



JAVA

GU, DA HAE

산 술 연 산 자

대 입 연 산 자

복 합 대 입 연 산 자

증 감 연 산 자

관 계 연 산 자

논 리 연 산 자

조 건 연 산 자

JAVA

연 산 자

산술 연산자

기능	연산자	예
더하기	+	$A + B$
빼기	-	$A - B$
곱하기	*	$A * B$
나누기	/	A / B
모듈러 (Modulus) 나눈 후 나머지	%	$A \% B$

나누기의 제수는 0이 될 수 없다.

모듈러는 배수 검사할 때 주로 사용한다.

예제

```
public class Ex01 {  
    public static void main(String[] args) {  
        float n1, n2;  
        int n3, n4;  
  
        n1 = 1.0f + 0.456789f; // 1.0에 0.456789를 더해서 n1에 저장  
        n2 = 1.0f - 0.456789f; // 1.0에서 0.456789를 빼서 n2에 저장  
  
        System.out.println(n1); // 1.456789  
        System.out.println(n2); // 0.543211  
  
        n3 = 2 * 3; // 2에 3를 곱해서 n1에 저장  
        n4 = 7 / 2; // 7에서 2를 나누어서 n2에 저장  
  
        System.out.println(n3); // 6  
        System.out.println(n4); // 3: 소수점을 사용하지 않고 최대한 나눌 수 있는 값이 3  
    }  
}
```

예제

```
public class Ex02 {  
    public static void main(String[] args) {  
        System.out.println(1 % 3); // 1: 1을 3으로 나누면 몫은 0 나머지는 1  
        System.out.println(2 % 3); // 2: 2를 3으로 나누면 몫은 0 나머지는 2  
        System.out.println(3 % 3); // 0: 3을 3으로 나누면 몫은 1 나머지는 0  
        System.out.println(4 % 3); // 1: 4를 3으로 나누면 몫은 1 나머지는 1  
        System.out.println(5 % 3); // 2: 5를 3으로 나누면 몫은 1 나머지는 2  
        System.out.println(6 % 3); // 0: 6을 3으로 나누면 몫은 2 나머지는 0  
    }  
}
```

예제

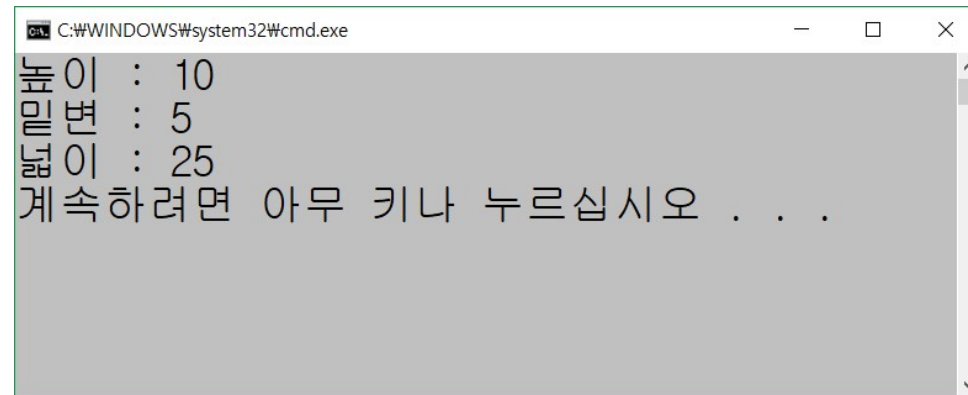
```
public class Ex03 {  
    public static void main(String[] args) {  
        System.out.println("정수형 나눗셈 : " + 7/3); // 2 : 나머지는 버려지고 몫만 출력  
        System.out.println("실수형 나눗셈 : " + 7.0f/3.0f); // 2.3333333 : 모두 float형  
        데이터이므로 실수형 나눗셈이 진행되어 나머지 없이 2.3333333이 출력된다.  
        System.out.println("실수형 나눗셈 : " + (float)7/3); // 2.3333333 : 7이 float  
        형으로 변환된다. 두 피연산자의 자료형이 일치하지 않으므로 형 변환 규칙에 의해 3이 3.0f로  
        자동 변환된다. 결국 실수형 나눗셈이 진행된다.  
    }  
}
```

