23-12-15

- Java 배열

배열 -> array list -> hashmap (자바에서 가장많이 사용하는 자료저장형태)

```
System.out.println(Arrays.toString(week)); //문자열로 바꿔줘야함
System.out.println(week);
```

자바 배열 형태라고 알려줌
week[3] 처럼 배열로 출력하면 그 안에 포함된 값이 나옴
arrays.toString(week)라고 하면 배열이 문자열로 바뀌어서 출력됨

```
[월, 화, 수, 목, 금, 토, 토]
[Ljava.lang.String;@4eec7777
```

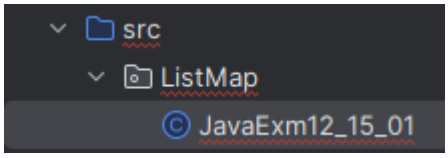
14일에 했던 거 이어서 설명

```
System.out.println(Arrays.toString(week)); //문자열로 바꿔줘야함
System.out.println(week[9]);
```

```
[월, 화, 수, 목, 금, 토, 토]
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException Create breakpoint : Index 9 out of bounds for length 7
    at JavaExm12_14_03.main(JavaExm12_14_03.java:18)
```

7까지 설정했는데, 9번째 출력하니까 에러가 남

배열 전체 값을 7로 설정했는데 9를 입력하면 에러가 남



package

package 이름 사칙연산>그 밑에 들어있는 모든class는 사칙연산을 위한 함수, method가들어있는 것.

package명 사용할 수 있다

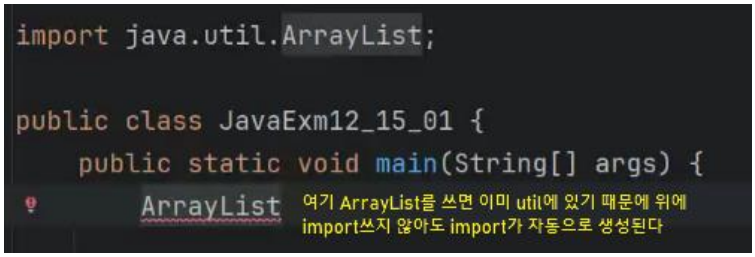
(package생성하니까 하나의 폴더가 생성됨)

```
package ListMap;
```

package 안에 class 생성하니까 package ListMap안에서 생성되었다고 알려준다

```
public class JavaExm12_15_01 {
    ps  예약어 템플릿 언어
    psf                                    public static final
    psfi                                   public static final int
    psfs                                   public static final String
    psvm                                   main() method declaration
    public String toString() {...}         Object
Press Enter to insert, Tab to replace  Next Tip
```

PSVM : 입력하면 직접입력하지 않아도 바로 public static void main이 입력된다



~~interface 연결?~~

객체지향의 장점 : 선언한 것을 가지고 올수있다?

자바를 사용하는 가장 큰 이유 :

ArrayList() 빈배열, 목록으로 만들 것이라는 것

그리고 아래 해당값을 집어 넣겠다는 것

1) 2번째 값을 get 해서 출력함

2) 전체 배열의 값, size 사용

3) 140포함되어 있으면 True를, 포함되어 있지 않으면 False를 출력

4) remove는 삭제한다는 것.

Remove 쓰면 두개가 나오는데 index로 하고 0을 입력하면 0번째 있는 값 삭제하라는 것.

- 제너릭

ArrayList<String> pitches=new ArrayList<String>();

