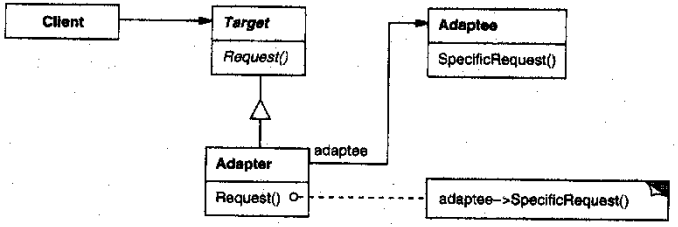
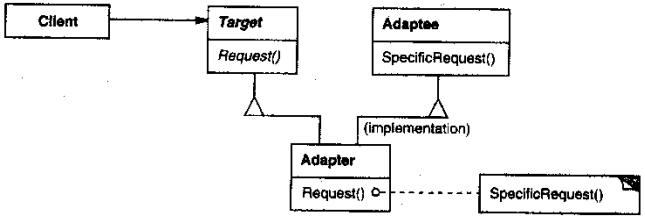
Design Patterns Information Sheet (no longer in strict alphabetical order)

Adapter. Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.

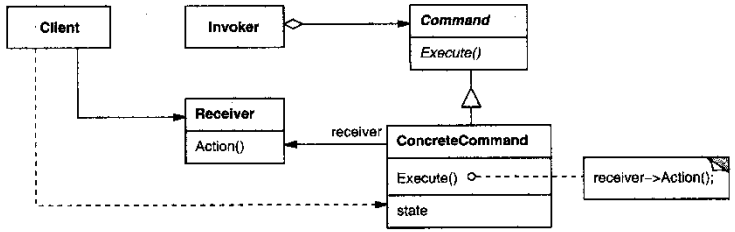
Object Adapter

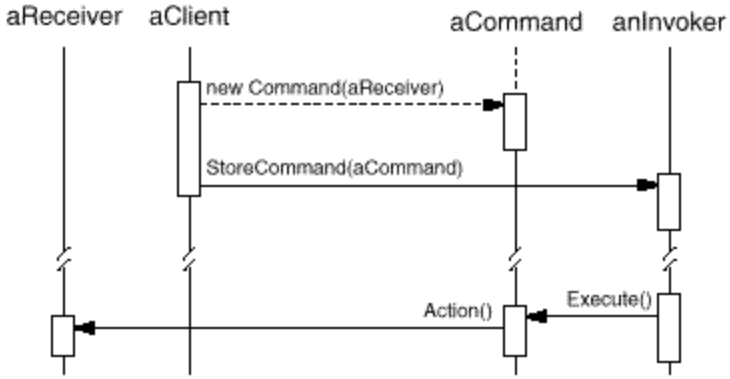


Class Adapter



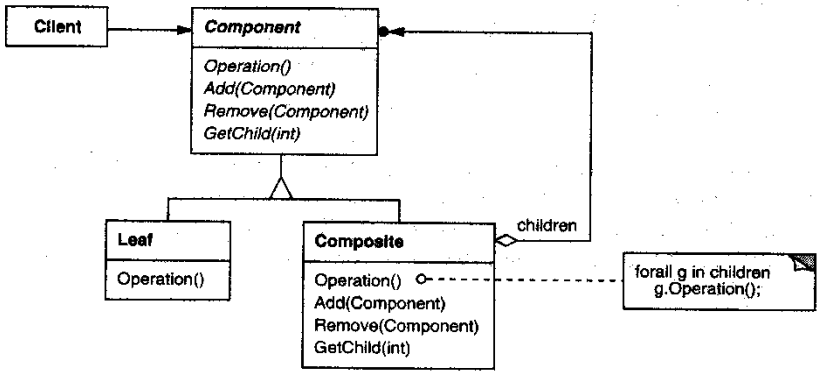
Command. Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.

