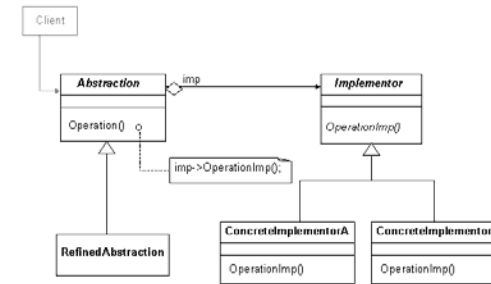


Structural Patterns (2)

- Recall that these patterns
 - deal with how classes are composed to form larger structures
 - let you vary some aspects of system structure independently of other aspects
 - can be used to make a system more robust to particular structural change
- We cover Bridge, Decorator, Flyweight
- Last time: Adapter, Proxy, Composite + Visitor (behavioral)

1

Bridge (1)



Intent: Decouples an abstraction from its implementation so that the two can vary independently.

2

Bridge (2)

- Lets you vary implementation of an object independently of its abstraction
- Applicable when
 - you want to avoid a permanent binding between an abstraction and its implementation
 - both the abstractions and their implementations should be extensible by subclassing
 - changes in the implementation of an abstraction should have no impact on clients
 - you want to hide the implementation of an abstraction completely from clients (C++)
 - you have a proliferation of classes, whose hierarchy indicates the need for splitting an object into two parts
 - you want to share an implementation among multiple objects, and this fact should be hidden from the client
- A well known example is the EJB technology and its various vendor specific implementations

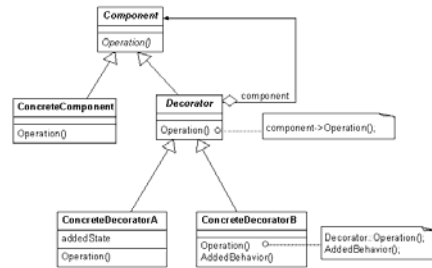
3

Bridge vs. Abstract Factory

- | | |
|--|---|
| <ul style="list-style-type: none"> Application concept vs. platform dependent implementation Structural and static | <ul style="list-style-type: none"> Factory vs. product Creational and non-static Can create and configure a bridge |
| <ul style="list-style-type: none"> Both have two class hierarchies Other similarities and differences? | |

4

Decorator (1)



Intent: Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.

5

Decorator (2)

- Lets you vary responsibilities of an object by allowing an open-ended number of additional responsibilities.
- Applicable when
 - adding responsibilities to individual objects dynamically and transparently, i.e., without affecting other objects
 - for responsibilities that can be withdrawn
 - extensions by subclassing is impractical

6

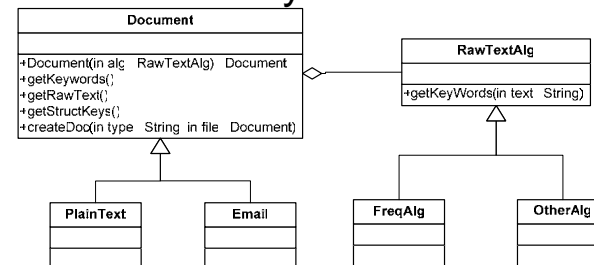
Recall the Example: Document Retrieval System

- 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/~nelig/CSC407F-2002/A2>

7

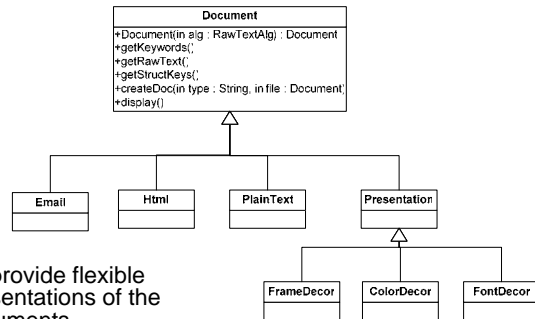
Bridge: Document Retrieval System



- Abstraction class
- Implementor class

8

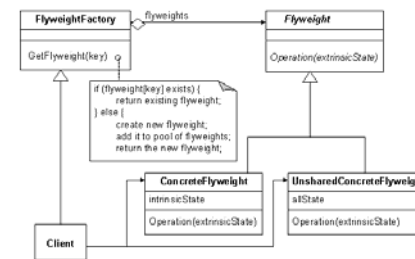
Decorator: Document Retrieval System



To provide flexible presentations of the documents.
Which key association is missing?

9

Flyweight (1)



Intent: Use sharing to support large number of fine-grained objects efficiently.

10

Flyweight (2)

- Lets you vary storage of a large number of objects efficiently
- Applicable when all of the following are true
 - An application uses a large number of objects
 - Storage costs are high because of the sheer quantity of objects
 - Most object state can be made extrinsic
 - Many groups of objects may be replaced by relatively few shared objects once extrinsic state is removed
 - The application does not depend on object identity

11