22. 09 .28 - Variable , Scanner , 연산자

블레이스 { }

코드들을 하나의 묶음으로 묶을 때 (파이썬의 들여쓰기와 같다)

Variable

1) Member variable

2) Local variable (지역)

점근범위와 생명주기는 맞물려서 돌아간다.

접근범위

범위(변수를 사용할 수 있는 범위) 는 시작점과 끝점이 있다.

→ 블레이스가 닫히면 접근할 수 있는 범위가 닫힌다.

생명주기

지역변수 기준으로 마지막 블레이스까지 실행이 되면 메모리 상에서 없어져 생명주기가 끝이 나버린다. 이후에는 블레이스 안의 변수를 사용 할 수 없다.

STDIO

입력은 키보드 마우스 아웃풋은 모니터

Standard

	Input	Output
파이썬	input ( )	print
자바	`Scanner`	`System.out.printx()`

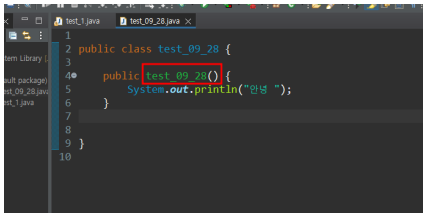
`ctl + Shif + O` → **해당되는 import 자동으로 작성**

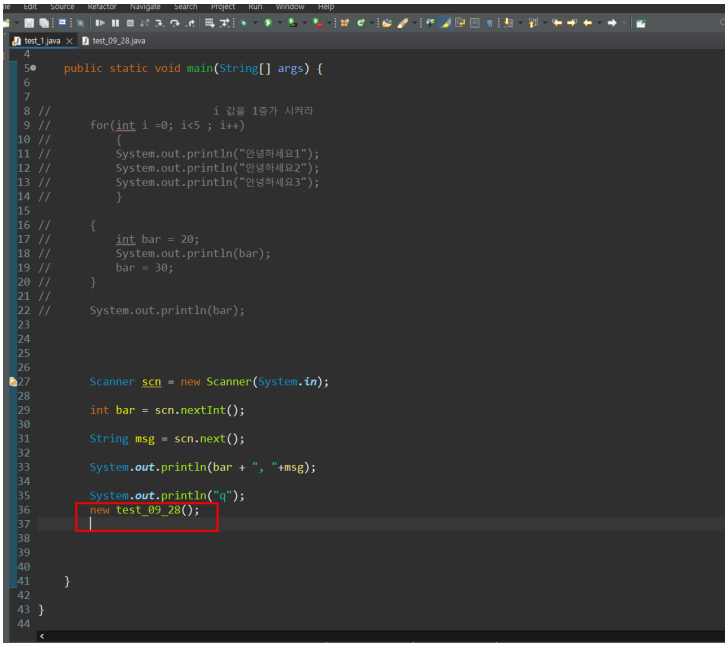
초기 설정함수

public 현재의 클래스명 ( )

```
public class test_09_28 {

    public test_09_28() {
        System.out.println("안녕 ");
    }


}
```

불러와서 출력할 수 있다.





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


//                        i 값을 1증가 시켜라
//      for(int i =0; i<5 ; i++)
//          {
//          System.out.println("안녕하세요1");
//          System.out.println("안녕하세요2");
//          System.out.println("안녕하세요3");
//          }

//      {
//          int bar = 20;
//          System.out.println(bar);
//          bar = 30;
//      }
//
//      System.out.println(bar);



        Scanner scn = new Scanner(System.in);

        int bar = scn.nextInt();

        String msg = scn.next();

        System.out.println(bar + ", "+msg);

        System.out.println("q");
        new test_09_28();


    }
}
```

클래스

메모리 상에서 틀로 만들어 놓는 것 → 메모리상에서 찍어 넣는것

변수와 **함수**가 하나의 클래스 안에 들어간다 → 이걸 찍어 내는 것이다 → 찍어 낸것이 메모리상에 올라간다

new 뒤에는 클래스 이름이 와야한다.

new 붕어빵틀에서 하나 구워줘

변수 앞에 클래스 이름이 나오면 레퍼런스 벨류이다. 레퍼런스벨류 는 무조건 4바이트!!!!

void : 함수의 자료형이 없다 → 반환값(형)이 없다

연산자(Operator)

연산자 (Operator)

연산자(Operator)란?

- **연산자(Operator)**
  - 어떠한 기능을 수행하는 기호(+,-,*,/ 등)
- **피연산자(Operand)**
  - 연산자의 작업 대상(변수,상수,리터럴,수식)

