

## chap2 키보드에서 입력된 내용 문자열로 얻기

```
1 package sec04.exam05;
2
3 import java.util.Scanner;
4
5 public class ScannerExample {
6     public static void main(String[] args) throws Exception {
7         Scanner scanner = new Scanner(System.in);
8         String inputData;
9
10        while(true) {
11            inputData = scanner.nextLine();
12            System.out.println("입력된 문자열: " + inputData );
13            // System.out.println("입력된 문자열: \"\" + inputData + "\"");
14            if(inputData.equals("q")) {
15                break;
16            }
17        }
18
19        System.out.println("종료");
20    }
21 }
```

- 3라인 import java.util.Scanner; - import문이라 하는데 Scanner가 java.util 패키지에 있다는 것을 컴파일러에게 알려주는 역할을 함
- 11라인 scanner.nextLine() : String 타입의 문자열을 읽음
- 14라인 문자열(String)이 동일한지 비교시는 equals() 메소드를 사용한다.  
(자바는 기본타입(byte, short, int, long, float, double, boolean)의 값이 동일한지 비교시에는 == 을 사용)

## 연습문제

다음과 같이 출력되도록 코드를 작성해 보세요~

```
이름: 감자바  
나이: 25  
전화: 010-1234-5678
```

힌트

- ① 이름, 나이, 전화 번호 변수 선언  
이름 : string, 나이 : int, 전화번호 : string
- ② System.out.println을 이용해 출력

## 연습문제

스캐너를 이용해서 이름, 주민번호 앞 6자리, 전화번호를 키보드에서 입력받고 출력하는 코드를 작성해 보세요.

```
[필수 정보 입력]
1. 이름: 홍길동
2. 주민번호 앞 6자리: 123456
3. 전화번호: 010-1234-5678
|
[입력된 내용]
1. 이름: 홍길동
2. 주민번호 앞 6자리: 123456
3. 전화번호: 010-1234-5678
```

힌트

```
System.out.println("[필수 정보 입력]");
System.out.print("1. 이름: ");
String name = scanner.nextLine();
```

Chapter

# 03

## 연산자

### 03-1. 연산자와 연산식

- 0. 시작하기 전에
- 1. 연산자의 종류
- 2. 연산의 방향과 우선순위
- 3. 키워드로 끝내는 핵심 포인트

## 0. 시작하기 전에

[핵심 키워드] : 연산자, 피연산자, 연산 방향, 연산 우선순위

[핵심 포인트]

- 프로그램에서 데이터를 처리하여 결과를 산출하는 것을 연산(operation)이라고 한다.
- 자바의 다양한 연산자를 알아보고, 연산자가 복합적으로 구성된 연산식에서의 우선순위를 알아본다.

### ❖ 연산자 (operator)

- 연산에 사용되는 표시나 기호

### ❖ 피연산자 (operand)

- 연산자와 함께 연산되는 데이터

### ❖ 연산식 (expression)

- 연산자와 피연산자 사용하여 연산 과정 기술한 것

$X + y$



연산자

$X - y$



피연산자

$X * y + z$

$x = y$

# 1. 연산자의 종류

## ❖ 자바에서 제공하는 연산자

- 산출되는 값의 타입이 연산자별로 다름

연산자 종류	연산자	피연산자 수	산출값	기능
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	1만큼 증가/감소
비교	==, !=, >, <, >=, <=, instanceof	이항	boolean	값의 비교
논리	!, &,  , &&,	단항 이항	boolean	논리 부정, 논리곱, 논리합
조건	(조건식) ? A : B	삼항	다양	조건식에 따라 A 또는 B 중 하나를 선택