자료형으로 설정해주지 않으면 int로 변환하거나 int에서 string으로 변경할 때 문제가 생긴다

변수명 사용하기 전에 자료형 지정해라

원래 arraylist안에 자료형을 써줘야 한다

무시하고 쓰는 것을 제너릭이 허용해줬다

<String>쓰는 것이 원칙이지만 쓰지 않아도 된다

=>자료형 정확하게 구분하고 지정할 때 사용한다

=>쓰지 않아도 되지만 쓰는 것이 좋다(구분이 가능)

```
1.5부터 문서 적용하게 되면서 스트링적어줌
ArrayList<String> pitches = new ArrayList<String>();
ArrayList<String> pitches = new ArrayList();
ArrayList pitches = new ArrayList();
예전에 쓰던것
```

```
String sa = (String) play.get(2);
형변환을 해줘야 하기 때문에 stringd
```

```
public static void main(String[] args) {
    ArrayList play = new ArrayList();
    play.add("129");
    play.add("135");
    play.add("145");

    String one = (String) play.get(0);
    String two = (String) play.get(1);
    String sa = (String) play.get(2);
```
generics이전

```
ArrayList<String> play = new ArrayList<>();

play.add("129");
play.add("135");
play.add("145");

String one = play.get(0);
String two = play.get(1);
String sam = play.get(2);
```
generics이후

```
public class JavaExm12_15_02 {
    public static void main(String[] args) {
        ArrayList<String> play = new ArrayList<>();
        play.add("129");
        play.add("135");
        play.add("145");
        System.out.println(play);

        String[] data = {"129","135","145"};
        ArrayList<String> play = new ArrayList<>(Arrays.asList(data));
        System.out.println(play);
```

내가 자료값으로 받아서 arrayList로 집어 넣겠다는 것이다.
data 변수가 3개의 값 가지고 있는데
뒤로 올 때는 data가 하나의 오브젝트 덩어리로 가지고 오는 것
data가 list안으로 포함해서 객체로 가지고 온다

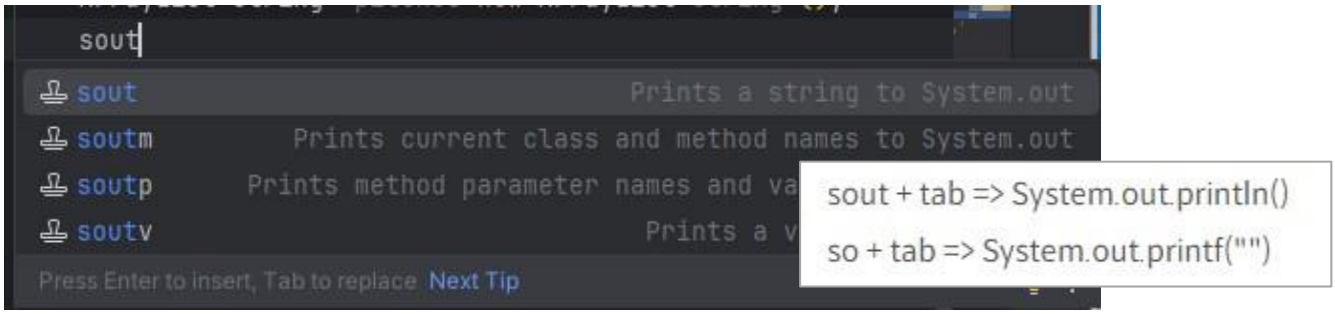
배열을 arrayList로 바꾼다 -> 오브젝트 형태로 바꿀 수 있기 때문에
자료 받는 것 데이터 받아와서 집어넣는다 -> 회원가입 할 때 사용가능
제너릭을 사용하는 이유 -> 3줄로 줄일 수 있다

```
ArrayList<String> play = new ArrayList<>(Arrays.asList("129","135","145"));
System.out.println(play);
```

ArrayList<String> play=new ArrayList<>(Arrays.asList("129","135","145"));
플레이에 데이터 넣을 때에 배열값 사용하지 않고 직접 데이터 넣을 수 있는 방법

배열(Array)과 ArrayList의 차이점

	Array	ArrayList
사이즈	초기화시 고정 int[] arr = new int[3];	초기화시 사이즈를 표시하지 않음. 크기가 가변적임 ArrayList<Integer> arrList = new ArrayList<>();
속도	초기화시 메모리에 할당되어 ArrayList보다 속도가 빠름	데이터 추가 삭제시 메모리를 재할당하기 때문에 속도가 Array보다 느림
크기 변경	사이즈 변경 불가	추가, 삭제 가능 add(), remove()
다차원	int[][][] multiArr = new int[3][3][3];	불가능



Sout은 System.out.println()을 바로 출력할 수 있는 약어이다.

```
ArrayList<String> play=new ArrayList<>(Arrays.asList("129","135","145"));
play.sort(Comparator.naturalOrder()); //오름차순
play.sort(Comparator.reverseOrder()); //내림차순
```

Sort 정렬할 때 comparator를 사용하여 오름차순과 내림차순으로 정렬할 수 있다.

```
ArrayList<String> play = new ArrayList<>(Arrays.asList("145","135","147"));
String result ="";
for (int i = 0; i < play.size(); i++) {
    result += play.get(i);
    result += ", ";
}
result = result.substring(0, result.length()-1);
System.out.println(result);
```