```
int a = 2 + 3
```

연산자 우선순위 (1)

종 류	연산방향	연산자	우선순위
단항 연산자	←	++ -- + - ~ ! (타입)	높음
산술 연산자	→	* / % 모듈레이션(모듈러 연산자)	
	→	+ -	
	→	<< >> >>>	
비교 연산자	→	< > <= >= instanceof	
	→	== !=	
논리 연산자	→	&	
	→	^	
	→	|	
	→	&&	
	→	||	
삼항 연산자	→	?:	
대입 연산자	←	= *= /= %= += -= <<= >>= >>>= &= ^= |=	낮음

[표3-1] 연산자의 종류와 우선순위

연산자 공부 時 3가지 관점으로 나눈다.

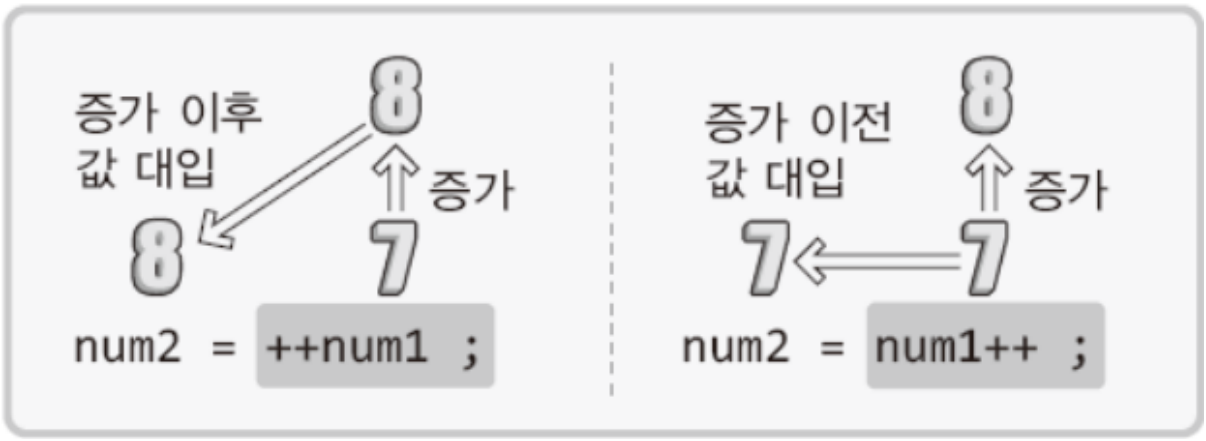
1. 기능별로 어떤 연산자들이 있는지 (산술 , 비교 , 논리 , 대입 등)
2. 연산자의 우선순위 (우선순위 굳이 맞추지 말고 **괄호 사용**하기)
   - 단항 연산자 - 最高
   - 대입 연산자 - 最低
3. 항의 개수(피연산자 개수)
   - **이항 연산시 자료형의 우선순위가 어떻게 되는지**
     (좌항과 우항이 다를 시 피연산자의 타입을 일치 시키기)

> 수식(익스프레션)

💡 단항 연산자

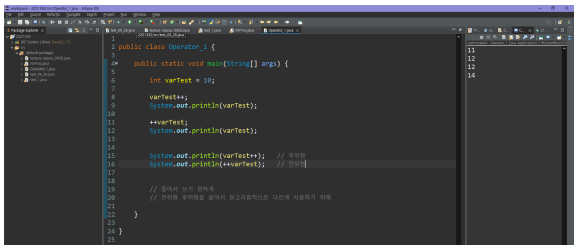
증가연산자 (++) : 피연산자의 값을 1 증가

감소연산자 (- -) : 피연산자의 값을 1 감소



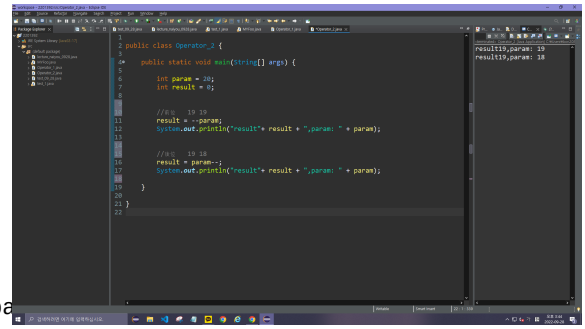
전위 후위

```
public class Operator_1 {

  public static void main(String[] args) {

    int varTest = 10;

    varTest++;
    System.out.println(varTest);  // 11

    ++varTest;
    System.out.println(varTest);   // 12


    System.out.println(varTest++);   // 후위형  [(한줄);가 끝나면 실행]  // 12
    System.out.println(++varTest);   // 전위형                              // 14

    // 사용 하는 이유
    // 줄여서 쓰기 편하게
    // 전위형 후위형을 붙여서 알고리즘적으로 다르게 사용하기 위해

  }

}
```



```
public class Operator_2 {
```

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

    int param = 20;
    int result = 0;


//  前位 19 19
    result = --param;
    System.out.println("result"+ result + ",pa


//  後位 19 18
    result = param--;
    System.out.println("result"+ result + ",param: " + param);

  }

}
```



> 後位 연산자는 전체 수식이 다 끝나고 나면 적용 한다

자바는 `Truthy` / `Falsey` 적용 하지 않는다 !!

! 는 `Truthy` `Falsey`를 서로 바꿀 때 사용

`Scanner` ⇒ Scanner scn = new Scanner(System.in); [Ctl + Shift + O]

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

      int bar1 = scn.nextInt(); // int 정수

      long bar2 = scn.nextLong(); // long 정수

      String msg = scn.next(); // 문자열

      float fTemp = scn.nextFloat(); // float 실수
```

```
        double dTemp = scn.nextDouble(); // double 실수

        System.out.println(bar1 + ", " + msg);
```

! 변수 → 변수의 반대