23-12-19

- Java

```
no usages
public class Car {                    필드
    no usages
    String company = "Hundia";
    no usages
    String model;
    no usages
    String color;
    no usages
    int speed;
```

미리 지정된 아이들 필드라고 부른다 -> 자료값 지정하는 곳

```
1 usage
public Korean(String n, String s){ //public 없어도 가능하다!! //함수=met
     name=n; //String 위에서 선언해줬는데?(static으로 했을 때)
     ssn=s;
 }
```

메소드 -> 클래스 안에 들어가 있는 것

- Korean/KoreanExm

```
public class Korean {
    no usages
    String nation="대한민국";
    2 usages
    String name;
    2 usages
    String ssn;

    1 usage
    Korean(String n, String s){ //public 없어도 가능하다!! //함수=method
        name=n;
        ssn=s;
    }
```

String nation="대한민국" : 선언과 초기화를 동시에 한 것.

Korean있는 자리에 Public을 써주는데 써주지 않아도 실행 가능하다

=> 생성자 이기 때문에, 생성자로 Public을 써줘도 생성자이다.

```
public class Korean {
    no usages
    String nation="대한민국";
    2 usages
    static String name;
    2 usages
    String ssn;

    1 usage
    static Korean(String n, String s){ //public 없어도 가능하다!! //함수=method
        name=n;
        ssn=s;
    }
```

Korean에 static으로 선언해줬다 그러면 위에 String name해준 것보다 더 위로 올려준 것이다

=> 그래서 실행이 되지 않음.

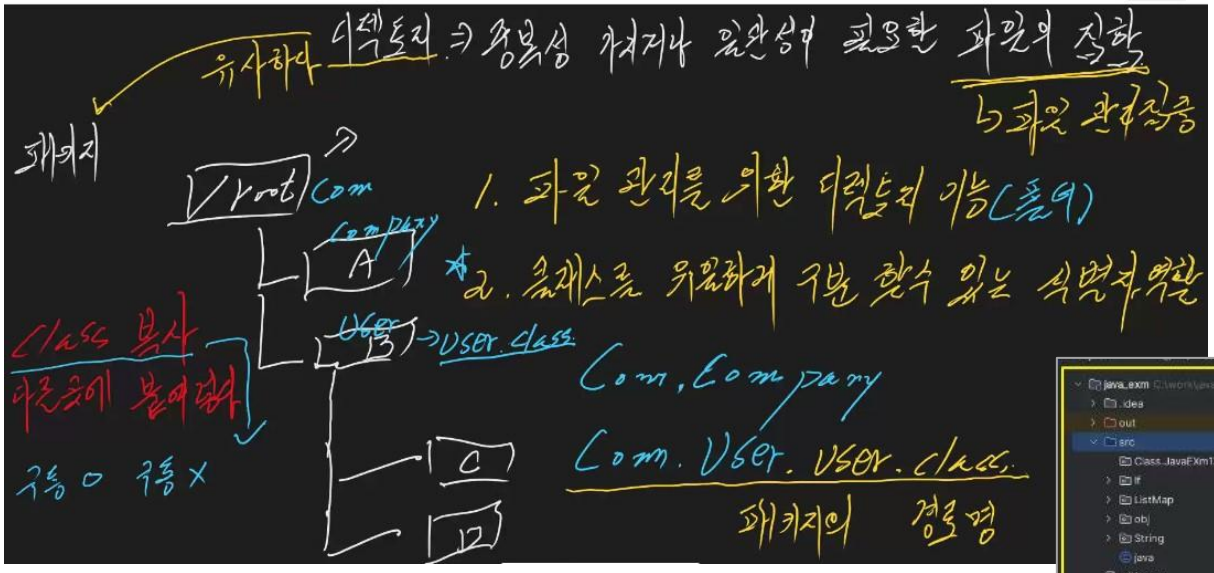
위에도 Static String name으로 써주면 실행이 가능하다. (저 선언한 것을 올려줬기 때문에 ssn으로 해준 것은 위에 String ssn에 static으로 써주지 않았기 때문이다. 이럴 때는 저 Korean안에 String으로 선언 해줘야한다.

