

The background is a light brown color with a repeating geometric pattern of triangles and dots. In the center is a spiral-bound notebook with a dark brown cover and a white page with vertical yellow lines. The notebook has a silver spiral binding at the top. A blue paperclip is attached to the left edge of the notebook. A yellow sticky note is attached to the right edge, and an orange sticky note is attached to the bottom right corner. The title "Padrões de Projeto Proxy e Facade" is written in the center of the notebook page. The words "Padrões de" and "Projeto" are in a dark blue font, while "Proxy e Facade" is in a dark red font.

Padrões de Projeto Proxy e Facade

Graciely Duarte Dias

Introdução

Projetar software reutilizável orientado a objetos é mais complicado do que projetar software orientado a objetos. Isso ocorre devido à necessidade de identificar objetos, definir classes, interfaces e hierarquias de herança. Projetistas experientes utilizam soluções que funcionaram no passado, resultando em padrões frequentes em sistemas orientados a objetos. Esses padrões tornam os projetos flexíveis e reutilizáveis, permitindo que decisões de projeto sejam tomadas automaticamente com base no conhecimento desses padrões.



Tabela de Conteúdos Abordados

01

Introdução aos
padrões de
projeto

02

Categorias dos
padrões de
projeto

03

Padrão de
projeto Proxy

04

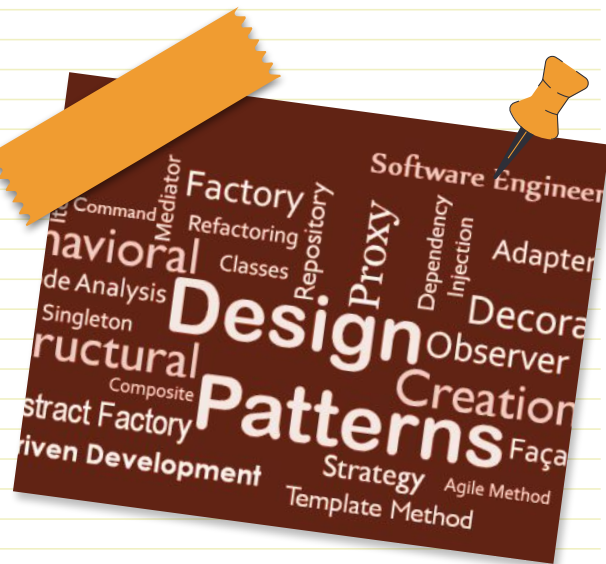
Padrão de
projeto Facade

05

Conclusão

01

Introdução aos padrões de projeto





Quando surgiu a ideia de padrão na computação?

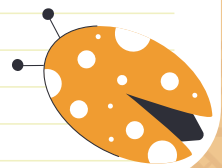
Em 1994, os autores Erich Gamma, John Vlissides, Ralph Johnson e Richard Helm publicaram o livro "Padrões de Projeto - Soluções Reutilizáveis de Software Orientado a Objetos", apresentando 23 padrões para problemas de projeto orientado a objetos. Conhecido como "o livro GoF" (Gang of Four), tornou-se um best-seller e impulsionou a popularidade da abordagem por padrões.

O que é padrões de projeto?

Christopher Alexander afirma: “cada padrão descreve um problema no nosso ambiente e o núcleo da sua solução, de tal forma que você possa usar esta solução mais de um milhão de vezes, sem nunca fazê-lo da mesma maneira”

Em geral, um padrão tem quatro elementos essenciais:

- O nome do projeto facilita a reflexão sobre o projeto, bem como a comunicação de seus custos, benefícios e outros aspectos para outras pessoas.
- O problema define o momento adequado para aplicar um padrão, descrevendo o contexto e o problema em si.
- A solução descreve os componentes do projeto, seus relacionamentos, responsabilidades e colaborações.
- As consequências referem-se aos resultados e análises das vantagens e desvantagens da aplicação do padrão.



