Hello, Turing!

today we will discuss:

Design Patterns

Speakers:

@Beknar Danabek

О чем поговорим

- Зависимость
- Почему зависимости это плохо?
- История
- Определение
- Польза шаблонов
- Квалификация
- Разберем несколько шаблонов

Зависимость

‘зависимый’

‘зависимость’

Почему зависимости это плохо?

Зависимости плохи тем, что снижают переиспользование.
Снижение переиспользования плохо по многим причинам.

Обычно переиспользование оказывает позитивное влияние на скорость разработки, качество кода, читаемость кода и т.д.

История

Кристофер Александер

Gang of Four

Определение

Паттерны проектирования — это часто встречающееся решение определённой проблемы при проектировании архитектуры программ.

Определение

Паттерны проектирования != Алгоритмы

Зачем знать паттерны?

- Проверенные решения
- Стандартизация кода
- Общий программистский словарь

Классификация паттернов

- Порождающие паттерны
- Структурные паттерны
- Поведенческие паттерны

Классификация паттернов

Порождающие паттерны беспокоятся о гибком создании объектов без внесения в программу лишних зависимостей

- Factory Method
- Builder
- Abstract Factory
- Prototype
- Singleton

Классификация паттернов

Структурные паттерны показывают различные способы построения связей между объектами.

- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy

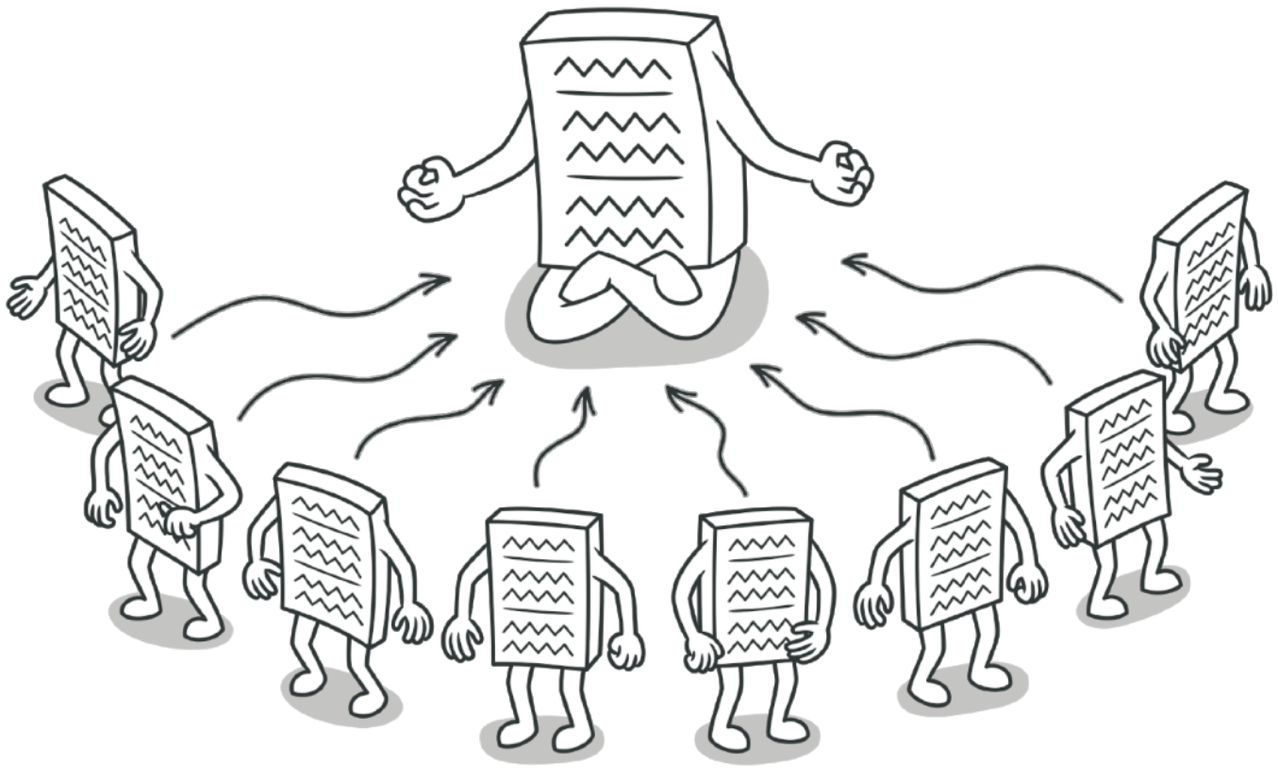
Классификация паттернов

Поведенческие паттерны заботятся об эффективной коммуникации между объектами.

