CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2002
Doc 18 Mediator & Type Object

Contents

Mediator .................................................................................................................. 2
Structure .................................................................................................................. 2
Motivating Example ................................................................................................. 4
Issues ....................................................................................................................... 6
Type Object .............................................................................................................. 9

References

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

Type Object, Ralph Johnson & Bobby Woolf in Pattern Languages of Program Design 3, Edited by Martin, Riehle, Buschmann, 1998, pp. 47-65

Copyright ©, All rights reserved. 2002 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.
Mediator

A mediator is responsible for controlling and coordinating the interactions of a group of objects (not data structures)

Structure
Classes

Mediator

Colleague

ConcreteMediator

ConcreteColleague1

ConcreteColleague2

Objects

aColleague

mediator

aConcreteMediator

aColleague

mediator

aColleague

mediator

aColleague

mediator
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

Objects

aClient
  director

aFontDialogDirector

aListBox
  director

aButton
  director

anEntryField
  director

Interaction

Mediator

aClient

AFontDialog
Director

Show Dialog()

Widget Changed()

GetSelection()

SetText()

Colleagues

aListBox

anEntryField

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
Issues
How do Colleagues and Mediators Communicate?

1) Explicit methods in Mediator

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

    public void ListBoxItemSelected() { blah }

    public void ListBoxScrolled() { blah }

    etc.
}

2) Generic change method

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
    }
}
3) Generic change method overloaded

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

    public void widgetChanged( Button changedWidget)
    {
        if ( changedWidget == ok )
            blah
        else if ( changedWidget == cancel )
            more blah
    }

    public void widgetChanged( ListBox changedWidget)
    {
        now find out how it changed and
        respond properly
    }
}
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
Type Object

Intent

Decouples instances from their classes so those classes can be implemented as instances of a class

• Allows new classes to be created dynamically at runtime
• Lets a system provide its own type-checking rules

Also Known As

• Power Type
• Item Descriptor
• Metaobject
• Data Normalization
Motivation

Video Rental Store Inventory

Need to keep track of all the movies in the inventory

What
• About individual movies
• Multiple copies of a movie

Subclassing does not Work

What happens when new movies come out?

Instances of Videotape do not Work

Using one instance of Videotape class per movie
• Need to track multiple copies of a movie

Using one instance of Videotape for each copy of a movie
• Each copy contains a lot of duplicate information
Type Object Solution

Class Structure

```
Movie
| title() |
| rentPrice() |

VideoTape
| isRented() |
| renter() |
```

Object Structure

```
aMovie
| Monsoon Wedding |

aMovie
| SpiderMan |

aVideoTape
| John's Monsoon Wedding |

aVideoTape
| Pete's Monsoon Wedding |
```
Type Object Structure

<table>
<thead>
<tr>
<th>TypeClass</th>
<th>type attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>attribute</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TypeClass (Movie)**

- Is the class of TypeObject
- Has a separate instance for each type of Object

**TypeObject (Spiderman, Monsoon Wedding)**

- Is instance of TypeClass
- Represents a type of Object
- Implements some of the behavior for TypeClass

```
aTypeClass
TypeObject1

aTypeClass
TypeObject2

aClass
ObjectA
  type

aClass
ObjectB
  type
```