Effective Java 3/E

Item 49

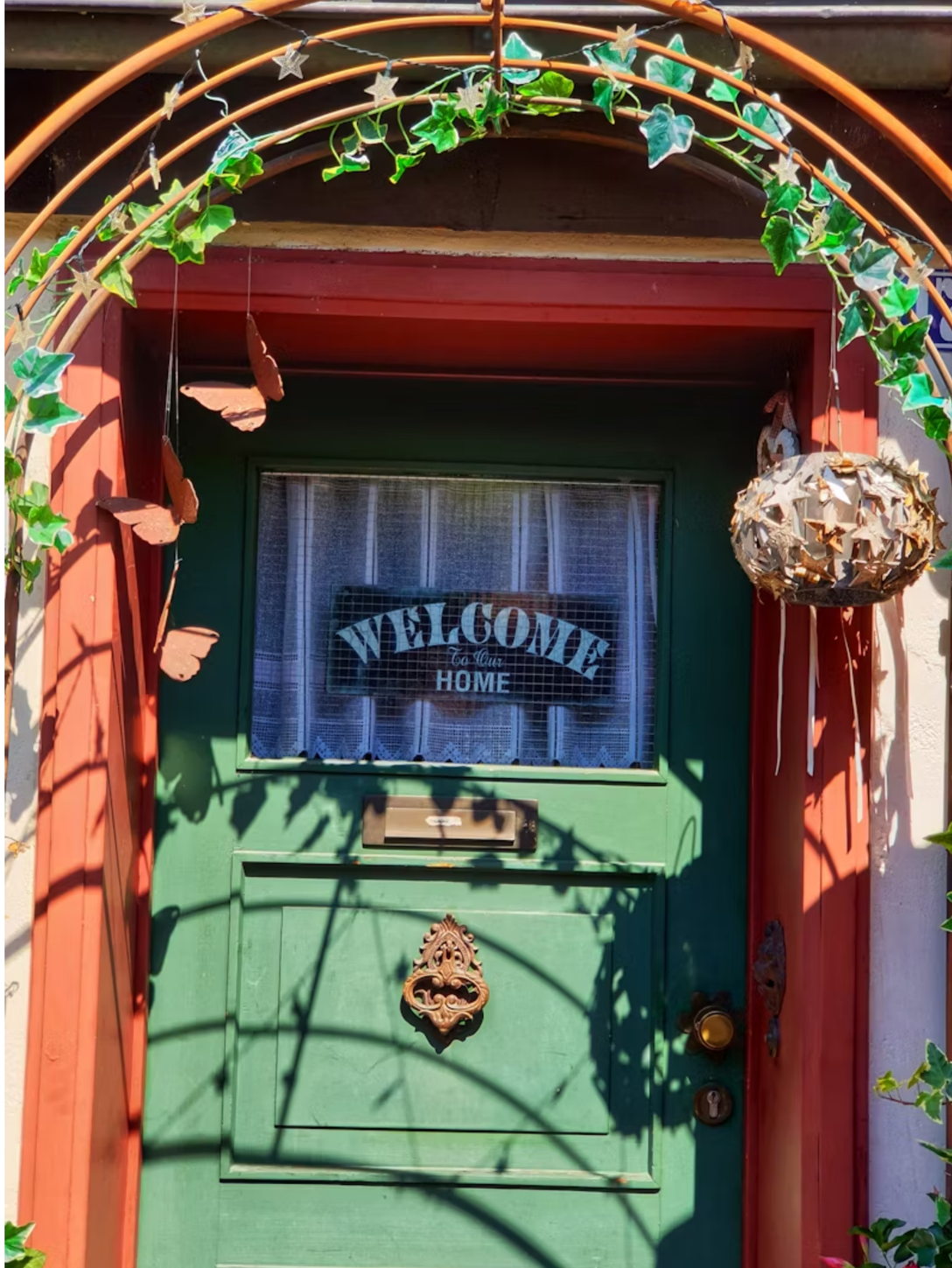
발표자 : 김형주

매개변수가 유효한지

검사하라.

Content

- 검사해야 하는 이유
- 검사하는 방법
- 예외
- 정리



Content

📌 매개변수 검사를 해야하는 이유

null

```
public List<String> getUserCategories(String email){

    UserDetail userDetail = userDetailRepository.findById(email)
        .orElseThrow(() -> new NoSuchElementException("추가 정보를 받지 않은 유저 입니다."));

    final Set<Interest> interests = userDetail.getInterests();

    return Stream.concat(Stream.concat(Stream.of("RECOMMEND"),interests.stream().map(Interest::name)),Stream.of(Category.values())
            .map(Category::name)
            .filter(categoryName -> interests.stream().map(Interest::name).noneMatch(categoryName::equals)))
            .toList();
  }
```

📌 매개변수 검사를 해야하는 이유

```java
public List<String> getUserCategories(String email) {

    UserDetail userDetail = userDetailRepository.findById(email)
        .orElseThrow(() -> new NoSuchElementException("추가 정보를 받지 않은 유저 입니다."));

    final Set<Interest> interests = userDetail.getInterests();

    return Stream.concat(Stream.concat(Stream.of("RECOMMEND"),interests.stream().map(Interest::name)),Stream.of(Category.values())
            .map(Category::name)
            .filter(categoryName -> interests.stream().map(Interest::name).noneMatch(categoryName::equals)))
            .toList();
  }
```

" 오류는 가능한 한 빨리 잡아야 한다. "

메서드가 수행되는 중간에 모호한 예외를 던지며 실패할 수 도 있다.