Vantagens oferecidas pela utilização dos padrões de projeto

Os padrões de projeto oferecem as seguintes vantagens:

Solução comprovada

Os padrões de projeto são soluções confiáveis, baseadas em melhores práticas da indústria, que oferecem uma abordagem segura para resolver desafios de desenvolvimento.

Reutilização de soluções

Ao usar padrões de projeto, você pode reutilizar soluções existentes, economizando tempo e esforço ao evitar a necessidade de criar tudo do zero a cada projeto.

Comunicação eficaz

Os padrões de projeto estabelecem uma terminologia comum, o que facilita a comunicação eficaz entre os membros da equipe. Isso garante que todos entendam os conceitos e soluções propostas, facilitando a discussão e colaboração.

Melhoria da qualidade do código

Os padrões de projeto levam a um código mais estruturado, modular e compreensível, resultando em software de melhor qualidade, menos complexo e mais fácil de manter.

Adaptabilidade e flexibilidade

Os padrões de projeto criam uma base sólida para sistemas flexíveis e adaptáveis, permitindo a introdução de novas funcionalidades ou alterações sem impactar o restante do sistema, graças à sua estrutura clara e modular.

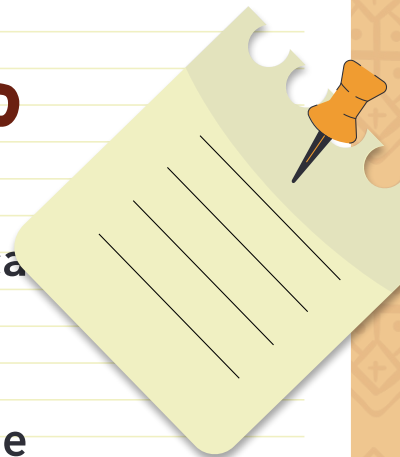


02

Categorias dos padrões de projeto

Categorias dos padrões de projeto

Os padrões de projeto variam em complexidade e escala de aplicabilidade. Os padrões idiomáticos são de baixo nível e aplicáveis a uma única linguagem, enquanto os padrões arquitetônicos são universais e utilizados na arquitetura completa de uma aplicação.



Categorias dos padrões de projeto

Criacionais

Os padrões criacionais fornecem mecanismos de criação de objetos que aumentam a flexibilidade e a reutilização de código.

Estruturais

Os padrões estruturais explicam como montar objetos e classes em estruturas maiores, enquanto ainda mantém as estruturas flexíveis e eficientes.

Comportamentais

Os padrões comportamentais cuidam da comunicação eficiente e da assinalação de responsabilidades entre objetos.

Catálogo de Padrões de Projeto



| ESCOPO | Criacionais | Estruturais | Comportamentais |
|--------|------------------|------------------|-------------------------|
| Classe | Factory Method | Adapter (classe) | Interpreter |
| | | | Template Method |
| Objeto | Abstract Factory | Adapter (objeto) | Chain of Responsibility |
| | Builder | Bridge | Command |
| | Prototype | Composite | Iterator |
| | Singleton | Decorator | Mediator |
| | | Facade | Memento |
| | | Flyweight | Observer |
| | | Proxy | State |
| | | | Strategy |
| | | | Visitor |
| | | | |

Padrões Criacionais

Factory Method

Fornecer uma interface para criar objetos em uma superclasse, mas permite que as subclasses alterem o tipo de objetos que serão criados

Abstract Factory

Permite que você produza famílias de objetos relacionados sem ter que especificar suas classes concretas

Builder

Permite a produção diferentes tipos e representações de um objeto usando o mesmo código de construção

Prototype

Permite copiar objetos existentes sem fazer seu código ficar dependente de suas classes

Singleton

Permite a você garantir que uma classe tenha apenas uma instância, enquanto provê um ponto de acesso global para essa instância

Padrões Estruturais

Adapter

Permite objetos com interfaces incompatíveis colaborarem entre si

Composite

Permite que você componha objetos em estruturas de árvores e então trabalhe com essas estruturas como se elas fossem objetos individuais

Decorator

Permite o acoplamento de novos comportamentos para objetos ao colocá-los dentro de invólucros de objetos que contêm os comportamentos

Facade

Fornece uma interface simplificada para uma biblioteca, um framework, ou qualquer conjunto complexo de classes

Bridge

Permite que você divida uma classe grande ou um conjunto de classes intimamente ligadas em duas hierarquias separadas (abstração e implementação) que podem ser desenvolvidas independentemente umas das outras.

