

# [2기][Item 57] 지역변수의 범위를 최소화하라

지역 변수의 범위를 최소화 해야 한다

지역변수의 범위를 줄이는 기법

변수는 가장 처음 쓰일 때 선언한다.

거의 모든 지역변수는 선언과 동시에 초기화해야 한다.

메서드를 작게 유지하고 한 가지 기능에 집중한다

참고자료

## 지역 변수의 범위를 최소화 해야 한다

- 클래스와 멤버의 접근 권한을 최소화하라는 이야기와 상통
- 지역변수의 유효 범위를 최소로 줄이면 코드 가독성과 유지보수성이 높아지고 오류 가능성이 낮아진다.

## 지역변수의 범위를 줄이는 기법

1. 가장 처음 쓰일 때 선언한다.
2. 거의 모든 지역변수는 선언과 동시에 초기화해야 한다.
  - try-catch
  - for 반복문의 반복 변수반복 변수가 반복문 밖에서도 사용되는데 아니라면 while 보다 for 문이 더 좋다
3. 메서드를 작게 유지하고 한 가지 기능에 집중한다

## 변수는 가장 처음 쓰일 때 선언한다.

- 사용하려면 멀었는데 미리 선언하면 코드가 어수선해져 가독성이 떨어진다.
- 변수를 실제로 사용하는 시점에 타입과 초깃값이 기억나지 않을 수 있다.

- 변수가 쓰이는 범위보다 너무 앞서 선언하거나 다 쓴 뒤에도 살아 있으면 좋지 않다.  
→ 의도한 범위 내에서만 사용하고 범위 앞 or 뒤에서 변수를 사용하지 않도록 하자

## 거의 모든 지역변수는 선언과 동시에 초기화해야 한다.

- 초기화에 필요한 정보가 충분치 않다면 충분해질 때까지 선언을 미뤄야 한다.

### 1) try-catch

- 초기화하는 표현식에서 예외를 던질 가능성이 있다면 try 블록 내에서 초기화  
그렇지 않으면 예외가 블록을 넘어 메서드까지 전파된다.
- ▼ 변수 값을 try 블록 바깥에서도 사용해야 한다면 선언을 try 블록 앞에서 한다.

```
public static void main(String[] args) {
    // Translate the class name into a Class object
    Class<? extends Set<String>> cl = null;
    try {
        // Unchecked cast!
        cl = (Class<? extends Set<String>>) Class.forName(args[0]);
    } catch (ClassNotFoundException e) {
        fatalError("Class not found.");
    }

    // Get the constructor
    Constructor<? extends Set<String>> cons = null;
    try {
        cons = cl.getDeclaredConstructor();
    } catch (NoSuchMethodException e) {
        fatalError("No parameterless constructor");
    }

    ...
}
```

### 2) 반복문

- for 반복문에서 사용되는 반복 변수의 사용 범위
  - 반복문의 몸체

- for 키워드와 반복문 사이의 괄호 안
- 반복 변수가 반복문 밖에서도 사용 되는게 아니라면 while 문 보다 for 문을 쓰는 것이 좋다.

▼ 컬렉션 순회 시의 권장 관용구

```
for (Element e: c) {
    ...
}
```

반복자가 필요한 경우 for-each 가 아닌 전통적인 for 문을 사용하는 아래의 관용구가 권장된다.

반복자가 필요할 때? 컬렉션을 순회하면서 원소를 삭제해야 하는 경우 등

(이 경우는 반복자를 사용하지 않으면 ConcurrentModificationException이 발생할 수 있다)

```
for (Iterator<Element> i = c.iterator(); i.hasNext(); ) {
    Element e = i.next();
    ...
}
```

- 복사 붙여넣기 오류
  - for 문은 while문 보다 짧아서 가독성이 좋다.
- ▼ 같은 값을 반환하는 메서드를 매번 호출할 때 권장되는 반복문 관용구

```
for (int i = 0; i < expensiveComputation(); i++) {
    ...
}
```

```
for (int i = 0, n = expensiveComputation(); i < n; i++) {
    ...
}
```

## 메서드를 작게 유지하고 한 가지 기능에 집중한다

- 한 메서드에서 여러 가지 기능을 처리하게 되면  
한 기능에만 관련된 지역변수라도 (다른 기능에서는 사용하지 않더라도)  
다른 기능을 수행하는 코드에서 접근이 가능해져 버린다  
→ 메서드를 기능 별로 쪼개서 해결한다.

## 참고자료

### 자바 메모리 구조 뿌시기 [ JVM이란? ]


자바를 처음 배우려고 하시는 분들, 자바를 사용하고 계시지만 메모리 구조에 대한 감이 안잡히는 분들도움이 되셨으면 합니다 ^^

 <https://youtu.be/AWXPnMDZ9I0>



### [자바] 컬렉션에서 원소 삭제하기 (ConcurrentModificationException 피하면서)

리스트를 순회하면서 특정 원소를 삭제하고 싶을 때가 있습니다. 예를 들어, 다음과 같이 알파벳과 숫자가 섞여있는 리스트가 있다고 가정해봅시다. 저는 이 리스트에서 숫자인 원소들은 모두 삭제하고 싶습니다. 가장 먼저 떠오르는 방법은 boolean remove(Object o) 메소드를 사용하는 것입니다. for 루프를 돌면서 해당 원소가 숫자인지 체크 후에 숫자이면

 <https://www.daleseo.com/how-to-remove-from-list-in-java/>