

03 연산자와 조건

Sangwook Lee

03 연산자와 조건

1. 연산자 개요

2. 다양한 연산자

3. 조건문

1. 연산자 개요

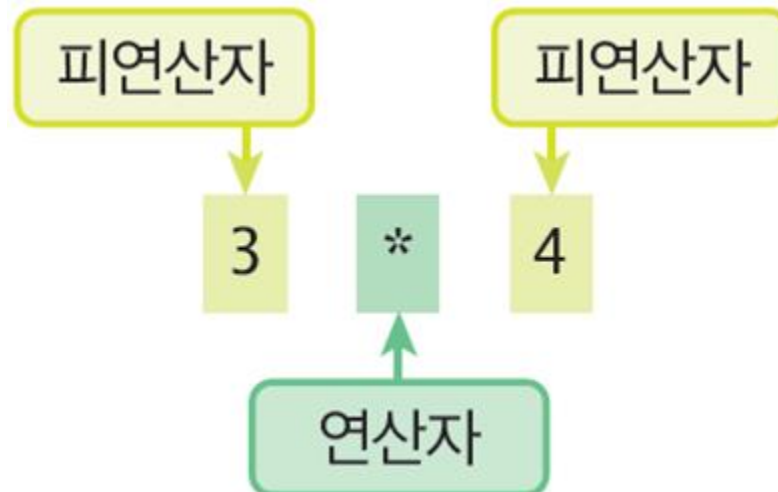
- 표현식과 연산자 종류

- 연산자(operator)

- 연산을 수행하는 문자 또는 문자 조합

- 피연산자(operand)

- 연산에 참여하는 변수나 상수, 또는 표현식

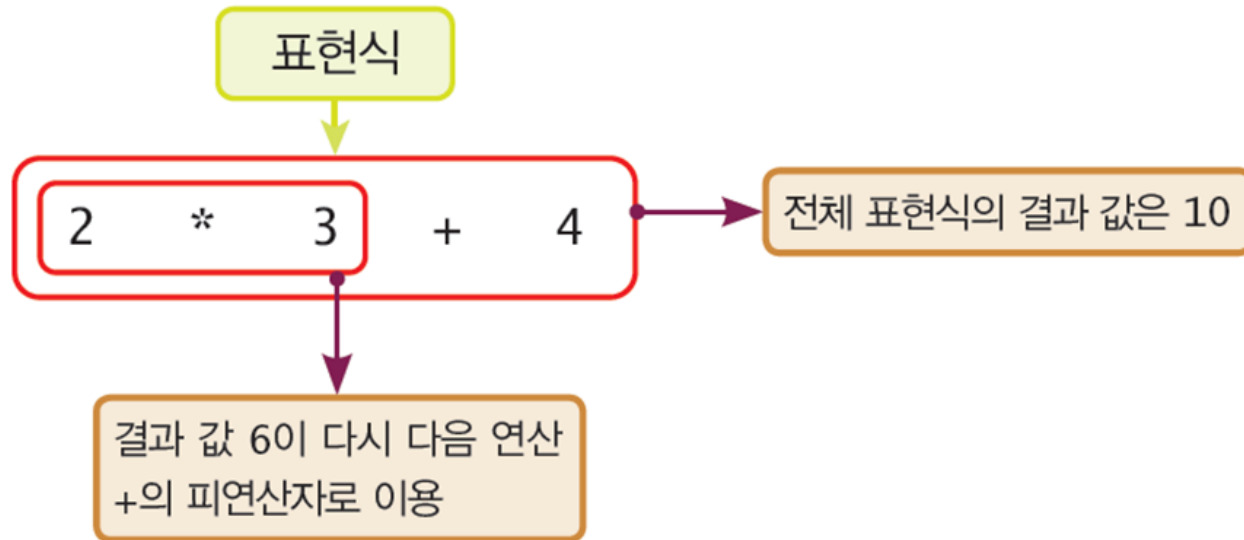


1. 연산자 개요

- 표현식과 연산자 종류

- 표현식(expression)

- 연산자와 피연산자로 구성된 연산식
- 항상 하나의 결과 값을 가짐



1. 연산자 개요

- 표현식과 연산자 종류

- 연산자 종류

- 단항 연산자(unary operator)

- ❖ 전위(prefix) 단항 연산자

- » ++a

- ❖ 후위(postfix) 단항 연산자

- » a++

- 이항 연산자(binary operator)

