23-12-18

- JAVA
- 능력단위평가(DB 재시험)

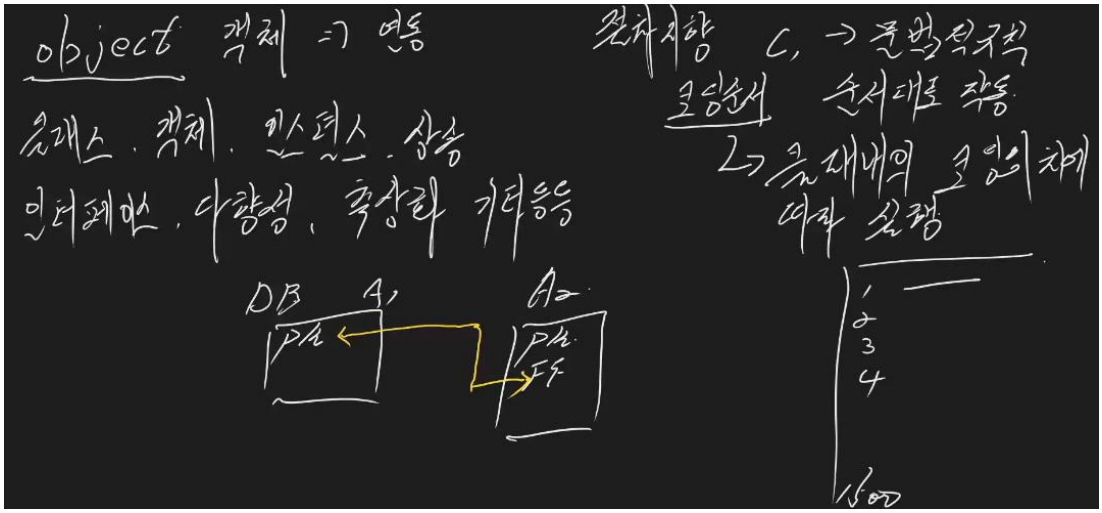
<문제>

1. 총 열개의 데이터를 배열로 입력 받고 평균을 구하시오
   각각의 입력값이 60점 미만인 경우 불합격 처리하고 합격과 불합격을 출력하시오.

```java
public class JavaExm12_18_01 {
    public static void main(String[] args) {

        ArrayList<Integer> num = new ArrayList<>(Arrays.asList(70,60,55,75,98,80,80,95,100)); //배열을 문자열로 저장
        System.out.println(num);

        int sum=0;
        for(int i=0;i<num.size();i++){
            sum+=num.get(i);
            // sum += num[i];
            System.out.println(sum);
            if (num[i]<60) System.out.println("입력값이 60점 미만으로 불합격입니다.");
            else System.out.println("입력값이 60점 이상으로 합격입니다.");
        }
        float avg=0;
        avg=(float)sum/num.size();
        System.out.println("평균"+avg);
        if(avg < 60) System.out.println("평균이 60점 이상입니다.");
        else if(avg>=60) System.out.println("평균이 60점 미만입니다.");
    }
}
```



- **절차지향(c, 문법적규칙)**

코딩 위치에 따라 실행-> 1~1500번까지 진행할 때 순서대로 진행한다는 것 중간에 건너 뛸 수 없다는 것

- **객체지향**

중간에 필요하지 않는 데이터 사용하지 않도록 해야 한다

객체가 나오면 class, 객체, 인스턴스, 상속, 다양성, 추상화

객체 지향(Object oriented) 프로그래밍이란? # 우리가 실생활에서 쓰는 모든 것을 객체라 하며, 객체 지향 프로그래밍은 프로그램 구현에 필요한 객체를 파악하고 각각의 객체들의 역할이 무엇인지를 정의하여 객체들 간의 상호작용을 통해 프로그램을 만드는 것을 말한다. 2014. 9. 14.

