**Mediator Pattern**Group 07 – Dung  
References:   
Design Patterns: Elements of Reusable Object-Oriented Software - Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (Gang of Four Design Patterns)

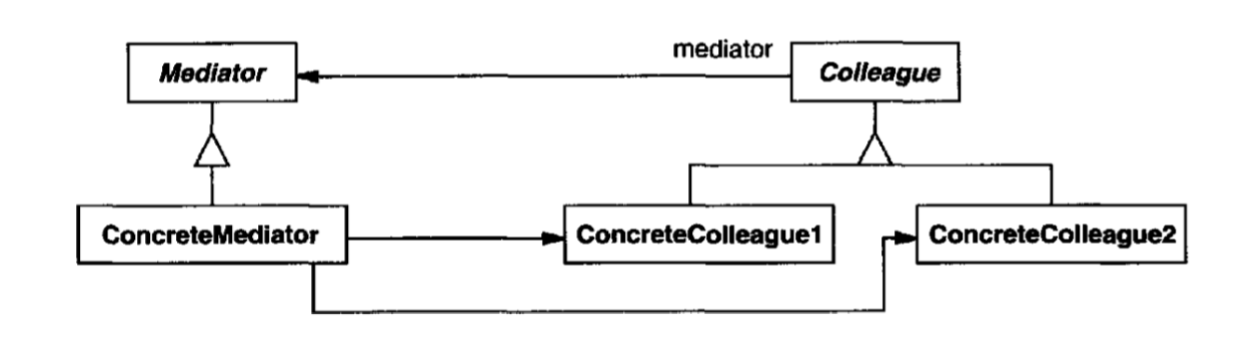
1. Example

We design a kind of “messenger” application that allows users to contact in a group. When a member makes a call to that group, all members should receive it immediately and their status should be changed to “calling”. However maybe some of them want some peace at that time, they set their status to “busy” which means no one can call them and their status remain “busy” all time, the caller may should be notified that this person can’t join now.

1. Definition

“Mediator Pattern defines an object encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.”

-Gang of Four Design Patterns



(Gang of Four Design Patterns)

* Instead of communicating directly to each other, the objects known as **Colleague** should work through a manager known as **Mediator** which encapsulates their interaction.
* The concrete mediator will manage the list of colleagues, the specific interactive methods of their colleagues.
* The concrete mediator classes inherit a mediator abstract class, while the concrete colleague classes inherit a colleague one.
* To communicate with each other, the colleague will first notify the mediator then it’s the mediator’s work to send it to the receiver.

1. Some other examples

* In a war, an army may consists of several kinds of transportation like helicopters, ships, tanks,… when the war begin or when they first come to a zone, if one of them detect an enemy, it should notify all the others about the appearance and location of the enemy.
* Air traffic controller, which the airport control room manages the communication between different planes to ensure their will be no accident (reference: Geeksforgeeks)
* Telephone routing system, when a user make a call, the router will check if the destination is free and connect each other.

1. Advantages and disadvantages

Advantages:

* It limits subclassing. A mediator localizes behavior that otherwise would be distributed among several objects. Changing this behavior requires subclassing Mediator only; Colleague classes can be reused as is.
* It decouples colleagues. A mediator promotes loos e coupling between colleagues. You can vary and reuse Colleague and Mediator classes independently.
* It simplifies object protocols. A mediator replaces many-to-many interactions with one-to-many interactions between the mediator and its colleagues. One-to-many relationships are easier to understand, maintain, and extend.
* It abstracts how objects cooperate. Making mediation an independent concept and encapsulating it in an object lets you focus on how objects interact apart from their individual behavior. That can help clarify how objects interact in a system

Disadvantages:

* It centralizes control. The Mediator pattern trades complexity of interaction for complexity in the mediator. Because a mediator encapsulates protocols, it can become more complex than any individual colleague. This can make the mediator itself a monolith that's hard to maintain.