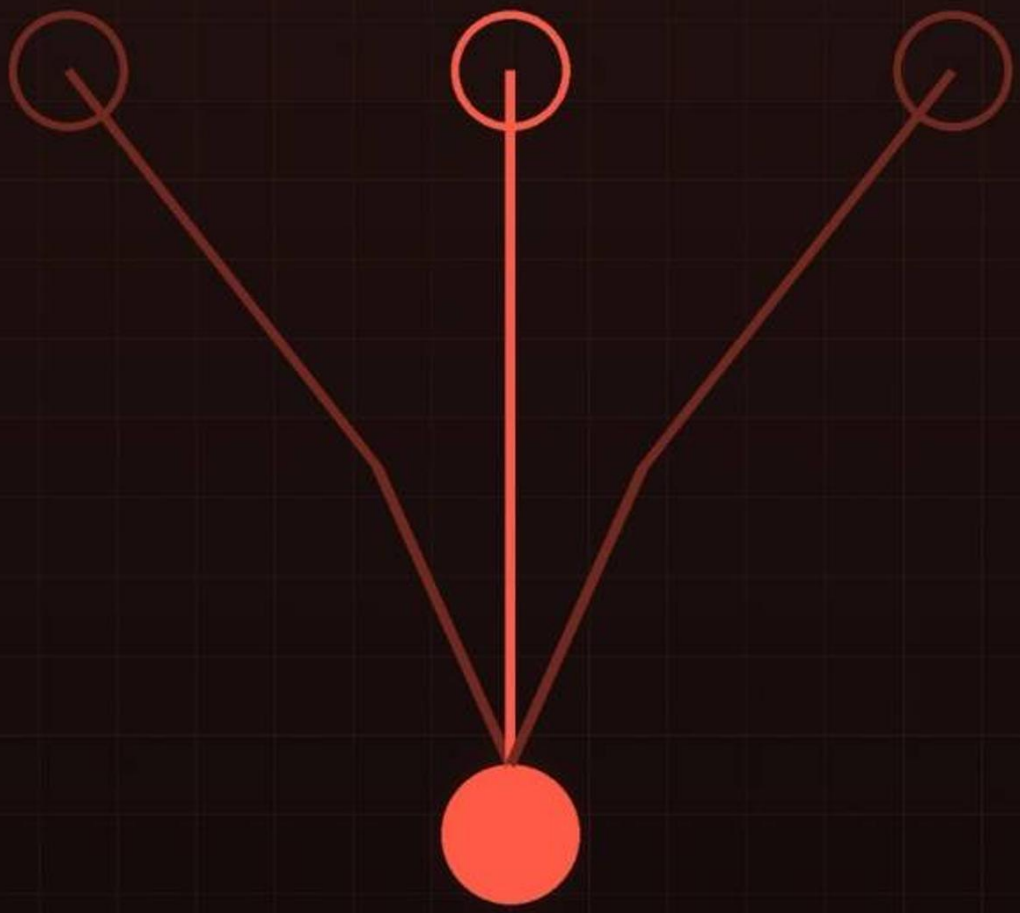
GoF · 21/23 ПОВЕДЕНЧЕСКИЙ

Python C++23 Rust Java



Стратегия

Strategy

Алексей Примак

Алексей Примак

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

«Автор»
2026

Примак А.

Стратегия. Паттерн GoF на Python, C++23, Rust и Java /
А. Примак — «Автор», 2026

Одна задача может решаться несколькими взаимозаменяемыми способами: маршрут строится для автомобиля, пешком или на велосипеде, доставка выполняется разными способами, файлы сжимаются разными алгоритмами. «Стратегия» показывает, как вынести алгоритмы в отдельные объекты и выбирать их во время работы программы. Паттерн GoF реализован на Python, C++23, Rust и Java и дополнен практическими вариантами и заданиями.