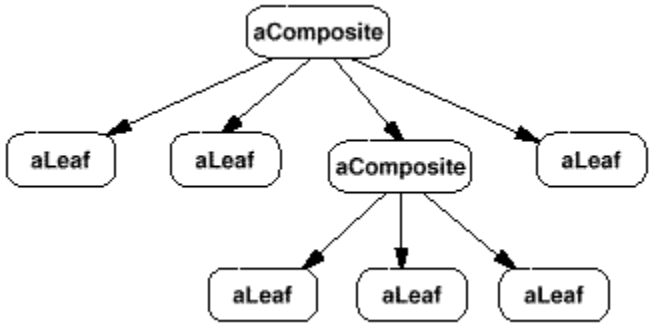




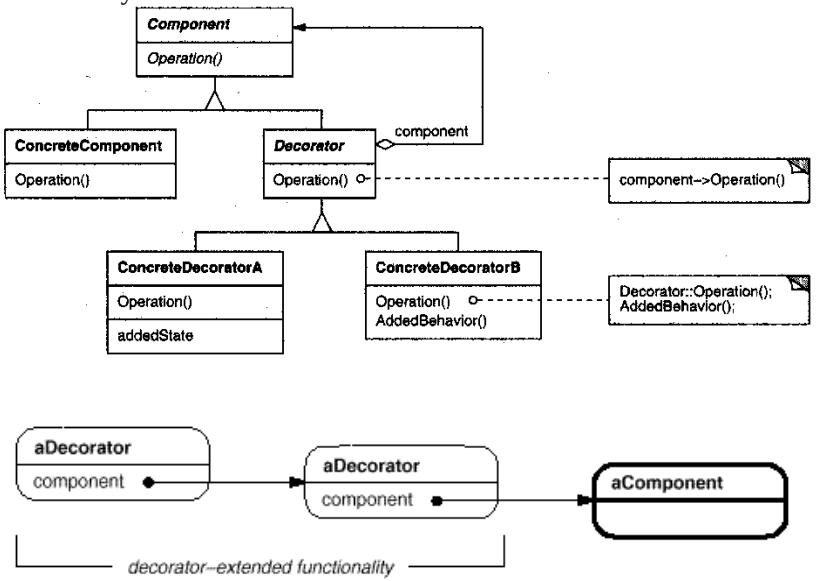
Design Patterns Information Sheet (no longer in strict alphabetical order)

Composite. Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.

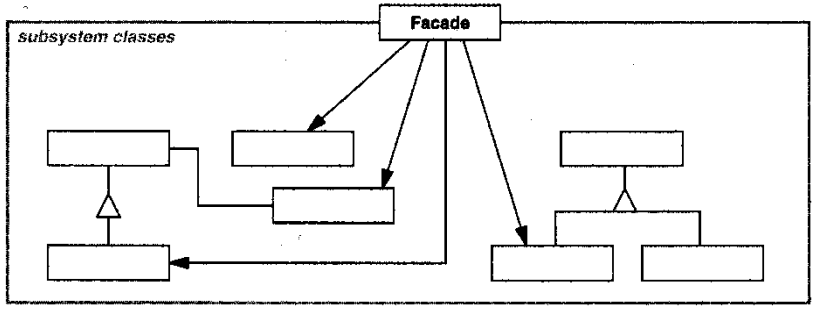




Decorator. Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.

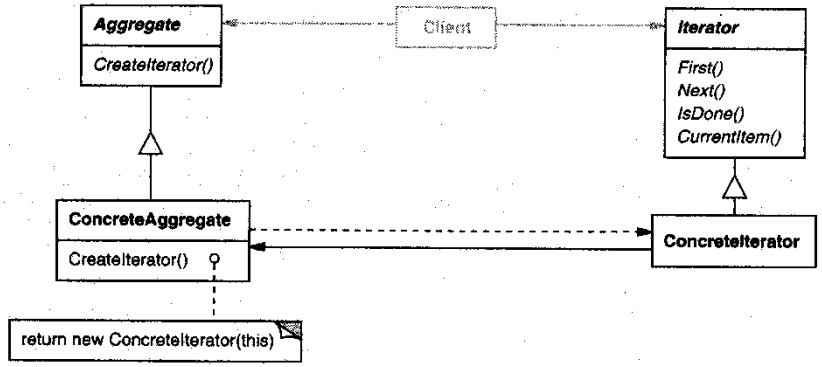


Facade. Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.

