References

Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helm, Johnson, Vlissides, 1995, pp. 273-282

A mediator is responsible for controlling and coordinating the interactions of a group of objects.
Participants

Mediator

Defines an interface for communicating with Colleague objects

ConcreteMediator

Implements cooperative behavior by coordinating Colleague objects

Knows and maintains its colleagues

Colleague classes

Each Colleague class knows its Mediator object

Each colleague communicates with its mediator whenever it would have otherwise communicated with another colleague
Motivating Example - Dialog Boxes
How does this differ from a God Class?
When to use the Mediator Pattern

When a set of objects communicate in a well-defined but complex ways

When reusing an object is difficult because it refers to and communicates with many other objects

When a behavior that's distributed between several classes should be customizable without a lot of subclassing
How do Colleagues and Mediators Communicate?

Explicit methods in Mediator

class DialogDirector
{
    private Button ok;
    private Button cancel;
    private ListBox courses;

    public void listBoxItemSelected() { blah}

    public void listBoxScrolled() { blah }
    etc.
}
How do Colleagues and Mediators Communicate?

Generic change notification

class DialogDirector {
    private Button ok;
    private Button cancel;
    private ListBox courses;

    public void widgetChanged( Object changedWidget) {
        if ( changedWidget == ok ) blah
        else if ( changedWidget == cancel ) more blah
        else if ( changedWidget == courses ) even more blah
Differences from Facade

Facade does not add any functionality, Mediator does

Subsystem components are not aware of Facade

Mediator's colleagues are aware of Mediator and interact with it