Universidade Federal de Ouro Preto

CSI032 - Programação de Computadores II

Padrões de Projeto - Singleton

Professor: Dr. Rafael Frederico Alexandre
Contato: rfalexandre@decea.ufop.br
Colaboradores: Eduardo Matias Rodrigues
Contato: eduardo.matias@aluno.ufop.edu.br

Padrões de Projeto - Singleton

Singleton

Padrões de projeto

Motivação

1. Projetar softwares *reutilizaveis* orientado a objetos é difícil;
   1. O projeto deve resolver o problema atual, mas também deve ser genérico para atender problemas e requisitos futuros;
   2. O re-projeto deve ser evitado, pelo menos minimizado.
2. Não devemos tentar resolver todos os problemas a partir do zero, em vez disso devemos optar por soluções que já se mostraram eficientes.

Padrões de projeto

Motivação

1. Podemos encontrar padrões em projetos de software;
   1. Quando projetistas encontram uma boa solução para um problema específico, eles a usam repetidamente, logo, encontramos padrões de classes e de comunicação entre objetos.

Padrões de projeto

Analogia

1. Padrões de projeto são encontrados nas mais diversas áreas, roteiristas de cinemas por exemplo raramente projetam suas tramas do zero;
    1. Um filme de herói segue alguns padrões como "O herói problemático"ou "A Jornada do heroi", por isso quando vemos um filme as vezes podemos intuir facilmente cenas que irão acontecer;
2. Projetistas de software também seguem padrões, você já deve ter sentido um *déja vu* durante um projeto (aquela sensação que já resolveu um problema parecido, mas não lembra exatamente como).

Padrões de projeto

Motivação

1. Cada padrão de projeto nomeia, explica e avalia um aspecto de projeto de um sistema orientado a objetos;
2. O objetivo é capturar a experiência de projeto de uma forma que os programadores possam reaproveitá-las depois;

Padrões de projeto

Elementos

1. Cada padrão de projeto tem quatro elementos essenciais:
   1. O **nome** do padrão, que deve ser autoexplicativo;
   2. O **problema** que ele resolve;
   3. A **solução**: como o padrão de projeto resolve o problema;
   4. As **consequências**: vantagens e desvantagens de se utilizar o padrão.

Padrões de projeto
Organização [GAMMA, 1994]

1. Como existem muitos padrões de projetos, é necessário uma maneira de organizá-los;
2. O GoF (Gang of Four, vide [GAMMA, 1994]) classifica um padrão quanto a sua **finalidade**, ou seja, o que um padrão faz;
    1. Os padrões podem ter finalidade de criação, estrutural ou comportamental;

Padrões de projeto
Finalidades [GAMMA, 1994]

1. Padrões de criação: se preocupam com o processo de criação de objetos;
2. Padrões estruturais: lidam com a composição de classes ou de objetos;
3. Padrões comportamentais: descreve a maneira como classes ou objetos interagem.

Padrões de projeto

Padrões GoF

Criação	Estrutural	Comportamental
Factory Method	Adapter	Interpreter
Abstract Factory	Bridge	Template Method
Builder	Composite	Chain of Responsibility
Prototype	Decorator	Command
Singleton	Façade	Iterator
	Flyweight	Mediator
	Proxy	Memento
		Observer
		State
		Strategy
		Visitor

Singleton

Singleton

Motivação [de Lima, 2010]

1. Quando estamos construindo um sistema, podemos precisar que determinadas classes sejam instânciadas apenas uma vez, por exemplo:
   1. Um sistema de arquivos;
   2. Um ponto de acesso ao banco de dados;
   3. Um objeto que contém a configuração do programa;
   4. Um spooler de impressão;
2. Como exemplificado, esse requisito é recorrente em diversos projetos.

Singleton
Exemplo

1. Suponha que estamos construindo um sistema e precisamos implementar a funcionalidade de imprimir relatórios. Para isso temos uma classe **Impressora** e devemos garantir que exista apenas um objeto dessa classe, ou seja, essa classe deve ser instânciada apenas uma vez e toda classe do sistema deve acessar o mesmo objeto ("como se fosse uma variável global de impressão") **impressora**.
2. Obs: este é apenas um exemplo para introduzir o padrão Singleton.