Flyweight

Permite colocar mais objetos na quantidade de RAM disponível ao compartilhar partes comuns de estado entre os múltiplos objetos ao invés de manter todos os dados em cada objeto

Proxy

Permite o fornecimento de um substituto ou um espaço reservado para outro objeto. Um proxy controla o acesso ao objeto original, permitindo que você faça algo ou antes ou depois do pedido chegar ao objeto original.

Padrões Comportamentais

Interpreter

Define representações para gramáticas e abstrações para análise sintática

Command

Encapsula uma mensagem como um objeto, de modo que se possa parametrizar clientes com diferentes mensagens

Strategy

Define uma família de algoritmos, encapsula cada um deles, e torna a escolha de qual usar flexível. O Strategy desacopla os algoritmos dos clientes que os usa.

Template Method

Define o esqueleto de um algoritmo em uma operação. Permitindo que subclasses componham o algoritmo

Iterator

Provê um modo de acesso a elementos de um agregado de objetos, sequencialmente, sem exposição de estruturas internas

State

Deixa um objeto mudar seu comportamento quando seu estado interno muda, mudando, efetivamente, a classe do objeto.

Chain of Responsibility

Encadeia os objetos receptores e transporta a mensagem através da corrente até que um dos objetos responda

Mediator

Define um objeto que encapsula as interações dentre desse grupo

Observer

Provê sincronização, coordenação e consistência entre objetos relacionados

Visitor

Representa uma operação a ser realizada sobre elementos da estrutura de um objeto

Memento

Captura e externaliza o estado interno de um objeto



03

Padrão de projeto Proxy

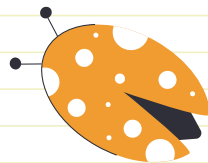
Proxy (Surrogate)

Objetivo

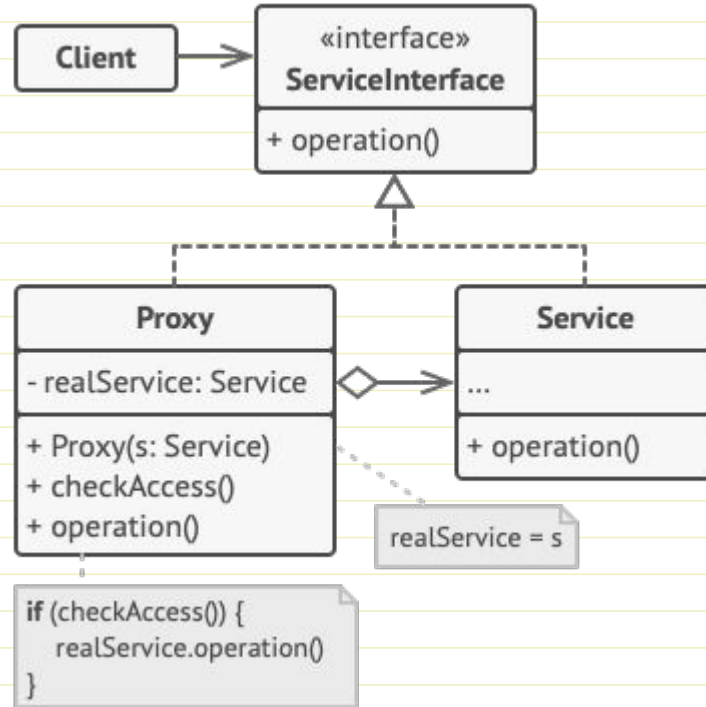
O padrão Proxy atua como um representante do objeto real, controlando o acesso e adicionando funcionalidades extras, sem expor diretamente o objeto original ao cliente.