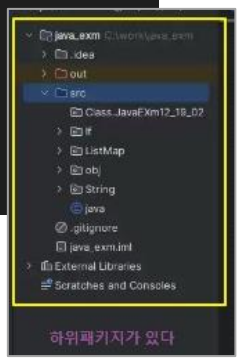
실행순서 : KoreanExm에서 main 먼저 실행->Korean생성자 -> Korean클래스

- Car/CarExm

```java
public class CarExm {
    public static void main(String[] args) {
        Car car1=new Car();
        System.out.println(car1.company);
        System.out.println();

        Car car2=new Car();
        System.out.println(car2.company);
        System.out.println(car2.model);

        Car car3=new Car();
        System.out.println(car3.company);
        System.out.println(car3.model);

        Car car4=new Car();
        System.out.println(car4.company);
        System.out.println(car4.model);
        System.out.println(car4.maxSpeed);
    }
}
```

Car이라는 class를 받아와서 인스턴스 생성하는 것
하나의 클래스로 여러 개의 인스턴스를 생성하여 사용해준 것.





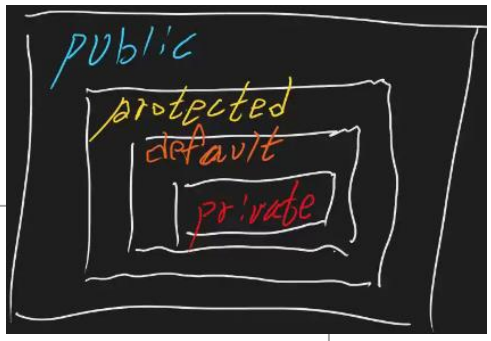
패키지 : 디렉토리랑 유사함,

1. 파일 관리를 위한 디렉토리 기능
2. 클래스를 유일하게 구분할 수 있는 식별자 역할

