

[2기][아이템40] Override 주석을 일관성 있게 사용하자

Point

- 수퍼 타입의 메소드를 오버라이드 하는 모든 메소드에 Override 주석을 달아주면 컴파일러가 굉장히 많은 에러를 막아 줄 수 있다.
- 단 실제 클래스에서 수퍼 클래스의 추상 메소드를 오버라이드하는 메소드에 주석을 달 필요는 없다
(애네는 어차피 컴파일러가 검사해주기 때문이다)

Code

```
public class Combination {
    private final char first;
    private final char second;

    public Combination(char first, char second) {
        this.first = first;
        this.second = second;
    }

    public boolean equals(Combination b) {
        return b.first == first && b.second == second;
    }

    public int hashCode() {
        return 31 * first + second;
    }
}
```

```
public static void main(String[] args) {
    Set<Combination> s = new HashSet<>();
    for (int i = 1; i <= 3; i++)
        for (char ch = '1'; ch <= '3'; ch++)
            s.add(new Combination(ch, ch)); // ['1','1'], ['2','2'], ['3','3']
    System.out.println(s.size());
}
```

▼ console

9

▼ why?

equals(...) 를 잘 재정의 했고 따라서 콘솔에 3이 찍힐 것을 기대할 수 있다.

equals(...) 를 재정의 할 때 hashCode() 도 같이 재정의 하라는 규칙도 잘 따른 듯 싶다.

하지만 위의 equals(...) 는 오버라이딩이 아니라 오버로딩을 한 경우이다.

```
public boolean equals(Object obj) {  
    return (this == obj);  
}
```



오버라이딩 아님! 오버로딩!

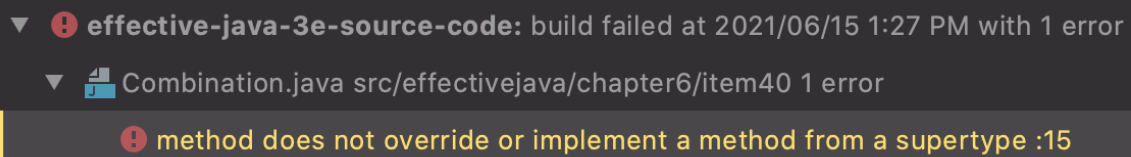
```
public boolean equals(Combination b) {  
    return b.first == first && b.second == second;  
}
```


따라서 객체의 동일성 판단시 Combination 의 equals(...) 을 사용하는게 아니라 Object 의 equals(...) 를 사용하게 되고 이는 객체 참조를 비교하여 동일 여부를 파악하기 때문에 Set 의 사이즈는 9가 출력된다.


@Override


- 수퍼 클래스의 메소드를 오버라이드 함을 명시적으로 나타내며, 오버라이딩에 문제가 있을 때 컴파일러의 도움을 받을 수 있다.

```
@Override
public boolean equals(Combination b) {
    return b.first == first && b.second == second;
}
```



▼  **effective-java-3e-source-code:** build failed at 2021/06/15 1:27 PM with 1 error

▼  Combination.java src/effectivejava/chapter6/item40 1 error

 **method does not override or implement a method from a supertype :15**

@Override 쪽에 컴파일 에러가 난다. 오버라이딩에 문제가 있음을 컴파일러가 도와주는 것이다.

```
@Override public boolean equals(Object o) {
    if (!(o instanceof Combination))
        return false;
    Combination b = (Combination) o;
    return b.first == first && b.second == second;
}
```

- @Override 주석을 다는 것이 필수는 아니지만 컴파일러의 도움을 받기 위해 습관적으로 일관성 있게 사용하는 것이 좋다.
- 추상 클래스나 인터페이스에서 자신의 수퍼 클래스나 수퍼 인터페이스 모든 메소드에 @Override 를 붙임으로써 수퍼 타입의 메소드를 재정의 할 뿐 새로운 메소드를 추가하고 싶지 않는 경우 도움을 받을 수 있다.
 - 예를 들어 Set 인터페이스는 Collection 인터페이스를 상속하고 있는데 Collection 인터페이스의 메소드를 재정의만 하고 새로운 메소드를 추가하고 있지 않다. 이 경우 Set 인터페이스의 모든 메소드에 @Override 주석을 달아 놓음으로써 실수로 새로운 메소드를 추가하는 경우를 방지할 수 있다.

굳이 @Override 를 달지 않아도 되는 경우 (예외)

- 추상 클래스를 상속받아 선언된 실체 클래스
- 인터페이스를 구현하기 위해 선언된 실체 클래스
- 해당 경우는 굳이 @Override 를 달지 않아도 오버라이딩 하지 않은 메소드가 하나라도 있다면 컴파일러가 알려준다.

```
interface Animal {  
    public abstract void cry();  
}
```