Aplicabilidade

O padrão Proxy é usado para obter uma referência mais versátil a um objeto. Ele intercepta solicitações do cliente, adiciona ações extras, como cache e controle de acesso, e resolve problemas de desempenho e tarefas auxiliares. O Proxy restringe o acesso direto ao objeto real e adiciona funcionalidades adicionais, sendo amplamente aplicado para melhorar o controle, desempenho e funcionalidade de um sistema.



Proxy (Surrogate) - Estrutura



Proxy (Surrogate)

Conceitos envolvidos

O padrão Proxy envolve encapsulamento, interface e delegação. Ele encapsula a complexidade da interação com o objeto real, fornecendo uma interface simplificada e preservando sua funcionalidade principal.

Consequências

O padrão Proxy otimiza a cópia de objetos complexos através do copy-on-write. Ele adia a cópia até que seja necessária, reduzindo o consumo de recursos. Ao utilizar a contagem de referências, o Proxy determina quando copiar o objeto e diminui a contagem quando não é mais necessário. Isso resulta em uma redução significativa no custo da cópia de objetos pesados.

Proxy (Surrogate)

Prós

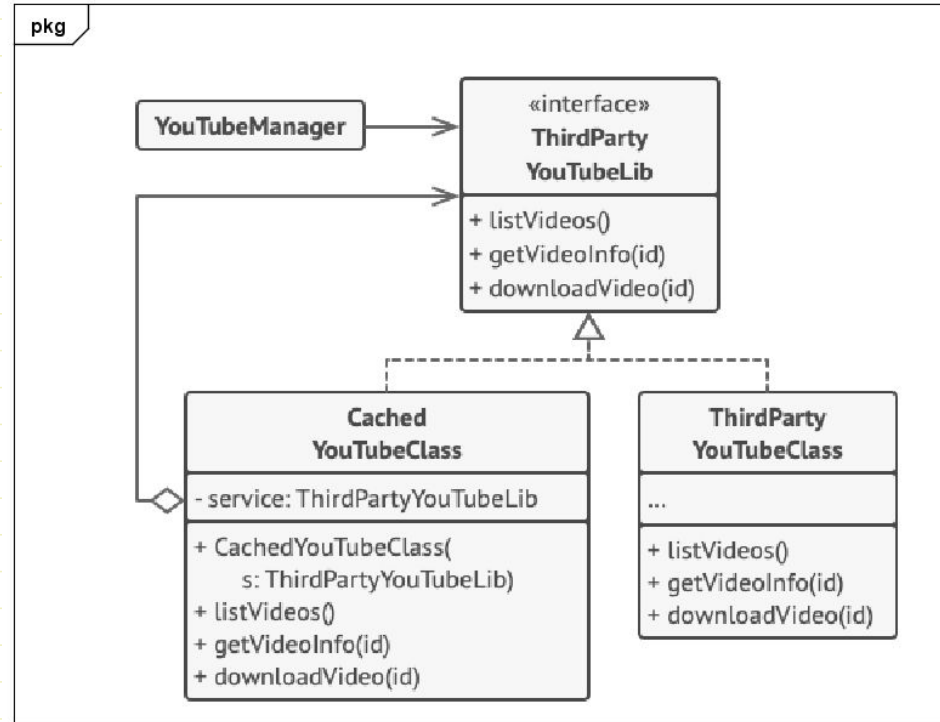
- ✓ Você pode controlar o objeto do serviço sem os clientes ficarem sabendo.
- ✓ Você pode gerenciar o ciclo de vida de um objeto do serviço quando os clientes não se importam mais com ele.
- ✓ O proxy trabalha até mesmo se o objeto do serviço ainda não está pronto ou disponível.
- ✓ Princípio aberto/fechado. Você pode introduzir novos proxies sem mudar o serviço ou clientes.

Contras

- ✗ O código pode ficar mais complicado uma vez que você precisa introduzir uma série de novas classes.
- ✗ A resposta de um serviço pode ter atrasos.