# 1. 연산자의 종류

- 연산식은 반드시 하나의 값 산출
- 하나의 값이 오는 모든 자리에 연산식 사용 가능

- 변수에 연산식의 값을 저장

```
int result = x + y;
```

- 다른 연산식의 피연산자 위치에 연산식 대입 가능

- x와 y를 더해서 5보다 작으면
- 참의 값을 result에 저장

```
boolean result = (x + y) < 5;
```



## 2. 연산의 방향과 우선순위

### ❖ 복합적으로 구성된 연산식의 연산

#### ■ 우선순위에 따라 수행

- : 단항 → 이항 → 삼항
- : 산술 → 비교 → 논리 → 대입

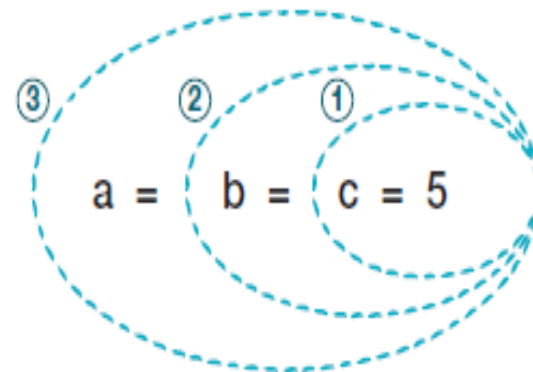
```
x > 0 && y < 0
```

#### ■ 우선순위 같은 연산자는 왼쪽에서 오른쪽 방향으로 수행

```
100 * 2 / 3 % 5
```

- 예외 : 대입 연산자

```
a = b = c = 5;
```



## 2. 연산의 방향과 우선순위

### ❖ chap3

```
1 package sec02.exam01;
2
3 public class SignOperatorExample {
4     public static void main(String[] args) {
5         int x = -100;
6         int result1 = +x;
7         int result2 = -x;
8         System.out.println("result1=" + result1);
9         System.out.println("result2=" + result2);
10
11         byte b = 100;
12         //byte result3 = -b; //컴파일 에러
13         int result3 = -b;
14         System.out.println("result3=" + result3);
15
16     }
17 }
```


## 2. 연산의 방향과 우선순위

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


## 2. 연산의 방향과 우선순위

- 괄호를 사용해 먼저 처리할 연산식 묶기

```
int var1 = 1;  
int var2 = 3;  
int var3 = 2;  
int result = var1 + var2 * var3;
```



```
int result = (var1 + var2) * var3;
```



### 3. 키워드로 끝내는 핵심 포인트

- **연산자**: 연산의 종류를 결정짓는 기호. 산술(+, -, \*, /, %), 증감(++, --), 비교(==, !=, ...), 논리(&&, ||, ...), 대입(=, +=, -=, ...) 등이 있다.
- **피연산자**: 연산식에서 연산되는 데이터(값). 예를 들어, 연산식  $3 + x$ 에서 3과 변수  $x$ 가 피연산자이다.
- **연산 방향**: 연산식에서 같은 종류의 연산자가 여러 개 사용될 경우 왼쪽에서 오른쪽으로 또는 오른쪽에서 왼쪽으로 연산되는 방향이 있다. 대부분의 연산자는 왼쪽에서 오른쪽으로 연산이 되지만, 증감(++, --)과 대입(=, +=, -=)은 오른쪽에서 왼쪽으로 연산된다.
- **연산 우선순위**: 서로 다른 연산자들이 복합적으로 구성되면 우선적으로 연산되는 연산자가 있다. 하지만 괄호 ()로 감싼 연산이 최우선순위를 갖기 때문에 복잡한 연산식에서 연산의 순서를 정하고 싶을 때에는 괄호 ()를 활용한다.

Chapter

# 03

## 연산자

### 03-2. 연산자의 종류

- 0. 시작하기 전에
- 1. 단항 연산자
- 2. 이항 연산자
- 3. 삼항 연산자
- 4. 키워드로 끝내는 핵심 포인트

## 0. 시작하기 전에

[핵심 키워드] : 증감 연산자, 비교 연산자, 논리 연산자, 대입 연산자, 삼항 연산자

[핵심 포인트]

- 피연산자 수에 따라 단항, 이항, 삼항 연산자로 구분하여 학습한다.

❖ 피연산자 수에 따라 단항, 이항, 삼항 연산자로 구분

- 단항 연산자: 부호, 증감 연산자
- 이항 연산자: 산술, 비교, 논리 연산자
- 삼항 연산자: 조건 연산자

단항 연산자:  $++x;$   
                   $\downarrow$  ← 피연산자의 수

이항 연산자:  $\frac{x}{1} + \frac{y}{1};$

삼항 연산자:  $\frac{(sum > 90)}{1} ? \frac{"A"}{1} : \frac{"B"}{1};$



# 1. 단항 연산자

## ❖ 부호 연산자

- boolean 타입과 char 타입을 제외한 기본 타입에 사용

연산식		설명
+	피연산자	피연산자의 부호 유지
-	피연산자	피연산자의 부호 변경

- 정수 및 실수 타입 변수 앞에 붙는 경우

- ```
int x = -100;
int result1 = +x;
int result2 = -x;
```

