Ficha Resumo - Padrão de Projeto: Singleton

O que é?

O Singleton é um padrão de projeto da categoria criacional, cujo objetivo é garantir que uma classe tenha apenas uma única instância e fornecer um ponto de acesso global a essa instância.

A ideia é simples, mas poderosa: em determinadas situações, você não quer ou não pode ter múltiplas instâncias de uma classe, seja por razões de lógica de negócio, consumo de recursos ou consistência de estado. O Singleton resolve isso controlando a criação do objeto e armazenando a instância criada de forma que ela seja sempre reutilizada.

Por que usar?

Existem diversos cenários onde o Singleton se mostra útil:

Um sistema de logging que precisa ser acessado de vários lugares e manter o mesmo destino de saída (arquivo, terminal, etc).

Um gerenciador de configurações que carrega dados uma única vez e deve disponibilizá-los globalmente.

Um pool de conexões com banco de dados que deve ser único para evitar múltiplas conexões desnecessárias.

Um cache centralizado onde os dados são acessados e modificados por diferentes partes do sistema.

Como funciona?

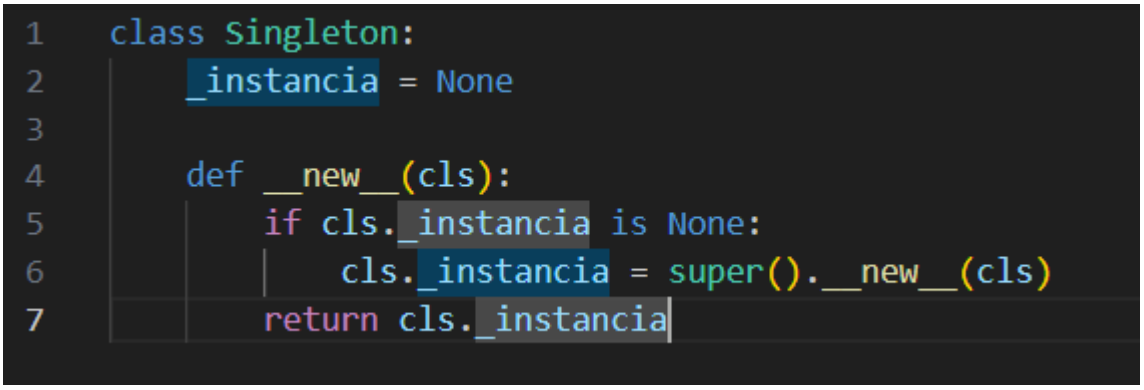
A implementação do Singleton envolve geralmente os seguintes elementos:

Um atributo privado que armazena a única instância da classe.

Um construtor privado ou protegido (dependendo da linguagem) para evitar instância direta com new.

Um método público de acesso, normalmente chamado de getInstance() ou equivalente, que retorna a instância criada — ou a cria, se ainda não existir.

Exemplo básico em Python:



```python
class Singleton:
    _instancia = None

    def __new__(cls):
        if cls._instancia is None:
            cls._instancia = super().__new__(cls)
        return cls._instancia
```

No exemplo acima:

O método especial __new__ é sobrescrito para controlar a criação da instância.

A instância é armazenada como um atributo de classe (_instancia), garantindo que sempre a mesma seja retornada.

Variações importantes

Lazy Initialization: a instância só é criada quando realmente for necessária (adiando a criação até o primeiro uso).

Thread-Safe Singleton: em ambientes concorrentes (como aplicações web ou multithread), o Singleton precisa ser protegido contra condições de corrida com locks, synchronized, mutex ou outro mecanismo apropriado.

Eager Initialization: a instância é criada logo na carga da aplicação. Útil quando se sabe que o Singleton será usado com certeza e pode ser mais performático.

Singleton com Enum (em Java): uma forma recomendada e segura de criar Singletons usando o recurso de enumeração, que é naturalmente serializável e resistente a reflexão.

Vantagens

Instância única garantida, evitando duplicidade de objetos críticos.

Redução de consumo de recursos, evitando múltiplas alocações desnecessárias.

Ponto de acesso global fácil de utilizar.

Útil para representar entidades globais no sistema, como configurações, cache, drivers etc.

Desvantagens

Difícil de testar: como o Singleton mantém estado global, pode dificultar testes automatizados, especialmente quando o estado precisa ser isolado entre testes.

Quebra do Princípio da Responsabilidade Única (SRP): a classe passa a ter duas responsabilidades — sua lógica principal e o controle de sua própria instância.

Acoplamento global: muitas partes do código dependem diretamente do Singleton, tornando mudanças mais difíceis.

Problemas com multithreading: sem proteção adequada, múltiplas instâncias podem ser criadas em ambientes concorrentes.

Cuidados ao usar

Avalie se o Singleton realmente é necessário. Muitas vezes, uma injeção de dependência (como em frameworks que usam IoC) resolve o problema de forma mais elegante e testável.

Certifique-se de que o padrão seja thread-safe, se usado em ambientes paralelos.

Evite o uso abusivo como um “atalho” para variáveis globais.

Alternativas

Inversão de Controle (IoC) e Injeção de Dependência (DI) são alternativas modernas que evitam acoplamento excessivo e facilitam testes.

Para bibliotecas que expõem objetos globais (como logging em Python), geralmente o Singleton já está implementado internamente, e não é necessário recriar isso manualmente.