연산자
(Operator)

▶ 연산자(Operator)

연산에서 사용되는 표시나 기호

✓ 연산

프로그램에서 데이터를 처리하여 결과를 산출하는 것

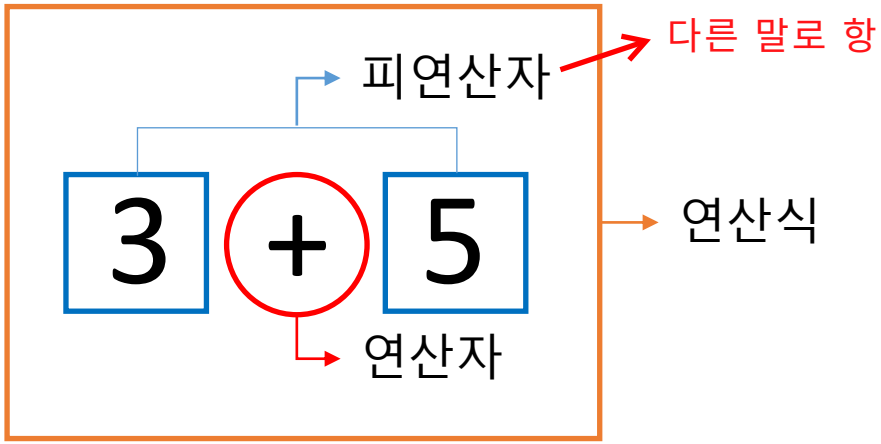
✓ 피연산자 → 연산을 당하는 놈

연산에 사용되는 데이터

✓ 연산식

연산자와 피연산자를 이용해서 연산과정을 기술한 것

▶ 연산자(Operator)



연산식은 반드시 하나의 결과 값을 산출

여러 개의 연산자가 하나의 연산식에 올 경우 연산자 우선순위에 따라 연산

동일한 우선순위를 가질 경우 왼쪽부터 오른쪽의 방향으로 연산

(단, 단항/대입 연산자 제외)

(괄호가 우선순위 중 제일 높음)

▶ 연산자(Operator)

✓ 단항 연산자

피연산자(항) 하나를 이용해 연산

✓ 이항 연산자

피연산자(항) 두개를 이용해 연산

✓ 삼항 연산자

피연산자(항) 세개를 이용해 연산

▶ 연산자 종류와 우선 순위

KH정보교육원

종류	구분	세부 구분	연산자	우선순위
최우선 연산자	직접 접근 연산자		( ) . { }	1
단항 연산자			+ - ! (자료형) ++ -- ~	2
이항 연산자	산술 연산자		* / %	3
			+ -	4
	쉬프트 연산자		>> << >>>	5
	비교 연산자		> < >= <=	6
			== !=	7
	논리 연산자	비트 논리 연산자	&	8
			^	9
			|	10
		일반 논리 연산자	&&	11
			||	12
삼항 연산자			(조건식) ? 참일 때 사용 값 : 거짓일 때 사용 값	13
대입 연산자	순수 대입		=	14
	복합 대입	산술 대입	+= -= *= /= %=	
		쉬프트 대입	<<= >>= >>>=	
		비트 논리 대입	&= ^= |=	
나열 연산자			,	15

▶ 산술 연산자

연산 방법과 우선 순위가 일반 수학과 동일

✓ 산술 연산자 설명

+ : 더하기

- : 빼기

* : 곱하기

/ : 나누기의 몫 구하기

% : 나누기의 나머지 구하기

▶ 산술 연산자

KH정보교육원

✓ * / % 예시

```
int a = 10, b = 20, c = 0;
c = a * b;
c = a / b;
c = a % b;
```

* '/' 연산 시 형 변환에 유의해야 함

✓ + - 예시

```
int a = 10, b = 20, c = 0;
c = a + b;
c = a - b;
```

▶ 복합 대입 연산자

다른 연산자와 대입 연산자가 결합한 것으로
자기 자신과 연산 후 연산 결과를 자기 자신에게 누적 대입

a += 10	═══	a = a + 10
a -= 10	═══	a = a – 10
a *= 10	═══	a = a * 10
a /= 10	═══	a = a / 10
a %= 10	═══	a = a % 10

* 증감 연산과 비슷해 보이지만 증감연산자(++, --)는 1씩 증가
대입 연산자는 원하는 값을 증가시키고 그 변수에 저장 가능

▶ 증감 연산자 → 단항 연산자

✓ 증감 연산자 : ++ --

피연산자의 값에 1을 더하거나 빼는 연산자

위치에 따라 출력 값이 다르게 나타남

전위 연산 : 먼저 연산 후 다른 연산 실행 ++n

후위 연산 : 다른 연산 우선 실행 후 연산 n++

전후위가 한 프로그램에 여럿 존재하면 헷갈리니까
암묵적으로 후위 연산을 많이 사용

▶ 증감 연산자

✓ 전위 연산자 예시

```
int a = 10;
int b = ++a;
System.out.println(a + ", " + b);
```

✓ 후위 연산자 예시

```
int a = 10;
int b = a++;
System.out.println(a + ", " + b);
```

▶ 비교 연산자

왼쪽을 기준으로 예) 5>3 이면
5는 3보다 크다 이런 식으로 읽는다

✓ > >= <= <

두 피연산자의 값의 크기 비교

기본형 boolean, 참조형을 제외하고 나머지 자료형에 모두 사용 가능

✓ > >= <= < 연산자 예시

```
if(a < b) {}

int result = a > b ? a++ : b--;

for(int a = 0; a <= b; a++) {}

while(a >= b) {}
```

▶ 비교 연산자

✓ == !=

데이터가 같은지, 다른지 비교할 때 쓰이며,
비교 결과 값으로 항상 논리 값(true, false)이 나옴
피연산자로 모든 자료형(기본형, 참조형) 사용 가능

a == b : a와 b가 같으면 true
a != b : a와 b가 다르면 true

▶ 비교 연산자

✓ == 연산자 예시

```
if(a == b) {}

int result = a == b ? a++ : b--;
```

✓ != 연산자 예시

```
if(a != b) {}

int result = a != b ? a++ : b--;
```

▶ 논리 연산자

(a && b || c && d)와 같은 식이 있을때
논리연산자의 우선 순위는 &&이기 때문에
&&먼저 연산 후 || 연산

KH정보교육원

논리 값 두 개를 비교하는 연산자

&& : 두 피연산자가 모두 true일 때 true 반환 (AND)

|| : 두 피연산자 중 하나만 true여도 true 반환 (OR)

a	b	a && b	a || b
true	true	true b값 추정 가능	true b값 추정 불가능
true	false	false b값 추정 가능	true b값 추정 불가능
false	true	false b값 추정 불가능	true b값 추정 가능
false	false	false b값 추정 불가능	false b값 추정 가능

▶ 논리 부정 연산자 (단항 연산자)

✓ 논리 부정 연산자 : !

논리 값을 부정하여 반대 값으로 변경

제어문을 활용할 때 많이 쓰임

✓ 논리 부정 연산자 예시

```
boolean bool1 = true;
boolean bool2 = !bool1;
System.out.println(bool2);
```

▶ 삼항 연산자

조건식 ? 식1 : 식2;

조건식의 결과 값에 따라 연산을 처리하는 방식

결과 값이 참일 경우 식1, 거짓일 경우 식2 수행

삼항 연산자 안에 삼항 연산자를 중첩하여 쓰는 것도 가능

✓ 삼항 연산자 예시

```
int result1 = a > b ? a++ : b--;
int result2 = a < b ? a++ : (b == 0 ? a-- : b++);
```