Effective Java Study

Woowacourse_study 4th

Item 76 by 알파

실패 원자성이란..?

호출된 메서드가 실패해도 해당 객체가 메서드 호출 전 상태를 유지하는 특성

굳이..?

만약 실패 원자성을 띄지 않는다면, 어떠한 상태를 가지는 지

알기 힘들다

Rollback을 위해 추가적인 작업이 필요하다

How to 1

메서드를 불변 객체로 설계한다

How to 1

불변 객체는 태생적으로

실패 원자적!

Validate 이후 객체를 생성하니까

```java
public class Boxer() {
    private final int height;
    private final int weight;
    private final int reach;
    private final List<Result> history;

    private Boxer(int height, int weight, int reach, List<Result> history) {
        this.height = height;
        this.weight = weight;
        this.history = List.copyOf(history);
    }

    public static Boxer of(int height, int weight, int reach, List<Result> history) {
        return new Boxer(height, weight, reach, history);
    }

    public List<Result> getHistory() {
        return Collection.unmodifiableList(this.history);
    }

    private void validateRange(int value) {
        if (value < 1 || value > 300) {
            throw new IllegalArgumentExceoption();
        }
    }
}
```

How to 1

불변 객체는 태생적으로

실패 원자적!

```
public class Boxer() {
    private final int height;
    private final int weight;
    private final int reach;
    private final List<Result> history;

    private Boxer(int height, int weight, int reach, List<Result> history) {
        this.height = height;
        this.weight = weight;
        this.history = List.copyOf(history);
    }

    public static Boxer of(int height, int weight, int reach, List<Result> history) {
        return new Boxer(height, weight, reach, history);
    }

    public List<Result> getHistory() {
        return Collection.unmodifiableList(this.history);
    }

    private void validateRange(int value) {
        if (value < 1 || value > 300) {
            throw new IllegalArgumentExceoption();
        }
    }
}
```

How to 1

그럼 가변 객체는?

How to 2

매개변수의 유효성을 검사한다

```
public Object pop() {
    if (size == 0) {
        throw new EmptyStackException();
    }
    Object result = elements[--size];
    elements[size] = null;
    return result;
}
```

How to 2

일반화한다면, 실패할만한 모든 코드를 객체의

상태가 바뀌기 전에 배치한다

How to 2

Ex) TreeMap

```
Associates the specified value with the specified key in this map. If the map previously contained a
mapping for the key, the old value is replaced.

Params: key – key with which the specified value is to be associated
        value – value to be associated with the specified key

Returns: the previous value associated with key, or null if there was no mapping for key. (A null
         return can also indicate that the map previously associated null with key.)

Throws: ClassCastException – if the specified key cannot be compared with the keys currently in the
        map
        NullPointerException – if the specified key is null and this map uses natural ordering, or its
        comparator does not permit null keys
535  public V put(K key, V value) {
536      Entry<K,V> t = root;
537      if (t == null) {
538          compare(key, key); // type (and possibly null) check
539
540          root = new Entry<>(key, value,  parent: null);
541          size = 1;
542          modCount++;
543          return null;
544      }
545      int cmp;
```

How to 2

인터페이스 A 타입으로

묶이는 B와 C!

```java
public class B implements A {
    private int value;

    @Override
    public void someAbstract() {
        System.out.println("Override Done");
    }

    public B(int value) {
        this.value = value;
    }
}

public class C implements A {
    private int value;

    @Override
    public void someAbstract() {
        System.out.println("Override Done");
    }

    public C(int value) {
        this.value = value;
    }
}
```

How to 2

TreeMap<A, Integer>로 선언된 TreeMap에

B객체와 C객체를 넣으면..?

