

Behavioral Patterns

- Example: Document Retrieval System
- Observer
- Mediator
- Template
- Strategy
- Visitor (covered previously)

T9: Behavioral Pattern

2005 Fall, CSC407 (by W.Liu)

1

Recall Example: Document Retrieval System

- Overview:
 - A system that enables the users to add and remove documents to and from the system, and retrieve documents based on simple keywords queries. This example is taken from 2002 fall term assignment 2.
- Key requirements:
 1. Document types include plain text, email, and HTML
 2. Queries and their results are stored
 3. Query results must be consistent with the current document collections in the system
 4. New queries can be added, and existing queries can be deleted or modified
 5. Allows dynamic use of different raw text keyword extraction algorithms
 6. Allows the use of off-the-shelf HTML keyword extraction package

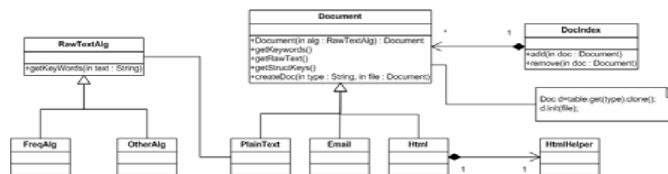
More details of this system are available at:
<http://www.cdf.utoronto.ca/~nelg/CSC407F-2002/A2>

T9: Behavioral Pattern

2005 Fall, CSC407 (by W.Liu)

2

Document Retrieval System: Partial Solution Part I

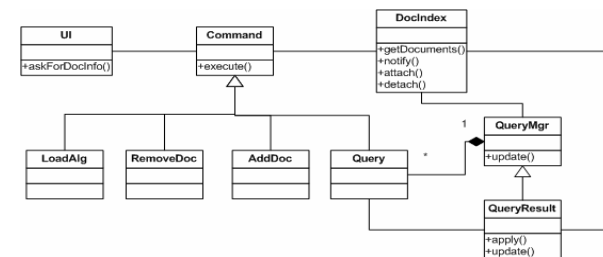


T9: Behavioral Pattern

2005 Fall, CSC407 (by W.Liu)

3

Partial Solution Part II

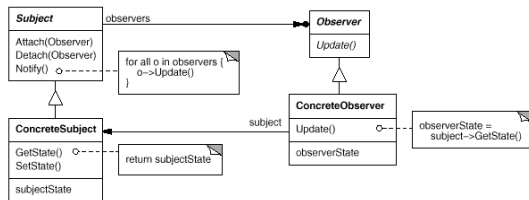


T9: Behavioral Pattern

2005 Fall, CSC407 (by W.Liu)

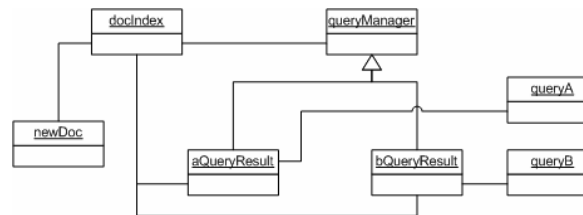
4

Review: Observer Pattern



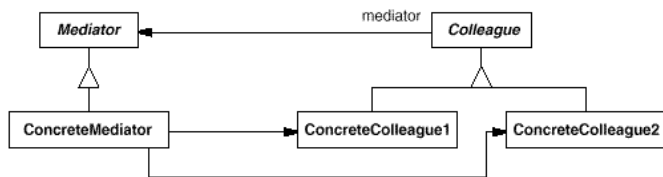
- Also known as Publisher-Subscriber
 - subject: publisher
 - observer: subscriber
- Subject.Notify() sends messages to all of its subscribers for change notice
- The subscribers (ConcreteObserver) then pickup the change of state in the publisher (ConcreteSubject) by direct contact
- Note that the subscriber only queries the state change, but does not modify the state of the publisher

Observer Pattern Used in the Example



- Participants:
 - Subject/ConcreteSubject : DocIndex
 - Observer: QueryManager
 - ConcreteObserver: QueryResult
- Applicability and purpose:
 - Query results can instantly reflect changes made to the document collection
 - Maintains the consistency of query results, thus satisfy Key Req #3
- How do we make use of the Mediator in Solution Part II?

