Singleton – Design Pattern

O design pattern ou padrão de projeto Singleton é um padrão que vem sendo usado utilizado pela indústria da engenharia de software desde os meados da década de 90. A sua popularidade pode ser atribuída ao livro Design Patterns: Elements of Reusable Object-Oriented Software, escrito por quatro autores que ficaram conhecidos como a “Gangue dos Quatro”: Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides.

O padrão Singleton é um design pattern que garante que **uma classe possua apenas uma instância**, enquanto permite um **ponto de acesso global para esta mesma instância.** Um motivo comum de querermos garantir que uma classe possua apenas uma instância seria o de controlar o acesso a um recurso compartilhado, como por exemplo um banco de dados ou um outro arquivo relacionado com a aplicação que venha a ser desenvolvida. Importante notar nesse caso que esse objetivo não poderia ser atingido através de um método construtor, pois a chamada do método construtor deve sempre retornar uma nova instância de uma determinada classe.

Em relação ao ponto de acesso global para a instância do padrão Singleton, é comum em aplicações alguns objetos essenciais serem armazenados em variáveis globais que podem ser acessadas por todo o sistema em questão. No entanto, apesar de ser útil, essa prática pode ser insegura pois essas variáveis globais podem ter seu conteúdo substituído ou modificado eventualmente no desenvolvimento da aplicação, podendo causar bugs ou mal funcionamento do sistema. Portanto, assim como em uma variável global do sistema, no padrão Singleton a instância única da classe pode ser acessada de qualquer lugar do código, mas essa instância é também protegida para não ter o seu conteúdo modificado ou substituído através de qualquer outro código da aplicação. Além disso, torna-se uma boa prática ter por fins de controle um código essencial da sua aplicação restrito a apenas uma classe, ainda mais se o resto da aplicação depender do código relacionada com essa instância do design pattern.

Todas as implementações do padrão de projeto Singleton possuem estas duas etapas em comum:

**- tornar “private” o método construtor padrão, para prevenir que outros objetos utilizem o operador “new ” com a classe Singleton e obtenham uma nova instância.**

**- criação de um método estático (static) de criação que funciona como um método construtor, chamando o método construtor privado e guardando o objeto criado em um campo estático. Todas as chamadas subsequentes para este método retornam assim este objeto já armazenado ou criam um novo se ainda não houver um.**

Portanto se o código tiver acesso a uma classe Singleton, o seu método estático pode ser chamado a qualquer momento, retornando o mesmo objeto criado e desativando qualquer outra forma de instanciação de um novo objeto de uma classe. O padrão Singleton é assim geralmente usado quando uma classe do seu programa tiver apenas uma instância disponível para todos os clientes ou usuários, como por exemplo um objeto JSON do banco de dados compartilhado em diferentes partes do programa. O padrão Singleton também é usado quando é necessário um controle mais restrito sobre variáveis globais, pois garante que há apenas uma instância da classe (sendo possível alterar essa limitação de acordo com a necessidade do projeto) .

**Implementação:**

**1- Adicionar um campo estático e privado na classe para armazenar a instância Singleton.**

**2- Declarar um método público estático para poder acessar a instância Singleton.**

**3- Implementar “inicialização preguiçosa” no método público estático; criando um novo objeto apenas na primeira chamada e retornando sempre ele nas chamadas seguintes.**

**4- Tornar privado o método construtor da classe onde se pretende desenvolver. O método estático da classe poderá chamar o construtor mas não os outros objetos externos.**

**5- Substituir no código do cliente as chamadas do construtor Singleton pelas chamadas do método especial público estático de criação, criado no item 2.**

**Vantagens do Singleton:**

**1- Ponto de acesso global para a instância única (praticidade).**

**2- Maior controle sobre variáveis globais, públicas ou objetos compartilhados no programa.**

**Desvantagens do Singleton:**

**1- Viola o princípio da responsabilidade única, o padrão resolve dois problemas ao mesmo tempo.**

**2- Pode esconder más práticas, como quando os componentes do programa tem acesso demasiada ás informações de outros componentes.**

**3- Requer tratamento especial em ambientes ramificados (multithread) para evitar que os ramos (threads) criem a mesma instância Singleton diversas vezes no programa.**

**4- Difícil executar testes unitários no código do cliente pois o método construtor da classe Singleton é privado e geralmente é impossível fazer um override em métodos estáticos, no caso o método especial para inicializar a única instância do programa.**