23-12-20

- Java

문제 1. 오류찾기

```
package String;

public class JavaExm12_20_01 {
    public static void main(String[] args) {
        Tv t =new Tv();
    }

}

class Product{
    int price;
    int bonusPoint;

    Product(){}

    Product(int price) {
        this.price=price;
        bonusPoint=(int)(price/10.0);
    }
}

class Tv extends Product{
    Tv() {}

//    Tv(int price) {
//        this.price=price;
//    }
    public String toString(){
        return "Tv";
    }

}
```

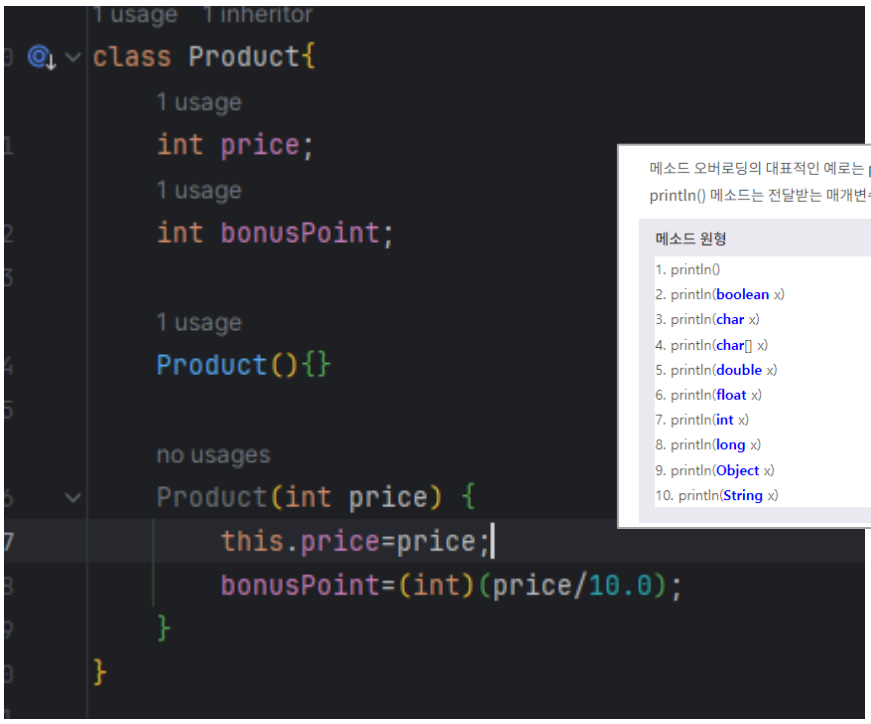
생성자가 없었던 문제이다.

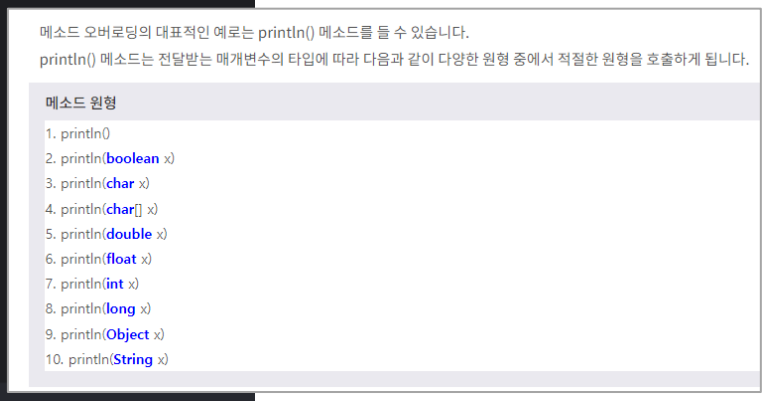
Product() {} 이렇게 선언해주면 문제가 생기지 않는다.

Super 사용

- YouTv

-Final : 못을 받은 것 변경하지 못하게 고정하는 것





생성자 만들어주면 오버로딩 되어, 오류가 나지 않는다.

