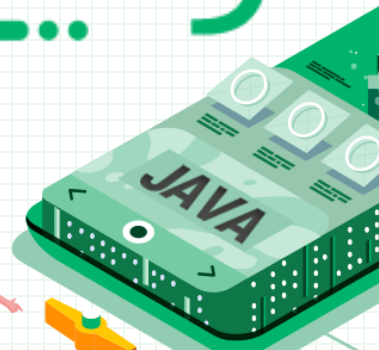




# 안드로이드 프로그래밍을 위한 자바기초 \_...



## ③ 연산자의 기본 문법 이해하기



## 학습목표

- 수식과 연산자의 개념을 이해하고 연산자의 종류를 이해할 수 있다.
- 다양한 연산자를 활용하여 프로그래밍을 작성할 수 있다.



## 학습내용

- 연산자 이해하기
- 연산자 활용하기

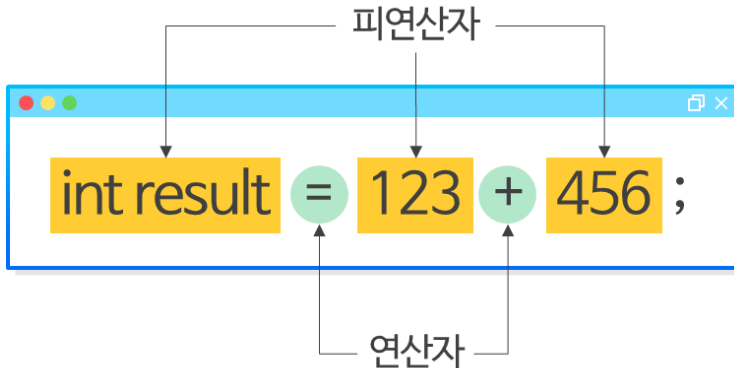
## ③ 연산자의 기본 문법 이해하기

## 연산자 이해하기

## ➤ 수식(expression) 이해하기

## ▶ 수식이란?

- 피연산자(operand)와 연산자(operator)로 구성된 문장
- 피연산자는 상수값 또는 변수를 사용



## ➤ 연산자(operator) 이해하기

## 1) 연산자 형식

- 단항 연산자 - 피연산자가 한 개인 연산자
- 이항 연산자 - 피연산자가 두 개인 연산자
- 삼항 연산자 - 피연산자가 세 개인 연산자

## 연산자 형식과 기호

단항 연산자	이항 연산자				삼항 연산자
-	*	=	<=		?:
~	/	<<	>=	^	
!	%	>>	==	&&	
++	+	<	!=		
--	-	>	&		

### ③ 연산자의 기본 문법 이해하기

## 연산자 이해하기

### 2) 연산자의 종류


#### 연산자종류와기호

산술 연산자	+ - * / %
대입 연산자	= += -= *= /= %=
증감 연산자	++ --
관계 연산자	== != > < >= <=
논리 연산자	! &&
조건 연산자	? :
비트 연산자	&   ^ ~ << >>  = ^= ~= <<= >>=
기타 연산자	(자료형) instanceof

### 3) 연산자 우선순위

- 우선순위가 높은 연산자가 있는 수식이 먼저 계산됨

우선순위



1	괄호, 단항 연산자	( ) - ~ ! ++ -- (자료형)
2	산술 연산자	* / % + - 비트 연산자(<< >> >>>)
3	관계 연산자	== != > < >= <= instanceof
4	비트 연산자	&   ^ ~
5	논리 연산자	&&
6	조건 연산자	? :
7	대입 연산자	= += -= *= /= %=