디렉토리 : 중복성 가지거나 일관성이 필요한 파일의 집합->파일 관리집중

Class복사 : 다른곳에 붙여넣다 -> 패키지안에 상속되어 있는 것이기 때문에 문제 생길수도 있음

- 객체 / 패키지 / public / import / class



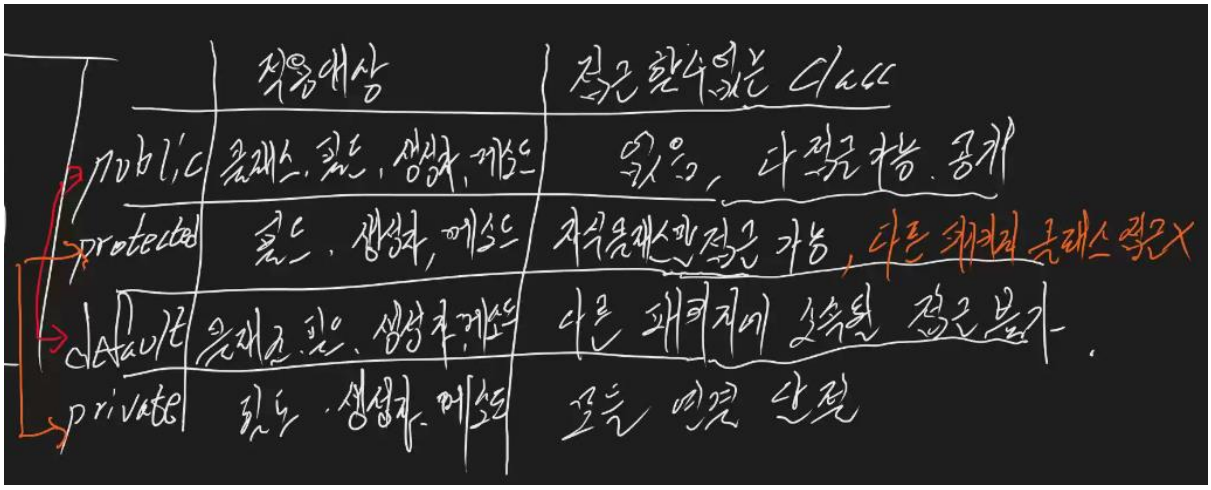
public > protected > default > private

- **public** : 접근을 제한이 없음
- **protected** : 동일한 패키지 내에 존재하거나 파생클래스에서만 접근 가능
- **default** : 아무런 접근 제한자를 명시하지 않으면 default 값이 되며, 동일한 패키지 내에서만 접근이 가능
- **private** : 자기 자신의 클래스 내에서만 접근이 가능

마무리! 표로 한 번에 정리하기

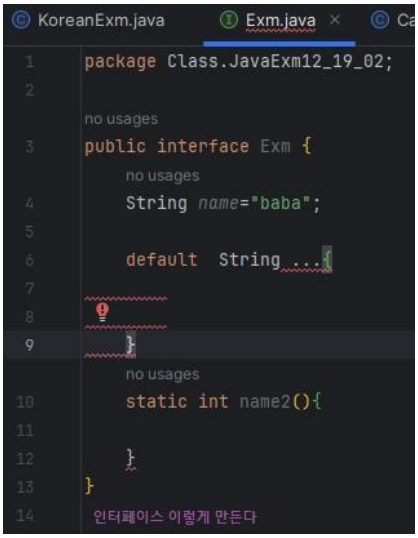
접근 제한자: 클래스와 인터페이스를 다른 패키지에서 사용하지 못하도록 막을 필요가 있습니다. 그리고 객체 생성을 막기 위해 생성자를 호출하지 못하게 하거나 필드나 메소드를 사용하지 못하도록 막아야 되는 경우도 있습니다. 이때 접근 제한자를 사용할 수 있습니다.

접근 제한	적용 대상	접근할 수 없는 클래스
public	클래스, 필드, 생성자, 메소드	없음
protected	필드, 생성자, 메소드	자식 클래스가 아닌 다른 패키지에 소속된 클래스
default	클래스, 필드, 생성자, 메소드	다른 패키지에 소속된 클래스
private	필드, 생성자, 메소드	모든 외부 클래스



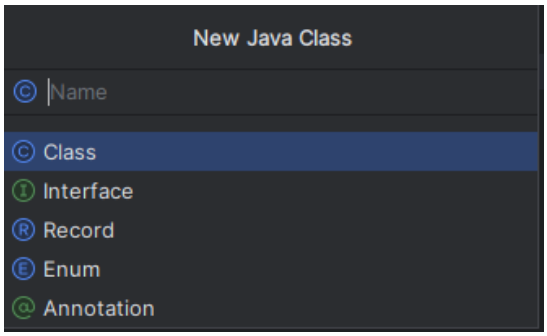
주로 public과 private를 사용함

- 인터페이스(interface)