Void는 리턴값 없음 -> 대신 사용할 수 있는 것들이 int, String 같은 것.

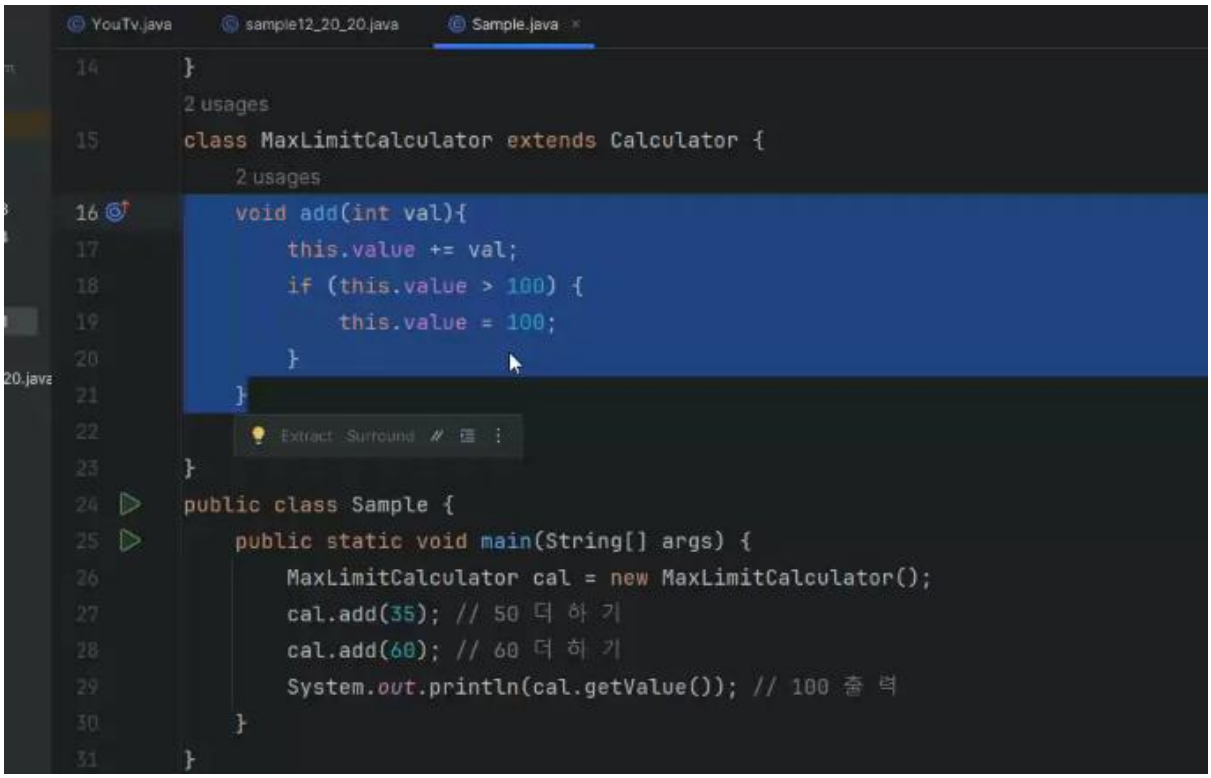
문제 2.

- JavaExm12_20_02 패키지 > Sample.java

```
객체 변수 value 가 100 보다 큰 값은 가질 수 없도록 제한하는 MaxLimitCalculator 클래스를 만들어 보자.
즉, 다음과 같이 동작해야 한다.

MaxLimitCalculator cal = new MaxLimitCalculator();
cal.add(50); // 50 더 하 기
cal.add(60); // 60 더 하 기
System.out.println(cal.getValue()); // 100 출 력

단, MaxLimitCalculator 클래스는 반드시 다음의 Calculator 클래스를 상속해서 만들어야 한다.
class Calculator {
            int value;
Calculator() {
this.value = 0;
}
void add(int val) {
this.value += val;
}
int getValue() {
return this.value;
}
}
```



