EJ

이펙티브 자바 CP.8

작성 일시	@2023년 3월 4일 오후 2:23
최종 편집 일시	@2023년 3월 4일 오후 3:45
유형	이펙티브 자바
작성자	종현 박
참석자	
언어	

8️⃣ 일반적인 프로그래밍 원칙

▼ 57. 지역변수의 범위를 최소화하라

- 개요
  - 클래스와 멤버의 접근 권한을 최소화하라 (아이템 15)와 취지가 비슷하다.
  - 지역변수의 유효 범위를 최소로 줄이면 코드 가독성과 유지보수성이 높아지고 오류 가능성은 낮아진다.
- **지역변수의 범위를 줄이는 가장 좋은 방법은 역시 가장 처음에 쓰일 때 선언하기 이다.**

- 지역변수를 생각 없이 선언하다 보면 변수가 쓰이는 범위보다 너무 앞서 선언하거나, 다 쓴 뒤에도 여전히 살아 있게 되기 쉽다.

```
// C++, 개인적으로 학부생때 이런 코드를 짜는데, 이해가 안 됐다.
int i, j;
for (i = 0; i < 10; i++) {
    ...
}
... // i 관련된 코드 없음
```

- 지역변수의 범위는 선언된 지점부터 그 지점을 포함한 블록이 끝날 때까지이므로, 실제 사용하는 블록 바로 바깥에 선언된 변수는 그 블록이 끝난 뒤까지 살아있게 된다.

- **거의 모든 지역변수는 선언과 동시에 초기화해야 한다.**
  - 초기화에 필요한 정보가 충분하지 않다면 충분해질 때까지 선언을 미뤄야 한다.
  - `try-catch` 문에서는 예외다.
    변수를 초기화하는 표현식에서 검사 예외를 던질 가능성이 있다면 `try` 블록 안에서 초기화해야 한다.
    변수 값을 `try` 블록 바깥에서도 사용해야 한다면 `try` 블록 앞에서 선언해야한다.
- 반복문
  - 반복문의 변수의 값을 반복문 종료된 뒤에도 써야 하는 상황이 아니라면 `while` 문 보다는 `for` 문을 쓰는 편이 낫다.
  - ex) 컬렉션이나 배열을 순회하는 권장 관용구

```
for (Element e : c) {
    ... // e 로 무언가 한다.
}
```

  - ex) 컬렉션이나 배열의 index 를 사용해야 하는 경우의 관용구

```
for (Iterator<Element> i = c.iterator(); i.hasNext(); ) {
    Element e = i.next();
    ... // e와 i 로 무언가 한다.
}
```

  - ex) 문제가 될 수도 있는 상황

```
Iterator<Element> i = c.iterator();
while(i.hasNext()) {
    doSomething(i.next());
}
...
Iterator<Element> i2 = c2.iterator();
while(i.hasNext()) {  // 버그
    doSomethingElse(i2.next());
}
```

        - 문제 `i2.hasNext()` 가 아닌 `i.hasNext()` 로 항상 비어 있다고 생각 할 수 있다.
        - `for` 문을 사용하면 반복문 안에서 지역변수가 초기화, 종료 됨으로, 이런 문제가 나올 수 없다.
    - ex) 문제 코드를 `for`문 으로 변경한 코드

```
for (Iterator<Element> i = c.iterator(); i.hasNext(); {
    Element e = i.next();
    ...
}
// i 를 찾을 수 없다며 컴파일 에러를 낸다.
for (Iterator<Element> i2 = c.iterator(); i.hasNext(); {
    Element e2 = i2.next();
    ...
}
```

        - `for` 문의 경우 복붙하는 코드에서 `c`를 `c2` 만 바꿔줘도 된다.
        - while 문 보다 짧아서 가독성이 좋다.
- 메서드를 작게 유지하고 한 가지 기능에 집중하도록 만드는게 좋다.
    - 한 메서드에서 여러 가지 기능을 처리한다면 그 중 한 기능과만 관련된 지역변수라도 다른 기능을 수행하는 코드에서 접근할 수 있을 것이다.

▼ 58. 전통적인 for 문보다는 for-each문을 사용하라

▼ 59. 라이브러리를 익히고 사용하라

▼ 60. 정확한 답이 필요하다면 float 과 double은 피하라

▼ 61. 박싱된 기본 타입보다는 기본 타입을 사용하라

▼ 62. 다른 타입이 적절하다면 문자열 사용을 피하라

▼ 63. 문자열 연결은 느리니 주의하라

▼ 64. 객체는 인터페이스를 사용해 참조하라

▼ 65. 리플렉션보다는 인터페이스를 사용하라

▼ 66. 네이티브 메서드는 신중히 사용하라

▼ 67. 최적화는 신중히하라

▼ 68. 일반적으로 통용되는 명명 규칙을 따르라