GoF · 10/23 СТРУКТУРНЫЙ

Python C++23 Rust Java





Фасад

Facade

Алексей Примак

Алексей Примак

Фасад. Паттерн GoF на Python, C++23, Rust и Java

https://litres.ru/74810697
SelfPub; 2026

Аннотация

Сложная операция часто требует согласованного вызова нескольких подсистем. «Фасад» показывает, как скрыть эту сложность за единым интерфейсом: от проверки остатков и оплаты до доставки и уведомления клиента. Паттерн GoF разобран на практической задаче и реализован на Python, C++23, Rust и Java. В книге рассмотрены структура фасада, варианты организации логики, модификации и самостоятельные работы.

Содержание

Введение 4
Как читать 5
Глава 10. Реализация на Python 6
10.1. Зачем нужен фасад 6
10.2. Участники и схема 7
10.3. Классический пример: оформление заказа 8
Конец ознакомительного фрагмента. 13

Фасад. Паттерн GoF на Python, C++23, Rust и Java

Введение

Эта книга посвящена одному паттерну проектирования — **Фасад**(Facade). Он относится к каталогу «Банды четырёх» (Gang of Four, GoF). Здесь паттерн разобран сразу в четырёх реализациях: на Python, C++23, Rust и Java. Каждая реализация — отдельная часть книги: сначала задача и общая схема, затем код, модификации, гибридные варианты и самостоятельные работы.

Как читать

Основы каждого языка здесь не повторяются: они собраны в книгах серий по соответствующему языку. Разделы каждой части нумеруются заново, поэтому ссылка «раздел 1.2» указывает на раздел той же части, где она сделана.

Глава 10. Реализация на Python

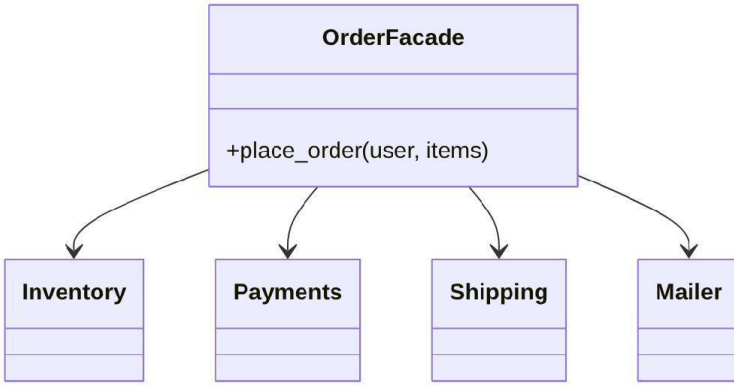
10.1. Зачем нужен фасад

Чтобы оформить заказ в интернет-магазине, нужно проверить остатки на складе, зарезервировать товар, списать деньги, создать отправку в службе доставки и отправить письмо. Пять подсистем, у каждой свой интерфейс, свои ошибки и свой порядок вызовов. Если каждый обработчик веб-запроса будет делать это сам, логика размножится, а ошибки отката (деньги списали, а доставка не создалась) появятся в самых неожиданных местах.

Фасад — простой интерфейс к сложной подсистеме. Он знает, каких участников вызвать и в каком порядке, а клиент вызывает один метод place_order. Подсистемы при этом остаются доступны напрямую для тех, кому нужен тонкий контроль.

В стандартной библиотеке Python много фасадов: shutil.copytree прячет обход каталогов и копирование метаданных, subprocess.run — создание процесса, каналы и ожидание, urllib.request.urlopen — сокеты, HTTP и перенаправления.

10.2. Участники и схема



Facade— знает, каким подсистемам делегировать запрос, и координирует их.

Subsystem classes— выполняют реальную работу и ничего не знают о фасаде.

10.3. Классический пример: оформление заказа

Файл 01_order_facade.py:

01_order_facade.py Python

```python
class OutOfStock(Exception):
    pass


class PaymentDeclined(Exception):
    pass


class Inventory:
    def __init__(self, stock: dict[str, int]) -> None:
        self.stock = stock

    def reserve(self, items: dict[str, int]) -> None:
        for sku, qty in items.items():
            if self.stock.get(sku, 0) < qty:
                raise OutOfStock(sku)
        for sku, qty in items.items():
            self.stock[sku] -= qty

    def release(self, items: dict[str, int]) -> None:
        for sku, qty in items.items():
            self.stock[sku] += qty


class Payments:
    def __init__(self, balances: dict[str, int]) -> None:
        self.balances = balances

    def charge(self, user: str, amount: int) -> str:
        if self.balances.get(user, 0) < amount:
            raise PaymentDeclined(user)
        self.balances[user] -= amount
        return f"PAY-{user[:3].upper()}-{amount}"

```

Листинг 10.1 *— Классический пример: оформление зака-*

01_order_facade.py (продолжение) Python

```python

class Shipping:
    def __init__(self) -> None:
        self.counter = 1000

    def create(self, user: str, items: dict[str, int]) -> str:
        self.counter += 1
        return f"TRK{self.counter}"


class Mailer:
    def send(self, user: str, text: str) -> None:
        print(f"  письмо для {user}: {text}")


class OrderFacade:
    PRICES = {"книга": 450, "кружка": 300, "футболка": 900}

    def __init__(self, inventory: Inventory, payments: Payments)
        -> None:
        self.inventory, self.payments = inventory, payments
        self.shipping, self.mailer = Shipping(), Mailer()

    def place_order(self, user: str, items: dict[str, int]) ->
        str:
        total = sum(self.PRICES[sku] * qty for sku, qty in items
            .items())
        try:
            self.inventory.reserve(items)
        except OutOfStock as e:
            return f"отказ: нет товара «{e}»"
        try:
            payment = self.payments.charge(user, total)
        except PaymentDeclined:
```

Листинг 10.1 (продолжение) *— Классический пример: оформление заказа*

01_order_facade.py (продолжение) Python

```
            self.inventory.release(items)  # откат резерва
            return f"отказ: не хватает средств на {total} ₽"
        track = self.shipping.create(user, items)
        self.mailer.send(user, f"заказ на {total} ₽ оплачен ({
            payment}), трек {track}")
        return f"успех: {track}"


if __name__ == "__main__":
    inv = Inventory({"книга": 5, "кружка": 1, "футболка": 2})
    shop = OrderFacade(inv, Payments({"anna": 2000, "boris": 500
        }))
    print(shop.place_order("anna", {"книга": 2, "кружка": 1}))
    print(shop.place_order("anna", {"кружка": 1}))
    print(shop.place_order("boris", {"футболка": 1}))
    print("Остатки:", inv.stock)
```

```
$ python 01_order_facade.py
  письмо для anna: заказ на 1200 ₽ оплачен (PAY-ANN-1200), трек
    TRK1001
успех: TRK1001
отказ: нет товара «кружка»
отказ: не хватает средств на 900 ₽
Остатки: {'книга': 3, 'кружка': 0, 'футболка': 2}
```

Листинг 10.1 (продолжение) *— Классический пример: оформление заказа*

Разбор.Клиенту нужно знать один метод. Самое ценное

в фасаде — согласованность: если оплата не прошла, резерв снимается, и футболок снова две. Без фасада каждый вызывающий код должен был бы помнить об этом откате. Фасад не скрывает подсистемы полностью: inv доступен, и можно, например, проверить остатки напрямую.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, купив полную легальную версию на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.