Design Patterns

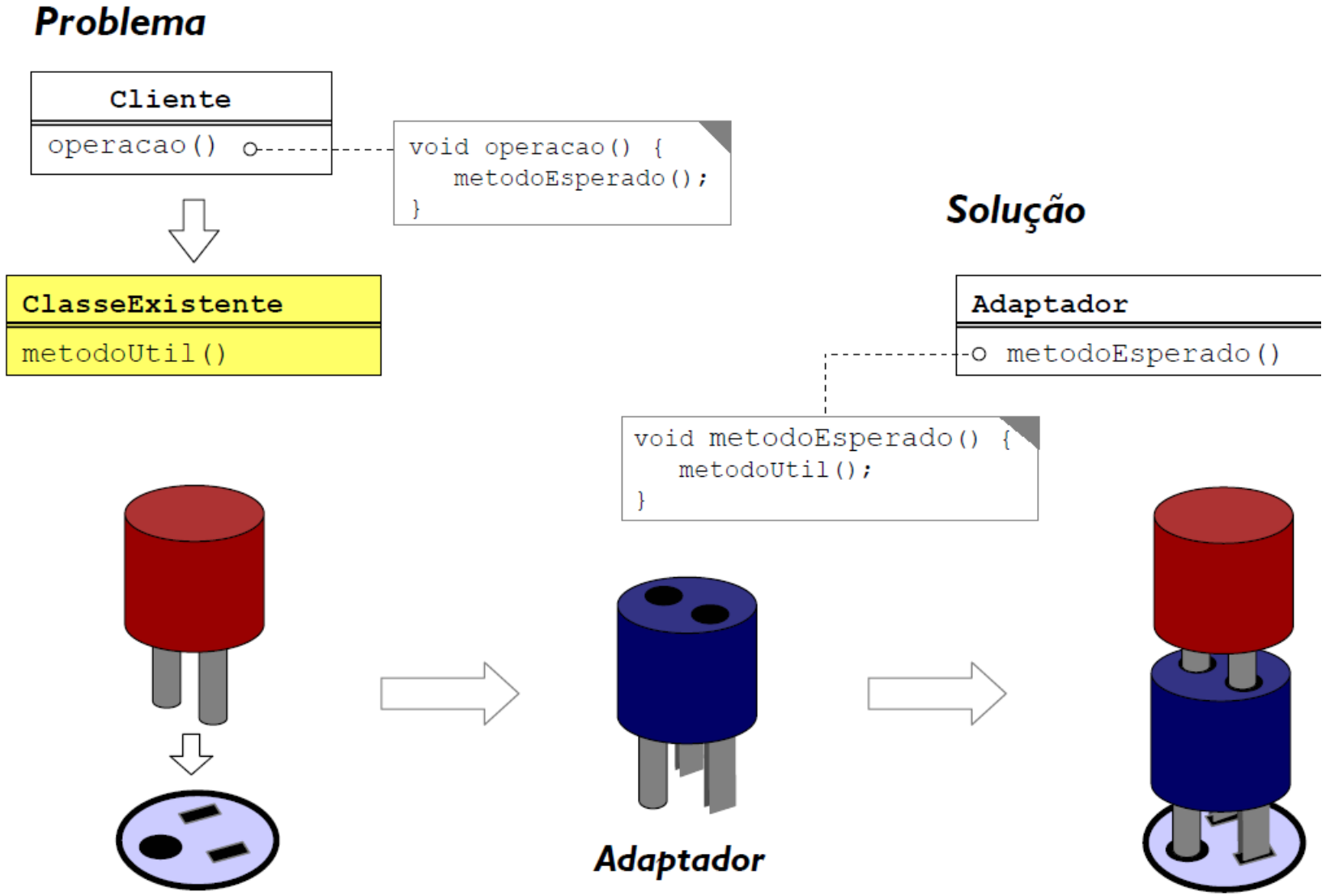
Classificação (GoF)

		Propósito		
		1. Criação	2. Estrutura	3. Comportamento
Escopo	Classe	Factory Method	Class Adapter	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Object Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