무언가를 출력할 때 반복문 사용해서 출력하기

```
public static void main(String[] args) {
    ArrayList<String> play = new ArrayList<>(Arrays.asList("145","135","147"));
    String result = String.join( delimiter: ", ", play);
    System.out.println(result);
```

join이라는 argument존재, join은 배열에도 사용된다

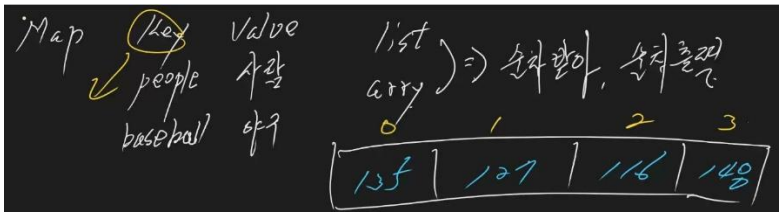
- Map

배열의 시작이 Map이다.

key에 대한 값이 되는 것

key에 대한 value값을 가지고 오는 것 . json형태

JavaScript Object Notation (JSON)은 Javascript 객체 문법으로 구조화된 데이터를 표현하기 위한 문자 기반의 표준 포맷입니다.



list, array => 순차 출력, index 넘버로 받아오는 것

> ***Map***
>
> Map 자료구조의 특징은 키(Key)와 값(Value)이다.
> 키를 통하여 값에 접근할 수 있는 구조이다.
>
> List나 배열은 인덱스로 접근한다. 인덱스는 단순히 순서만 나타낸다. 그러나 Map의 키는 개발자가 의미를 부여할 수 있다.

map은 key값만 찾아서 바로 출력할 수 있음
필요한 부분 key값만 찾아서 data 가지고 오는 구조 => 빠르다
그래서 자료구조는 map구조로 받아와야 한다 => Hash구조

Hash라고 하면 암호화 기술도 같이 포함하는 기술이다.
체인에 대한 기술 -> map에서 튀어나온 것

```java
public class JavaExm12_15_03 {
    public static void main(String[] args) {
        HashMap<String, String> map=new HashMap<>(); //빈칸으로 받는다
        map.put("people","사람");
        map.put("baseball","야구");
        System.out.println(map.get("people"));
        //key값 String으로 되어 있으니까 "" 안에넣기
    }
}
```

```
사람
```

HashMap => 객체지향 프로그램
method만 만들어 놓으면 다른 사람이 만들어 놓은 것을 내 것처럼 사용할 수 있다.

key값과 value값인데, ""를 쓴 것은 String이니까 그렇게 쓴 것
출력할 때도 key값이 String으로 되어 있으니까 ""안에 넣기
=> JSON형태의 기본, 이제 사용하는 데이터를 이렇게 사용할 것.

hash암호화기법에 따라 순서가 결정됨
hash는 저장되는 것은 랜덤이다.

put은 key와 value값을 집어넣는 것.

```java
Map<String, Integer> LinkedHashMap =new LinkedHashMap<>();
LinkedHashMap.put("sim",55);
LinkedHashMap.put("kim",15);
LinkedHashMap.put("lim",35);

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

```
{sim=55, kim=15, lim=35}
```

- **linkedHashmap**은 입력된 순서대로 출력(잘 사용하지 않는다)

```java
Map<String, Integer>  TreeMap = new TreeMap<>();
TreeMap.put("sim",55);
TreeMap.put("kim",15);
TreeMap.put("zoo",35);

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

```
{kim=15, sim=55, zoo=35}
```

- **TreeMap**은 key 값의 오름차순으로 출력

```java
public class JavaExm12_15_04 {
    public static void main(String[] args) {
        HashSet<String> set=new HashSet<>(Arrays.asList("h","e","l","l","o"));
        //빈값으로 받는 것이 아니라 Arrays.asList로 받는다
        System.out.println(set);

            hashmap은 리스트와 어레이와 같이 인덱싱이 있는것이 아니라서 출력할 때 순차적
            으로 출력되는 것이 아니다.
            나온 결과값이 오름차순이다!!
    }
```

```
[e, h, l, o]
```

- **HashSet** 중복값 제거 + 입력된 순서대로 출력되지 않는다. + 오름차순 출력

  해당하는자료 교집합, 합집합, 차집합으로 해당하는 자료를 찾는다

  - retainAll 교집합
  - addAll 합집합
  - removeAll 차집합(s1에만 해당하는 값만 나옴)