예제

```
public class Ex04 {  
    public static void main(String[] args) {  
        System.out.println("정수형 나머지 : " + 7%3); // 1  
        System.out.println("실수형 나머지 : " + 7.0f % 2.0f); // ? : 실수형 데이터로 나머  
        지 연산을 하는 것은 수학적으로 문제가 있다. 에러는 발생하지 않지만 실수형 나눗셈은 존재하  
        지 않는다.  
    }  
}
```

해보기

삼각형의 높이와 밑변을 변수에 저장하여 넓이를 계산하자. 저장한 높이, 밑변, 넓이를 출력해보자.



```
C:\WINDOWS\system32\cmd.exe
높이 : 10
밑변 : 5
넓이 : 25
계속하려면 아무 키나 누르십시오 . . .
```


대입 연산자

기능	연산자	예
대입(할당)	=	<pre>int a = 10; a = 20;</pre>

복합 대입 연산자

| 산술 연산 후, 변수에 할당 연산을 하는 기호

기능	연산자	예	기타
더하기 후 대입	<code>+=</code>	<code>A += B</code>	<code>A = A + B</code> 와 동일
빼기 후 대입	<code>-=</code>	<code>A -= B</code>	<code>A = A - B</code> 와 동일
곱하기 후 대입	<code>*=</code>	<code>A *= B</code>	<code>A = A * B</code> 와 동일
나누기 후 대입	<code>/=</code>	<code>A /= B</code>	<code>A = A / B</code> 와 동일
모듈러 후 대입	<code>%=</code>	<code>A %= B</code>	<code>A = A % B</code> 와 동일

복합 대입 연산자는 산술 연산자와 대입 연산자를 사용하여 만든 연산자.

산술 연산 후 대입 연산을 진행한다.

왼쪽 피연산자는 반드시 변수여야 한다.

J A V A

예제

```
public class Ex05 {  
    public static void main(String[] args) {  
        double e = 3.1;  
        e += 2.1;  
        e *= 2;  
  
        int n = 5;  
        n *= 2.2;  
  
        System.out.println(e);  
        System.out.println(n);  
    }  
}
```

해보기

다음 소스코드를 완성하여 0을 두 번 출력해보자.

```
public class Quest {  
    public static void main(String[] args) {  
        int n1 = 15;  
        int n2 = 27;  
        int n3 = __;  
  
        n1 %= n3;  
        n2 %= n3;  
  
        System.out.println(n1);  
        System.out.println(n2);  
    }  
}
```

증감 연산자

| 변수를 1증가, 감소 연산하는 기호

기능	연산자	예
1증가	++	++A
		A++
1감소	--	--A
		A--

증감 연산자는 변수 하나의 값을 1증가 하거나 1감소 할 수 있다.

전치(prefix)는 **1증가를 먼저** 한 후 다른 연산을 수행한다.

후치(postfix)는 **다른 연산을 먼저** 한 후 1증가를 수행한다.

J A V A

예제

```
public class Ex06 {  
    public static void main(String[] args) {  
        int n1 = 2;  
        int n2 = 2;  
        int n3;  
        int n4;  
  
        n3 = n1++; // n1의 값을 n3에 할당한 뒤 n1의 값을 1 증가시킴  
        n4 = n2--; // n2의 값을 n4에 할당한 뒤 n2의 값을 1 감소시킴  
  
        System.out.println(n3);  
        System.out.println(n4);  
    }  
}
```

예제

```
public class Ex07 {  
    public static void main(String[] args) {  
        int n1 = 2;  
        int n2 = 2;  
  
        System.out.println(n1++ + " " + n2--); // 2 2: n1, n2의 값을 먼저 출력한 뒤 증감 연산자 동작  
        System.out.println(n1 + " " + n2); // 3 1: 증감 연산자가 동작한 결과  
  
        n1 = n2 = 2;  
  
        System.out.println(++n1 + " " + --n2); // 3 1: 증감 연산자가 먼저 동작한 뒤 n1, n2의 값을 출력  
        System.out.println(n1 + " " + n2); // 3 1: 앞과 같은 값이 출력됨  
    }  
}
```