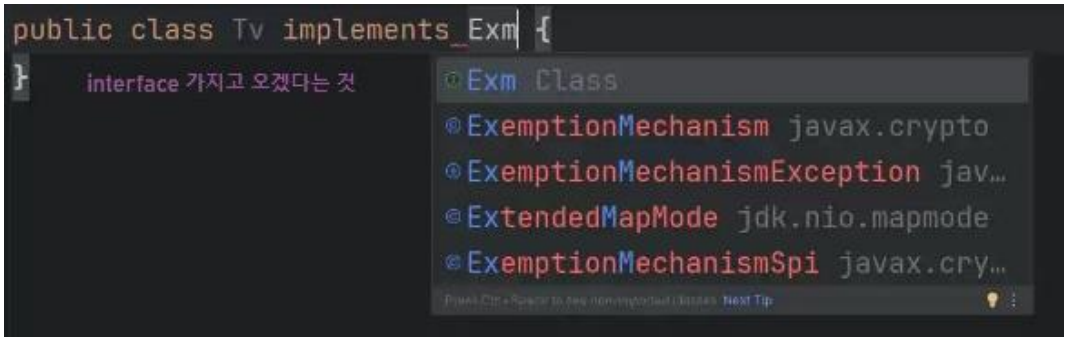
Interface : 객체간 교환성 증대

객체는 클래스도 변수도 메소드도 객체로 만들어서 불러올 수 있다

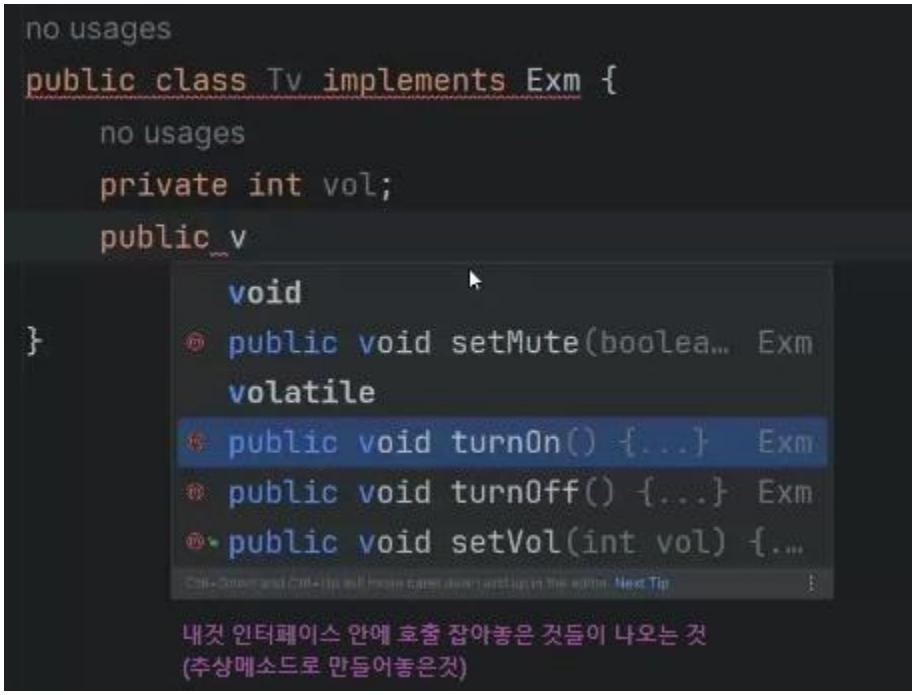


여기에서 interface를 선택해서 사용한다

Annotation은 백엔드에서 사용하는 것.



