



# **JAVA 2강 연산자, 제어문**

---

**양 명 속**

**[now4ever7@gmail.com]**
















# 목차

---

- 연산자
- 제어문
  - 조건문
  - 반복문

# 연산자의 종류

이항은 + - \* /

종류	연산방향	연산자	우선순위
단항 연산자		<span style="border: 1px solid red;">++ --</span> + - ~ ! (타입)	높음
산술 연산자		* / %	
		+ -	
		<< >> >>>	
비교 연산자		< > <= >= instanceof	
		== !=	
논리 연산자		&	
		^	
			
		&&	
			
삼항 연산자		?:	
대입 연산자		= *= /= %= += -= <<= >>= >>>= &= ^=  =	낮음

# 연산자의 우선순위

+ - \* / > < == && || =

- 산술 > 비교 > 논리 > 대입
  - 대입은 제일 마지막에 수행
- 단항 > 이항 > 삼항
- 단항 연산자와 대입 연산자를 제외한 모든 연산의 진행방향은 왼쪽에서 오른쪽이다.





## 연산자 우선순위의 예

수식	설명
$-x + 3$	단항 > 이항, $x$ 의 부호를 바꾼 다음 덧셈 수행
$x + 3 * y$	$(3*y)$ 가 먼저
$x << 2 + 1$	산술 > 쉬프트, $(2+1)$ 먼저
$x + 3 > y - 2$	산술 > 비교, $(x+3)$ 과 $(y-2)$ 가 먼저
$x > 3 \ \&\& \ x < 5$	비교 > 논리, $(x>3)$ 과 $(x<5)$ 가 먼저
$x < -1 \    \ x > 3 \ \&\& \ x < 5$	and > or, and와 or가 함께 사용하는 경우 괄호 사용해서 명확히 하자. $x < -1 \    \ (x > 3 \ \&\& \ x < 5)$
<code>int result = x + y * 3;</code>	대입은 제일 나중에, 연산의 최종결과가 변수 <b>result</b> 에 저장



# 산술 연산자

연산자	설 명
+	덧셈
-	뺄셈
*	곱셈
/	나눗셈 ( 나누기 연산자는 양쪽 피연산자 중 하나라도 실수여야만 실수 연산을 함 피연산자가 둘 다 정수형이면 정수 나누기를 하여 소수점 이하를 버림)
%	나머지 연산자
<< >> >>>	쉬프트 연산자

# 예제

정수	의 덧셈	a = 7 + 3 : 10
정수	의 뺄셈	b = 7 - 3 : 4
정수	의 곱셈	a * b : 40
실수	의 나눗셈	7.0 / 3 : 2.3333333333333335
정수	의 나눗셈	7 / 3 : 2
정수	의 나머지 계산	7 % 3 : 1

```
class ArithOpTest
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int a = 7 + 3;
```

```
        int b = 7 - 3;
```

```
        int c = a * b;
```

```
        double d = 7.0 / 3;
```

```
        int e = 7 / 3;    //몫
```

```
        int f = 7 % 3;    //나머지
```

```
        System.out.println("정수의 덧셈 a = 7 + 3 : " + a);
```

```
        System.out.println("정수의 뺄셈 b = 7 - 3 : " + b);
```

```
        System.out.println("정수의 곱셈 a * b : " + c);
```

```
        System.out.println("실수의 나눗셈 7.0 / 3 : " + d);
```

```
        System.out.println("정수의 나눗셈 7 / 3 : " + e);
```

```
        System.out.println("정수의 나머지 계산 7 % 3 : " + f);
```

```
    }
```

```
}
```

# 쉬프트 연산자 - << >> >>>

■ << >> >>>

1칸 이동하면 2의 1승, 2칸은 2의 2승

- 정수형 변수에만 사용
- 피연산자의 각 자리(2진수로 표현했을 때)를 오른쪽으로 또는 왼쪽으로 이동(shift)한다
- 오른쪽으로 n자리 이동하면 피연산자를  $2^n$ 으로 나눈 것과 같은 결과를 얻음
- 왼쪽으로 n자리 이동하면  $2^n$ 으로 곱한 것과 같은 결과를 얻음

$x \ll n : x * 2^n$

$x \gg n : x / 2^n$  (x가 음수인 경우는  $x / 2^n$ 의 결과를 가지지 않음)

- << 연산자 : 부호 상관없이 왼쪽으로 이동시키며 빈칸을 0으로 채움
- >> 연산자 : 음수인 경우 부호를 유지시키기 위해 음수인 경우 빈자리를 1로 채움
- >>> 연산자 : 부호에 상관없이 항상 빈칸을 0으로 채움



```
-8  
111111111111111111111111111111110000  
  
-8 << 1    = -16  
111111111111111111111111111111110000  
  
-8 << 2    = -32  
11111111111111111111111111111111000000
```

```
-8  
11111111111111111111111111111111000  
  
-8 >> 1 = -4  
11111111111111111111111111111111100  
  
-8 >> 2 = -2  
11111111111111111111111111111111110
```

```
class ShiftOp {
    public static void main(String args[]) {
        int temp; // 계산 결과를 담기 위한 변수

        System.out.println(-8);
        // -8을 2진수 문자열로 변경한다.
        System.out.println(Integer.toBinaryString(-8));

        System.out.println();

        temp = -8 << 1;
        System.out.println( "-8 << 1  = " + temp);

        System.out.println(Integer.toBinaryString(temp));
        System.out.println();

        temp = -8 << 2;
        System.out.println( "-8 << 2  = " + temp);
        System.out.println(Integer.toBinaryString(temp));
        System.out.println();
    }
}
```

```
System.out.println();
System.out.println(-8);
System.out.println(Integer.toBinaryString(-8));
System.out.println();

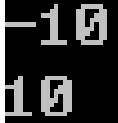
temp = -8 >> 1;
System.out.println("-8 >> 1 = " + temp);
System.out.println(Integer.toBinaryString(temp));
System.out.println();

temp = -8 >> 2;
System.out.println("-8 >> 2 = " + temp);
System.out.println(Integer.toBinaryString(temp));
System.out.println();
```





# 단항 연산자 - 부호 연산자



-10  
10

- +, -

- 피연산자의 부호를 변경하는데 사용
- **boolean**형과 **char** 형을 제외한 나머지 기본형에 사용

```
class Operator3
{
    public static void main(String[] args)
    {
        int i = -10;
        i = +i;
        System.out.println(i);
        i=-10;
        i = -i;
        System.out.println(i);
    }
}
```



# 단항 연산자 - 증감 연산자

연산자	설 명
++	++a // 연산 전에 1 증가 ( <b>a=a+1</b> )
	a++ // 연산 후에 1 증가
--	--a // 연산 전에 1 감소 (a=a-1)
	a-- // 연산 후에 1 감소

## 예제-단독으로 사용되는 경우

```
class Operator1
{
    public static void main(String args[])
    {
        //어떤 수식에 포함된 것이 아니라 단독으로 사용되는 경우 => 전위형과
        후위형 결과는 같다
        int i=5;
        i++;           // i=i+1;과 같은 의미. ++i;로 바꿔 써도 결과는 같다.
        System.out.println(i);
        i=5;           //결과를 비교하기 위해 i값을 다시 5로 변경.
        ++i;
        System.out.println(i);
    }
}
```

# 예제-수식에 포함되는 경우

```
class Operator2 {  
    public static void main(String args[]) {  
        int i=5;  
        int k=0;  
        k = i++;  
        System.out.println("k=i++; 실행 후, i=" + i + ", k="+ k);  
  
        i=5;           // 결과를 비교하기 위해, i와 j의 값을 다시 5와 0으로 변경  
        k=0;  
        k = ++i;  
        System.out.println("k=++i; 실행 후, i=" + i + ", k="+ k);  
    }  
}
```

**i**

**5**

**k**

**0**

```
k=i++; 실행 후, i=6, k=5  
k=++i; 실행 후, i=6, k=6
```

# 증감 연산자 예제

```
a=3, b=3  
<선증가> a=4  
<후증가> b=3  
a=4, b=4
```

```
class IncrementOp1  
{  
    public static void main(String[] args)  
    {  
        int a=3, b=3;  
  
        System.out.println("a=" +a +", b="+ b);  
        System.out.println("(선증가) a=" + ++a);  
        System.out.println("(후증가) b=" + b++);  
        System.out.println("a=" +a +", b="+ b);  
    }  
}
```

**a**

**3**

**b**

**3**

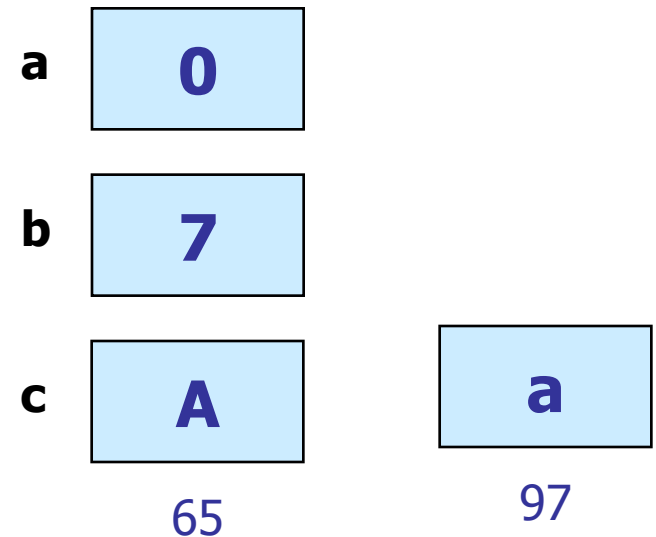
# 증감 연산자 실습

```
class IncrementOp2{
    public static void main(String[] args) {
        int a = 0;
        System.out.println("a : " + a);
        System.out.println("a++ : " + a++); //a가 사용되고 난 후에 1 증가
        System.out.println("++a : " + ++a); //a가 1 증가된 값을 사용

        double b = 7;
        System.out.println("b : " + b);
        System.out.println("b-- : " + b--);
        System.out.println("--b : " + --b);

        char c = 'A';
        System.out.println("c : " + c);
        System.out.println("++c : " + ++c);
        c = 'a';
        System.out.println("c : " + c);
        c++;
        System.out.println("c++ : " + c);
    }
}
```

```
a : 0
a++ : 0
++a : 2
b : 7.0
b-- : 7.0
--b : 5.0
c : A, ++c : B
c : a
c++ : b
```







# 단항연산자 - 비트 전환연산자 ~

---

■ ~

- 정수형과 `char` 형에만 사용
- 피연산자를 2진수로 표현했을 때, 0은 1로, 1은 0으로 바꾼다
- 연산자 ~ 에 의해 비트전환되고 나면 피연산자의 부호가 반대로 변경됨

# 예제

```
b = 10
~b = -11
~b+1 = -10
```

```
class Operator4 {
    public static void main(String[] args)
    {
        byte b = 10;
        System.out.println("b = " + b );
        System.out.println("~b = " + ~b);
        System.out.println("~b+1 = " + (~b+1));
    }
}
```

2진수

0	0	0	0	1	0	1	0
1	1	1	1	0	1	0	1
1	1	1	1	0	1	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	1	0

10진수

10
-11
-11
+) 1
-10

-16+4+1

-16+4+2

# 단항 연산자-논리 부정 연산자 !

■ !

