

# *Effective Java Study*

Woowacourse\_study 4th



**Item 21 by 알파**

**In Java 8..**

**Java 8이 등장하면서 여러 가지가 바뀌게 되었다.**

**대표적으로 랴다, 스트림 등이 있지만**

**인터페이스 입장에서는 default 메소드가 추가된 것이다!**

**Default method? 뭘데 이건?**

**메소드의 접근 제어자가 default인 메소드로서,  
Default의 특징은 “같은 패키지 내”의 접근만 허용한다는 것!**

그런데 Java 8 이전 버전에서 인터페이스는 계속 쓰여왔다.  
그 당시의 인터페이스는 추상 클래스 말고는 어떠한 클래스도  
가질 수 없었다!

쓸 수 있는게 많아졌네?

“Java 8 이전에서 인터페이스는 무조건 추상 메소드만 가질 것  
이라는 전제 하에 인터페이스를 implement한 클래스는  
사실상 디폴트 메소드의 존재를 모른채 삽입된다!”

## Java 8에서 Collection에 추가된 removeIf 디폴트 메소드

```
@Override
public boolean removeIf(final Predicate<? super E> filter) {
    Objects.requireNonNull(filter);
    boolean result = false;
    for (Iterator<E> it = iterator(); it.hasNext(); ) {
        if (filter.test(it.next())) {
            it.remove();
            result = true;
        }
    }
    return result;
}
```

**만약 Multi-Thread라면?**

**removeIf에서 Lock을 걸어주지 않기 때문에 에러가 나거나  
원하지 않는 결과가 발생할 가능성이 매우 높다.**

**실제 해당 도서가 쓰여진 시점에서는 removeIf에 대한 조치  
가 이루어지지 않음.**

**해결책?**

**removeIf를 재정의하자!**



## removeIf 호출 이전에 lock!

```
@Override
public boolean removeIf(final Predicate<? super E> filter) {
    synchronized(lock) {
        return decorated().removeIf(filter);
    }
}
```

**왜 갑자기 public?**

**인터페이스의 메소드를 Override할 시 무조건 public으로  
Override 해야한다.**

“Java 플랫폼에 속하지 않는 제3의 기존 컬렉션 구현체들 (Apache의 synchronizedCollection)은 이런 언어 차원의 인터페이스 변화에 발맞춰 수정될 기회가 없었으며, 그 중 일부는 여전히 수정되지 않고 있다!”

## 결론!

“디폴트 메소드는 런타임 오류를 일으킬 수 있다. 인터페이스를 설계할 때는 여전히 세심한 주의를 기울여야 한다.

인터페이스라면 릴리즈 전에 반드시 테스트를 거쳐야 한다.”

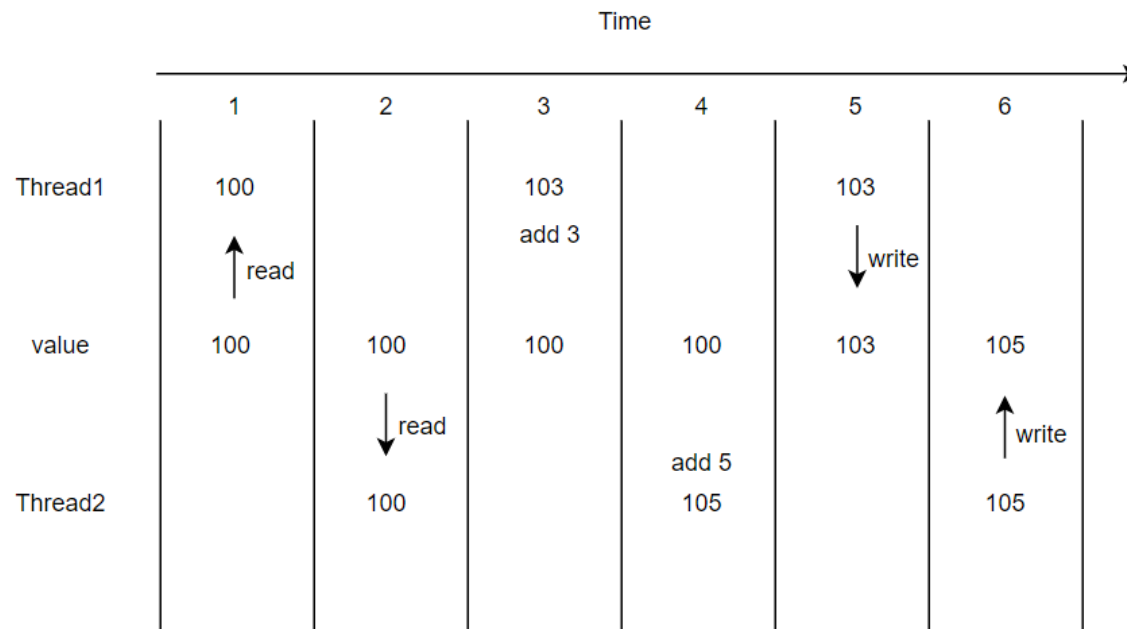
번외

왜 lock이 필요할까?