- 삼항 연산자(ternary operator)

1. 연산자 개요

- 표현식과 연산자 종류
 - 연산자 우선 순위

우선순위	연산자 이름	연산자	연산방향
1	후위 단항	var++ var--	왼쪽에서 오른쪽으로 →
2	전위 단항	++var --var +expr -expr ~ !	
3	곱셈 부류	* / %	
4	덧셈 부류	+ -	
5	비트 이동	<< >> >>>	
6	관계	< > <= >= instanceof	
7	동등	== !=	
8	비트 AND	&	
9	비트 배타적 OR	^	
10	비트 OR		
11	논리 AND	&&	
12	논리 OR		
13	조건 삼항	expr ? x : y	
14	대입	= += -= *= /= %= &= ^= = <<= >>= >>>=	←

03 연산자와 조건

1. 연산자 개요
2. 다양한 연산자
3. 조건문

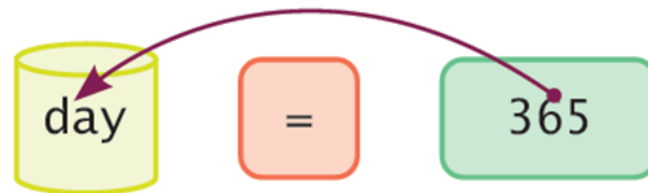
2. 다양한 연산자

- 대입 연산자와 산술 연산자

- 대입 연산자(assignment operator)

- = 할당 연산자 = 치환 연산자
- 오른쪽 값을 왼쪽 변수에 저장하는 연산자
- 왼쪽은 반드시 값을 저장할 수 있는 변수

`int day = 365;`



2. 다양한 연산자

- 대입 연산자와 산술 연산자

- 산술 연산자

- 더하기(additive operator)
- 곱하기(multiplication operator)
- 빼기(subtraction operator)
- 나누기(division operator)
- 나머지 연산자(remainder operator)

구분	연산자	연산자 의미	예	결과	비고
산술 연산자	+	더하기	3 + 7	10	문자열 연결 연산자 "java" + "lang"
	-	빼기	7 - 3	4	
	*	곱하기	7 * 3	21	별표 * 기호 ^{asterisk}
	/	나누기	7 / 2	3	정수와의 나누기는 결과도 정수
	%	나머지	7 % 2	1	백분율이 아님

2. 다양한 연산자

- 대입 연산자와 산술 연산자

- 축약 대입 연산자

- = 복합 대입 연산자(compound assignment operator)
 - 산술 연산자 + 대입 연산자

연산자	연산 예	연산자 의미
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x \% = y$	$x = x \% y$

2. 다양한 연산자

[예제 3-1] 대입, 산출, 축약 대입 연산자 이용 (p.75)

2. 다양한 연산자

- 증감 연산자와 조건 연산자
 - 증가 연산자와 감소 연산자

연산자 위치	연산자	연산 결과 값	연산 후 n의 값
전위	$++n$	$n+1$ (이전 n 에서 1 증가된 값)	1 증가
	$--n$	$n-1$ (이전 n 에서 1 감소된 값)	1 감소
후위	$n++$	n (그대로 이전 n 값)	1 증가
	$n--$	n (그대로 이전 n 값)	1 감소

