Padrões de Projeto: Singleton

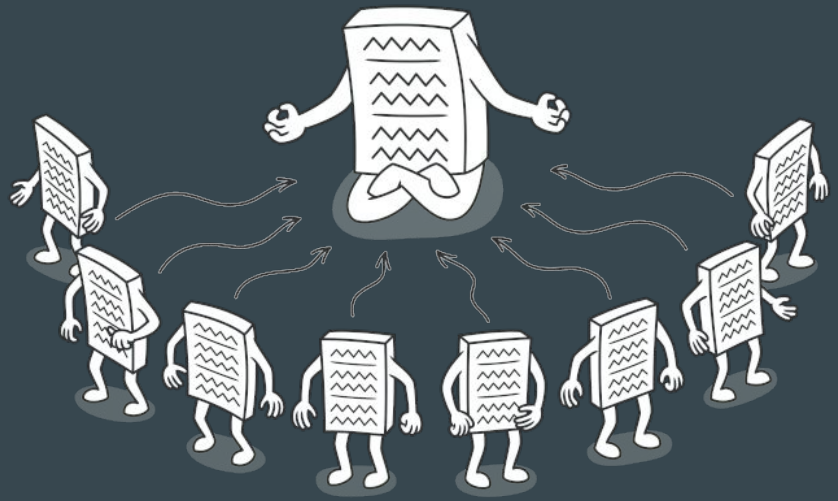
Carolina Stefanello e Mateus Zucco

Singleton

...

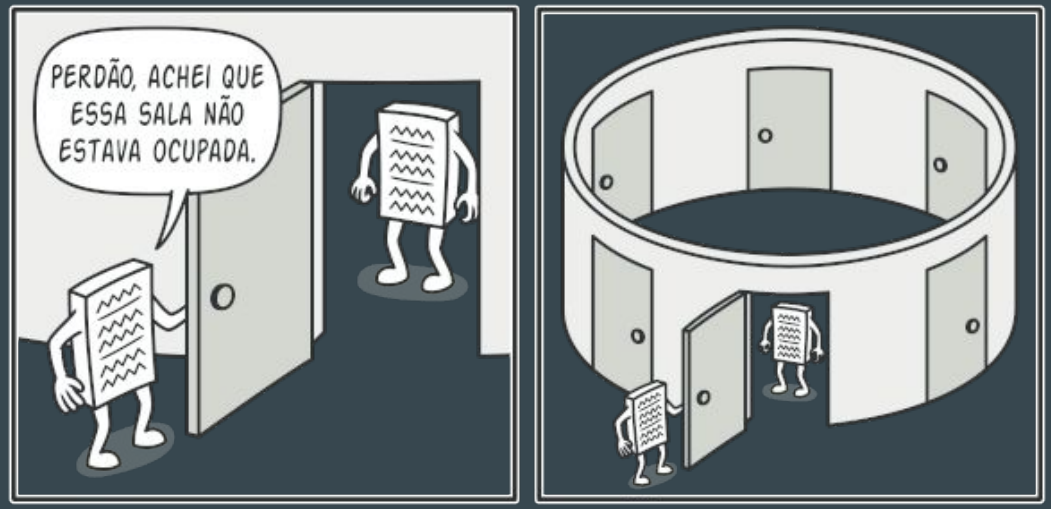
(Carta única)

Esse padrão de projeto, conhecido como Singleton, assegura que **uma classe possua somente uma instância ao longo da aplicação** , além de oferecer um acesso global **centralizado** a essa única instância.



Problemática 1: não há um meio de controlar o número de instâncias de uma classe, especialmente para acessos de recursos compartilhados

Solução 1: já tendo um objeto da classe instanciado previamente, ao criar um novo objeto a classe retorna a mesma instância já criado anteriormente



Problemática 2:

Variáveis globais apesar de muito úteis, elas são inseguras devido ao fato de permitir sobrescrever seus conteúdos.

