chapter 10: Classes

Class Should be SMALL

Single Responsibility Principle (SRP)

> class or module should have only one Reason to Change

- 클래스에 선언되어 있는 변수나 메소드가 바뀌는 경우가 여러 가지면 안되고, 단 하나이어야 한다.

```
public class SuperDashboard extends JFrame implements MetaDataUser
    public Component getLastFocusedComponent()
    public void setLastFocused(Component lastFocused)
    public int getMajorVersionNumber()
    public int getMinorVersionNumber()
    public int getBuildNumber()
}
```

version, Component 객체에 따라서 바뀜 → 하나 이상

```
public class Version {
    public int getMajorVersionNumber()
    public int getMinorVersionNumber()
    public int getBuildNumber()
}
```

version에만 responsibility 가짐!

- 거대한 클래스 조금 vs **작은 클래스 많이**
  - single responsibility를 지키려면 클래스를 많이 쪼개는 상황이 오는데, 이게 오히려 가독성을 떨어트리는게 아닌가 하는 생각이 들 수 있다
  - 그러나 공구함에 전부 던져 놓은 것보다 수납장에 잘 정리해 놓은 것이 찾기 쉽듯이, 클래스를 잘 organize하게 되면 개발자 입장에서 필요한 부분의 코드를 빠르게 찾을 수 있으며 적은 수정만 해도 되도록 한다.

Cohesion (응집도)

- 클래스 변수들을 메소드에서 얼마나 사용하는가의 척도이다.
  - 응집도가 높다 → 모든 메소드에서 모든 변수를 사용한다
  - 응집도가 낮다 → 일부 메소드에서 사용하지 않는 변수가 존재

- 만약 응집도가 낮은데 일부 메소드들이 비슷한 변수를 사용한다면?
  클래스를 새로 하나 만들어서 메소드들을 묶고 응집도를 높일 수 있다.

- 거대한 함수를 쪼개는 방법

    - 작은 부분을 새로운 함수로 만든다
        - 이때 함수 인자로 넘겨주지 말고, 클래스 변수에 선언해 놔서 파라미터가 필요 없게 한다

```
class...
    func1()
        func2(int x)...
        ...
```

```
class...
    private int x;
    func1()...
        func2()
```

    - 응집도가 낮아졌다면 새로운 클래스로 쪼갠다.
    - 따라서 큰 함수를 쪼개면서 클래스가 여러개 나오는 것은 자연스럽다.
- 쪼개더라도 알고리즘이나 문제 해결 과정이 바뀌어서는 안된다. 단지 코드의 형식을 다르게 했을 뿐이다 (this is Not a rewrite)

Organizing for Change

클래스에 새로운 기능이 계속 추가되어야 하는 상황을 가정하자.

- 만약 클래스 안에 메소드를 추가 하는 식으로 한다면
    - 클래스는 점점 거대해지며
    - responsibility는 늘어나고
    - cohesion은 줄어들고
    - 예상치 못한 충돌도 많아진다.
- 따라서 abstract class를 활용해서 확장하는 식으로 만들면 좋다.
  → class should be
  open for extension but **closed for modification**