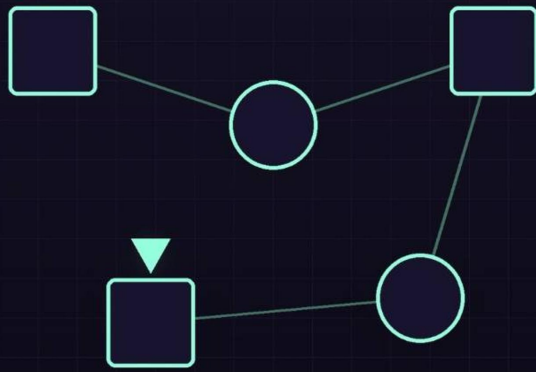
GoF · 23/23 ПОВЕДЕНЧЕСКИЙ

Python C++23 Rust Java



Посетитель

Visitor

Алексей Примак

Алексей Примак

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

https://litres.ru/74810682
SelfPub; 2026

Аннотация

Как добавлять множество операций к устойчивой структуре объектов, не раздувая каждый класс новыми методами? «Посетитель» разбирает этот вопрос на примере дерева документа, где над абзацами, заголовками, изображениями и таблицами выполняются разные операции. Паттерн GoF реализован на Python, C++23, Rust и Java. В книге рассматриваются структура посетителя, практические варианты, модификации и самостоятельные задания.

Содержание

Введение 4
Как читать 5
Глава 23. Реализация на Python 6
23.1. Зачем нужен посетитель 6
23.2. Участники и схема 8
23.3. Классический пример: документ 10
Конец ознакомительного фрагмента. 15

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

Введение

Эта книга посвящена одному паттерну проектирования — **Посетитель**(Visitor). 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 23. Реализация на Python

23.1. Зачем нужен посетитель

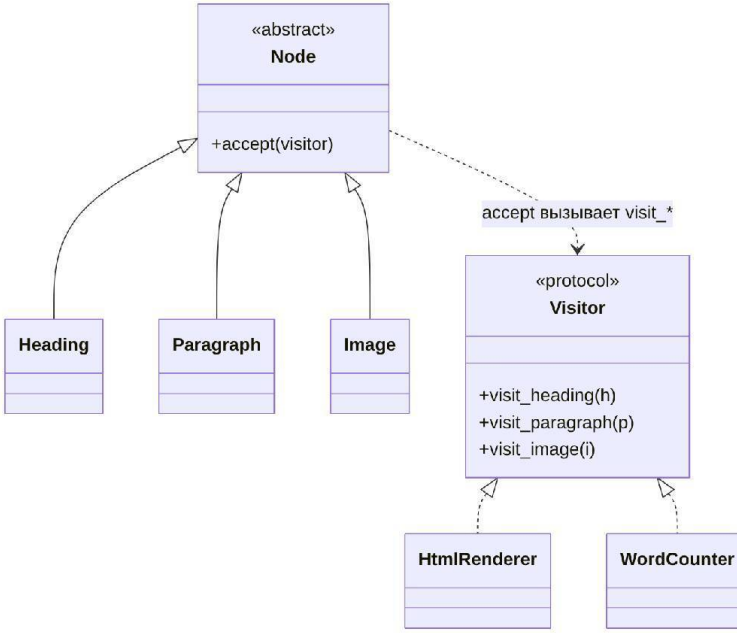
Есть дерево документа: абзацы, заголовки, картинки, таблицы. Над ним нужно выполнять много разных операций: отрисовать в HTML, в Markdown, посчитать слова, проверить орфографию, собрать оглавление. Если добавлять каждую операцию методом в каждый класс узла, классы разрастаются, а логика одной операции размазывается по десятку файлов.