How to 2

TreeMap의 put에서 알려준 ClassCastException

```
Exception in thread "main" java.lang.ClassCastException Create breakpoint : com.company.B cannot be cast to java.lang.Comparable
    at java.util.TreeMap.compare(TreeMap.java:1294)
    at java.util.TreeMap.put(TreeMap.java:538)
    at com.company.MyStack.main(MyStack.java:14)
```

How to 2

디버깅의 세계로..

```
public V put(K key, V value) {   key: B@506    value: 1
    Entry<K,V> t = root;   t (slot_3): null
    if (t == null) {   t (slot_3): null
        compare(key, key); // type (and possibly null) check   key: B@506


        root = new Entry<>(key, value,  parent: null);
        size = 1;
        modCount++;
        return null;
    }
```

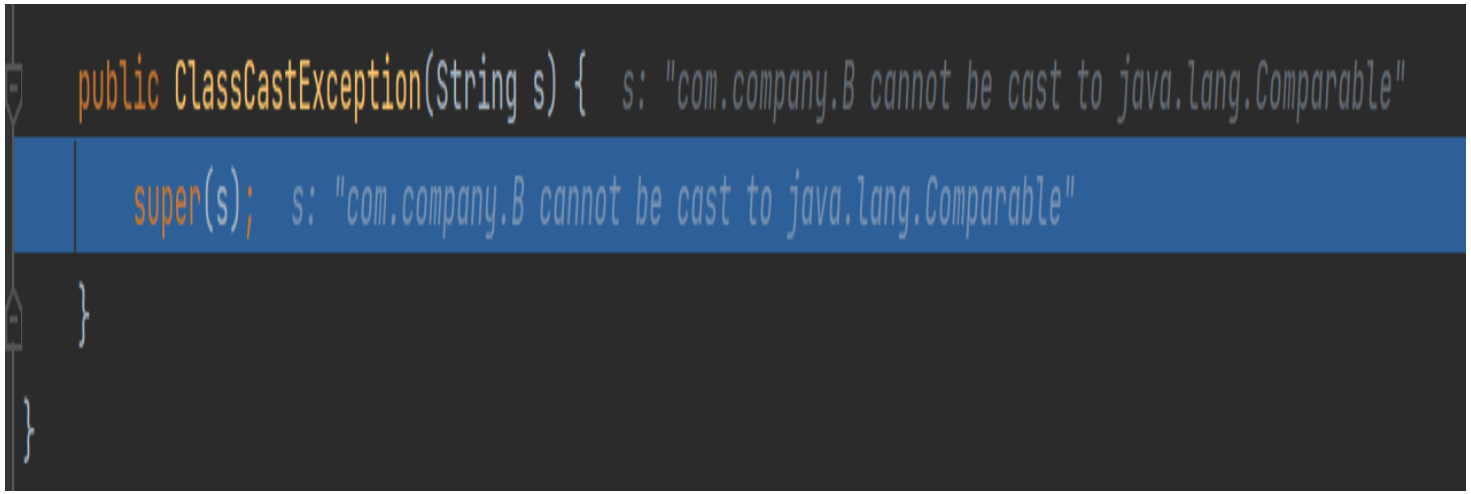
How to 2

디버깅의 세계로..

```
    Compares two keys using the correct comparison method for this TreeMap.
    /unchecked/
    final int compare(Object k1, Object k2) {   k1: B@506    k2: B@506
        return comparator==null ? ((Comparable<? super K>)k1).compareTo((K)k2)   k1: B@506    k2: B@506
            : comparator.compare((K)k1, (K)k2);
    }
```