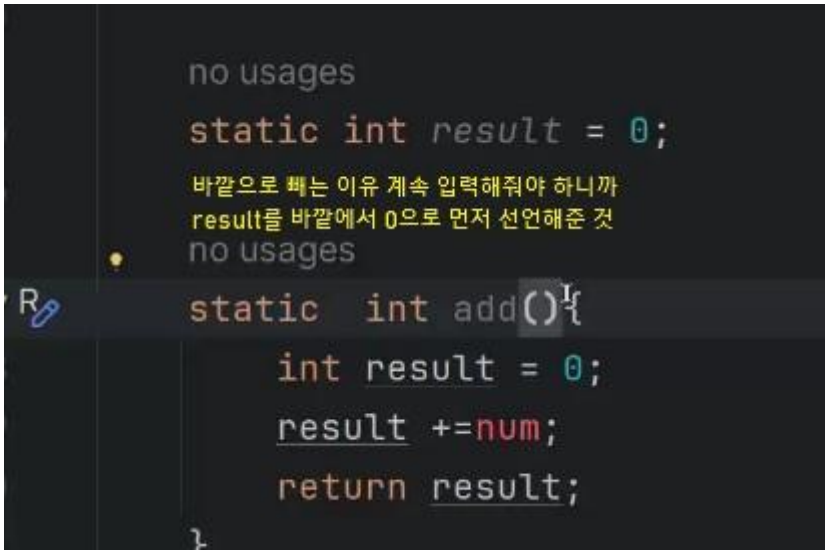
A2는 A1에 종속되어 있는 것이다

A1은 부모 class, A2는 자식 class.

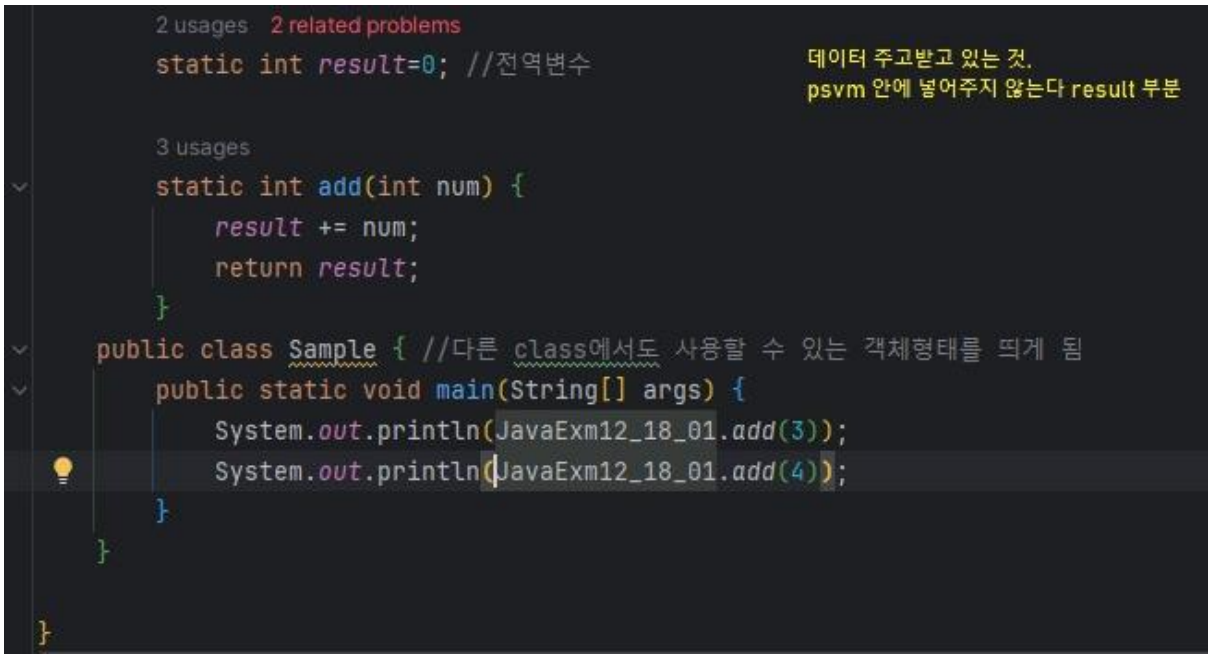
부모에 있는 것을 자식이 사용할 수 있다(포함되어 있다) 그것이 상속이다

(각각 만들어지는 것들이 연동되어 가는 것이 객체지향언어의 시작)