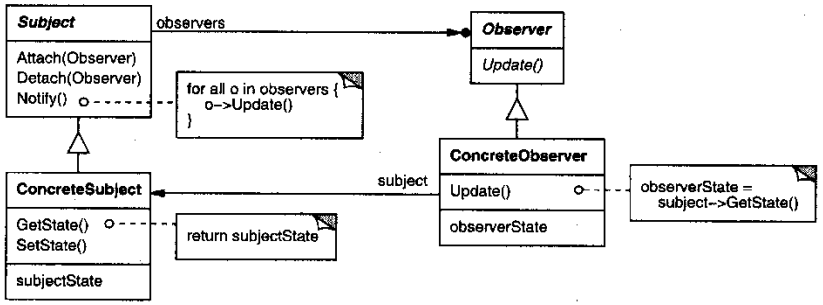


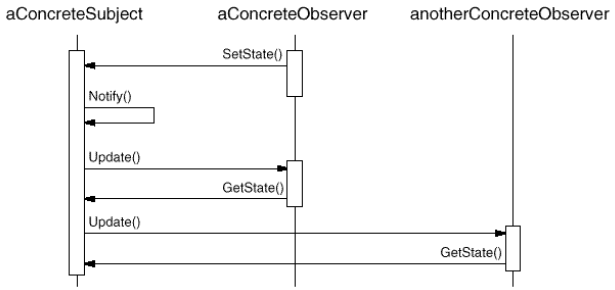
Design Patterns Information Sheet (no longer in strict alphabetical order)

Iterator. Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.

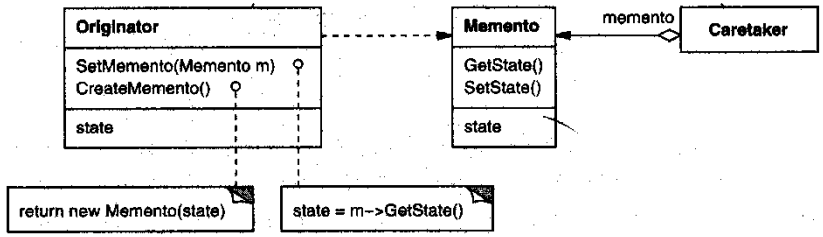


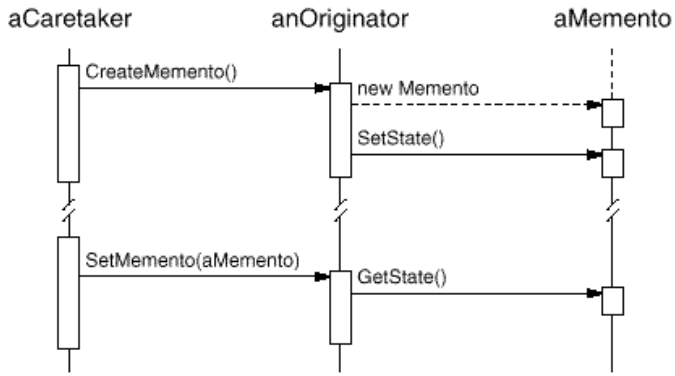
Observer. Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.





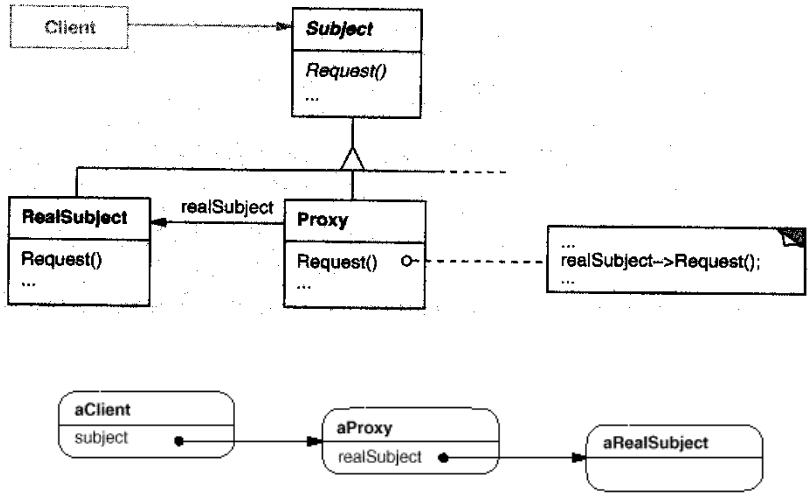
Memento. Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later.



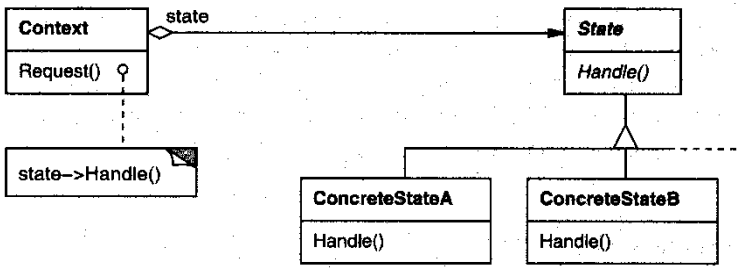


Design Patterns Information Sheet (no longer in strict alphabetical order)

