GoF · 14/23 ПОВЕДЕНЧЕСКИЙ

Python C++23 Rust Java



Команда

Command

Алексей Примак

Алексей Примак

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

«Автор»
2026

Примак А.

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

Как превратить действие в объект, чтобы его можно было отменять, повторять, ставить в очередь или выполнять позже? «Команда» разбирает этот подход на примере текстового редактора и других практических сценариев. Паттерн GoF реализован на Python, C++23, Rust и Java. В книге рассматриваются участники схемы, работа с историей действий, очередями и расширениями, а также практические и самостоятельные задания.

© Примак А., 2026
© Автор, 2026

Содержание

Введение 5
Как читать 6
Глава 14. Реализация на Python 7
14.1. Зачем нужна команда 7
14.2. Участники и схема 8
14.3. Классический пример: редактор с отменой 11
Конец ознакомительного фрагмента. 14

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

Введение

Эта книга посвящена одному паттерну проектирования — **Команда**(Command). 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 14. Реализация на Python

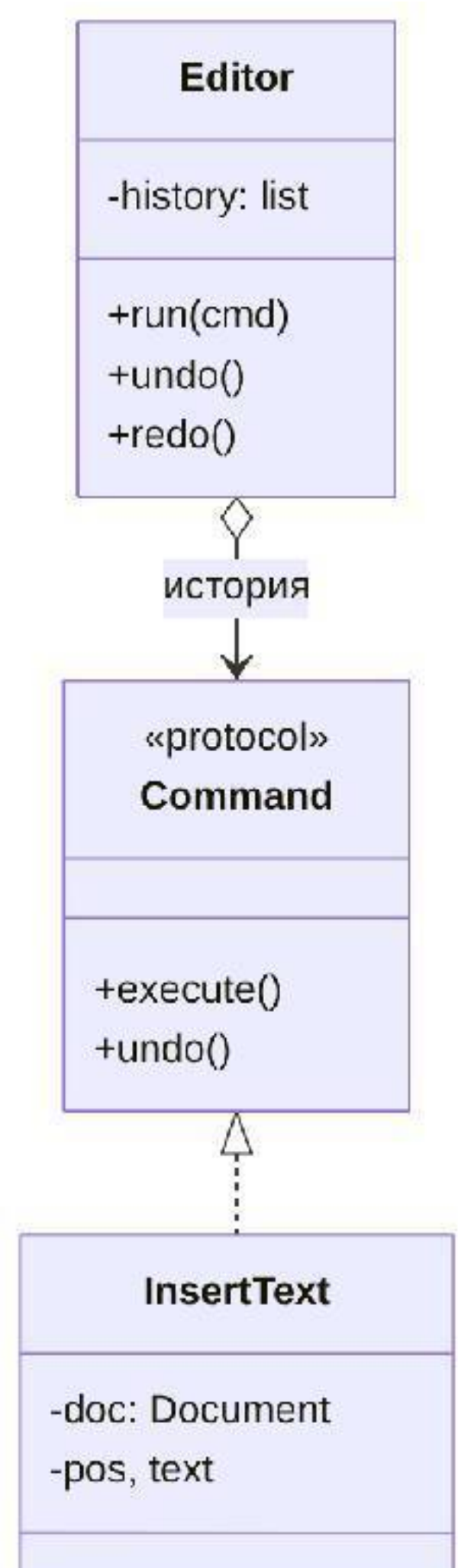
14.1. Зачем нужна команда

В текстовом редакторе пользователь печатает, удаляет и форматирует, а потом нажимает «Отменить» — и последнее действие откатывается. Затем «Повторить». Кнопка панели, пункт меню и сочетание клавиш выполняют одно и то же действие. Задачи откладывают в очередь, чтобы выполнить позже или в другом потоке. Всё это возможно, если превратить действие в объект.

Команда инкапсулирует запрос как объект: что сделать, над чем и с какими параметрами. Такой объект можно сохранить в историю, поставить в очередь, передать по сети, отменить, выполнить повторно или объединить с другими в макрокоманду.

