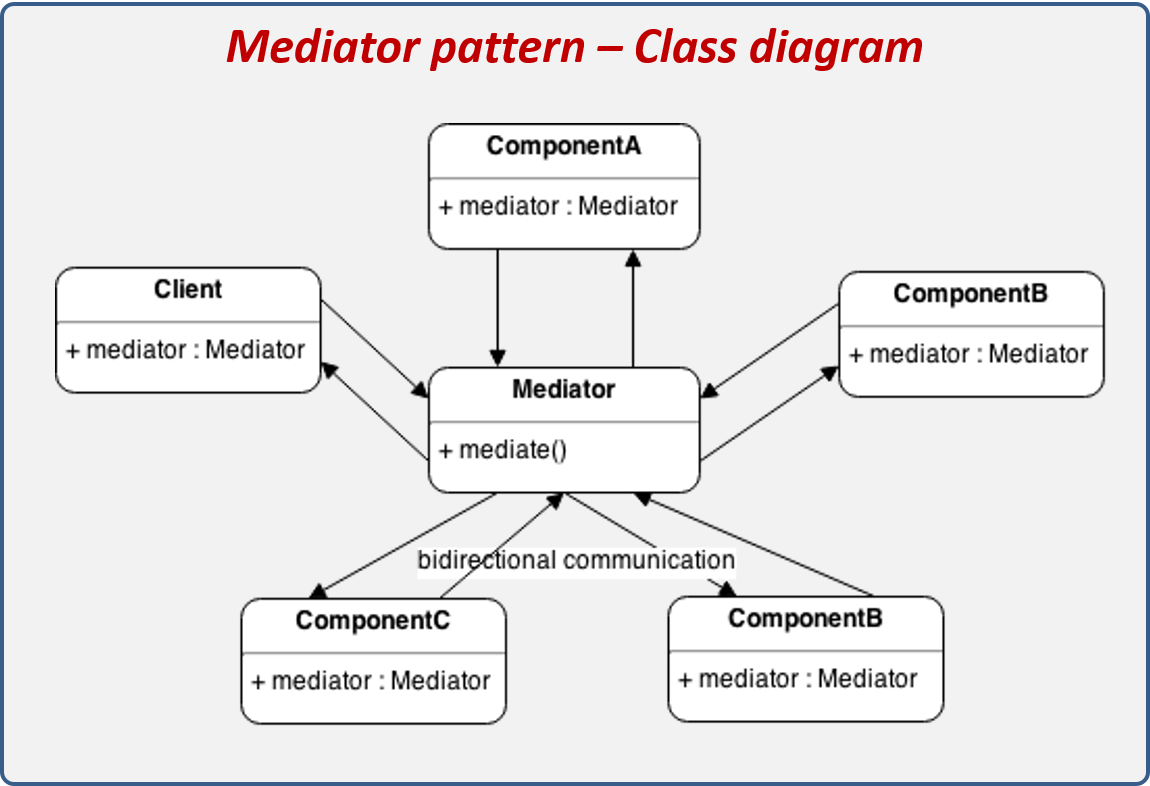
**StarLabs 2022 - Documentation**

**Mediator Pattern**

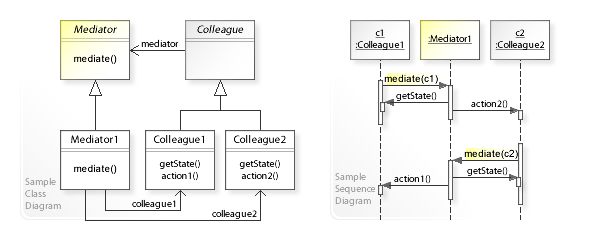
Mediator is a behavioral design pattern that lets you reduce chaotic dependencies between objects. The pattern restricts direct communications between the objects and forces them to collaborate only via a mediator object.



***Real world example:***

Pilots of aircraft that approach or depart the airport control area don’t communicate directly with each other. Instead, they speak to an air traffic controller, who sits in a tall tower somewhere near the airstrip. Without the air traffic controller, pilots would need to be aware of every plane in the vicinity of the airport, discussing landing priorities with a committee of dozens of other pilots. That would probably skyrocket the airplane crash statistics.

The tower doesn’t need to control the whole flight. It exists only to enforce constraints in the terminal area because the number of involved actors there might be overwhelming to a pilot.



***Applicability:***

1. Use the Mediator pattern when it’s hard to change some of the classes because they are tightly coupled to a bunch of other classes.
2. The pattern lets you extract all the relationships between classes into a separate class, isolating any changes to a specific component from the rest of the components.
3. Use the pattern when you can’t reuse a component in a different program because it’s too dependent on other components.
4. After you apply the Mediator, individual components become unaware of the other components. They could still communicate with each other, albeit indirectly, through a mediator object. To reuse a component in a different app, you need to provide it with a new mediator class.
5. Use the Mediator when you find yourself creating tons of component subclasses just to reuse some basic behavior in various contexts.
6. Since all relations between components are contained within the mediator, it’s easy to define entirely new ways for these components to collaborate by introducing new mediator classes, without having to change the components themselves.

***Advantages of Mediator Pattern:***

1. *Single Responsibility Principle*. You can extract the communications between various components into a single place, making it easier to comprehend and maintain.
2. *Open/Closed Principle*. You can introduce new mediators without having to change the actual components.
3. You can reduce coupling between various components of a program.
4. You can reuse individual components more easily.

***Disadvantages of Mediator Pattern:***

1. Over time a mediator can evolve into a God Object.