## ③ 연산자의 기본 문법 이해하기

## 연산자 활용하기

## ➤ 산술 연산자 이해하기

## 1) 산술 연산자란?

- 이항 연산자
  - ✓ 덧셈, 곱셈, 뺄셈, 나눗셈은 정수형과 실수형 계산 가능
  - ✓ 나머지는 정수형 계산만 가능

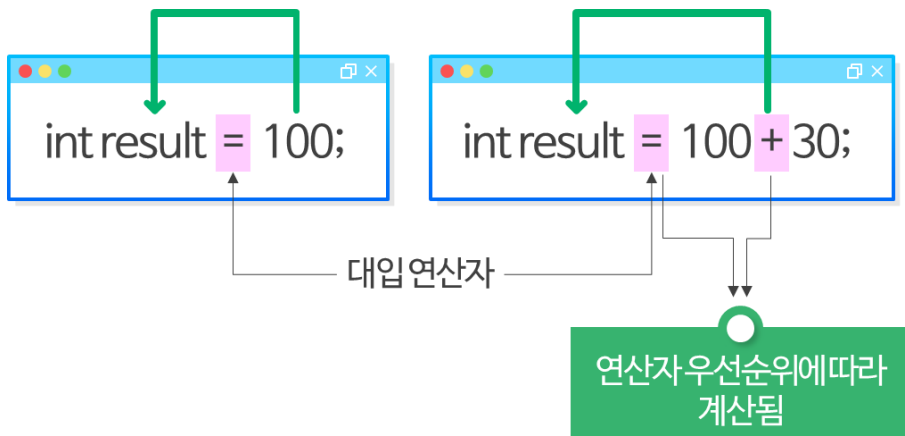
## 2) 산술 연산자 종류

기호	의미	설명
+	덧셈	<ul style="list-style-type: none"> <li>정수형과 정수형의 결과는 정수형</li> <li>정수형과 실수형의 결과는 실수형</li> <li>실수형과 실수형의 결과는 실수형</li> </ul>
-	뺄셈	
*	곱셈	
/	나눗셈	<ul style="list-style-type: none"> <li>정수형과 정수형의 나눗셈 결과는 정수형</li> <li>정수형과 실수형의 나눗셈 결과는 실수형</li> <li>실수형과 실수형의 나눗셈 결과는 실수형</li> <li>0으로 나누면 실행오류 발생</li> </ul>
%	나머지	계산 결과값: 나눗셈을 한 후 나머지 값

## ➤ 대입 연산자 이해하기

## 1) 대입 연산자란?

- 이항 연산자
- 연산자 오른쪽에 있는 값을 왼쪽에 대입 또는 저장하는 연산자



## ③ 연산자의 기본 문법 이해하기

## 연산자 활용하기

## 2) 복합 대입 연산자란?

- 단순 대입 연산자 (=)와 산술 연산자를 복합적으로 사용하는 연산자

기호	사용 예	동일한 의미
+=	sum += 3;	sum = sum + 3;
-=	sum -= 3;	sum = sum - 3;
*=	sum *= 3;	sum = sum * 3;
/=	sum /= 3;	sum = sum / 3;
%=	sum %= 3;	sum = sum % 3;

## ➤ 증감 연산자 이해하기

## 1) 증감 연산자란?

- 단항 연산자
- 정수형 변수의 값을 1 증가시키거나 감소시킬 때 사용
- 변수이름 앞 또는 뒤에 붙여서 사용
- 연산자 기호
  - ✓ 1 증가시키는 기호 : ++
  - ✓ 1 감소시키는 기호 : --

## 2) 후위/전위 증감 연산자

- 후위 증감 연산자
  - ✓ 변수 이름 뒤에 증감 연산자를 사용
  - ✓ 전체 수식을 계산한 다음 제일 마지막에 증감 연산자 계산
  - ✓ 예) a++, b--
- 전위 증감 연산자
  - ✓ 변수 이름 앞에 증감 연산자를 사용
  - ✓ 전체 수식을 계산하기 전에 증감 연산자를 계산한 후 수식을 계산
  - ✓ 예) ++a, --b;