해보기

다음 소스 코드를 완성하여 2와 7을 출력해보자.

```
public class Quest {  
    public static void main(String[] args) {  
        int num1 = ①_____;  
        int num2 = ②_____;  
        int num3;  
        int num4;  
        num1++;  
        num3 = --num1;  
        --num2;  
        num4 = num2++;  
        System.out.println(num3);  
        System.out.println(num4);  
    }  
}
```

Handwritten red annotations: 2, 7, 3, 8, 7, 2, 7

산 술 연 산 자

대 입 연 산 자

복 합 대 입 연 산 자

증 감 연 산 자

관 계 연 산 자

논 리 연 산 자

조 건 연 산 자

JAVA

3. 연 산 자



관계 연산자

| 대소 관계를 비교하는 기호

기능	연산자	예
작다	<	$A < B$
크다	>	$A > B$
작거나 같다	<=	$A <= B$
크거나 같다	>=	$A >= B$
같다	==	$A == B$
같지 않다	!=	$A != B$

관계 연산자는 값을 **비교**할 때 사용하고 결과로 참(true)또는 거짓(false)이 나온다.

예제

```
public class Ex08 {  
    public static void main(String[] args) {  
        int n1 = 10;  
  
        System.out.println(n1 == 10); // 1: n1이 10과 같은지  
        System.out.println(n1 != 5); // 1: n1이 5와 다른지  
        System.out.println(n1 > 10); // 0: n1이 10보다 큰지  
        System.out.println(n1 < 10); // 0: n1이 10보다 작은지  
        System.out.println(n1 >= 10); // 1: n1이 10보다 크거나 같은지  
        System.out.println(n1 <= 10); // 1: n1이 10보다 작거나 같은지  
    }  
}
```

삼항 연산자

| Ternary operator

기능	연산자	예
조건식이 참이면 : 앞의 값을 반환, 거짓이면 : 뒤의 값을 반환	? :	A > B ? printf(“%d”, A) : printf(“%d”, B);

먼저 참/거짓을 판단할 변수나 **조건식**을 지정한 뒤 ? 다음에 **참과 거짓일 때 사용할 값**을 나열한다. 각 값은 :로 구분하며 : 앞은 참일 때, : 뒤는 거짓일 때 사용할 값이다.

조건식은 결과로 **참** 혹은 **거짓**이 나오는 식을 말한다.

예제

```
public class Ex09 {  
    public static void main(String[] args) {  
        int n1 = 5;  
        int n2;  
        n2 = n1 ? 100 : 200;  
        // n1이 참이면 n2에 100을 할당, 거짓이면 n2에 200을 할당  
        System.out.println(n2);  
        // 100: n1이 5이므로 참. n2에는 100이 할당됨  
    }  
}
```

예제

```
public class Ex10 {  
    public static void main(String[] args) {  
        int num1 = 10;  
        int num2;  
        num2 = num1 == 10 ? 100 : 200;  
        // num1이 10이면 num2에 100을 할당, 10이 아니면 num2에 200을 할당  
  
        System.out.println(num2); // 100: num1이 10이므로 num2에는 100이 할당됨  
    }  
}
```

해보기

관계 연산자를 사용하여 전부 참이 출력되게 완성해보자.

```
public class Quest {  
    public static void main(String[] args) {  
        int n1 = 27;  
  
        System.out.println(n1 ①__ 10);  
        System.out.println(n1 != 5);  
        System.out.println(n1 >= 27);  
        System.out.println(n1 ②__ 27);  
        System.out.println(n1 ③__ 30);  
        System.out.println(n1 <= 27);  
    }  
}
```

논리 연산자

| 논리값을 계산하는 기호

기능	연산자	예
AND; 논리곱	&&	A && B
OR; 논리합		A B
NOT; 논리부정	!	!A

진리표

| Truth Table

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

A	NOT
0	1
1	0

AND는 두 값이 모두 참여야 결과가 참이 나온다.

OR은 두 값이 모두 거짓이어야 결과가 거짓이 나온다.

NOT은 결과가 반대로 나온다.

