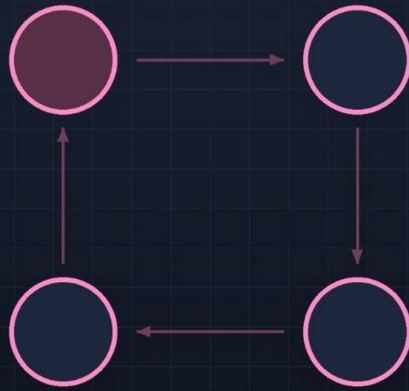
GoF · 20/23 ПОВЕДЕНЧЕСКИЙ

Python C++23 Rust Java



Состояние

State

Алексей Примак

Алексей Примак

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

https://litres.ru/74810687
SelfPub; 2026

Аннотация

Сложная логика по состояниям часто превращается в длинную цепочку условий. «Состояние» показывает, как представить состояния объекта отдельными компонентами и перенести поведение из условных операторов в соответствующие классы. На примере жизненного цикла заказа разобраны правила переходов и допустимых операций. Паттерн GoF реализован на Python, C++23, Rust и Java, с модификациями и практическими заданиями.

Содержание

Введение 4
Как читать 5
Глава 20. Реализация на Python 6
20.1. Зачем нужно состояние 6
20.2. Участники и схема 7
20.3. Классический пример: заказ 10
Конец ознакомительного фрагмента. 16

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

Введение

Эта книга посвящена одному паттерну проектирования — **Состояние**(State). 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 20. Реализация на Python

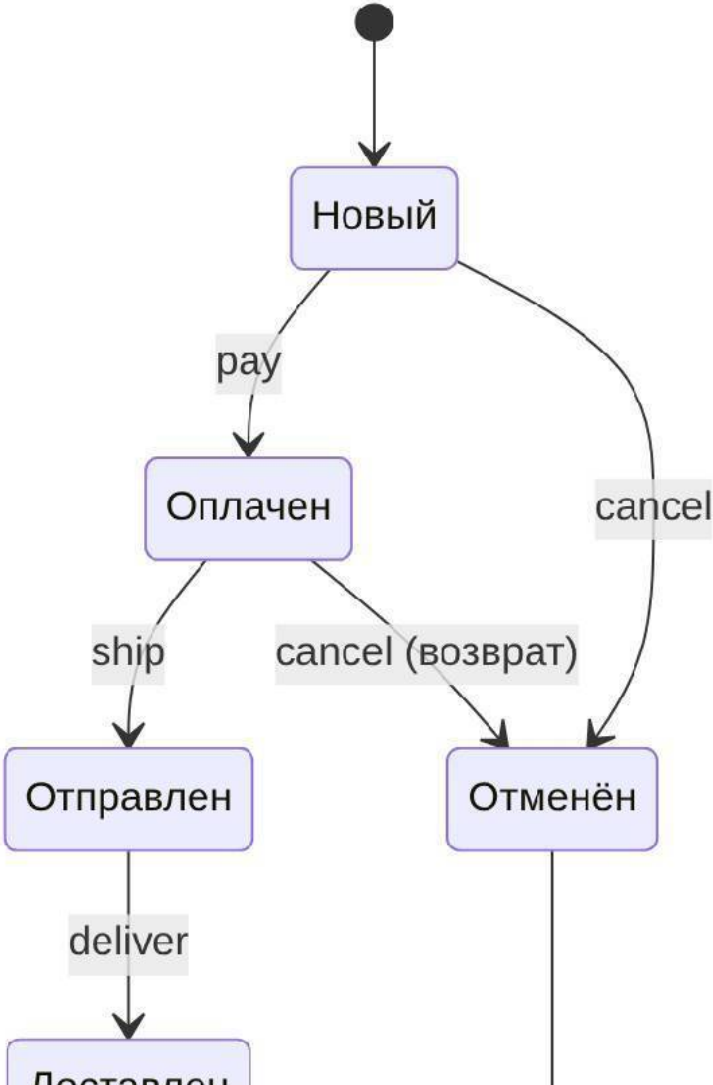
20.1. Зачем нужно состояние

Заказ в интернет-магазине бывает новым, оплаченным, отправленным, доставленным и отменённым. Отменить можно новый и оплаченный заказ, но не отправленный. Оплатить — только новый. Если всё это писать условиями в каждом методе, получится лес из if self.status == …, и каждое новое состояние заставит править все методы сразу.

