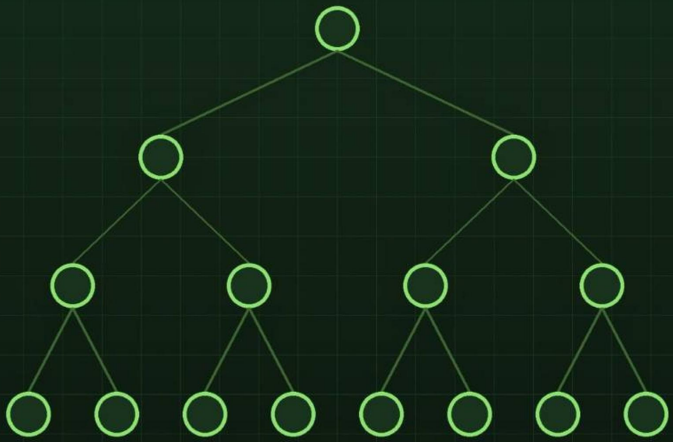
GoF · 08/23

СТРУКТУРНЫЙ

Python C++23 Rust Java



Компоновщик

Composite

Алексей Примак

Алексей Примак

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

https://litres.ru/74810752
SelfPub; 2026

Аннотация

Файлы и папки, меню и подменю, товары и наборы — всё это структуры, похожие на деревья. «Компоновщик» показывает, как обращаться с отдельным объектом и группой объектов через единый интерфейс. Паттерн GoF разобран на практических примерах и реализован на Python, C++23, Rust и Java. В книге рассмотрены классическая структура, варианты обхода и работы с деревьями, модификации и задания.

Содержание

Введение 4
Как читать 5
Глава 8. Реализация на Python 6
8.1. Зачем нужен компоновщик 6
8.2. Участники и схема 7
8.3. Классический пример: файловая система 10
Конец ознакомительного фрагмента. 14

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

Введение

Эта книга посвящена одному паттерну проектирования — **Компоновщик**(Composite). 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 8. Реализация на Python