Review: Mediator Pattern

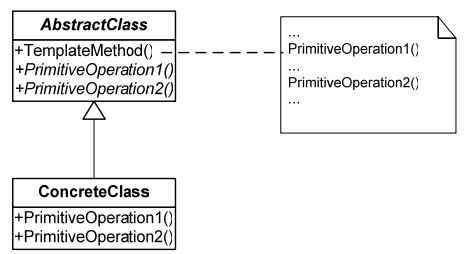


- This pattern is commonly used in enforcing a constraint that involves multiple participants (concrete colleagues)
- Scenario: when a change occurs to a participating concrete colleague, it notifies its mediator, and the mediator will then carry out predefined actions to update all related colleagues
- Benefits
 - the concrete colleagues don't have to know who their peers are or even what the constraint is about
 - multiple constraints can be defined independently from the concrete colleague classes

Observer vs. Mediator

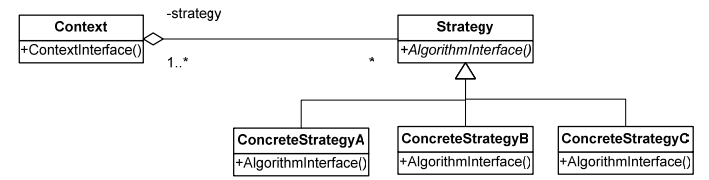
- Both patterns apply to situations where actions are required in response to a change
- Observer:
 - there might be multiple observers
 - each observer knows how to **query** the state change in its subject
- Mediator:
 - there is usually one mediator per pattern (or constraint)
 - all the response actions are stored in the mediator
 - the mediator may modify states of the concrete colleagues
 - this pattern may implicitly use the observer pattern

Review: Template Method Pattern



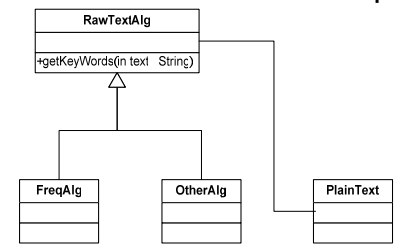
- Uses inheritance to vary part of an algorithm
- The superclass operation (i.e. Template Method) defines a sequence of steps required to perform some function
- The subclass fills in only the specific behaviors through the implementing methods (i.e. Primitive Operations)
- Implicitly using Factory Method (the Primitive Operations)
- Example: the standard interface for *HttpServletRequest* has
 - `service()`, `doGet()`, `doPost()`

Review: Strategy Pattern



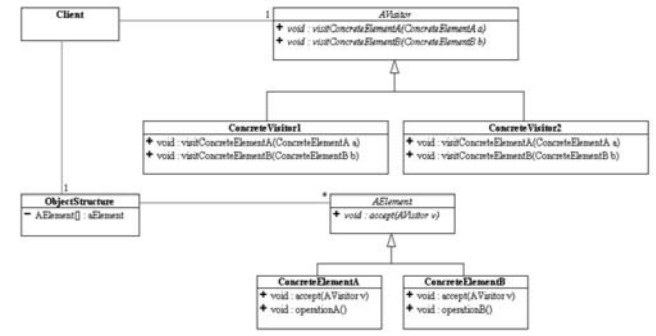
- Uses delegation to vary the entire algorithm
- Different algorithms may be appropriate at different times
- Client gets referred to the algorithm through its *Context*
- To simplify large body of conditional branches
- Notice that the entire algorithm is changed when a new concrete strategy is used in the Strategy pattern
- But in the **Template Method**, only individual primitive operations of the algorithm change when a new concrete class is used, the overall sequence of steps (defined in the template method) do not change

Strategy Pattern Used in the Example



- Participants:
 - Context : PlainText
 - Strategy : RawTextAlg
 - ConcreteStrategy: FreqAlg, OtherAlg
- Applicability and purpose:
 - Allows the user to dynamically use different keyword extraction algorithms
 - Thus satisfies Key Req #5

Review: Visitor Pattern



Visitor (con't)

- The Visitor substructure represents the hierarchy of the actions/operations
- The Element substructure represents the hierarchy of the objects
- Creating n by m methods instead of creating n by m class hierarchy since each object needs its own action and each action needs to be defined for every object
 - helps to avoid creating overwhelmingly many of classes
- Particularly useful when the object structure is of Composite type that involves various traversal mechanism
- Realizes double polymorphic dispatch (multiple polymorphism)

T9: Behavioral Pattern

2005 Fall, CSC407 (by W.Liu)

13

Using Visitor in the Running Example

- The original problem statement does not require the use of this pattern
- Suppose we have a new requirement that asks us to implement undo and redo for each command (loadAlg, addDoc, removeDoc, query)
- If we hard code the undo and redo mechanism within each command type, we will have more difficulty to modify the undo / redo algorithm in the future
- Therefore, we want to implement undo / redo separately from our command objects
- Using Visitor pattern, we can accomplish these goals
- Describe how you would do it using slide 6

T9: Behavioral Pattern

2005 Fall, CSC407 (by W.Liu)

14

Review and Exercises

We have seen many of the following patterns in the running example

- Behavioral
 - Observer
 - Visitor
 - Mediator (with modifications)
 - Strategy
 - Template method (Document and its subclasses)
- Structural
 - Adapter
- Creational
 - Singleton
 - Prototype
 - Factory method

How would you apply the following?

- Command
- Iterator

T9: Behavioral Pattern

2005 Fall, CSC407 (by W.Liu)

15