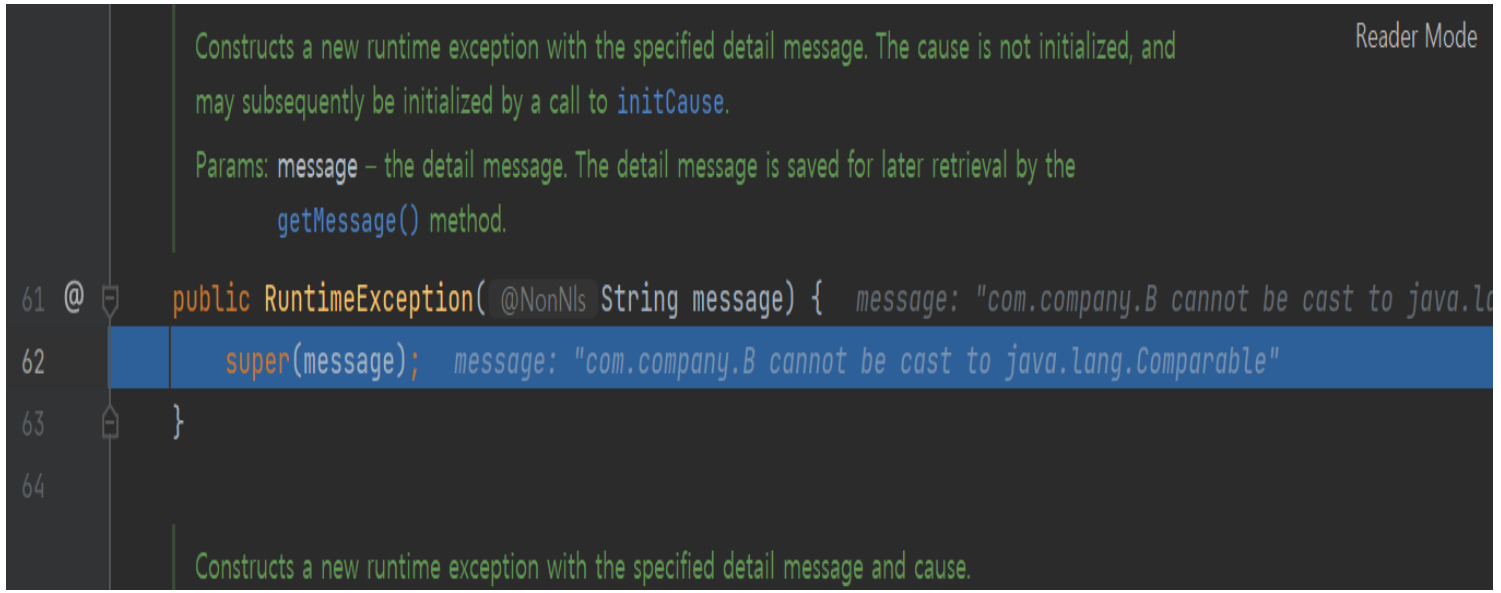
How to 2

디버깅의 세계로..



How to 2

디버깅의 세계로..