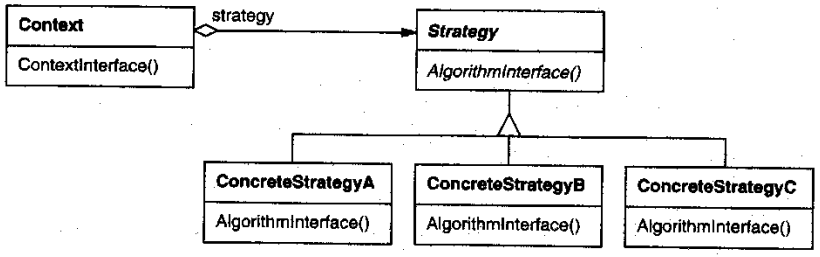
Proxy. Provide a surrogate or placeholder for another object to control access to it.



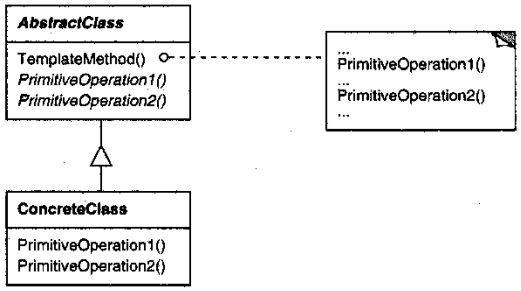
State. Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.



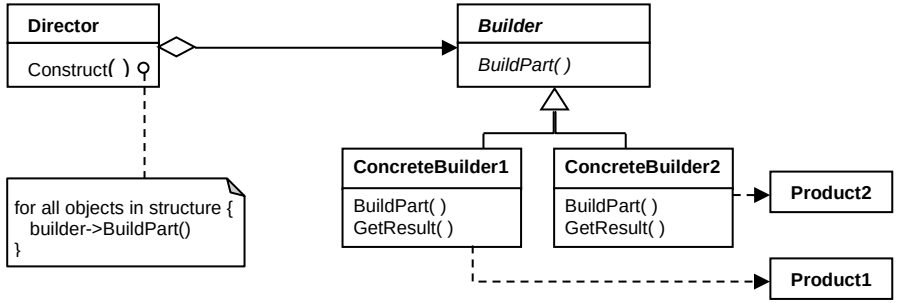
Strategy. Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.



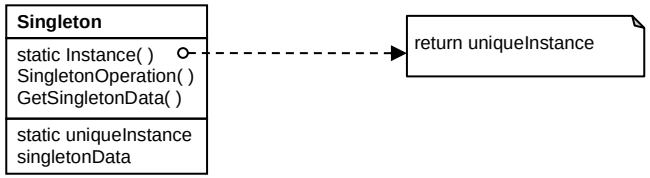
Template Method. Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.



Builder. Separate the construction of a complex object from its representation so that the same construction process can create different representations.