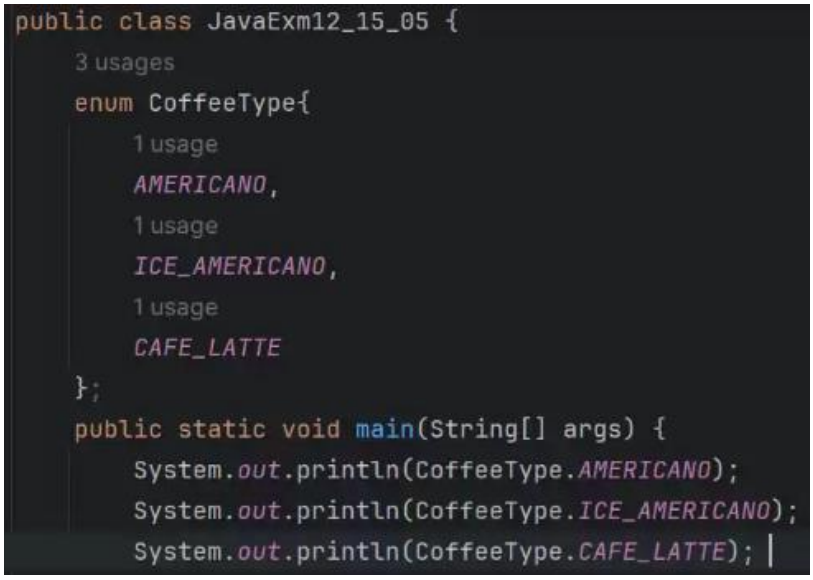
```java
HashSet<Integer> s1=new HashSet<>(Arrays.asList(1,2,3,4,5,6,7));
HashSet<Integer> s2=new HashSet<>(Arrays.asList(2,4,6,8,10));

//교집합
HashSet<Integer> intersection=new HashSet<>(s1); //s1을 자료로 생성
intersection.retainAll(s2);
System.out.println(intersection);

//합집합
HashSet<Integer> union=new HashSet<>(s1);
union.addAll(s2);
System.out.println(union);

//차집합
HashSet<Integer> subtract=new HashSet<>(s1);
subtract.removeAll(s2);
System.out.println(subtract);
```

```
[2, 4, 6]
[1, 2, 3, 4, 5, 6, 7, 8, 10]
[1, 3, 5, 7]
```

(union 이름은 주어진 것)

array와 list는 주소값을 인덱싱

hashmap은 인덱싱이 없다.



- **enum** 으로 상수 집합을 만들고

public static void printCoffeePrice(CoffeeType) {

HashMap<CoffeeType, Integer> priceMap=new HashMap<>();

아메리카노 총 99잔 팔 수 있고 1잔남았을 때 팔리면 sold out이렇게 만드려고 하는 것

오브젝트로 해서 자료값으로 쓰겠다고 변수를 넣어놨다 () 하면 빈 값이 되는 것

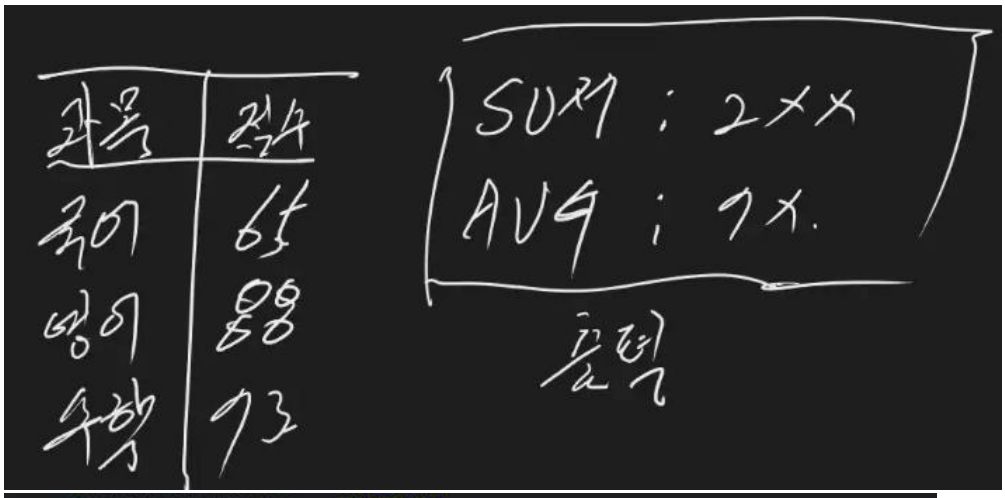
위에 enum만들어놓은것 사용하지 않는 것

그렇게 되면 데이터 안에 만들어야 한다.