- boolean 형에만 사용
- true는 false로, false는 true로 변경
- 조건문과 반복문의 조건식에 사용됨

```
false  
true  
false  
a>b : false  
!(a>b) : true
```

```
class Operator5 {  
    public static void main(String[] args) {  
        boolean power = false;  
        System.out.println(power);  
        power = !power;                // power의 값이 false에서 true로 바뀐다.  
        System.out.println(power);  
        System.out.println(!power);   // power의 값이 true에서 false로 바뀐다.  
        int a=3, b=5;  
        boolean bool = a>b;  
        System.out.println("a>b : " + bool);  
        bool = !(a>b);  
        System.out.println("!(a>b) : " + bool);  
    }  
}
```



# 비교 연산자

연산자	설 명
<	Less than
<=	Less than or equal
>	greater than
>=	Greater than or equal
==	Equal
!=	Not equal



# 예제

```
class RelationOp
{
    public static void main(String[] args)
    {
        int a = 7;
        int b = 3;

        System.out.println("a : " + a + ", b : " + b);
        System.out.println("a == b : " + (a == b));
        System.out.println("a != b : " + (a != b));
        System.out.println("a > b : " + (a > b));
        System.out.println("a < b : " + (a < b));
        System.out.println("a >= b : " + (a >= b));
        System.out.println("a <= b : " + (a <= b));
    }
}
```

```
a : 7, b : 3
a == b : False
a != b : True
a > b : True
a < b : False
a >= b : True
a <= b : False
```



# 논리 연산자

• x 는 0보다 크고, 10보다 작다  
`x>0 && x<10`

연산자	설 명	예
<code>&amp;&amp;</code>	논리 AND	<code>if ((x&lt;y) &amp;&amp; (x&gt;0))</code>
<code>  </code>	논리 OR	<code>if ((x&lt;y)    (x&gt;0))</code>

# 논리 연산자

$x > 0$	$x < 100$	$x > 0 \ \&\& \ x < 100$	$x > 0 \    \ x < 100$	$x > 0 \ \wedge \ x < 100$
---------	-----------	--------------------------	------------------------	----------------------------

$x > 0$	$y > 0$	$x > 0 \ \&\& \ y > 0$	$x > 0 \    \ y > 0$	$x > 0 \ \wedge \ y > 0$
---------	---------	------------------------	----------------------	--------------------------

A	B	$A \ \&\& \ B$	$A \    \ B$	$A \ \wedge \ B$
True	True	True	True	False
True	False	False	True	True
False	True	False	True	True
False	False	False	False	False

$\wedge$  (XOR) : 피연산자들의 값이 같으면 결과는 false, 다르면 true



# 논리 연산자 - 비트 연산자

연산자	설 명	예
&	AND	<pre>int result = bi &amp; bi2; //result=00000000</pre>
	OR	<pre>int result = bi   bi2; //result=00110011</pre>
^	XOR	<pre>int result = bi ^ bi2; //result=00110011</pre>



# 논리 연산자 - 비트 연산자

- 비트연산자 - 이진 비트연산을 수행함
  - 이진수로 표현했을 때의 각 자리수를 **and, or, xor** 연산을 수행
  - **float, double** 형을 제외한 모든 기본형에서 사용가능
  - 피연산자가 **boolean**형인 경우 조건식간의 연결에 사용할 수 있음
- **|** 연산자 : 피연산자중 한쪽의 값이 1이면 1을 결과로 얻고, 그 외에는 0을 얻음
- **&** 연산자 : 양쪽이 모두 1이어야 1을 결과로 얻고, 그 외에는 0
- **^** 연산자 : 피연산자의 값이 서로 다를때만 1을 결과로 얻고, 같을 때는 0

x	y	$x   y$	$x \& y$	$x \wedge y$
1	1	1	1	0
1	0	1	0	1
0	1	1	0	1
0	0	0	0	0

# 예제

```
x는 3이고, y는 5일 때,  
x | y = 7  
x & y = 1  
x ^ y = 6  
true | false = true  
true & false = false  
true ^ false = true
```

```
class BitOp  
{  
    public static void main(String[] args)  
    {  
        int x = 3; //0011  
        int y = 5; //0101  
        System.out.println("x는 " + x + "이고, y는 " + y + "일 때, ");  
        System.out.println("x | y = " + (x|y)); //0111 => 7  
        System.out.println("x & y = " + (x&y)); //0001 => 1  
        System.out.println("x ^ y = " + (x^y)); //0110 => 6  
  
        System.out.println("true | false = " + (true|false));  
        System.out.println("true & false = " + (true&false));  
        System.out.println("true ^ false = " + (true^false));  
    }  
}
```

# 논리 연산자

true || ....

false && ....

⇒ 뒤는 실행 안 함

true | ....

false & ....

=> 뒤도 실행

- 논리 연산자

- 값 연산 후 결과는 반드시 참 또는 거짓이 됨

연산자	&		^	&&	
의미	AND 연산	OR 연산	XOR	SC-AND	SC-OR

이 뒤의 두 개가 더 빠름

- &&, || Short-Circuit AND, OR 연산자 : 두 값을 가지고 논리 연산할 경우 첫 번째 값만을 가지고 참, 거짓을 판단할 수 있다면, 거기서 바로 연산을 빠져 나와 효과적이고 빠른 코드를 만들 수 있는 연산자
  - || 연산자는 첫 번째 값이 참이면 그 연산은 무조건 참이 되고, && 연산자는 첫 번째 값이 거짓이면 그 연산은 무조건 거짓이 되는 경우



# 논리 연산자 예제

```
a : True, b : False
a && b : False
a || b : True
!a : False
```

```
class ConditionalOp
{
    public static void main(String[] args)
    {
        int x=10, y=-20;
        boolean a = x>0; //true
        boolean b = y>0; //false

        System.out.println("a : " + a + ", b : " + b);
        System.out.println("a && b : " + (x>0 && y>0));
        System.out.println("a || b : " + (x>0 || y>0));
        System.out.println("!a : " + (!(x>0)));
    }
}
```

```
int p=0, q=1;
if( (p++==0) | (q++==2) )
    p=42;

System.out.println("p="+p+", q="+q);
p=0; q=1;

if( (p++==0) || (q++==2) )
    p=42;

System.out.println("p="+p+", q="+q);
```



# 대입 연산자

연산자	설 명
=	대입
+=	덧셈 연산 후 대입
-=	뺄셈 연산 후 대입
*=	곱셈 연산 후 대입
/=	나눗셈 연산 후 대입
%=	나머지 연산자 연산 후 대입



# 대입 연산자

## ■ 대입 연산자

- 새로운 값을 변수나 속성 등에 대입할 때 사용
- 우변의 변수나 식을 평가한 결과를 좌변의 변수에 대입
- 좌변과 우변의 타입이 일치해야 함(형 변환)
- 대입 연산자를 여러 번 사용하여 복수 개의 변수에 동일한 값을 대입할 수도 있음
  - 예) `i=j=k=5;`

`=, +=, -=, *=, /=, %=, &=, |=, ^=,  
<<=, >>=, >>>=`



# 예제

```
class AssignOp
{
    public static void main(String[] args)
    {
        int a = 7;
        int b = 3;
        System.out.println("a : " + a + ", b : " + b);

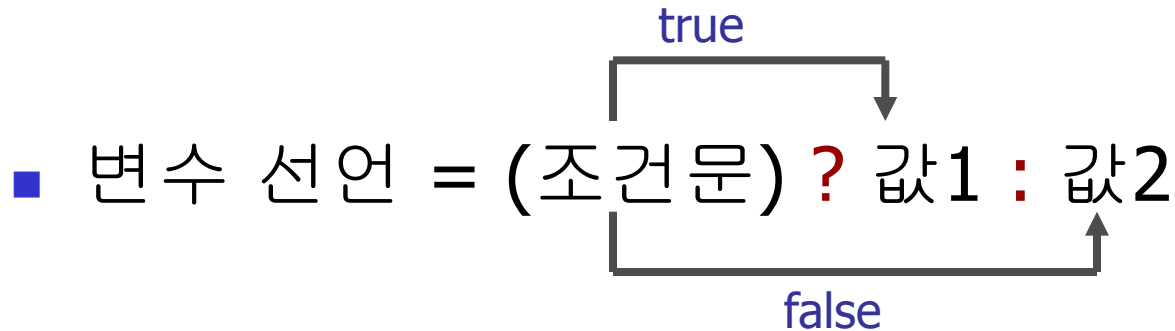
        a += b;    //a=a+b
        System.out.println("a += b");
        System.out.println("a : " + a);

        a /= b;    //a=a/b
        System.out.println("a /= b");
        System.out.println("a : " + a);

        a*=2; //a=a*2
    }
}
```

```
a : 7, b : 3
a += b
a : 10
a /= b
a : 3
```

# 삼항 연산자 (조건 연산자)



- 조건문의 결과에 따라 실행되는 내용이 달라짐
  - 조건문의 결과가 **true**이면 '값1' 을 변수에 대입
  - 조건문의 결과가 **false**이면 '값2'를 변수에 대입
- 주의 - 변수와 값1, 값2는 데이터타입이 같아야 함

조건식 => 결과가 true, false => 비교연산자나 논리연산자가 필요함





# 예제

```
a= 10, b= 20, 더 큰수 : 20  
같다  
남자
```

```
class ThreeOp
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int a=10, b=20;
```

```
        int c=(a>b) ? a: b;
```

```
        System.out.println("a= "+ a +", b= "+ b +", 더 큰수 : " + c);
```

```
        int d=30;
```

```
        System.out.println(d==30 ? "같다" : "다르다");
```

```
        //d==30? System.out.println("같다") : System.out.println("다르다"); //에러
```

```
        int i=1;
```

```
        String str= (i != 1)? "여자" : "남자";
```

```
        System.out.println(str);
```

```
        //i != 0? str="남자" : str="여자"; //에러
```

```
    }
```

```
}
```

