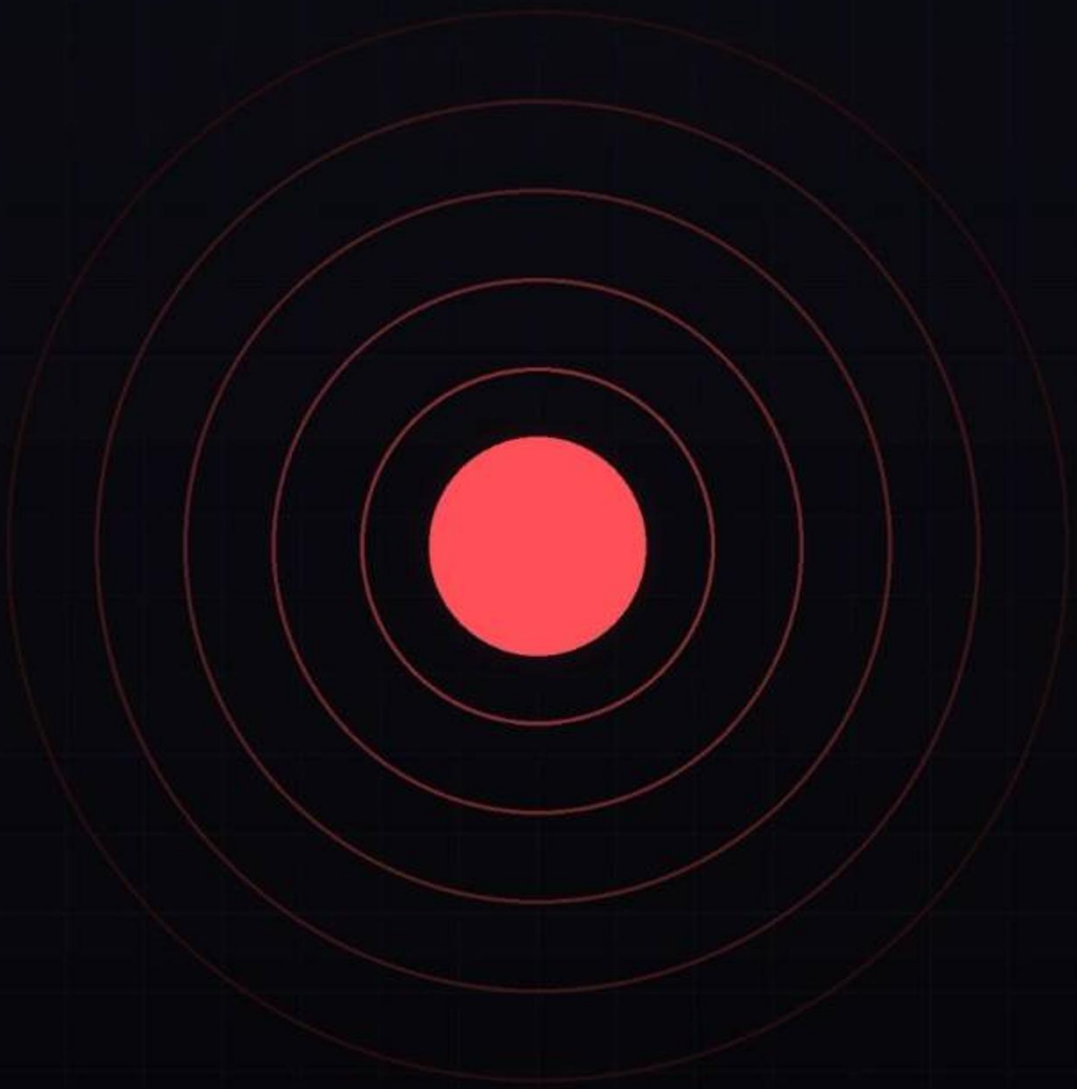
GoF · 05/23

ПОРОЖДАЮЩИЙ

Python C++23 Rust Java



Одиночка

Singleton

Алексей Примак

Алексей Примак

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

«Автор»
2026

Примак А.

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

Когда действительно нужен единственный экземпляр объекта и как организовать доступ к нему? «Одиночка» разбирает паттерн GoF на задачах конфигурации, журналирования и общих реестров. После теории идут реализации на Python, C++23, Rust и Java, а также практические варианты и ограничения подхода. Книга помогает понять не только структуру паттерна, но и ситуации, в которых его применение оправдано.

