이펙티브 자바

Item 76 ~ 78

2024/07/16

발표자: 박지원

Item 76 가능한 한 실패 원자적으로 만들라

예외가 발생해도 메서드 호출 전 상태를 유지해야 한다.

실패 원자성

1. **불변 객체 사용**
2. **로직 실행전 유효성 검사**

    실패할 가능성이 있는 모든 코드를 상태를 바꾸기 전에 실행

Item 76 가능한 한 실패 원자적으로 만들라

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

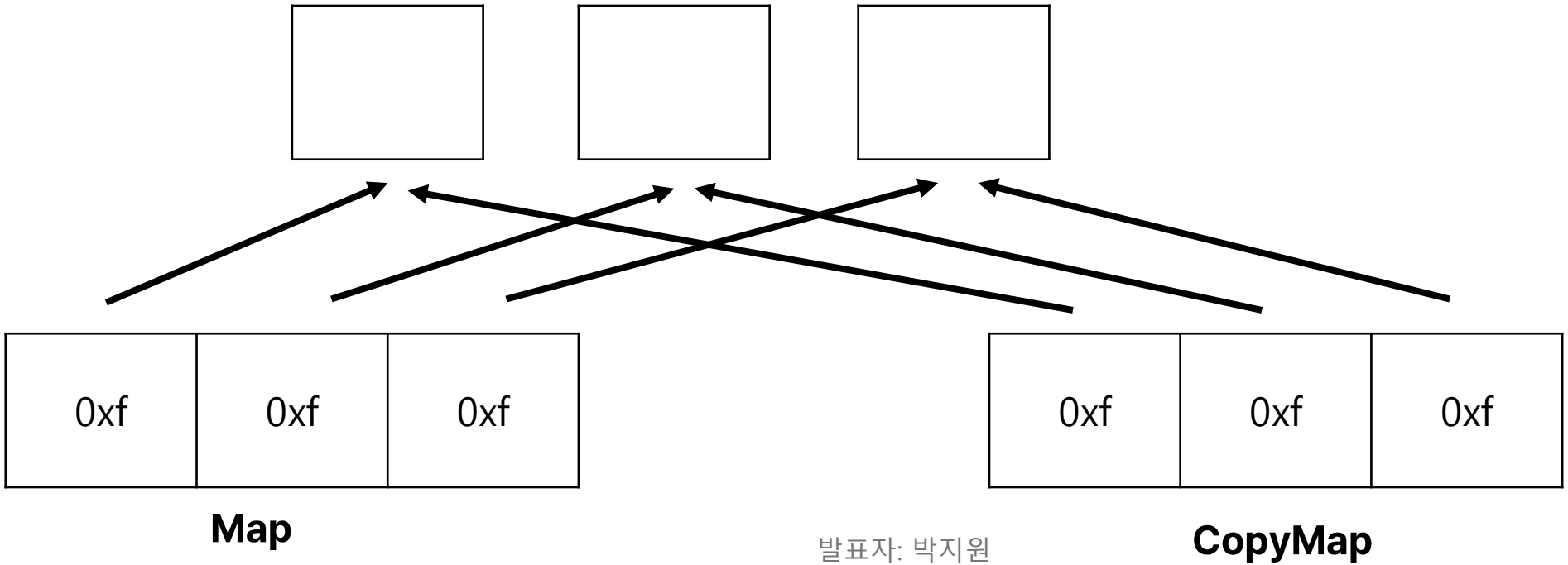
```
public synchronized E pop() {
    E       obj;
    int     len = size();

    obj = peek();
    removeElementAt(len - 1);

    return obj;
}
/* 중략
*/
public synchronized E peek() {
    int     len = size();

    if (len == 0)
        throw new EmptyStackException();
    return elementAt(len - 1);
}
```

발표자: 박지원

Item 76 가능한 한 실패 원자적으로 만들라

3. 임시 복사본 에서 작업을 수행후 원래 객체와 교체

Map을 전부 복사?



Item 76 가능한 한 실패 원자적으로 만들라

4. 실패를 가로채는 복구 코드를 작성하여 작업 전 상태로 되돌리기

동시성 상황에서 실패 원자성을 지키기 어렵다.

Error는 복구 할 수 없다.

발표자: 박지원

Item 77 예외를 무시하지 말라

Try-catch 예외를 무시하는 가장 쉬운 방법

```
try {
    ...
} catch (SomeException e) {
}
```

Item 77 예외를 무시하지 말라

FileInputStream의 closed 경우 무시될 수 있음

로그, 예외 변수 ignored

Item 77 예외를 무시하지 말라

스프링에서 실패 원자성

Http의 경우 stateless하므로 실패 원자성을 크게 생각해보지 않음

@transactional 에서의 roll back

Unchecked의 경우 rollback

Checked의 경우 rollback되지 않음(try catch, throws)

Item 77 예외를 무시하지 말라

https://www.youtube.com/watch?v=_WkMhytqoCc

자바 공부를 어떻게 하길래, "언체크드 예외 발생시 트랜잭션 롤백?"