HashMap으로 CoffeType위에 만들어 놓은 자료를 가지고 온 것이다.

<문제>

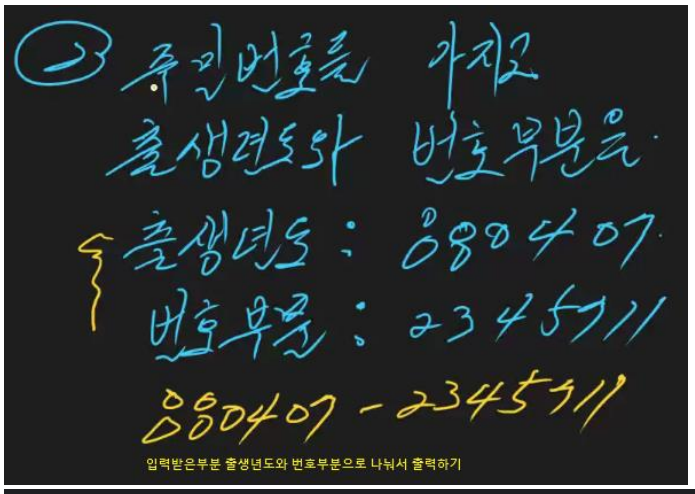
1. 과목의 점수가 주어질 때 합계와 평균 구하기



JavExm12_15_06.java × 1번 문제 답 :

```java
public static void main(String[] args) {
    int a = 65;
    int b = 88;
    int c = 73;
    int sum = a+b+c;
    int avg = (a+b+c)/3;
    System.out.println( "sum : " + sum);
    System.out.println( "Avg : " + avg);
```

2. 주민번호를 가지고 출생년도와 번호부분 나눠서 출력




```
String num = "880407-2345771";
System.out.println("출생년도: "+num.substring(0,6));
System.out.println("번호부분: "+num.substring(7,13));
```

```
출생년도: 880407
출생년도: 234577
```

풀어본 것

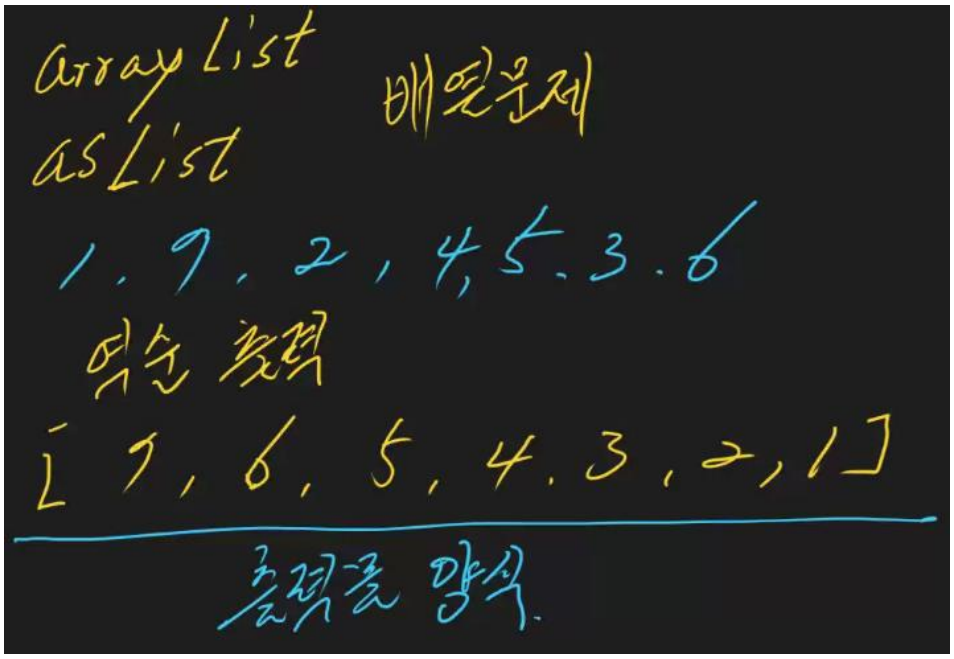
```
String data="880407-2345711";
String birth="";
String no="";

birth=data.substring(0,data.length()-8);
no=data.substring(7,data.length());

System.out.println("출생년도 : "+birth);
System.out.println("주민번호 : "+no);
```