- Chain of responsibility
- Command
- Iterator
- Mediator
- Memento
- Observer
- State
- Template Method
- Visitor

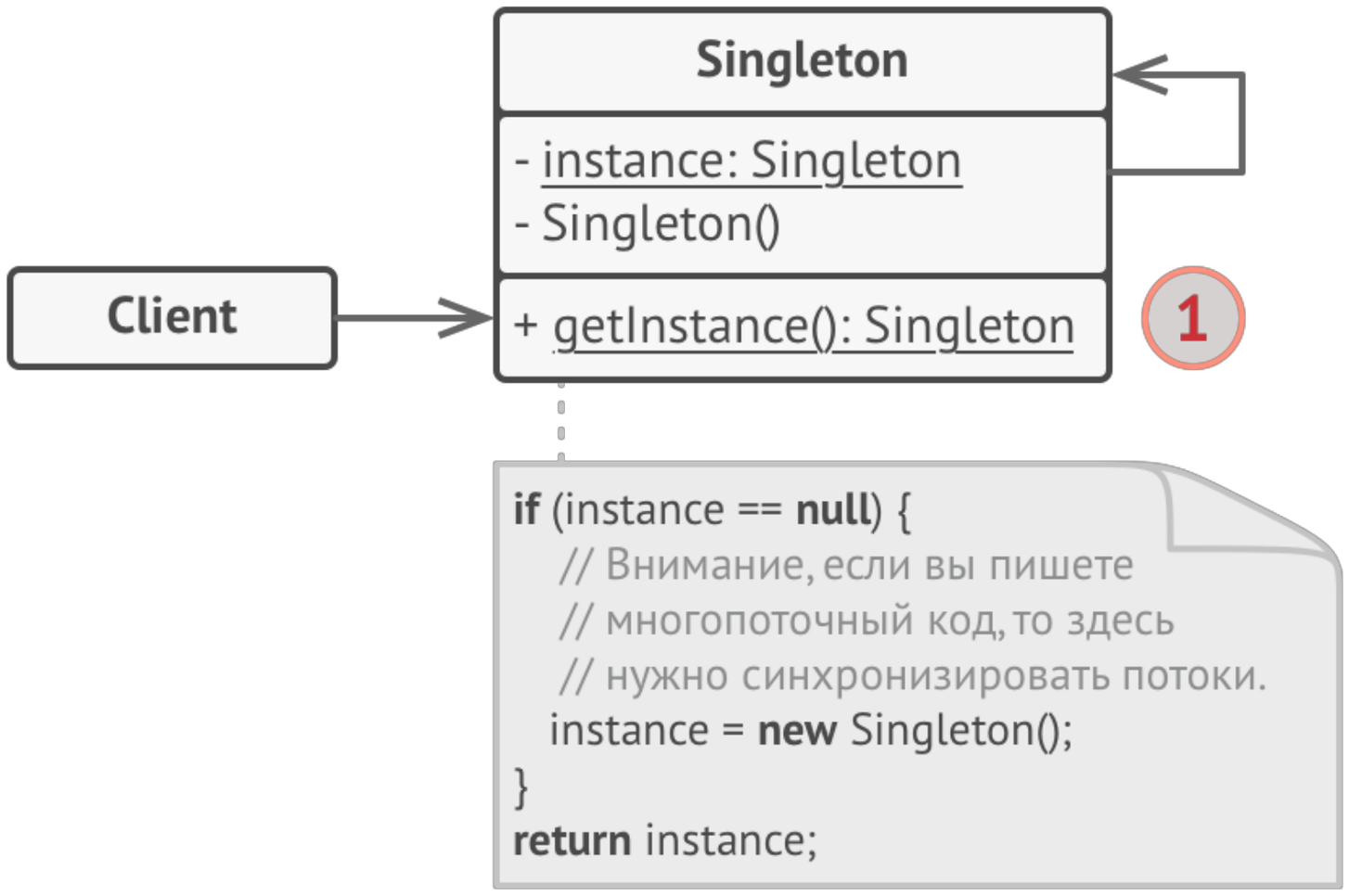
Рассмотрим некоторые шаблоны

Singleton



Одиночка (Singleton) — это порождающий паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.