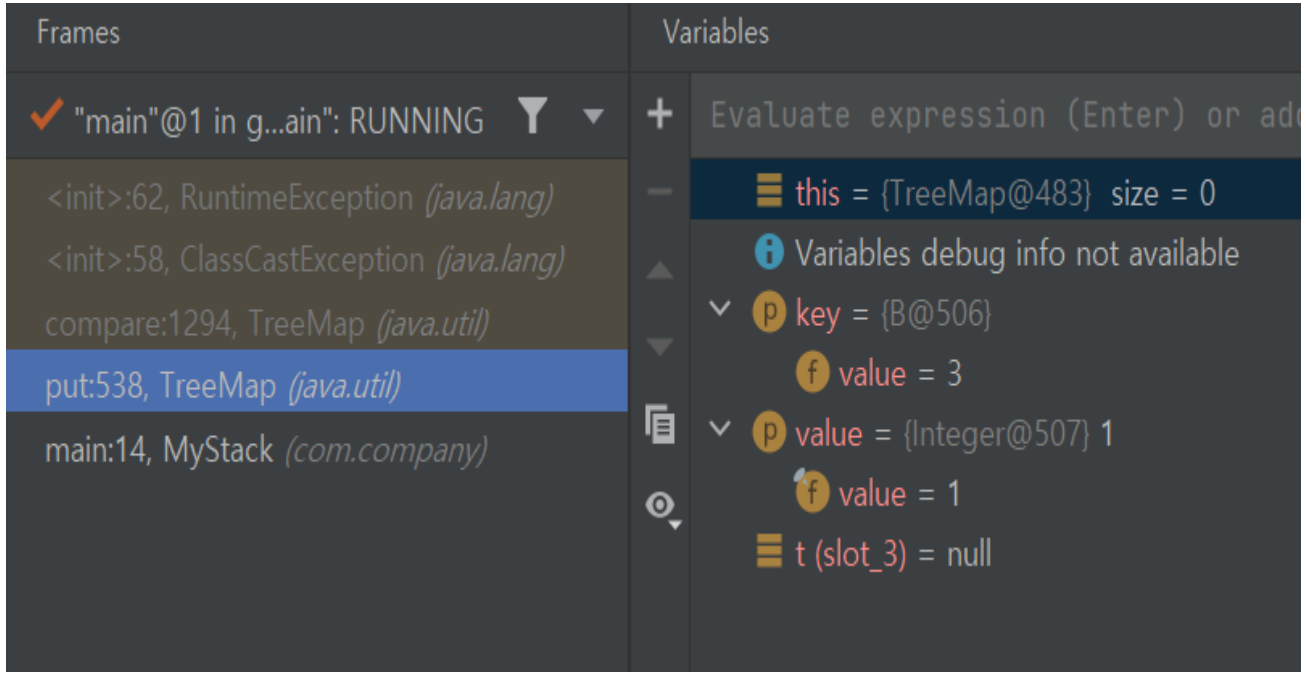


How to 2

에러가 발생하나,

TreeMap에 변화는 없다

= 실패원자성!



How to 3

로직을 실행하기 전,

복사본에 로직을 수행 후,

성공적으로 수행이 완료되면

원래의 객체와 swap

How to 3

Ex) 정렬 메서드

How to 3

디버깅의 세계로..

```
    @Override
    /unchecked/
    public void sort(Comparator<? super E> c) {   c: null
        final int expectedModCount = modCount;   expectedModCount (slot_2): 4
        Arrays.sort((E[]) elementData, fromIndex: 0, size, c);   c: null
        if (modCount != expectedModCount) {
            throw new ConcurrentModificationException();
        }
        modCount++;
    }
}
```

How to 3

디버깅의 세계로..

```
@Contract(mutates = "param1")
public static <T> void sort( @NotNull T[] a,  @Range(from = 0, to = java.lang.Integer.MAX_VALUE) int fromIndex,
                             @Nullable Comparator<? super T> c) {   c: null
    if (c == null) {   c: null
        sort(a, fromIndex, toIndex);
    } else {
        rangeCheck(a.length, fromIndex, toIndex);
        if (LegacyMergeSort.userRequested)
            legacyMergeSort(a, fromIndex, toIndex, c);
        else
            TimSort.sort(a, fromIndex, toIndex, c,  work: null,  workBase: 0,  workLen: 0);
    }
}
```

How to 3

디버깅의 세계로..

```
    To be removed in a future release.
private static <T> void legacyMergeSort(T[] a, int fromIndex, int toIndex,
                                        Comparator<? super T> c) {
    T[] aux = copyOfRange(a, fromIndex, toIndex);
    if (c==null)
        mergeSort(aux, a, fromIndex, toIndex, -fromIndex);
    else
        mergeSort(aux, a, fromIndex, toIndex, -fromIndex, c);
}
```

How to 4

작업 도중의 에러를 가로채는

복구 코드를 작성하여 Rollback

그러나..

멀티 스레드 환경의 경우

동기화 없이 같은 객체를 수정하는 경우

ConcurrentModificationException을 잡아도

해당 객체가 쓸 수 있다고 생각하면 안된다

그러나..

항상 실패 원자성을 갖게 하는 것이 권장되나,

실패 원자성을 갖기 위해 너무나 많은 복잡도와 비용이 추가된다면

할 필요는 없다

결론

메서드 명세에서 기술한 예외라면, 예외가 발생해도

발생하기 전의 객체와 동등한 상황이어야 한다

결론

그러나, 실패 원자성을 담보할 수 없다면

예외 이후 객체의 상황을 API 설명에 명시해야 한다

References

Joshua Bloch, 『Effective Java 3/E』, 이복연 역 (서울 : 인사이트, 2018), pp. 407 – 409

E.O.D

Item 76 by 알파