100이 넘어가면 다른 함수를 실행하도록 설정하고 싶다 (add가 아닌 다른이름으로)

this.value += val; //이걸 add에서만 하고 싶다

방법여러가지임

```java
package Class.JavaExm12_20_02;


//문제2
public class JavaExm12_20_03 {
    public static void main(String[] args) {
        MaxLimitCalculator cal =new MaxLimitCalculator();
        cal.add(50);
        cal.add(60);

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

```java
1 usage  1 inheritor
class Calculator{
    6 usages
    int value;
    1 usage
    Calculator() {
        this.value=0;
    }
    2 usages
    void add(int val){ //add를 이거 안쓰고 아래것을 쓴다는건가
        this.value+=val;
    }
    1 usage
    int getValue() {
        return this.value;
    }
}
```

```java
2 usages
class MaxLimitCalculator extends Calculator {

    2 usages
    void add(int val){ //add를 이거 안쓰고 아래것을 쓴다는건가
        this.value+=val;
        if(this.value>100){
            addadd(val);
        }
    }

    1 usage
    int addadd(int val) {
        //this.value += val; //이걸 add에서만 하고 싶다
        return this.value = 100;
    }
}
```

그래서 사실 상속받지 않으면, 그 값을 add 안에 if문으로 넣어주는 것이 가장 깔끔하다
상속받아서 add를 가지고 와서 오버라이딩을 한다. 그리고 if문을 add에다 넣어주는 것
그리고 100이 넘어가면 addadd 함수를 실행하라!
add함수는 100이 넘어가면 100으로 출력하라는 것

문제 3.

다음과 같이 주어진 정수가 홀수인지 짝수인지 판별해 주는 Calculator 클래스를 작성해 보자. 이때 홀수이면 true, 짝수면 false 를 리턴해야 한다.

```
Calculator cal = new Calculator();
System.out.println(cal.isOdd(3)); // 3 은 홀 수 이 므 로 true 출 력
System.out.println(cal.isOdd(4)); // 4 는 짝 수 이 므 로 false 출 력

true
false
```

```java
package Class.JavaExm12_20_02;

public class JavaExm12_20_04 {
    public static void main(String[] args) {
        Calculator cal =new Calculator();
        System.out.println(cal.isOdd( num: 3));
        System.out.println(cal.isOdd( num: 4));
    }
}
2 usages
class Calculator{
    no usages
    int num;
    2 usages
    boolean isOdd(int num) {
        if (num % 2 != 0) {
            return true;
        } else /*if (this.num % 2 == 0)*/ {
            return false;
        }
    }
}
```

```
true
false
```

풀어본 것

```java
2 usages
class Calculator{
//    int num;
    2 usages
    boolean isOdd(int num) {
        /*if (this.num % 2 == 0)*/
        return num % 2 != 0; //true와 false니까 바로 출력이 가능하게 설정가능(홀수 true, 짝수 false)
```

이렇게 풀어도 됨, boolean이니까, 참과 거짓만 나올 것이다.

return값을 저렇게 넣어두면 홀수가 true가 되고 짝수가 false가 되니까 그 값을 받아와서 출력함

문제 4.

Calculator 클래스를 상속하는 UpgradeCalculator 를 만들고 값을 뺄 수 있는 minus 메서드를 추가해 보자. 즉, 다음과 같이 동작하는 클래스를 만들어야 한다.

```java
public class JavaExm12_20_05 {
    public static void main(String[] args) {
        Calculator1 cal = new Calculator1();
        cal.add(10);
        System.out.println(cal.getValue()); // 10 출 력

        UpgradeCalculator cal1 = new UpgradeCalculator();
        cal1.add(10);
        cal1.minus( val: 3);
        System.out.println(cal1.getValue()); // 10 에 서 3 을 뺀 7 을 출 력
    }
}
3 usages  1 inheritor
class Calculator1 {
    4 usages
    int value;
    2 usages
    Calculator1() {
        this.value = 0;
    }
    2 usages
    void add(int val) {
        this.value += val;
    }
    2 usages
    int getValue() {
        return this.value;
    }
}
2 usages
class UpgradeCalculator extends Calculator1{
    1 usage
    void minus(int val){
        this.value -= val;
    }
```

Calculator1을 상속받아 UpgradeCalculator라는 class를 생성했고, 거기에 minus라는 method생성 그것을 main에서 받아서 10-7를 적용하게 만들었다.

문제 5.

다음과 같이 정수 배열 또는 정수의 리스트로 그 평균을 구해 리턴하는 Calculator 클래스를 작성해 보자.