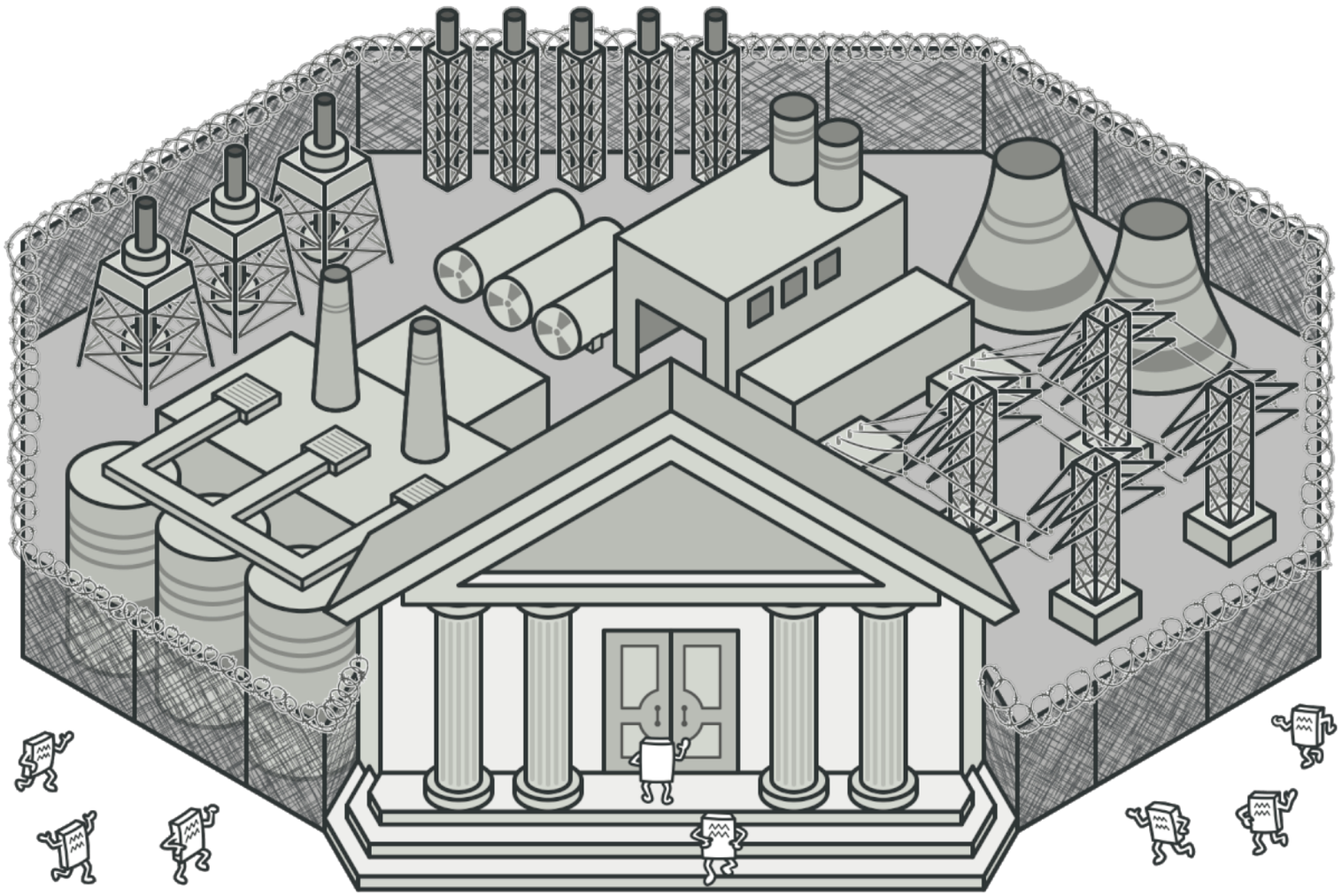
Структура Singleton



Одиночка определяет статический метод **getInstance**, который возвращает единственный экземпляр своего класса.