# 실습

조건식 => 결과가 true, false => 비교연산자나 논리연산자가 필요함

- 1. int 변수에 임의의 값을 넣고, 그 값이 양수인지 음수인지 판별하여 출력하기

`num=-25 => 음수`

- 삼항 연산자 이용

- 2. int 변수에 임의의 값을 넣고, 그 값이 홀수인지, 짝수인지 판별하여 출력해보기

- 삼항 연산자, %연산자 이용

`num=17 => 홀수`

- 3. 키보드 입력을 받아서 홀수인지, 짝수인지 처리하기

숫자를 입력하세요

59

`n=59 => 홀수`



# 제어문

---



# 제어문 : 조건문/반복문

---

- 조건문
  - if~else문
  - switch~case문
- 반복문
  - for문
  - while문
  - do while문
- 기타
  - continue문
  - break문
  - goto문



# 흐름제어

---

- 제어문
  - 프로그래밍의 실행순서나 흐름을 제어할 때 사용
  - 실제 프로그램에서는 조건 비교, 반복실행 등이 필요
- 조건문
  - 조건의 진위 여부에 따라 명령의 실행 여부를 결정하는 문장
  - 특정조건이 만족될 때에만 실행하고 싶은 경우 사용
  - **if 문**
    - 조건을 검사해서 어떤 일을 해야 할지를 결정함
    - if 문을 이용해 조건이 참인지 거짓인지를 조사하여 프로그램의 흐름을 바꿔줄 수 있음
  - **switch 문**
    - if 문은 한 번에 한 가지 조건만 비교할 수 있음
    - switch 문은 한번에 여러 개의 조건을 비교할 수 있음

# 조건문

- 조건식 : 결과가 **true** 또는 **false**

- 1) 비교연산자 이용

변수 > 값

변수 == 값

- 2) 논리연산자도 이용

변수 > 값 && 변수 < 값

- 조건문

- 조건을 판단한 후 조건에 따라 분기하여 수행

- if 문

- 조건을 만족하면 문장을 실행하고 다음 라인을 수행

- if 문의 조건을 만족하지 않으면 수행을 한 번도 하지 않음.

- 형식

```
if (조건식)
```

```
{
```

```
    문장;
```

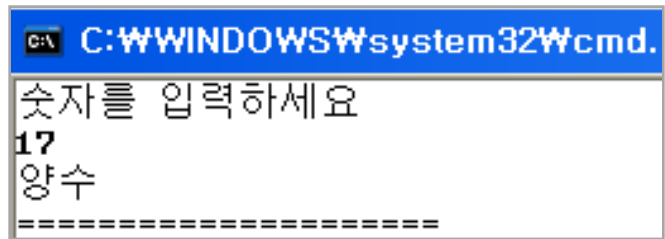
```
}
```

# 예제

```
import java.util.*;
class IfTest5
{
    public static void main(String[] args)
    {
        System.out.println("숫자를 입력하세요");
        Scanner sc = new Scanner(System.in);
        int num = sc.nextInt();

        if(num > 0)
            System.out.println("양수");
        else if (num == 0)
            System.out.println("0");
        else
            System.out.println("음수");

        System.out.println("=====");
    }
}
```



```
C:\WINDOWS\system32\cmd.
숫자를 입력하세요
17
양수
=====
```

# 예제

```
import java.util.*;
```

```
class IfTest  
{
```

```
    public static void main(String[] args)  
    {
```

```
        Scanner sc = new Scanner(System.in);  
        System.out.println("이용약관에 동의합니까? (Y)es/(N)o");  
        String agree = sc.nextLine();
```

```
        if (agree.equals("Y"))  
        {  
            System.out.println("동의하였습니다!");  
        }
```

```
        System.out.println("=====");
```

```
    }  
}
```

```
이용약관에 동의합니까? <Y>es/<N>o  
Y  
동의하였습니다!  
=====
```

```
이용약관에 동의합니까? <Y>es/<N>o  
N  
=====
```

- 기본자료형에서 등가연산자(==)는 값을 비교
- 참조형에서 등가연산자는 주소값을 비교
- 문자열 내용비교 => String 클래스의 equals()메소드 이용.
- public boolean equals(Object str)





# 조건문

---

- if ~ else 문

- 형식

```
if (조건)
{
    문장 1 ;
}
else
{
    문장 2 ;
}
```

- 조건을 만족하면 조건 다음의 문장1을 수행하고 조건을 만족하지 않으면 else 다음의 문장 2를 수행

# 예제

```
import java.util.*;
class IfTest1
{
    public static void main(String[] args)
    {
        System.out.println("이용약관에 동의합니다? (Y)es/(N)o");
        Scanner sc = new Scanner(System.in);
        String agree = sc.nextLine();

        if (agree.equals("Y"))
        {
            System.out.println("동의하였습니다!");
        }
        else
        {
            System.out.println("동의하지 않았습니다!!");
        }

        System.out.println("=====");
    }
}
```

```
이용약관에 동의합니다? <Y>es/<N>o
Y
동의하였습니다!
=====
```

```
이용약관에 동의합니다? <Y>es/<N>o
N
동의하지 않았습니다!!
=====
```



# 조건문

## ■ 다중 if 문

### ■ 형식

- 조건1이 참인지 거짓인지 검사해서
  - 참이면 문장1을 실행,
  - 참이 아니면 **else if** 다음의 조건2가 참인지 거짓인지 검사해서
    - 참이면 문장2를 거짓이면 다음의 기본문장을 실행

```
if (조건1)
{
    문장 1 ;
}
else if (조건2)
{
    문장 2 ;
}
else
{
    기본문장;
}
```

# 예제

```
import java.util.*;
class IfTest2{
    public static void main(String[] args){
        System.out.println("이용약관에 동의합니까? (Y)es/(N)o");
        Scanner sc = new Scanner(System.in);
        String agree = sc.nextLine();

        if (agree.equals("Y"))
        {
            System.out.println("동의하였습니다!");
        }
        else if (agree.equals("N"))
        {
            System.out.println("동의하지 않았습니다!!");
        }
        else
        {
            System.out.println("잘못 입력하셨습니다.");
        }
        System.out.println("=====");
    }
}
```

```
이용약관에 동의합니까? <Y>es/<N>o
Y
동의하였습니다!
=====
```

```
이용약관에 동의합니까? <Y>es/<N>o
N
동의하지 않았습니다!!
=====
```

```
이용약관에 동의합니까? <Y>es/<N>o
a
잘못 입력하셨습니다.
=====
```



## 예제

---

- 성별을 입력받아서
- M 이면 "남자이시군요"
- F 이면 "여자이시군요"
- 나머지는 "잘못 입력했습니다!" 출력하기

```
당신의 성별은 무엇입니까? <M>ale/<F>emale  
F  
여자이시군요.
```



# IF문

- 기본자료형에서 등가연산자(==)는 값을 비교
- 참조형에서 등가연산자는 주소값을 비교
- 문자열 내용비교 => String클래스의 equals()메소드 이용.
- public boolean equals(Object str)

```
import java.util.*;
class IfTest3{
    public static void main(String[] args) {
        System.out.println("당신의 성별은 무엇입니까? (M)ale/(F)emale");
        Scanner sc = new Scanner(System.in);
        String gender = sc.nextLine();

        String str = "";
        if (gender.equals("M")){
            str = "남자이시군요.";
        }
        else if (gender.equals("F")){
            str = "여자이시군요.";
        }
        else{
            str = "잘못 입력하셨습니다.";
        }
        System.out.println(str);
    }
}
```



## 실습

- 평균을 입력 받아 학점 구하기
  - 90점 이상 : A
  - 80점 이상 : B
  - 70점 이상 : C
  - 50점 이상 : D
  - 50점 미만 : F

평균을 입력하세요

84

평균 : 84

학점 : B



## If 문 - 중첩 if 문

---

### ■ 중첩 if 문

- If문이 중복으로 사용된 것
- 바깥쪽 if문이 참이 되어야 안쪽 if문을 수행함

```
If (조건문)
{
    if (조건문 )
        명령문1;
    else
        명령문2;
}
```



# 중첩 if문

```
회원여부를 입력하세요<1. 회원, 2. 비회원>  
1  
회원이 구매한 금액을 입력하세요  
750000  
구매금액별 사은품 : 스피커
```

```
import java.util.*;  
class MemberIfTest{  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        System.out.println("회원여부를 입력하세요(1. 회원, 2. 비회원)");  
        int memberFlag = sc.nextInt();  
        String gift=""; //사은품  
        if(memberFlag==1){ //회원인 경우만 처리  
            System.out.println("회원이 구매한 금액을 입력하세요");  
            int purchaseAmount = sc.nextInt();  
            if(purchaseAmount>=1000000){  
                gift="외장하드";  
            }else if(purchaseAmount>=500000){  
                gift="스피커";  
            }else if(purchaseAmount>=100000){  
                gift="마우스";  
            }else{  
                gift="10만원 미만은 사은품이 없습니다";  
            }  
        }else{  
            gift="비회원은 사은품 증정 불가";  
        }  
        System.out.println("구매금액별 사은품 : "+gift);  
    }  
}
```

# switch문

제어변수 : byte, short, int, char 자료형만 사용가능  
- jdk7.0 이상 : String도 사용가능

## ■ switch문

```
switch(제어 변수)
{
    case 값1 :
        실행블록1;
        break;
    case 값2 :
        실행블록2;
        break;
    default :
        실행블록 default 처리;
        break;
}
```

※ case 블록은 {}로 묶여있지 않음

- 비교할 **조건이 많은 경우**, 정해져 있는 값을 가진 경우
  - if else문 대신 switch문 사용
- switch문에서는 비교연산이 올 수 없다

- 제어변수에 해당하는 값이 “값1”이면 **case 값1**의 블록으로 위치 이동
- 제어 변수의 값과 일치하는 **case**의 명령을 찾아 실행하는 다중 선택문



# switch 문

---

- 여러 개의 case 구문을 사용하여 여러 조건 처리
- **case 다음에는 상수만** 쓸 수 있음
  - 변수나 범위를 지정할 수 없음
- 제어변수 - **byte, short, int, char** 자료형의 값을 나타낼 수 있는 **필드나 식**
- case 문 안에 있는 코드는 모두 한 묶음이므로 {}로 블록을 묶을 필요는 없음
  - 묶음의 끝을 표시하기 위해 **case 문의 끝에는 항상 break 문이 있어야 함**



