**Canva Presentation:**

<https://www.canva.com/design/DAFjOVKKMoI/EG5ek5oRYkUWAyJ1iiKBIg/edit?utm_content=DAFjOVKKMoI&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>

**Reflexão Computacional (conceito geral):**

* [https://pt.wikipedia.org/wiki/Reflexão\_(programação)#:~:text=Em%20ciência%20da%20computação%2C%20reflexão,(em%20tempo%20de%20compilação)](https://pt.wikipedia.org/wiki/Reflex%C3%A3o_(programa%C3%A7%C3%A3o)#:~:text=Em%20ci%C3%AAncia%20da%20computa%C3%A7%C3%A3o%2C%20reflex%C3%A3o,(em%20tempo%20de%20compila%C3%A7%C3%A3o)).

**Texto base 1:**

• Reflection. This is the ability of a program to manipulate data on itself or its execution environment or state. Reflective programs can adjust their behavior based on their context.

O padrão arquitetural Reflection (Reflexão) é uma abordagem amplamente utilizada em sistemas distribuídos para permitir a **introspecção e adaptação dinâmica** do sistema em tempo de execução. Ele fornece mecanismos que permitem que os componentes de um sistema distribuído **examinem e modifiquem sua própria estrutura interna e comportamento**, bem como o comportamento de outros componentes.

O objetivo principal da reflexão é fornecer flexibilidade e adaptabilidade aos sistemas distribuídos, permitindo que eles se ajustem às mudanças nas necessidades e condições do ambiente em que estão sendo executados. Isso é especialmente relevante em cenários em que os **requisitos do sistema podem mudar com o tempo ou em que os componentes do sistema precisam se adaptar a diferentes contextos de execução.**

Existem várias técnicas e mecanismos associados ao padrão Reflection em sistemas distribuídos. Aqui estão alguns dos principais elementos desse padrão:

1. **Metadados**: A reflexão depende fortemente do uso de metadados, que são informações sobre os componentes e estruturas do sistema. Os metadados descrevem as características, propriedades e comportamentos dos componentes, permitindo que sejam examinados e manipulados em tempo de execução.
2. **Introspecção**: A introspecção é a capacidade de um componente examinar seus próprios metadados e obter informações sobre sua estrutura interna, propriedades e comportamento. Por meio da introspecção, um componente pode descobrir quais métodos, atributos e interfaces estão disponíveis, bem como suas características e tipos.
3. **Modificação dinâmica**: O padrão Reflection também permite a modificação dinâmica dos componentes em tempo de execução. Isso significa que um componente pode alterar seu comportamento, adicionar ou remover métodos, atributos ou interfaces, ou até mesmo criar novos componentes durante a execução. Isso fornece uma grande flexibilidade e adaptabilidade aos sistemas distribuídos, permitindo que eles se ajustem às mudanças e necessidades em tempo real.
4. **Descoberta e acesso dinâmico**: A reflexão facilita a descoberta e o acesso dinâmico aos componentes em um sistema distribuído. Os componentes podem ser localizados e acessados em tempo de execução com base em seus metadados e características. Isso permite uma comunicação dinâmica entre os componentes, possibilitando a cooperação e a interação flexível entre eles.
5. **Extensibilidade**: O padrão Reflection promove a extensibilidade dos sistemas distribuídos, permitindo que novos componentes sejam adicionados de forma dinâmica durante a execução. Os componentes existentes podem interagir e cooperar com os novos componentes, sem a necessidade de modificação ou recompilação do sistema como um todo.
6. **Suporte a adaptabilidade**: A reflexão é particularmente útil para fornecer suporte à adaptabilidade em sistemas distribuídos. Por meio da introspecção e modificação dinâmica, os componentes podem ajustar seu comportamento com base em informações do ambiente, condições de execução ou mudanças nos requisitos do sistema. Isso permite que o sistema distribuído seja mais flexível e capaz de se adaptar a diferentes situações.

Em resumo, o padrão Reflection de sistemas distribuídos permite a introspecção e a modificação dinâmica dos componentes em tempo de execução, proporcionando flexibilidade, adaptabilidade e extensibilidade aos sistemas distribuídos. Ele promove a descoberta e o acesso dinâmico aos componentes, bem como suporta a cooperação e a interação flexível entre eles. Esse padrão é uma ferramenta poderosa para projetar sistemas distribuídos que possam se ajustar e se adaptar às mudanças em tempo real.

