2022.07.11

리팩토링
Refactoring

이예슬

리팩토링(Refactoring)이란?

- Re + factor + ing = 재구성
- 코드를 다듬어 개선하는 과정
  - 이해하기 쉬운 코드로
  - 수정하기 쉬운 코드로
  - 복잡한 코드를 단순화 한 코드로
- 프로그램의 전체 구조는 그대로 유지
  - 성능 개선

리팩토링이 왜 필요한가?

- 코드는 작성하는 시간보다 읽히는 시간이 더 길다!
- 코드의 가독성 증가
  - 코드를 공유하며 더 나은 기능을 개발
- 유지보수 용이
- 중복코드 제거(코드의 간결성)
- 성능 개선

리팩토링 실전 예제

- 복권프로그램

• 동전을 입력 받으면서 실행

• 실행마다 랜덤한 확률(10%)로 당첨/꽝 여부와 그에 따른 잔액 출력

ㅡ꽝: -1000

ㅡ당첨: +5000

• 각 실행 후에는 끝낼지 여부 확인

예제 코드 ➡ 리팩토링 코드

```
from random import randint

InputCoin=(input("동전을 넣으세요: "))
coin=int(InputCoin)

while coin >= 1000:
    if randint(1, 10) == 1:
        print("당첨!(+5000)")
        coin += 5000
    else:
        print("꽝!(-1000)")
        coin -= 1000

    print("현재금액:", coin)
    i = input("₩n끝내시겠습니까?: ")
    if i == "y": break

print("출력 금액: ", coin)
```

```
def insertCoin():
    return int(input("동전을 넣으세요: "))
def Lotto(coin):
    if randint(1, 10) == 1:
        print("당첨!(+5000)")
        return coin + 5000
    else:
        print("꽝!(-1000)")
        return coin - 1000
def show(coin):
    print("현재금액:", coin)

#----------------------------------------
from random import randint
coin = insertCoin()

while coin >= 1000:
    coin = Lotto(coin)
    show(coin)
    goAndStop = input("₩n끝내시겠습니까?: ")
    if goAndStop == "y": break

print("출력 금액: ", coin)
```

함수화를 통해 역할 구분

함수명과 변수명을 직관적으로