- 부호연산의 결과는 int

```
byte b = 100;
byte result = -b; //컴파일 에러 발생
```

# 1. 단항 연산자

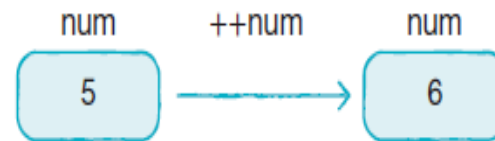
## ❖ 증감 연산자

- boolean 타입 외 모든 기본 타입 피연산자에 사용 가능

| 연산식  |      | 설명                             |
|------|------|--------------------------------|
| ++   | 피연산자 | 다른 연산을 수행하기 전에 피연산자의 값을 1 증가시킴 |
| --   | 피연산자 | 다른 연산을 수행하기 전에 피연산자의 값을 1 감소시킴 |
| 피연산자 | ++   | 다른 연산을 수행한 후에 피연산자의 값을 1 증가시킴  |
| 피연산자 | --   | 다른 연산을 수행한 후에 피연산자의 값을 1 감소시킴  |

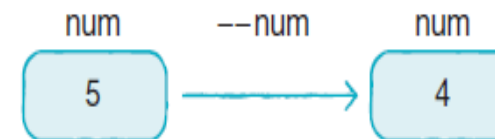
- 증가 연산자 (++)

- 피연산자 값에 1을 더하여 그 결과를 다시 피연산자에



- 감소 연산자(--)

- 피연산자 값에서 1 빼고 그 결과를 다시 피연산자에 저



# 1. 단항 연산자

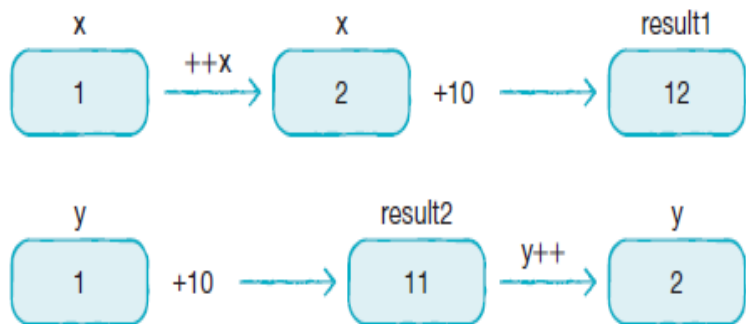
- 변수의 앞뒤 어디에든 올 수 있음

`++i;`  
`i++;` } 모두 `i=i+1`로 동일

`--i;`  
`i--;` } 모두 `i=i-1`로 동일

- 다른 연산자와 함께 사용될 경우 증감 연산자 위치에 따라 결과 달라질 수 있음에 주의

```
int x = 1;  
int y = 1;  
int result1 = ++x + 10;  
int result2 = y++ + 10;
```



# 1. 단항 연산자

## ❖ 증감연산자

```
1 package sec02.exam02;
2
3 public class IncreaseDecreaseOperatorExample
4 {
5     public static void main(String[] args) {
6         int x = 10;
7         int y = 10;
8         int z;
9         System.out.println("-----");
10        x++;
11        ++x;
12        System.out.println("x=" + x);
13
14        System.out.println("-----");
15        y--;
16        --y;
17        System.out.println("y=" + y);
```

```
18        System.out.println("-----");
19        z = x++;
20        System.out.println("z=" + z);
21        System.out.println("x=" + x);
22
23        System.out.println("-----");
24        z = ++x;
25        System.out.println("z=" + z);
26        System.out.println("x=" + x);
27
28        System.out.println("-----");
29        z = ++x + y++;
30        System.out.println("z=" + z);
31        System.out.println("x=" + x);
32        System.out.println("y=" + y);
33    }
34 }
```

# 1. 단항 연산자

## ❖ 논리 부정 연산자

- true를 false로, false를 true로 변경
  - 조건문과 제어문에서 조건식 값 부정하여 실행 흐름 제어
  - 토글 (toggle) 기능
- boolean 타입에만 사용 가능

| 연산식 |      | 설명                                                   |
|-----|------|------------------------------------------------------|
| !   | 피연산자 | 피연산자가 true이면 false 값을 산출<br>피연산자가 false이면 true 값을 산출 |