예제

```
public class Ex11 {  
    public static void main(String[] args) {  
        System.out.println(true && true);  
        System.out.println(true && false);  
        System.out.println(false && true);  
        System.out.println(false && false);  
    }  
}
```

J A V A

예제

```
public class Ex12 {  
    public static void main(String[] args) {  
        System.out.println(true || true);  
        System.out.println(true || false);  
        System.out.println(false || true);  
        System.out.println(false || false);  
    }  
}
```

J A V A

예제

```
public class Ex13 {  
    public static void main(String[] args) {  
        System.out.println(!true);  
        System.out.println(!false);  
    }  
}
```

예제

```
public class Ex14 {  
    public static void main(String[] args) {  
        int n1 = 20, n2 = 10;  
        int n3 = 30, n4 = 15;  
        System.out.println(n1 > n2 && n3 > n4);  
        System.out.println(n1 > n2 && n3 < n4);  
        System.out.println(n1 > n2 || n3 < n4);  
        System.out.println(n1 < n2 || n3 < n4);  
        System.out.println(!(n1 > n2));  
    }  
}
```

예제

```
public class Ex15 {  
    public static void main(String[] args) {  
        boolean b1 = true;  
        boolean b2 = false;  
        System.out.println(b1 && b2 ? "True" : "False");  
        System.out.println(b1 || b2 ? "True" : "False");  
    }  
}
```

해보기

논리 연산자를 사용하여 전부 참이 출력되게 완성해보자.

```
public class Quest {  
    public static void main(String[] args) {  
        boolean b1 = true, b2 = false;  
        System.out.println((b1 ①___ b1) ? "참" : "거짓");  
        System.out.println((b1 ②___ b2) ? "참" : "거짓");  
        System.out.println(!③___ ? "참" : "거짓");  
    }  
}
```

비트 연산자

| 비트 단위로 연산하는 기호

기능	연산자	예
비트 AND	&	A & B
비트 OR		A B
비트 XOR	^	A ^ B
비트 NOT	~	~A

J A V A

진리표

| Truth Table

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

XOR는 두 값이 달라야 결과가 참이 나온다.

J A V A

예제

```
public class Ex16 {  
    public static void main(String[] args) {  
        byte n1 = 13;  
        byte n2 = 7;  
        int n3 = n1 & n2;  
  
        System.out.println(n3);  
    }  
}
```

J A V A

예제

```
public class Ex17 {  
    public static void main(String[] args) {  
        int n1 = 5;  
        int n2 = 3;  
        int n3 = -1;  
  
        System.out.println(n1 & n2);  
        System.out.println(n1 | n2);  
        System.out.println(n1 ^ n2);  
        System.out.println(~n3);  
    }  
}
```

비트 시프트 연산자

기능	연산자	예	기타
비트 왼쪽 이동	<<	A << 2	이동으로 인한 빈 공간은 0으로 채움
비트 오른쪽 이동	>>	A >> 2	이동으로 인한 빈공간은 음수의 경우 1, 양수의 경우 0으로 채움
비트 오른쪽 이동	>>>	A >>> 2	이동으로 인한 빈 공간은 0으로 채움

J A V A

예제

```
public class Ex18 {  
    public static void main(String[] args) {  
        System.out.println(2 << 1);  
        System.out.println(2 << 2);  
        System.out.println(2 << 3);  
  
        System.out.println(8 >> 1);  
        System.out.println(8 >> 2);  
        System.out.println(8 >> 3);  
  
        System.out.println(-8 >> 1);  
        System.out.println(-8 >> 2);  
        System.out.println(-8 >> 3);  
  
        System.out.println(-8 >>> 1);  
    }  
}
```

J A V A

해보기1

정수 7의 비트 열을 기반으로 2의 보수를 취하면 -7이 된다. 실제로 그런지 정수 7에 대한 2의 보수를 계산하여 출력해보자.

J A V A

해보기2

int형 정수 15678의 오른쪽에서 세 번째 비트와 다섯 번째 비트가 각각 어떻게 되는지 확인하여 출력해보자.

해보기3

<< 연산은 대부분의 경우 피연산자의 값에 2의 배수를 곱하는 결과를 보인다. 그러나 MSB를 변경 시켜서 이상한 결과를 보이는 경우도 있다. 음의 정수와 양의 정수를 하나씩 예로 들어서 엉뚱한 결과가 언제 발생하는지 설명하고, 이를 증명하기 위한 프로그램도 작성해 보자.