```
int[] data = {1, 3, 5, 7, 9};
Calculator cal = new Calculator();
int result = cal.avg(data);
System.out.println(result); // 5 출 력
5
ArrayList<Integer> data = new ArrayList<>(Arrays.asList(1, 3, 5, 7, 9));
Calculator cal = new Calculator();
int result = cal.avg(data);
System.out.println(result); // 5 출 력
5
```

array로 받는 하나와 array List로 받는 하나가 있다

```
package Class.JavaExm12_20_02;

//import java.util.ArrayList;
//import java.util.Arrays;
import java.util.*;

//메소드 오버로딩 -> 같은 이름으로도 다른기능을 수행함

public class JavaExm12_20_06 {
    public static void main(String[] args) {

        int[] data = {1, 3, 5, 7, 9};
        Calculator06 cal = new Calculator06();
        int result = cal.avg(data);
        System.out.println(result); // 5 출 력
//        5
        ArrayList<Integer> data1 = new ArrayList<>(Arrays.asList(1, 3, 5, 7, 9));
        Calculator06 cal1 = new Calculator06();
        int result1 = cal1.avg(data);
        System.out.println(result1); // 5 출 력
//        5

    }
}
```

```
5
5
```

int[] data = {1,3,5,7,9} // Array(배열)로 받는 방법

ArrayList<Integer> data1 = new ArrayList<>(Arrays.asList(1,3,5,7,9)); // ArrayList로 받는 방법

-> 두가지 마다 데이터 넣는 방법과 사용하는 방법이 다르다

```java
6 usages
class Calculator06 {
    3 usages
    int avg(int[] data){
        int total=0;
        for(int i=0;i<data.length;i++){
            total+=data[i]; //i만입력하면 0~data.length의 길이니까 data안에 [i]넣어주기
        }
        //이게뭐지
//        for (int num : data){
//            total+=num;
//        }
        return total / data.length;
    }

    no usages
    int avg(ArrayList<Integer> data1) {
        int total = 0;
        for (int i = 0; i < data1.size(); i++) {
            total += data1.get(i); //arraylist니까 data.get(i)써야한다
        }
//        for(int num:data){
//            total += num;
//        }
        return total / data1.size(); //여기서는 length가 아니라 size로 들어와야함
    }
}
```

같은 avg이지만 실제 작업처리는 다르다

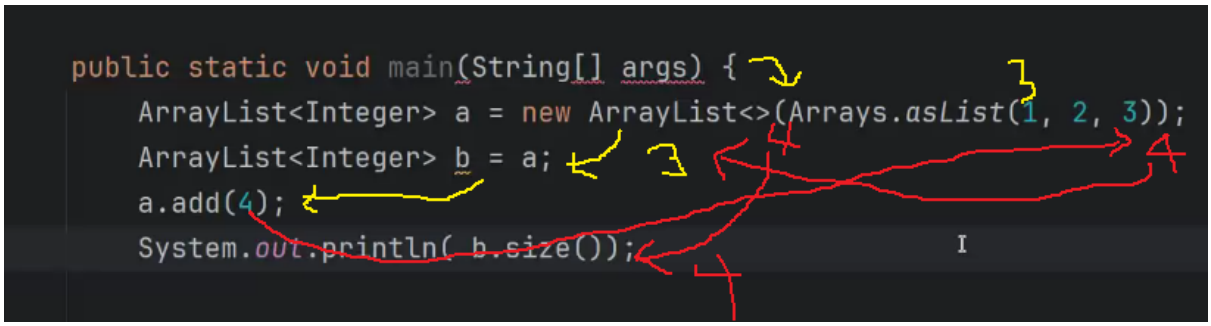
요약

	array	arrayList
사이즈	초기화시 고정	초기화시 사이즈를 표시하지 않음. 사이즈가 동적이다.
속도	초기화 시 메모리에 할당되어 속도 빠름	추가시 메모리를 재할당하여 속도가 느림
변경	사이즈 변경 불가	추가,삭제 가능
다차원	가능	불가능
타입	primitive type(int,byte, char etc), object	object elemnet만 가능
제네릭	사용 불가능	사용 가능(타입 안정성 보장)
길이	length 변수	size() 메서드
변수 추가	assignment 연산자 사용	add() 메소드 사용

문제 6.