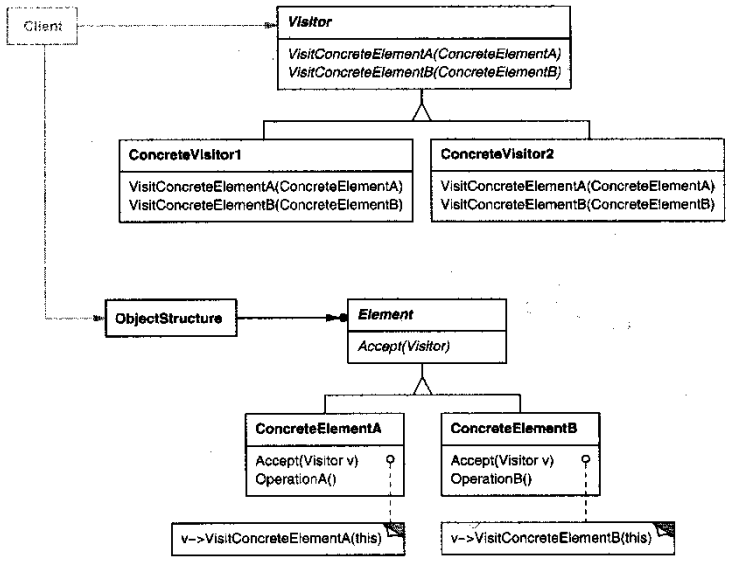


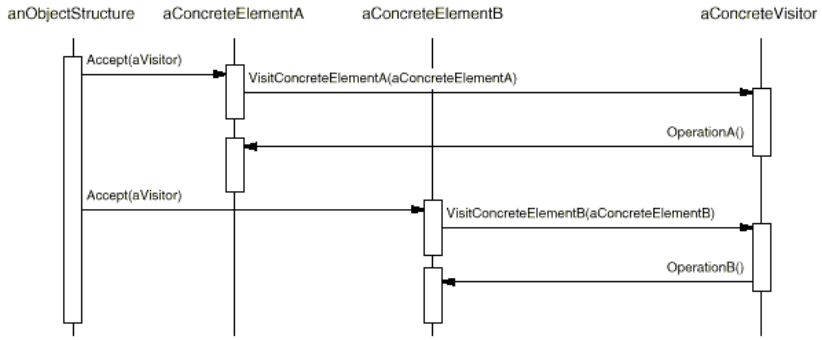
Singleton. Ensure a class only has one instance, and provide a global point of access to it.



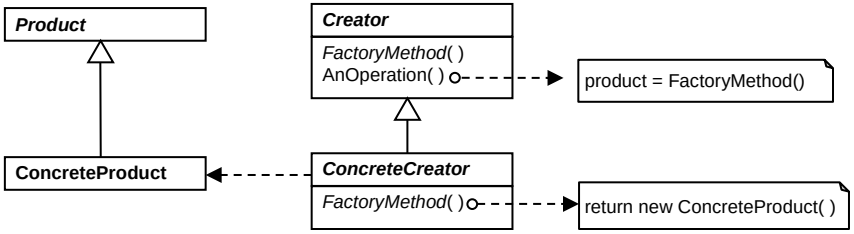
Design Patterns Information Sheet (no longer in strict alphabetical order)

Visitor. Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.

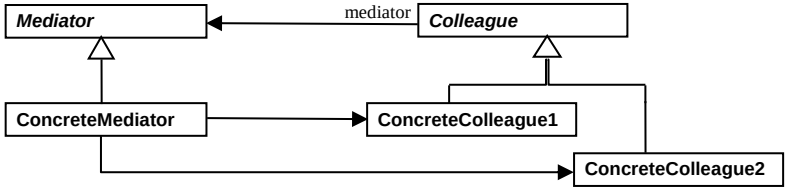


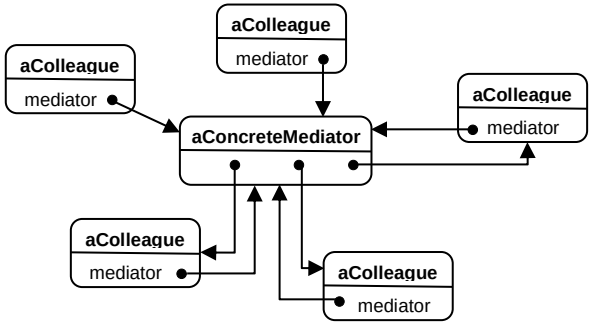


Factory Method. Define an interface for creating an object, but let subclasses decide which class to instantiate.



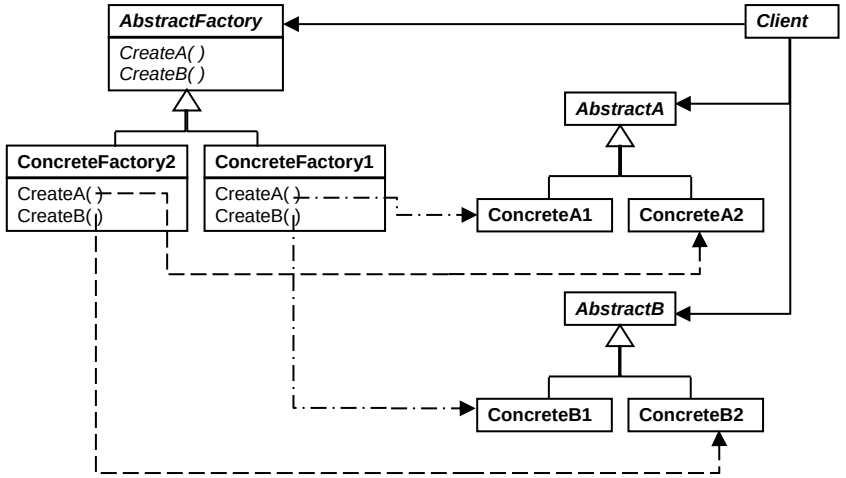
Mediator. Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.





Design Patterns Information Sheet (no longer in strict alphabetical order)

Abstract Factory. Provide an interface for creating families of related or dependent objects without specifying their concrete classes.



Chain of Responsibility. Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request.