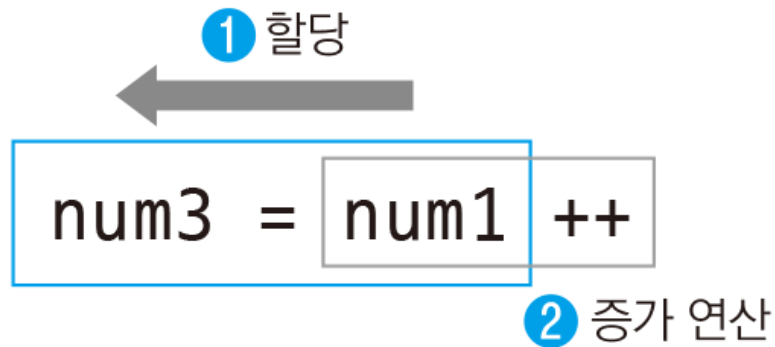


여기서 말하는 결과 값은...
속한 문장(위치)에서의 반환 값

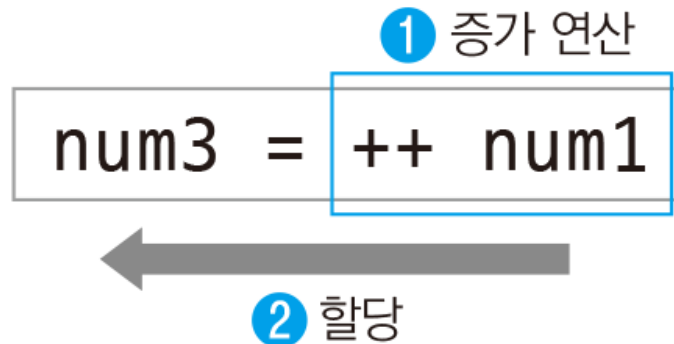
2. 다양한 연산자

- 증감 연산자와 조건 연산자

- 후위 증가 연산자와 반환 값



- 전위 증가 연산자와 반환 값



2. 다양한 연산자

[예제]

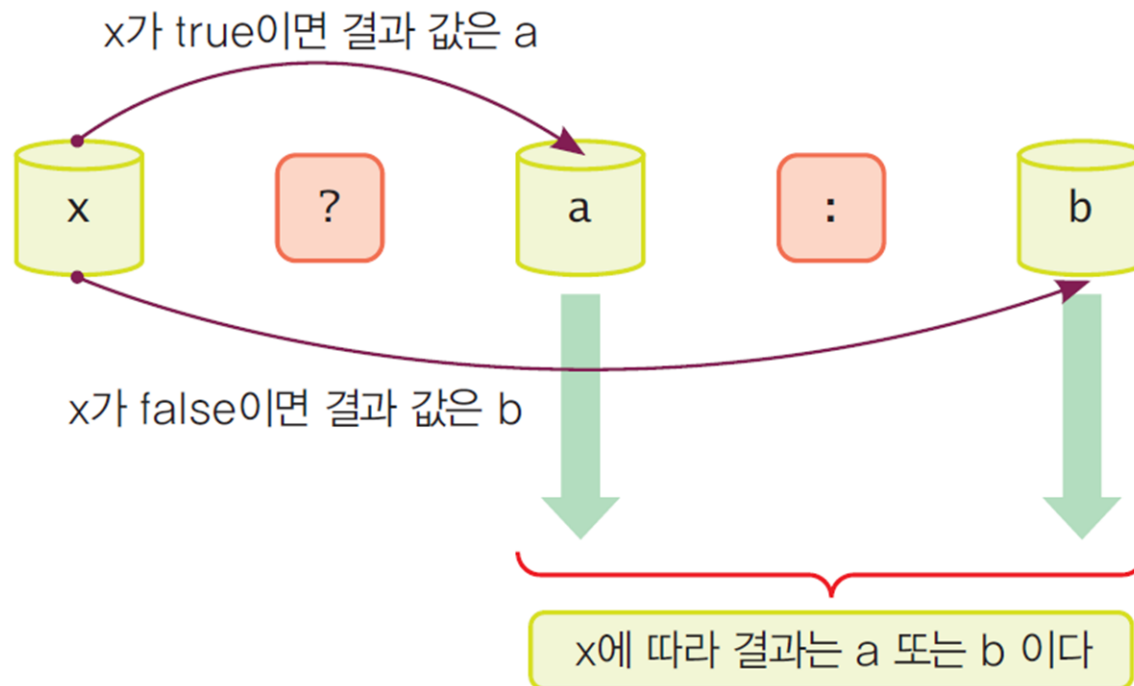
```
public class ex {  
  
    public static void main(String[] args) {  
        int num1 = 10;  
        int num2 = (num1--) + 2 + (num1--);  
  
        System.out.printf("num1: %d %n", num1);  
        System.out.printf("num2: %d %n", num2);  
  
    }  
  
}
```

— 실행결과

```
num1: 8  
num2: 21
```

2. 다양한 연산자

- 증감 연산자와 조건 연산자
 - 조건 연산자(conditional operator)
 - 유일한 삼항 연산자(ternary operator)



2. 다양한 연산자

- 증감 연산자와 조건 연산자
 - 조건 연산자(conditional operator)
 - 예

```
String str = (point%2 == 0) ? "짝수" : "홀수" ;
```

```
double max = (x > y) ? x : y ;
```

