발표자

배지훈

내용

둘러보자

코드의 이해와 리팩토링을 하기 전 가장 첫번째로 테스트가 얼마나 되는지 확인했다.

SerialDate 같은 경우 테스트 커버리지가 50% 정도 되었다.

해당 테스트를 통해서 커버리지가 얼마나되는지 확인했고 테스트를 하지 않거나 부실한 테스트를 추가했다.

그러면서 자신이 통과해야될 것 같은 경우도 같이 확인할 수 있었다.

고쳐보자

쓰잘데기 없는 주석을 지워버리기 (p.347)

라이센스 정보, 저작권 과 같은 경우 필요하다.

하지만 작성자, 변경 이력과 같은 경우는 소스 코드 관리 툴을 이용하게 되면 가볍게 무시할 수 있다.

- 변수에 대해서 역사와 의미를 남기는 주석은 좋은 주석이다. (p.350)

```
const int EARLIEST_DATE_ORDINAL = 2;       // 1/1/1900
const int LATEST_DATE_ORDINAL = 3344231;   // 12/31/9999
```

클래스와 함수의 기능을 명확하게

추상 클래스에 있는 변수가 오직 하나의 클래스에서만 사용한다면 그것은 옮기는 것이 좋을 것이다.

그것을 통해 해당 변수와 그 변수를 사용하는 거리가 좁아지고 실제 코드를 이해하는 도움이 될 것이다.

Naming

`INCLUDED_NONE`, `INCLUDE_FIRST`, `INCLUDE_SECOND`, `INCLUDE_BOTH` 이런 변수명이 있다.

해당 변수명은 포함되어 있는 범위를 나타내고 수학에서는 어느 특정 범위에 있는가에 대한 것과 일치한다.

해당 변수를 `CLOSED`, `CLOSED_LEFT`, `CLOSED_RIGTH`, `OPEN` 와 이름을 변경했을 때 조금 더 쉽게 이해된다는 생각이 든다.

Naming

```
public DayDate addDays(int days) {
    return factory.makeDate(toOrinal() + days);
}
```

위에 함수명을 보게 되면 `addDays(int days)` 는 새로운 DayDate객체를 반환해준다. 사용자 입장에서 해당 메소드를 입고 이 메소드를 사용하면서 어떤 생각을 할 것인가?

새로운 객체를 반환한다고 생각할 것인가? 아니면 해당 인스턴스 밸류 값이 업데이트되기를 원할것인가?

서술적 로컬 변수

```
boolean method(int year) {
    return year % 4 == 0 && (!(year % 100 == 0) || year % 400 == 0);
}

---
boolean method(int year) {
    bool fourth = year % 4 == 0
    bool hundredth = (year % 100 == 0)
    bool fourHundredth = year % 400 == 0
    return fourth && (!hundredth || fourHundredth);
}
```

모든 값들에 대한 의미가 없이 하나의 라인으로 사용해도 된다. 하지만 해당 값들에 대한 의미를 로컬 변수로 명확하게 하게 되면 코드를 이해하는데 큰 도움이 된다.

정리

1. 주석 정리
2. Enum을 활용한 정적 변수 활용
3. 정적 변수와 정적 데이터를 UitIs라는 파일로 만듬
4. 추상적인 개념에 메소드는 추상 클래스로 구체적인 메소드와 변수는 구현 클래스로

결론

보이스카우트 규칙. 언제나 우리가 처음 소스코드를 본 것보다 나아지게 만들자.