클린코드

3장 함수

작게 만들어라! ~ 함수 인수

2023.06.20 무니

3장 함수

작게 만들어라!

- 함수를 만드는 첫째 규칙은 ‘작게!’
- 함수를 만드는 둘째 규칙은 ‘더 작게!’
- 근거를 대기는 곤란하지만 저자의 경험상 작은 함수가 좋았다.
- if, else, while 문 등에 들어가는 블록은 한 줄 이어야 한다.
- 함수 안에서 들여쓰기 는 2단을 넘어서면 안된다.

3장 함수
작게 만들어라!

- 자바 코드 예

```
# 자바 코드
public static String renderPageWithSetupsAndTeardowns( PageData pageData, boolean isSuite) throws E
        boolean isTestPage = pageData.hasAttribute("Test");
        if (isTestPage) {
                WikiPage testPage = pageData.getWikiPage();
                StringBuffer newPageContent = new StringBuffer();
                includeSetupPages(testPage, newPageContent, isSuite);
                newPageContent.append(pageData.getContent());
                includeTeardownPages(testPage, newPageContent, isSuite);
                pageData.setContent(newPageContent.toString());
        }
        return pageData.getHtml();
}
```



```
# 자바 코드
public static String renderPageWithSetupsAndTeardowns( PageData pageData, boolean isSuite) throws E
   if (isTestPage(pageData))
        includeSetupAndTeardownPages(pageData, isSuite);
   return pageData.getHtml();
}
```

- 파이썬 코드 예

```
# 파이썬 코드
def render_page_with_setups_and_teardowns(page_data, is_suite):
    is_test_page = page_data.has_attribute("Test")
    if is_test_page:
        test_page = page_data.get_wiki_page()
        new_page_content = []
        include_setup_pages(test_page, new_page_content, is_suite)
        new_page_content.append(page_data.get_content())
        include_teardown_pages(test_page, new_page_content, is_suite)
        page_data.set_content(''.join(new_page_content))
    return page_data.get_html()
```



```
# 파이썬 코드
def render_page_with_setups_and_teardowns(page_data, is_suite):
    if is_test_page(page_data):
        include_setup_and_teardown_pages(page_data, is_suite)
    return page_data.get_html()
```

3장 함수

한 가지만 해라!

- 함수는 한가지를 해야 한다. 그 한가지를 잘 해야 한다. 그 한가지만을 해야 한다.
- 지정된 함수 이름 아래에서 추상화 수준이 하나인 단계만 수행한다면 그 함수는 한 가지 작업만 하는 것이다.
- 함수를 여러 섹션으로 나눌 수 있다면 그 함수는 여러작업을 하는 셈이다.

3장 함수

함수 당 추상화 수준은 하나로!

- 함수가 ‘한가지’ 작업만 하려면 함수 내 모든 문장의 추상화 수준이 동일해야 된다.
- 만약 한 함수 내에 추상화 수준이 섞이게 된다면 읽는 사람이 헷갈린다.
- 위에서 아래로 내려갈 수록 추상화 수준이 한단계씩 낮아진다.
  - - 설정 페이지와 해제 페이지를 포함하려면 설정 페이지를 포함하고 해제 페이지를 포함한다
  - - 설정 페이지를 포함하려면 슈트이면 슈트 설정 페이지를 포함한 후 일반 설정 페이지를 포함한다
  - - 슈트 설정 페이지를 포함하려면 ...

3장 함수

Switch 문

- switch 문은 한번에 N개의 작업을 하며 코드가 길어지기 때문에 쓰지 않는 것이 좋다.
- 꼭 써야하는 경우라면 다형적 객체를 생성하는 코드에서 상속 관계로 숨기자.
- 단 한번만 허용한다.

3장 함수

Switch 문

- 자바 코드 예

```
# 자바 코드
public Money calculatePay(Employee e) throws InvalidEmployeeType {
        switch (e.type) {
                case COMMISSIONED:
                        return calculateCommissionedPay(e);
                case HOURLY:
                        return calculateHourlyPay(e);
                case SALARIED:
                        return calculateSalariedPay(e);
                default:
                        throw new InvalidEmployeeType(e.type);
        }
}
```



```
public abstract class Employee {
        public abstract boolean isPayday();
        public abstract Money calculatePay();
        public abstract void deliverPay(Money pay);
}
------------------
public interface EmployeeFactory {
        public Employee makeEmployee(EmployeeRecord r) throws InvalidEmployeeType;
}
------------------
public class EmployeeFactoryImpl implements EmployeeFactory {
        public Employee makeEmployee(EmployeeRecord r) throws InvalidEmployeeType {
                switch (r.type) {
                        case COMMISSIONED:
                                return new CommissionedEmployee(r) ;
                        case HOURLY:
                                return new HourlyEmployee(r);
                        case SALARIED:
                                return new SalariedEmploye(r);
                        default:
                                throw new InvalidEmployeeType(r.type);
                }
        }
}
```

