CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2006

Doc 18 Mediator
April 24, 2007

References

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

The Design Patterns Smalltalk Companion, Alpert, Brown, Woolf, 1998, pp. 287-296

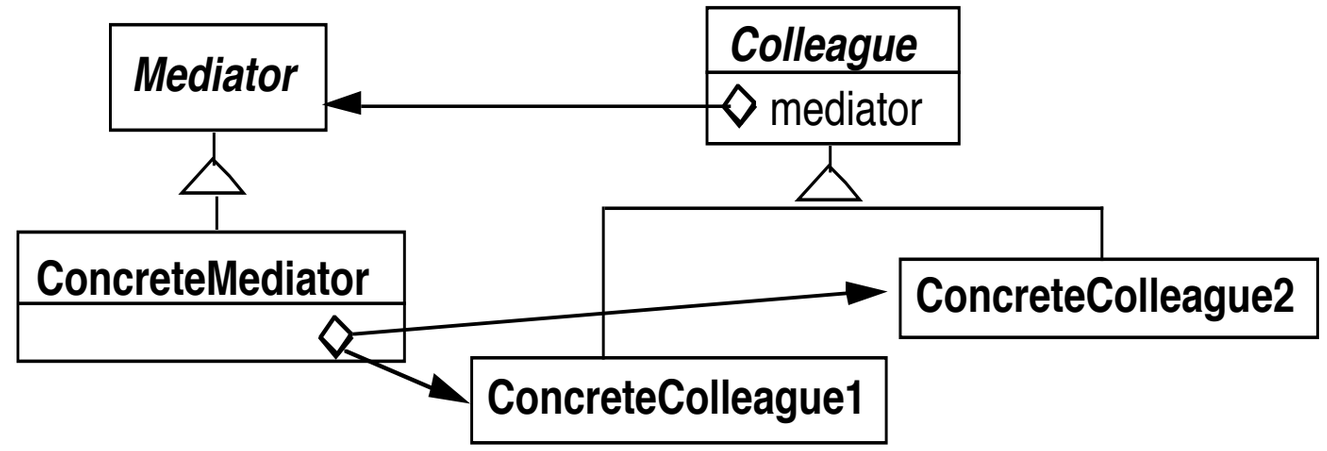
```
public class TrimBlanksTests extends TestCase {
   public void test() {

                assertEquals("a c", "  a c   ".trimBlanks()); //"--a-c---" - = blank
                assertEquals("", "".trimBlanks());
   }
}
```

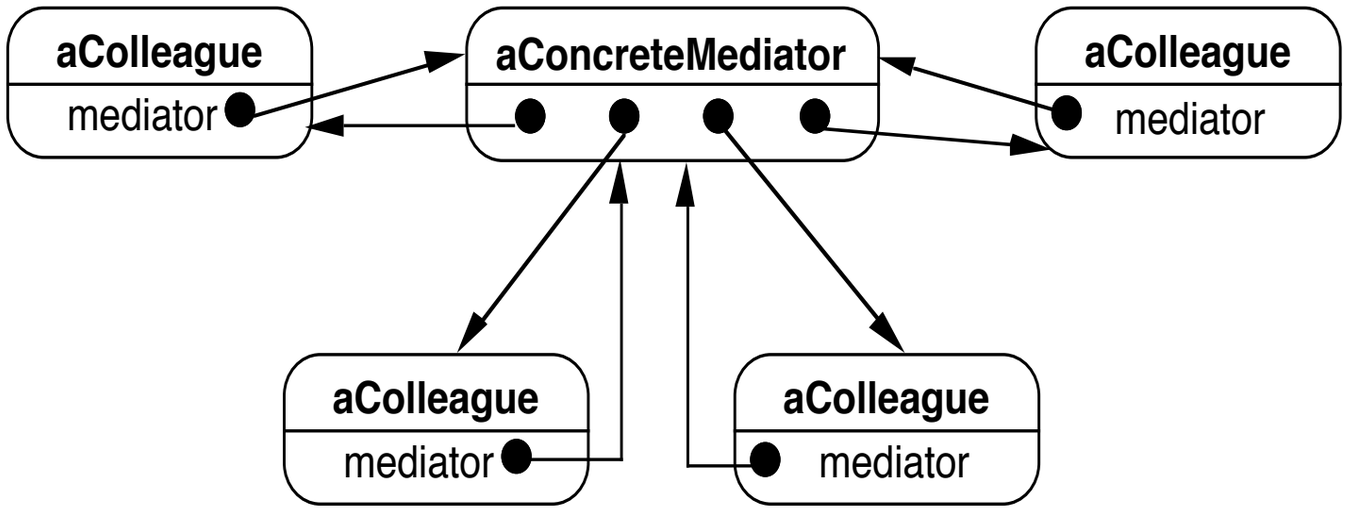
```
class A {
	public Key keyFrom(HttpRequest aRequest) {
		if (aRequest.identifier().size() < depth()){
			return keyForSmallSize(aRequest);
		}
		processAction(aRequest);
		return matchingElement(aRequest);
	}

	public processAction(HttpRequest aRequest) {
	}

	abstract Key keyForSmallSize(HttpRequest
aRequest);
}
```

```
class B extends A {
	public Key keyForSmallSize(HttpRequest aRequest) {
		if (aRequest.hasPostDataAt("Command"))
			return aRequest.postDataAt("Command")
		else
			return defaultKey();
	}

	public processAction(HttpRequest aRequest) {
		if (containsAction(matchingElement(aRequest)))
			aRequest.decodeFormData();
	}
}

class C extends A {
	public Key keyForSmallSize(HttpRequest aRequest) {
		return defaultKey();
	}
}
```

