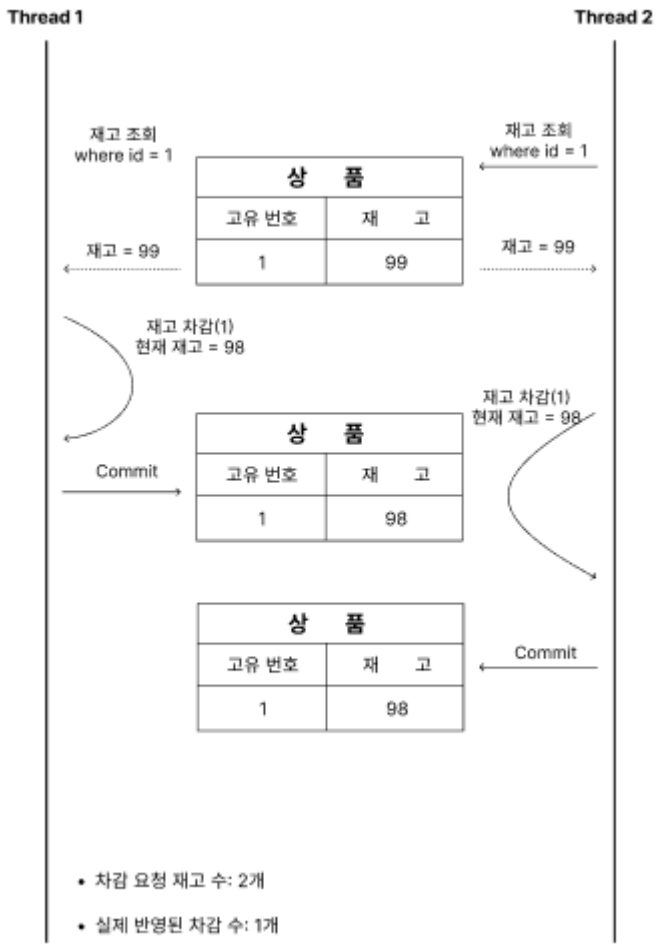
상품의 재고 차감(Stock) 상세

문제 식별

동일한 상품에 대한 주문이 동시에 반복적으로 수행될 경우 재고 차감이 정상적으로 이루어지지 않은 문제



- 실제 재고 차감에 대한 요청은 총 2건 (예상 기대값 99 - 2 개)
- 최종 재고 상태: 98개

분석

상품의 재고에 대하여 다양한 문제 상황이 발생할 수 있습니다.

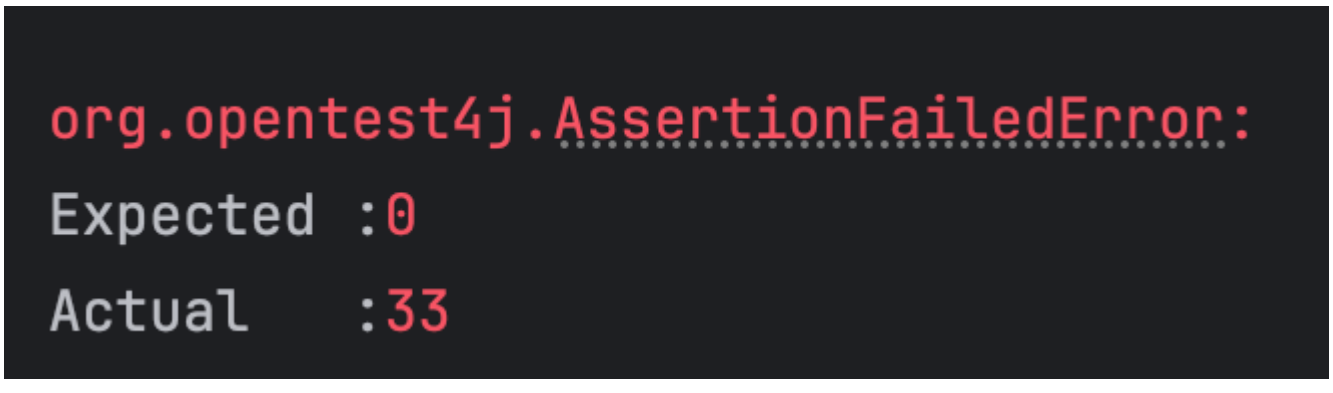
- 한가지 상품의 재고 차감 요청이 동시에 수행되어 트랙잭션이 유실
- 한가지 상품의 재고 조회와 재고 수정 작업이 동시에 진행되어 조회 결과가 달라짐
- 하나의 트랜잭션에서 재고의 조회가 2번 이상 발생하였을 경우 동시에 진행되는 재고 변경 트랜잭션으로 인해 값이 달라지는 문제