Proxy (Surrogate)





04 Padrão de projeto Facade

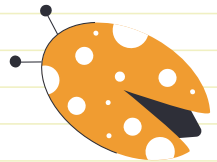
Facade

Objetivo

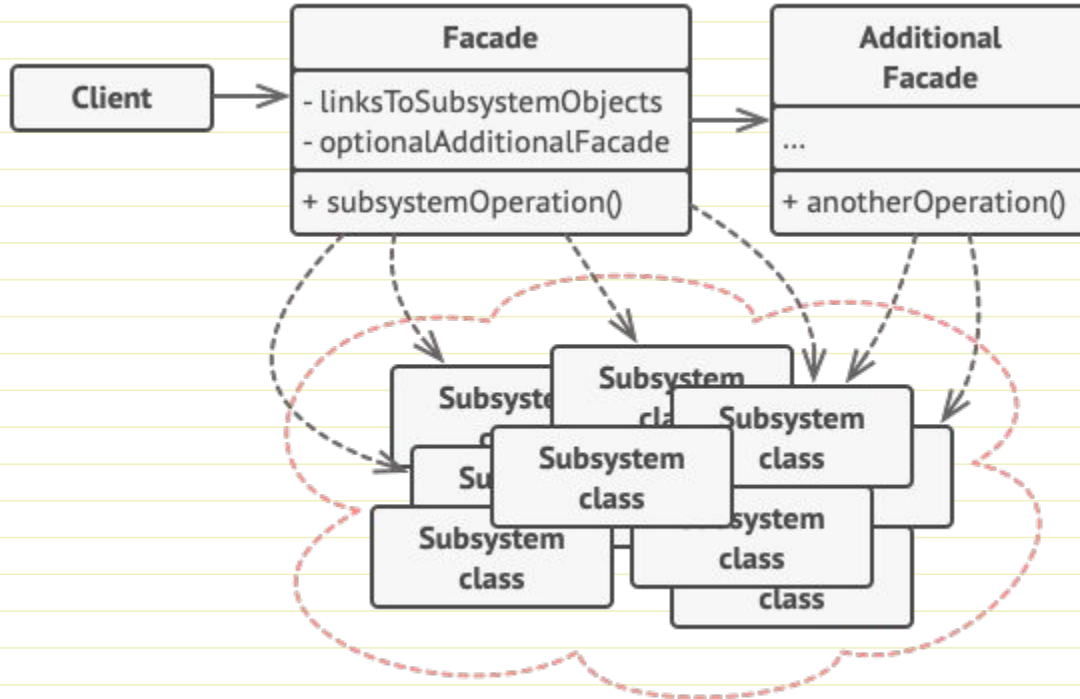
O padrão Facade simplifica o uso de um subsistema complexo, fornecendo uma interface unificada e de nível mais alto.

Aplicabilidade

O padrão Facade simplifica a interação com um subsistema complexo, encapsulando sua lógica interna e oferecendo uma interface amigável aos clientes. Ele melhora a modularidade, desacoplamento e legibilidade do código, facilitando a manutenção e evolução do sistema.



Facade - Estrutura



Facade

Conceitos envolvidos

O padrão Facade envolve encapsulamento, abstração e composição. Ele atua como uma fachada para o subsistema, fornecendo uma interface simplificada e isolando os clientes da complexidade interna.

Consequências

O padrão Facade oferece benefícios como:

- Isolamento dos clientes dos componentes do subsistema, simplificando o uso e reduzindo interações.
- Acoplamento fraco entre o subsistema e seus clientes, permitindo variações sem afetar os clientes e eliminando dependências complexas.
- Flexibilidade para utilizar classes do subsistema, oferecendo facilidade de uso e generalização.

