

# 제10장 응집성: 소프트웨어의 관련 요소들은 한곳에

## 10.1 모듈성과 응집성: 설계의 기초

- 응집성이 필요한 이유 : 관련 없는 것들을 더 멀리 떨어뜨릴 수 있는' 기술이 필요하지만, '관련 있는 것들을 더 가깝게 배치'해야 할 필요성도 심각하게 받아들일 필요가 있다.

## 10.2 응집성을 개선하기 위한 리팩토링 사례 하나

-

▼ 코드 10.1 순진한 응집성을 보여주는 정말 나쁜 코드

```
public class ReallyBadCohesion
{
    public boolean loadProcessAndStore() throws IOException
    {
        String[] words;
        List<String> sorted;

        try (FileReader reader =
            new FileReader("./resources/words.txt"))
        {
            char[] chars = new char[1024];
            reader.read(chars);
            words = new String(chars).split(" \\0");
        }
        sorted = Arrays.asList(words);
        sorted.sort(null);

        try (FileWriter writer =
            new FileWriter("./resources/test/sorted.txt"))
        {
            for (String word : sorted)
            {
                writer.write(word);
                writer.write("\\n");
            }
            return true;
        }
    }
}
```

▼ 코드 10.2 응집성이 약간 개선된 나쁜 코드

```
public class BadCohesion
{
    public boolean loadProcessAndStore() throws IOException
    {
        String[] words = readWords();
        List<String> sorted = sortWords(words);
        return storeWords(sorted);
    }

    private String[] readWords() throws IOException
    {
        try (FileReader reader =
            new FileReader("./resources/words.txt"))
        {
            char[] chars = new char[1024];
            reader.read(chars);
            return new String(chars).split(" |\\0");
        }
    }

    private List<String> sortWords(String[] words)
    {

```

196

```
        List<String> sorted = Arrays.asList(words);
        sorted.sort(null);
        return sorted;
    }

    private boolean storeWords(List<String> sorted) throws IOException
    {
        try (FileWriter writer =
            new FileWriter("./resources/test/sorted.txt"))
        {
            for (String word : sorted)
            {
                writer.write(word);
                writer.write("\n");
            }
            return true;
        }
    }
}
```

코드에서 밀접하게 관련된 부분이 더 명확하게 묘사되고 문자 그대로 서로 더 가깝게 연결되어 있어 응집성이 훨씬 더 높다

코드 10.2의 정보는 코드 10.1의 정보보다 응집성이 더 높다. 이제 코드가 기능적으로 동일하더라도 코드의 어느 부분이 서로 더 밀접하게 관련되어 있는지 훨씬 더 명확해졌다.

코드 10.2에는 코드 줄이 더 많다는 점에 주목하자

타이핑을 줄이기 위해 코드를 최적화하는 행위는 실수다. 우리는 잘못된 목표를 위해 최적화하고 있다. 코드는 의사소통 도구이므로 의사소통을 위해 사용해야 한다

**코드의 주요 목표는 아이디어를 사람에게 전달하는 것이다.**

우리는 가능한 한 명확하고 간결하게 아이디어를 표현하기 위해 코드를 작성하며, 적어도 그렇게 해야

한다. 결코 모호함을 희생해 간결함을 선택해서는 안 된다.

### 10.3 DDD의 컨텍스트를 활용한 응집성 개선

- 응집성은 인간의 복잡성을 관리하는 다른 도구와는 달리 컨텍스트에 따라 달라진다. 즉 '이 모든 것이 다른 것 과같지 않을 수 있다.'

의사결정을 내리는 데 효과적인 도구 중 하나는 도메인 주도 설계다. 문제 영역이 우리의 사고와 설계를 인도하게 하면 장기적으로 더 수익성이 높은 경로를 식별하는 데 도움이 된다.

