

# 이펙티브 자바

Item 54 ~ 56

2024/05/19

# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

방어적 복사 + 얇은 복사 + null 반환

```
private final List<Object> ObjectInStock = new ArrayList<>();

public List<Object> getObjects() {
    return ObjectInStock.isEmpty() ? null
        : new ArrayList<>(ObjectInStock);
}
```

NullPointerException 을 주의하자

null 반환에 대해서 예외 처리를 해야한다.

# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

## Null 반환

메모리 절약

Null 에 대한 처리 필요

VS

## Empty List 반환

NullPointerException 예방

메모리 낭비

예측 불가 에러(white paper)

# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

## 1. 빈 컬렉션 반환

```
public static List<Object> getObjects() {  
    return new ArrayList<>(ObjectInStock);  
}
```

## 2. 불변 객체 반환

```
List<Object> objects = Collections.emptyList();
```

# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

## 1. Size 0인 배열 생성

```
public Cheese[] getCheeses(){  
    return cheesesInStock.toArray(new Cheese[0]);  
}
```

## 2. 배열 재사용

```
private static final Cheese[] EMPTY_CHEESE_ARRAY = new Cheese[0];  
  
public Cheese[] getCheeses() {  
    return cheesesInStock.toArray(EMPTY_CHEESE_ARRAY);  
}
```

# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

**toArray(T[] a)** : list를 배열로 반환한다.

```
public Cheese[] getCheeses(){  
    return cheesesInStock.toArray(new Cheese[0]);  
}
```

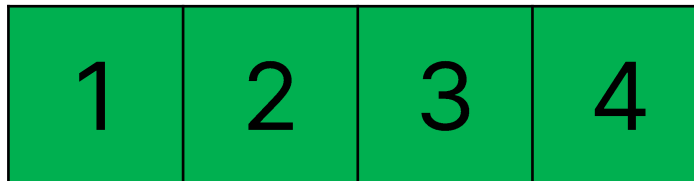
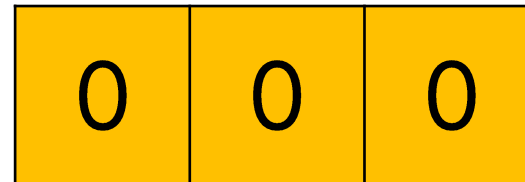
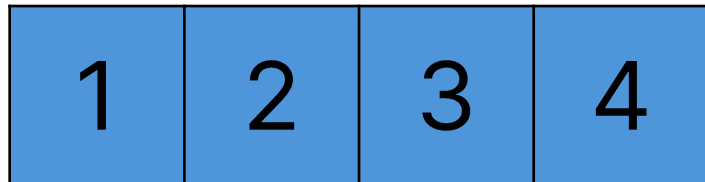
```
public <T> T[] toArray(T[] a) {  
    if (a.length < size)  
        // Make a new array of a's runtime type, but my contents:  
        return (T[]) Arrays.copyOf(elementData, size, a.getClass());  
    System.arraycopy(elementData, 0, a, 0, size);  
    if (a.length > size)  
        a[size] = null;  
    return a;  
}
```

1. 배열 길이 < 리스트 길이
2. 배열 길이 == 리스트 길이
3. 배열 길이 > 리스트 길이

# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

## 1. 배열 길이 < 리스트 길이

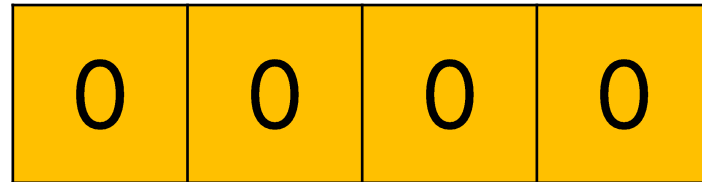
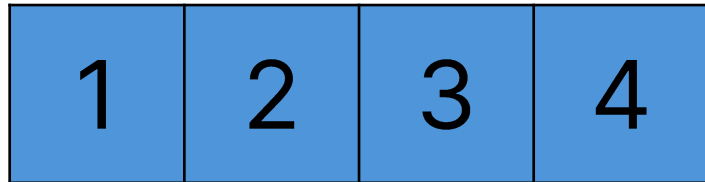
리스트 길이만큼 배열을 복사(깊은 복사)하여 반환(새로 생성)



# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

## 2. 배열 길이 = 리스트 길이

리스트 길이만큼 배열을 복사(깊은 복사)하여 반환

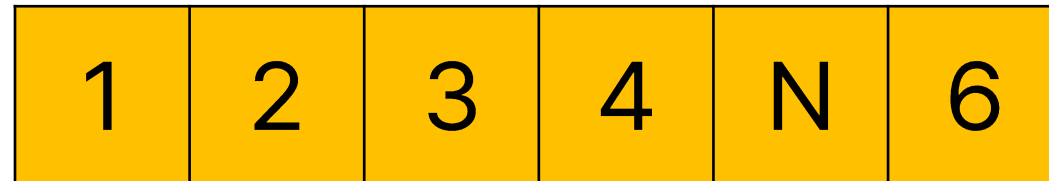
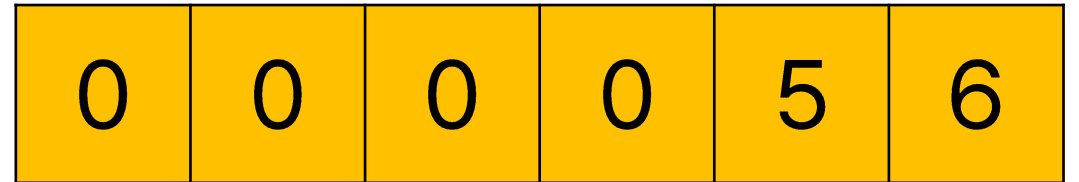
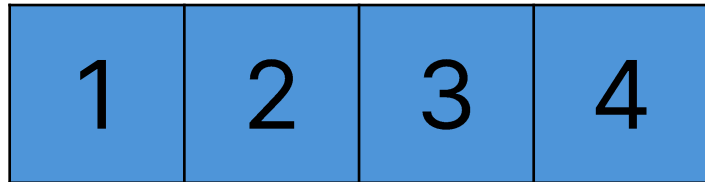




# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

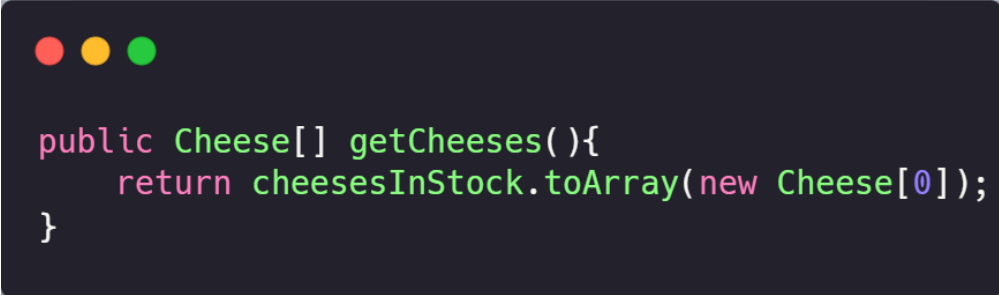
3. 배열 길이 > 리스트 길이

배열[리스트 길이]=null



# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

1. 파라미터의 형태(클래스 리터럴 X)
2. 파라미터가 인스턴스
3. 인스턴스의 배열 크기가 0



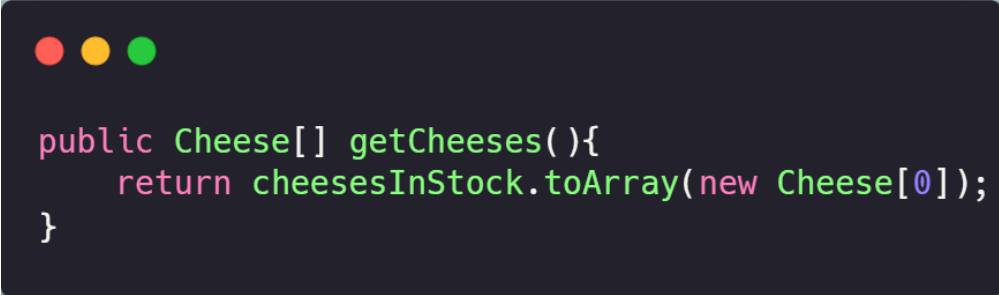
```
public Cheese[] getCheeses(){  
    return cheesesInStock.toArray(new Cheese[0]);  
}
```

# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

## 1. 파라미터의 형태(클래스 리터럴 X)

제네릭은 컴파일 타임에 타입이 정해진다.