java파일이다 -> interface로 만들어놓은 Exm을 가지고와서 Tv class를 만들겠다는 것

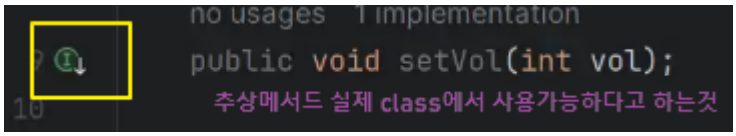


public v라고 입력하면 인터페이스 안에 호출로 선언한 것들이 나온다

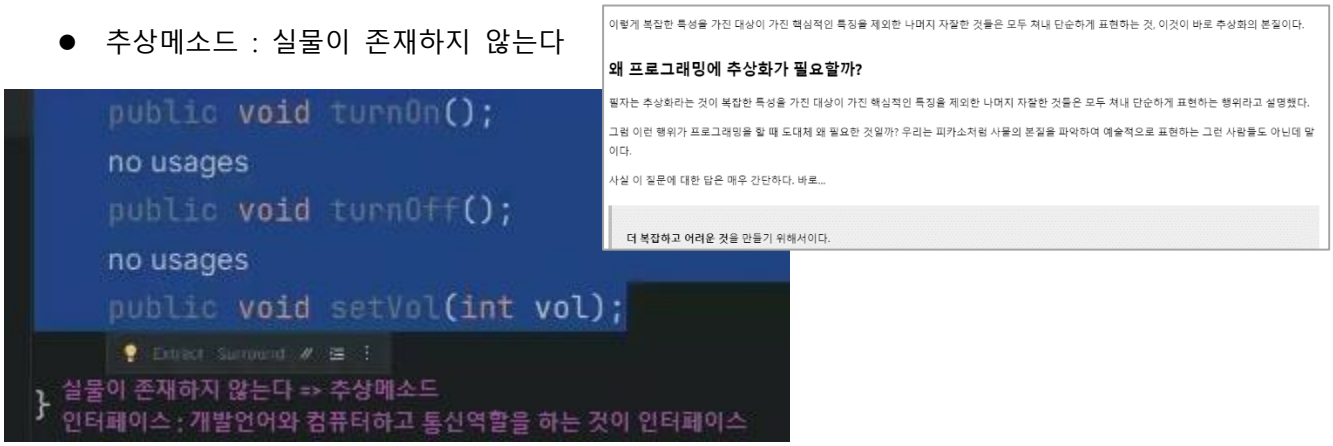
> 인터페이스(interface)는 서로 다른 두 개의 시스템, 장치 사이에서 정보나 신호를 주고받는 경우의 접점이나 경계면이다. 즉, 사용자가 기기를 쉽게 동작시키는데 도움을 주는 시스템을 의미한다. 컴퓨팅에서 컴퓨터 시스템끼리 정보를 교환하는 공유 경계이다.

인터페이스 만드는 이유: 어디서든 사용 가능, 클래스의 대안

인터페이스(interface) : 개발언어와 컴퓨터하고 통신역할을 함



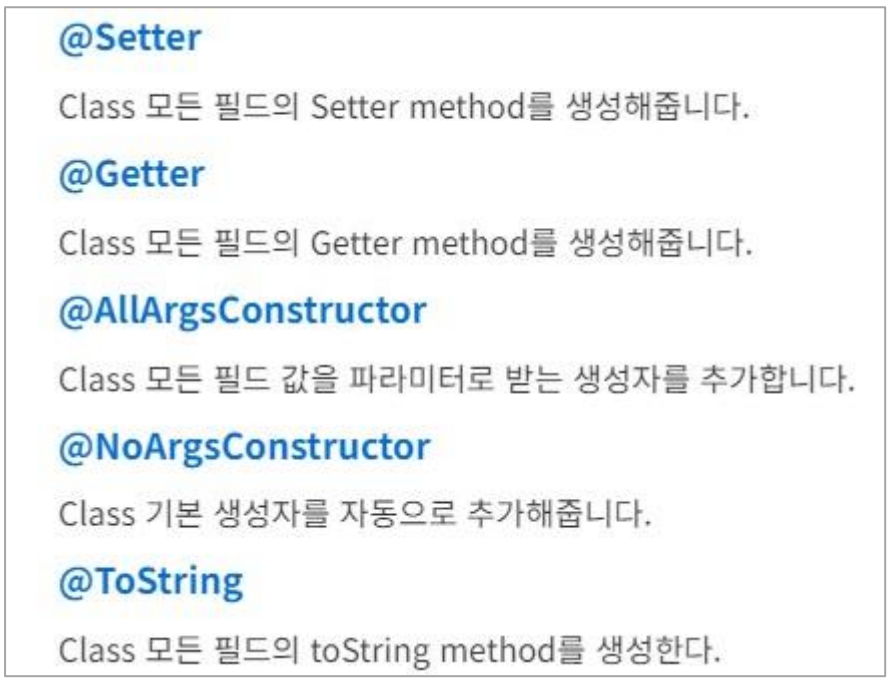
- 추상메소드 : 실물이 존재하지 않는다



(JavaExm12_19_02패키지 안에 Exm인터페이스, Tv자바)