잘 수행되지만 잘못된 결과값을 반환할 수 도 있다.

잘 수행되었지만 그 과정에서 다른 객체의 상태에 영향을 주어

미래에 메서드와 관련 없는 오류를 낼 수 있다.

실패 원자성을 어기게 된다.

:호출된 메서드가 실패하더라도 해당 객체는 메서드 호출 전 상태를 유지해야 한다

Content

📌 방법 1 : 조건문

```java
public String method(String email) throws IllegalAccessException {

    if(email == null){
        throw new IllegalAccessException("email이 null 입니다.");
    }

    ...
}
```

```
/Users/khj/Library/Java/JavaVirtualMachines/graalvm-jdk-21.0.3/Contents/Home/bin/java -javaagent:/
Exception in thread "main" java.lang.IllegalAccessException Create breakpoint : email이 null 입니다.
```

📌 방법 2 : @NonNull

```
public String method(@NonNull String email){

    ...

}
```

📌 방법 3 : requireNonNull

```java
public String method(String email){

  String parameter = Objects.requireNonNull(email,"email이 null 입니다.");

  ...
}
```

```
Exception in thread "main" java.lang.NullPointerException Create breakpoint : email이 null 입니다.
  at java.base/java.util.Objects.requireNonNull(Objects.java:259)
  at main.effective_java.stage_8.item_49.Test.method(Test.java:27)
  at main.effective_java.stage_8.item_49.Test.main(Test.java:18)
```

📌 추가 사항

```java
public static int checkFromIndexSize(int fromIndex, int size, int length) {
  return Preconditions.checkFromIndexSize(fromIndex, size, length, null);
}

public static int checkFromToIndex(int fromIndex, int toIndex, int length) {
  return Preconditions.checkFromToIndex(fromIndex, toIndex, length, null);
}

public static int checkIndex(int index, int length) {
  return Preconditions.checkIndex(index, length, null);
}
```

📌 추가 사항

```java
public class Test {

  public static void main(String[] args){
    List<Integer> list = List.of(1,2,3,4,5);
    System.out.println(test.indexValidator(list));
  }

  public boolean indexValidator(List<Integer> list){
    Objects.checkIndex(5,list.size());
    return true;
  }
}
```

```
> Exception in thread "main" java.lang.IndexOutOfBoundsException Create breakpoint : Index 5 out of bounds for length 5 <3 internal lines>
    at java.base/java.util.Objects.checkIndex(Objects.java:385)
    at main.effective_java.stage_8.item_49.Test.method(Test.java:35)
    at main.effective_java.stage_8.item_49.Test.main(Test.java:20)

  Process finished with exit code 1
```

📌 방법 4 : assert (단언문)

```java
public String method(String email){

    assert email != null : "Email should not be null";
    ...
}
```

```
Exception in thread "main" java.lang.AssertionError Create breakpoint : Email should not be null
  at main.effective_java.stage_8.item_49.Test.method(Test.java:26)
  at main.effective_java.stage_8.item_49.Test.main(Test.java:17)

Process finished with exit code 1
```

📌 assert 주의사항

Assertion Error 만 발생시킴

~~에러의 발생 원인에 따라 다른 조치를 취해야 하는 경우~~

assert 외의 다른 방법이 더 어울린다.

유저 입력과 같은 메서드 입력을 검사할때는 잘 사용되지 않는다.

테스트 환경와 같이 일시적인 디버깅에 주로 사용된다.

Content

📌 예외

1. 유효성 검사 비용이 지나치게 높은 경우

2. 유효성 검사가 실용적이지 않은 경우

📌 유효성 검사가 실용적이지 않은 경우

```
public void doReverseSort(List<String> list){

    Collections.sort(list, Collections.reverseOrder());
}
```

정렬가능한 객체가 들어있는지

검사해야 하지 않나?

📌 유효성 검사가 실용적이지 않은 경우

```
default void sort(Comparator<? super E> c) {
    Object[] a = this.toArray();

    Arrays.sort(a, (Comparator) c);

    ListIterator<E> i = this.listIterator();
    for (Object e : a) {
        i.next();
        i.set((E) e);
    }
}
```

ClassCastException

암묵적 유효성 검사가 존재하는 경우

+ 암묵적 유효성 검사에 너무 의존하는 것도 조심하자.

Content

실패의 원자성을 위해

매개변수 유효성 검사는 필수

반복문 / @NonNull

requireNonNull / assert

+ 추가적으로 @throws을 활용하여 문서화도 같이 하자.

Item 49