© Примак А., 2026
© Автор, 2026

Содержание

Введение 5
Как читать 6
Глава 5. Реализация на Python 7
5.1. Что ограничивает паттерн 7
5.2. Участники и схема 8
5.3. Модуль как одиночка 10
5.4. Классическая реализация через new 12
Конец ознакомительного фрагмента. 14

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

Введение

Эта книга посвящена одному паттерну проектирования — **Одиночка**(Singleton). 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 5. Реализация на Python

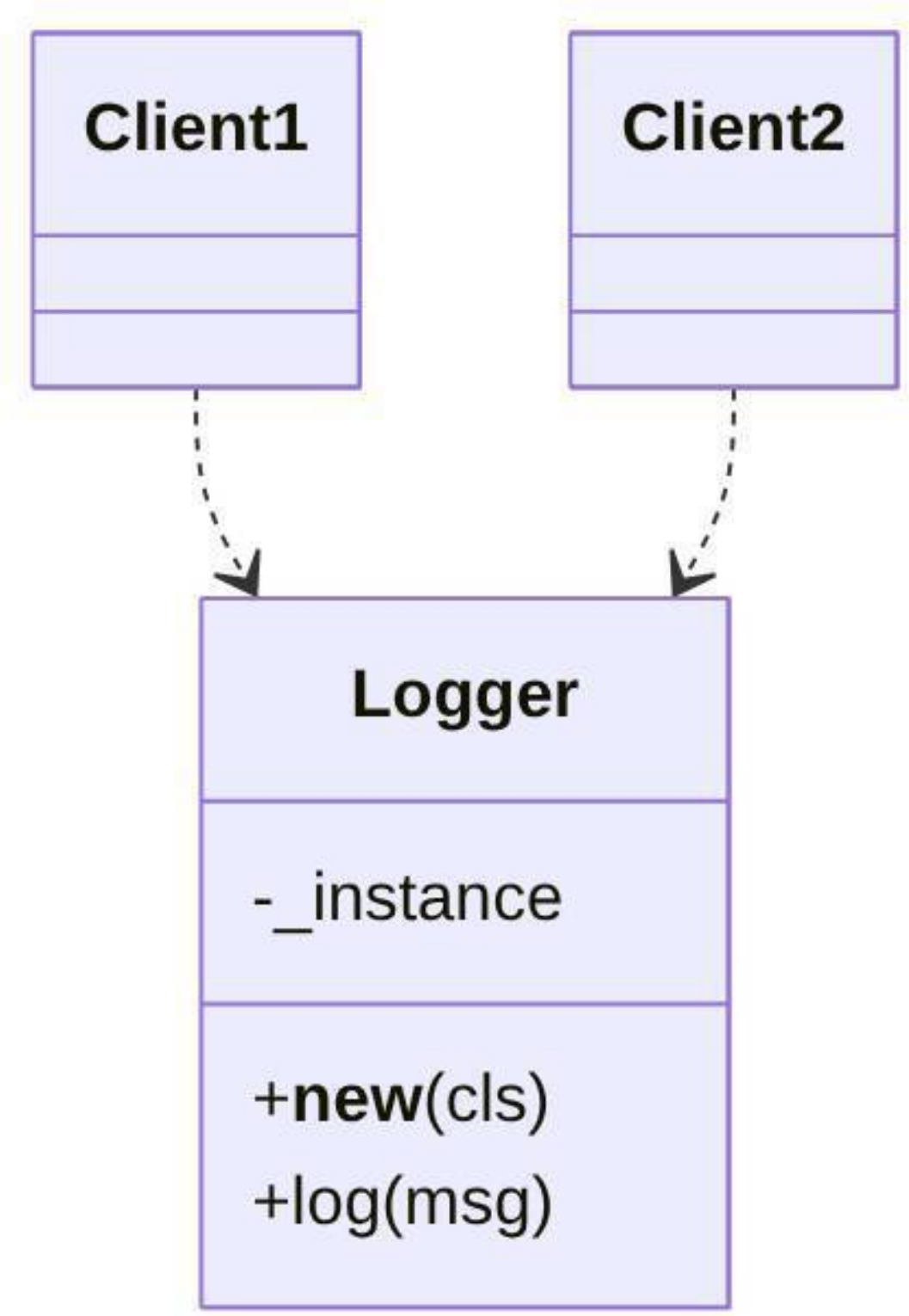
5.1. Что ограничивает паттерн

Одиночка гарантирует, что у класса есть ровно один экземпляр, и даёт к нему глобальную точку доступа: журнал событий, конфигурация, реестр плагинов, пул соединений.