# switch 문

---

- 여러 개의 case를 **or** 로 연결하려면
  - case 문 자체를 비워둘 수 있음
  - case 만 써 두고 명령을 기술하지 않으면 아래쪽의 case 에 있는 명령을 실행
    - 여러 개의 case에 대해 동일한 처리를 할 수 있음

```
case 1:  
case 2:  
case 3:  
    System.out.println("3보다 작거나 같은 숫자");  
    break;
```

# 예제

three

```
class SwitchTest1
{
    public static void main(String[] args)
    {
        int i = 3;

        switch(i)
        {
            case 1:
                System.out.println("one");
                break;
            case 2:
                System.out.println("two");
                break;
            case 3:
                System.out.println("three");
                break;
            default:
                System.out.println("그 외의 숫자!");
                break;
        }
    }
}
```

- if 문과 비교

```
if (i == 1)
    System.out.println("one");
else if(i==2)
    System.out.println("two");
else if(i==3)
    System.out.println("three");
else
    System.out.println("그 외의 숫자!");
```

# 예제

```
성별을 입력하세요. F/M
m
남자이시군요
```

- if 문과 비교

```
if (gender.equals("M"))
{
    str = "남자이시군요.";
}
```

```
import java.io.IOException;
```

```
class SwitchTest2{
```

```
    public static void main(String[] args) throws IOException{
```

```
        System.out.println("성별을 입력하세요. F/M");
```

```
        char gender = (char)System.in.read(); //사용자가 입력한 아스키코드값을 반환해 줌
                                                //앞의 1바이트만 읽는다
                                                //ABC 를 입력 => A에 해당하는 코드 65를 리턴
```

```
        String str = "";
```

```
        switch(gender){
```

```
            case 'M':
```

```
                str = "남자이시군요";
```

```
                break;
```

```
            case 'F':
```

```
                str = "여자이시군요";
```

```
                break;
```

```
            default:
```

```
                str = "잘못 입력하셨습니다! ";
```

```
                break;
```

```
        }
```

```
        System.out.println(str);
```

```
    }
```

```
public abstract int read() throws IOException
```

키보드로 부터 1바이트를 읽어오는 메서드  
-입력한 값의 아스키코드 값을 리턴해줌  
- char 값을 읽어올 때 사용

```
Scanner sc = new Scanner(System.in);
String sGender = sc.nextLine();
char gender = sGender.charAt(0);
```



## 예제

```
산술연산자를 입력하세요(<+, -, *, />
*
곱하기
```

- switch 문 이용
  - +, -, \*, / 를 입력 받아서
  - + 이면 "더하기"
  - - 이면 "빼기"
  - \* 이면 "곱하기"
  - / 이면 "나누기"
  - 그 이외는 "잘못 입력!" 출력하기



# 예제

```
import java.io.*;
class SwitchTest4{
    public static void main(String[] args) throws IOException{
        System.out.println("산술연산자를 입력하세요(+, -, *, /)");
        char op = (char)System.in.read();
        String str = "";
        switch(op){
            case '+': str = "더하기";
                     break;
            case '-': str="빼기";
                     break;
            case '*': str="곱하기";
                     break;
            case '/': str="나누기";
                     break;
            default: str = "잘못 입력... ";
                     break;
        }
        System.out.println(str);
    }
}
```





## 예제

---

- switch 문 이용하여 학점 구하기

```
grade="";
switch ( (int)avr/10)
{
    case 10:
    case 9: grade="A";          break;
    case 8: grade="B";          break;
    case 7: grade="C";          break;
    case 6: grade="D";          break;
    default: grade="F";
}
System.out.println("학점 : " + grade);
```



## 실습

---

- switch 문 이용
- 취미를 입력 받아서, 입력 받은 취미를 화면에 출력
  - 취미는 1. 영화 2. 축구 3. 야구 4. 등산 중에서 선택하게 하고, 화면 출력시 입력 받은 취미가 1 이면 ‘영화를 선택하였습니다’라고 출력함

```
취미를 입력하세요 : 1. 영화 2. 축구 3. 야구 4. 등산
3
야구를 선택하였습니다
```



## 실습

---

- If 문 이용 – 날짜를 입력 받아서  
날짜가        1 ~ 10일 : 초순,  
              11~20일 : 중순,  
              21~31일 : 하순,  
              나머지 : 잘못 입력하였습니다    출력

```
날짜<일>를 입력하세요<1~31>  
9  
9 일: 초순
```

```
날짜<일>를 입력하세요<1~31>  
35  
35 일: 잘못 입력하였습니다.
```



## 실습

---

- switch 문 이용 - 월을 입력 받아 해당하는 사분기 출력
  - 1, 2, 3월 : 1사분기,
  - 4, 5, 6월 : 2사분기,
  - 7, 8, 9월 : 3사분기,
  - 10, 11, 12월 : 4사분기,
  - 나머지 : 잘못 선택했어요

```
월을 입력하세요  
4  
4월은 2사분기입니다.
```

```
월을 입력하세요  
13  
잘못 선택했어요
```

# 과제

## ■ [if 문 이용]

- 1. 정수를 입력 받아서
  - 0보다 크면 "양수입니다."
  - 0보다 작으면 "음수입니다."
  - 나머지는 "0 입니다." 출력하기
- 2. 아이디와 비밀번호를 입력 받아서
  - 아이디가 "hong" 이고, 비밀번호가 "1234"이면
    - "로그인되었습니다."
  - 나머지
    - "아이디나 비밀번호가 틀렸습니다." 출력하기

```
정수를 입력하세요!  
59  
양수입니다.
```

```
아이디를 입력하세요  
hong  
비밀번호를 입력하세요  
1234  
로그인되었습니다
```

```
아이디를 입력하세요  
hong  
비밀번호를 입력하세요  
777  
아이디나 비밀번호가 틀렸습니다!!
```



# 과제

```
직업을 입력하세요 : A. 회사원 B. 학생 C. 주부 D. 기타  
b  
학생이시군요
```

- [switch 문 이용]
  - 직업을 입력 받아서
    - A 이면 "회사원이시군요"
    - B 이면 "학생이시군요"
    - C 이면 "주부이시군요"
    - D 이면 "기타를 선택했어요"
    - 나머지 "잘못 입력했어요" 출력하기
  - API 에서 Character 클래스에서 대문자로 변환하는 메서드 찾아서 적용하기



## 과제2

```
산술연산자를 입력하세요(<+, -, *, /, %>
*
실수를 두 개 입력하세요
5.3
10.2
결과값:54.059999999999995
```

- [switch문 이용]
- 산술연산자 중 하나를 선택하게 하고,
- 실수 두 개를 입력 받아서
  - +을 선택했으면 두 수 더하기
  - -을 선택했으면 두 수 빼기
  - \*을 선택했으면 두 수 곱하기
  - /을 선택했으면 두 수 나누기
  - %을 선택했으면 두 수의 나머지를 구하여 결과값 출력하기

## 과제2

- [중첩 If 문 이용]
- 나이를 입력 받아서,  
나이가 20세 미만이면 에러 메시지를 출력하고,  
나이가 20세 이상인 경우에는 다시 취미를 입력 받고, 입력 받은 취미  
를 화면에 출력
  - 취미는 1. 영화 2. 축구 3. 야구 4. 등산 중에서 선택하게 하고, 화  
면 출력시 입력 받은 취미가 1 이면 '영화를 선택하였습니다'라고  
출력함

```
나이를 입력하세요
10
20세 이상만 본 설문에 응할 수 있습니다.
```

```
나이를 입력하세요
21
취미를 입력하세요 : 1. 영화 2. 축구 3. 야구 4. 등산
1
영화를 선택하였습니다
```



## 과제3

```
국어, 영어, 수학 점수를 입력하세요
70
85
64

평균 : 73.0, 학점 : C
합격여부 : 합격
```

- 국어, 영어, 수학 점수를 입력 받아서, 평균을 구한 후, "합격", "불합격" 출력하기
  - 평균이 70 이상이고, 모든 과목이 50 이상이어야 합격
  - 불합격의 경우는 과락과 불합격으로 구분
    - 과락 : 평균이 70 이상이라도, 한 과목이라도 50미만이면 과락
    - 불합격 : 나머지는 불합격
- 평균으로 학점 구하기 - switch 문 이용
  - 90 이상이면 "A", 80 이상이면 "B", ...

```
국어, 영어, 수학 점수를 입력하세요
49
90
100

평균 : 79.66666666666667, 학점 : C
합격여부 : 과락
```

```
국어, 영어, 수학 점수를 입력하세요
68
71
51

평균 : 63.333333333333336, 학점 : D
합격여부 : 불합격
```

## 과제3

- 아스키 코드값  
0 ~ 9 : 48 ~ 57  
A ~ Z : 65 ~ 90  
a ~ z : 97 ~ 122

정수를 입력하세요

17

입력한 정수 : 17, 홀수

삼항연산자 이용 결과=>홀수

### ■ [if 문 이용]

- 1. 정수를 입력 받아서, 짝수인지 홀수인지 화면에 출력하기
  - if~else문을 삼항 연산자로 바꿔 작성하기
- 2. 임의의 값을 입력 받아서,  
입력한 값이 0~9사이의 값이면 "숫자입니다",  
입력한 값이 A~Z, a~z이면 "알파벳 문자입니다"  
그외 문자이면 "기타 문자입니다"를 출력하기
  - System.in.read() 이용
- 3. java.lang.Character 클래스의 메소드를 이용하여 숫자인지, 알파벳인지, 기타문자인지 확인하여 출력하기
  - API 문서에서 숫자인지 여부를 확인하는 메서드, 알파벳인지 여부를 확인하는 메서드 2개 찾아서 이용 (is로 시작하는 메서드)

```
0~9 나 알파벳, 그외 문자를 입력하세요
5
입력한 값: 5
숫자입니다
```

```
0~9 나 알파벳, 그외 문자를 입력하세요
*
입력한 값: *
기타 문자입니다
```



# 반복문

---



# 반복문

---

- Hello java 를 20번 출력해야 한다면?
- 반복문 - 반복적인 작업을 수행할 때 사용
  - for문
  - while문
  - do while문, 확장 for문
- for 문
  - 지정된 횟수만큼 반복해서 실행하는 구문
  - 반복 횟수가 미리 정해진 처리에 주로 사용



# for 문

---

## ■ for 문

### ■ 특징

- [시작값], [최종값], [증가값]을 설정하여 반복적인 수행을 할 수 있는 문장
- 규칙적인 증가를 하는 경우에 많이 사용

### ■ 형식

```
for (초기식; 조건식; 증감식)
{
    반복 명령 ;
}
```

- 한 문장 이상인 경우에는 ‘{’와 ‘}’ 기호를 이용하여 문장을 묶어줌.