Facade



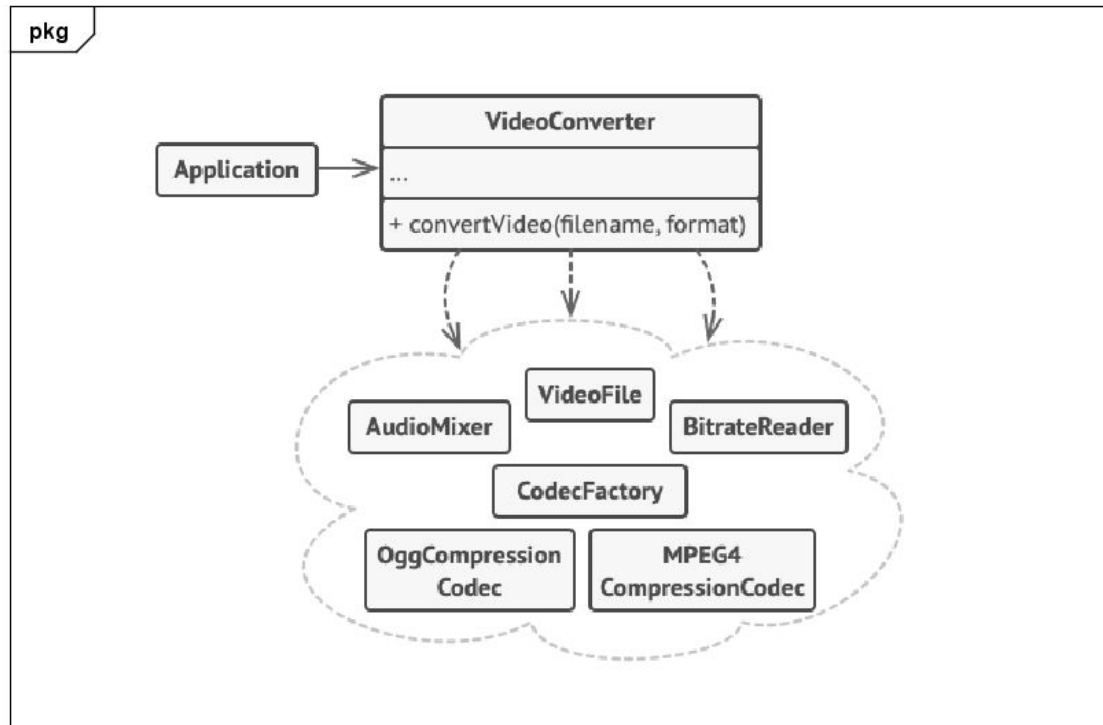
Prós

✓ Você pode isolar seu código da complexidade de um subsistema.

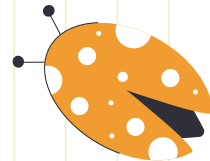
Contras

✗ Uma fachada pode se tornar um objeto deus acoplado a todas as classes de uma aplicação.

Facade



05 Conclusão



Conclusão

O uso de padrões de projeto oferece flexibilidade, documenta soluções reutilizáveis e simplifica o desenvolvimento e o reaproveitamento de código.

Os padrões Proxy e Facade são amplamente utilizados no desenvolvimento de software para resolver problemas comuns, melhorar a estrutura dos sistemas e promover a modularidade, reusabilidade e flexibilidade do código. Esses padrões fornecem abstrações úteis e estratégias eficazes, resultando em sistemas mais robustos e fáceis de evoluir.

Material complementar

Referências complementares para aprofundamento

Vídeos

- Padrões de Projeto (Design Patterns - GoF) - Introdução - Parte 1/45
- Padroes de Projeto



Referências

Referências consultadas para embasar a elaboração da apresentação

Livros

- GAMMA, Erich. Padrões de projetos: soluções reutilizáveis. Bookman editora, 2009.
- SHVETS, Alexander. Mergulho nos Padrões de Projeto. Refactoring.Guru, 2021.