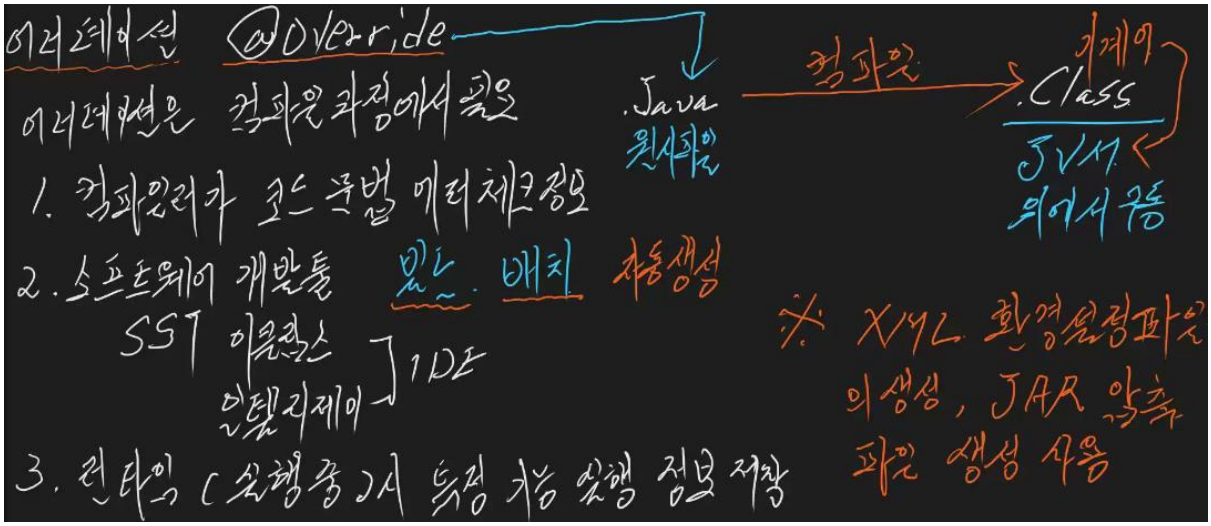
- 어노테이션(annotation)
  : 에플리케이션이 처리할 자료가 컴파일과 실행 과정상에서 코드를 어떻게 컴파일 하고 처리할지에 대한 정보

어노테이션(메타데이터) 종류 중 하나이다.