# for 문

- 0부터 2까지 3번 반복

```
①처음한번만   ②           ④
for ( int i=0 ; i<3 ; i++ )
{
  ③ System.out.println(i);
}
```

0 1 2

- 첫 번째 루프의 흐름

①→②→③→④ [i=0]

- 두 번째 루프의 흐름

②→③→④ [i=1]

- 세 번째 루프의 흐름

②→③→④ [i=2]

- 네 번째 루프의 흐름

② [i=3] 따라서 탈출!

- for문은 시작과 동시에 딱 한번 초기식을 실행함
- 그리고 나서 바로 조건식을 검사함
- 조건식이 만족되지 않으면 한번도 루프를 실행하지 못하고 종료하게 됨
- 조건이 만족된다면 루프를 실행함
- 루프를 실행하고 나면 조건식을 실행해서 루프를 계속 돌릴 것인지 결정해야 하는데, 그 전에 반드시 증감식을 실행하게 됨

# 예제

class ForTest2

```
{
    public static void main(String[] args)
    {
        for(int i=0; i<3; i++)
        {
            System.out.println("hello java");
        }
    }
}
```

```
hello java
hello java
hello java
```

```
for(int i=3; i>0; i--)
{
    System.out.println(i);
}
```

```
3
2
1
```

class ForTest1

```
{
    public static void main(String[] args)
    {
        for(int i=0; i<3; i++)
        {
            System.out.println(i);
        } //for
        System.out.println("----ㄷㄷ----");
    }
}
```

```
0
1
2
-----ㄷㄷ-----
```

```
10 java
8 java
6 java
4 java
2 java
0 oracle
-2 spring
-4 spring
-6 spring
```



## 예제

```
반복하고 싶은 횟수를 입력하세요
3

0 : Hello JAVA!
1 : Hello JAVA!
2 : Hello JAVA!
```

- 사용자로부터 하나의 숫자를 입력 받아서 그 수만큼 hello java 를 출력하는 프로그램 작성하기

```
import java.util.*;
class ForTest3
{
    public static void main(String[] args)
    {
        System.out.println("반복하고 싶은 횟수를 입력하세요");
        Scanner sc = new Scanner(System.in);
        int count = sc.nextInt();

        System.out.println();
        for (int i=0; i<count ;i++ )
        {
            System.out.println(i + " : Hello JAVA!");
        }
    }
}
```



# 예제

```
1~ n 까지의 덧셈 : n 을 입력하세요
5
0+1
1+2
3+3
6+4
10+5

1~5까지의 합: 15
```

- 출력부분의 위치
- for문 안에서 출력하느냐  
바깥에서 출력하느냐에 따라  
다름

- 1~n 까지의 덧셈 결과를 출력하는 프로그램
  - n : 사용자로부터 입력 받는다

```
import java.util.*;
class ForTest4{
    public static void main(String[] args){
        System.out.println("1~ n 까지의 덧셈 : n 을 입력하세요");
        Scanner sc = new Scanner(System.in);
        int num = sc.nextInt();

        int sum=0;
        for(int i=1; i<=num; i++){
            System.out.println(sum + "+" + i);
            sum += i; //sum = sum + i
            //System.out.println("1~" + i +"까지의 합: " + sum);
        }
        System.out.println("1~" +num +"까지의 합: " + sum);
    }
}
```

- 1~n 까지의 짝수의 합 구하기
- 1~n 까지의 홀수의 합 구하기

• 1~3까지 합  
sum=0  
sum=sum+i

i=1 : 0+1  
i=2 : 0+1+2  
i=3 : 0+1+2+3

```
int sum=0;
for(int i=1; i<=3; i++){
    sum += i; //sum = sum+i
}
```



# for 문

- for문 내에서 선언된 변수는 블록을 빠져나가면 소멸

```
for (int i = 0; i < 3; i++)  
{  
    System.out.println(i);  
}  
System.out.println(i); // 에러 발생
```

```
int i = 0; //초기화  
for ( ; i < 3; i++)  
{  
    System.out.println(i + ": 번째");  
}  
System.out.println("for문 탈출 이후 i : " + i );
```

```
0: 번째  
1: 번째  
2: 번째  
for문 탈출 이후 i : 3
```

# 블록이란?

## ■ 단일 블록

```
{  
    // code  
}
```

중첩된 블록내에서는  
동일한 변수명  
사용이 불가능

```
{  
    int i;  
    ...  
    {  
        int i;  
        ...  
    }  
}
```

다른 블록내에서는  
동일한 변수명  
사용 가능

```
{  
    int i;  
    ...  
}  
...  
{  
    int i;  
    ...  
}
```



# 변수의 범위(Scope)

---

```
class ForTest5{
    public static void main(String[] args) {
        //변수의 범위(scope)
        int a=10; //main() 메서드내에서 사용
        System.out.println(a);

        for (int i=0; i<3 ; i++ )    //for(int a=0; a<3; a++) 사용 불가
        {
            System.out.print("hello! ");
            System.out.println(a);
        }//for

        for(int i=0; i<3; i++)
        {
            int n=20;
            System.out.println("i=" + i + ", n=" + n);
        }//for

        // System.out.println(i); //에러
        // System.out.println(n); //에러

    } //main
} //class
```

# 예제

## ■ for 문의 중첩

```
class ForTest6
{
    public static void main(String[] args)
    {
        for(int i=0; i<3; i++) //바깥 for
        {
            System.out.println("=====현재 i : " + i);
            for(int j=0; j<2; j++) //안쪽 for
            {
                System.out.println("현재 j : " + j);
            }
        } //바깥 for
    } //main
} //class
```

```
=====현재 i : 0
현재 j : 0
현재 j : 1
=====현재 i : 1
현재 j : 0
현재 j : 1
=====현재 i : 2
현재 j : 0
현재 j : 1
```



```
=====현재 i : 0
현재 j : 0
현재 j : 1
✓
=====현재 i : 1
현재 j : 0
현재 j : 1
✓
=====현재 i : 2
현재 j : 0
현재 j : 1
✓
```



# 예제

- 중첩된 for 문 이용 : 2단~9단까지 구구단을 출력

```
class ForTest7
{
    public static void main(String[] args)
    {
        for(int dan=2; dan<10; dan++)
        {
            for(int j=1; j<10; j++)
            {
                System.out.println(dan + " * " + j + " = " + dan*j );
            }
            System.out.println();
        }
    }
}
```

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
```



# for 문

---

- for 문의 각 식은 필요 없을 시 생략 가능
  - 조건식을 생략하면 무한히 반복되는 무한 루프를 만들 수 있음
  - for(;;)
  - 무한 루프를 빠져나올 때는 break 문을 사용

# 예제

```
import java.util.*;
class ForLoop
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);

        for (;;)
        {
            System.out.println("게임 중~");
            System.out.println("계속 하시겠습니까?(Y/N)");
            String gameYn = sc.nextLine();

            if(gameYn.equals("N"))
            {
                break;
            } //if
        } //for
    } //main
} //class
```

```
게임 중~
계속 하시겠습니까?(Y/N)
y
게임 중~
계속 하시겠습니까?(Y/N)
n
계속하려면 아무 키나 누르십시오 . . .
```



# 실습

## ■ for 문 이용

### ■ 반복하고 싶은 횟수(n)를 입력 받아서

- 1. 그 횟수만큼 "재미있는 java!" 출력하기
- 2. 0부터 n-1 까지 출력하기  $0 \sim 2$
- 3. n까지의 합계 구하여 출력하기  $1 + 2 + 3$

```
반복하고 싶은 횟수를 입력하세요
3
재미있는 java!
재미있는 java!
재미있는 java!

i=0
i=1
i=2

i=0, sum=0
i=1, sum=1
i=2, sum=3
i=3, sum=6
0부터 3까지의 합=6
```



## 실습

- for문을 이용하여 알파벳 소문자(a~z) 출력하기

```
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
```

- 중첩 for문 이용
  - 구구단을 가로로 출력

```
2*1=2   3*1=3   4*1=4   5*1=5   6*1=6   7*1=7   8*1=8   9*1=9
2*2=4   3*2=6   4*2=8   5*2=10  6*2=12  7*2=14  8*2=16  9*2=18
2*3=6   3*3=9   4*3=12  5*3=15  6*3=18  7*3=21  8*3=24  9*3=27
2*4=8   3*4=12  4*4=16  5*4=20  6*4=24  7*4=28  8*4=32  9*4=36
2*5=10  3*5=15  4*5=20  5*5=25  6*5=30  7*5=35  8*5=40  9*5=45
2*6=12  3*6=18  4*6=24  5*6=30  6*6=36  7*6=42  8*6=48  9*6=54
2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49  8*7=56  9*7=63
2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64  9*8=72
2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

# while 문

```
....  
ResultSet rs=ps.executeQuery();  
while(rs.next()){  
    String writer=rs.getString("writer");  
}...
```

## ■ while 문

- 특정 조건을 주고 그 조건이 만족될 때까지 계속해서 반복시키는 것
- 반복 횟수가 가변적인 처리에 적합

## ■ while(반복 조건)

```
{  
    반복 명령;  
}
```

- ‘반복의 조건’이 만족되는 동안 ‘반복 내용’을 반복 실행함

```
Scanner sc = new Scanner("\ n\ n가나다 하나 둘\ n\ n라\ n\ n");  
String str="";  
while(sc.hasNext()){  
    str = sc.nextLine();  
    System.out.println("출력:"+str);  
}
```

- ‘반복조건’이 참인 동안은 계속 반복 수행된다
- while ~ 내부문장에서 조건이 거짓이 되도록 만들어서 loop를 빠져 나오게 해야 함
- 처음부터 조건이 거짓이면 반복되는 실행구문은 한번도 처리하지 않음.