## 2. 연산의 방향과 우선순위

### ❖ 논리 연산자

```
1 package sec02.exam09;
2
3 public class LogicalOperatorExample {
4     public static void main(String[] args) {
5         int charCode = 'A';
6
7         if( (charCode>=65) & (charCode<=90)){
8             System.out.println("대문자 이군요");
9         }
10
11        if( (charCode>=97) && (charCode<=122)){
12            System.out.println("소문자 이군요");
13        }
14
15        if( !(charCode<48) && !(charCode>57)){
16            System.out.println("0~9 숫자 이군요");
17        }
```

```
19         int value = 6;
20
21         if( (value%2==0) | (value%3==0)){
22             System.out.println("2 또는 3의 배수 이군요");
23         }
24
25         if( (value%2==0) || (value%3==0)){
26             System.out.println("2 또는 3의 배수 이군요");
27         }
28     }
29 }
```

## 2. 이항 연산자

### ❖ 산술 연산자

| 연산식  |   |      | 설명                                |
|------|---|------|-----------------------------------|
| 피연산자 | + | 피연산자 | 덧셈 연산                             |
| 피연산자 | - | 피연산자 | 뺄셈 연산                             |
| 피연산자 | * | 피연산자 | 곱셈 연산                             |
| 피연산자 | / | 피연산자 | 왼쪽 피연산자를 오른쪽 피연산자로 나눴셈 연산         |
| 피연산자 | % | 피연산자 | 왼쪽 피연산자를 오른쪽 피연산자로 나눈 나머지를 구하는 연산 |

- 피연산자 타입이 동일하지 않을 경우 아래 규칙에 따라 일치시켜 연산 수행
  - 피연산자가 byte, short, char 타입일 경우 모두 int 타입으로 변환
  - 피연산자가 모두 정수 타입이고 long 타입 포함될 경우 모두 long 타입으로 변환
  - 피연산자 중 실수 타입이 있을 경우 허용 범위 큰 실수 타입으로 변환

## 2. 이항 연산자

### ❖ 산술 연산자

```
result1=7  
result2=3  
result3=10  
result4=2  
result5=1  
result6=2.5
```

```
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);
```



## 2. 이항 연산자

### ❖ 문자열 결합 연산자 (+)

- + 연산자의 피연산자 중 한 쪽이 문자열인 경우

```
String str1 = "JDK" + 6.0;  
String str2 = str1 + " 특징";
```

```
"JDK" + 3 + 3.0;
```

```
3 + 3.0 + "JDK";
```

## 2. 이항 연산자

### ❖ 문자열 결합 연산자

```
1 package sec02.exam06;
2
3 public class StringConcatExample {
4     public static void main(String[] args) {
5         String str1 = "JDK" + 6.0;
6         String str2 = str1 + " 특징";
7         System.out.println(str2);
8
9         String str3 = "JDK" + 3 + 3.0;
10        String str4 = 3 + 3.0 + "JDK";
11        System.out.println(str3);
12        System.out.println(str4);
13    }
14 }
```

## 2. 이항 연산자

### ❖ 비교 연산자

- 피연산자의 대소 비교하여 true/false 산출: 조건문이나 반복문에서 실행 흐름 제어
- 동등 비교 연산자는 모든 타입에 사용 가능
- 크기 비교 연산자는 boolean 외 모든 기본 타입에 사용 가능

| 구분       | 연산식   |    |       | 설명                 |
|----------|-------|----|-------|--------------------|
| 동등<br>비교 | 피연산자1 | == | 피연산자2 | 두 피연산자의 값이 같은지를 검사 |
|          | 피연산자1 | != | 피연산자2 | 두 피연산자의 값이 다른지를 검사 |
| 크기<br>비교 | 피연산자1 | >  | 피연산자2 | 피연산자1이 큰지를 검사      |
|          | 피연산자1 | >= | 피연산자2 | 피연산자1이 크거나 같은지를 검사 |
|          | 피연산자1 | <  | 피연산자2 | 피연산자1이 작은지를 검사     |
|          | 피연산자1 | <= | 피연산자2 | 피연산자1이 작거나 같은지를 검사 |

