MEDIATOR

PATRON DE DISEÑO

Nos permite disminuir las dependencias incoherentes entre objetos, restringiendo la comunicación directa entre objetos.

Solución

EL patrón Mediator nos propone, que cada componente para ser independiente debe utilizar un objeto especial mediador, este se encarga de redireccionar las llamadas a los componentes que se desea utilizar. En otras palabras, el objeto mediador se encargará de recolectar la información y validarlo según el criterio de donde se use el formulario.

Analogía al mundo real

Un ejemplo que nos proponen es una pista de aterrizaje, donde tenemos una torre central que todos los pilotos de los aviones se comunican y la torre se encarga de organizarlos y responde a cada uno. En caso de no existir una torreta los aviones deberían conectarse directamente y discutir la prioridad de aterrizaje entre ellos.

Estructura

01

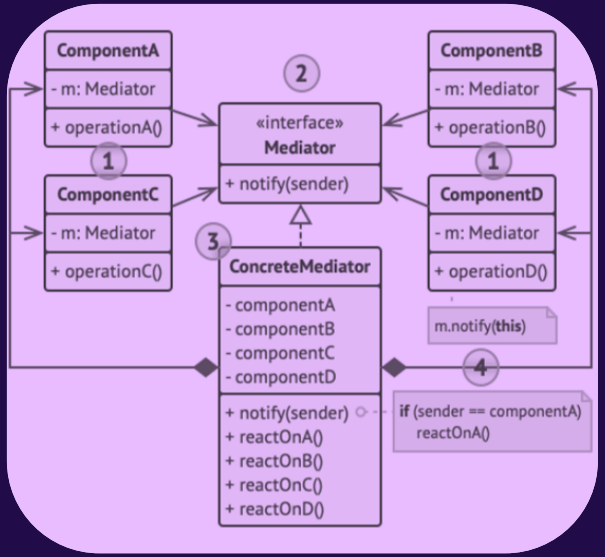
Componentes

Tiene una referencia a una interfaz mediadora.

02

Interfaz Mediadora

Declara métodos de comunicación entre componentes.



03

Mediadores Concretos

Encapsulan relaciones entre varios componentes.

04

Notificación

Se encarga de notificar a la respuesta o el mensaje

Aplicaciones

1

Resulta difícil cambiar algunas de las clases

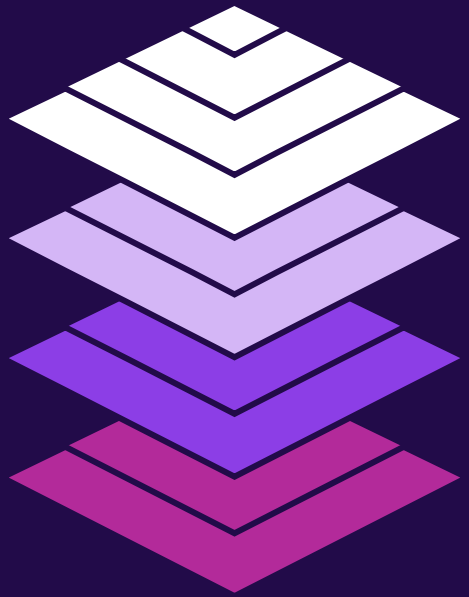
2

No se puede reutilizar un componente

3

Disponer de varias subclases de componentes

SALES AND DISTRIBUTION



- **PASO 1**
  Identifica clases fuertemente acopladas
- **PASO 2**
  Declara la interfaz mediadora
- **PASO 3**
  Implementa la clase concreta mediadora
- **PASO 4**
  Componentes deben invocar el método de notificación

PROS

- Principio de responsabilidad única
- Introducir nuevos mediadores
- Reducir el acoplamiento
- Reutilizar componentes

CONTRAS

- Un mediador puede evolucionar a un objeto todopoderoso

IMPLEMENTACION

Presentacion en JAVA

GRACIAS!

BIBLIOGRAFIA:
https://refactoring.guru/es/design-patterns/mediator/java/example