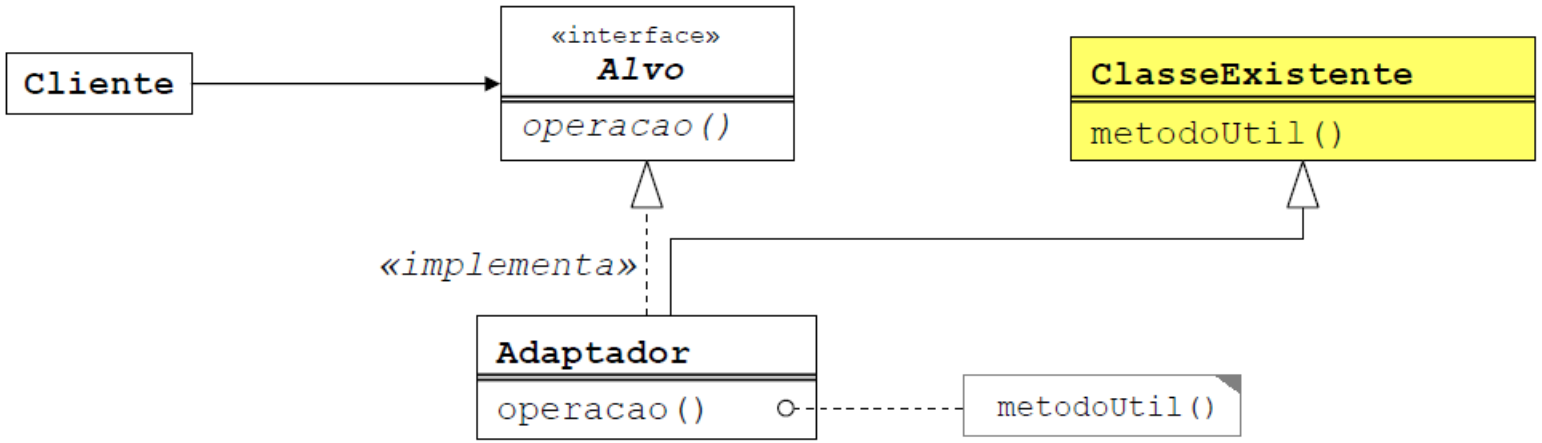
Class Adapter

"Objetivo: converter a interface de uma classe em outra interface esperada pelos clientes. Adapter permite a comunicação entre classes que não poderiam trabalhar juntas devido à incompatibilidade de suas interfaces." [GoF]

Problema e Solução



Como fazer ?



- *Cliente: aplicação que colabora com objetos aderentes à interface Alvo*
- *Alvo: define a interface requerida pelo Cliente*
- *ClasseExistente: interface que requer adaptação*
- *Adaptador (Adapter): adapta a interface do Recurso à interface Alvo*

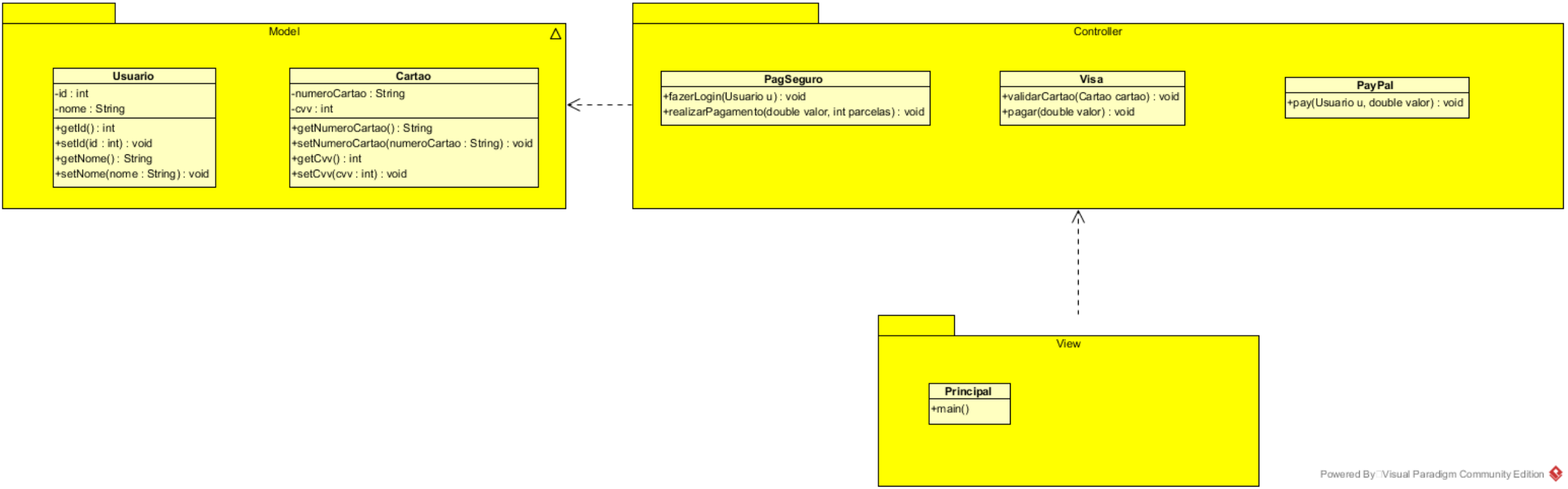
Quando usar ?