아래는 **재고에 대한 차감 요청이 동시에 발생**할 경우를 테스트한 코드입니다.

```
@Test
void 상품_재고_차감_동시성_테스트() throws InterruptedException {
    ItemCommand.Deduction command = new ItemCommand.Deduction(testItem.getId(), 1);

    List<Runnable> tasks = List.of(
            () -> itemService.deductStock(command)
    );

        //when
    ConcurrentTestExecutor.execute(50, 50, tasks);
    itemRepository.flush();
    Item updatedItem = itemRepository.findById(testItem.getId())
            .orElseThrow();

        //then
```

```
    assertEquals(0, updatedItem.stock());
}
```

- **실행결과**



해결

동일한 자원에 대한 접근이 빈번하게 발생되며 요청의 종류 또한 수정/조회가 다양하게 발생하는 도메인으로 판단되어 비관적 락 (**@Pessimistic Lock**) 적용

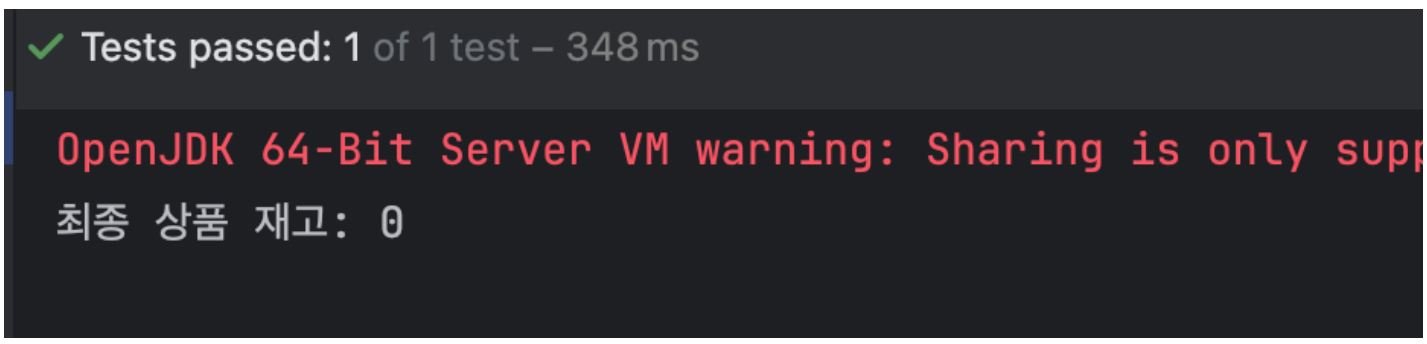
- ItemJpaRepository.java

```
public interface ItemJpaRepository extends JpaRepository<Item, Long> {

    @Lock(LockModeType.PESSIMISTIC_WRITE)
    @Query("select i from Item i where i.id = :id")
    Optional<Item> findByIdWithPessimisticLock(@Param("id") Long id);

}
```

- 실제 update query

```
Hibernate: update item set name=?,price=?,stock=? where item_id=?
```

- 실행 결과



대안

재고 처리 요청을 순차적으로 처리하는 방법의 도입을 고려해볼 수 있겠습니다.
예) Atomic Update, 메시지 큐 (MQ) 직렬화 처리