Mediator

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

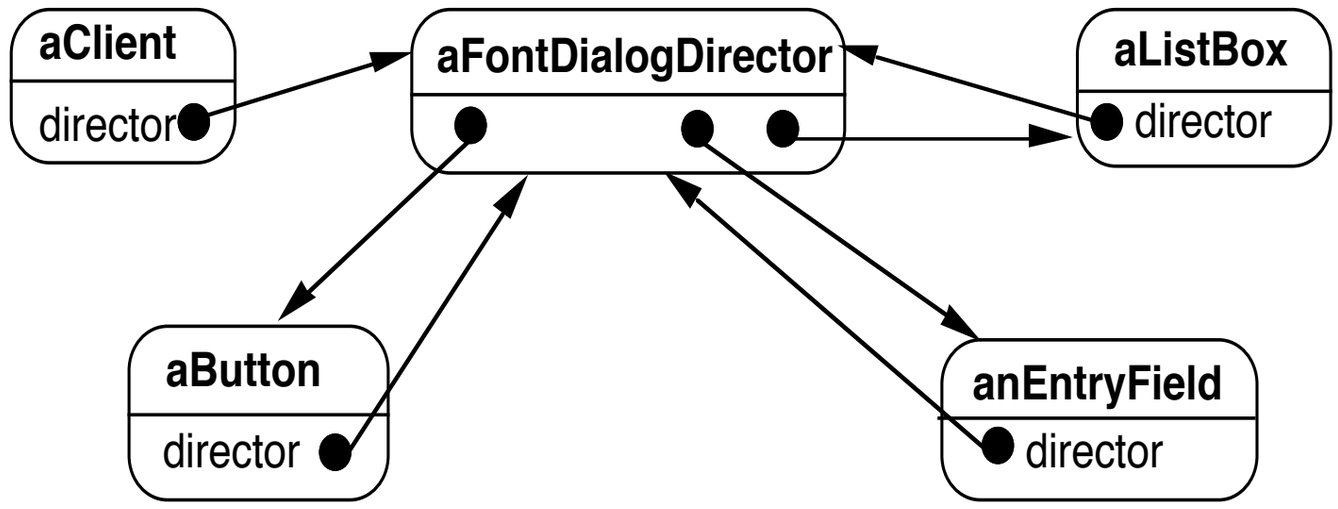
Knows and maintains its colleagues

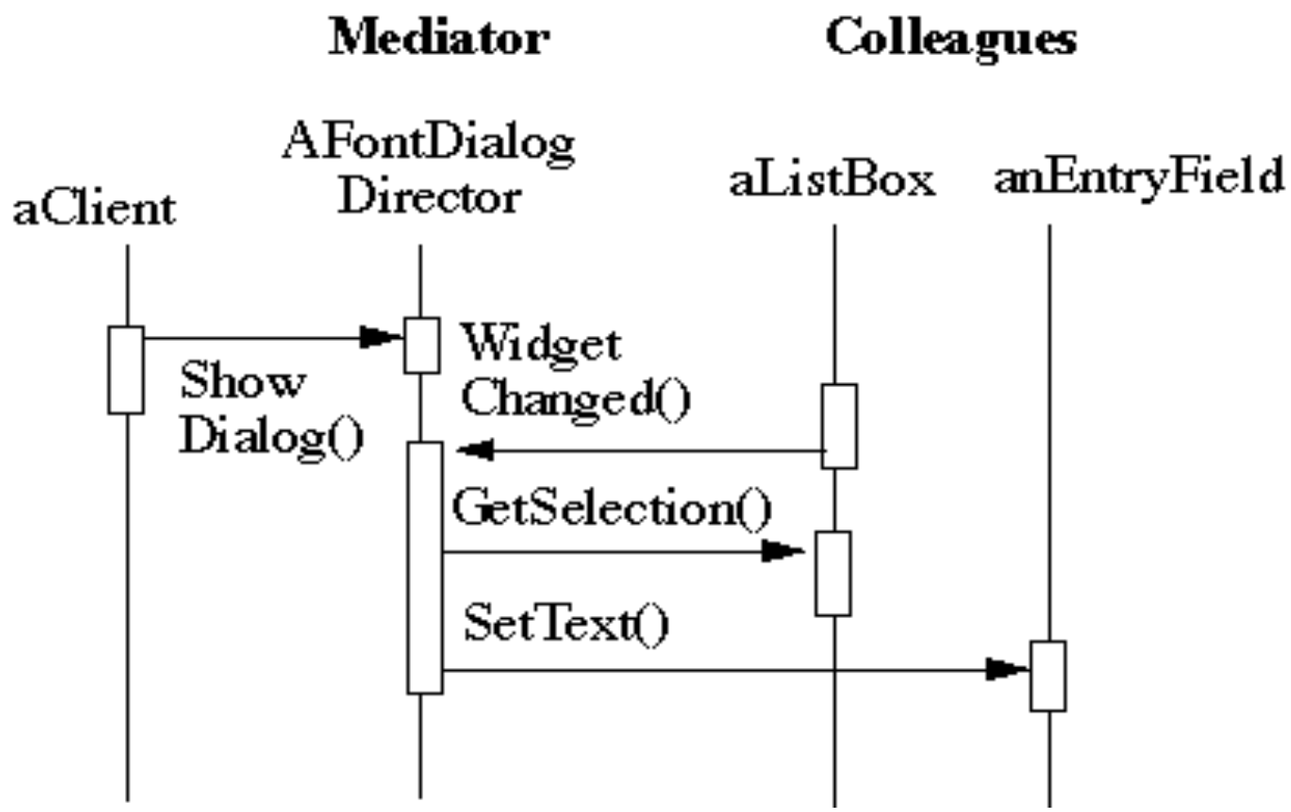
Colleague classes

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
```