```
public static void main(String[] args) {
    ArrayList<Integer> a = new ArrayList<>(Arrays.asList(1, 2, 3));
    ArrayList<Integer> b = a;
    a.add(4);
    System.out.println( b.size());
```

눈으로 보고 풀기 값은 무엇이 출력이 될까?



출력은 4가 됨.

a.add(4); 하는 순간 a에 추가가 되고, 다시 b로 내려와서 받기 때문이다 (헷갈림)

```
package Class.JavaExm12_20_02;

import java.util.ArrayList;
import java.util.Arrays;

public class JavaExm12_20_07 {
    public static void main(String[] args) {
        ArrayList<Integer> a=new ArrayList<>(Arrays.asList(1,2,3));
        System.out.println(a.size());
        ArrayList<Integer> b=a; //
        System.out.println(a.size());
        System.out.println(b.size());
        a.add(4);
        System.out.println(a.size());
        System.out.println(b.size());
        //순서대로 다시 내려옴
    }
}
```

```
3
3
3
4
4
```

a에 추가해준 것이 다시 순서대로 내려와서 b도 4가 됨.

add를 컨트롤 누르고 클릭하면, add 내장함수의 기능 볼 수 있다.

-> 함수가 ++ 되고 add 되고 다시 return true가 되는 것을 확인할 수 있다.

```
465    public boolean add(E e) {
466        modCount++;
467        add(e, elementData, size);
468        return true;
469    }
```

문제 7.

다음은 광물 계산기 프로그램을 구현한 것이다. 이 프로그램은 금 (Gold) 인 경우 100, 은 (Silver) 인 경우 90, 동 (Bronze) 의 경우는 80 의 가치를 더하는 기능 (add 메서드) 이 있다. 하지만 이 광물 계산기는 광물 메서드가 추가될 때마다 add 메서드를 추가해야 한다는 불편함이 있다. 광물이 추가되더라도 MineralCalculator 클래스를 변경할 필요가 없도록 코드를 수정해 보자.

```
package Class.JavaExm12_20_02;

public class JavaExm12_20_09 {
    public static void main(String[] args) {
        MineralCalculator cal = new MineralCalculator();
        cal.add(new Gold());
        cal.add(new Silver());
        cal.add(new Bronze());
        System.out.println(cal.getValue()); // 270 출 력
    }
}
interface Mineral{
    int getValue();
}

class Gold implements Mineral{ //interface 가지고 와서 클래스 Gold 로 만들어줌
    public int getValue(){
        return 100;
    }
}
class Silver implements Mineral{
    public int getValue(){
        return 90;
    }
}
class Bronze implements Mineral{
    public int getValue(){
        return 80;
    }
}