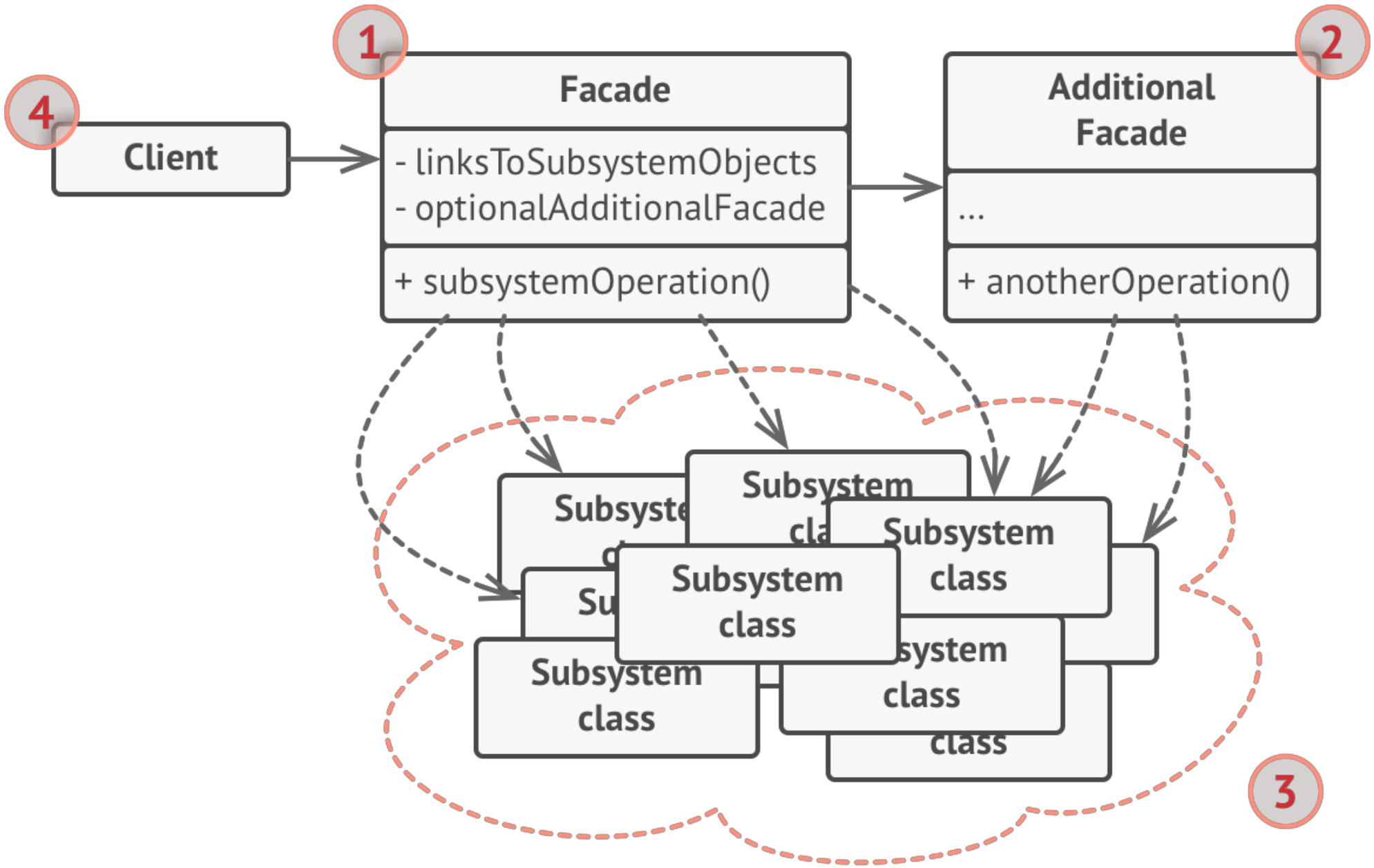
Конструктор одиночки должен быть скрыт от клиентов. Вызов метода getInstance должен стать единственным способом получить объект этого класса.

