Arquitetura de Software

Padrões de Projetos – Padrões Criacionais (Continuação)

Nairon Neri Silva

Singleton

Intenção Oficial

- Garantir que uma classe tenha somente uma instância e fornecer um ponto global de acesso para a mesma

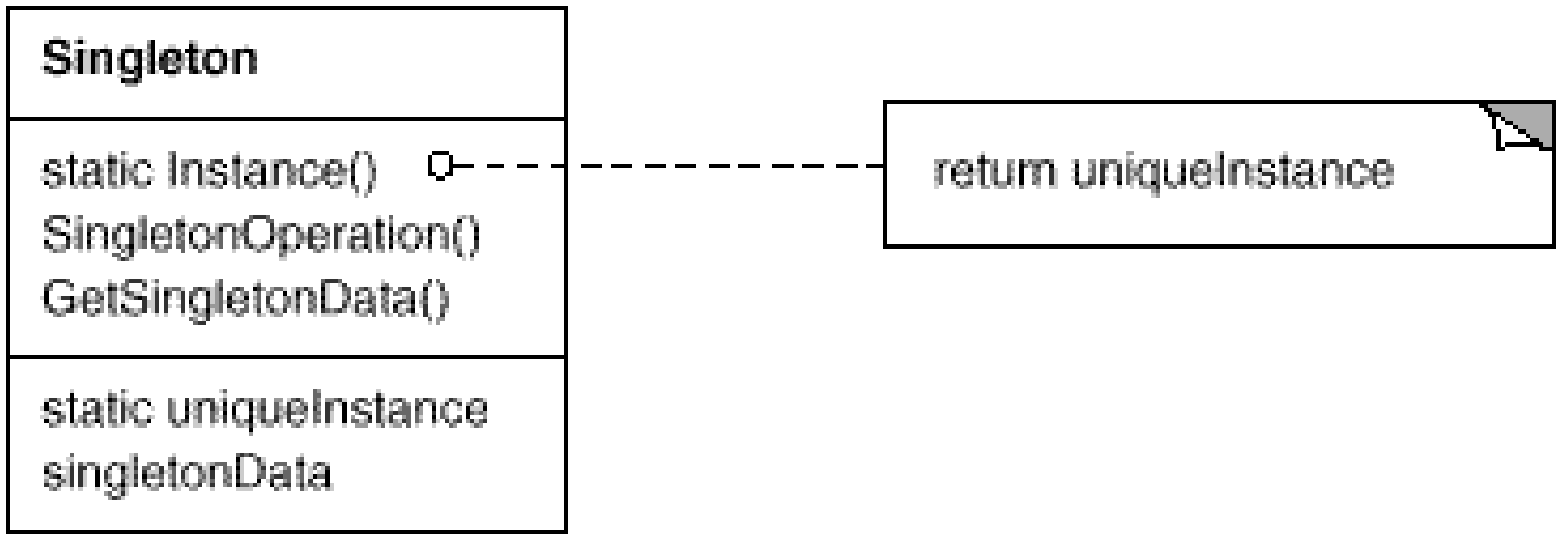
Motivação

- É importante para algumas classes ter uma, e apenas uma, instância.
- Exemplos:
  - Embora possam existir muitas impressoras em um sistema, deveria haver somente um sistema de *spooler* de impressoras
  - Da mesma forma, deveria haver somente um sistema de arquivos e um gerenciador de janelas.
  - O governo de um país, já que cada país pode ter somente um governo oficial
  - Um sistema de contabilidade será dedicado a servir somente a uma companhia
- Como garantimos que uma classe tenha somente uma instância e que essa seja facilmente acessível?

Aplicabilidade

1. Quando deve haver apenas uma instância de uma classe, e essa instância deve dar acesso aos clientes através de um ponto bem conhecido

2. Quando a única instância tiver de ser extensível através de subclasses, possibilitando aos clientes usarem uma instância estendida sem alterar o seu código


Singleton
static Instance()
SingletonOperation()
GetSingletonData()
static uniqueInstance
singletonData
return uniqueInstance

Participantes

- **Singleton**
  - Define uma operação (de classe - estática) que permite aos clientes acessarem sua única instância
  - Pode ser responsável pela criação da sua própria instância única

Consequências Boas

1. Acesso controlado à instância única
2. É possível e fácil aumentar o número de instâncias (caso seja necessário)
3. O Singleton só é criado no momento do uso
4. Substitui variáveis globais
5. Melhora o desempenho do software quando utilizado em classes que nunca mudam de estado ou que sempre precisam de somente uma instância executando;

Consequências Ruins

1. Aumenta a complexidade dos testes
2. Ignora o princípio da responsabilidade única, o padrão resolve dois problemas ao mesmo tempo:
   I. *Garante que a classe tem somente uma instância;*
   II. *Fornece um ponto único de acesso a essa classe.*
3. Precisa de especial atenção em caso de processamento concorrente
4. Um dos autores do GOF, Erich Gamma, descreveu que este seria o único padrão que ele retiraria do livro em caso de refatoração da publicação.

Situação Problema

Imagine que você precise mostrar a versão do seu sistema em vários locais...

Qual o problema? Ao trocar a versão teria que trocar em todos os locais.

Solução usando Singleton

```
1.  public class InformacoesGlobais {
2.      private String versao = "2.0.3.140";
3.      private static InformacoesGlobais instance;
4.      private InformacoesGlobais() { }
5.      public static InformacoesGlobais getInstance() {
6.          if (instance == null)
7.              instance = new InformacoesGlobais();
8.          return instance;
9.      }
10.     public String getVersao() {
11.         return versao;
12.     }
13. }
```

Como usar ?

```
InformacoesGlobais.getInstance().getVersao();
```

Fonte dos Slides

https://www.youtube.com/watch?v=x9h8MgAvi_I

https://www.devmedia.com.br/padrao-de-projeto-singleton-em-java/26392

https://www.thiengo.com.br/padrao-de-projeto-singleton

https://refactoring.guru/pt-br/design-patterns/singleton/java/example

Atividade

1) Explique de forma suscinta o objetivo do padrão Factory Method;
2) Explique de forma suscinta o objetivo do padrão Prototype;
3) Explique de forma suscinta o objetivo do padrão Singleton;
4) Cite e explique pelo menos um situação real de aplicação (diferentes dos citados até o momento) de cada um dos padrões abaixo :
   A. Factory Method;
   B. Prototype;
   C. Singleton.