3장 함수

Switch 문

- 파이썬 코드 예

```python
# 파이썬 코드
class InvalidEmployeeType(Exception):
    pass

def calculate_pay(e):
    if e.type == "COMMISSIONED":
        return calculate_commissioned_pay(e)
    elif e.type == "HOURLY":
        return calculate_hourly_pay(e)
    elif e.type == "SALARIED":
        return calculate_salaried_pay(e)
    else:
        raise InvalidEmployeeType(e.type)
```



```python
# 파이썬 코드
from abc import ABC, abstractmethod

class EmployeeFactory(ABC):
    @abstractmethod
    def make_employee(self, r):
        pass
------

class EmployeeFactoryImpl(EmployeeFactory):
    def make_employee(self, r):
        if r.type == "COMMISSIONED":
            return CommissionedEmployee(r)
        elif r.type == "HOURLY":
            return HourlyEmployee(r)
        elif r.type == "SALARIED":
            return SalariedEmployee(r)
        else:
            raise InvalidEmployeeType(r.type)
```

3장 함수

서술적인 이름을 사용하라!

- “코드를 읽으면서 짐작했던 기능을 각 루틴이 그대로 수행한다면 깨끗한 코드라 불러도 되겠다” - 워드
- 작은 함수는 그 기능이 명확하므로 이름을 붙이기가 더 쉬우며, 일관성 있는 서술형 이름을 사용한다면 코드를 순차적으로 이해하기도 쉬워진다
- 함수 이름 예시: SetUpTeardownIncluder, isTestable

3장 함수

함수 인수

- 함수에서 이상적인 인수 개수는 0개(무항).
- 인수는 코드 이해에 방해가 되는 요소이므로 최선은 0개이고, 차선은 1개뿐인 경우이다.
- 출력인수(함수의 반환 값이 아닌 입력 인수로 결과를 받는 경우)는 이해하기 어려우므로 웬만하면 쓰지 말자.

3장 함수

함수 인수 - 많이 쓰는 단항 형식

- 함수에 인수 1개를 넘기는 경우
- - 질문을 던지는 경우 ex) boolean fileExists(“MyFile")
- - 인수를 뭔가로 변환해 결과를 반환하는 경우 ex) InputStream fileOpen(“MyFile")
- - 출력 인수 없이 시스템 상태를 바꾸는 이벤트
  ex) passwordAttemptFailedNtimes(int attempts)

3장 함수

함수 인수 - 플래그 인수

- 플래그 인수 (boolean 인수)는 금물. 참이면 이거 하고 거짓이면 저걸 한다는 여러가지를 하겠다는 말이니.
- 함수 두 개로 나누자

3장 함수

함수 인수 - 0항 함수

- Point p = new Point(0, 0) 처럼 인수 2개가 한 값을 표현하는 요소라면 상관없지만, 그 외에는 항을 줄여보자.
- writeField(outputStream, name)
  - - writeField 메서드를 outputStream 클래스 구성원으로 만들어서 outputStream.writeField(name)으로 호출
  - - outputStream을 현재 클래스 구성원 변수로 만들어 인수로 넘기지 않기
  - - FieldWriter라는 새 클래스를 만들어서 구성자(init?)에서 outputStream을 받고 write 메서드를 구현하기

3장 함수

함수 인수 - 삼항 함수

- 인수가 3개인 함수는 독자가 주춤하게 되므로 좋지 않다. 최대한 쓰지 말자.
- 단, assertEquals(1.0, amount, .001)은 주춤하게 되지만 그만한 가치가 있다. 부동소수점 비교가 상대적이라는 사실은 중요하기 때문.
- 충분히 고려하고 사용하자.

3장 함수

함수 인수 - 인수 객체

- 인수가 2-3개 필요하다면 일부를 독자적인 클래스 변수로 선언해서 인수 개수를 줄여보자
  Circle makeCircle(double x, double y, double radius); ->
  Circle makeCircle(Point center, double radius)

3장 함수

함수 인수 - 인수 목록

- String.format("%s worked %.2f" hours.", name, hours); or
- "{} worked {:.2f} hours.".format(name, hours) 처럼 인수 개수가 가변적인 함수도 있다.
- 이럴 경우에는 List형 인수 하나로 취급할 수 있다. 이런 논리로 보면 위 함수는 사실 이항 함수이다.
- 가변 인수를 취하는 모든 함수에 같은 원리가 적용된다. 이 경우에도 삼항을 넘어서지 않도록 하자.

3장 함수

함수 인수 - 동사와 키워드

- 단항 함수는 동사(명사) 짝을 이루어 이름을 짓자.
- ex) writeField(name)
- 인수를 함수이름에 추가하는 것도 좋다.
- assertExpectedEqualsActual(expected,actual) -> 인수 순서를 기억할 필요가 없어진다.