Facade

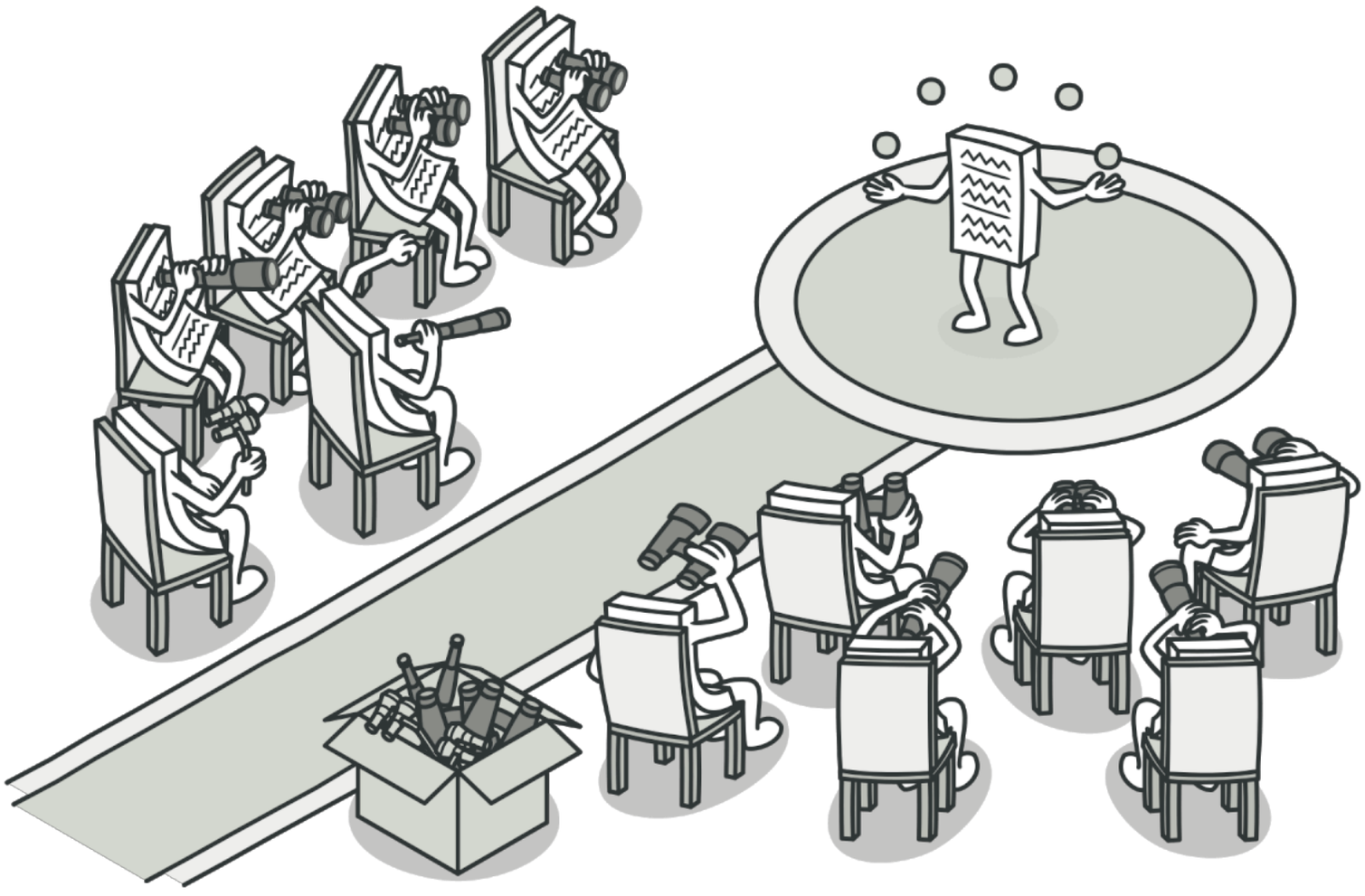


Фасад — это структурный паттерн проектирования, который предоставляет простой интерфейс к сложной системе классов, библиотеке или фреймворку.