선생님 답

```
String result ="";
for(int i=0;i<play.size();i++){
    result += play.get(i);
    result += ", ";
}
result=result.substring(0, result.length()-1);
System.out.println(result);


for(int i=0; i< play.size() ; i++ ){
    System.out.printf(play.get(i)+" ");
}
```

나래님 답

3. 배열 값을 역순으로 출력하기



.arrayList, asList사용 -> 역순출력

```
ArrayList<Integer> arr=new ArrayList<>(Arrays.asList(1,7,2,4,5,3,6));

arr.sort(Comparator.naturalOrder());
System.out.println(arr);
```

```
[1, 2, 3, 4, 5, 6, 7]
```

오름차순으로 풀어본 것

```
        String result="";
        ArrayList<Integer> arr=new ArrayList<>(Arrays.asList(1,7,2,4,5,3,6));

        for (int i=arr.size()-1; i>=0; i--){
            result += arr.get(i);

        }System.out.println(result);
//        arr.sort(Comparator.naturalOrder());
//        System.out.println(arr);

    }
}
```

```
6354271
```

for문으로 풀어본 것 => 주소값이 0~6이니까 길이에서 -1을 해주고 끝 값이 0이어야 함

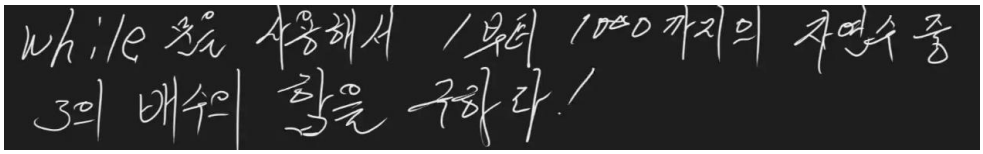
4. 맞는 것은?

⇨ everywhere

*if같은 경우에
그 경우가 해당하면
출력하고 나간다.
*else if니까 true가
나오는 순간 stop된다.

```
public class Sample {
    public static void main(String[] args) {
        String a = "write once, run anywhere";
        if (a.contains("wife")) {
            System.out.println("wife");
        } else if (a.contains("once") && !a.contains("run")) {
            System.out.println("once");
        } else if (!a.contains("everywhere")) {
            System.out.println("everywhere");
        } else if (a.contains("anywhere")) {
            System.out.println("anywhere");
        } else {
            System.out.println("none");
        }
    }
}
```

5. while 문을 사용해서 1~1000까지 3의 배수의 합을 구하라




```java
public static void main(String[] args) {
    int num=1;
    int sum=0;
    while (num<=100){
        num++;
        if (num%3==0){
            sum+=num;
//              System.out.println(sum);
        }
    }
    System.out.println(sum);
```

1683

1~1000인데, 100까지로 보고 100까지 구했다. 구한 방법

```java
public static void main(String[] args) {
    int num=1;
    int sum=0;
    while (num<=1000){
        num++;
        if (num%3==0){
            sum+=num;
//              System.out.println(sum);
        }
    }
    System.out.println(sum);
```