적어도 반환 타입을 명시할 필요가 있다.



```
public Cheese[] getCheeses(){  
    return cheesesInStock.toArray(new Cheese[0]);  
}
```

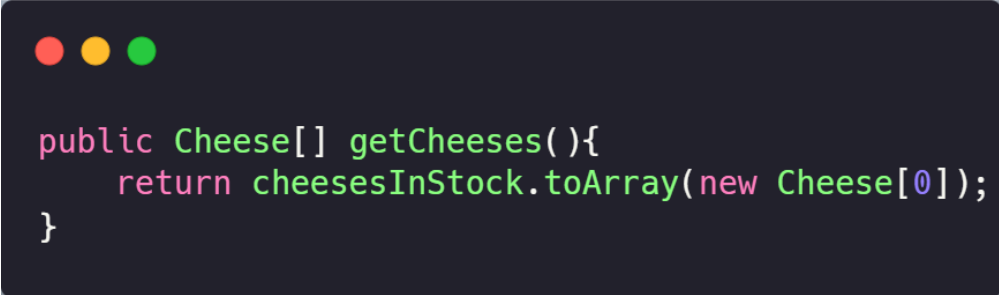
# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

## 2. 파라미터가 인스턴스

런타임에 타입을 확인할 방법이 없다.

Integer와 Integer[]는 다른 타입이다.

인스턴스 자체를 넘긴다.



```
public Cheese[] getCheeses(){  
    return cheesesInStock.toArray(new Cheese[0]);  
}
```

# Item 54 null이 아닌, 빈 컬렉션이나 배열을 반환하라

## 3. 인스턴스의 배열 크기가 0

배열크기가 리스트보다 크다면 불필요한  
공간이 생긴다.

크기가 0인 배열은 불변이다.



```
public Cheese[] getCheeses(){  
    return cheesesInStock.toArray(new Cheese[0]);  
}
```



```
public Object[] toArray() {  
    checkForComodification();  
    return Arrays.copyOfRange(root.elementData, offset, offset + size);  
}
```

# Item 55 옵셔널 반환은 신중히 하라

컬렉션의 경우 빈 컬렉션을 반환한다

그러면 단일 인스턴스는??

Optional<T> 를 사용하여 포장(Java 8)

Optional은 상태가 두개이다.

# Item 55 옵셔널 반환은 신중히 하라

내부적으로 예외 상황을 확인 할 수 있다.

외부에서 확인 불가 하다.

```
public static <E extends Comparable<E>> E max(Collection<E> c) {  
    if(c.isEmpty()) {  
        throw new IllegalArgumentException("빈 컬렉션");  
    }  
  
    E result = null;  
  
    for (E e : c) {  
        if(result == null || e.compareTo(result) > 0) {  
            result = Objects.requireNonNull(e);  
        }  
    }  
  
    return result;  
}
```

# Item 55 옵셔널 반환은 신중히 하라

외부에게 null값을 가질 수 있음을 알린다.

```
public static <E extends Comparable<E>> Optional<E> max2(Collection<E> c) {  
    if(c.isEmpty()) {  
        return Optional.empty();  
    }  
  
    E result = null;  
    for (E e : c) {  
        if(result == null || e.compareTo(result) > 0) {  
            result = Objects.requireNonNull(e);  
        }  
    }  
  
    return Optional.of(result);  
}
```



# Item 55 옵셔널 반환은 신중히 하라

스트림에서 Optional

종단 연산은 Optional을 반환



```
Optional<T> max(Comparator<? super T> comparator);
```

가독성이 높은 예외 처리



```
Optional lastWordInLexicon = max(words).orElse("단어가 없습니다")
```

# Item 55 옵셔널 반환은 신중히 하라

orElseGet으로 supplier의 Lazy Evaluation을 이용

```
public T orElseGet(Supplier<? extends T> supplier) {  
    return value != null ? value : supplier.get();  
}
```

컬렉션, 스트림, 배열, Optional(컨테이너 타입)은 Optional 타입 참조로 사용 X

박싱된 기본 타입을 담은 Optional을 사용X(OptionalInt, OptionalLong)

잘만 사용하

# Item 56 공개된 API 요소에는 항상 문서화 주석을 작성하라

메서드용 문서화 주석은 메서드와 클라이언트 사이의 규약을 명료하게 기술  
~~어떻게 동작~~ -> 무엇을 하는지

@param

@return

@throws

```
//import statements

/**
 * @author    someone@gmail.com
 * @version   1.0.0
 * @since     1.0.0
 */
public class Test {
    // class body
}
```

## Item 56 공개된 API 요소에는 항상 문서화 주석을 작성하라

한 클래스(or 인터페이스) 안에서 요약 설명이 똑같은 멤버(or 생성자)가 둘 이상이면 안 된다.

제네릭 타입이나 제네릭 메서드를 문서화할 때는 모든 타입 매개변수에 주석을 달아야 한다.

열거 타입을 문서화할 때는 상수들에도 주석을 달아야 한다.

애너테이션 타입을 문서화할 때는 멤버들에도 모두 주석을 달아야 한다.

스레드 안전 수준을 명세해야 한다.