Thread 1 : sum += 3;

Thread 2 : sum += 5;

Answer is 105...?

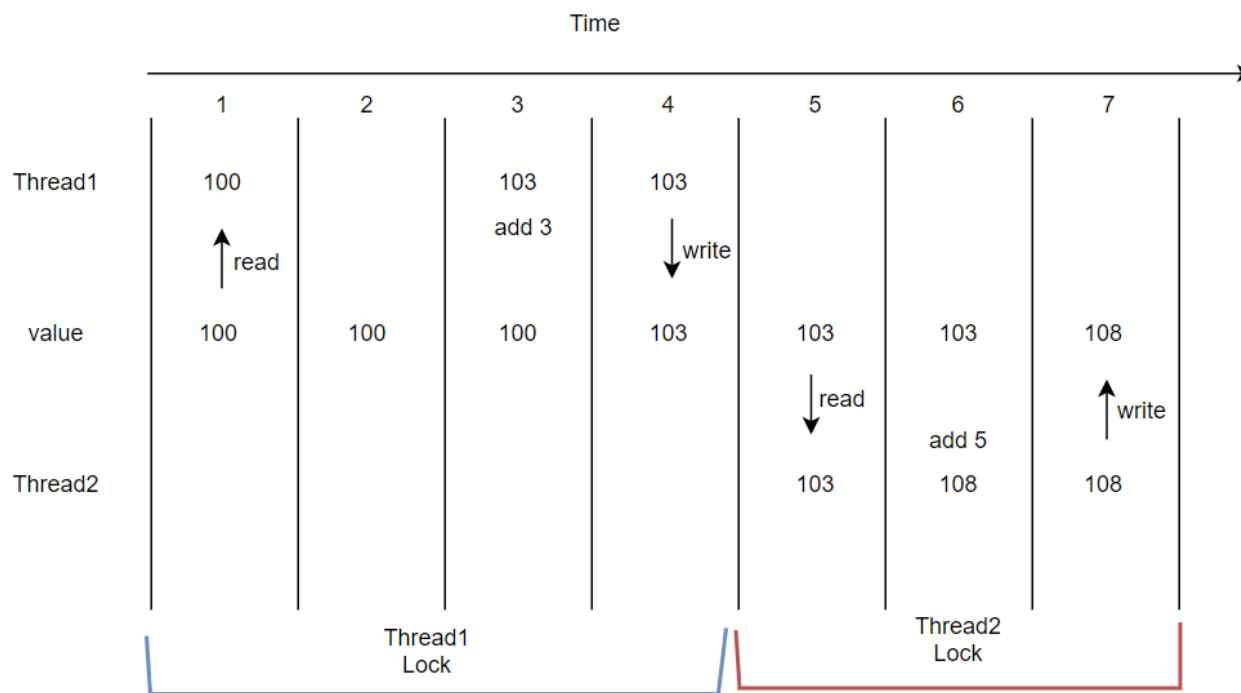


왜 lock이 필요할까?

Thread 1 : sum += 3;

Thread 2 : sum += 5;

Answer is 108...!



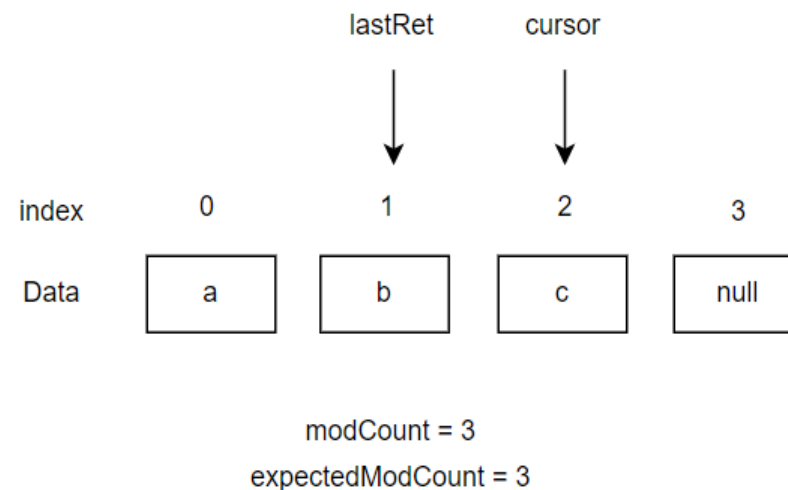
Iterator를 remove한다고?

