TEC | Tecnológico de Costa Rica

Patrón de diseño Mediador

Marco Rodríguez Vargas

Programación Orientada a Objetos

Analogía en el mundo real

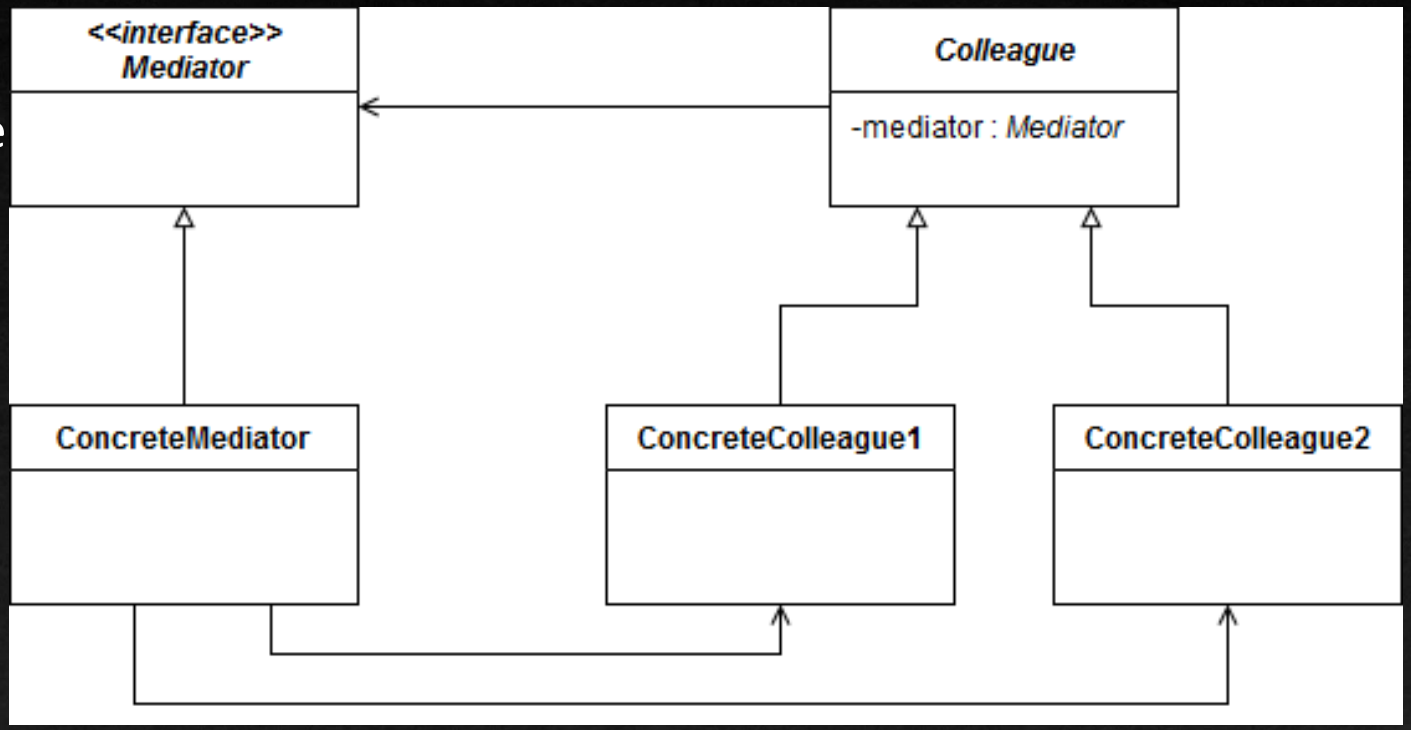
- Los pilotos de aviones no hablan directamente entre ellos para decidir quien es el siguiente en aterrizar su avión. Todas las comunicaciones pasan por la torre de control.

Intención

- Es reducir la complejidad y las dependencias entre objetos estrechamente acoplados que se comunican directamente entre sí. Esto se logra creando un objeto mediador que se encarga de la interacción entre los objetos dependientes. En consecuencia, toda la comunicación pasa por el mediador.

Diagrama UML

- El mediador define la interfaz que usan los objetos Colleague para comunicarse.
- Colleague define la clase abstracta que contiene una sola referencia al Mediador.
- Concrete Mediator encapsula la lógica de interacción entre los objetos Colleague.
- ConcreteColleague1 y ConcreteColleague2 se comunican solo a través del Mediador. De esta manera reutilizar ConcreteColleague1 y ConcreteColleague2 es mas fácil.



Aplicación

- El Patrón Mediador es una buena opción si tenemos que lidiar con un conjunto de objetos que están estrechamente acoplados y son difíciles de mantener. De esta manera podemos reducir las dependencias entre objetos y disminuir la complejidad general.

Retomando el ejemplo inicial pero en código

```
class ATCMediator implements IATCMediator
{
    private Flight flight;
    private Runway runway;
    public boolean land;

    public void registerRunway(Runway runway)
    {
        this.runway = runway;
    }

    public void registerFlight(Flight flight)
    {
        this.flight = flight;
    }

    public boolean isLandingOk()
    {
        return land;
    }

    @Override
    public void setLandingStatus(boolean status)
    {
        land = status;
    }
}

interface Command
{
    void land();
}

interface IATCMediator
{

    public void registerRunway(Runway runway);
```

```
interface IATCMediator
{

    public void registerRunway(Runway runway);

    public void registerFlight(Flight flight);

    public boolean isLandingOk();

    public void setLandingStatus(boolean status);
}

class Flight implements Command
{
    private IATCMediator atcMediator;

    public Flight(IATCMediator atcMediator)
    {
        this.atcMediator = atcMediator;
    }

    public void land()
    {
        if (atcMediator.isLandingOk())
        {
            System.out.println("Successfully Landed.");
            atcMediator.setLandingStatus(true);
        }
        else
            System.out.println("Waiting for landing.");
    }

    public void getReady()
    {
        System.out.println("Ready for landing.");
    }

}
```

```
class Runway implements Command
{
    private IATCMediator atcMediator;

    public Runway(IATCMediator atcMediator)
    {
        this.atcMediator = atcMediator;
        atcMediator.setLandingStatus(true);
    }

    @Override
    public void land()
    {
        System.out.println("Landing permission granted.");
        atcMediator.setLandingStatus(true);
    }

}

class MediatorDesignPattern
{
    public static void main(String args[])
    {

        IATCMediator atcMediator = new ATCMediator();
        Flight sparrow101 = new Flight(atcMediator);
        Runway mainRunway = new Runway(atcMediator);
        atcMediator.registerFlight(sparrow101);
        atcMediator.registerRunway(mainRunway);
        sparrow101.getReady();
        mainRunway.land();
        sparrow101.land();

    }
}
```

Salida esperada del programa

- Ready for landing.
- Landing permission granted.
- Successfully Landed.

Bibliografía

- Mediator, Refactoring Guru. https://refactoring.guru/images/patterns/diagrams/mediator/live-example.png?id=aa1de3cb7b63aa623e63578a1e43399a.
- Woyke, K. (2022) *El patron mediador en Java*, *Baeldung*. https://www.baeldung.com/java-mediator-pattern (September 19, 2022).