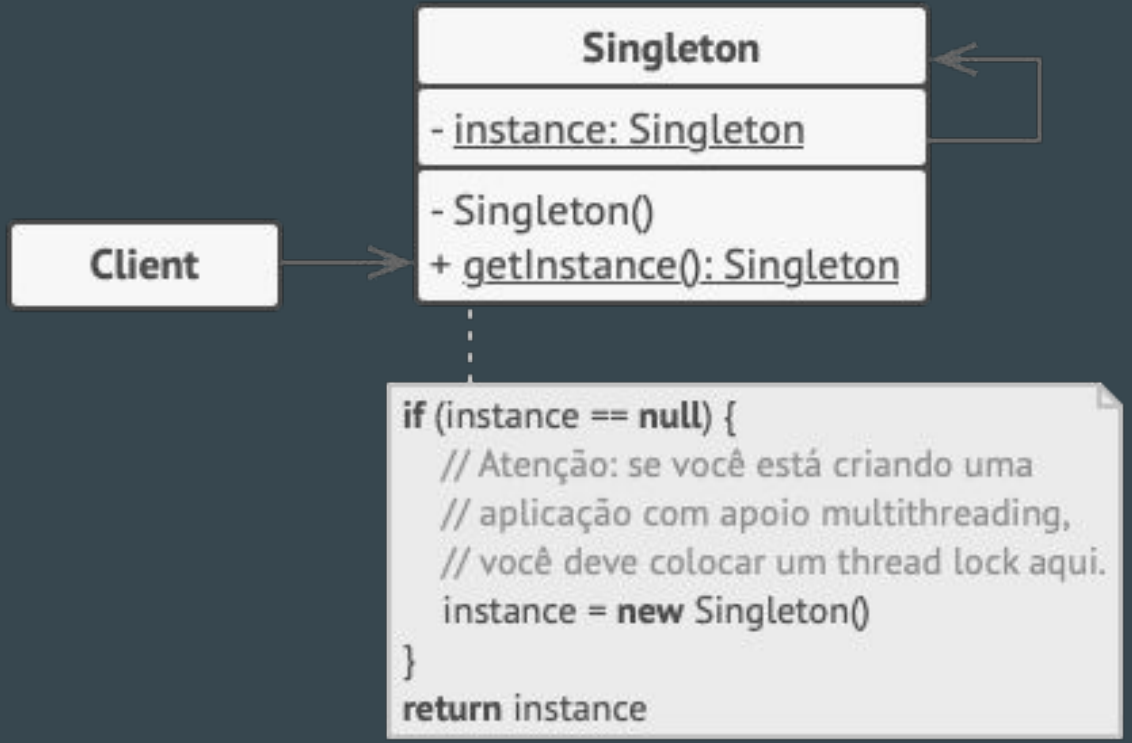
Solução 2:

O padrão Singleton permite acessar o objeto de qualquer linha do código, devido ao seu acesso global, e também protege o conteúdo do objeto devido a seu construtor privado

Características para a utilização:

A implementação do Singleton segue dois passos principais:

- primeiro, torna o **construtor privado** para impedir que outros objetos criem novas instâncias diretamente;
- segundo, cria um método estático que age como construtor, invocando o construtor privado para criar uma única instância e armazená-la em um campo estático. Assim, ao chamar esse método, **o mesmo objeto é retornado sempre** , garantindo uma única instância acessível globalmente.


Singleton
- instance: Singleton
- Singleton()
+ getInstance(): Singleton
Client
if (instance == null) {
// Atenção: se você está criando uma
// aplicação com apoio multithreading,
// você deve colocar um thread lock aqui.
instance = new Singleton()
}
return instance

```
// A classe Database define o método `getInstance` que permite
// clientes acessar a mesma instância de uma conexão a base de
// dados através do programa.
class Database is
    // O campo para armazenar a instância singleton deve ser
    // declarado como estático.
    private static field instance: Database

    // O construtor do singleton devem sempre ser privado para
    // prevenir chamadas diretas de construção com o operador
    // `new`.
    private constructor Database() is
        // Algum código de inicialização, tal como uma conexão
        // com um servidor de base de dados.
        // ...

    // O método estático que controla acesso à instância do
    // singleton
    public static method getInstance() is
        if (Database.instance == null) then
            acquireThreadLock() and then
                // Certifique que a instância ainda não foi
                // inicializada por outra thread enquanto está
                // estiver esperando pela liberação do `lock`.
                if (Database.instance == null) then
                    Database.instance = new Database()
        return Database.instance

    // Finalmente, qualquer singleton deve definir alguma lógica
    // de negócio que deve ser executada em sua instância.
    public method query(sql) is
        // Por exemplo, todas as solicitações à base de dados de
        // uma aplicação passam por esse método. Portanto, você
        // pode colocar a lógica de throttling ou cache aqui.
        // ...

class Application is
    method main() is
        Database foo = Database.getInstance()
        foo.query("SELECT ...")
        // ...
        Database bar = Database.getInstance()
        bar.query("SELECT ...")
        // A variável `bar` vai conter o mesmo objeto que a
        // variável `foo`.
```

Prós:

- Garantia que uma classe terá apenas uma instância.
- Ponto de acesso global para essa instância.
- O objeto singleton é criado apenas quando for requisitado pela primeira vez.

Contras:

- Pode violar o princípio da responsabilidade única, pois tende a resolver múltiplos problemas ao mesmo tempo.
- Além disso, pode ocultar deficiências no design, como quando diferentes componentes de um sistema possuem um conhecimento excessivo sobre os outros.
- Em ambientes multithreaded, é necessário um cuidado extra para evitar que várias threads criem instâncias do singleton simultaneamente.

Referência:

REFACTORING GURU. Singleton. Disponível em: https://refactoring.guru/pt-br/design-patterns/singleton. Acesso em: 31 de out. de 2024.

SHVETS, Alexander. Mergulho nos Padrões de Projeto. v. 2021-1.14. Refactoring.Guru, 2021.

MIRANDA, Otávio. Canal Otávio Miranda. Disponível em: https://www.youtube.com/c/Ot%C3%A1vioMiranda Acesso em: 31 de out. de 2024.