Java Design Patterns

Singleton

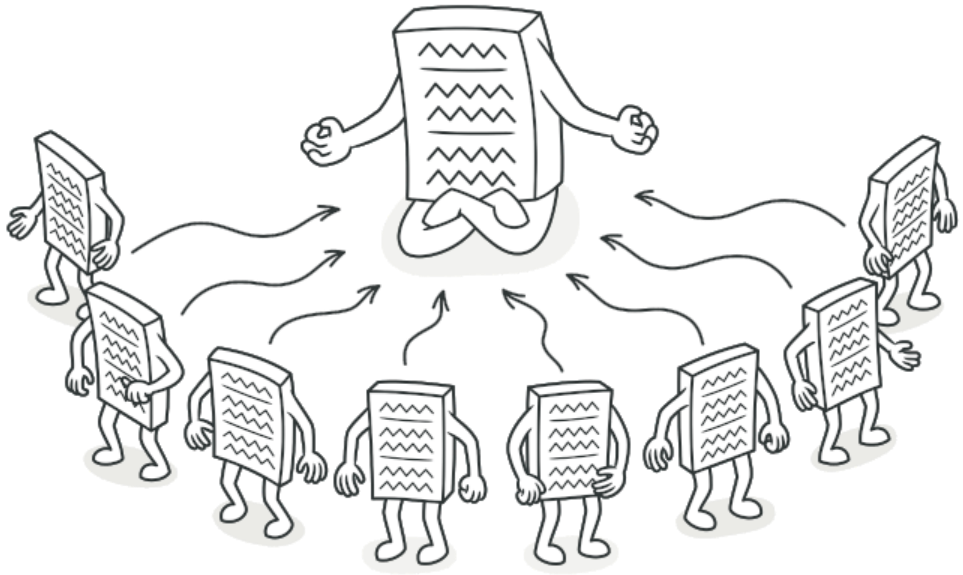
Тема

Паттерн Singleton

Суть паттерна

Singleton

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



Проблема и ее решение

Singleton

Одиночка решает сразу две проблемы:

- Гарантирует наличие единственного экземпляра класса
- Предоставляет глобальную точку доступа

Решение:

Все реализации одиночки сводятся к тому, чтобы скрыть конструктор по умолчанию и создать публичный статический метод, который и будет контролировать жизненный цикл объекта-одиночки.

Аналоги из жизни

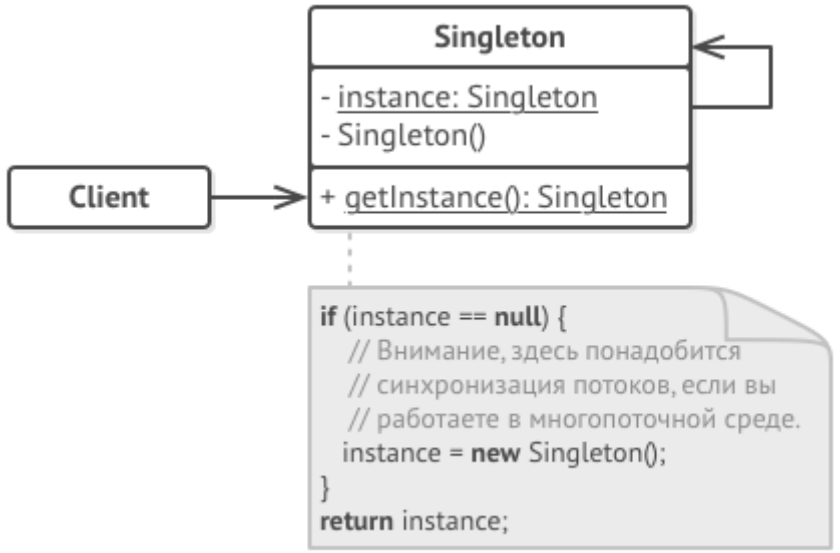
Singleton

Правительство государства — хороший пример одиночки. В государстве может быть только одно официальное правительство. Вне зависимости от того, кто конкретно заседает в правительстве, оно имеет глобальную точку доступа «Правительство страны N».

Структура паттерна

Singleton

Одиночка определяет статический метод getInstance(), который возвращает единственный экземпляр класса *Одиночки*.
Конструктор одиночки должен быть скрыт от клиентов. Вызов getInstance() должен быть единственным способом получить объект этого класса.



Псевдокод паттерна

Пример

В этом примере роль Одиночки отыгрывает класс подключения к базе данных.

```
class Database is
        private field instance: Database

        static method getInstance() is
                if (this.instance == null) then
                        acquireThreadLock() and then
                                // На всякий случай ещё раз проверим не был ли объект создан
                                // другим потоком, пока текущий ждал освобождения блокировки.
                                if (this.instance == null) then
                                this.instance = new Database()
                        return this.instance
        private constructor Database() is
        // Здесь может жить код инициализации подключения к серверу баз данных.
        // ...
        public method query(sql) is
        // Все запросы к базе данных будут проходить через этот метод. Поэтому
        // имеет смысл поместить сюда какую-то логику кеширования.
        // ...
class Application is
        method main() is
                Database foo = Database.getInstance() foo.query("SELECT ...")
                // ...
                Database bar = Database.getInstance() bar.query("SELECT ...")
                // В переменной bar содержится тот же объект, что и в foo.
```

Применимость

Singleton

1. Если в программе должен быть единственный экземпляр какого-то класса, доступный всем клиентам. Например, общий доступ к базе данных из разных частей программы.
2. Когда вам хочется иметь больше контроля над глобальными переменными.

Алгоритм реализации

Singleton

1. Добавьте в класс приватное статическое поле, которое будет содержать одиночный объект.
2. Объявите статический создающий метод, который будет использоваться для получения одиночки.
3. Добавьте «ленивую инициализацию» (создание объекта при первом вызове метода) в создающий метод одиночки.
4. Сделайте конструктор класса приватным.
5. В клиентском коде замените вызовы конструктора вызовами создающего метода.

Преимущества и недостатки

Плюсы и недостатки

Плюсы:

- Гарантирует наличие единственного экземпляра класса.
- Предоставляет к нему глобальную точку доступа.
- Реализует отложенную инициализацию объекта-одиночки.

Минусы:

- Нарушает принцип единственной ответственности класса.
- Маскирует плохой дизайн.
- Проблемы мульти поточности.
- Требует постоянного создания Mock-объектов при юнит-тестирования.

Отношения с другими паттернами

Отношения к другим паттернам

- Фасад можно сделать Одиночкой, так как обычно нужен только один объект-фасад.
- Паттерн Легковес может напоминать Одиночку, если для конкретной задачи у вас получилось уменьшить количество объектов к одному. Но помните, что между паттернами есть два кардинальных отличия:
  1. Объекты-легковесы должны быть неизменяемыми, тогда как объект-одиночка допускает изменение своего состояния.
  2. Вы можете иметь множество объектов легковесов одного класса, в отличие от одиночки, который требует наличия только одного объекта.
- Абстрактная фабрика, Строитель и Прототип могут быть реализованы при помощи Одиночки.

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association