아이템17 (변경 가능성을 최소화하라)

불변 클래스란 인스턴스의 내부 값을 수정할 수 없는 클래스이다. 불변 인스턴스에 간직된 정보는 고정되어 객체가 파괴되는 순간까지 절대 달라지지 않는다. 자바에도 다양한 불변 클래스가 존재한다. Ex) String, BigInteger

String 불변클래스 검증

```
public class Main {

    public static void main(String[] args) {
        String a= new String( original: "hi");
        System.out.println(System.identityHashCode(a));
        a = "bye";
        System.out.println(System.identityHashCode(a));
    }
}
```

객체의 해쉬코드 출력 코드

```
"C:\Program Files\Java\jdk-11.0.11\bin\java.exe"
2083562754
1239731077

Process finished with exit code 0
```

결과

위와 같이 String 객체에 다른 문자를 재할당을 하게 되면 객체의 값이 바뀌는 것이 아니라 새로운 객체를 생성해 할당해준다.

불변 클래스는 가변 클래스보다 설계하고 구현하고 사용하기 쉬우며 오류가 생길 여지도 적고 안전하다. 불변 클래스를 만들기 위해선 다음과 같은 규칙을 지켜야한다.

불변 클래스 규칙

1. 객체의 상태를 변경하는 메서드를 제공하지 않는다.

불변 클래스는 객체의 상태가 변경돼서는 안된다.

2. 클래스를 확장할 수 없도록 한다.

하위 클래스에서 객체의 상태를 변하게 만드는 것을 막아준다. 대표적인 방법은 final선언이 있다.

3. 모든 필드를 final로 선언한다.

시스템이 강제하는 수단을 이용해 설계자의 의도를 명확히 드러내는 방법이다.

4. 모든 필드를 private으로 선언한다.

필드가 참조하는 가변객체를 클라이언트에서 직접 접근해 수정되는 것을 막아준다.

5. 자신 외에는 내부의 가변 컴포넌트에 접근할 수 없도록 한다.

클래스에 가변 객체를 참조하는 필드가 하나라도 있다면 클라이언트에서 그 객체의 참조를 얻을 수 없도록 해야 한다. 이런 필드는 절대 클라이언트가 제공한 객체 참조를 가리키게 해서는 안되며 접근자 메서드가 필드 그대로 반환해서도 안된다. 생성자, 접근자, readObject메서드 모두에서 방어적 복사를 수행해야한다.

불변 객체의 장점

1. 불변 객체는 단순하다.

불변 객체는 생성된 시점의 상태를 파괴될 때까지 그대로 간직하여 모든 생성자가 클래스 불변식을 보장한다면 영원히 불변으로 남는다. 반면 가변 객체는 임의의 복잡한 상태에 놓일 수 있으므로 믿고 사용하기 어려울 수 있다.

2. 불변 객체는 근본적으로 스레드 안전하여 따로 동기화할 필요가 없다.

여러 스레드가 동시에 사용해도 절대 훼손되지 않는다. 불변 객체에 대해서는 그 어떤 스레드도 다른 스레드에 영향을 줄 수 없으니 불변 객체는 안심하고 공유할 수 있다. 따라서 불변 클래스라면 한번 만든 인스턴스를 최대한 재활용하기를 권한다. 재활용을 통해 메모리 사용량과 가비지 컬렉션의 비용을 줄일 수 있다.

3. 불변 객체를 자유롭게 공유할 수 있으므로 점은 방어적 복사가 필요없다.

아무리 복사해봐야 원본과 똑같으니 복사가 의미가 없다. 그러니 불변 클래스는 clone을 제공하지 않는 것이 좋다.

4. 불변 객체끼리는 내부 데이터를 공유할 수 있다.

객체를 만들 때 다른 불변 객체들을 구성요소로 사용하면 이점이 많다. 구조가 아무리 복잡하더라도 불변식을 유지하기 수월하다. 불변 객체는 그 자체로 실패 원자성을 제공한다. 상태가 절대 변하지 않으니 잠깐이라도 불일치 상태에 빠질 가능성이 없다.

불변 클래스에도 단점

1. 값이 다르다면 반드시 독립된 객체로 만들어야한다.

2. 원하는 객체를 완성하기까지의 단계가 많고 그 중간 단계에서 만들어진 객체들이 모두 버려진다면 성능에 문제가 생긴다. 이 문제를 대처하는 방법은 두가지가 있다.

불변 클래스 성능 저하 대처법

1. 다단계 연산들을 예측하여 기본 기능으로 제공한다.

2. package-private의 가변 동반 클래스를 제공한다. Ex) String 클래스와 StringBuilder클래스

불변 클래스를 설계하는 대표적인 방법은 모든 생성자를 private 혹은 package-private으로 만들고 public 정적 팩터리를 제공하는 방법이 있다.

Package-private 구현 클래스를 원하는 만큼 만들어 활용할 수 있으나 패키지 바깥의 클라이언트에서 바라본 이 불변 객체는 사실상 final이다. public이나 protected생성자가 없어 다른 패키지에서는 이 클래스를 확장이 불가능하기 때문이다. 정적 팩터리 방식을 사용하여 다수의 구현 클래스를 활용한 유연성을 제공하고 객체 캐싱 기능을 추가해 성능을 끌어올릴 수도 있다.

불변 클래스 유의사항

1. 모든 필드가 final이고 어떤 메서드도 그 객체를 수정할 수 없어야 한다는 규칙은 과한 감이 있어 성능을 위해 어떤 메서드도 객체의 상태 중 외부에 비치는 값을 변경할 수 없다로 완화할 수 있다.

2. 성능 때문에 어쩔 수 없다면 불변 클래스와 쌍을 이루는 가변 동반 클래스를 public클래스로 제공하자 ex) String, StringBuilder

3. 생성자는 불변식 설정이 모두 완료된 초기화가 완벽히 끝난 상태의 객체를 생성해야 한다.

생성자와 정적 팩터리 외에는 초기화 메서드를 public으로 제공해서는 안된다. 객체를 재활용할 목적으로 상태를 다시 초기화 하는 메서드도 안된다. 복잡성만 커지고 성능 이점은 거의 없다.

정리

클래스는 꼭 필요한 경우가 아니라면 불변이어야 한다. 불변 클래스는 가변 클래스보다 설계하고 구현하고 사용하기 쉬우며 오류가 생길 여지도 적고 안전하다. 불변 클래스는 장점이 많으며 단점은 특정 상황에서의 잠재적 성능저하 뿐이므로 가변 클래스 대신 불변 클래스를 사용하자. 만약 가변 클래스를 사용하더라도 꼭 변경해야 할 필드를 뺀 나머지를 모두 final로 선언해 변경할 수 있는 부분을 최소화하여 예측하기 쉽고 오류가 생길 가능성을 줄이자.