class MineralCalculator {
    int value = 0;
    public void add(Mineral mineral) {
        this.value += mineral.getValue();
    }
//    왜 다 필요 없어지는 거지???? -> 위에 값을 줬기 때문에! 없어도 됨
//    public void add(Silver silver) {
//        this.value += 90;
//    }
//    public void add(Bronze bronze) {
//        this.value += 80;
//    }
    public int getValue() {
        return this.value;
    }
}
```

궁금한점

- public class는 main에만 생성이 되는 것인가
  class 안에 public이 있으면 그 안에 메소드도 public 써주지 않아도 가능한가?

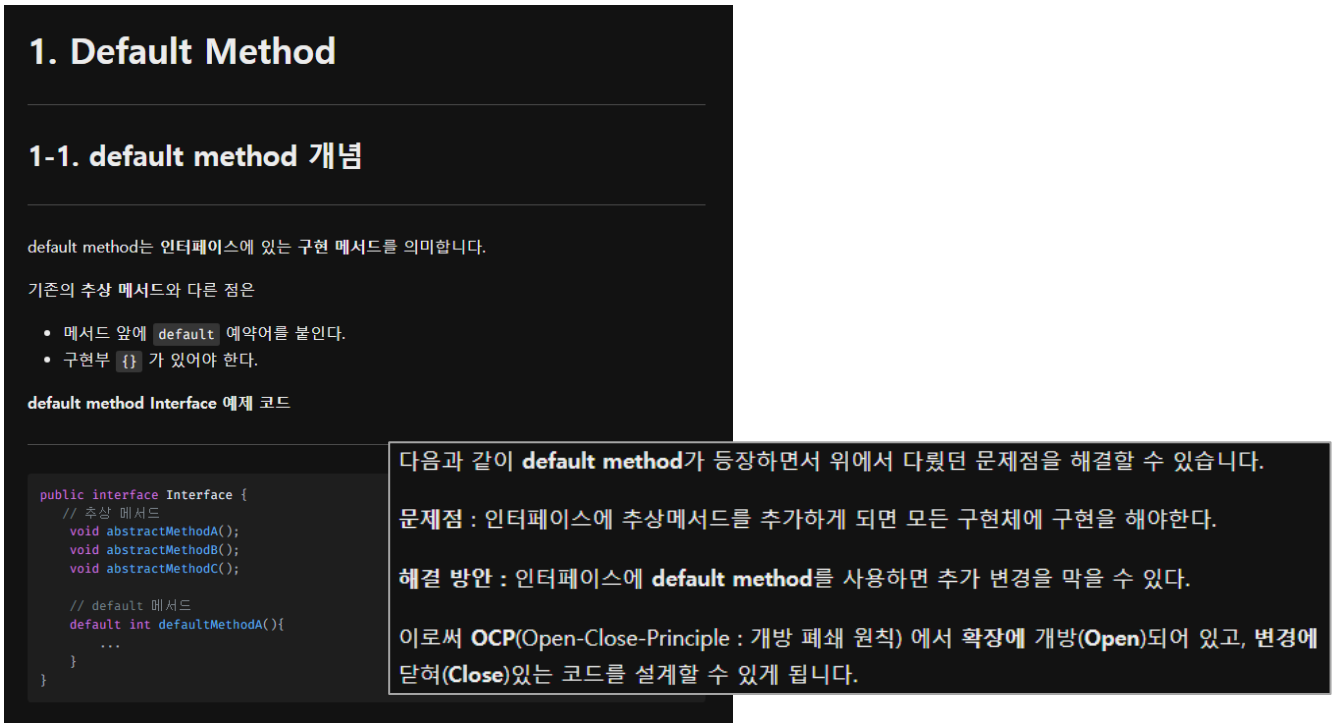
Program2라는 클래스 앞에 public이라는 접근제어자를 붙였더니 이번에는 컴파일 에러가 발생한다.
그럼 아래와 같이 수정을 해보자

```
public class Program {
}

class Program2 {
}
```

문제없이 컴파일이 잘 된다. 어떤 차이일까?
위와 같이 컴파일되는 이유는 파일명(Program.java)때문이다.
자바에는 클래스명 앞에 public이란 접근제어자를 추가하면 그 클래스의 이름은 파일명과 동일해야만 하는 규칙이 있다.

- interface일 때 Default값



https://velog.io/@heoseungyeon/%EB%94%94%ED%8F%B4%ED%8A%B8-%EB%A9%94%EC%84%9C%EB%93%9CDefault-Method