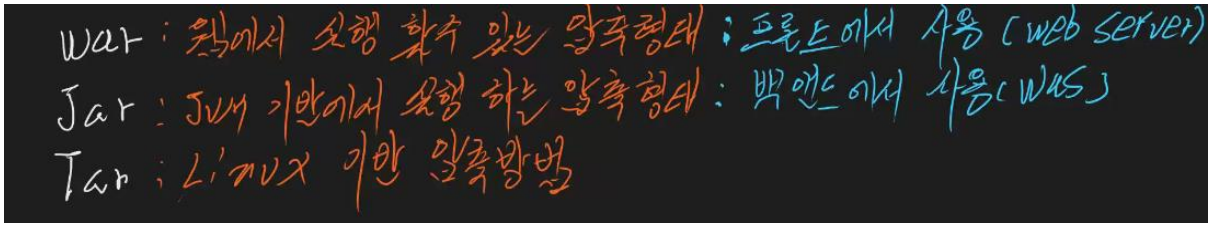
Setter 는 외부의 값을 받아와 필드의 값을 변경

Getter 는 외부로 필드 값을 전달



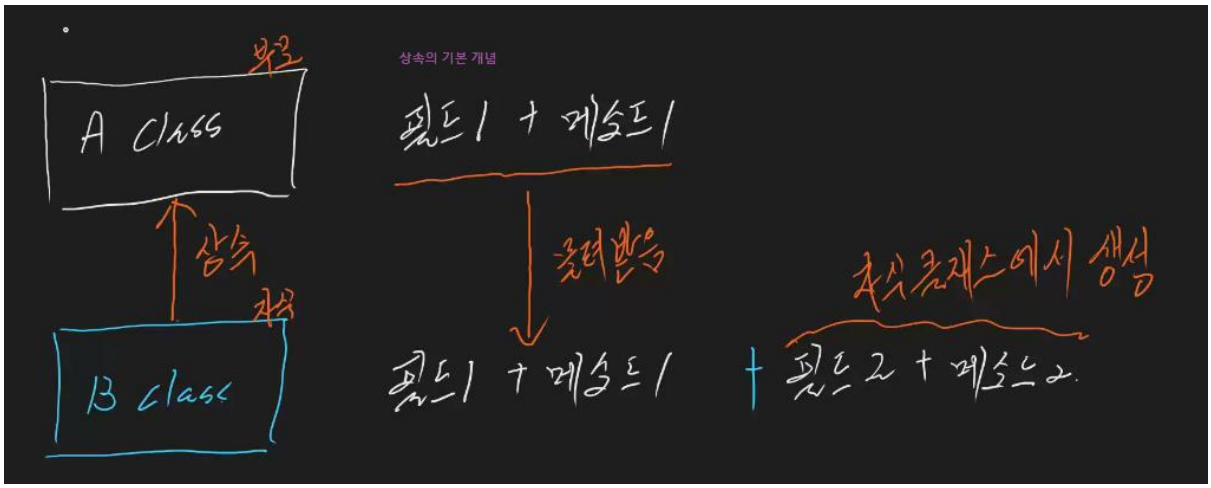
어노테이션 실제 작업할 내용이 있는 것이다, 컴파일 과정에서 필요한 것이다

Java(원시파일) -> class(JVM 위에서 구동)
: 자바는 자체적으로 컴파일 한다



War: 웹에서 실행할 수 있는 압축형태(거의 프론트에서 사용, web server 에 필요한 것이 war)
Jar: JVM 기반에서 실행하는 압축형태(백단에서 사용, was 에 필요한 것이 jar)
Tar: Linux 기반 압축방법

- 상속의 기본개념



study

```
class Animal {
    String name;

    void setName(String name) {    public를 쓰지 않았지만 class가 public이니까 public인가
        this.name = name;
    }
}

class Dog extends Animal {  // Animal 클래스를 상속한다.
}

public class Sample {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.setName("poppy");
        System.out.println(dog.name);
    }
}
```