2. 다양한 연산자

- 관계 연산자와 논리 연산자

- 관계 연산자(relational operators)

- 2개의 피연산자 크기 비교
- 피연산자: 정수, 실수, 문자
- 결과: boolean (true 또는 false)
- 종류 및 연산 예

연산자	연산 예	비고	연산 결과
>	5 > 3		true
>=	3.0 >= 4.0 - 1.0	3.0 >= (4.0 - 1.0)	true
<	3 * 2 < 4	(3 * 2) < 4	false
<=	3 <= 3		true
==	'A' == 'a'	대문자와 소문자는 다름	false
!=	3 % 2 != 2	(3 % 2) != 2	false

2. 다양한 연산자

- 관계 연산자와 논리 연산자

- 논리 연산자

- 피연산자: boolean
- 결과: boolean
- 종류 및 연산 예

연산자	이름	연산 예	결과	연산자 의미
&&	AND	(3 > 4) && (3 == 4)	false	피연산자가 모두 true인 경우만 true, 아니면 false
	OR	(3 <= 4) (3 == 4)	true	피연산자가 모두 false인 경우만 false, 아니면 true
^	XOR	(3 >= 4) ^ (3 < 4)	true	피연산자가 서로 다르면 true, 같으면 false
!	NOT	! (3/2 == 1)	false	!x에서 x의 논리 값과 반대

논리 연산자 &&와 ||는...

피연산자 두 개 중에서 왼쪽 피연산자만으로 결과가 결정되면
오른쪽 피연산자는 평가하지 않음

2. 다양한 연산자

- 비트 연산자

- 비트 논리 연산자

- 비트 단위로 논리 연산 수행
 - 피연산자: 정수(byte, short, int, long), 문자(char)
 - 결과: 정수(int)

- 비트 보수 연산자(unary bitwise complement)

- 각 비트를 0은 1로, 1은 0으로 바꿈

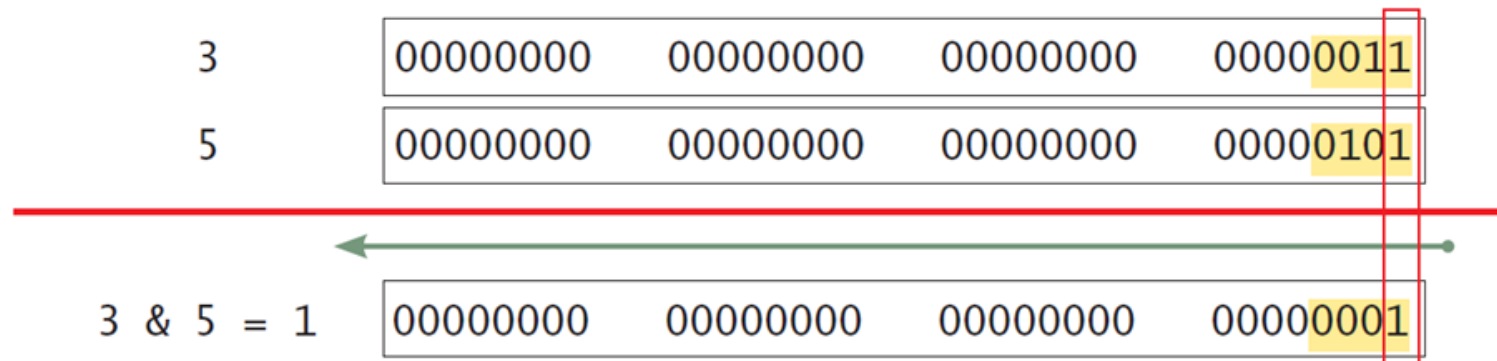
2. 다양한 연산자

- 비트 연산자

- 비트 논리 및 보수 연산자 종류

연산자	연산자 이름	사용	의미
&	비트 AND	op1 & op2	비트가 모두 1이면 결과는 1, 아니면 0
	비트 OR	op1 op2	비트가 적어도 하나 1이면 결과는 1, 아니면 0
^	비트 배타적 OR(XOR)	op1 ^ op2	비트가 서로 다르면 결과는 1, 같으면 0
~	보수(complement)	~op1	비트가 0이면 결과는 1, 1이면 0

- 비트 논리 연산 예



2. 다양한 연산자

- 음수의 표현 방법

1. 부호화 절댓값(sign/magnitude)