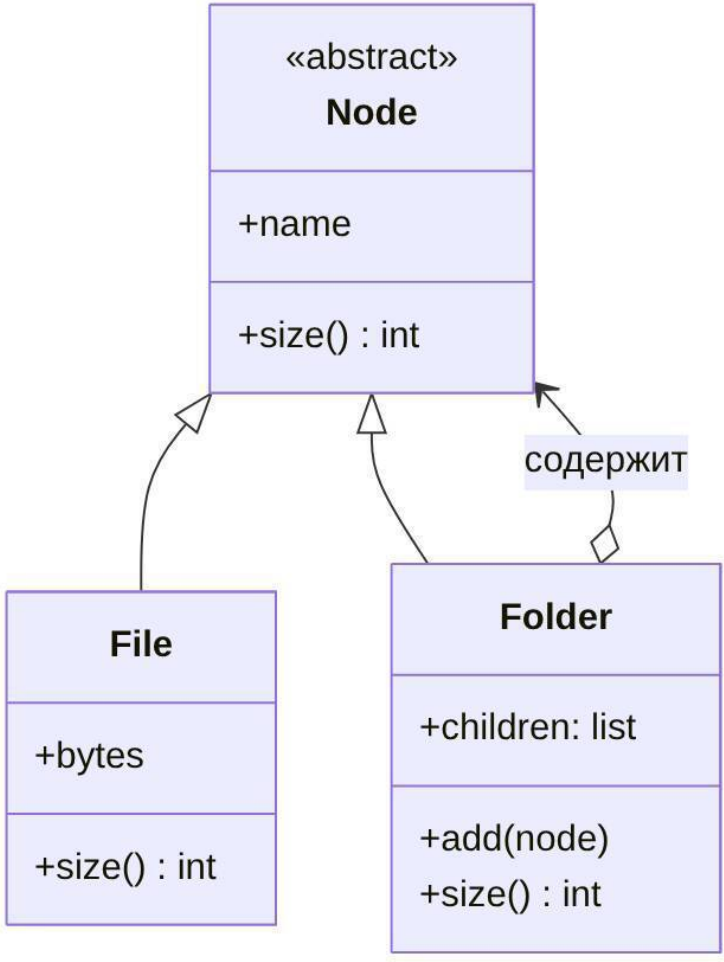
8.1. Зачем нужен компоновщик

Файловая система состоит из файлов и папок, а папки содержат файлы и другие папки. Меню состоит из пунктов и подменю. Заказ — из товаров и наборов товаров. Во всех случаях мы имеем дерево, и хочется обращаться с одиночным элементом и с группой одинаково: «посчитай размер», «покажи», «найди».

Компоновщик объединяет объекты в древовидные структуры и даёт листьям и контейнерам общий интерфейс. Клиент вызывает size() у корня и не думает, сколько там уровней вложенности: контейнер сам передаст вызов детям и сложит результаты.

В Python компоновщик естественно сочетается с рекурсией, генераторами (yield from) и протоколами контейнеров (__iter__, __len__).

8.2. Участники и схема


«abstract»
Node
+name
+size() : int
содержит
File
+bytes
+size() : int
Folder
+children: list
+add(node)
+size() : int

Component— общий интерфейс листьев и контейнеров.
Leaf— элемент без детей, делает реальную работу.
Composite— хранит детей и делегирует им операции.

8.3. Классический пример: файловая система

Файл 01_filesystem.py:

01_filesystem.py Python

```python
from __future__ import annotations

from abc import ABC, abstractmethod
from typing import Iterator


class Node(ABC):
    def __init__(self, name: str) -> None:
        self.name = name

    @abstractmethod
    def size(self) -> int: ...

    @abstractmethod
    def show(self, indent: int = 0) -> Iterator[str]: ...


class File(Node):
    def __init__(self, name: str, size: int) -> None:
        super().__init__(name)
        self._size = size

    def size(self) -> int:
        return self._size

    def show(self, indent: int = 0) -> Iterator[str]:
        yield f"{'  ' * indent}{self.name} ({self._size} Б)"


class Folder(Node):
    def __init__(self, name: str, *children: Node) -> None:
        super().__init__(name)
        self.children: list[Node] = list(children)

```

Листинг 8.1 *— Классический пример: файловая система*

01_filesystem.py (продолжение) Python

```python
    def add(self, node: Node) -> Folder:
        self.children.append(node)
        return self

    def size(self) -> int:
        return sum(child.size() for child in self.children)

    def show(self, indent: int = 0) -> Iterator[str]:
        yield f"{'  ' * indent}{self.name}/ [{self.size()} Б]"
        for child in self.children:
            yield from child.show(indent + 1)


if __name__ == "__main__":
    root = Folder(
        "project",
        File("README.md", 1200),
        Folder("src", File("main.py", 3400), File("util.py", 800
            ), Folder("tests", File("test_main.py", 950))),
        Folder("empty"),
    )
    root.add(File("LICENSE", 1070))
    print("\n".join(root.show()))
    print("Только файл:", File("x.txt", 5).size())
```

```
$ python 01_filesystem.py
project/ [7420 Б]
  README.md (1200 Б)
  src/ [5150 Б]
    main.py (3400 Б)
    util.py (800 Б)
    tests/ [950 Б]
      test_main.py (950 Б)
  empty/ [0 Б]
  LICENSE (1070 Б)
Только файл: 5
```

Листинг 8.1 (продолжение) *— Классический пример: файловая система*

Разбор. Folder.size складывает размеры детей, не зная, файлы это или папки: каждый ребёнок сам умеет отвечать. Пустая папка корректно даёт 0 — sum от пустого генератора равен нулю. show — генератор, и yield from передаёт наверх строки из поддеревьев любой глубины. add возвращает self, поэтому вызовы можно сцеплять.

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.