조회수 3.8만회 • 2년 전

백기선

3분 45초에 "언체크익셉션 같은 경우..." 이건 "체크드 익셉션"으로 고쳐서 들어주세요. 인프런 "더 자바"

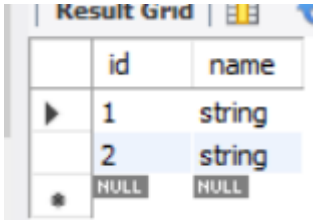
	Checked Exception	Unchecked Exception
처리여부	반드시 예외를 처리해야 함	명시적인 처리를 강제하지 않음
확인시점	컴파일 단계	실행단계
예외발생시 트랜잭션 처리	roll-back 하지 않음	roll-back 함
대표 예외	Exeption의 상속받는 하위 클래스 중 Runtime Exception을 제외한 모든 예외 • IOException • SQLException	RuntimeException 하위 예외 • NullPointerException • IllegalArgumentException • IndexOutOfBoundException • SystemException

발표자: 박지원

Item 77 예외를 무시하지 말라

checked 상황

```java
@Transactional
public TestEntity saveTestEntity(TestEntity testEntity) {
    TestEntity save = testRepository.save(testEntity);
    try{
        throw new IOException();
    }
    catch (Exception e){

    }
    return save;
}
```

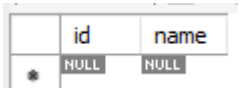



```java
@Transactional
public TestEntity saveTestEntity(TestEntity testEntity) throws IOException {
    TestEntity save = testRepository.save(testEntity);
    throw new IOException();
}
```

Item 77 예외를 무시하지 말라

Roll back 설정

```java
@Transactional(rollbackFor = Exception.class)
public TestEntity saveTestEntity(TestEntity testEntity) throws IOException {
    TestEntity save = testRepository.save(testEntity);
    throw new IOException();
}
```




```java
@Transactional(rollbackFor = IOException.class)
public TestEntity saveTestEntity(TestEntity testEntity) throws IOException {
    TestEntity save = testRepository.save(testEntity);
    throw new RuntimeException();
}
```

Item 77 예외를 무시하지 말라

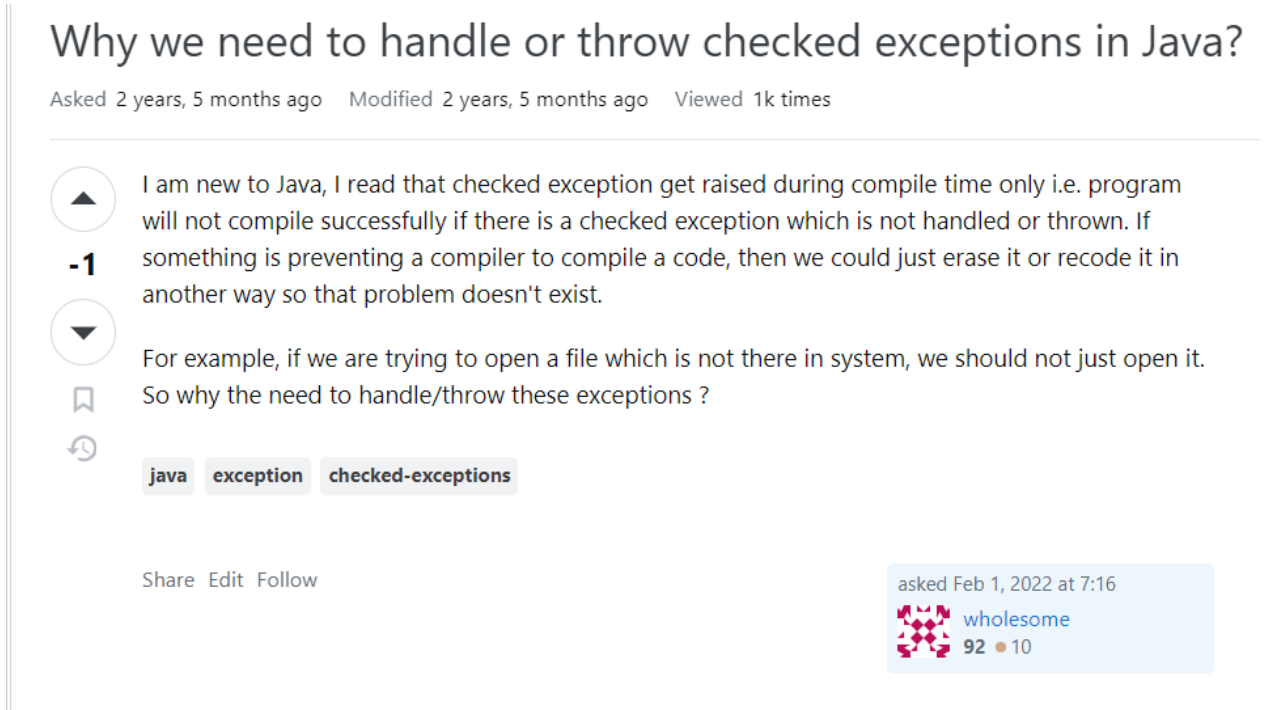
Transactional 기본설정으로 해도 문제 없어 보이지만..