- 양수의 MSB(Most Significant Bit)를 1로 바꿈
- $-2^{N-1}+1 \sim 2^{N-1}-1$

2. 1의 보수(one's complement)

- $\sim$ 양수
- $-2^{N-1}+1 \sim 2^{N-1}-1$

3. 2의 보수(two's complement)

- $\sim$ 양수+1
- $-2^{N-1} \sim 2^{N-1}-1$

2. 다양한 연산자

- 음수의 표현 방법
 - 4비트 정수의 음수 표현

[sign/magnitude]

-8	-7	-6	-5	-4	-3	-2	-1	-0	0	1	2	3	4	5	6	7
	1111	1110	1101	1100	1011	1010	1001	1000	0000	0001	0010	0011	0100	0101	0110	0111

[one's complement]

-8	-7	-6	-5	-4	-3	-2	-1	-0	0	1	2	3	4	5	6	7
	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

[two's complement]

-8	-7	-6	-5	-4	-3	-2	-1	-0	0	1	2	3	4	5	6	7
1000	1001	1010	1011	1100	1101	1110	1111		0000	0001	0010	0011	0100	0101	0110	0111

2. 다양한 연산자

- 비트 연산자

- Quiz

1 = 00000000_00000000_00000000_00000001

$\sim 1 + 1$ = 11111111_11111111_11111111_11111111

$x \& -1$ = x

$x | 0$ = x

2. 다양한 연산자

- 비트 연산자

- 비트 이동 연산자(bit shift operator)

- 비트 단위로 왼쪽이나 오른쪽으로 이동
 - 종류

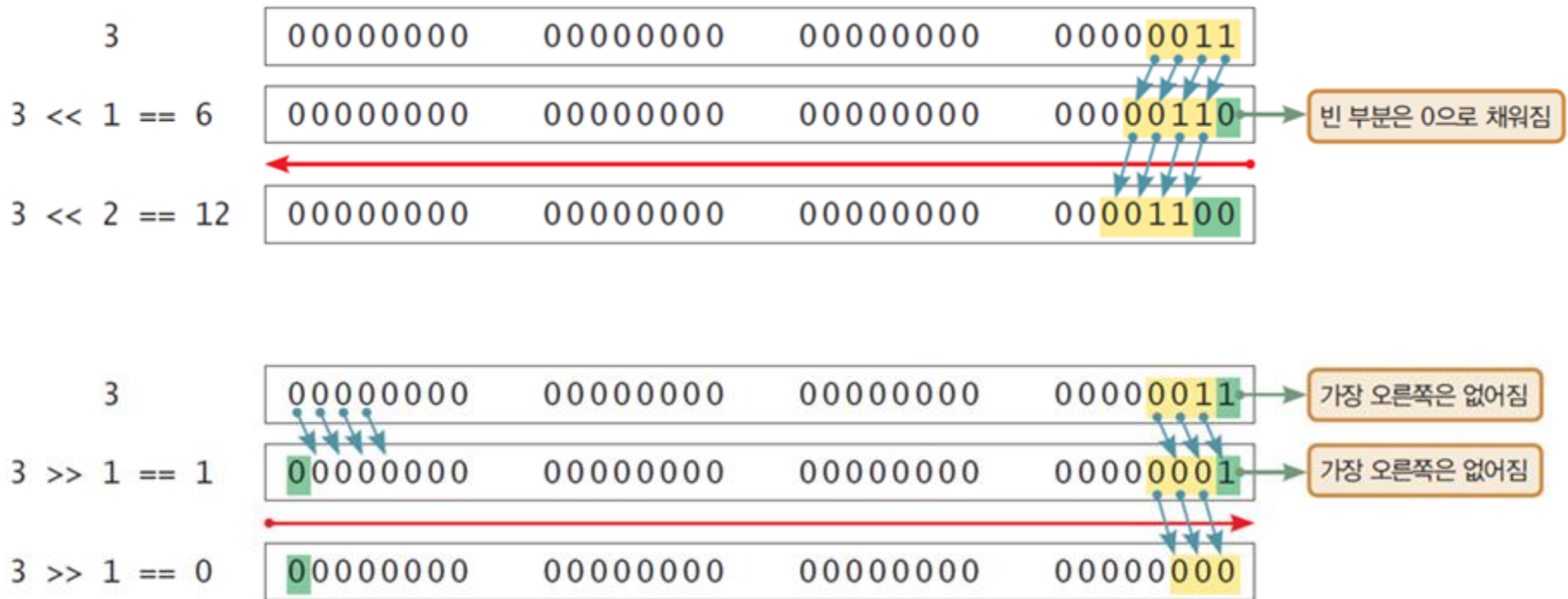
연산자	이름	사용	연산 방법	새로 채워지는 비트
>>	Signed left shift	op1 >> op2	op1을 오른쪽으로 op2 비트만큼 이동	가장 왼쪽 비트인 부호 비트는 원래의 비트로
<<	Signed right shift	op1 << op2	op1을 왼쪽으로 op2 비트만큼 이동	가장 오른쪽 비트를 모두 0으로 채움
>>>	Unsigned right shift	op1 >>> op2	op1을 오른쪽으로 op2 비트만큼 이동	가장 왼쪽 비트인 부호 비트는 모두 0으로 채워짐