В Python любая функция уже объект, поэтому простейшая команда — это функция или functools.partial. Класс нужен, когда у команды есть дополнительное поведение: отмена, описание, сериализация.

14.2. Участники и схема


Editor
-history: list
+run(cmd)
+undo()
+redo()
история
«protocol»
Command
+execute()
+undo()
InsertText
-doc: Document
-pos, text

Command— интерфейс выполнения (и отмены).
Concrete Command— связывает получателя и действие.
Receiver— объект, который выполняет реальную работу.
Invoker— инициирует выполнение и хранит историю.
Client— создаёт команды и настраивает получателей.

14.3. Классический пример: редактор с отменой

Файл 01_editor.py:

01_editor.py — Python

```python
from typing import Protocol


class Document:
    def __init__(self) -> None:
        self.text = ""


class Command(Protocol):
    def execute(self) -> None: ...
    def undo(self) -> None: ...


class Insert:
    def __init__(self, doc: Document, pos: int, s: str) -> None:
        self.doc, self.pos, self.s = doc, pos, s

    def execute(self) -> None:
        t = self.doc.text
        self.doc.text = t[: self.pos] + self.s + t[self.pos:]

    def undo(self) -> None:
        t = self.doc.text
        self.doc.text = t[: self.pos] + t[self.pos + len(self.s)
            :]


class Delete:
    def __init__(self, doc: Document, pos: int, n: int) -> None:
        self.doc, self.pos, self.n = doc, pos, n
        self.removed = ""

    def execute(self) -> None:
        t = self.doc.text
```

Листинг 14.1 *— Классический пример: редактор с отменой*

01_editor.py (продолжение) Python

```
        self.removed = t[self.pos: self.pos + self.n]
        self.doc.text = t[: self.pos] + t[self.pos + self.n:]

    def undo(self) -> None:
        t = self.doc.text
        self.doc.text = t[: self.pos] + self.removed + t[self.
            pos:]


class Editor:
    def __init__(self, doc: Document) -> None:
        self.doc = doc
        self.done: list[Command] = []
        self.undone: list[Command] = []

    def run(self, cmd: Command) -> None:
        cmd.execute()
        self.done.append(cmd)
        self.undone.clear()  # новое действие обнуляет «повтор»

    def undo(self) -> None:
        if self.done:
            cmd = self.done.pop()
            cmd.undo()
            self.undone.append(cmd)

    def redo(self) -> None:
        if self.undone:
            cmd = self.undone.pop()
            cmd.execute()
            self.done.append(cmd)


if __name__ == "__main__":
```

Листинг 14.1 (продолжение) *— Классический пример: редактор с отменой*

01_editor.py (продолжение) Python

```python
    doc = Document()
    ed = Editor(doc)
    ed.run(Insert(doc, 0, "Привет мир"))
    ed.run(Insert(doc, 6, ","))
    ed.run(Delete(doc, 0, 7))
    print(repr(doc.text))
    ed.undo()
    print(repr(doc.text))
    ed.undo()
    print(repr(doc.text))
    ed.redo()
    print(repr(doc.text))
    ed.run(Insert(doc, len(doc.text), "!"))
    ed.redo()  # ничего не делает: история повторов сброшена
    print(repr(doc.text), "| в истории:", len(ed.done))
```

```
$ python 01_editor.py
' мир'
'Привет, мир'
'Привет мир'
'Привет, мир'
'Привет, мир!' | в истории: 3
```

Листинг 14.1 (продолжение) *— Классический пример: редактор с отменой*

Разбор. Каждая команда знает, как отменить себя. Delete запоминает удалённый текст при выполнении — без этого отмена невозможна. Редактор хранит два стека: выполненные и отменённые команды. Новое действие очищает стек повторов, иначе история стала бы противоречивой. Сам редактор ничего не знает о вставках и удалениях — он работает с любыми объектами, у которых есть execute и undo.

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.