Перед реализацией нужно определить **область единственности**: «один на процесс», «один на поток», «один на тест». И отделить два решения: единственность экземпляра — свойство задачи, а глобальный доступ — удобство, за которое платят тестируемостью.

В Python у одиночки есть особенность: **модуль уже является одиночкой**. Модуль выполняется один раз при первом импорте и кэшируется в sys.modules; все последующие import возвращают тот же объект. Поэтому в идиоматичном Python одиночку часто вообще не пишут как класс — достаточно объекта на уровне модуля. Классические реализации тоже существуют, и их полезно знать, чтобы понимать чужой код и выбирать осознанно.

5.2. Участники и схема



Singleton— класс с единственным экземпляром.

Точка доступа— конструктор, функция или атрибут модуля, всегда возвращающие один объект.

5.3. Модуль как одиночка

Файл 01_module.py:

01_module.py — Python

```python
import sys
import types

# Имитируем отдельный модуль settings.py внутри одного файла.
source = """
print("  (модуль settings выполняется)")
debug = False
workers = 4
_listeners = []

def set_debug(value):
    global debug
    debug = value
    for fn in _listeners:
        fn(value)
"""
settings = types.ModuleType("settings")
exec(source, settings.__dict__)
sys.modules["settings"] = settings

if __name__ == "__main__":
    import settings as s1
    import settings as s2

    print("Один и тот же объект:", s1 is s2)
    s1.set_debug(True)
    print("Изменение видно через второй импорт:", s2.debug)
    print("Потоков:", s2.workers)
```

```
$ python 01_module.py
  (модуль settings выполняется)
Один и тот же объект: True
Изменение видно через второй импорт: True
Потоков: 4
```

Листинг 5.1 *— Модуль как одиночка*

Разбор.Чтобы пример поместился в один файл, модуль settings создан вручную и записан в sys.modules — ровно так поступает механизм импорта. В реальном проекте это был бы обычный файл settings.py. Код модуля выполнился один раз, а оба import вернули один объект. Изменение, сделанное через одну ссылку, видно через другую.

Для большинства задач — конфигурация, реестры, общие ресурсы — этого достаточно. Модуль даёт единственность, ленивую загрузку (при первом импорте) и понятную точку доступа.

5.4. Классическая реализация через new

Если нужен именно класс — например, чтобы наследоваться или передавать объект туда, где ожидается экземпляр, — единственность обеспечивают в __new__.

Файл 02_new.py:

02_new.py — Python

```python
import threading


class Logger:
    _instance: "Logger | None" = None
    _lock = threading.Lock()

    def __new__(cls) -> "Logger":
        if cls._instance is None:
            with cls._lock:
                if cls._instance is None:  # двойная проверка
                    print("  (журнал создан)")
                    instance = super().__new__(cls)
                    instance.lines = []
                    instance._write_lock = threading.Lock()
                    cls._instance = instance
        return cls._instance

    def log(self, msg: str) -> None:
        with self._write_lock:
            self.lines.append(msg)


if __name__ == "__main__":
    threads = [threading.Thread(target=lambda i=i: Logger().log(
        f"поток {i}")) for i in range(8)]
    for t in threads:
        t.start()
    for t in threads:
        t.join()
    a, b = Logger(), Logger()
    print("Один экземпляр:", a is b)
    print("Записей:", len(a.lines))
```

```
$ python 02_new.py
  (журнал создан)
Один экземпляр: True
Записей: 8
```

Листинг 5.2*— Классическая реализация через __new__*

Разбор.__new__ создаёт объект, __init__ его настраивает. Переопределив __new__, класс возвращает уже существующий экземпляр. Инициализация атрибутов перенесена в __new__, а не в __init__: __init__ вызывался бы при каждом Logger() и сбрасывал бы список.

Двойная проверка: первая без блокировки — быстрый путь, когда экземпляр уже есть; вторая под блокировкой — защита от ситуации, когда два потока одновременно прошли первую проверку. Восемь потоков создали журнал ровно один раз.

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.