2. 다양한 연산자

- 비트 연산자

- 비트 이동 연산자(bit shift operator)

- 예



2. 다양한 연산자

- 비트 연산자
 - Quiz

임의의 정수 x 의
오른쪽에서 n 번째 비트를 표현하는 수식은?

$$x \gg (n-1) \& 1$$

2. 다양한 연산자

[예제] $0.1 \times 0.25 = 0.025$?

```
float a, b, c;  
  
a = (float) 0.1;  
b = (float) 0.25;  
c = a * b;  
  
if (c == 0.025)  
    System.out.println("c is equal to 0.025");  
else  
    System.out.println("c is not equal to 0.025");
```

— 실행결과

```
c is not equal to 0.025
```

2. 다양한 연산자

- 실수의 표현

- IEEE 754 부동소수점 표현 방식

1. 10진수를 2진수로
2. 정규화된 **가수** $\times 2^{\text{지수}}$ 형태로 표현
3. 부호(sign), 지수(exponent), 가수(mantissa) 저장

- ※ 정규화

- 가수의 첫째 자리가 밑수보다 작은 한자리 자연수가 되도록 바꾸는 것
 - $345.67 = 345.67 \times 10^0 = 3.4567 \times 10^2$
 - $0.00101_{(2)} = 0.00101 \times 2^0 = 1.01 \times 2^{-3}$

2. 다양한 연산자

- 실수의 표현

– 0.25의 표현

$$0.25 = 2^{-2} = 0.01_{(2)} = 1.0_{(2)} \times 2^{-2}$$

❖ 부호: 0

❖ 지수: $-2 + 127 = 125 = 01111101_{(2)}$

❖ 가수: $000000000...000000000_{(2)}$

가수의 첫 숫자 1은 어디에?

2진수의 경우 정규화 후 가수의 첫 숫자는 항상 1이기 때문에 저장할 필요가 없음

0	01111101	0000000000000000000000000
---	----------	---------------------------

부호
(1-bit)

지수
(8-bit)

가수
(23-bit)

가장 작은 지수(-127)를 00000000,
가장 큰 지수(+128)를 11111111로 표현하기 위해
지수에 127을 더함

2. 다양한 연산자

- 실수의 표현

- 0.1의 표현

$$0.1 - 0.0625 = 0.0375$$

$$\rightarrow 0.0625 = 2^{-4} = 0.0001_{(2)}$$

$$0.0375 - 0.03125 = 0.00625$$

$$\rightarrow 0.03125 = 2^{-5} = 0.00001_{(2)}$$

$$0.00625 - 0.00390625 = 0.00234375$$

$$\rightarrow 0.00390625 = 2^{-8} = 0.00000001_{(2)}$$

.....

$$0.1 = 0.000110011..._{(2)} = 1.10011..._{(2)} \times 2^{-4}$$

❖ 부호: 0

❖ 지수: $-4 + 127 = 123 = 01111011_{(2)}$

❖ 가수: $10011..._{(2)}$

부동소수점 표현 방식은...

아주 큰 수와 아주 작은 수를 효율적(?)으로 나타낼 수 있지만,
정확하게 표현하지 못하는 수도 있음

03 연산자와 조건

1. 연산자 개요
2. 다양한 연산자
3. 조건문

3. 조건문

프로그램을 작성하다 보면,
어떤 조건에 따라 문장을 선택을 하거나
어떤 조건에 따라 문장을 반복해야 되는
상황이 발생하는데...