## ③ 연산자의 기본 문법 이해하기

## 연산자 활용하기

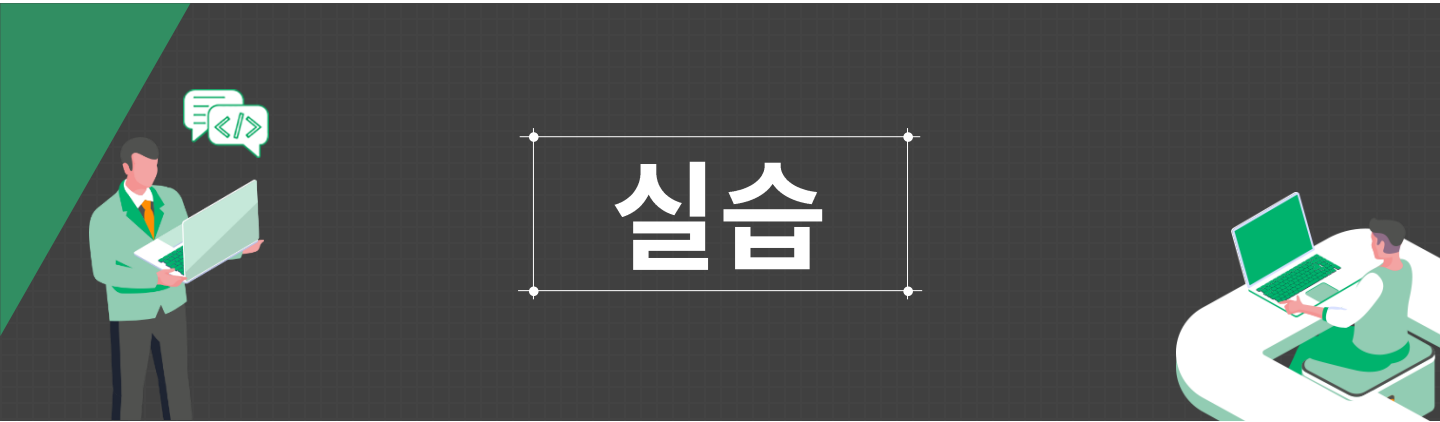
### ▶ 관계 연산자 이해하기

#### ▶ 관계 연산자란?

- (1) 이항 연산자
- (2) 연산자를 기준으로 양쪽의 피연산자를 비교하여 참(true)과 거짓(false)을 판별
- (3) 관계 연산자 종류

기호	설명	사용 예	의미
==	같다	$A == B$	A와 B가 같다
!=	다르다	$A != B$	A와 B가 다르다
>	크다	$A > B$	A가 B보다 크다
<	작다	$A < B$	A가 B보다 작다
>=	크거나 같다	$A >= B$	A가 B보다 크거나 같다
<=	작거나 같다	$A <= B$	A가 B보다 작거나 같다

## ③ 연산자의 기본 문법 이해하기



## 첫 번째 연산자 활용 실습



## 실행 화면

```
100 + 30 = 130
100 - 30 = 70
100 * 30 = 3000
100 / 30 = 3
100 % 30 = 10
100 + 30 * 3 = 190
50.45 + 15.98 = 66.43
50.45 - 15.98 = 34.47
50.45 * 15.98 = 806.191
50.45 / 15.98 = 3.1570713391739678
(int)50.45 + (int)15.98 = 65
(int) (50.45 + 15.98) = 66
(int)50.45 / (int) 15.98 = 3...
```

- 소스 파일명 : [OpExam\_1.java]
- 자세한 내용은 실습 영상을 확인해보세요.



## ③ 연산자의 기본 문법 이해하기

## 연산자 활용하기

## ➤ 논리 연산자 이해하기

## 1) 논리 연산자의 개요

- (1) 단항 연산자 1개, 이항 연산자 2개
- (2) 피연산자를 대상으로 참(true)과 거짓(false)을 판별
- (3) 피연산자 : 상수, 변수, 관계식
- (4) 참이면 true, 거짓이면 false

## 관계 연산자와 논리 연산자의 비교

항목	관계 연산자	논리 연산자
형식	이항 연산자	단항 연산자, 이항 연산자
동작	피연산자의 값을 비교	피연산자의 참과 거짓을 비교
결과값	참이면 true, 거짓이면 false	참이면 true, 거짓이면 false

## 2) 논리 연산자의 종류

항목	기호	형식	설명
NOT	!	단항 연산자	피연산자를 부정
AND	& &	이항 연산자	피연산자 모두 참이면 참
OR		이항 연산자	피연산자 모두 거짓이면 거짓

## ③ 연산자의 기본 문법 이해하기

## 연산자 활용하기

## 3) 논리 연산자의 우선순위

- NOT > AND > OR
- 1. 괄호()
- 2. 괄호 안 관계 연산자
- 3. NOT
- 4. AND
- 5. OR
- 6. 대입 연산(=)