Cursor : 다음 원소를 가리킴

lastRet : cursor 이전 원소를 가리킴

ModCount : 현재까지 원소의 개수

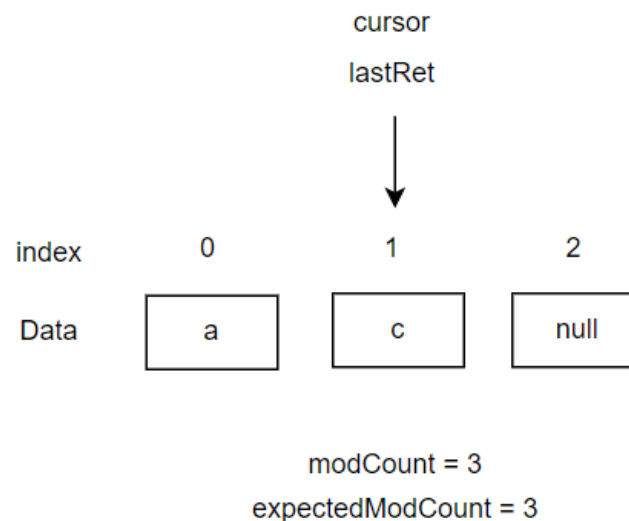
expectedModCount : 전체 원소 개수





Iterator를 remove한다고?

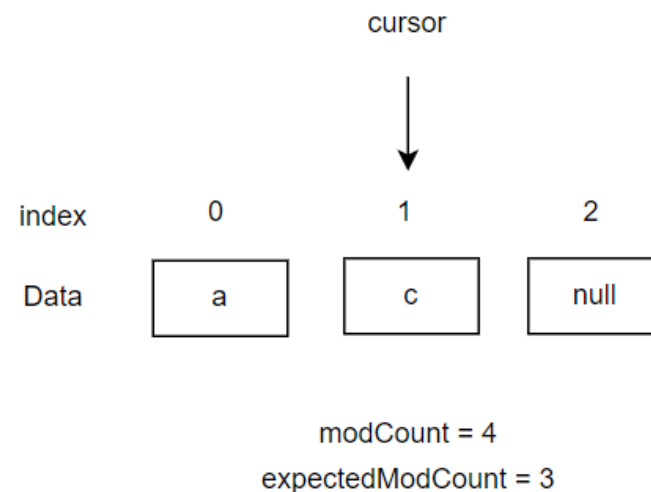
lastRet을 remove한 후,  
Cursor는 lastRet이 가리키는 원소를  
가리킨다.



Iterator를 remove한다고?

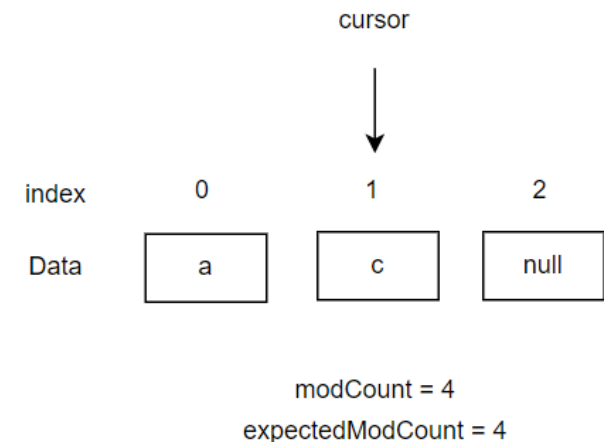
lastRet는 -1이 되고

modCount는 증가시킨다.



Iterator를 remove한다고?

ModCount와 expectedModCount는  
같아야 하므로 expectedModCount에  
현재 modCount 값을 복사한  
이후 hasNext에 의해 종료된다.



## References

Joshua Bloch, 『Effective Java 3/E』, 이복연 역 (서울 : 인사이트, 2018), pp. 136 - 138

Raul-Gabriel Urma, Mario Fusco, Alan Mycroft 『모던 자바 인 액션』, 우정은 역 (서울 : 한빛미디어, 2019), p. 57

남궁성, 『자바의 정석』 (경기 : 도우출판, 2016), p. 382

## References

<https://github.com/apache/commons-collections/blob/master/src/main/java/org/apache/commons/collections4/Collection/SynchronizedCollection.java>

<https://stackoverflow.com/questions/15993356/how-iterators-remove-method-actually-remove-an-object>

<https://brandpark.github.io/java/2021/01/24/iterator.html>

***E.O.D***



**Item 21 by 알파**