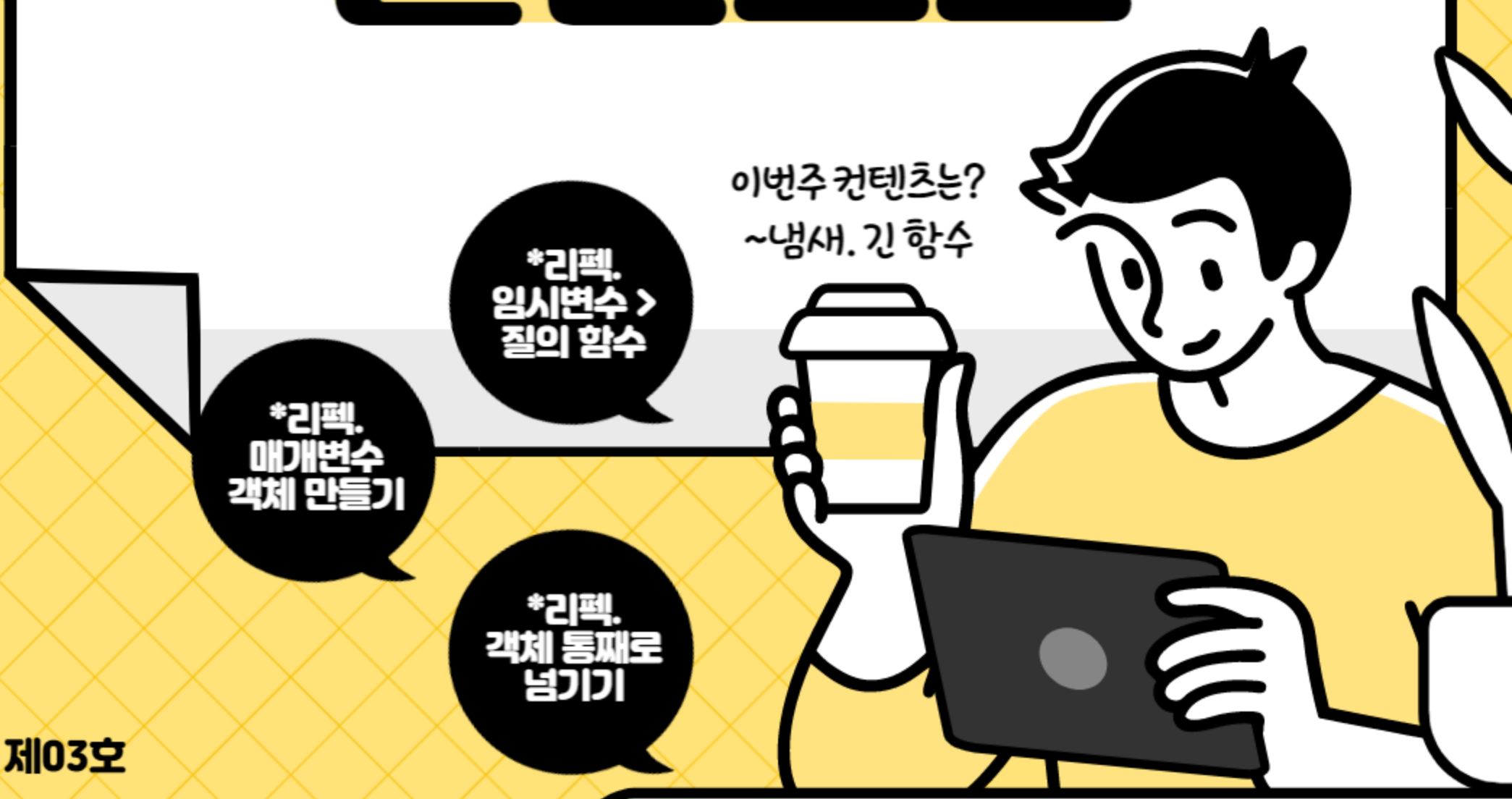
전략파트너개발그룹

쉽게 익히는 클린코드



제03호

~냄새. 긴 함수

짧은 함수 vs 긴 함수

- 함수가 길 수록 더 이해하기 어렵다. vs 짧은 함수는 더 많은 문맥 전환을 필요로 한다.
- "과거에는" 작은 함수를 사용하는 경우 서브루틴 호출로 인한 오버헤드가 있었다.
- 작은 함수에 "좋은 이름"을 사용했다면 해당 함수의 코드를 보지 않고도 이해할 수 있다.
- 어떤 코드에 *"주석"을 남기고 싶다면*, 주석 대신 함수를 만들고 함수의 이름으로 "의도"를 표현해보자.

~냄새. 긴 함수

사용할 수 있는 리펙토링 기술

- 99%는 "함수 추출하기(Extract Function)"로 해결할 수 있다.
- 함수를 분리하면서 해당 함수로 전달해야 할 매개변수가 많아진다면 다음과 같은 리펙토링을 고려해볼 수 있다.
  - 임시 변수를 질의 함수로 바꾸기
  - 매개변수 객체 만들기
  - 객체 통째로 넘기기
  - 함수를 명령으로 바꾸기
- "조건문 분해하기"를 사용해 조건문 분리할 수 있다.
- 같은 조건으로 여러개 Switch문이 있다면, "조건문을 다형성으로 바꾸기"을 사용할 수 있다.
- 반복문 안에서 여러 작업을 하고 있어서 하나의 메소드로 추출이 어렵다면, "반복문 쪼개기"를 적용할 수 있다.

주황색 리펙토링 방법은 다음호에서 만나요

*리펙토링. 임시 변수를 질의 함수로 바꾸기

Replace Temp with Query

1 *변수*를 사용하면 반복해서 **동일한 식을 계산**하는 것을 피할 수 있고, **이름을 사용해 의미를 표현**할 수도 있다.

2 긴 함수를 리펙토링할 때, 그러한 **임시 변수를 함수로 추출하여 분리**한다면 빼난 함수로 **전달해야할 매개변수를 줄일** 수 있다.