## ➤ 조건 연산자 이해하기

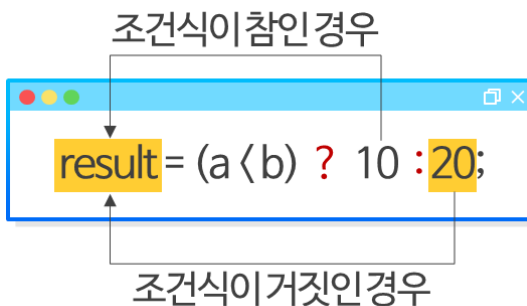
## 1) 조건 연산자의 개요

- (1) 삼항 연산자
- (2) 피연산자의 역할

- ✓ 첫 번째 피연산자 - 조건식 (참 또는 거짓)
- ✓ 두 번째 피연산자 - 조건식이 참인 경우 수행하는 상수, 변수, 수식
- ✓ 세 번째 피연산자 - 조건식이 거짓인 경우 수행하는 상수, 변수, 수식

```
result = (조건식) ? (상수,변수,수식) : (상수,변수,수식) ;
```

## 2) 조건 연산자 사용 예



### ③ 연산자의 기본 문법 이해하기

## 연산자 활용하기

### ➤ 비트 연산자 이해하기

#### 1) 비트 연산자의 개요

(1) 비트(bit) 단위로 연산

(2) 종류

- ✓ 비트 논리 연산자
- ✓ 비트 시프트(shift) 연산자

#### 비트 연산자의 종류

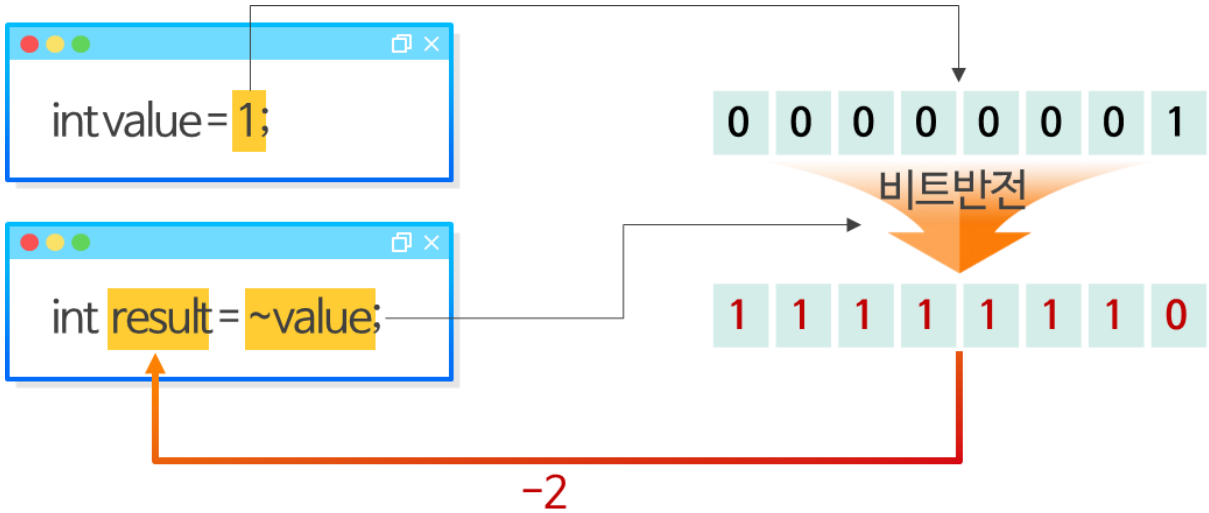
종류	항목	기호	형식	설명
비트 논리 연산자	NOT	~	단항 연산자	비트 반전(0→ 1, 1→ 0)
	AND	&	이항 연산자	피연산자가 모두 1인 경우만 1
	OR			피연산자가 모두 0인 경우만 0
	XOR	^		피연산자가 서로 다르면 1
시프트 연산자	왼쪽 시프트	<<		왼쪽으로 비트 이동(부호 유지)
	오른쪽 시프트	>>		오른쪽으로 비트 이동(부호 유지)
		>>>		부호 비트를 포함한 오른쪽으로 비트 이동