생각해 볼 수 있는 문제 상황: SQLException

주의합시다..

Item 77 예외를 무시하지 말라

checked은 왜 에러 처리를 강제할까?



Why we need to handle or throw checked exceptions in Java?

Asked 2 years, 5 months ago Modified 2 years, 5 months ago Viewed 1k times

-1

I am new to Java, I read that checked exception get raised during compile time only i.e. program will not compile successfully if there is a checked exception which is not handled or thrown. If something is preventing a compiler to compile a code, then we could just erase it or recode it in another way so that problem doesn't exist.

For example, if we are trying to open a file which is not there in system, we should not just open it. So why the need to handle/throw these exceptions ?

java exception checked-exceptions

Share Edit Follow

asked Feb 1, 2022 at 7:16
wholesome
92 • 10

발표자: 박지원

Item 77 예외를 무시하지 말라

해당 메서드에 이러한 에러가 발생할 수 있음을 throws로 표시한다.

In a perfect world

Since the syntax of Java is such that if it is defined that a function throws an Exception, you must either handle the Exception or also make your function throw a Checked Exception using the `throws` keyword in function signature so that it is handled by JVM when thrown. But remember that Exception in Java is always handled, either its you that is handling it or the JVM(which shall crash the program & is generally a bad practice which is why we have checked Exception) – humble_barnacle Feb 1, 2022 at 7:24

In a perfect world where everyone only writes correct code, then we would not need exceptions, but we don't write perfect code so checked exceptions are a great way to easily pick up the most common and obvious errors that would quite likely crash your application at some point in time. Also, in response to "we should not just open it" what should your code do in a situation where your code depends on a file to function? – sorifiend Feb 1, 2022 at 7:24

Item 77 예외를 무시하지 말라

그렇게 하기로 약속했다

컴파일에 checked exception을 확인할 수 없는 경우도 있는데?

Just to reiterate. The compiler **cannot** check that the file exists because it doesn't know what pathname the user is **going to** provide. And even if it did know, AND it checked[1] that the file existed at compile, it **couldn't** know if the file was **going** to still exist at runtime, possibly on a completely different machine on a different network ... many years in the future.

발표자: 박지원

Item 77 예외를 무시하지 말라

예외 처리를 기능적으로 접근하면?

```
var file = new File("test.txt");
if (!file.exists()) {
    if(file.createNewFile()) {
        if (!file.isDirectory()) {
            if (!isUsed(file)) {
                var writer = new FileWriter(file);
                // ...
            }
        }
    }
}
```

```
var file = new File("test.txt");
try {
    var writer = new Filewriter(file);
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
```

발표자: 박지원

Item 77 예외를 무시하지 말라

Checked exception은 컴파일러가 개발자에게 구현을 강제하는 상황

컴파일러가 예외를 발생시키고 끝내기에는 부과 처리가 필요한 상황

개발자에게 예외 발생 가능성을 인지시키기 위함도 있다.(throws)

발표자: 박지원

Item 78 공유 중인 가변 데이터는 동기화해 사용하라

동기화: 멀티 스레드 환경에서 하나의 메서드나 블록에 하나의 스레드만

접근하여 작업수행을 보장하는 것

락(lock)을 걸어 리소스의 접근 권한을 획득한다.

동기화를 하지 않는다면?

객체 상태가 일관되지 않는다. 다른 스레드의 변화를 확인 못할 수 있다.

Item 78 공유 중인 가변 데이터는 동기화해 사용하라

무한 루프

```
private static boolean stopRequested;

public static void main(String[] args) throws InterruptedException {

    Thread thread = new Thread(() -> {
        int i = 0;
        while (!stopRequested) {
            i++;
        }
    });

    thread.start();
    TimeUnit.SECONDS.sleep(1);
    stopRequested = true;
}
```

