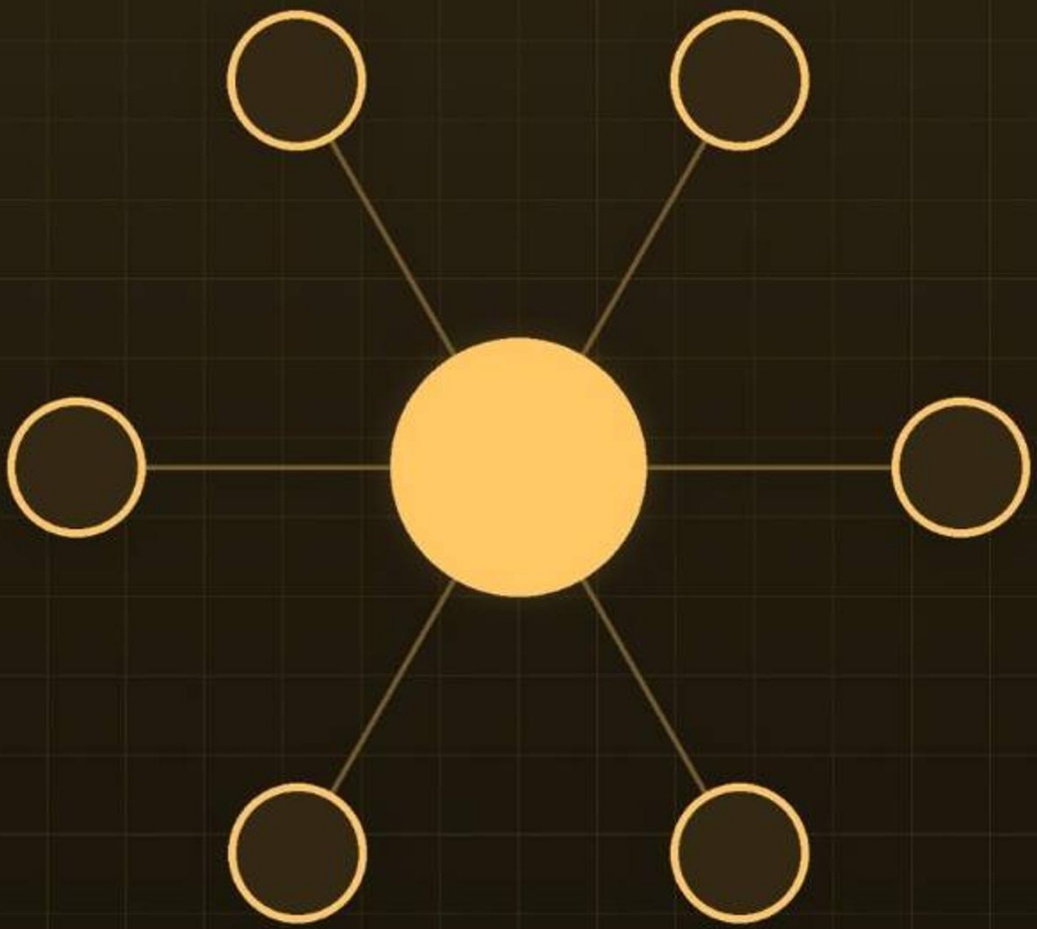
GoF · 17/23

ПОВЕДЕНЧЕСКИЙ

Python C++23 Rust Java



Посредник

Mediator

Алексей Примак

Алексей Примак

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

«Автор»
2026

Примак А.

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

Когда элементы интерфейса начинают напрямую управлять друг другом, система быстро превращается в паутину зависимостей. «Посредник» показывает, как вынести координацию взаимодействия в отдельный объект. На примере формы регистрации разобраны роли участников и способы уменьшить связанность. Паттерн GoF реализован на Python, C++23, Rust и Java, дополнен практическими вариантами и самостоятельными заданиями.