## 2. 이항 연산자

### ❖ 논리 연산자

- boolean 타입만 사용 가능

| 구분               | 연산식   |               | 결과    | 설명                                              |
|------------------|-------|---------------|-------|-------------------------------------------------|
| AND<br>(논리곱)     | true  | &&<br>또는<br>& | true  | 피연산자 모두가 true일 경우에만 연산 결과가 true                 |
|                  | true  |               | false |                                                 |
|                  | false |               | true  |                                                 |
|                  | false |               | false |                                                 |
| OR<br>(논리합)      | true  | <br>또는<br>    | true  | 피연산자 중 하나만 true이면 연산 결과는 true                   |
|                  | true  |               | false |                                                 |
|                  | false |               | true  |                                                 |
|                  | false |               | false |                                                 |
| XOR<br>(배타적 논리합) | true  | ^             | true  | 피연산자가 하나는 true이고 다른 하나가 false일 경우에만 연산 결과가 true |
|                  | true  |               | false |                                                 |
|                  | false |               | true  |                                                 |
|                  | false |               | false |                                                 |
| NOT<br>(논리 부정)   |       | !             | true  | 피연산자의 논리값을 바꿈                                   |
|                  |       |               | false |                                                 |

## 2. 이항 연산자

### ❖ 대입 연산자

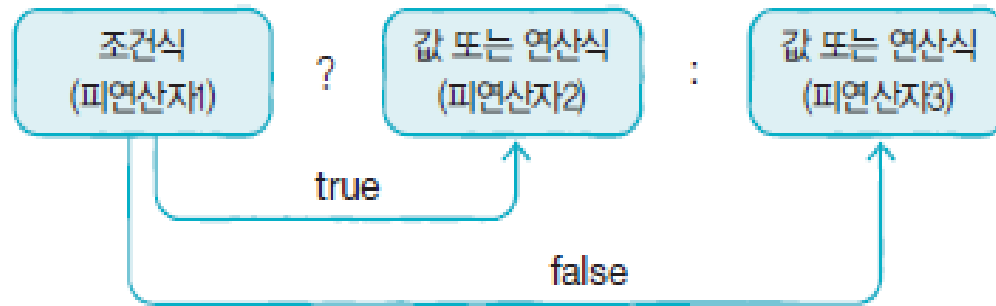
- 오른쪽 피연산자의 값을 왼쪽 피연산자인 변수에 저장

| 구분        | 연산식 |    |      | 설명                      |
|-----------|-----|----|------|-------------------------|
| 단순 대입 연산자 | 변수  | =  | 피연산자 | 오른쪽의 피연산자의 값을 왼쪽 변수에 저장 |
| 복합 대입 연산자 | 변수  | += | 피연산자 | 변수=변수+피연산자와 동일          |
|           | 변수  | -= | 피연산자 | 변수=변수-피연산자와 동일          |
|           | 변수  | *= | 피연산자 | 변수=변수*피연산자와 동일          |
|           | 변수  | /= | 피연산자 | 변수=변수/피연산자와 동일          |
|           | 변수  | %= | 피연산자 | 변수=변수%피연산자와 동일          |
|           | 변수  | &= | 피연산자 | 변수=변수&피연산자와 동일          |
|           | 변수  | =  | 피연산자 | 변수=변수 피연산자와 동일          |
|           | 변수  | ^= | 피연산자 | 변수=변수^피연산자와 동일          |

### 3. 삼항 연산자

#### ❖ 삼항 연산자

- 3개의 피연산자를 필요로 하는 연산자
- ? 앞의 조건식에 따라 콜론 앞뒤의 피연산자 선택



```
int score = 95;  
char grade = (score > 90) ? 'A' : 'B';
```

=

```
int score = 95;  
char grade;  
if(score > 90) {  
    grade = 'A';  
} else {  
    grade = 'B';  
}
```

### 3. 삼항 연산자

#### ❖ 삼항 연산자

```
1 package sec02.exam11;
2
3 public class ConditionalOperationExample {
4     public static void main(String[] args) {
5         int score = 85;
6         char grade = (score > 90) ? 'A' : ((score > 80) ? 'B' : 'C');
7         System.out.println(score + "점은 " + grade + "등급입니다.");
8     }
9 }
10
```

## 4. 키워드로 끝내는 핵심 포인트

- **증감 연산자**: ++, --를 말하며 변수의 값을 1씩 증가, 1씩 감소시킴
- **비교 연산자**: ==, != 등을 말하며 값이 같은지, 다른지를 비교하고 boolean 값을 산출
- **논리 연산자**: &&, ||, ! 등을 말하며 논리곱, 논리합, 논리 부정을 수행하고 boolean 값을 산출
- **대입 연산자**: =, +=, -= 등을 말하며 오른쪽의 값을 왼쪽에 대입하거나 연산 후 대입
- **삼항 연산자**: (조건식) ? A : B를 말하며 조건이 true이면 A를 산출하고, false이면 B를 산출





Thank You!