이때 제어문(선택문, 반복문)을 사용하면
프로그램의 진행 순서를 변경할 수 있음

3. 조건문

- 제어문

- 프로그램을 위에서 아래로 순서대로 실행하지 않고, 특정 문장을 선택하여 실행하거나, 특정 문장을 반복적으로 실행하도록 명령하는 문장
- 종류
 - 선택문: if, switch
 - 반복문: for, while
 - 분기문: break, continue

3. 조건문

- **if**

- 조건이 참인 경우에만 문장을 선택

```
if (cond) {  
    stmt1;  
    stmt2;  
}
```

```
int degree = 30;
```

```
if (30 <= degree)
```

```
System.out.println("날씨가 덥습니다.");
```

조건식은 괄호가 필요하며 결과는 **boolean** 값이어야 한다.

이 문장은 블록으로 구성될 수 있으며, 블록인 경우 블록의 모든 문장이 실행된다.

3. 조건문

- if - else

- 조건이 참인 경우와 거짓인 경우 다른 문장을 선택

```
if (cond) {  
    stmt1;  
    stmt2;  
} else {  
    stmt3;  
    stmt4;  
}
```

```
public static void main(String[] args) {  
    int score = 65;  
  
    if (60 <= score)  
        System.out.println("합격입니다.");  
    else  
        System.out.println("불합격입니다.");  
}
```

3. 조건문

- if – else if

– 어떤 조건이 참인 경우와, 그 조건이 거짓이면서 다른 조건이 참인 경우 다른 문장을 선택

```
if (cond1) {  
    stmt1;  
    stmt2;  
} else if (cond2) {  
    stmt3;  
    stmt4;  
}
```

```
if (90 <= point)  
    grade = 'A';  
else if (80 <= point)  
    grade = 'B';  
...  
else  
    grade = 'F';
```

이 문장이 실행되는 조건은 else if (80 <= point)이므로 실제 (80 <= point && point < 90)이 된다.

3. 조건문

- Nested if

- 2개 이상의 조건이 모두 참인 경우와 1개의 조건만 참인 경우 다른 문장을 선택

```
if (cond1) {  
    if (cond2) {  
        stmt1;  
        stmt2;  
    }  
    else {  
        stmt3;  
        stmt4;  
    }  
}
```

```
if (type == 1) {
```

```
    if (score >= 70)
```

```
        System.out.println("1종 면허 시험 합격");
```

```
    else
```

```
        System.out.println("1종 면허 시험 불합격");
```

```
} else if (type == 2) {
```

```
    if (score >= 60)
```

외부 if에서 먼저 종류를 먼저 구분한다.

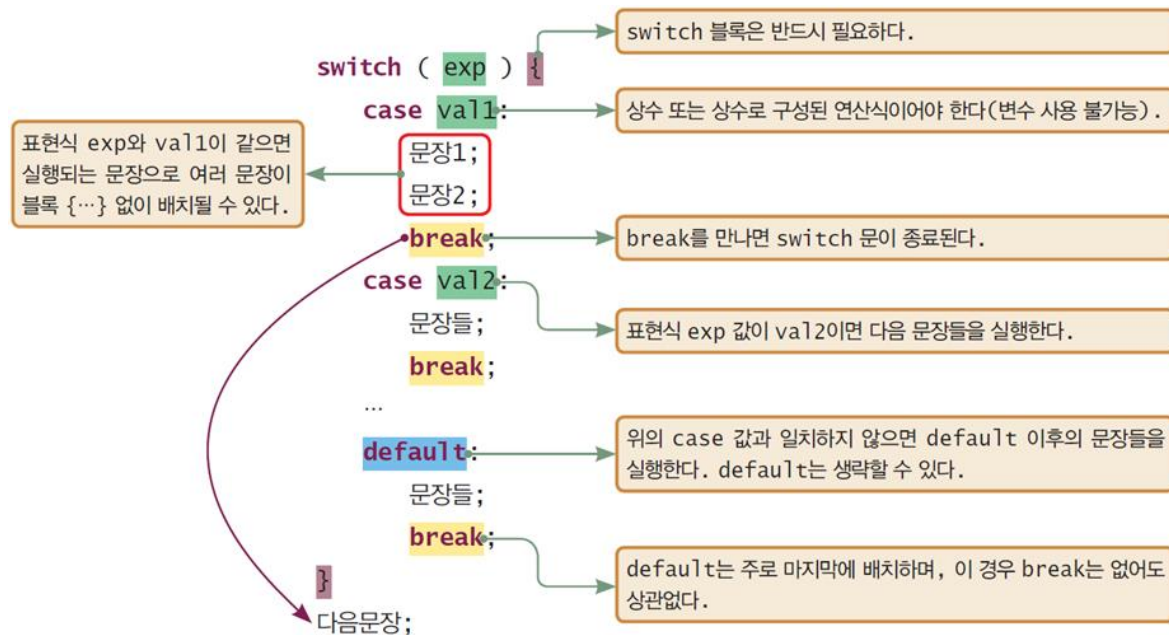
내부에서 점수로 합격 판정을 수행한다.