© Примак А., 2026
© Автор, 2026

Содержание

Введение 5
Как читать 6
Глава 17. Реализация на Python 7
17.1. Зачем нужен посредник 7
17.2. Участники и схема 8
17.3. Классический пример: форма регистрации 11
Конец ознакомительного фрагмента. 14

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

Введение

Эта книга посвящена одному паттерну проектирования — **Посредник**(Mediator). 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 17. Реализация на Python

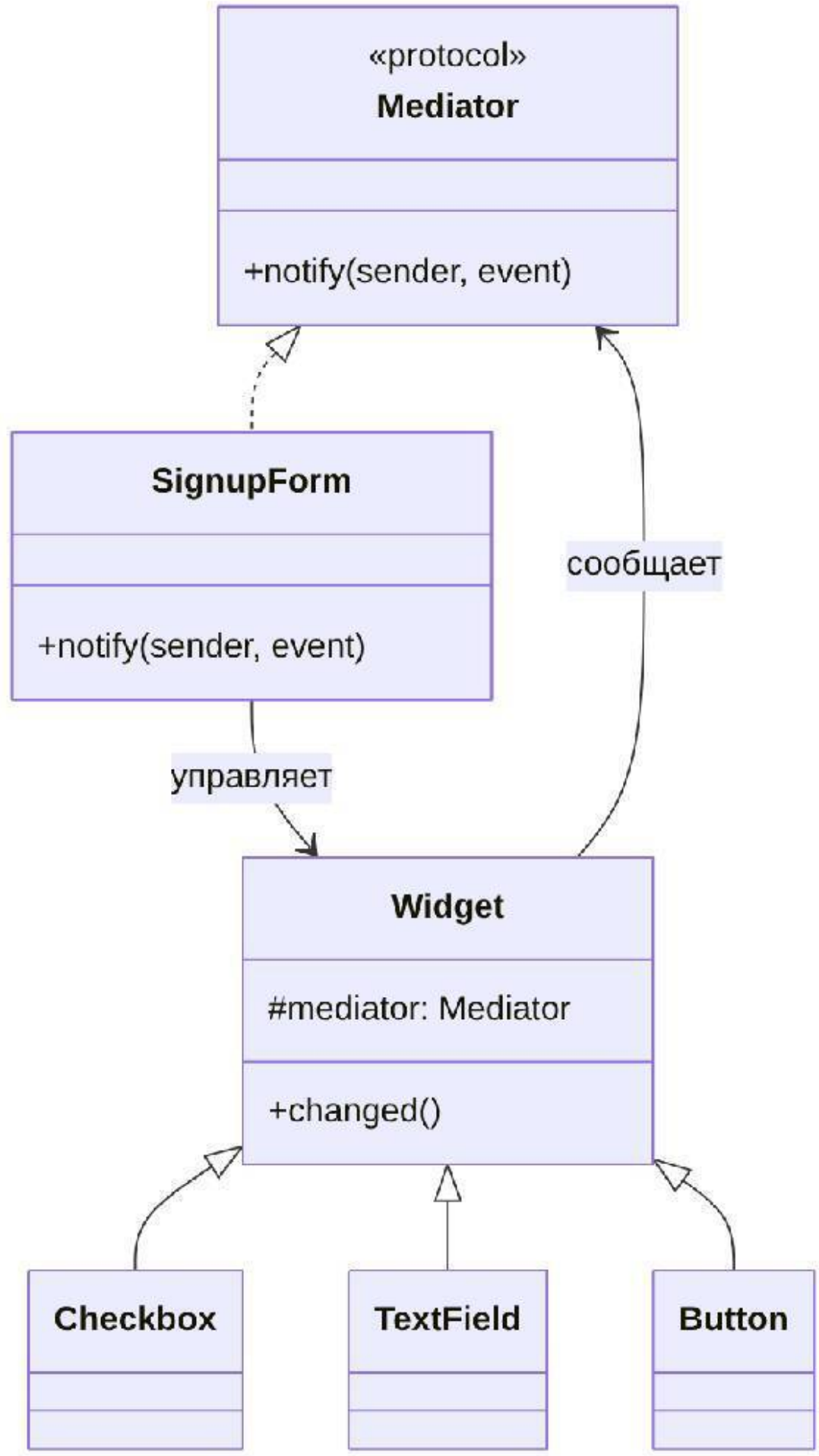
17.1. Зачем нужен посредник

Форма регистрации: при выборе «юридическое лицо» появляется поле ИНН, кнопка «Отправить» активна только при заполненных полях и отмеченном согласии, а смена страны меняет маску телефона. Если каждый элемент знает о других и сам их меняет, получается паутина связей: десять элементов — до девяноста зависимостей. Любое изменение ломает что-нибудь в неожиданном месте.