© Примак А., 2026
© Автор, 2026

Содержание

Введение 5
Как читать 6
Глава 21. Реализация на Python 7
21.1. Зачем нужна стратегия 7
21.2. Участники и схема 8
21.3. Классический пример: стоимость доставки 10
Конец ознакомительного фрагмента. 12

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

Введение

Эта книга посвящена одному паттерну проектирования — **Стратегия**(Strategy). 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 21. Реализация на Python

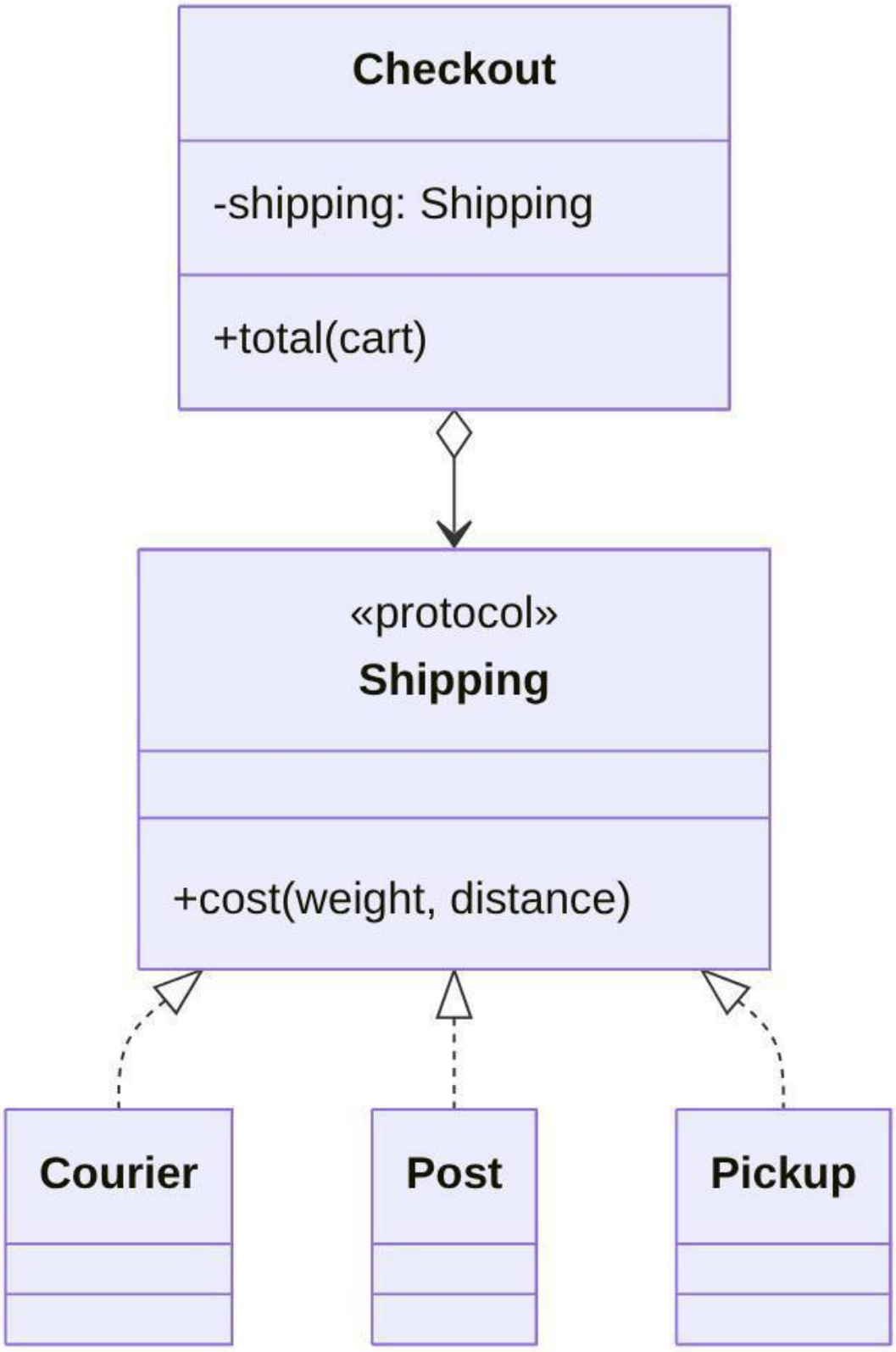
21.1. Зачем нужна стратегия

Навигатор строит маршрут для автомобиля, пешехода и велосипедиста. Магазин считает доставку курьером, почтой или самовывозом. Сервис сжимает файлы разными алгоритмами. Во всех случаях задача одна, а способов её решить несколько, и выбирают их во время работы.

Стратегия выносит каждый алгоритм в отдельный взаимозаменяемый объект с общим интерфейсом. Контекст хранит ссылку на стратегию и вызывает её, не зная, какая именно используется. Новый алгоритм — новая стратегия, без правки контекста.

В Python стратегия чаще всего — просто функция. Встроенные функции уже принимают стратегии: sorted(items, key=…), max(…, key=…), map(fn, …). Класс нужен, когда у стратегии есть настройки или несколько связанных методов.

21.2. Участники и схема



Strategy— общий интерфейс алгоритмов.
Concrete Strategy— конкретный алгоритм.
Context— использует стратегию через общий интерфейс.

21.3. Классический пример: стоимость доставки

Файл 01_shipping.py:

01_shipping.py — Python

```python
from dataclasses import dataclass
from typing import Protocol


class Shipping(Protocol):
    name: str

    def cost(self, weight_kg: float, distance_km: float) -> int:
        ...


@dataclass
class Courier:
    name: str = "курьер"
    per_km: int = 25

    def cost(self, weight_kg, distance_km):
        return 300 + self.per_km * round(distance_km) + (200 if
            weight_kg > 10 else 0)


