# **Padrão Singleton**

Existem casos em que algumas classes deveriam logicamente ter apenas uma única instância. Exemplos disso são factories e objetos que acessem algum recurso não compartilhado, como acesso ao banco de dados e memória. Porém, se implementarmos as classes de modo tradicional, nada impede que vários objetos da mesma classe sejam instanciados.

O Singleton é um padrão de projeto criacional, que garante que apenas um objeto desse tipo exista e forneça um único ponto de acesso a ele para qualquer outro código.

O padrão Singleton impõe o projeto colocando a responsabilidade da criação e da intermediação do acesso à instância no próprio objeto. Dessa forma, apenas uma instância da classe será criada, além de ter um único ponto de acesso a essa instância.

No entanto, uma classe singleton comum, conhecido também como singleton ingênuo ou lazy singleton, não se comporta corretamente em um ambiente multithread. Vários threads podem chamar o método de criação simultaneamente e obter várias instâncias da classe Singleton.

## Padrão Singleton Thread-safe

Para corrigir o problema do lazy singleton, você deve sincronizar os threads durante a primeira criação do objeto Singleton. Para que isso seja possível, será necessário criar um objeto Lock dentro da classe Singleton que é responsável pelo bloqueio, isto é, para sincronizar threads no primeiro acesso ao objeto Singleton.

Agora, imagine que o programa acabou de ser executado. Como ainda não há instância de Singleton, vários threads podem passar simultaneamente pela condicional anterior e chegar a esse ponto quase ao mesmo tempo. O primeiro deles adquirirá bloqueio e prosseguirá, enquanto o restante aguardará na fila de espera.

A primeira thread a adquirir o bloqueio, atinge essa condicional, entra e cria a instância Singleton. Depois de deixar o bloco de bloqueio, um thread que pode estar esperando pela liberação do bloqueio pode entrar nesta seção. Mas como o campo Singleton já foi inicializado, o thread não criará um novo objeto.