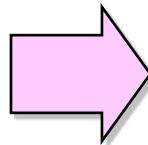
# while 문

## ■ while 문

### ■ 형식

```
while (조건식)
{
    명령문;
}
```

```
초기값;
while (조건식)
{
    명령문;
    증감식;
}
```

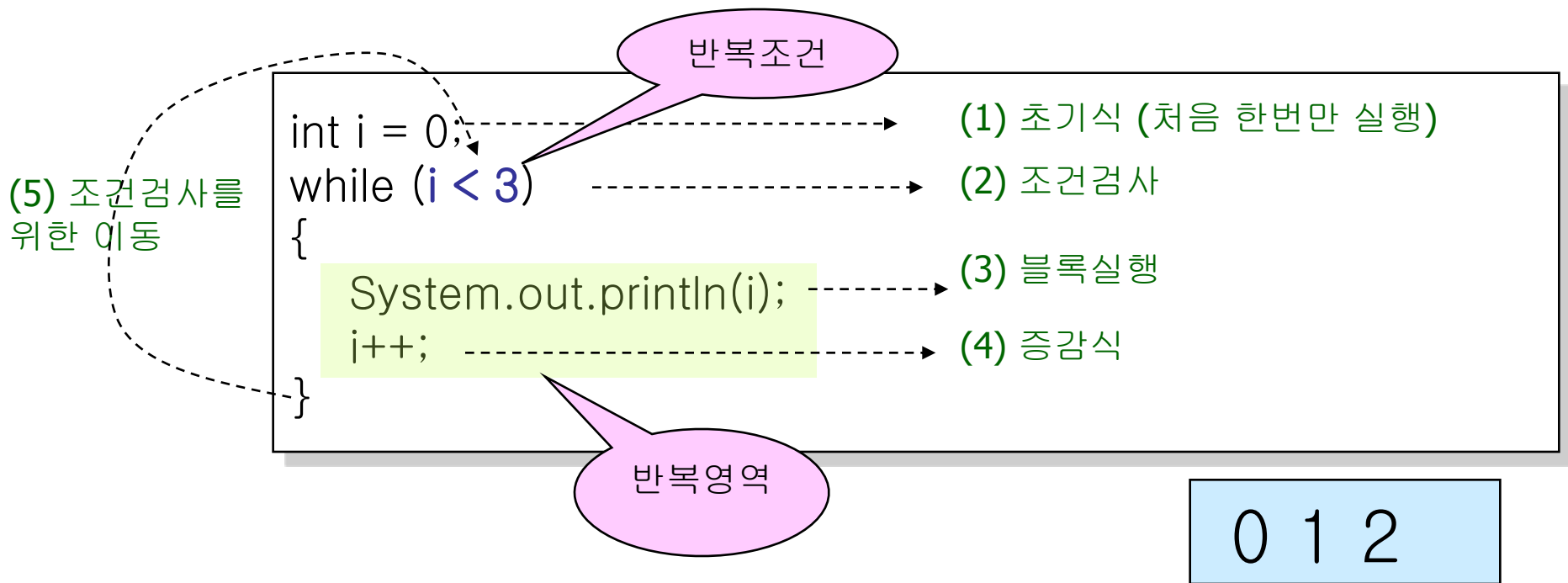


```
for (초기값; 조건식; 증감식)
{
    명령문;
}
```

# while 문

```
for(int i=0;i<3;i++)  
{  
    System.out.println(i);  
}
```

## ■ 조건이 만족하는 동안 반복 실행





## 예제 – while

```
class WhileTest1
{
    public static void main(String[] args)
    {
        int n = 1;

        while (n<3)
        {
            System.out.println("현재 n의 값 : " + n);
            n++;
        }
    }
}
```

```
현재 n의 값 : 1
현재 n의 값 : 2
```

## 무한 루프 : while (true)

```
import java.util.*;
class WhileTest2{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        while(true)
        {
```

```
            System.out.println("정수를 입력하세요! (끝낼때는 0 입력!);");
            int num = sc.nextInt();
            String str = "";
            if(num == 0)
                break;
            else if (num>0)
                str = "양수입니다.";
            else if (num<0)
                str = "음수입니다.";
```

```
            System.out.println(str + "\n");
```

```
        }//while
    }//main
```

```
}
```

```
정수를 입력하세요! <끝낼때는 0 입력!>
15
양수입니다.
정수를 입력하세요! <끝낼때는 0 입력!>
-79
음수입니다.
정수를 입력하세요! <끝낼때는 0 입력!>
36
양수입니다.
정수를 입력하세요! <끝낼때는 0 입력!>
0
계속하려면 아무 키나 누르십시오 . . .
```



# do while 문

- 선 실행 후 조건 처리

- 조건보다 명령이 먼저 온다
- 조건이 거짓이더라도 실행구문을 최소한 한번은 수행

```
do
{
    명령문;
}
while (조건식);
```

```
int i = 0;
do
{
    System.out.println(i); // 구문을 먼저 실행
    i++;
} while (i < 3);
```

```
0 1 2
```





## 예제 – do while

```
class DoWhileTest2
{
    public static void main(String[] args)
    {
        int n=1;
        do{
            System.out.println("현재 n의 값 : " + n);
            n++;
        }while (n>3) ;
    }
}
```

현재 n의 값 : 1

# 예제

```
import java.util.*;
class DoWhileTest
{
```

```
    public static void main(String[] args)
    {
```

```
        int num;
```

```
        Scanner sc = new Scanner(System.in);
```

```
        do
```

```
        {
```

```
            System.out.print("숫자를 입력하세요!(끝낼때는 0) ");
```

```
            num = sc.nextInt();
```

```
            System.out.println("입력한 값: " + num);
```

```
        }
```

```
        while (num != 0);
```

```
    }
```

```
}
```

```
( 숫자를 입력하세요!(끝낼때는 0) 7
  입력한 값: 7
( 숫자를 입력하세요!(끝낼때는 0) 0
  입력한 값: 0
  계속하려면 아무 키나 누르십시오 . . .
```

- 사용자로부터 입력받은 수를 보여주되, 0 이 입력되면 실행을 중지
- 입력 받은 값이 반복문의 조건식에 사용되기 때문에 먼저 입력을 받아서 계속 반복할 것인지를 결정할 수 있음

# enhanced for 문 (for each문)

- 일정한 개수로 구성된 특정 집합의 요소들을 반복 처리할 때 사용됨
- 배열의 모든 요소를 순회할 때가 가장 전형적임
- 배열 또는 컬렉션(해시, 리스트 등)에 데이터가 있는 만큼 반복

```
for (타입변수 : 배열)
{
    명령문;
}
```

```
int[] number= {10, 20, 30, 40};

for (int n : number)
{
    System.out.println(n);
}
```

제어변수 **n** 은 읽기 전용  
반복문 내에서 **n**의 값을 변경할 수 없음

10 20 30 40



## 예제 for

```
class ForeachTest
{
    public static void main(String[] args)
    {
        int[] num= {12, 43, 64, 56, 32};

        for (int n : num)
        {
            System.out.println(n);
        }
    } //main
} //class
```

```
12
43
64
56
32
```

```
String[] hobby= {"movie", "reading", "sports"};

for (String s : hobby)
{
    System.out.println(s);
}
```

```
movie
reading
sports
```



## 예제-컬렉션

---

```
import java.util.*;
```

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

```
for(int i =0 ; i<5; i++)
```

```
{
```

```
    list.add(i);
```

```
}
```

```
for(int n : list)
```

```
{
```

```
    System.out.println(n);
```

```
}
```

```
0  
1  
2  
3  
4
```

# 반복문

- while(반복 조건)

```
{  
    반복할 실행블록;  
}
```

- 조건이 참인 동안은 계속 반복 수행된다
- while ~ 내부문장에서 조건이 거짓이 되도록 만들어서 loop를 빠져 나오게 해야 함
- 처음부터 조건이 거짓이면 반복되는 실행구문은 한번도 처리하지 않음.

- do

```
{  
    실행블록;  
}  
while(반복 조건);
```

조건이 거짓이더라도 실행구문을 반드시 한번은 수행

- for(초기식;조건식;증감식)

```
{  
    실행블록;  
}
```

지정된 횟수만큼 반복해서 실행하는 구문

- for(자료형 변수이름 : 배열이나 컬렉션)

```
{  
    실행블록;  
}
```

- 배열이나 컬렉션에 있는 원소들을 차례대로 순회하는 반복문
- 참조타입의 loop를 돌때 사용

## 실습

- 사용자로부터 하나의 숫자를 입력 받아, 그 수만큼 3의 배수를 출력하시오

- 예) 사용자로부터 5를 입력 받았다면 3 6 9 12 15를 출력

```
3의 배수의 개수를 입력하세요
4
3      6      9      12
```

- for문 이용 - 계승(factorial)을 계산하는 프로그램 작성
  - 사용자로부터 n이라는 수를 입력 받으면 n!을 계산해서 출력해준다.

$(n! = 1*2*3*4*...*n)$

```
n의 계승 구하기 : n을 입력하세요
4
1~4까지의 곱<계승, factorial>: 24
```



## 실습

---

- while 이용 – 0과 100사이에 존재하는 짝수의 합 구하기

```
0~100까지 짝수의 합 : 2550
```

- while 이용 – 사용자로부터 입력 받은 숫자에 해당하는 구구단을 역순으로 출력하라

```
출력할 구구단의 단을 입력하세요  
7  
7 * 9 = 63  
7 * 8 = 56  
7 * 7 = 49  
7 * 6 = 42  
7 * 5 = 35  
7 * 4 = 28  
7 * 3 = 21  
7 * 2 = 14  
7 * 1 = 7
```



# 실습

- 1~99 사이에 있는 정수 중에서 7의 배수이거나 9의 배수인 정수를 출력하는 프로그램 작성
  - 7의 배수이자 9의 배수인 수는 한번만 출력되면 됨

```
7의 배수이거나 9의 배수
7      9      14      18      21      27      28      35      36      42
45     49     54     56     63     70     72     77     81     84
90     91     98     99     계속하려면 아무 키나 누르십시오 . . .
```

- 가로로 5개씩만 출력하기

```
7의 배수이거나 9의 배수
7      9      14      18      21
27     28     35     36     42
45     49     54     56     63
70     72     77     81     84
90     91     98     99     계속하려면 아무 키나 누르십시오
```

# 과제-중첩 for문

- 다음 결과를 출력하는 프로그램을 중첩 for문을 이용하여 작성하기

```
*  
**  
***  
****  
*****
```

```
*****  
*****  
*****  
****  
***  
**  
*
```

```
1 부터 10까지의 합  
1  
1+2  
1+2+3  
1+2+3+4  
1+2+3+4+5  
1+2+3+4+5+6  
1+2+3+4+5+6+7  
1+2+3+4+5+6+7+8  
1+2+3+4+5+6+7+8+9  
1+2+3+4+5+6+7+8+9+10
```

```
*-*-*  
*-*-*  
*  *-*  
*  *-*  
*-*  *
```