Exemplo
Classe Impressora

```
public class Impressora {

    public void imprimir(String relatorio) {
        System.out.println(relatorio);
    }
}
```

Exemplo
Classe de teste

```
public static void main(String[] args) {

    Impressora impressora = new Impressora();
    System.out.println("Objeto 1: " + impressora);

    Impressora impressora2 = new Impressora();
    System.out.println("Objeto 2: " + impressora2);
}
```

Exemplo
Saída do programa

```
run:
Objeto 1: teste1.Impressora@15db9742
Objeto 2: teste1.Impressora@6d06d69c
BUILD SUCCESSFUL (total time: 0 seconds)
```

1. Podemos ver que foram criados dois objetos distintos da classe **Impressora**, o que vai contra nosso requisito de unicidade;
    1. Dessa forma, vários objetos de impressora podem ser criados em várias classes do sistema;
2. Como resolver esse problema?

Solução

Singleton [de Lima, 2010]

1. Queremos que a classe **Impressora** tenha apenas uma instância e um ponto de acesso global a ela;
2. O padrão de projeto **Singleton** descreve a solução para esse problema:
   1. Quando o objeto for requisitado pela primeira vez, a instância deve ser criada;
   2. Nas próximas vezes que o objeto for requisitado, a instância criada na primeira vez deve ser fornecida.

Solução
Singleton

1. A classe Impressora deve:
   1. armazenar a única instância existente;
   2. garantir que apenas uma única instância será criada;
   3. prover acesso a tal instância.
2. Diagrama de classes:

Singleton

Singleton

Classe ImpressoraSingleton

```
public class ImpressoraSingleton {

    private static ImpressoraSingleton instancia = null;

    private ImpressoraSingleton() { /* */ }

    public static ImpressoraSingleton GetInstancia() {
        if (instancia == null) {
            instancia = new ImpressoraSingleton();
        }
        return instancia;
    }

    public void imprimir(String relatorio) {
        System.out.println(relatorio);
    }
}
```

Singleton
Classe de teste

```
public static void main(String[] args) {

    ImpressoraSingleton impressora = ImpressoraSingleton.GetInstancia();
    System.out.println("Objeto 1: " + impressora);

    ImpressoraSingleton impressora2 = ImpressoraSingleton.GetInstancia();
    System.out.println("Objeto 2: " + impressora2);

    ImpressoraSingleton impressora3 = ImpressoraSingleton.GetInstancia();
    System.out.println("Objeto 3: " + impressora3);
}
```

Singleton
Saída do programa

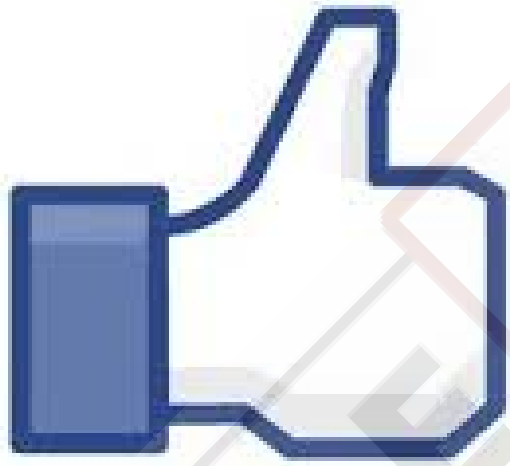
```
run:
Objeto 1: teste1.ImpressoraSingleton@15db9742
Objeto 2: teste1.ImpressoraSingleton@15db9742
Objeto 3: teste1.ImpressoraSingleton@15db9742
BUILD SUCCESSFUL (total time: 0 seconds)
```

1. Podemos ver que as variáveis de impressão armazenam a referência do mesmo objeto;
2. Em qualquer classe do sistema que for necessário fazer uma impressão, basta fazer uma chamada ao método *GetInstancia* e o mesmo objeto sempre será retornado!

Singleton

Singleton
Consequências [GAMMA, 1994]

1. A classe **Singleton** possui total controle sobre como e quando sua instância é acessada;
2. Temos um ponto de acesso global, assim não precisamos criar variáis globais em toda classe que quisermos imprimir um documento, evitando a poluição do espaço de nomes;



MUITO OBRIGADO!!!

Referências Bibliográficas I

de Lima, E. S. (2010).
Aula 05 – padrões gof (singleton e iterator).
http://edirlei.3dgb.com.br/index.php?option=com$_c$*ontentview* = *articleid* = 103*Itemid* = 123.

GAMMA, Erich. HELM, R. J. R. V. J. (1994).
Padrões de Projeto - Soluções reutilizaveis de software orientado a objetos.
1994 M. C. Escher / Gordon Art - Baam - Holland, São Paulo, Brasil, 1st. edition.