## ③ 연산자의 기본 문법 이해하기

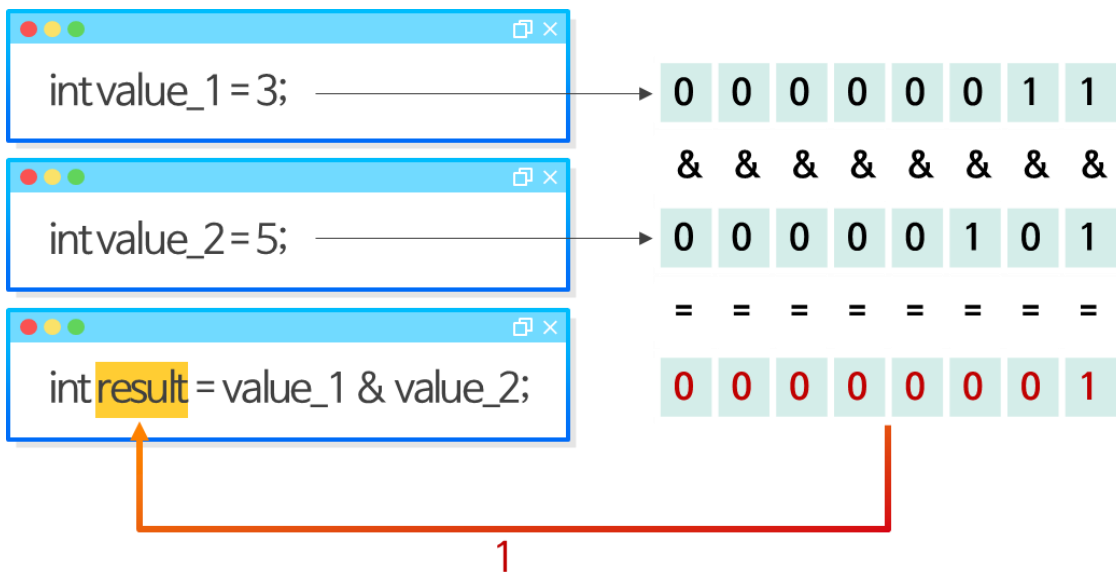
## 연산자 활용하기

## 2) 비트 연산자 사용 예

## (1) 비트 NOT 연산



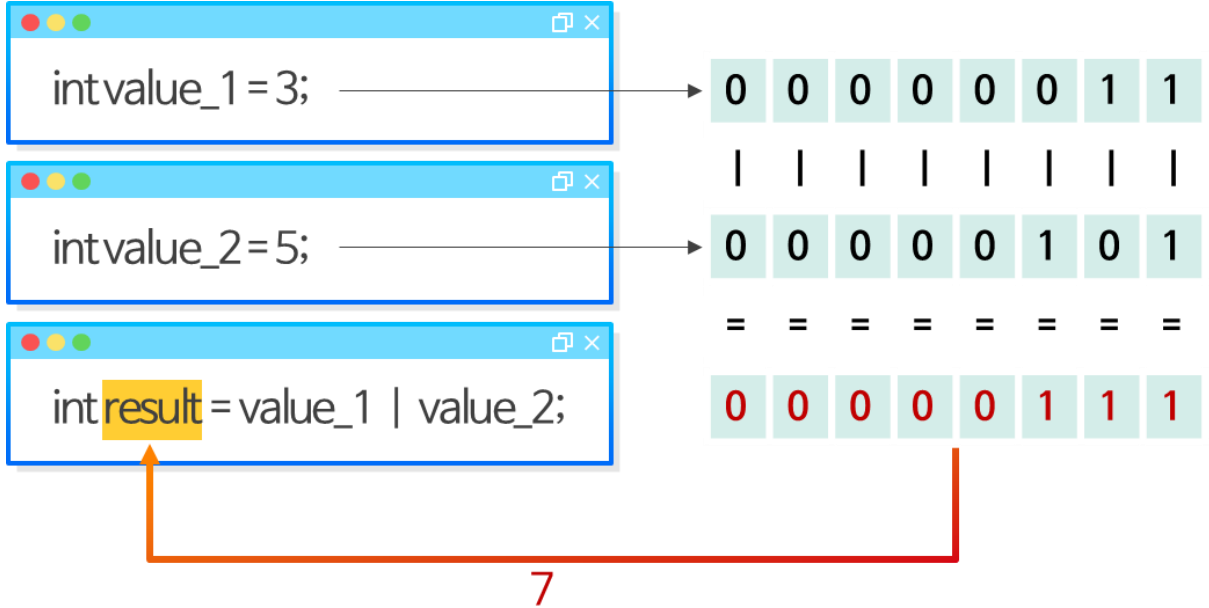
