Singleton 💻

Análise e Desenvolvimento de Sistemas - 6º Período

Singleton

- Padrão **criacional** que garante que uma classe tenha apenas uma instância
- Fornece ponto de acesso global a essa instância.
- Apesar de ser um acesso global (parecido como uma variável global), **protegemos** essa instância de ser sobrescrita.

Singleton

- Um dos motivos seria **controlar o acesso a um recurso compartilhado**, por exemplo um arquivo.
- Nesse sentido, lembra o padrão **Proxy**, mas no proxy queremos a criação **on-demand** e no Singleton uma única instância
- Um único objeto é criado. Quando outro objeto for "requisitado", retorna aquele já criado
- Sua principal diferença entre os outros padrões criacionais é a preocupação com a Singularidade

Cenário

- Um S.O, por exemplo, tem apenas 1 file system. Mas vários sistemas e subsistemas precisam interagir com ele. O mesmo vale para um gerenciador de janelas do S.O.
- No entanto, não precisamos criar **N** instâncias para servir esses subsistemas. Podemor ter apenas uma e fornecer um **ponto de acesso global**

Cenário

- Conexão com banco de dados é um excelente exemplo. Quando precisamos de **N** objetos que representam uma conexão, não precisamos abrir essa conexão várias vezes. Usamos o mesmo objeto instanciado para usar a conexão

Singleton

- A melhor maneira de lidar com isso é deixar **a própria classe responsável pelo gerenciamento da criação das instâncias**
- O desafio é lidar com o **construtor**. A cada vez que chamamos a classe, o construtor inicializa os valores e gera um objeto novo

Linguagens que possuem modificadores de acesso

- Para linguagens que possuem modificadores de acesso (Java, C#), uma maneira seria declarar o **construtor como privado**. Assim, outras classes não conseguem usar o Singleton com o operador **new**

Linguagens que possuem modificadores de acesso

- Cria um método **estático** que chama o construtor privado e gerencia as outras chamadas a classe. Se for chamado novamente, não chama o construtor privado mas retorna o objeto já criado. **Todas as chamadas após a primeira retorna esse objeto em cache**

Usando modificadores de acesso - Java

```
public class Singleton {

    private static Singleton uniqueInstance;

    /*Contrutor privado*/
    private Singleton() {
    }

    /*Método estático que controla a criação da instância*/
    public static Singleton getInstance() {
        if (uniqueInstance == null)
            uniqueInstance = new Singleton();

        return uniqueInstance;
    }
}
```

Singleton no python

- O python não faz uso desses modificadores de acesso, portanto precisamos utilizar outros recursos
- Não existem as palavras reservadas **public** e **private**
- Para implementar o Singleton, vamos manipular **um método mágico do python: __new__**

Singleton no python

- No python, os métodos mágicos (dunder methods) implementam comportamentos internos dos objetos
- Podemos sobrescrever esses **métodos**, permitindo assim um **comportamento personalizado**

Singleton no python

- **__init__**: Já conhecemos e utilizamos. É um **método de instância** responsável por inicializar um objeto recém criado. Recebe o próprio objeto como primeiro argumento (**self**).
- **__new__**: Cria uma nova instância. Recebe a própria classe como primeiro argumento e o restante dos argumentos que foram passados para o construtor do objeto.

Singleton no python

- Quando um instância é requisitada, o __new__ é responsável por retornar esse objeto, e após isso, o __init__ desse objeto recebe os valores de construtor para inicializá-los.
- Portanto o __new__ é chamado antes do __init__ quando "pedimos" a criação de um objeto
- Documentação python sobre dunder methods

Quando usamos o Singleton

- Quando queremos **apenas uma instância** da classe e mesma deve ser acessível ao client a qualquer momento (várias chamadas)
- Quando a única instância deve ser extensível por subclasse e o client usar essa instância estendida sem modificar o seu código