
이펙티브 자바

item73 ~ item75

24/07/09

Item 73

추상화 수준에 맞는 예외를 던지라

아이템 73 추상화 수준에 맞는 예외를 던지라

깔끔하지 않은 예외 처리 = 저수준 예외를 바깥으로 전파할 때

아이템 73 추상화 수준에 맞는 예외를 던지라

깔끔하지 않은 예외 처리 = 저수준 예외를 바깥으로 전파할 때

이러한 상황을 발생시키지 않으려면,
상위 계층에서는 저수준 예외를 잡아 자신의 추상화 수준에 맞는 예외로 바꿔서 던진다.

그러나 만약 저수준 예외가 디버깅 시에 도움을 줄 수 있다면,
예외 연쇄를 사용한다. (불투명 에러 처리)?

아이템 73 추상화 수준에 맞는 예외를 던지라 (불투명 에러 처리)

타입이 아니라 행위(behavior)에 대해 error assertion 을 하자.

```
type temporary interface {  
    Temporary() bool  
}  
  
// IsTemporary returns true if err is temporary.  
func IsTemporary(err error) bool {  
    te, ok := err.(temporary)  
    return ok && te.Temporary()  
}
```

아이템 73 추상화 수준에 맞는 예외를 던지라 (불투명 에러 처리)

```
func findAndSaveNameServerInfo(ns []string, outputDB *sql.DB, searchId int) error {
    var nameserverIPs []string
    var err error

    for _, nameserver := range ns {
        IPs, err := getIPAddresses(nameserver)
        if err != nil {
            // getIPAddresses에서 발생한 err에 네임서버 정보 context가 추가되었다.
            errors.Wrap(err, "\n--->Error for Nameservers : "+nameserver+err.Error()+"\n")
            continue
        }
        .
        .
    }
}

func main(){
    .
    .
    err = findAndSaveNameServerInfo(ns, outputDB, url.URLId)
    if err != nil {
        logger.Warn("Problem with find and save NS : " + err.Error())
    }
    .
    .
}
```

```
func getIPAddresses(url string) ([]string, error) {
    // Using LookupHost to lookup IP addresses of domain
    ips, err := net.LookupHost(url)
    if err != nil {
        //LookupHost 에서 발생한 err에 "LookupHost failed"라는 context가 추가되었다.
        return nil, errors.Wrap(err, "LookupHost failed : ")
    }

    err := invalidUrl(url)
    if err != nil {
        //invalidUrl 에서 발생한 err에 "Invalid Url"라는 context가 추가되었다.
        return nil, errors.Wrap(err, "Invalid Url : ")
    }

    return ips, nil
}
```

아이템 73 추상화 수준에 맞는 예외를 던지라

예외 연쇄

```
try{
    ... // 저수준 추상화를 이용한다.
}catch(LowerLevelException cause){
    // 저수준 예외를 고수준 예외에 실어 보낸다.
    throw new HigherLevelException(cause);
}
```

예외 연쇄용 생성자

```
class HigherLevelException extends Exception{
    HigherLevelException(Throwable cause){
        super(cause);
    }
}
```

아이템 73 추상화 수준에 맞는 예외를 던지라



저수준 메서드의 예외를 피할 수 없다면, 예외 연쇄 대신 상위 계층에서 해당 예외를 스스로 처리하고 API 호출자는 모르게 하는 방법도 존재.

예외 연쇄

```
try{
    ... // 저수준 추상화를 이용한다.
}catch(LowerLevelException cause){
    // 저수준 예외를 고수준 예외에 실어 보낸다.
    throw new HigherLevelException(cause);
}
```

예외 연쇄용 생성자

```
class HigherLevelException extends Exception{
    HigherLevelException(Throwable cause){
        super(cause);
    }
}
```


아이템 73 추상화 수준에 맞는 예외를 던지라

NOTE 결론

하지만 저수준 메서드의 예외를 상위 계층에서 처리하기 어렵고, 저수준 메서드에서 부득이하게 예외가 발생하는 경우 예외 연쇄를 통해 고수준 예외를 던지고, 근본적인 원인 또한 알리자.

Item 74

메서드가 던지는 모든 예외를 문서화하라

아이템 74 메서드가 던지는 모든 예외를 문서화하라

메서드가 던지는 예외는 해당 메서드의 사용법을 아는 데에 중요한 저몹이다.

검사 예외는 항상 따로 선언하고, 예외가 발생하는 각 상황을 자바독의 `@throws` 태그를 사용해 정확히 문서화하자.

물론 비검사 예외 (프로그래밍 오류)도 문서화를 해두면 좋다.

public 메서드에서 전제 조건을 문서화하는 것이 좋은데, 그 수단으로 가장 좋은 것이 비검사 예외들을 문서화하는 것이다.



```
/**
 * Constructs a newly allocated {@code Integer} object that
 * represents the {@code int} value indicated by the
 * {@code String} parameter. The string is converted to an
 * {@code int} value in exactly the manner used by the
 * {@code parseInt} method for radix 10.
 *
 * @param s the {@code String} to be converted to an {@code Integer}.
 * @throws NumberFormatException if the {@code String} does not
 *         contain a parsable integer.
 *
 * @deprecated
 * It is rarely appropriate to use this constructor.
 * Use {@link #parseInt(String)} to convert a string to a
 * {@code int} primitive, or use {@link #valueOf(String)}
 * to convert a string to an {@code Integer} object.
 */
@Deprecated(since="9")
public Integer(String s) throws NumberFormatException {
    this.value = parseInt(s, 10);
}
```