- *Sempre que for necessário adaptar uma interface para um cliente*
- *Class Adapter*
  - *Quando houver uma interface que permita a implementação estática*

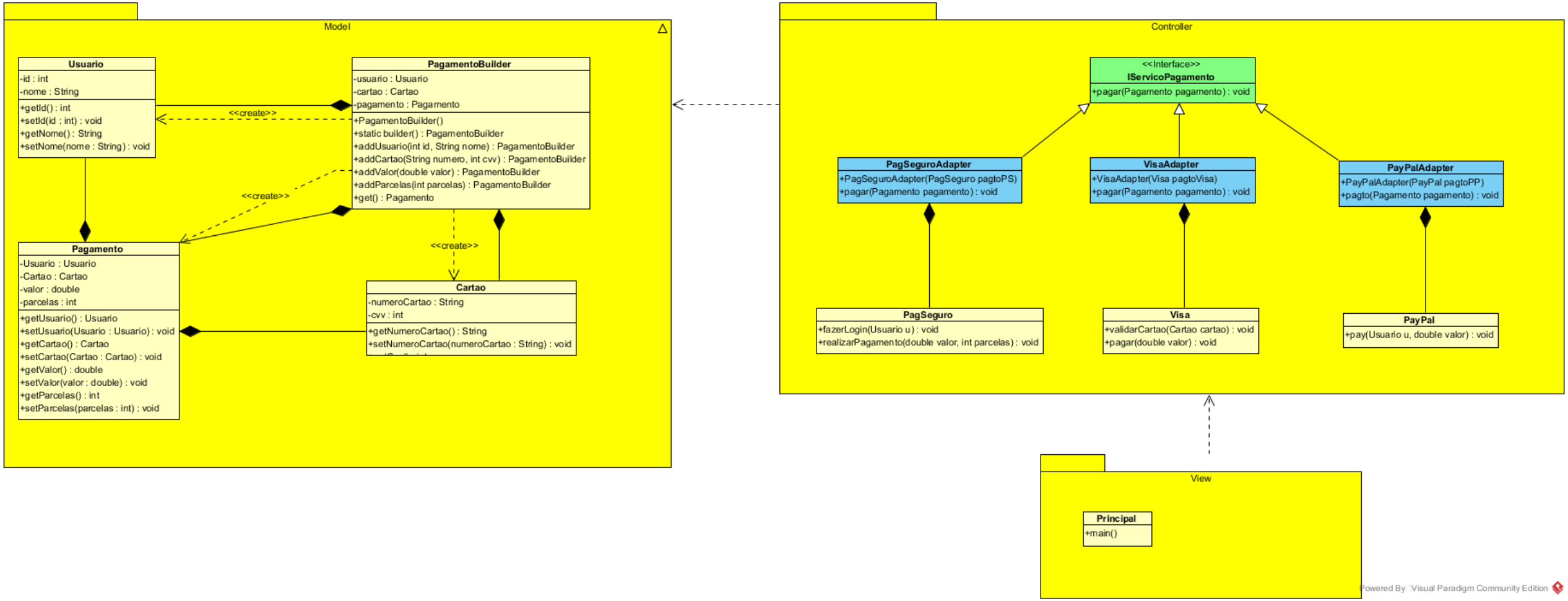
Prática e aplicação em Java

- Considere uma aplicação que deva permitir fazer pagamentos em diversas operadoras distintas, com comportamentos distintos.
  - Como exemplo, usaremos **Visa, PayPal** e **PagSeguro**
- Para fazer pagamentos no PagSeguro, é necessário fazer login do usuário e fazer o pagamento que deve receber o valor e a quantidade de parcelas
- Para fazer pagamentos no Visa, validar o cartão de crédito e pagar o valor devido
- Para fazer pagamentos no PayPal, é necessário fazer o pay passando o usuário, que será validado e o valor, na mesma operação.
- Apesar de se tratarem de operações de pagamentos, consideramos, portanto que são classes incompatíveis entre si, gerando o diagrama de pacotes a seguir

