전략파트너개발그룹

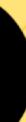
쉽게익히는 클린코드

*리펙. 필드 옮기기 ~(

이번주컨텐츠는? ~냄새. 산탄총수술

*리펙. 함수 인라인

> *리펙. 클래스 인라인





~냄새. 산탄총 수술

Shotgun Surgery

- 어떤 한 변경 사항이 생겼을 때 여러 모듈을 (여러 함수 또 는 여러 클래스를) 수정해야 하는 상황
 - "**뒤엉킨 변경**" 냄새와 유사하지만 반대의 상황이다.
 - 예) 새로운 결제 방식을 도입하려면 여러 클래스의 코 드를 수정해야 한다.
- 변경 사항이 여러 곳에 흩어진다면 찾아서 고치기도 어렵고 중요한 변경 사항을 놓칠 수 있는 가능성도 생긴다.



~냄새. 산탄총 수술

사용할 수 있는 리펙토링 기술

- "함수 옮기기(Move Function)" 또는 "필드 옮기기(Move Field)"를 사용해서 필요한 변경 내역을 하나의 클래스로 모을수 있다.
- 비슷한 데이터를 사용하는 여러 함수가 있따면 "여러 함수를 클 래스로 묶기(Combine Functions into Class)"를 사용할 수 있다.
- "단계 쪼개기(Split Phase)"를 사용해 공통으로 사용되는 함 수의 결과물들을 하나로 묶을 수 있다.
- "함수 인라인(Inline Function)"과 "클래스 인라인(Inline Class)"로 흩어진 로직을 한 곳으로 모을 수도 있다.

주황색 리펙토링 방법은 이전 호에서 확인해주세요

*리펙토링. 필드 옮기기

Move Field

- 좋은 데이터 구조를 가지고 있다면, 해당 데이터에 기반한 어떤 행위를 코드로(메소드나 함수) 옮기는 것도 간편하고 단순해진다.
- 처음에는 타당해 보였던 설계적인 의사 결정도 프로그램이 다루고 있는 도메인과 데이터 구조에 대해 더 많이 익혀나가면서, 틀린 의사 결정으로 바뀌는 경우도 있다.

필드 옮기는 단서

- * 어떤 데이터를 항상 어떤 레코드와 함께 전달하는 경우
- 어떤 레코드를 변경할 때 다른 레코드에 있는 필드를 변경해야 하는 경우
- * 여러 레코드에 동일한 필드를 수정해야 하는 경우
- * (위 언급된 '레코드'는 클래스 또는 객체로 대체할 수도 있음)

```
class ClassA {
    aDataFieldA;
}
class ClassB {
    class ClassB {
        aDataFieldA;
}
```

*리펙토링. 함수 인라인



Inline Function

- "함수 추출하기"의 반대에 해당하는 리팩토링
 - 함수 추출하기: 함수로 추출하여 함수 이름으로 의도를 표현하는 방법
- 간혹, 함수 본문이 함수 이름만큼 또는 그보다 더 잘 의도를 표현하는 경우도 있다.
- 함수 리팩토링이 잘못된 경우에 여러 함수를 인라인하여 커다른 함수를 만든 다음에 다시 함수 추출하기를 시도할 수 있다.
- 단순히 메소드 호출을 감싸는 우회형(Indirection) 메소드라면 인라인으로 없앨 수 있다.
- 상속 구조에서 오버라이딩하고 있는 메소드는 인라인 할 수 없다.(해당 메소는 일종 의 규약 이기 때문)

```
int getRatings() {
    return moreThanFiveLateDeliveries() ? 2 : 1;
}
boolean moreThanFiveLateDeliveries() {
    return numberOfLateDeliveries > 5;
}
```

```
int getRatings() {
  return numberOfLateDeliveries > 5 ? 2 : 1;
}
```





*리펙토링. 인라인 클래스

Inline Class

- "클래스 추출하기(Extract Class)"의 반대에 해당하는 리팩토링
- 리팩토링을 하는 중에 클래스의 책임을 옮기다보면 클래스의 존재 이유가 빈약 해지는 경우가 발생할 수 있다.
- 두개의 클래스를 여러 클래스로 나누는 리팩토링을 하는 경우에, 우선 "클래스 인라인"을 적용해서 두 클래스의 코드를 한 곳으로 모으고, 그 다음 "클래스 추 출하기"를 적용해서 새롭게 분리하는 리팩토링을 적용할 수 있다.

```
class Person {
    getName() { ... }
    getOfficeAreaCode() { return teleNum.getAreaCode(); }
    getOfficeNumber() { return teleNum.getNumber(); }
}
class TelephoneNumber {
    getAreaCode() { ... }
    getNumber() { ... }
    getName() { ... }
    getOfficeAreaCode() { return this.officeAC; }
```

getOfficeNumber() { return

this.officeNum: }

클린코드 꿀팁을 놓치고 있다면?

물로모든