https://www.incodom.kr/%EA%B0%9D%EC%B2%B4_%EC%A7%80%ED%96%A5

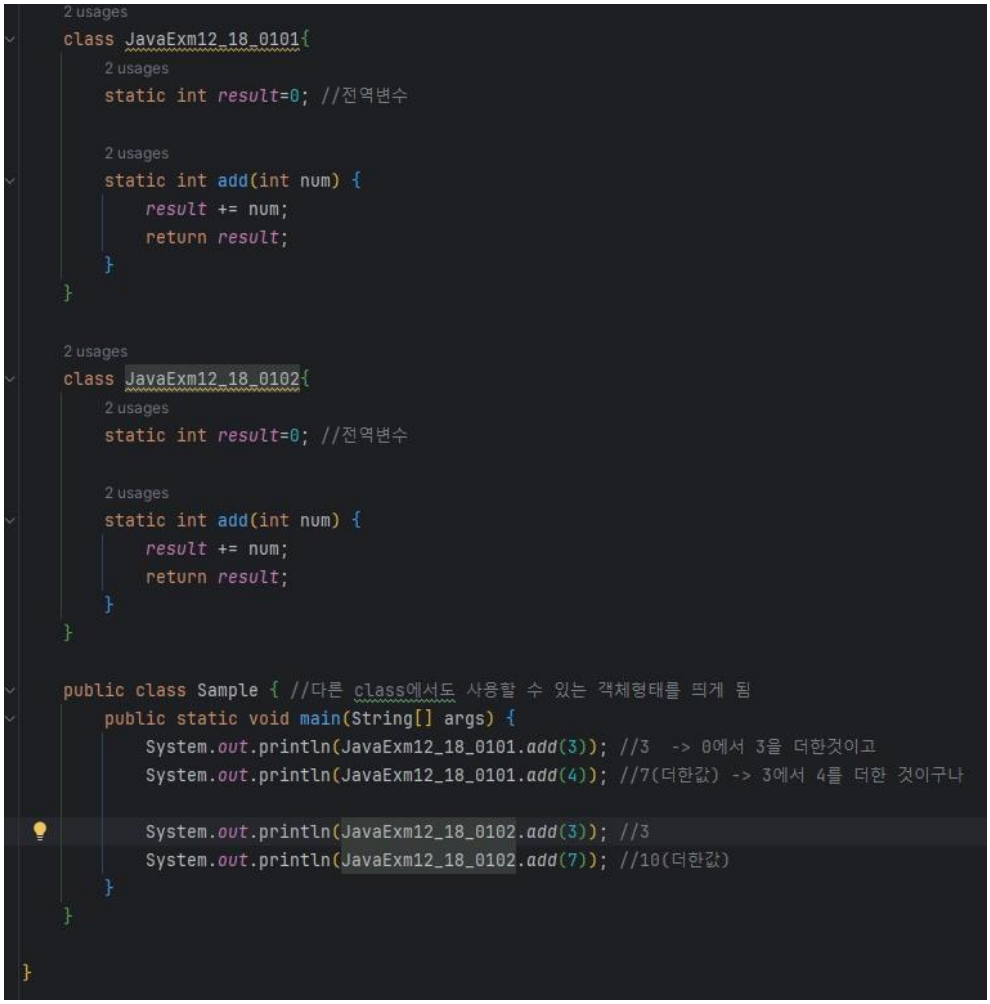


result는 전역변수인 것. add라는 메소드 안에 있는 것이 아니라 하나의 클래스 안에서 result의 값을 불러서 사용할 수 있음.



Add(3)은 3이라는 값을 변수에 지정해 준 것이다.

add가 메소드의 값이니까 num을 3이라고 지정해줘서 결과 값은 3+4인 7이 나옴



```
2 usages
class JavaExm12_18_0101{
    2 usages
    static int result=0; //전역변수

    2 usages
    static int add(int num) {
        result += num;
        return result;
    }
}

2 usages
class JavaExm12_18_0102{
    2 usages
    static int result=0; //전역변수

    2 usages
    static int add(int num) {
        result += num;
        return result;
    }
}

public class Sample { //다른 class에서도 사용할 수 있는 객체형태를 띄게 됨
    public static void main(String[] args) {
        System.out.println(JavaExm12_18_0101.add(3)); //3  -> 0에서 3을 더한것이고
        System.out.println(JavaExm12_18_0101.add(4)); //7(더한값) -> 3에서 4를 더한 것이구나

        System.out.println(JavaExm12_18_0102.add(3)); //3
        System.out.println(JavaExm12_18_0102.add(7)); //10(더한값)
    }
}

}
```

result=0; 으로 전역변수를 설정해주고 그 결과 값을 가지고 와서 3과4를 더하는 시스템을 만들어 주었다.