3. 조건문

- switch

- 연산 결과에 따라 다른 문장 선택
- 연산 결과와 case 값이 일치하는 내부 문장 실행
- 허용 연산 결과 값: byte, short, char, int, String 등

연산 결과 값으로 float와 double은 지원하지 않음



3. 조건문

- [예제 3-17] switch 문을 이용한 두 실수의 사칙 연산

```
double x, y;
int op;
java.util.Scanner input = new java.util.Scanner(System.in);

System.out.print("두 실수 입력: ");
x = input.nextDouble();
y = input.nextDouble();
System.out.print("번호선택 1(+), 2(-), 3(*), 4(/): ");
op = input.nextInt();

switch (op) {
    case 1:
        System.out.printf("%.2f + %.2f = %.2f\n", x, y, x + y);
        break;
    case 2:
        System.out.printf("%.2f - %.2f = %.2f\n", x, y, x - y);
        break;
    case 3:
        System.out.printf("%.2f * %.2f = %.2f\n", x, y, x * y);
        break;
    case 4:
        System.out.printf("%.2f / %.2f = %.2f\n", x, y, x / y);
        break;
    default:
        System.err.printf("연산자 번호를 잘못 입력했습니다.\n");
}
}
```

Exercise

- 다음 프로그램의 오류를 찾아 수정하십시오.

```
int x = 20;
switch (x) {
    case x > 0:
        System.out.printf("양수\n");
        break;
    case x < 0:
        System.out.printf("음수\n");
}
```



case 값으로 boolean 사용 불가!

```
int x = 20;
switch (x > 0 ? 1 : -1) {
    case 1:
        System.out.printf("양수\n");
        break;
    case -1:
        System.out.printf("음수 또는 0\n");
}
```

Exercise

- 세 수를 입력 받아 가장 큰 수를 구하는 프로그램을 조건 연산자를 이용하여 작성하시오.

– 코드

```
import java.util.Scanner;

public class Ex03-1 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.printf("실수1 입력: ");
        double x = s.nextDouble();
        System.out.printf("실수2 입력: ");
        double y = s.nextDouble();
        System.out.printf("실수3 입력: ");
        double z = s.nextDouble();

        System.out.printf("입력한 실수 중에서 가장 큰 수는 %f입니다.\n",
            x > y ? (x > z ? x : z) : (y > z ? y : z));
    }
}
```

Exercise

- 지불할 금액을 정수로 입력 받아 화폐단위가 각각 몇 개씩 필요한지 출력하는 프로그램을 작성하시오.

— 조건

- 입력은 최소 천원 단위로 입력
- 화폐단위는 50000, 10000, 5000, 1000 4가지이며, 가능한 큰 화폐단위로 지불

— 실행 예

- 입력이 236,000이면...
 - ❖ 50000원권 4개
 - ❖ 10000원권 3개
 - ❖ 5000원권 1개
 - ❖ 1000원권 1개

Exercise

- 지불할 금액을 정수로 입력 받아 화폐단위가 각각 몇 개씩 필요한지 출력하는 프로그램을 작성하시오.

– 코드

```
import java.util.Scanner;

public class Ex03-2 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.printf("금액 입력(천원 단위 이상) >> ");
        int x = s.nextInt();

        System.out.printf("입력한 금액: %d\n", x);
        System.out.printf("오만원권 %d개\n", x / 50000);
        System.out.printf("만원권 %d개\n", x % 50000 / 10000);
        System.out.printf("오천원권 %d개\n", x % 10000 / 5000);
        System.out.printf("천원권 %d개\n", x % 5000 / 1000);
    }
}
```