#### 도메인 주도 설계(DDD, domain-driven design)

-본질적으로 관심 있는 비즈니스 영역의 시뮬레이션으로 코드의 핵심 동작을 포착하는 것을 목표로 하는 설계 접근 방식이다

또한 DDD는 '경계 컨텍스트(bounded context)'라는 개념도 도입했다 이는 공통 개념을 공유하는 시스템의 일부다

도메인 주도 설계는 더 나은 설계를 만들기 위한 강력한 도구이며, 설계 노력에 지침을 제공하고 코드의 모듈성, 응집성, 관심사 분리를 개선하는 데 도움을 주는 일련의 구조화 원칙을 제공한다

### 10.4 소프트웨어에서 '고성능'의 의미란

- 고성능 시스템에는 단순하고 잘 설계된 코드가 필요하다

최신 컴파일러는 최신 하드웨어에서 효율적으로 실행되도록 코드를 최적화하는 환상적인 작업을 수행한다. 코드가 단순하고 예측 가능할 때 최적화 기능이 뛰어나며, 코드가 복잡할수록 컴파일러의 최적화 기능을 통해 얻을 수 있는 도움은 줄어든다.

## 10.5 결합도와 응집성 사이의 관계

- 결합도coupling : 코드 A와 B가 주어졌을 때, A가 변경되었기 때문에 B가 동작을 변경해야 할 때 결합도가 생긴다.
- 응집성cohesion : A를 변경하면 B가 변경되어 둘 다 새로운 가치를 추가할 수 있을 때 응집성이 생긴다.

결합도는 어떤 의미에서 응집성의 비용이다. 응집성이 높은 시스템 영역에서는 더 긴밀하게 결합될 가능성이 높다

## 10.6 TDD 로 응집성을 높이자

- 코드를 작성하기 전에 테스트 케이스를 만들면, 코드에 대한 외부 API/인터페이스의 설계에 집중할 수 있다.

명세를 충족하는 데 필요한 사안보다 너무 많은 코드를 작성하면 개발 프로세스를 속이고 구현의 응집성을 떨어뜨리게 된다. 코드를 너무 적게 작성하면 동작 의도가 충족되지 않는다. TDD의 규율은 응집성을 위한 최적의 지점 에 도달하도록 장려 한다

## 10.7 응집성 있는 소프트웨어를 만들려면

- 응집성의 핵심 척도는 변경의 범위 또는 비용이다. 코드 기반을 변경하기 위해 여러 곳을 돌아다니며 변경해야 한다면 응집성 있는 시스템이라고 할 수 없다. 응집성은 기능적인 관련성을 측정하는 척도다. 응집성은 목적의 관련성을 측정하는 척도다.

궁극적으로 이는 해결하려는 문제의 컨텍스트에 따른 설계 선택이며, 상황에 따라 달라진다.

## 10.8 응집성이 부족할 때 치러야 할 대가

- 응집성은 내가 말하는 '복잡성 관리 도구'에서 가장 직접적으로 정량화하기 어려운 측면이지만, 매우 중요한 요소다 응집성이 떨어지면, 코드와 시스템의 유연성이 떨어지고 테스트하거나 작업하기가 더 어려워진다는 문제가 생긴다

응집성이 떨어지는 상황을 발견하는 간단하고 주관적인 방법이 있다. 코드를 읽다가 '이 코드가 무슨 일을 하는지 모르겠다'고 생각한 적이 있다면 아마도 응집성이 좋지 않기 때문일 것이다.

## 10.9 개발 조직 관점에서 응집성의 중요성

-응집성은 정보 수준에서 작동하는 아이디어이므로 우리가 일하는 조직을 합리적으로 구조화하는 데도 중요하다. 가장 확실한 예는 팀 조직이다 팀의 정보와 기술에 응집성이 있다는 뜻이며, 팀이 결정을 내리고 진전을 이루기 위해 필요한 모든 것을 다 갖추고 있다는 뜻이기도 하다

### 정리:

응집성은 복잡성을 관리하기 위한 아이디어 목록에서 아마도 가장 쉬운 아이디어 일 것이다  
응집성은 모듈성의 반대 개념으로, 주로 모듈성과 함께 고려할 때 의미가 있다.