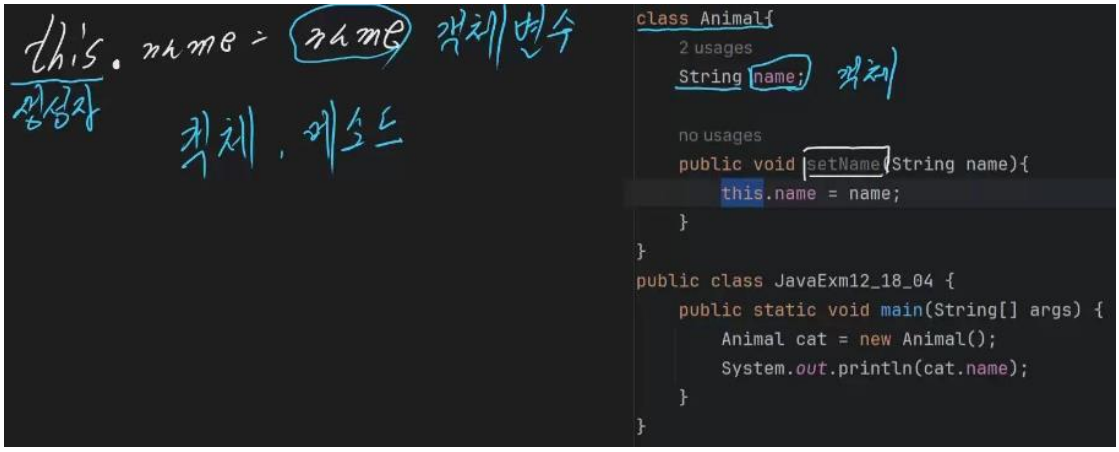
클래스(class)

자바에서 클래스(class)란 객체를 정의하는 틀 또는 설계도와 같은 의미로 사용됩니다.
자바에서는 이러한 설계도인 클래스를 가지고, 여러 객체를 생성하여 사용하게 됩니다.

클래스는 객체의 상태를 나타내는 필드(field)와 객체의 행동을 나타내는 메소드(method)로 구성됩니다.
즉, 필드(field)란 클래스에 포함된 변수(variable)를 의미합니다.
또한, 메소드(method)란 어떠한 특정 작업을 수행하기 위한 명령문의 집합이라 할 수 있습니다.

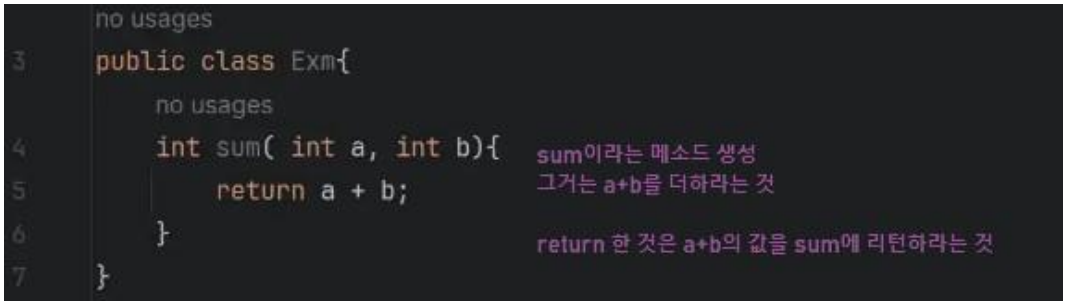
class란, 객체를 정의하는 틀 또는 설계도와 같은 의미로 사용한다.

이와 같이 함수 안에서 만들어진 변수를 **지역변수**라고 하고, 함수 밖에서 만들어진 변수를 전역변수라고 합니다. **지역변수**는 함수가 호출되면 만들어져서, 함수의 실행이 끝날 때 함께 없어지는 반면, 전역변수는 함수와는 관계없이 항상 꿋꿋이 지구를 지킨답니다. 그래서 영어로 전역변수를 global이라는 말로 표현하지용…



animal 이라는 클래스를 사용하고 객체를 전체 선언(전역번수 선언) ,

그리고 SetName이라는 메소드 안에 밖에 있는 전역변수를 this로 가지고 왔다.



sum이라는 메소드 생성 -> 그 메소드의 기능은 a+b를 더하라는 것이다.