Структура Facade

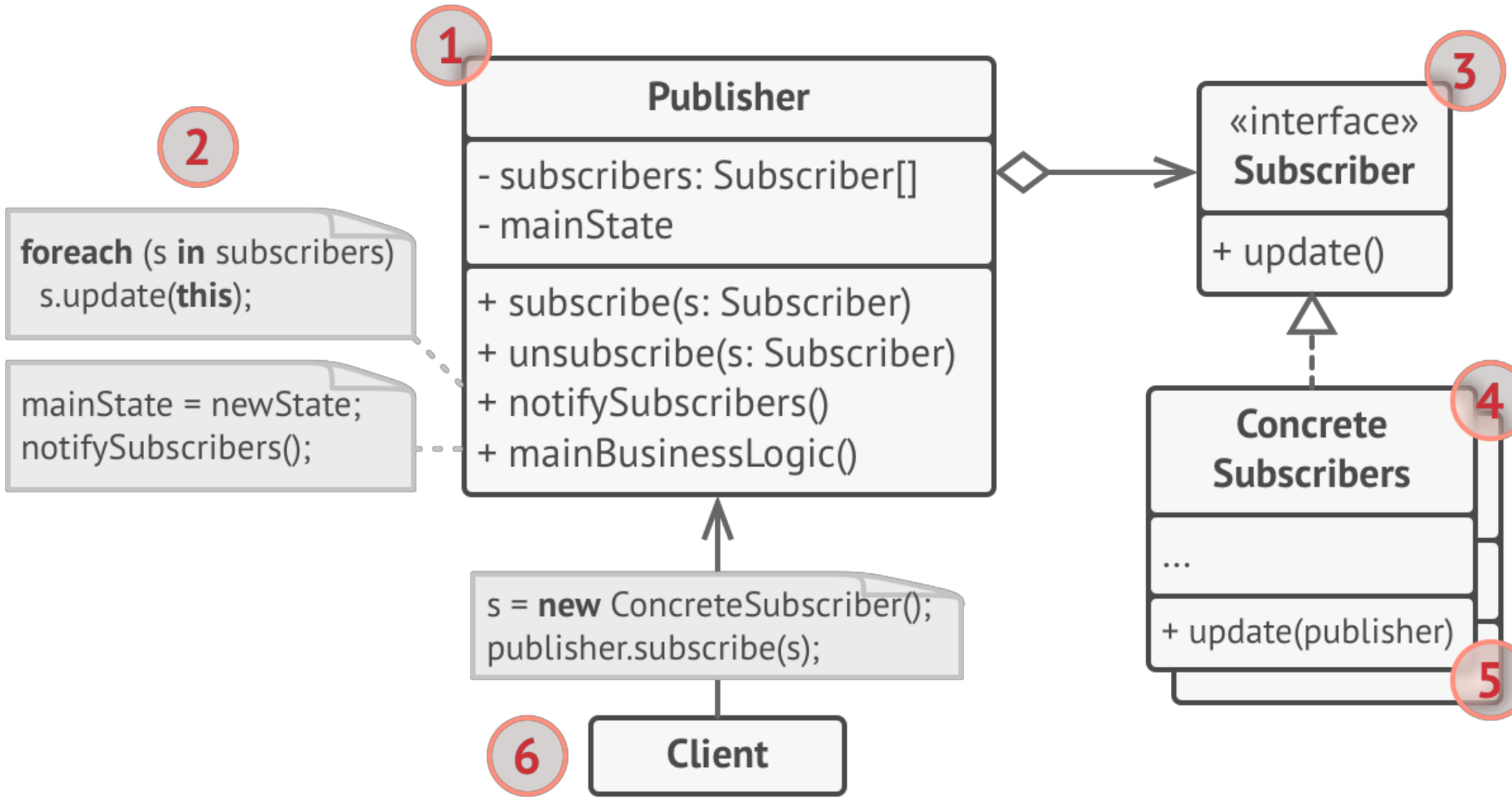


Observer



Наблюдатель (Observer) — это поведенческий паттерн проектирования, который создаёт механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах.

Структура Observer



Антипаттерны

Антипаттерны - шаблоны ошибок, которые совершаются при решении различных задач.

- The Blob
- Continuous Obsolescence
- Lava Flow
- Ambiguous Viewpoint
- Poltergeists
- Boat Anchor
- Golden Hammer
- Dead End
- Cut-and-Paste Programming
- и еще..

Мудрость

Чем больше паттернов я придумал засунуть в свое приложение - тем лучше

Подытожим

- Паттерны не серебряная пуля
- Все они имеют очень много общего с реальной жизнью и позволяют делать код насколько же простым для чтения и понимания, как и то, что мы видим в реальной жизни
- Используйте их с умом и только там, где они действительно нужны
- Паттерны устаревают, превращаются в анти-паттерны по мере развития технологий
- Главное не заболеть "шаблоном проектирования головного мозга"

Как понять шаблоны проектирования?

- Попробуйте для начала научиться просто "видеть" их в используемых библиотеках
- Постарайтесь осознать, доводилось ли вам сталкиваться в работе раньше с чем-то, что является или могло бы легко стать одним из шаблонов
- Используйте метафоры
- В новых проектах, держите в голове полученные по шаблонам знания
- **Все приходит с опытом**

Рекомендованные материалы

- SourceMaking
- Refactoring.Guru
- Design Patterns for humans
- Паттерны ООП в метафорах
- Design Patterns Implemented on Python
- Design Patterns Implemented on Java

- AntiPatterns on SourceMaking
- Антипаттерны на русском
- 9 анти-паттернов, о которых должен знать каждый программист

/danabeknar

/danabeknar

Thank you, Turing! 👋