● 변수 : 기억장소(공간) , 메모리블럭 , 기억장소를 식별하기 위한 이름

int a, b; // 변수의 역할 : 변수에 값을 저장(기억) , 변수의 값을 이용

a=10;

b=a;

●자료형(공간의 **크기**와 공간의 내용을 **해석**하는 방법)

int : 정수형 , 4byte 예) int iNum=10; (byte:1byte, short:2byte , **int:4byte** , long:8byte)

char : 문자형 , 'a' ,'한' , 2byte 예) char ch='한' ;

double: 실수형 , 8byte 예) double dNum =20.4; (float:4byte , **double : 8byte**)

boolean: 논리형 (true =1, false=0) 예) boolean result= true;

변수값의 범위

```
byte b =127;        (1byte) : -128~127
short s  = 32767;   (2byte) : -32768 ~ 32767
int i=2147483647;   (4byte) :  -2147483648 ~ 2147483647
long l  = 9223372036854775807L ; // 922 3372 0368 5477 5807 L   (약922경)
```

● 상수: 변수를 상수화 (한 번 초기화 하고 값을 바꾸지 못하게 하겠다.) :상수화변수

```
final int MAX =10;     // 대문자로 표기함
                       // 가독성, 유지보수를 좋게 한다.
```

● 리터럴: 값 자체. 10, 10.2 , 'a' , "문자열" 값 자체를 리터럴 이라고함 (상수라고 부르기도 함)

=>리터럴이 저장되는 공간이 별도로 있음 (상수영역에 저장됨)

<정수의 기본은 int, 실수형의 기본은 double>

<값도 자료형을 가진다. 기억하기- 정수형: int , 실수형: double 으로 공간을 확보하고 저장된다>

```
long   lNum  =  892333L ;
int      iNUm  =334;
float   fNum  =  23.3f
float   fNum2  =  23.3;
```

● cast 연산자 : 임시로 자료의 형을 바꿀 때 사용 함(명시적인 형변환)

```
int a=10;
int b=20;
float f= 23.9f;

int   result1= a+f ;              =>불가능 (작은공간에 큰값이 못 들어감)
float   result2 = a+b ;           => 가능( 큰공간에 작은값이 들어감)
```

```
int a=10;
float f= 23.9f;
int   result=   a+ (int) f ;

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

```
int c=10;
int d=3;
int result2 = c / (double) d ;
System.out.println( result2);
```

주의: 정수와 정수의 연산의 결과는 정수입니다.

● 연산자

-단항연산자 : n++

-이항연산자 : num1 + num2

-삼항연산자 : result= (num1> num2) ? num1 : num2

▶ 대입연산자: =

int num=10;

▶ 부호연산자: + , -

num= -num;

▶ 산술연산자

: + , - , *, / (나누기) , %(나머지)

int sum = kor+ eng;

▶증감.감소 연산자:

++ , -- (전위형, 후위형)

num++ , ++num , --num , num--

▶ 관계연산자 :

>, < , >= , <= , == , !=

int myAge =27 ;

boolean value = (myAge >25);

System.out.println(value);

▶ 논리연산자

&& (논리곱) , || (논리합) , 부정(!)

int num=15;

boolean result = ((num >=10) && (num <= 30)) result의 결과는?

▶복합 대입 연산자

+= , -=, *= , /= , %=

num += 10; // num = num +10 과 동일

▶ 조건연산자(삼항연산자)

조건식**?** 결과1(참) **:** 결과2(거짓) =>조건식에 따라 결과1 또는 결과2가 실행됨

예)

int num = (15 > 3) ? 10 : 20;

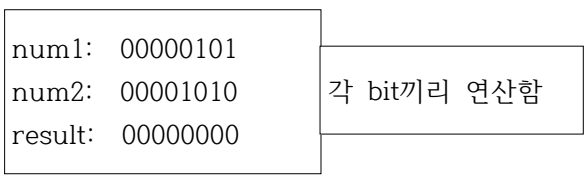
▶ 비트연산자

비트단위로 &, | , ^ , ~

예) int num1=5, num2=10;

int result = num1 & num2;

result의 예상값 ? 0입니다.



● 제어문(분기문, 반복문)

분기문: 조건에 따라 수행하는 명령문이 달라진다.

반복문 : 지정된 횟수만큼 또는 조건에 만족할 때 까지 반복한다

분기문

if(조건){ }

```
if( myAge > yourAge ) {
        System.out.println( "내가 형이야");
}
```

if(조건){

}else{

}

```
if( myAge > yourAge ) {
        System.out.println( "내가 형이야");
}else{
        System.out.println( "친구거나 너가 형이야");
}
```

if(조건){

}else if(조건){

}else if(조건){

}else{

}

```
if( myAge > yourAge ) {
        System.out.println( "내가 형이야");
}else if( myAge == yourAge){
        System.out.println( "우린 친구");
}else{
        System.out.println("너가 형이야");
}
```

// 차이

if(조건){ }

else if(조건) { }

// 둘의 차이 ??

if(조건){ }

if(조건){ }

```
if( score >= 90 ) {
        System.out.println( "학점은 A");
}
if( score >= 80){
        System.out.println( "학점 B");
}
```

//대상의 값은 숫자나 문자열도 가능함.

switch(대상)

{

case 값1:

case 값2:

case 값3:

default:

}

```
Scanner sc = new Scanner(System.in);
int menu=sc.nextInt();
switch( menu){
      case 1 : System.out.println("등록선택 하셨어요");
                break;
      case 2 : System.out.println("변경선택 하셨어요");
                break;
      case 3 : System.out.println("삭제선택 하셨어요");
                break;
      case 4 : System.out.println("조회선택 하셨어요");
                break;
      default:   System.out.println("메뉴가 잘못선택");
}
```

//반복
for(초기값: 조건: 증감){
    반복할 명령;

}

```
for( int i= 2 ; i<10 ; i++){
      System.out.println( i ) ;
}
```

while(조건) {
    반복할 명령;

}

```
int n=0, sum=0;
while( n < 10){
    n++;
    sum += n;
}
System.out.println( sum ) ;
```

do{
    반복할 명령;

}while(조건);

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

do{
    System.out.println( "점수 입력");
    int score = sc.nextInt();
}while( score >100 ||  score <0);
System.out.println( "점수는=" + score );
```

다중 for문
<구구단, 별찍기 연습>
(반복문 연습)

```
for( int i= 2 ; i<10 ; i++){

    for( int j=1 ; j<10; j++){
        System.out.println( i+ "*" +j +"=" + i *j );
    }
}
```

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

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

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

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

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

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

반복문에서 쓰이는

● break; // 반복구조를 빠져나올 때

```
int  n=0;
while(true){
    n++;
    sum  +=  n;
    if(  num   >=10  )  break;
}
System.out.println(  sum  );
```

● continue; // 반복구조에서 다음 반복으로 넘길 때

```
public  class  ExContinue  {

        public  static  void  main(String[]  args)  {
                int  total  =0;
                for(int  i=1;  i<=100;  i++)  {
                        if(  i  %2  ==0){
                            continue;
                        }
                        total  +=  i;
                }
                System.out.println("1~100까지  홀수의  합="+  total);
        }
}
```