Посетитель выносит операцию в отдельный объект. У посетителя есть по методу на каждый тип узла, а узлы умеют «принять» посетителя и вызвать нужный метод. Новая операция — новый посетитель, без изменения классов узлов. Обратная сторона: новый тип узла требует правки всех посетителей. Поэтому паттерн хорош, когда набор типов стабилен, а операции добавляются часто — как в компиляторах, где типы узлов дерева разбора меняются редко.

Классическая реализация опирается на **двойную диспетчеризацию**: node.accept(visitor) выбирает метод по типу узла, а внутри вызывается visitor.visit_paragraph(self). В Python есть и более лёгкие способы: выбор метода по имени класса (так устроен ast.NodeVisitor), functools.singledispatch

и сопоставление с образцом match.

23.2. Участники и схема



Visitor— объявляет метод посещения для каждого типа элемента.

Concrete Visitor— реализует одну операцию для всех ти-

пов.

Element— метод accept(visitor).

Object Structure— коллекция элементов, которую обходят.

23.3. Классический пример: документ

Файл 01_document.py:

01_document.py Python

```python
from __future__ import annotations

from dataclasses import dataclass
from html import escape


@dataclass
class Heading:
    level: int
    text: str

    def accept(self, v):
        return v.visit_heading(self)


@dataclass
class Paragraph:
    text: str

    def accept(self, v):
        return v.visit_paragraph(self)


@dataclass
class Image:
    src: str
    alt: str

    def accept(self, v):
        return v.visit_image(self)


class HtmlRenderer:
    def visit_heading(self, h):
```

Листинг 23.1 *— Классический пример: документ*

01_document.py (продолжение) Python

```python
        return f"<h{h.level}>{escape(h.text)}</h{h.level}>"

    def visit_paragraph(self, p):
        return f"<p>{escape(p.text)}</p>"

    def visit_image(self, i):
        return f'<img src="{i.src}" alt="{escape(i.alt)}">'


class WordCounter:
    def __init__(self) -> None:
        self.total = 0

    def visit_heading(self, h):
        self.total += len(h.text.split())

    def visit_paragraph(self, p):
        self.total += len(p.text.split())

    def visit_image(self, i):
        pass  # подпись к картинке не считаем


class TocBuilder:
    def __init__(self) -> None:
        self.items: list[str] = []

    def visit_heading(self, h):
        self.items.append("  " * (h.level - 1) + "- " + h.text)

    def visit_paragraph(self, p):
        pass

    def visit_image(self, i):
```

Листинг 23.1 (продолжение) *— Классический пример: документ*

01_document.py (продолжение) Python

```
        pass


if __name__ == "__main__":
    doc = [
        Heading(1, "Паттерны в Python"),
        Paragraph("Посетитель отделяет операции от структуры."),
        Heading(2, "Двойная диспетчеризация"),
        Paragraph("Метод выбирается по типу узла & посетителя."),
        Image("visitor.png", "Схема <посетителя>"),
    ]
    print("\n".join(node.accept(HtmlRenderer()) for node in doc))
    wc, toc = WordCounter(), TocBuilder()
    for node in doc:
        node.accept(wc)
        node.accept(toc)
    print("Слов:", wc.total)
    print("Оглавление:", *toc.items, sep="\n")
```

```
$ python 01_document.py
<h1>Паттерны в Python</h1>
<p>Посетитель отделяет операции от структуры.</p>
<h2>Двойная диспетчеризация</h2>
<p>Метод выбирается по типу узла &amp;amp; посетителя.</p>
<img src="visitor.png" alt="Схема &amp;lt;посетителя&amp;gt;">
Слов: 17
Оглавление:
- Паттерны в Python
  - Двойная диспетчеризация
```

Листинг 23.1 (продолжение) *— Классический пример: документ*

Разбор. Классы узлов содержат только данные и одну строку accept. Три операции — HTML, подсчёт слов, оглавление — живут в трёх посетителях, и каждая собрана в одном месте. Четвёртая операция, например экспорт в Markdown, — ещё один класс, а узлы не меняются. Функция escape защищает HTML от спецсимволов в тексте.

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.