Implementar



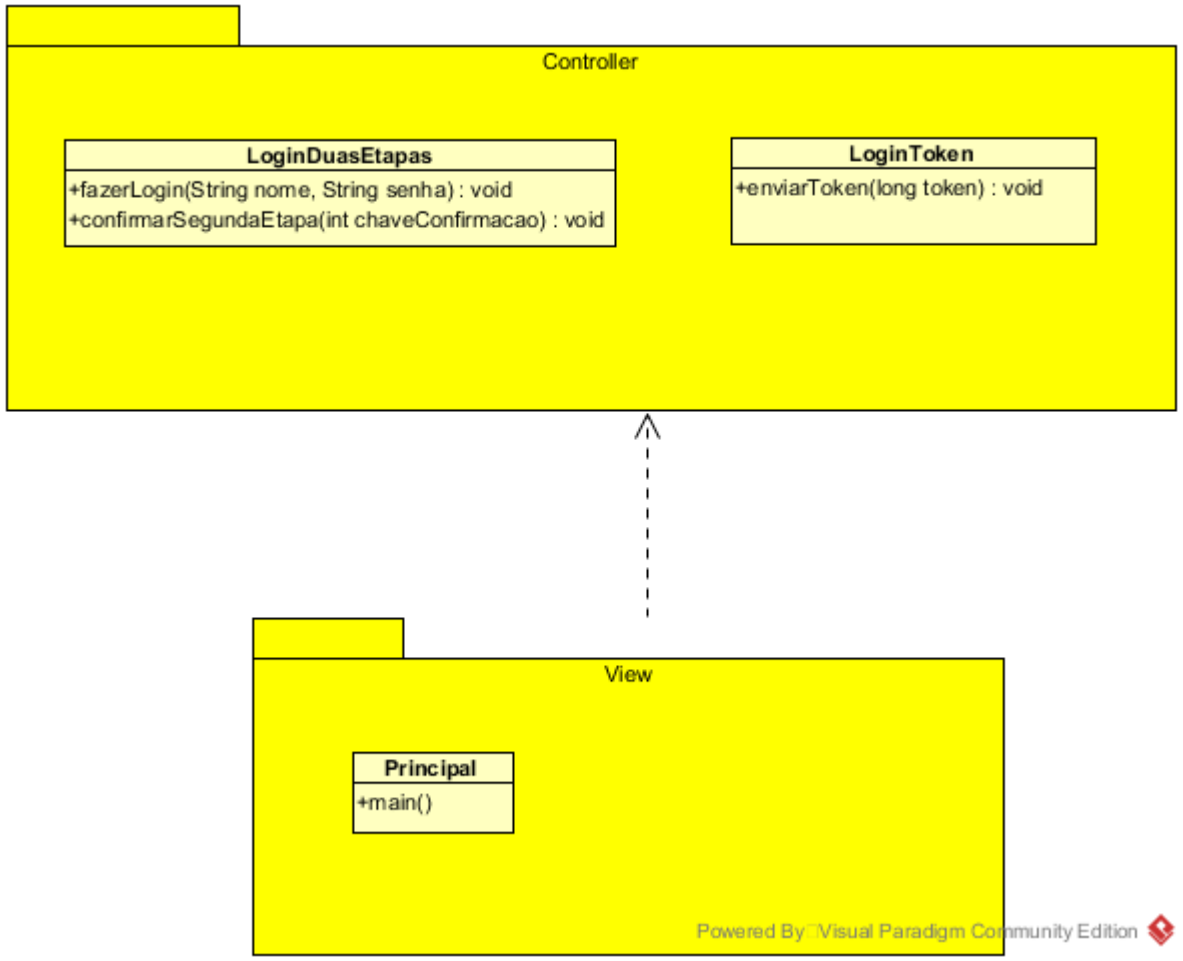
Class Adapter



Exercício

- Considere uma aplicação que permita fazer a autenticação de entrada em um outro sistema. Essa autenticação pode ser de duas formas distintas. Por login e senha, com autenticação em 2 fases ou pelo envio de um token gerado por um app.
  - Se fizer por 2 fases, deve-se fazer primeiro o login e senha e, depois mandar o código de confirmação
  - Se fizer por token, basta enviar a chave gerada
- O diagrama a seguir apresenta a primeira solução pensada
- Implementar uma solução que corrija a solução atual aplicando o Class Adapter Pattern e refazer o diagrama de pacotes
- Considere que uma autenticação pelo serviço do Google pode ser implementada e, para o google, é necessário fazer o login com usuário e senha, na sequência mandar um número recebido no smartphone e, caso o smartphone não esteja disponível, mandar um número gerado no e-mail.

Exercício


Controller
LoginDuasEtapas
+fazerLogin(String nome, String senha) : void
+confirmarSegundaEtapa(int chaveConfirmacao) : void
LoginToken
+enviarToken(long token) : void
View
Principal
+main()
Powered By Visual Paradigm Community Edition