## (2) 비트 AND 연산



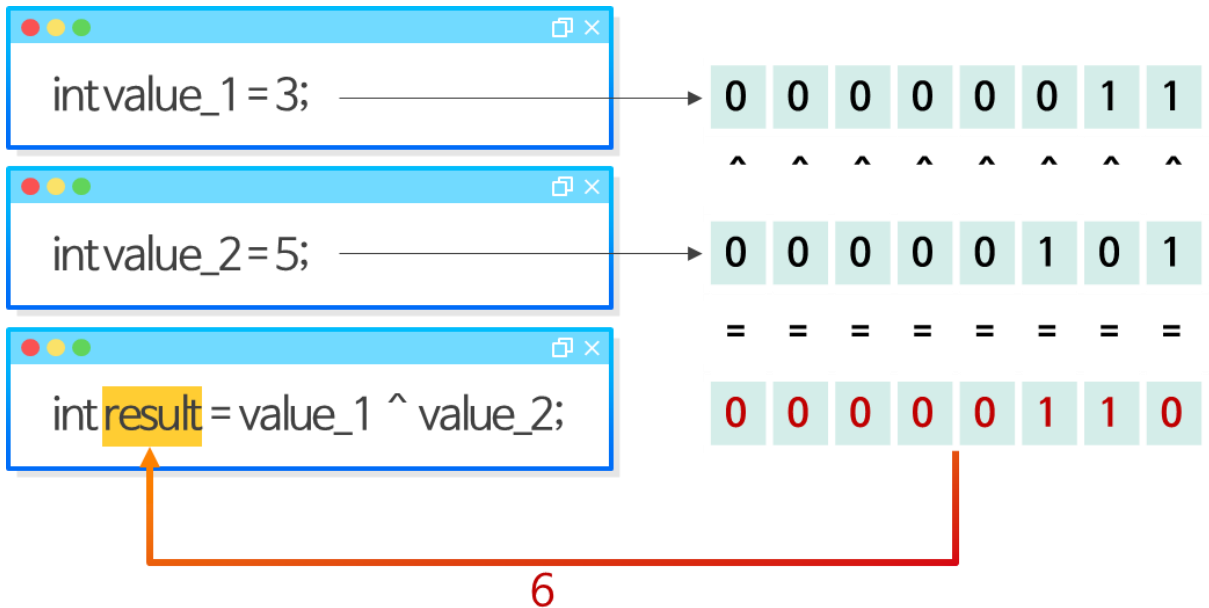
## ③ 연산자의 기본 문법 이해하기

## 연산자 활용하기

## (3) 비트 OR 연산



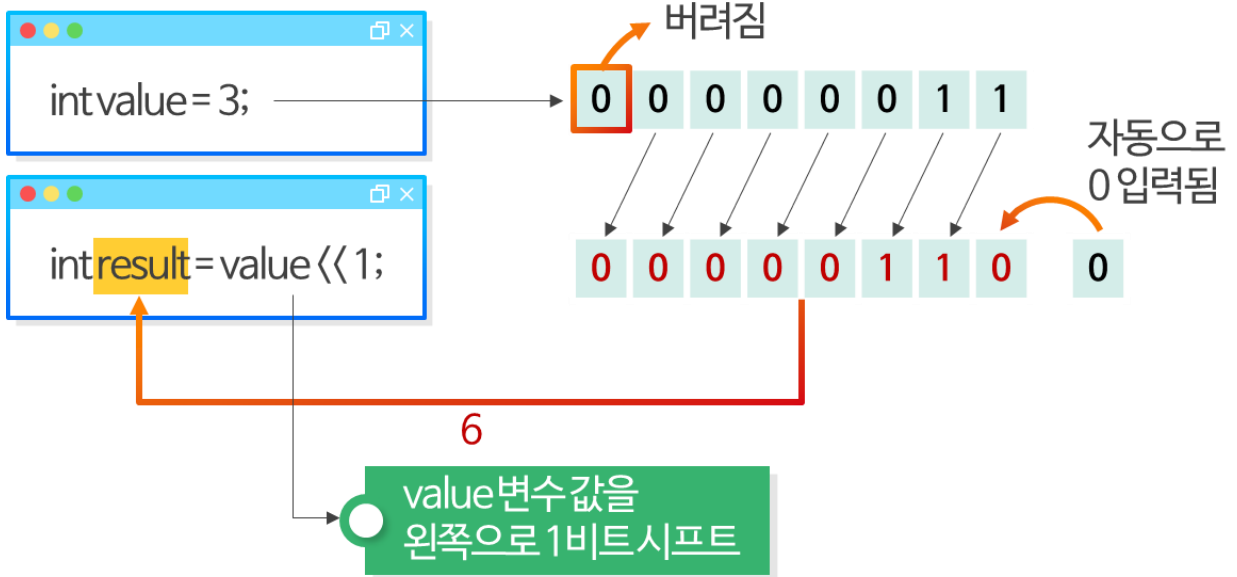
## (4) 비트 XOR 연산



