< 21. 12. 27 복습 >

< 문제은행 [2] >

- 문제 2.

```
final int START = 2;

for (i = START; i < 20; i++) {
    result = first + second;
    first = second;
    second = result;
    // 전체 뿌리기
    System.out.printf("%d번째 항 %d\n", i+1, result);
}
```

-> for문 초기식과 조건식에 숫자를 직접 입력하기보다는 변수를 활용해야 한다.

Ex) final int START = 2;

final int END = 20;

for (i = START ; i < END ; i++)

- 문제 10.

for문의 작동 순서

: for(초기화 ; 조건문 ; 증감문)

```
for (int i = 0; i < MAX; i++) {
    arr[i] = i + 1;
    System.out.printf("arr[%d] = %d\n", i, arr[i]);
}
```

1. 초기화 -> 2. 조건문 : 참 -> 3. 조건이 참일 때 실행될 문장 -> 4. 증감문

-> 5. 조건문 : 참 -> 6. 조건이 참일 때 실행될 문장 -> 7. 증감문

-> 8. 조건문 : 거짓 -> 9. for문 종료

< 배열 >

- 배열이란 동일한 자료형의 데이터를 연속된 공간에 저장하기 위한 자료구조이다.

- 배열을 만드는 방법

```
자료형[] 변수 = new 자료형[배열 크기];
변수[0] = 데이터 값;
변수[1] = 데이터 값;
```

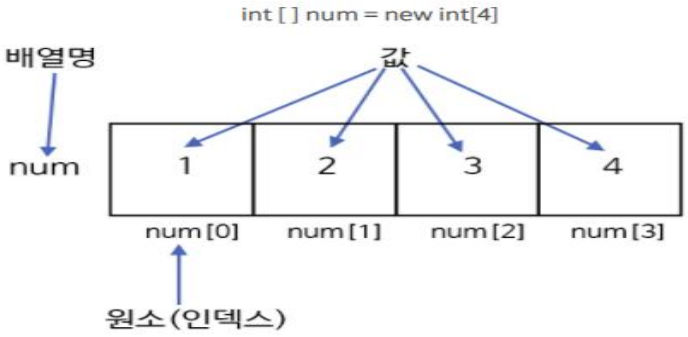
```
int[] arr = new int[MAX];

arr[0] = 1;
arr[1] = 1;
```

- 배열의 인덱스

: 배열의 각 저장공간을 배열의 요소라고 하며 배열이름[인덱스] 형식으로 배열의 요소에 접근한다.

인덱스(index)는 배열의 요소마다 붙여진 일련번호로 각 요소를 구별하는데 사용된다.

인덱스의 범위 : 0 ~ 배열길이 -1



-배열을 사용하는 이유

: 연관된 데이터를 저장하기 위한 변수의 선언을 줄여주며 반복문 등을 이용하여 계산과 같은 과정을 쉽게 처리할 수 있다.

- 배열은 for문과 궁합이 좋다.

```
for (int i = 0; i < MAX; i++) {
    arr[i] = i + 1;
    System.out.printf("arr[%d] = %d\n", i, arr[i]);
}
```

< Scanner >

- 자바에서는 Scanner를 이용해 사용자의 입력을 받을 수 있다.

- Scanner는 Class 타입의 데이터 타입이다.

- 기본적인 데이터 타입들을 Scanner의 메소드를 사용하여 입력받을 수 있다.

  ex) String(문자열)을 입력받고 싶다면 next( )나 nextLine( )

- 공백(띄어쓰기) 또는 개행(줄 바꿈)을 기준으로 읽는다.

-Scanner 객체 생성

```
클래스_이름  객체_이름 = new 클래스_이름();
```

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

-메소드를 이용하여 입력받기

```
int num = scan.nextInt();
```

scan.nextInt( )를 통해 정수를 입력 받을 수 있게 도와준다.

메소드명	설명
nextBoolean()	boolean의 자료형을 입력받습니다. 자료형은 두개인 true,false로 대소문자를 구분하지 않습니다.
nextByte()	byte의 자료형을 입력으로 받습니다. 입력 범위(127~ -128) 밖이면 InputMismatchException이 발생합니다.
nextShort()	short의 자료형을 입력받습니다.
nextInt()	int형의 자료형을 입력받습니다.
nextLong()	long형의 자료형을 입력받습니다.
nextDouble()	double형의 자료형을 입력받습니다.
nextFloat()	float형의 자료형을 입력받습니다.
next()	String형의 문자열을 입력받습니다. 이때 공백 문자까지 입력받습니다.
nextLine()	문자열을 입력받는데 다른 next~() 메소드와 다른 점은 줄단위로 입력받는다는 점입니다.

<BigInteger>

- BigInteger을 사용하는 이유

: int 메모리 크기인 4byte로 표현할 수 있는 범위는 -2,147,483,648 ~ 2,147,483,647으로

이 범위를 넘어서게 되면 값들이 제대로 출력되지 않는다.

반면 BigInteger는 문자열 형태로 이루어져 있어 숫자의 범위가 무한하기 때문에 아주 큰 숫자를 다루는

은행이나 돈과 관련된 개발을 진행할 때 사용하기 적합하다.

-BigInteger 계산

```
BigInteger bigNumber1 = new BigInteger("100000");
BigInteger bigNumber2 = new BigInteger("10000");

System.out.println("덧셈(+) :" +bigNumber1.add(bigNumber2));
System.out.println("뺄셈(-) :" +bigNumber1.subtract(bigNumber2));
System.out.println("곱셈(*) :" +bigNumber1.multiply(bigNumber2));
System.out.println("나눗셈(/) :" +bigNumber1.divide(bigNumber2));
System.out.println("나머지(%) :" +bigNumber1.remainder(bigNumber2));
```