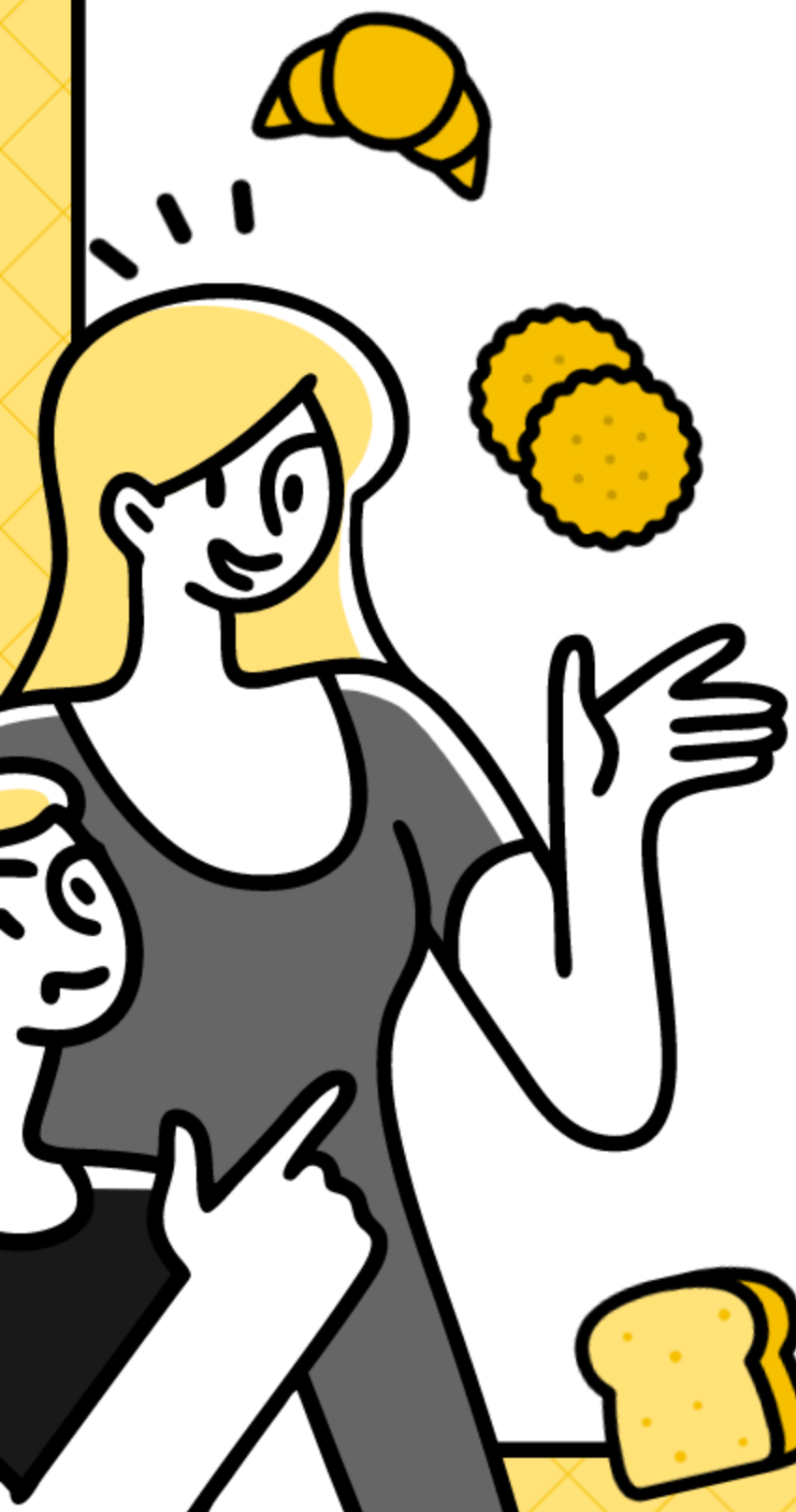
```
double basePrice = quantity *
itemPrice;
if(basePrice > 1000)
   return basePrice * 0.95;
else
   return basePrice * 0.98;
```

→

```
if(basePrice() > 1000)
   return basePrice() * 0.95;
else
   return basePrice() * 0.98;

double basePrice() {
   return quantity * itemPrice;
}
```

*리펙토링. 매개변수 객체 만들기



Introduce Parameter Object

같은 매개변수들이 여러 메소드에 걸쳐 나타난다면 그 매개변수들을 묶은 자료 구조를 만들 수 있다.
그렇게 만든 자료구조는:

- 해당 데이터간의 관계를 보다 명시적으로 나타낼 수 있다.
- 함수에 전달할 매개변수 개수를 줄일 수 있다.
- 도메인을 이해하는데 중요한 역할을 하는 클래스로 발전할 수도 있다.

```
amountInvoiced(sDate, eDate);
amountReceived(sDate, eDate);
```

↓

```
amountInvoiced(dateRange);
amountReceived(dataRange);
```

*리펙토링. 객체 통째로 넘기기

Preserve Whole Object

어떤 한 레코드에서 구할 수 있는 여러 값들을 함수에 전달하는 경우, 해당 매개변수를 **레코드 하나로 교체**할 수 있다.
이 방법을 적용 전에 **의존성을 고려**해야 한다.
어쩌면 해당 **메소드의 위치가 적절하지 않을 수도 있기** 때문에 적절한 수정이 필요할 수도 있다.

```
int low = daysTempRange.getLow();
int high = daysTempRange.getHigh();
if(plan.withinRange(low, high)) { ... }
```

```
if(plan.withingRnage(daysTempRange))
{ ... }
```

클린코드 꿀팁을 놓치고 있다면?

클린코드 팁 구독해요!

전략파트너개발그룹 - 양지용