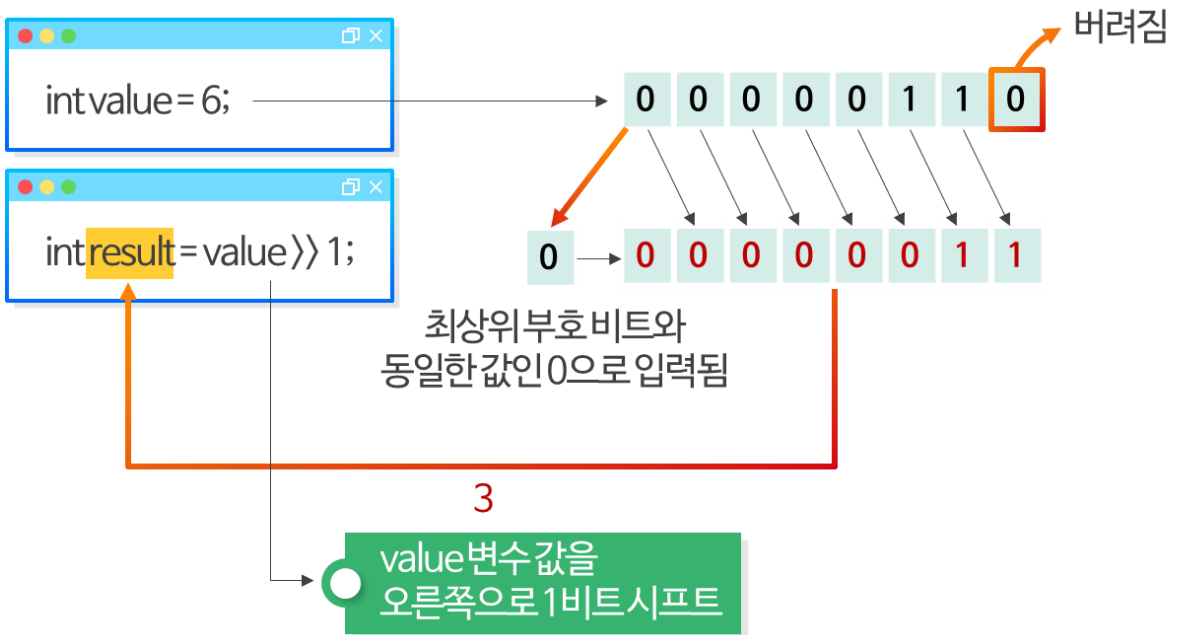
## ③ 연산자의 기본 문법 이해하기

## 연산자 활용하기

## (5) 왼쪽 시프트 연산 (부호 유지)



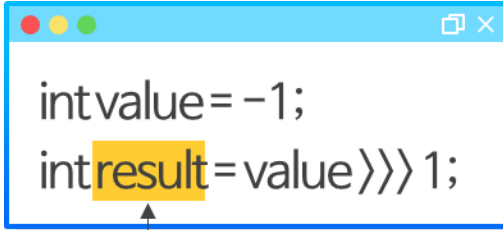
## (6) 오른쪽 시프트 연산 (부호 유지)