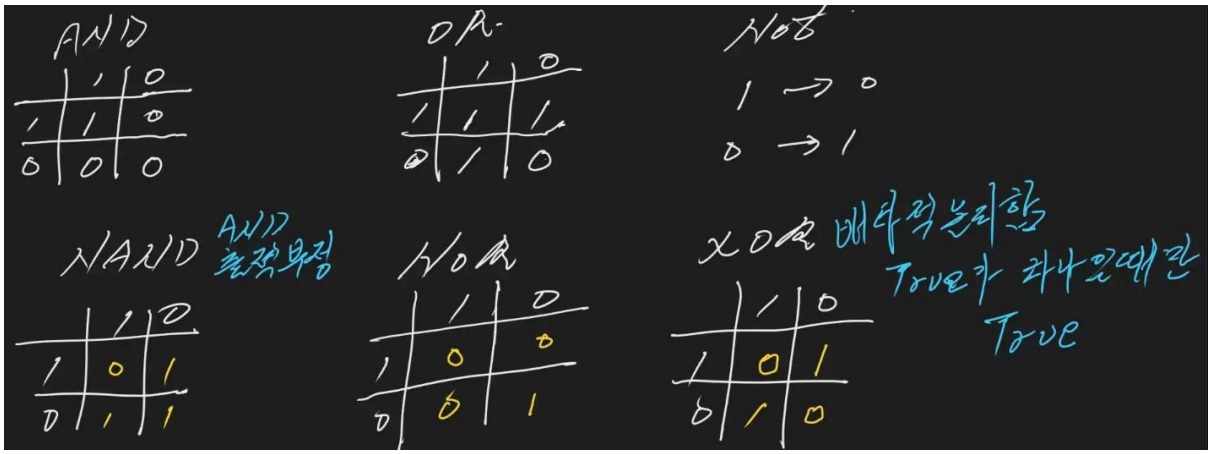
166833

1~1000까지 구한 것(위에 것에서 숫자만 바꿔줌)

```java
int sum = (n / d) * ((n / d) + 1) / 2 * d;

System.out.println( sum);


int n = 1000;
int d = 3;
int hap = 0;

for (int i = 1; i <= n; i++) {
    if (i % d == 0) {
        hap += i;
    }
}
    System.out.println(hap);
```

선생님 답(sum을 계산식에 넣어서 계산할 수 있다)

- 논리연산자



AND 두개 다 true여야 true

OR 한개만 true 여도 true, 둘다 false일 때, false 나옴

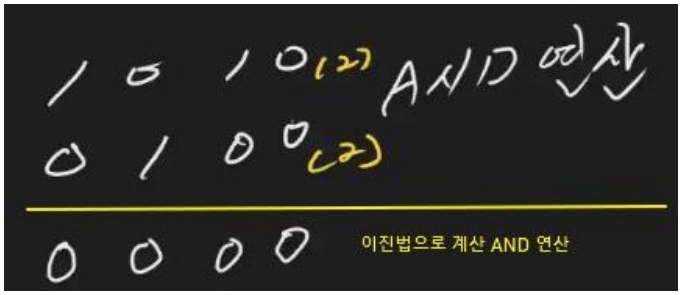
NOT true이면 false, false이면 true

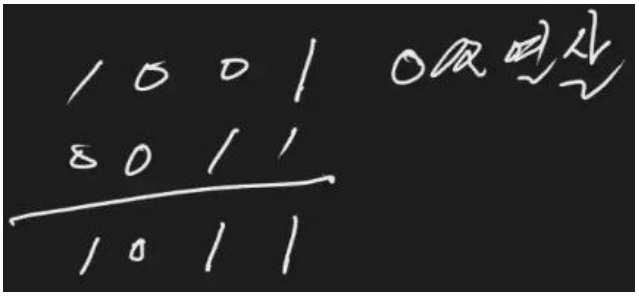
NAND 두개 다 true 이면 false (출력부정) _ and의 반대 값이 출력

NOR 한개라도 ture 이면 false (출력부정) _ or의 반대 값이 출력

XOR 배타적논리합 true가 하나일 때만 true

https://m.blog.naver.com/junb7/222792228046





- Switch문

case별로 출력

```
    switch (month){
          byte_short_char_int_enum String
        case 1: monthString = "jan";
            break;
        case 2: monthString = "feb";
            break;
        case 3: monthString = "mar";
            break;
        case 4: monthString = "apr";
            break;
        default: monthString = "없어요";
            break;
    }
    System.out.println(monthString);
}
```

- While문

```java
public class JavaExm12_15_07 {
    public static void main(String[] args) {
        int treeTen=0;
        while(treeTen<10){
            treeTen++;
            System.out.println("나무를 " +treeTen+ "번 찍었습니다.");
            if (treeTen==10){
                System.out.println("나무 넘어가요!");
            }
```

```
나무를 2번 찍었습니다.
나무를 3번 찍었습니다.
나무를 4번 찍었습니다.
나무를 5번 찍었습니다.
나무를 6번 찍었습니다.
나무를 7번 찍었습니다.
나무를 8번 찍었습니다.
나무를 9번 찍었습니다.
나무를 10번 찍었습니다.
나무 넘어가요!
```

while문은 조건이 다 마무리 될 때까지 실행하는 것. (0에서 시작해서 10미만까지 실행할거다)

