아이템23 (태그 달린 클래스보다는 클래스 계층구조를 활용하라)

태그 달린 클래스란?

현재 표현하는 의미를 태그 값으로 알려주는 클래스로 여러가지 의미를 표현할 수 있다.

태그 달린 클래스 예시

```java
public class Figure {

    //태그 필드 - 현재 모양을 나타냄
    enum Shape{CIRCLE,RECTANGLE};
    final Shape shape;
    //사각형에 쓰이는 필드
    double length;
    double width;
    //원에 쓰이는 필드
    double radius;
    //사각형 전용 생성자
    public Figure(double length, double width) {
        this.shape = Shape.RECTANGLE;
        this.length = length;
        this.width = width;
    }
    //원 전용 생성자
    public Figure(double radius){
        this.shape = Shape.CIRCLE;
        this.radius = radius;
    }
    double area(){
        switch (shape){
            case CIRCLE:
                return length*width;
            case RECTANGLE:
                return Math.PI * radius * radius;
            default:
                throw new AssertionError();
        }
    }
}
```

- 원과 사각형을 표현한 클래스

태그 달린 클래스의 단점

1. 열거 타입 선언, 태그 필드, switch 문 등 쓸데없는 코드가 많다.
2. 여러 구현이 한 클래스에 혼합돼 있어 가독성이 나쁘다.
3. 다른 의미의 코드도 언제나 함께 사용돼 메모리도 많이 사용한다.
4. 필드들을 final로 선언하기 위해선 다른 필드도 초기화해야 한다.
5. 엉뚱한 필드를 초기화해도 런타임때 에러가 발생한다.
6. 또다른 의미를 추가하려면 코드를 수정해야 한다.
7. 인스턴스 타입만으로는 현재 나타내는 의미를 알 수 없다.

이처럼 태그 달린 클래스는 장황하고 오류를 내기 쉽고 비효율적이다. 따라서 자바와 같은 객체 지향 언어는 클래스 계층구조를 활용한 서브타이핑을 제공한다.

서브타이핑과 서브클래싱의 차이

서브클래싱 : 다른 클래스의 코드를 재사용할 목적으로 상속하는 경우로 구현 상속 또는 클래스 상속이라고도 부른다.

서브타이핑 : 타입 계층을 구성하기 위해 상속을 하는 경우로 인터페이스 상속이라고 부르기도 한다.

서브 타이핑은 인터페이스 혹은 추상클래스를 사용하여 구현하면 된다.

서브 타이핑 예시

```
abstract class Figure{
    abstract double area();
}

class Circle extends Figure{

    final double radius;
    Circle(double radius){ this.radius = radius;}
    @Override
    double area() {
        return Math.PI * radius * radius;
    }
}

class Rectangle extends Figure{
    final double length;
    final double width;

    Rectangle(double length, double width){
        this.length = length;
        this.width = width;
    }
    @Override
    double area() {
        return length*width;
    }
}
```

- 원과 사각형을 표현한 클래스

태그 값에 따라 동작이 달라지는 메서드를 추상 메서드로 선언하고 공통으로 사용하는 데이터 필드를 루트 클래스로 올린다. 다음으로 루트 클래스를 확장한 구체 클래스를 의미별로 하나씩 정의한다.

클래스 계층구조 장점

1. 태그 달린 클래스의 단점을 모두 없애 준다.

2. 추상 메서드를 모두 구현했는지 컴파일러가 확인해준다.

3. 루트 클래스의 코드를 건드리지 않고도 다른 프로그래머들이 독립적으로 계층구조를 확장하고 함께 사용할 수 있다.

4. 타입이 의미별로 따로 존재하여 변수의 의미를 명시하거나 제한할 수 있고 특정 의미만 매개변수로 받을 수 있다.

5. 유연성은 물론 컴파일 타임 타입검사 능력을 높여준다.

정리

태그 달린 클래스를 써야하는 상황은 거의 없다. 새로운 클래스를 작성하는데 필드가 등장한다면 태그를 없애고 계층구조로 대체하는 방법을 생각해보자. 기존 클래스가 태그 필드를 사용하고 있다면 계층 구조로 리팩터링하는걸 고민해보자.