## ③ 연산자의 기본 문법 이해하기

## 연산자 활용하기

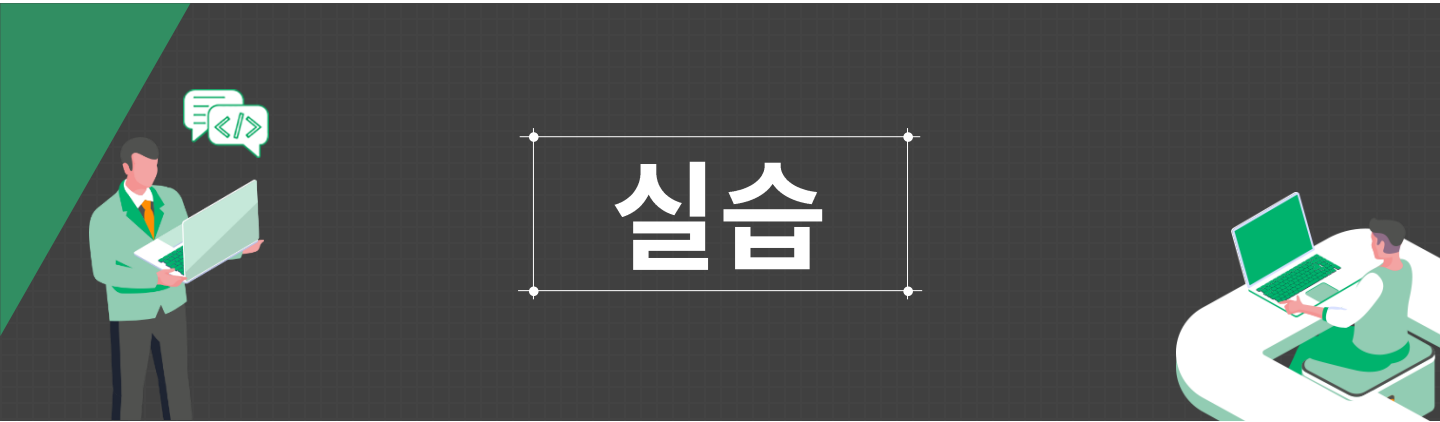
(7) 오른쪽 시프트 연산 (부호 포함)



```
int value = -1;  
int result = value >>> 1;
```

2147483647

## ③ 연산자의 기본 문법 이해하기



## 두 번째 연산자 활용 실습



## 실행 화면

```
!(15 < 25) : false
(15 > 10) && (25 < 20) : false
(15 > 10) || (25 < 20) : true
(15 > 10) || (25 < 20) && !(15 < 25) : true
(15 < 25) ? 10 : 20; value = 10
(15 < 25) ? a : b; value = 15
(15 < 25) ? b-a : a-b; value = 10
~1 : -2
3 & 5 : 1
3 | 5 : 7
3 ^ 5 : 6
10 << 1 : 20
10 >> 1 : 5...
```

- 소스 파일명 : [OpExam\_2.java]
- 자세한 내용은 실습 영상을 확인해보세요.





## 정리하기

### ■ 연산자 이해하기

- 수식이란?
  - 피연산자(operand)와 연산자(operator)로 구성된 문장
  - 피연산자는 상수 또는 변수를 사용
- 연산자의 형식
  - 단항 연산자 : 피연산자가 한 개의 연산자
  - 이항 연산자 : 피연산자가 두 개의 연산자
  - 삼항 연산자 : 피연산자가 세 개의 연산자
- 연산자 종류
  - 산술 연산자, 대입 연산자, 증감 연산자, 관계 연산자, 논리 연산자, 조건 연산자, 비트 연산자, 기타 연산자
- 연산자 우선순위

우선순위	종류	기호
1	괄호, 단항 연산자	() ~ ! ++ -- (자료형)
2	산술 연산자	* / % + - 비트 연산자(<< >> >>>)
3	관계 연산자	= != > < >= <= instanceof
4	비트 연산자	&   ^ ~
5	논리 연산자	&&
6	조건 연산자	?:
7	대입 연산자	= += -= *= /= %=



## 정리하기

### ■ 연산자 활용하기

- 산술 연산자
  - 덧셈, 뺄셈, 곱셈, 나눗셈 연산자는 정수형과 실수형 계산 가능
  - 나머지 연산자는 정수형 계산만 가능
- 대입 연산자
  - 연산자 오른쪽에 있는 값을 왼쪽에 대입 또는 저장하는 연산자
- 증감 연산자
  - 정수형 변수의 값을 1 증가시키거나 감소시킬 때 사용
- 관계 연산자
  - 연산자를 기준으로 양쪽의 피연산자를 비교하여 참(true)과 거짓(false)을 판별
- 논리 연산자
  - 피연산자를 대상으로 참(true)과 거짓(false)을 판별
- 조건 연산자
  - 피연산자의 역할
    - 첫 번째 피연산자 : 조건식(참 또는 거짓)
    - 두 번째 피연산자 : 조건식이 참인 경우 수행하는 상수, 변수, 수식
    - 세 번째 피연산자 : 조건식이 거짓인 경우 수행하는 상수, 변수, 수식
- 비트 연산자
  - 비트(bit) 단위로 연산
  - 종류
    - 비트 논리 연산자
    - 비트 시프트(shift) 연산자