# 분기문

---

- 분기문 - 제어를 다른 위치로 옮기는 명령
  - **break** - 반복문이나 switch문의 case를 벗어날 때 사용
    - 무한 루프를 탈출할 때 종종 사용
  - **continue** - 루프의 나머지 뒷 부분을 무시하고 반복문의 선두로 점프하여 다음 루프를 실행
    - 다음 스텝으로 넘어간다
    - 다음 반복위치로 이동
    - 루프에서 특정 값에 대한 처리를 제외하고자 할 때 필요
  - **return** - 메서드의 실행을 종료하고 호출원으로 복귀함
    - 메서드의 처리 결과를 호출원으로 돌려주는 기능도 함

# break / continue 문

- break문 - 반복문을 빠져나감
- continue문 - 다음 반복위치로 이동

```
int i = 0;
while (true)
{
    System.out.println(i);
    i++;
    if (i < 3)
        continue;
    else
        break;
}
```

반복조건  
검사로 이동

while문  
탈출!

```
0
1
2
```



# break/continue

---

## ■ break

- switch와 반복문(while, do while, for)에서만 사용 가능
- 프로그램 블록 안에서 실행을 중단하고 다음 블록으로 넘어가고자 할 때 사용
- 블록 내에서의 실행을 중단하고 빠져 나옴

## ■ continue

- 반복문 안에서만 사용 가능
- 반복문의 반복을 한 번 건너뛰고 다음 반복을 실행할 때 사용



# 예제

```
class ContinueTest
{
    public static void main(String[] args)
    {
        for (int i=1;i<=10 ;i++ )
        {
            if(i ==5) continue;
            System.out.println("i=" + i);
        }
        System.out.println("-----");

        for (int i=1;i<=10 ;i++ )
        {
            if(i ==5) break;
            System.out.println("i=" + i);
        }
    }
}
```

```
i=1
i=2
i=3
i=4
i=6
i=7
i=8
i=9
i=10
```

```
-----
i=1
i=2
i=3
i=4
```

# 예제

```
class ContinueTest
{
    public static void main(String[] args)
    {
        for (int i=0; i<3; i++)
        {
            for (int j=0; j<3; j++)
            {
                if(j==1) break;
                System.out.println("i="+i+", j="+j);
            }
            System.out.println("-----");
            for (int i=0; i<3; i++)
            {
                for (int j=0; j<3; j++)
                {
                    if(j==1) continue;
                    System.out.println("i="+i+", j="+j);
                }
            }
        }
    }
}
```

```
i=0, j=0
i=1, j=0
i=2, j=0
```

```
-----
i=0, j=0
i=0, j=2
i=1, j=0
i=1, j=2
i=2, j=0
i=2, j=2
```

# 예제

12  
64  
56  
32

12

```
class BreakTest
{
    public static void main(String[] args)
    {
        int[] values = {12, 43, 64, 56, 32};

        for (int n : values)
        {
            if(n == 43)
                break;

            System.out.println(n);
        }
    }
}
```

```
class ContinueTest2
{
    public static void main(String[] args)
    {
        int[] values = {12, 43, 64, 56, 32};
        for (int n : values)
        {
            if(n == 43)
                continue;
            System.out.println(n);
        }
    }
}
```



# 이름 붙은 반복문

$2 * 1 = 2$
$2 * 2 = 4$
$2 * 3 = 6$
$2 * 4 = 8$

- 여러 반복문이 중첩되어 있을 때 반복문 앞에 이름 (Label) 을 붙이고 break 문과 continue문에 이름 (Label)을 지정해 줌으로써 하나 이상의 반복문을 벗어나거나 반복을 건너 뛸 수 있다

```
import java.util.*;
class LabelFor{
    public static void main(String[] args) {
        //for문에 Loop1 이라는 이름을 붙였다
        Loop1: for (int i=2;i<=9;i++){
            for (int j=1;j<=9 ;j++ ) {
                if (j==5)
                    break Loop1;
                //break;
                //continue Loop1;
                //continue;
                System.out.println(i+"*"+j+"="+i*j);

            }//안쪽 for
            System.out.println();
        }//end of Loop1
    }
}
```

```
class FlowEx27
```

```
{  
    public static void main(String[] args)
```

```
{
```

```
    // for문에 Loop1이라는 이름을 붙였다.
```

```
    Loop1 : for(int i=2; i <=9; i++) {
```

```
        for(int j=1; j <=9; j++) {
```

```
            if(j==5)
```

```
                ● break Loop1;
```

```
                ● break;
```

```
                continue Loop1; ●
```

```
                continue; ●
```

```
                System.out.println(i+"*"+ j +"="+ i*j);
```

```
            } // end of for i ←
```

```
        } System.out.println();
```

```
    } // end of Loop1 ←
```

```
    }  
}
```



## if 실습

- 년도를 입력 받아서, 짝수해인지, 홀수해인지 구하고, 윤년인지, 평년인지 구하여 출력하기
  - 윤년
    - 1) 400으로 나누어 떨어지면 윤년
    - 또는
    - 2) 4로 나누어 떨어지고, 100으로 나누어 떨어지지 않으면 윤년

```
년도를 입력하세요
2012
2012년 : 짝수 해, 윤년
```

```
년도를 입력하세요
2013
2013년 : 홀수 해, 평년
```



## switch 실습

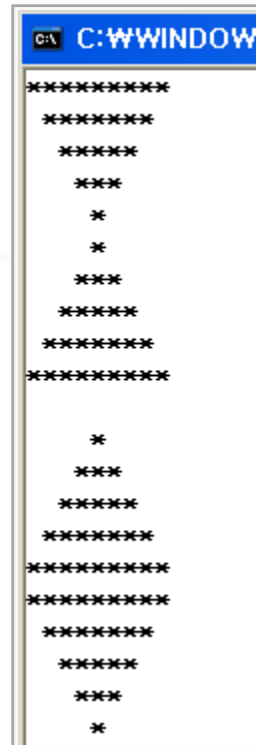
- 주민번호의 성별을 입력 받아서, 남자인지 여자인지 출력하기
  - 1 : 남자
  - 2 : 여자
  - 3 : 2000년 이후 출생한 남자
  - 4 : 2000년 이후 출생한 여자

주민번호 뒷자리의 성별에 해당하는 숫자를 입력하세요<1,2,3,4>  
4  
2000년 이후 출생한 여자

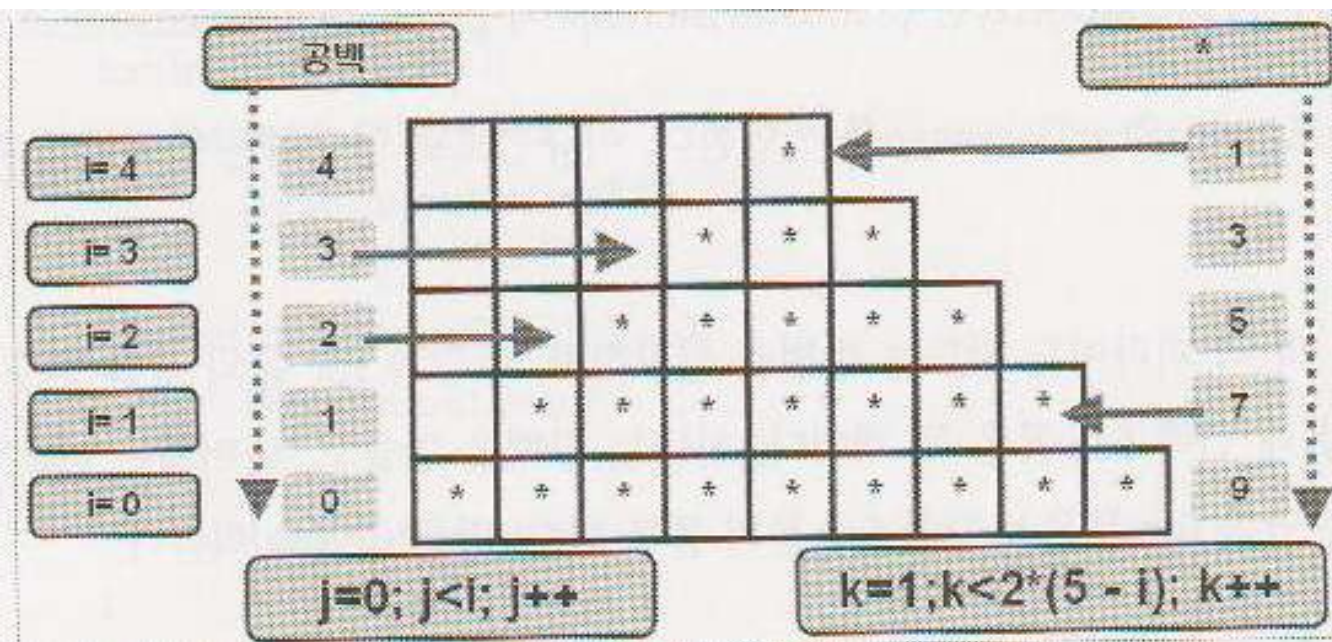
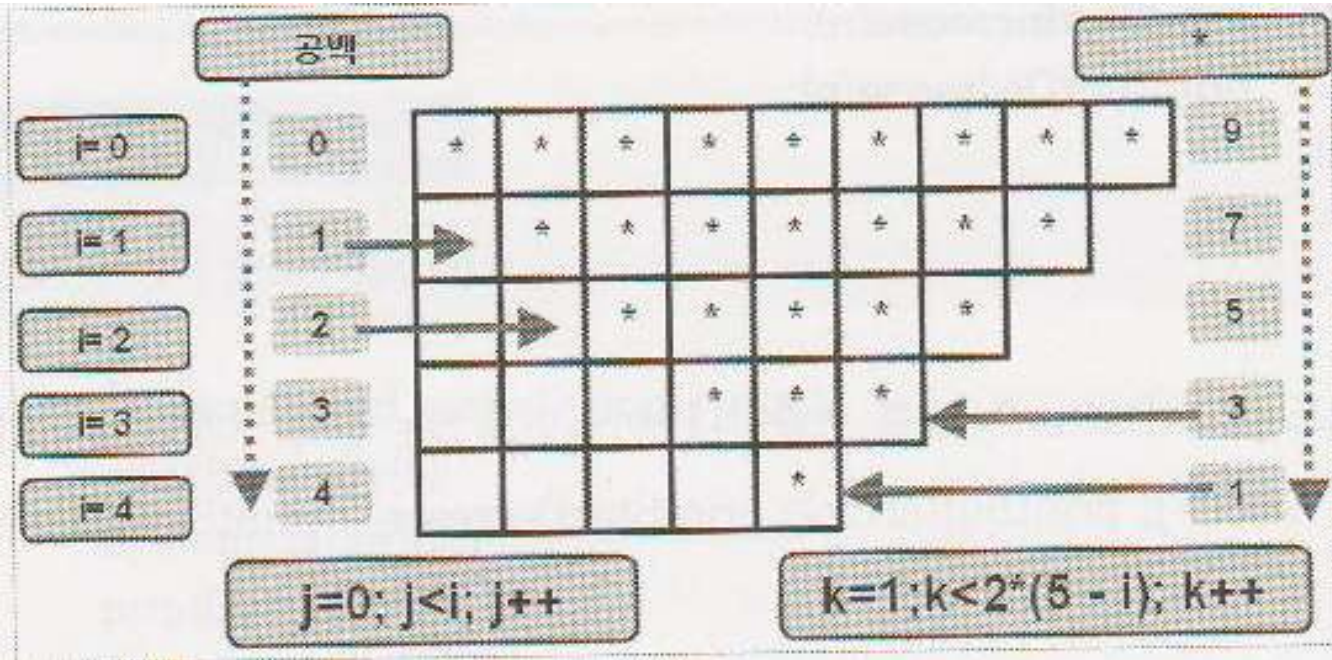
주민번호 뒷자리의 성별에 해당하는 숫자를 입력하세요<1,2,3,4>  
1  
남자

```
class StarPrint
{
    public static void main(String[] args)
    {
        //1. butterfly
        //star decrease
        for (int i=0;i<5 ;i++ )
        {
            for (int j=0;j<i ;j++ )
            {
                System.out.print(" ");
            }
            for (int k=1;k<2*(5-i) ;k++ )
            {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