아이템 74 메서드가 던지는 모든 예외를 문서화하라

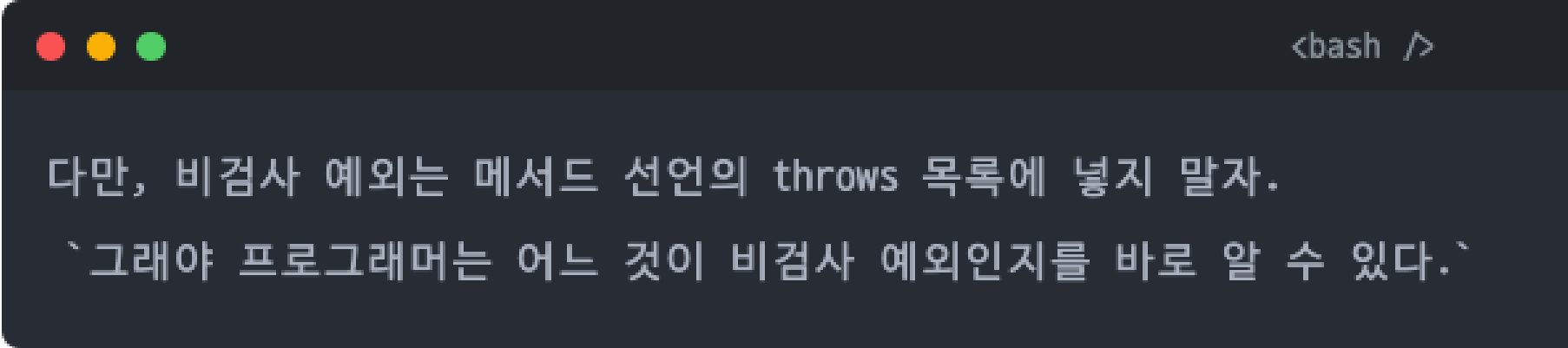
발생 가능한 비검사 예외를 문서로 남기는 일은 인터페이스 메서드에서 특히 중요하다.

이 조건이 인터페이스의 일반 규약에 속하게되어 그 인터페이스를 구현한 모든 구현체가 일관되게 동작하도록 도와준다.

아이템 74 메서드가 던지는 모든 예외를 문서화하라

발생 가능한 비검사 예외를 문서로 남기는 일은 인터페이스 메서드에서 특히 중요하다.

이 조건이 인터페이스의 일반 규약에 속하게되어 그 인터페이스를 구현한 모든 구현체가 일관되게 동작하도록 도와준다.



```
<bash />  
  
다만, 비검사 예외는 메서드 선언의 throws 목록에 넣지 말자.  
`그래야 프로그래머는 어느 것이 비검사 예외인지를 바로 알 수 있다.`
```

아이템 74 메서드가 던지는 모든 예외를 문서화하라

 **NOTE** 결론

메서드가 던질 가능성이 있는 모든 예외를 문서화하지만, 비검사 예외는 throws문에 기입하지 말자.

Item 75

예외의 상세 메시지에 실패 관련 정보를 담으라

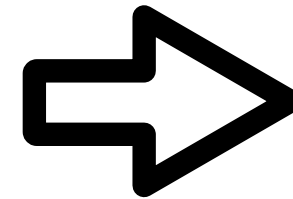
아이템 75 예외의 상세 메시지에 실패 관련 정보를 담으라

IndexOutOfBoundsException.java 예시

현재

```
<code />

/**
 * Constructs an {@code IndexOutOfBoundsException} with the specified detail
 * message.
 *
 * @param s the detail message
 */
public IndexOutOfBoundsException(String s) {
    super(s);
}
```



이렇게 한다면?

```
<code />

/**
 * IndexOutOfBoundsException을 생성한다.
 *
 * @param lowerBound 인덱스의 최솟값
 * @param upperBound 인덱스의 최댓값 + 1
 * @param index 인덱스의 실패값
 */
public IndexOutOfBoundsException(int lowerBound, int upperBound, int index) {
    // 실패를 포착하는 상세 메시지를 생성한다.
    super(String.format(
        "최솟값: %d, 최댓값: %d, 인덱스: %d",
        lowerBound, upperBound, index));

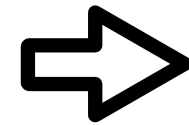
    // 프로그램에서 이용할 수 있도록 실패 정보를 저장해둔다.
    this.lowerBound = lowerBound;
    this.upperBound = upperBound;
    this.index = index;
}
```


아이템 75 예외의 상세 메시지에 실패 관련 정보를 담으라

IndexOutOfBoundsException.java 예시

현재

```
<code />
/**
 * Constructs an {@code IndexOutOfBoundsException} with the specified detail
 * message.
 *
 * @param s the detail message
 */
public IndexOutOfBoundsException(String s) {
    super(s);
}
```



이렇게 한다면?

```
<code />
/**
 * IndexOutOfBoundsException을 생성한다.
 *
 * @param lowerBound 인덱스의 최솟값
 * @param upperBound 인덱스의 최댓값 + 1
 * @param index 인덱스의 실패값
 */
public IndexOutOfBoundsException(int lowerBound, int upperBound, int index) {
    // 실패를 포착하는 상세 메시지를 생성한다.
    super(String.format(
        "최솟값: %d, 최댓값: %d, 인덱스: %d",
        lowerBound, upperBound, index));

    // 프로그램에서 이용할 수 있도록 실패 정보를 저장해둔다.
    this.lowerBound = lowerBound;
    this.upperBound = upperBound;
    this.index = index;
}
```

실패 순간을 포착하려면 발생한 예외에 관여된 모든 매개변수와 필드의 값을 실패 메시지에 담아야한다.

예외가 발생한 원인은 제각각이므로, 예외의 상세 메시지를 통해 무엇을 고쳐야할지 분석하는데 도움을 받을 수 있다.