

Що означає ця тема?

Техніка "Замінити параметр викликом методу" — це метод рефакторингу, який передбачає заміну параметра, що передається в метод, викликом іншого методу, який отримує це значення. Це допомагає зробити код чистішим, простішим і менш залежним від зовнішніх значень.

Для чого вона потрібна?

Основні причини використання цього рефакторингу:

Зменшення кількості параметрів у методах, що робить код більш зрозумілим.

Забезпечення узгодженості даних, оскільки значення отримується з об'єкта, а не передається вручну.

Мінімізація ризику помилок, пов'язаних із передачею некоректних значень.

Краща інкапсуляція, адже метод використовує внутрішні дані об'єкта.

```
class Order {  
    private Customer customer;  
  
    public double getFinalPrice(double discountRate) {  
        return getBasePrice() * (1 - discountRate);  
    }  
}
```

Метод `getFinalPrice(double discountRate)` приймає `discountRate` як параметр. Потім він використовує цей параметр для розрахунку кінцевої ціни (`getBasePrice() * (1 - discountRate)`).

Це означає, що кожен виклик методу потребує передачі `discountRate` явно.

```
class Order {  
    private Customer customer;  
  
    public double getFinalPrice() {  
        return getBasePrice() * (1 - customer.getDiscountRate());  
    }  
}
```

Метод `getFinalPrice()` більше не приймає `discountRate` як параметр.

Замість цього він напямую отримує `discountRate` від об'єкта `customer` через `customer.getDiscountRate()`.

Це усуває потребу передавати `discountRate` при кожному виклику, оскільки він уже доступний всередині `Order`.

Що помінялося?

- Метод `getFinalPrice()` більше не потребує параметра `discountRate`, оскільки отримує його з об'єкта `customer`.
- Код став простішим і чистішим, бо зникла необхідність передавати значення вручну.
- Покращилася інкапсуляція, оскільки метод більше не залежить від зовнішніх даних, а працює з внутрішнім станом об'єкта.
- Менше шансів на помилки, адже `discountRate` завжди буде відповідати значенню в `customer`.

Цей рефакторинг допомагає зробити код зрозумілим, надійнішим і менш залежним від зайвих параметрів.

THANK YOU