- For문

```java
public class JavaExm12_15_07 {
    public static void main(String[] args) {
        for (int i = 2; i < 10; i++) {
            for (int j = 1; j < 10; j++) {
                System.out.println(i*j+ " ");
                System.out.print(i+"*"+j+"="+i*j+ " ");
            }
            System.out.println("");
        }
```

```
2*1=2 2*2=4 2*3=6 2*4=8 2*5=10 2*6=12 2*7=14 2*8=16 2*9=18
3*1=3 3*2=6 3*3=9 3*4=12 3*5=15 3*6=18 3*7=21 3*8=24 3*9=27
4*1=4 4*2=8 4*3=12 4*4=16 4*5=20 4*6=24 4*7=28 4*8=32 4*9=36
5*1=5 5*2=10 5*3=15 5*4=20 5*5=25 5*6=30 5*7=35 5*8=40 5*9=45
6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36 6*7=42 6*8=48 6*9=54
7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49 7*8=56 7*9=63
8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64 8*9=72
9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```

구구단을 for문으로 실행

```java
    public static void main(String[] args) {
        for(int i=1; i<10; i++) {
//
            for(int j=2; j<10; j++) {
//              System.out.print(j+"*"+i+"="+j*i+" ");
                System.out.format("%d * %d = %d",i,j,(i*j)," ");
            }
            System.out.println("");  // 줄을 바꾸어 출력하는 역할을 한다.
        }
```

format을 쓰면 %d와 %s 를 쓸 수 있다.

```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\
1 * 2 = 21 * 3 = 31 * 4 = 41 * 5 = 51 * 6 = 61 * 7 = 71 * 8 = 81 * 9 = 9
2 * 2 = 42 * 3 = 62 * 4 = 82 * 5 = 102 * 6 = 122 * 7 = 142 * 8 = 162 * 9 = 18
3 * 2 = 63 * 3 = 93 * 4 = 123 * 5 = 153 * 6 = 183 * 7 = 213 * 8 = 243 * 9 = 27
4 * 2 = 84 * 3 = 124 * 4 = 164 * 5 = 204 * 6 = 244 * 7 = 284 * 8 = 324 * 9 = 36
5 * 2 = 105 * 3 = 155 * 4 = 205 * 5 = 255 * 6 = 305 * 7 = 355 * 8 = 405 * 9 = 45
6 * 2 = 126 * 3 = 186 * 4 = 246 * 5 = 306 * 6 = 366 * 7 = 426 * 8 = 486 * 9 = 54
7 * 2 = 147 * 3 = 217 * 4 = 287 * 5 = 357 * 6 = 427 * 7 = 497 * 8 = 567 * 9 = 63
8 * 2 = 168 * 3 = 248 * 4 = 328 * 5 = 408 * 6 = 488 * 7 = 568 * 8 = 648 * 9 = 72
9 * 2 = 189 * 3 = 279 * 4 = 369 * 5 = 459 * 6 = 549 * 7 = 639 * 8 = 729 * 9 = 81
```

%d 와 %s를 써줘서 출력한 것 -> **format**을 쓸 때 가능

i와 j를 거꾸로 써줘서 결과값이 1*2, 1*3 … 이렇게 설정해준 것

```java
public static void main(String[] args) {
    int[] mark={90,55,60,75,85,43};
    for (int i=0;i<mark.length;i++){
        if(mark[i] < 60){
            continue;
        }
        System.out.println((i+1)+"학생 합격");

        //for문
//          if(mark[i] >= 60){
//              System.out.println((i+1)+"번은 합격");
//          }
//          else {
//              System.out.println((i+1)+"번은 아니지롱");
//          }
    }
```

+ continue : continue를 쓰면 true가 되었을 때 조건문으로 돌아간다 (for문)

(그래서 그 값이 해당되면 다시 조건문으로 돌아가는 것.)

Println을 쓰면 강제 줄바꿈이 적용된다. 그래서 print를 쓰면 줄바꿈이 적용되지 않음

```java
public static void main(String[] args) {
    for(int i=1; i<10; i++) {
        for(int j=2; j<10; j++) {
            System.out.print(j+"*"+i+"="+j*i+" ");
            System.out.format("%d * %d = %d",i,j,(i*j)," ");
        }
        System.out.println(""); // 줄을 바꾸어 출력하는 역할을 한다.
```

이중for문 문제

1. 결과값만 나오는 것에서 몇단인지 보여주게 설정하기
2. 2단 한줄씩인것을 2,3,4,단 으로 해주기

?? 궁금한것 continue가 if문 안에 사용되는 경우도 있는가 => 있다.