발표자: 박지원

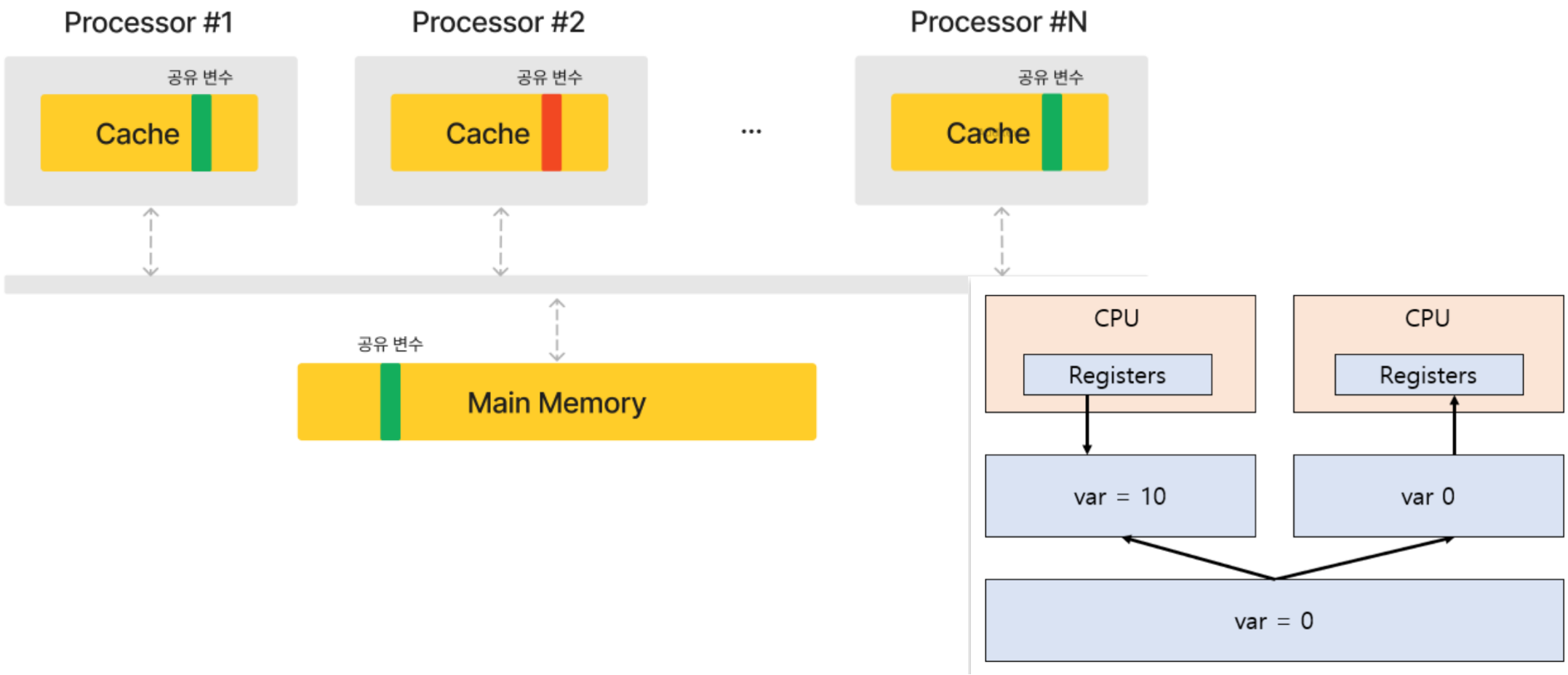
Item 78 공유 중인 가변 데이터는 동기화해 사용하라

```
counter = counter + 1;      100 mov 0x8049a1c, %eax
                            105 add $0x1, %eax
                            108 mov %eax, 0x8049a1c
```

			100	0	50
	mov		105	**50**	50
	add		108	**51**	50
Context switch					
			100	0	50
		mov	105	**50**	50
		add	108	**51**	50
		mov	113	51	**51**
Context switch					
			108	51	**51**
	mov		113	51	**51**

발표자: 박지원

Item 78 공유 중인 가변 데이터는 동기화해 사용하라



발표자: 박지원

Item 78 공유 중인 가변 데이터는 동기화해 사용하라

동기화: 배타적 수행 + 스레드 간 통신

1. Synchronized
2. Volatile
3. AtomicLong

```
private static synchronized void requestStop() {
    stopRequested = true;
}

private static synchronized boolean stopRequested() {
    return stopRequested;
}
```

Item 78 공유 중인 가변 데이터는 동기화해 사용하라

```
Thread incrementer1 = new Thread(() -> {
    for (int i = 0; i < 100000; i++) {
        counter++; // Not atomic
    }
});

Thread incrementer2 = new Thread(() -> {
    for (int i = 0; i < 100000; i++) {
        counter++; // Not atomic
    }
});

incrementer1.start();
incrementer2.start();
```

```
Counter: 143929
```

발표자: 박지원

Item 78 공유 중인 가변 데이터는 동기화해 사용하라

그이전에 가변 데이터 변경은 단일 스레드에서만 진행합시다

발표자: 박지원