Mediator:

**Other name (if any)**

placeholder

**What it does**

The same type of object do not interact with each other directly. They interact through another class of object which is mediator.

**Where to use**

1. Your system involves a set of objects that need to communicate with each other in a complex manner, and you want to avoid direct dependencies between them.  
2. It is commonly used in message-based systems likewise chat applications.  
3. It is used in online bidding system.  
4. It is used in airline management system.  
5. You need a centralized mechanism to coordinate and control the interactions between objects.

**Steps**

1. Colleague Interface: It will have the functions which are important for the similar objects. Like addData/sendMessage/placeBid, receiveNotification/receiveMessage, getData/getName.  
2. Concrete Colleague Class: It will have the concrete mediator object. In the addData/sendMessage/placeBid it will addData through the concrete mediator object. It will implement the functions of the colleague interface.  
3. Mediator Interface: It will have the functions like addColleague/addData and sendNotifications.  
4. Concrete Mediator Class: It will have the list of colleague interface object. It will implement the functions of the mediator interface. In the sendNotification function it will call the receiveNotification function of the concrete colleague list.  
5. Client Code: Concrete mediator object will be created using concrete mediator class. It will be passed to the constructor of concrete colleague class. Call the addData/sendMessage/placeBid function of the concrete colleague class.

**Special cases (if any)**

placeholder

**Advantages**

1. It centralizes the control.

**Disadvantages**

1. The interactions between components are straightforward, and introducing a mediator would add unnecessary complexity.  
2. Introducing a mediator could introduce a performance overhead, especially in situations where direct communication between components is more efficient.

**Code**

Coding Concept

**Difference with similar pattern**

Observer Design Pattern:  
Proxy Design Pattern:

**Diagram**

Coding Concept