

## 21.12.23 복습

- 코드 작성 시 상수값을 통해 구성하는 것보다 변수명을 활용해 변경사항이 생겼을 때 유연하게 대처할 수 있도록 구성하는 것이 바람직하다.

### <변수명 작성 규칙>

1. 변수명은 대소문자를 구분한다.
2. 예약어는 변수명으로 사용이 불가능하다.

ex) int int; (x)

3. 숫자는 뒤에 와야 한다.

ex) int 23a; (x)

4. 상수는 변수로 사용 불가능하다.

5. under bar는 사용 가능

ex) int \_abc;

6. 연산자는 사용 불가능하다.

7. Camel Notation: 낙타 표기법

= 각 단어의 첫 글자를 대문자로 표기하고 붙여 쓰되, 맨 처음 단어는 소문자로 표기

띄어쓰기 대신 대문자로 단어를 구분하는 표기 방식

ex) int myNumber;

boolean isNext;

8. 헝가리안 표기법 : 변수의 자료형을 변수명의 접두어로 붙이는 방식

ex) int iCharPosEdit;

### <관습>

- 상수값을 지정할 때 변수명은 전부 대문자로 만든다.
- 클래스 이름은 대문자로 시작한다. 만약 다중 단어로 구성된다면 단어 이니셜마다 대문자로 구성한다.
- 변수나 메서드(함수)의 경우 소문자로 시작한다. 이후에는 클래스와 동일

### <final>

- final은 상수값을 지정한다. 상수값을 지정할 때 변수명은 전부 대문자로 만든다.

ex) final int MAX1 = 90;

- final을 사용하는 이유?

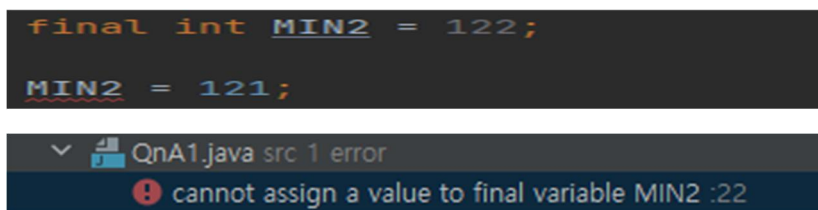
-> 수천 개의 파일, 수만 ~ 수십만 라인으로 구성된 프로그램의 경우에도 final을 사용해 관리하면 단순히 선언부의 숫자만 바뀌도 모든 파일과 모든 코드에 이를 적용할 수 있다는 이점이 있다.

- final과 int의 차이

-> int는 선언 후에도 값 변경이 가능하다.

반면 final은 상수를 만들기 때문에 선언 후 변경이 불가능하다.

```
final int MIN2 = 122;  
  
MIN2 = 121;
```



QnA1.java src 1 error  
cannot assign a value to final variable MIN2 :22

## < 논리연산자 >

||는 OR연산자: 두 개의 조건 중 하나만 일치하면 진행

&&는 AND연산자: 두 개의 조건 모두 만족해야 진행

//OR는 합집합, AND는 교집합

!은 NOT연산으로 참을 거짓으로 만들고 거짓을 참으로 만든다.

## <복합 대입 연산자>

대입연산자와 다른 연산자가 함께 사용되는 연산자로 변수를 중복해서 사용하는 것을 줄여주는 역할을 한다.

```
+=, -=, *=, /=, %=
x += y ==> x = x + y
x -= y ==> x = x - y
x *= y ==> x = x * y
x /= y ==> x = x / y
x %= y ==> x = x % y
```

## <While>

- while ( 조건 )

{

조건이 만족할 경우 수행할 내용

}

- 무한 반복

:조건이 항상 true이기 때문에 while문 내부 내용이 무한 반복된다.

-> While(true) {

수행할 내용

}

## <증감 연산자>

프로그램에서 변수값을 1 증가시키거나 1 감소시키는 상황에 자주 쓰이는 연산자로

증가연산자(++)과 감소연산자(--)로 나뉜다.

증가연산자는 변수값을 1 증가시키고, 감소연산자는 변수값을 1 감소시킨다.

또한 증감연산자는 전위(++n)과 후위(n++)로 나뉜다.

- 전위 연산자 vs 후위 연산자

```
int i = 0;
```

전위 연산자의 경우 해당 줄이 실행되기 이전에 덧셈이 완료된다.

```
System.out.println("i = " + (++i));
```

-> 1이 출력된다.

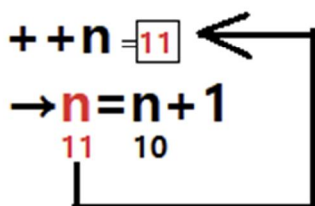
후위 연산자의 경우 해당 줄이 실행된 이후에 덧셈이 완료된다.

```
System.out.println("i = " + (i++));
```

-> 0이 출력된다.

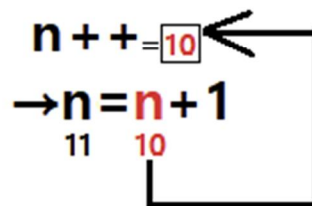
### <전위>

만약, n=10 이면



### <후위>

만약, n=10 이면



## <For문>

- for( 초기화 ; 조건 ; 증감)

{

조건이 만족된다면 동작시킬 코드

}

```
for (int i = 1; i <= 10; i++) {  
    System.out.println("i = " + i);  
}
```

초기화 부분은 for문 진입 시 최초 한 번만 실행된다.

조건은 매 반복마다 검사한다.

증감은 중괄호 내부의 내용이 끝나면 진행한다.

초기화, 증감, 조건 모두 없어도 된다. 이 경우에는 for문 내부의 코드가 무한반복된다.

- for문 무한반복

for ( ;; )

{

동작시킬 코드

}