Посредник забирает эти связи себе. Элементы (коллеги) сообщают посреднику о событиях — «меня изменили», — а посредник решает, кого и как обновить. Коллеги знают только посредника, а не друг друга. Схема «многие ко многим» превращается в «многие к одному».

Знакомые примеры посредников — диспетчерская вышка аэропорта (самолёты не договариваются друг с другом напрямую), чат-комната, шина событий, контроллер в архитектуре MVC.

17.2. Участники и схема


«protocol»
Mediator
+notify(sender, event)
SignupForm
+notify(sender, event)
управляет
сообщает
Widget
#mediator: Mediator
+changed()
Checkbox
TextField
Button

Mediator— интерфейс для общения коллег.
Concrete Mediator— знает всех коллег и реализует их взаимодействие.
Colleague— компонент, который сообщает посреднику о событиях.

17.3. Классический пример: форма регистрации

Файл 01_signup_form.py:

01_signup_form.py — Python

```python
from __future__ import annotations


class Widget:
    def __init__(self, name: str, form: SignupForm) -> None:
        self.name, self.form = name, form
        self.visible, self.enabled = True, True

    def changed(self) -> None:
        self.form.notify(self)


class Field(Widget):
    def __init__(self, name: str, form: SignupForm) -> None:
        super().__init__(name, form)
        self.value = ""

    def type(self, text: str) -> None:
        self.value = text
        self.changed()


class Check(Widget):
    def __init__(self, name: str, form: SignupForm) -> None:
        super().__init__(name, form)
        self.checked = False

    def toggle(self) -> None:
        self.checked = not self.checked
        self.changed()


class SignupForm:
    def __init__(self) -> None:
```

Листинг 17.1 *— Классический пример: форма регистрации*

01_signup_form.py (продолжение) Python

```python
        self.email = Field("email", self)
        self.company = Check("юрлицо", self)
        self.inn = Field("ИНН", self)
        self.agree = Check("согласие", self)
        self.submit = Widget("Отправить", self)
        self.notify(None)  # начальное состояние

    def notify(self, sender: Widget | None) -> None:
        self.inn.visible = self.company.checked
        inn_ok = not self.company.checked or len(self.inn.value)
            in (10, 12)
        self.submit.enabled = "@" in self.email.value and inn_ok
            and self.agree.checked

    def state(self) -> str:
        inn = "ИНН виден" if self.inn.visible else "ИНН скрыт"
        return f"{inn}, кнопка {'активна' if self.submit.enabled
            else 'неактивна'}"


if __name__ == "__main__":
    form = SignupForm()
    print("Начало:         ", form.state())
    form.email.type("anna@mail.ru")
    form.agree.toggle()
    print("Почта+согласие:", form.state())
    form.company.toggle()
    print("Юрлицо:         ", form.state())
    form.inn.type("7707083893")
    print("ИНН введён:     ", form.state())
    form.agree.toggle()
    print("Без согласия:   ", form.state())
```

```
$ python 01_signup_form.py
Начало:         ИНН скрыт, кнопка неактивна
Почта+согласие: ИНН скрыт, кнопка активна
Юрлицо:         ИНН виден, кнопка неактивна
ИНН введён:     ИНН виден, кнопка активна
Без согласия:   ИНН виден, кнопка неактивна
```

Листинг 17.1 (продолжение) *— Классический пример: форма регистрации*

Разбор. Поля и флажки ничего не знают друг о друге: при изменении они вызывают changed(), а форма-посредник пересчитывает всё состояние. Все правила видимости и доступности собраны в одном методе notify. Чтобы добавить правило «маска телефона зависит от страны», правится только посредник.

Посредник здесь пересчитывает всё при любом событии — для формы это проще и надёжнее, чем разбирать, кто именно изменился. Аргумент sender позволяет реагировать выборочно, когда пересчёт дорог.

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.