Curso: Sistemas de Informação	
Disciplina: Padrões de projeto	**Período:** 8
Professor: Luiz Gustavo Dias	**Tipo:** Estudo Complementar
Objetivo: Princípios da Engenharia que norteiam a prática	

DESIGNE PATTERNS: SINGLETON

Intenção

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. Por exemplo, embora possam existir muitas impressoras em um sistema, deveria haver somente um spooler de impressoras. Da mesma forma, deveria haver somente um sistema de arquivos e um gerenciador de janelas. Um filtro digital terá somente um conversor A/D. Um sistema de contabilidade será dedicado a servir somente a uma companhia.

Como garantimos que uma classe tenha somente uma instância e que essa instância seja facilmente acessível? Uma variável global torna um objeto acessível, mas não impede você de instanciar múltiplos objetos.

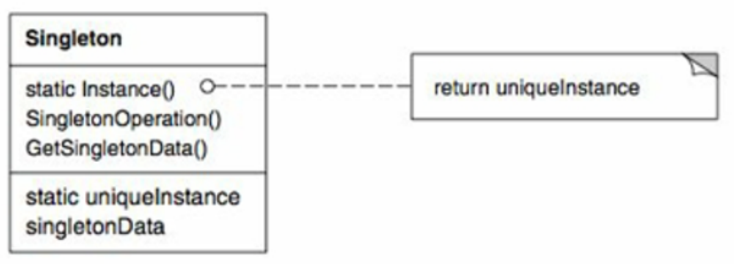
Uma solução melhor seria tornar a própria classe responsável por manter o controle da sua única instância. A classe pode garantir que nenhuma outra instância seja criada (pela interceptação das solicitações para criação de novos objetos), bem como pode fornecer um meio para acessar sua única instância. Este é o padrão Singleton.

Aplicabilidade

- Quando for preciso haver apenas uma instância de uma classe, e essa instância tiver que dar acesso aos clientes através de um ponto bem conhecido;

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

Estrutura



Participantes

Singleton:

- define uma operação Instance que permite aos clientes acessar em sua única instância. Instance é uma operação de classe (ou seja, em Smalltalk é um método de classe e em C++ é uma função-membro estática).
- pode ser responsável pela criação da sua própria instância única.

Colaborações

Os clientes acessam uma instância Singleton unicamente pela operação Instance do Singleton.

Consequências

O padrão Singleton apresenta vários benefícios:

1. Acesso controlado à instância única. Como a classe Singleton encapsula a sua única instância, possui controle total sobre como e quando os clientes a acessam.

2. Espaço de nomes reduzido. O padrão Singleton representa uma melhoria em relação ao uso de variáveis globais. Ele evita a poluição do espaço de nomes com variáveis globais que armazenam instâncias únicas.

3. Permite um refinamento de operações e da representação. A classe Singleton pode ter subclasses e é fácil configurar uma aplicação com uma instância dessa classe estendida. Você pode configurar a aplicação com uma instância da classe de que necessita em tempo de execução.

4. Permite um número variável de instâncias. O padrão torna fácil mudar de idéia, permitindo mais de uma instância da classe Singleton. Além disso, você pode usar a mesma abordagem para controlar o número de instâncias que a aplicação utiliza. Somente a operação que permite acesso à instância de Singleton necessita ser mudada.

5. Mais flexível do que operações de classe. Uma outra maneira de empacotar a funcionalidade de um singleton é usando operações de classe (ou seja, funções-membro estáticas em C++ ou métodos de classe em Smalltalk). Porém, as técnicas de ambas as linguagens tornam difícil mudar um projeto para permitir mais que uma instância de uma classe. Além disso, as funções- membro estáticas em C++ nunca são virtuais, o que significa que as subclasses não podem redefini-las polimorficamente.