INSTITUTO FEDERAL DE SANTA CATARINA

JULIANA STEFANES DECARLLI (202020905222)
NICOLE ALICE VIEIRA (202020902743)
THIAGO ESPINDULA (201920801563)

Padrões de Projeto de Software

GASPAR
2023

1. Singleton

O Singleton é um padrão de projeto que garante que uma classe tenha apenas uma instância promovendo acesso global a essa instância. O Singleton é um padrão de projeto útil em situação que é necessário limitar a quantidade de instâncias de uma classe em todo o sistema.

Para implementar o Singleton, é necessário definir o construtor da classe como privado, impedindo assim que outras classes instanciem objetos diretamente e definir a classe como “final” em Java para impedir que a mesma seja herdada por outras classes. Em seguida, deve-se criar um atributo privado e estático do tipo da classe e um método estático que retorna esse atributo. No método estático, verifica-se se o atributo estático criado é igual a nulo e, se for o caso, a instância da classe é criada nesse momento. Esse método é o único que acessa a instância da classe, garantindo assim que ela seja instanciada apenas uma vez durante a inicialização do sistema.

Main.java

```java
public final class Singleton {

    private static Singleton instance;

    // Construtor privado para impedir instância direta
    private Singleton() {
        // Inicialização do objeto singleton
    }

    // Método estático para obter a instância única
    public static Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```

Exemplo de uso do padrão Singleton

2. Builder

O Builder é um padrão de projeto de criação, que permite que uma classe se preocupe com apenas uma parte da construção de um objeto. O padrão Builder permite construir objetos complexos passo a passo, permitindo que o objeto tenha diversas representações usando o mesmo código de construção.

O fato do processo de construção do objeto ser encapsulado em uma classe separada, facilita na criação de objetos, pois oculta a complexidade e detalhes. Também, o Builder permite a criação de diferentes representações de um objeto usando o mesmo processo de construção. Isso é útil quando o objeto a ser construído pode ter diferentes configurações ou opções.

Também, esse padrão se faz necessário na criação de objetos imutáveis com muitos atributos opcionais. Ao invés de criar diversos construtores com diferentes combinações de parâmetros, o Builder oferece uma forma mais flexível e legível de construir o objeto, permitindo que o cliente defina apenas os atributos necessários e omita os opcionais.

3. Prototype

O Prototype é um padrão de projeto criacional que cria objetos a partir de uma instância protótipo que é clonado.

Com a utilização do padrão temos como vantagens clonar objetos sem acoplar nas classes concretas, não ter códigos com inicializações repetidas podendo clonar protótipos pré-construídos e ter uma alternativa com herança quando se lidar com configurações pré-determinadas de objetos complexos.

O padrão é utilizado quando se é necessário clonar um objeto em tempo de execução, pode ser utilizado em sistemas com as seguintes situações: quando se utilizam classes definidas em runtime, quando utilizam o padrão Abstract Factory em criação de objetos ou quando o estado inicial dos componentes possuem poucas variações.

4. Referências

REFACTORING.GURU. **Singleton**. Disponível em: https://refactoring.guru/pt-br/design-patterns/singleton. Acesso em: 23 maio 2023.

DEVMEDIA. **Padrão de Projeto Singleton em Java**. Disponível em: https://www.devmedia.com.br/padrao-de-projeto-singleton-em-java/26392. Acesso em: 23 maio 2023.

DEVMEDIA. **Padrões de projeto em .NET: Prototype**. Disponível em: https://www.devmedia.com.br/padroes-de-projeto-em-net-prototype/4597. Acesso em: 23 maio 2023.

DEVMEDIA. **Design Patterns:**: aplicando os padrões builder, singleton e prototype. aplicando os padrões builder, Singleton e prototype. 2023. Disponível em: https://www.devmedia.com.br/design-patterns-aplicando-os-padroes-builder-singleton-e-prototype/31023. Acesso em: 23 maio 2023

IMASTERS. **Arquitetura e desenvolvimento de software – Parte 05: Prototype**. Disponível em: https://imasters.com.br/desenvolvimento/arquitetura-e-desenvolvimento-de-software-parte-05-prototype. Acesso em: 23 maio 2023.

REFACTORING.GURU. **Prototype**. Disponível em: https://refactoring.guru/pt-br/design-patterns/prototype. Acesso em: 23 maio 2023.

REFACTORING.GURU. **Builder**. Disponível em: https://refactoring.guru/pt-br/design-patterns/builder. Acesso em: 23 maio 2023.