**Texto base 2:**

**O padrão reflection permite que componentes distribuídos de um sistema interajam e se adaptem dinamicamente sem a necessidade de conhecimento prévio sobre a estrutura ou localização desses componentes.**

Características do Padrão Reflection:

1. Transparência: O padrão Reflection promove a transparência ao ocultar a complexidade da comunicação e interação entre componentes distribuídos.
2. Flexibilidade: Os componentes distribuídos podem ser adicionados, removidos ou modificados sem impactar os demais componentes do sistema.
3. Descoberta Dinâmica: Os componentes podem ser descobertos dinamicamente em tempo de execução, permitindo que o sistema se adapte a mudanças na sua topologia.

**Componentes do Padrão Reflection:**

O padrão Reflection envolve três componentes principais:

* Reflectors: São componentes distribuídos que fornecem uma interface comum para interagir com outros componentes.
* Reflectees: São componentes distribuídos que se registram com os Reflectors para se tornarem visíveis e acessíveis pelo sistema.
* Registry: É responsável por rastrear e armazenar as informações dos Reflectees registrados e disponíveis para interação.

**Fluxo de Trabalho do Padrão Reflection:**

Um Reflectee registra-se no Registry.

O Reflectee torna-se visível e acessível para outros componentes através do Reflectors.

Outros componentes podem interagir com o Reflectee usando o Reflectors, sem conhecer sua localização ou estrutura interna.

O Reflectee pode ser atualizado, removido ou adicionado dinamicamente sem afetar os outros componentes do sistema.

**Considerações Finais:**

O padrão Reflection é um poderoso padrão arquitetural para sistemas distribuídos que promove a transparência, flexibilidade e adaptabilidade.

Ele simplifica a interação e comunicação entre componentes distribuídos, permitindo que sejam adicionados, removidos ou atualizados dinamicamente sem afetar o restante do sistema.

Ao aplicar o padrão Reflection, os desenvolvedores podem construir sistemas distribuídos mais robustos, escaláveis e fáceis de manter.

A compreensão e aplicação adequada desse padrão podem ajudar a impulsionar a eficiência e a agilidade dos sistemas distribuídos.

**Desvantagens:**

1. Desempenho: A utilização intensiva da reflexão pode impactar negativamente o desempenho do sistema distribuído. As operações de introspecção e modificação dinâmica podem exigir um processamento adicional, resultando em uma redução da velocidade de execução.
2. Complexidade: O uso da reflexão pode aumentar a complexidade do código e da arquitetura do sistema distribuído. A introspecção e a modificação dinâmica exigem uma compreensão aprofundada dos metadados e do funcionamento interno dos componentes, o que pode tornar o código mais difícil de ser compreendido e mantido.
3. Erros em tempo de execução: Como a reflexão permite a manipulação do sistema em tempo de execução, erros podem ocorrer durante as operações de introspecção e modificação dinâmica. Se as alterações forem feitas de maneira incorreta, podem surgir problemas de consistência ou erros que podem ser difíceis de depurar.
4. Segurança: A reflexão também pode apresentar riscos de segurança. A capacidade de modificar dinamicamente a estrutura e o comportamento dos componentes em tempo de execução pode ser explorada por usuários maliciosos para realizar ações não autorizadas ou comprometer a integridade do sistema.

**Exemplos:**

Suponha que estamos desenvolvendo um sistema distribuído para gerenciar uma loja online. Temos vários componentes, como o carrinho de compras, o catálogo de produtos e o sistema de pagamento. Agora, vamos considerar o uso do padrão Reflection nesse contexto.

Introspecção:

1. Vamos supor que o componente do carrinho de compras deseja examinar seus próprios métodos e atributos em tempo de execução. Utilizando o padrão Reflection, o carrinho de compras pode acessar seus metadados e descobrir quais métodos estão disponíveis, como adicionar um item ao carrinho, remover um item ou calcular o valor total da compra.

Modificação dinâmica:

1. Digamos que durante a execução do sistema, o catálogo de produtos precise ser atualizado com novos itens. Utilizando o padrão Reflection, o componente do catálogo de produtos pode modificar dinamicamente sua estrutura interna para incluir os novos itens. Ele pode criar novos métodos e atributos para lidar com as operações relacionadas aos novos produtos sem exigir uma parada no sistema ou uma modificação no código-fonte.

Descoberta e acesso dinâmico:

1. Suponha que, durante a execução, o componente do carrinho de compras precise acessar o sistema de pagamento para processar o pagamento de uma compra. Utilizando o padrão Reflection, o carrinho de compras pode descobrir dinamicamente os métodos e interfaces disponíveis no sistema de pagamento, com base em seus metadados. Ele pode acessar esses componentes de pagamento de forma dinâmica, facilitando a comunicação e a interação entre os componentes.