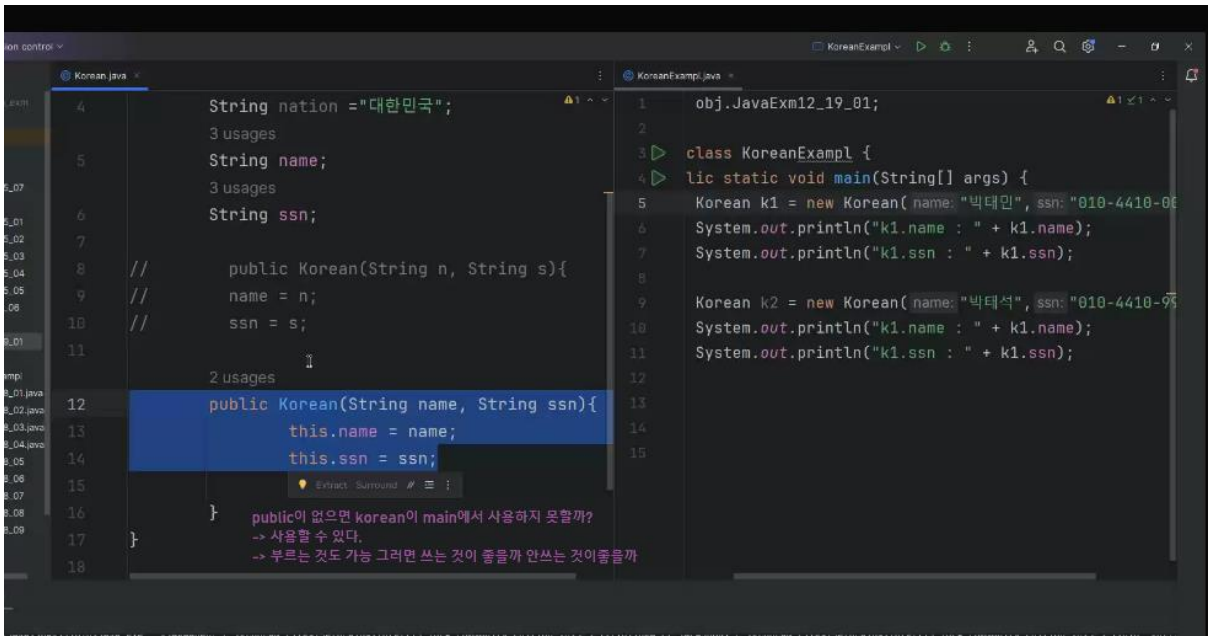
https://wikidocs.net/280

this.name=name;

this name은 String에 name을 받은 것이고,

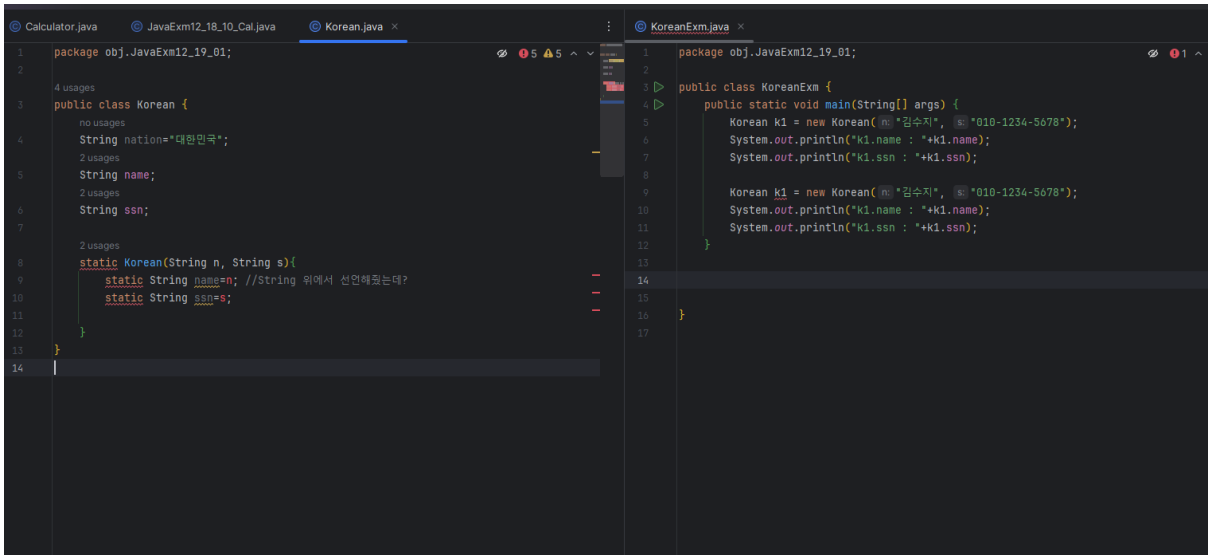
=name은 main에 쓴 poppy가 들어오는 것

println을 하게 되면 dog.name



저기 Korean앞에 public 붙이지 않아도 실행이 된다

그러면 public이 defalut값으로 변경이 되어서 동일한 패키지 내에서는 접근이 가능한것



오버로딩/오버라이딩

Import