```
//star increase
for (int i=5-1;i>=0 ;i-- )
{
    for (int j=0;j<i ;j++ )
    {
        System.out.print(" ");
    }
    for (int k=1;k<2*(5-i) ;k++ )
    {
        System.out.print("*");
    }
    System.out.println();
}
}
```









# 실습 - 증감연산자

---

```
class IncrementOp3
{
    public static void main(String[] args)
    {
        int i=6, k=6;

        System.out.println(i++);
        System.out.println(--i);
        System.out.println(++k);
        System.out.println(k--);
        System.out.println(--k);

        int m=i--;

        System.out.println("m="+m+", i=" +i +", k="+ k);
    }
}
```





# 실습

- 조건식으로 표현하시오
  - int 형 변수 i가 100보다 크고 300보다 작을 때 true인 조건식
  - char형 변수 ch가 공백이나 탭이 아닐 때 true인 조건식
  - char형 변수 ch가 'x' 또는 'X'일 때 true인 조건식
  - boolean형 변수 powerOn이 false일 때 true인 조건식
  - 문자열 참조변수 str이 "yes"일 때 true인 조건식
  - char형 변수 ch가 숫자('0'~'9')일 때 true인 조건식
  - char형 변수 ch가 영문자(대문자 또는 소문자)일 때 true인 조건식
  - int형 변수 year가 400으로 나누어 떨어지거나 또는 4로 나누어 떨어지고 100으로 나누어 떨어지지 않을 때 true인 조건식



## 실습 - if

---

- if 문 이용 – 월을 입력 받아 해당하는 사분기 출력
  - 1 ~ 3월 : 1사분기,
  - 4 ~ 6월 : 2사분기,
  - 7 ~ 9월 : 3사분기,
  - 10 ~ 12월 : 4사분기,
  - 나머지 : 잘못 선택했어요
- 성별이 1 이면 남자, 2이면 여자 출력
- 성별이 1 이나 3이면 남자, 2나 4이면 여자 출력



## 실습 - 반복문

73

totalSum=220

1+5=6

2+4=6

3+3=6

4+2=6

5+1=6

- 1부터 20까지의 정수 중에서 2 또는 3의 배수가 아닌 수(2의 배수도 아니고, 3의 배수도 아닌 수)의 총합을 구하시오.
- $1+(1+2)+(1+2+3)+(1+2+3+4)+\dots$   
 $+(1+2+3+\dots+10)$ 의 결과를 계산하시오.
- 두 개의 주사위를 던졌을 때, 눈의 합이 6 이 되는 모든 경우의 수를 출력하는 프로그램을 작성하시오.
  - 중첩 for문 이용

## 실습 - 반복문

```
4자리 이상의 숫자를 입력하세요
12345
sum=15

4자리 이상의 숫자를 입력하세요
12345
sum=15
```

- 숫자로 이루어진 문자열 `str`이 있을 때, 각 자리의 합을 더한 결과를 출력하는 코드를 완성하라. 만일 문자열이 "12345"라면, '1+2+3+4+5'의 결과인 15가 출력되어야 한다.
  - [Hint] String클래스의 `charAt(int i)`을 사용
  - 0의 유니코드는 48
- `int`타입의 변수 `num`이 있을 때, 각 자리의 합을 더한 결과를 출력하는 코드를 완성하라. 만일 변수 `num`의 값이 12345라면, '1+2+3+4+5'의 결과인 15를 출력하라.
  - [주의] 문자열로 변환하지 말고 숫자로만 처리해야 한다.
  - 숫자를 10으로 반복해서 나눠가면서, 10으로 나머지 연산을 하면 일의 자리를 얻어낼 수 있다.
  - 이 값들을 더하기만하면 변수 `num`에 저장된 숫자의 각 자리수를 모두 더한 값을 구할 수 있다.

num	num%10
12345	5
1234	4
123	3
12	2
1	1



## 실습-반복문

- 사용자로부터 입력 받은 문자열이 숫자인지를 판별하는 프로그램 작성하기
  - 반복문과 `charAt(int i)`를 이용해서 문자열의 문자를 하나씩 읽어서 검사한다.

```
값을 입력하세요  
123m5  
123m5는 숫자가 아닙니다.
```

```
값을 입력하세요  
1230  
1230는 숫자입니다.
```

# 실습

```
숫자를 입력하세요
1
숫자를 입력하세요
2
숫자를 입력하세요
3
숫자를 입력하세요
0
입력된 숫자의 합 : 6
```

- 1. 사용자로부터 0 이 입력될 때까지 계속해서 정수를 입력받아서 합을 구한다.
  - 0 이 입력되면, 지금까지 입력된 정수들의 합을 출력하는 프로그램 작성하기 (do~while 이용)
- 2. 입력된 정수의 전체 평균을 구하는 프로그램 작성하기
  - 먼저 입력할 정수의 개수를 사용자로부터 입력 받는다. 그리고 그 수만큼 정수를 입력 받아서, 입력 받은 수의 전체 평균을 계산하고 출력한다.
  - 입력 받은 값은 정수이지만, 출력되는 평균값은 실수가 되어야 한다
    - Math 클래스에서 반올림하는 메서드를 찾아서 이용
    - 소수 2째 자리까지 표현

```
입력할 정수의 개수를 입력하세요
3
정수를 입력하세요
77
정수를 입력하세요
80
정수를 입력하세요
96
입력된 정수의 전체 평균 : 84.33
```

## 실습

- 3. 사용자로부터 두 개의 정수를 입력 받아서 나눗셈을 하는 프로그램 작성하기
  - 두 개의 정수를 입력 받고, 나눗셈의 결과를 출력하는 일이 반복되어야 한다.
  - 두 개의 정수로 모두 0 이 입력되면 프로그램은 종료된다.
  - 정수형 나눗셈을 진행한다. 따라서 나눗셈의 결과는 몫과 나머지로 출력이 되어야 한다.
  - 제수(나누는 수)가 0 이면 나눗셈을 진행하지 못하니, 나눗셈 과정을 생략하고 다시 두 개의 정수를 입력 받도록 구현한다.
    - 무한 루프, break, continue 이용

```
두 개의 정수를 입력하세요<피제수 제수 순으로 입력>
15
6
몫 : 2, 나머지:3

두 개의 정수를 입력하세요<피제수 제수 순으로 입력>
77
0
제수가 0이므로 연산을 생략합니다.

두 개의 정수를 입력하세요<피제수 제수 순으로 입력>
0
0
```



## 실습

두 개의 정수를 입력하세요  
3  
5  
3~5사이의 정수의 합:12

- 4. 두 개의 정수를 입력 받아서, 입력 받은 두 정수와 그 사이에 존재하는 모든 정수들의 합을 계산하는 프로그램 작성하기
  - 예) 3과 5가 입력되면,  $3+4+5$  의 계산 결과가 출력되어야 한다
  - 단, 입력되는 숫자의 순서에 상관없이 동일한 결과가 출력되어야 한다. 즉 3과 5가 입력이 되건, 5와 3이 입력이 되건, 이에 상관없이  $3+4+5$ 의 계산 결과가 출력되어야 한다.
- 5. 1과 100 사이에 존재하는 모든 3의 배수와 5의 배수의 합을 계산하여 출력하는 프로그램 작성하기

3의 배수와 5의 배수의 합:2418



## 실습

- 6. 짝수 구구단만 (2,4,6,8단) 출력하는 프로그램을 작성하되, 2단은 2\*2까지, 4단은 4\*4까지, 6단은 6\*6까지, 8단은 8\*8까지만 출력하도록 구현한다
- 7. 5\*5 사각형 안에 숫자가 차례대로 들어가도록 출력하시오

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

2\*1=2  
2\*2=4  
  
4\*1=4  
4\*2=8  
4\*3=12  
4\*4=16

6\*1=6  
6\*2=12  
6\*3=18  
6\*4=24  
6\*5=30  
6\*6=36

8\*1=8  
8\*2=16  
8\*3=24  
8\*4=32  
8\*5=40  
8\*6=48  
8\*7=56  
8\*8=64



# 증감연산자 실습

---

```
class IncreTest
{
    public static void main(String[] args)
    {
        int i=3;

        System.out.println(i++ +", i++ => "+i);
        System.out.println(++i);
        System.out.println("i="+i);

        int k=++i;
        int n=i--;

        if(++i >= 10)
        {
            System.out.println("i는 10 이상!!");
        }

        System.out.println("i="+i+", k=" +k +", n="+ n);
    }
}
```