Exemplo 1: Introspecção no Carrinho de Compras

* Reflectors: O carrinho de compras atua como o reflector nesse exemplo. Ele usa os recursos de reflexão para examinar seus próprios métodos e atributos em tempo de execução.
* Reflectees: O carrinho de compras é o reflectee, pois é o componente que está sendo refletido e examinado.
* Registry: Neste exemplo, não há necessidade de um registry específico. O carrinho de compras pode acessar suas próprias informações de metadados internamente, sem depender de um registro externo.

Exemplo 2: Modificação Dinâmica no Catálogo de Produtos

* Reflectors: O catálogo de produtos age como o reflector nesse caso. Ele utiliza os recursos de reflexão para modificar dinamicamente sua estrutura interna, adicionando novos itens.
* Reflectees: Os novos itens que estão sendo adicionados ao catálogo de produtos são os reflectees. Eles são os objetos que estão sendo refletidos e modificados pelo catálogo.
* Registry: Nesse exemplo, um registry pode ser usado para rastrear e gerenciar os itens adicionados dinamicamente ao catálogo de produtos. O registry pode ser responsável por armazenar informações sobre os itens, como nome, preço e disponibilidade.

Exemplo 3: Descoberta e Acesso Dinâmico ao Sistema de Pagamento

* Reflectors: O carrinho de compras atua como o reflector nesse exemplo. Ele usa a reflexão para descobrir e acessar dinamicamente os componentes do sistema de pagamento.
* Reflectees: O sistema de pagamento é o reflectee nesse caso, pois é o componente que está sendo refletido e acessado dinamicamente.
* Registry: Um registry seria utilizado para rastrear e armazenar informações sobre os componentes do sistema de pagamento. Ele forneceria um mecanismo para o carrinho de compras descobrir e acessar os objetos relevantes, como métodos de pagamento específicos.

Em resumo, nos exemplos dados, o reflector é o componente que utiliza a reflexão para examinar, modificar ou descobrir informações sobre outros componentes (reflectees). Em alguns casos, pode ser necessário utilizar um registry para armazenar metadados e informações relevantes sobre os componentes disponíveis no sistema distribuído. Esses padrões de interação entre reflectors, reflectees e registrys são fundamentais para a aplicação efetiva do padrão Reflection em sistemas distribuídos.

**Bibliografia:**

* **"Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing" de Frank Buschmann, Kevlin Henney e Douglas C. Schmidt.**

Este livro aborda padrões arquiteturais para sistemas distribuídos, incluindo o padrão Reflection, fornecendo insights sobre sua aplicação e benefícios nesse contexto.

[Wiley.Pattern.Oriented.Software.Architecture.Volume.4.May.2007.pdf](https://drive.google.com/file/d/1QmCDJm9FETDSHLM65MQrT2rqfj-KoOHJ/view?usp=share_link)

* **"Distributed Systems: Concepts and Design" de George Coulouris, Jean Dollimore e Tim Kindberg.**

Este livro abrange conceitos e design de sistemas distribuídos, incluindo uma discussão sobre reflexão e sua relevância para sistemas distribuídos.

[George-Coulouris-Distributed-Systems-Concepts-and-Design-5th-Edition.pdf](https://drive.google.com/file/d/15jJMJYQDW2jqogDkADPoAYXAUuFyhXNm/view?usp=share_link)

* **"Software Architecture in Practice" de Len Bass, Paul Clements e Rick Kazman.**

Embora não seja exclusivamente dedicado ao padrão Reflection, este livro aborda várias técnicas e padrões arquiteturais, incluindo discussões sobre reflexão e sua aplicação em sistemas distribuídos.

[2013 - Book - Bass Kazman-Software Architecture in Practice.pdf](https://drive.google.com/file/d/1cR9aBrxGchSYCBNkEFpQ8e1AGKA963G7/view?usp=share_link)

**CODE EXAMPLE:**

Neste exemplo, temos três componentes:

1. **Reflectees:** As classes Car e Bike são as classes que podem ser refletidas. Cada uma delas possui um método específico (startEngine() e pedal(), respectivamente).
2. **Reflector:** A classe Reflector é responsável pela reflexão. Ela possui um registro (registry) que mapeia nomes de classes (car, bike) para suas respectivas classes (Car, Bike). O método register() é usado para registrar as classes no registry, e o método createInstance() utiliza a reflexão para criar instâncias das classes registradas.
3. **Exemplo de utilização:** No método main() da classe Main, criamos uma instância do Reflector e registramos as classes Car e Bike. Em seguida, usamos o Reflector para criar instâncias das classes registradas e chamamos seus métodos específicos.

**Ao executar esse código, você verá a saída:**

*Engine started*

*Pedaling*

Isso demonstra que as instâncias das classes Car e Bike foram criadas com sucesso usando reflexão e seus métodos específicos foram invocados.