Паттерн «Состояние» выносит поведение каждого состояния в отдельный объект. Основной объект (контекст) хранит ссылку на текущее состояние и делегирует ему вызовы. Переход — это замена объекта состояния. Снаружи кажется, что объект поменял класс.

В Python состояние можно реализовать классами, словарём переходов или даже генератором — выбор зависит от того, сколько поведения у каждого состояния.

20.2. Участники и схема


Новый
pay
Оплачен
cancel
ship
cancel (возврат)
Отправлен
Отменён
deliver

Context— объект, поведение которого зависит от состояния; хранит текущее состояние.

State— интерфейс состояния.

Concrete State— поведение в одном состоянии и правила перехода из него.

20.3. Классический пример: заказ

Файл 01_order_states.py:

01_order_states.py Python

```python
from __future__ import annotations


class OrderState:
    name = "?"

    def pay(self, order: Order) -> None:
        raise InvalidAction(f"нельзя оплатить: заказ {self.name}
            ")

    def ship(self, order: Order) -> None:
        raise InvalidAction(f"нельзя отправить: заказ {self.name
            }")

    def deliver(self, order: Order) -> None:
        raise InvalidAction(f"нельзя доставить: заказ {self.name
            }")

    def cancel(self, order: Order) -> None:
        raise InvalidAction(f"нельзя отменить: заказ {self.name}
            ")


class InvalidAction(Exception):
    pass


class New(OrderState):
    name = "новый"

    def pay(self, order):
        order.paid = order.total
        order.state = Paid()

```

Листинг 20.1 *— Классический пример: заказ*

01_order_states.py (продолжение) Python

```python
    def cancel(self, order):
        order.state = Cancelled()


class Paid(OrderState):
    name = "оплачен"

    def ship(self, order):
        order.state = Shipped()

    def cancel(self, order):
        order.log.append(f"возврат {order.paid} ₽")
        order.paid = 0
        order.state = Cancelled()


class Shipped(OrderState):
    name = "отправлен"

    def deliver(self, order):
        order.state = Delivered()


class Delivered(OrderState):
    name = "доставлен"


class Cancelled(OrderState):
    name = "отменён"


class Order:
    def __init__(self, total: int) -> None:
        self.total, self.paid = total, 0
```

Листинг 20.1 (продолжение) *— Классический пример: заказ*

01_order_states.py (продолжение) Python

```python
        self.state: OrderState = New()
        self.log: list[str] = []

    def do(self, action: str) -> None:
        before = self.state.name
        try:
            getattr(self.state, action)(self)
            self.log.append(f"{action}: {before} → {self.state.
                name}")
        except InvalidAction as e:
            self.log.append(f"{action}: отказ ({e})")


if __name__ == "__main__":
    a = Order(2500)
    for act in ("ship", "pay", "pay", "ship", "cancel", "deliver
        "):
        a.do(act)
    b = Order(900)
    for act in ("pay", "cancel", "pay"):
        b.do(act)
    print("Заказ A:", *a.log, sep="\n  ")
    print("Заказ B:", *b.log, sep="\n  ")
```

```
$ python 01_order_states.py
Заказ A:
  ship: отказ (нельзя отправить: заказ новый)
  pay: новый → оплачен
  pay: отказ (нельзя оплатить: заказ оплачен)
  ship: оплачен → отправлен
  cancel: отказ (нельзя отменить: заказ отправлен)
  deliver: отправлен → доставлен
Заказ B:
  pay: новый → оплачен
  возврат 900 ₽
  cancel: оплачен → отменён
  pay: отказ (нельзя оплатить: заказ отменён)
```

Листинг 20.1 (продолжение) *— Классический пример: заказ*

Разбор. Базовый класс состояния по умолчанию запрещает все действия, а каждое конкретное состояние разрешает только свои. Поэтому в коде нет ни одной проверки вида if status == … Отмена ведёт себя по-разному в зависимости от состояния: из «нового» заказ просто отменяется, из «оплаченного» — с возвратом денег. Новое состояние, например «на сборке», — это новый класс и правка двух соседних состояний, а не всех методов заказа.

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.