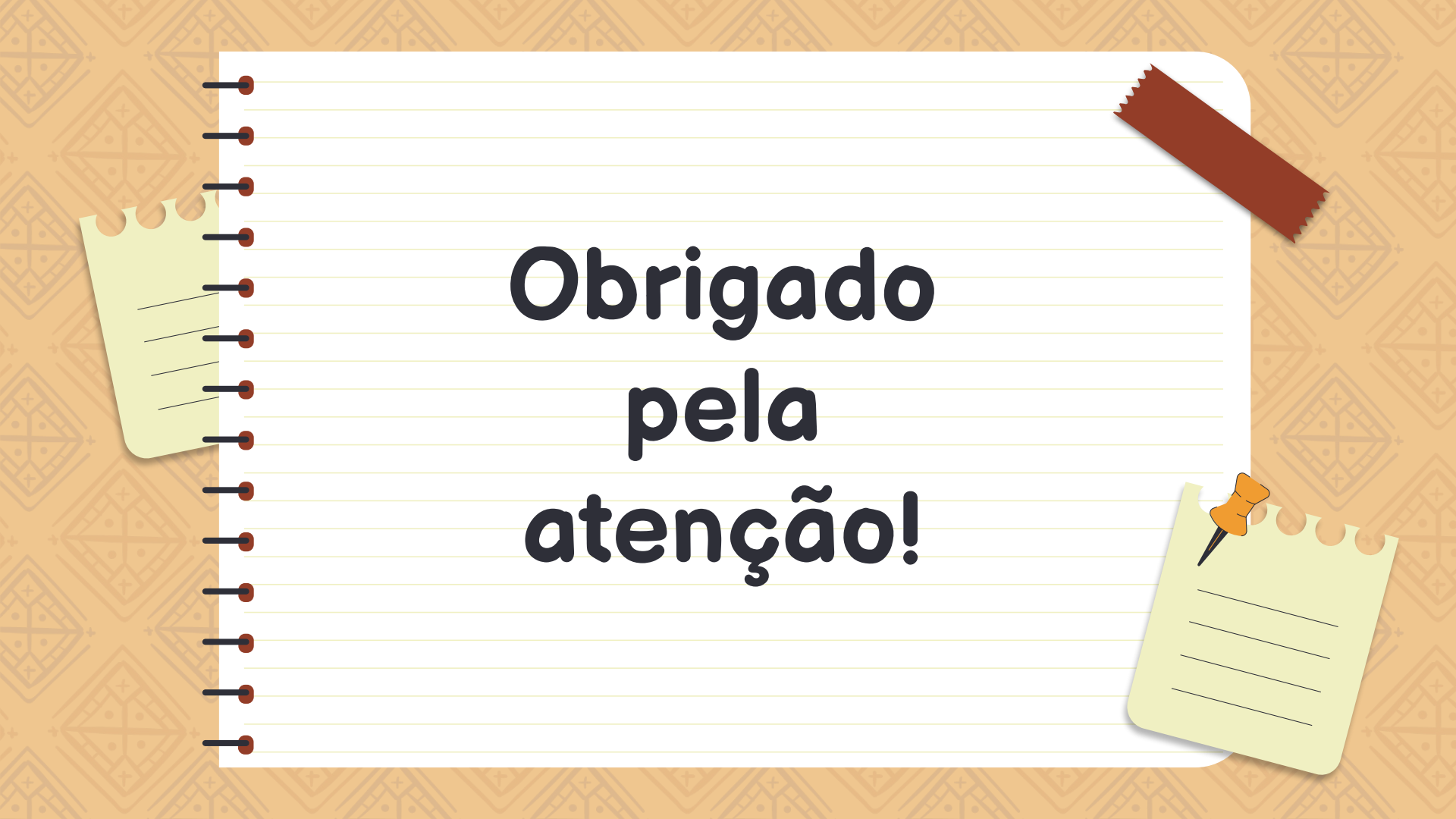
Documentos

- DESIGN PATTERNS - PADRÕES DE PROJETO
- Uma Introdução aos Padrões de Projeto com Java
-

Sites

- Conheça os Padrões de Projeto
- Padrões de Projeto / Design patterns
- Proxy
- Facade





**Obrigado
pela
atenção!**