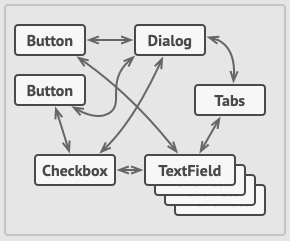
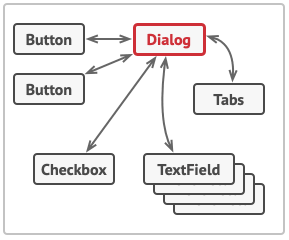
**Mediator**

The mediator design pattern is a behavioral pattern that encapsulates the interaction between a set of objects. The communication between these objects is restricted by forcing them to communicate only through a single object, the mediator. This way, the chaotic dependencies between objects are avoided.

**Problem**: Consider an application’s interface where the user is able to modify the profile’s information. On this interface, there are multiple buttons, checkboxes, text fields, and so on. These elements can interact with one another. For example, upon selecting a certain checkbox, a new text field appears. Another example would be a ‘Save’ button. Usually, when saving data, all the data fields must be validated. This type of connection between elements removes the possibility of reusability of the elements, as they are dependent of one another. (you can’t use the checkbox element independently of the text field). So, you either use all the elements, or none.



**Solution**: The connection between the elements that we want to be independent of one another should be removed. The communication between them should not be removed, though. In this case, they are connected with one another indirectly, through a **mediator**. Therefore, the dependencies are removed, each element depending only on a single mediator class.



**Example**: (‘Save’ button mentioned above – expand on the diagram above)

**Real-world analogy:** Control tower of an airport.