@dataclass
class Post:
    name: str = "почта"

    def cost(self, weight_kg, distance_km):
        return 250 + 60 * max(1, round(weight_kg))


@dataclass
class Pickup:
    name: str = "самовывоз"

    def cost(self, weight_kg, distance_km):
```

Листинг 21.1 *— Классический пример: стоимость доставки*

01_shipping.py (продолжение) Python

```python
        return 0


class Checkout:
    def __init__(self, goods_total: int, weight: float, distance: float) -> None:
        self.goods_total, self.weight, self.distance = goods_total, weight, distance

    def total(self, shipping: Shipping) -> int:
        delivery = shipping.cost(self.weight, self.distance)
        if self.goods_total >= 5000 and shipping.name != "самовывоз":
            delivery //= 2  # скидка на доставку для крупных заказов
        return self.goods_total + delivery


if __name__ == "__main__":
    order = Checkout(goods_total=4200, weight=3.4, distance=12)
    options = [Courier(), Courier("экспресс", per_km=60), Post(), Pickup()]
    for s in options:
        print(f"{s.name:<10} {order.total(s):>5} ₽")
    best = min(options, key=order.total)
    print("Выгоднее всего:", best.name)
    big = Checkout(goods_total=6000, weight=12, distance=12)
    print("Крупный заказ курьером:", big.total(Courier()), "₽")
```

```
$ python 01_shipping.py
курьер      4800 ₽
экспресс    5220 ₽
почта       4630 ₽
самовывоз   4200 ₽
Выгоднее всего: самовывоз
Крупный заказ курьером: 6400 ₽
```

Листинг 21.1 (продолжение) *— Классический пример: стоимость доставки*

Разбор. Checkout не знает формул доставки — только интерфейс cost. Стратегии настраиваются параметрами: «экспресс» — тот же курьер с другим тарифом. Так как стратегии взаимозаменяемы, их можно перебрать и выбрать лучшую одной строкой с min(…, key=…) — здесь сама функция order.total стала стратегией сравнения.

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.