- JavaExm12_18_05.java

```java
public class Exm{
    1 usage
    int sum( int a, int b){
        return a + b;
    }
}

public static void main(String[] args) {
    int a = 3;
    int b = 10;

    Exm hap = new Exm();
    int c = hap.sum(a, b);

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

```java
public class JavaExm12_18_05{
    1 usage
    int sum(int a, int b){
    return a + b;}

public static void main(String[] args) {
    int a = 3;
    int b = 10;

    JavaExm12_18_05 hap = new JavaExm12_18_05();
    int c = hap.sum(a, b);

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

반복문 돌려주면
매번 작업 돌려주지 않아도 결과 값 출력이 가능하다는
생각이 든다 ->
method가 그런 역할인가보다!!!

a	b	c
1	3	
5	6	
4	8	
5	7	
3	9	

- JavaExm12_18_06.java

```java
package obj;

public class JavaExm12_18_06 {
    1 usage
    String say(){ //method
        return "hello";
    }

    public static void main(String[] args) {
        JavaExm12_18_06 sam=new JavaExm12_18_06();
        String a=sam.say();
        System.out.println(a);
    }
}
```

JavaExm도 class 명이다
그 클래스로 새로운 객체인 sam을 만들었는데
그 객체에 say라는 메소드를
a에 넣어준다.
그리고 그걸 출력해주면
"hello"가 나오는 것을 확인할 수 있다.

- javaExm12_18_08.java

```java
package obj;

public class JavaExm12_18_08 {
    2 usages
    void sayNick(String nick){

        System.out.println("나의 별명은 " + nick + "이다.");
    }

    public static void main(String[] args) {
        JavaExm12_18_08 prn = new JavaExm12_18_08();
        prn.sayNick("papa");
        prn.sayNick("fool");
    }
```

```java
package obj;

public class JavaExm12_18_08 {
    4 usages
    void sayNick(String nick){
//        if (nick == "fool") {
//            return;
//        }

        if ("fool".equals(nick)) {
            return;
        }
        System.out.println("나의 별명은 " + nick +
}
Extract Surround
public static void main(String[] args) {
    JavaExm12_18_08 prn = new JavaExm12_18_08
    prn.sayNick("papa");
    prn.sayNick("fool");
```

```java
if (nick == "fool") return;
if ("fool".equals(nick)) return;
```

위에 두개는 같은 명령을 실행하는 코드이지만,
보이는 것이 다르다

equals를 사용해서 fool이라는 이름 가졌다면 return;
해주는 것으로 코드 작성할 수 있다.

- 메소드의 4가지 종류

입력값이 없는 메서드

06

입력값이 없는 메서드가 존재할까? 당연히 그렇다. 다음 예를 살펴보자.

```java
String say() {
    return "Hi";
}
```

```java
1 usage
String say(){
    return "hello";
}
```
입력값이 없는 메소드

say 메서드의 입출력 자료형은 다음과 같다.

- 입력값 : 없음
- 리턴값 : String 자료형

리턴값이 없는 메서드

07

리턴값이 없는 메서드 역시 존재한다. 다음 예를 보자.

```java
void sum(int a, int b) {
    System.out.println(a+"과 "+b+"의 합은 "+(a+b)+"입니다.");
}
```
return 값이 없다
=> 이럴 때 void 적어줘라
void는 리턴 값 없음!

sum 메서드의 입출력 자료형은 다음과 같다.

- 입력값 : int 자료형 a, int 자료형 b
- 리턴값 : void (리턴값 없음)

리턴값이 없는 메서드는 명시적으로 **리턴 자료형 부분에 void라고 표기**한다. 리턴값이 없는 메서드
과 같이 사용한다.

- JavaExm12_18_09

```java
public static void main(String[] args) {
    JavaExm12_18_09 java=new JavaExm12_18_09();
    java.a=1;
    System.out.println(java.a);
    java.varTest(); //a++이 작동함.
    System.out.println(java.a);
}
}
```

```
1
2
```

```java
void varTest(){ //입력값이 없음=> () 안에 아무것도 안넣었으니까.
    this.a++;    this 생성자 생성해줘야함
}
```

<node, npm 설치_VS code로 함>

```
nvm use 18.14.1
PS C:\Users\you> nvm use 18.14.1
Now using node v18.14.1 (64-bit)
PS C:\Users\you> npm install -g eslint

added 99 packages in 6s
npm notice
npm notice New major version of npm available! 9.3.1 -> 10.2.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.5
npm notice Run npm install -g npm@10.2.5 to update!
npm notice
PS C:\Users\you> nvm

Running version 1.1.11.
```

```
PS C:\Users\you> nvm install 18.14.1
Downloading node.js version 18.14.1 (64-bit)...
Complete
npm v9.3.1 installed successfully.


Installation complete. If you want to use this version, type

nvm use 18.14.1
PS C:\Users\you> nvm use 18.14.1
Now using node v18.14.1 (64-bit)
PS C:\Users\you> npm install -g eslint

added 99 packages in 6s
npm notice
npm notice New major version of npm available! 9.3.1 -> 10.2.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.5
npm notice Run npm install -g npm@10.2.5 to update!
npm notice
```