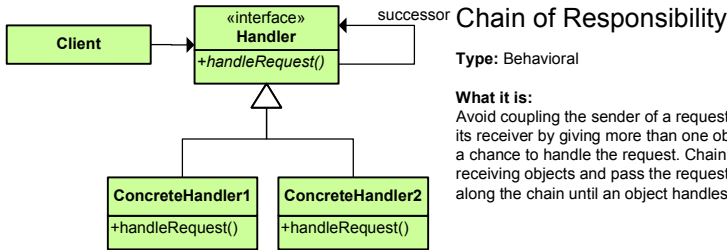


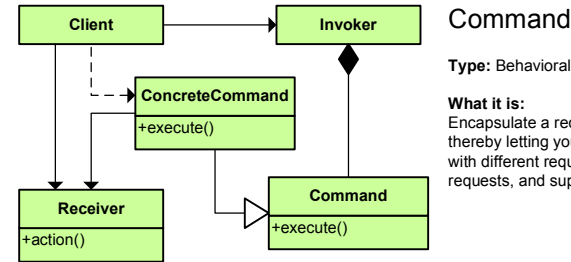
- | | | |
|---------------------------|------------------|-------------------|
| C Abstract Factory | S Facade | S Proxy |
| S Adapter | C Factory Method | B Observer |
| S Bridge | S Flyweight | C Singleton |
| C Builder | B Interpreter | B State |
| B Chain of Responsibility | B Iterator | B Strategy |
| B Command | B Mediator | B Template Method |
| S Composite | B Memento | B Visitor |
| S Decorator | C Prototype | |



Chain of Responsibility

Type: Behavioral

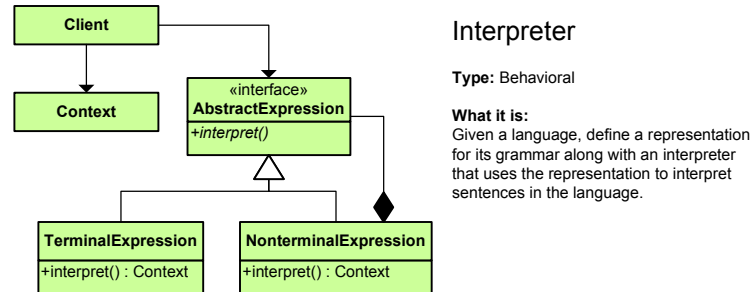
What it is: Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.



Command

Type: Behavioral

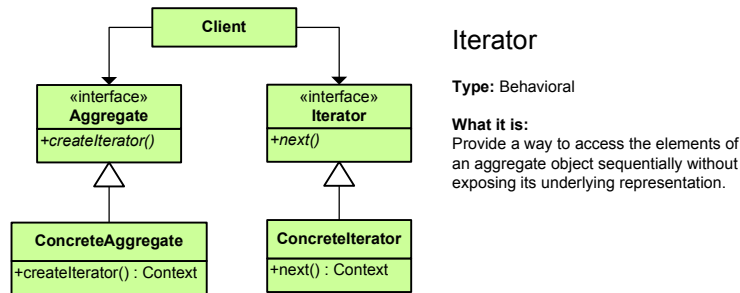
What it is: Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.



Interpreter

Type: Behavioral

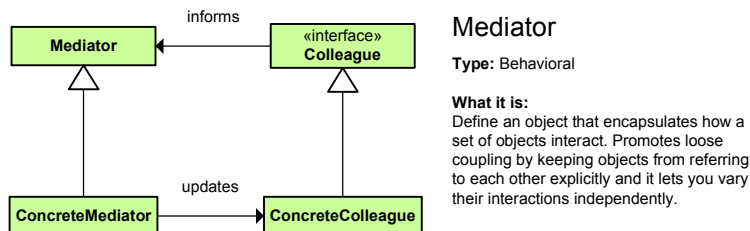
What it is: Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.



Iterator

Type: Behavioral

What it is: Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.



Mediator

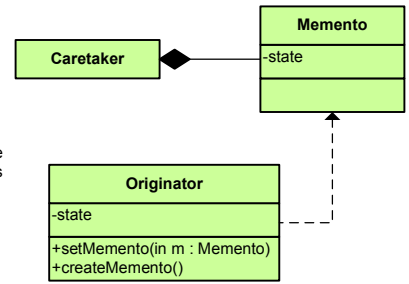
Type: Behavioral

What it is: Define an object that encapsulates how a set of objects interact. Promotes loose coupling by keeping objects from referring to each other explicitly and it lets you vary their interactions independently.

Memento

Type: Behavioral

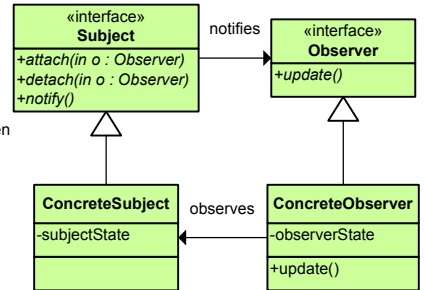
What it is: Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later.



Observer

Type: Behavioral

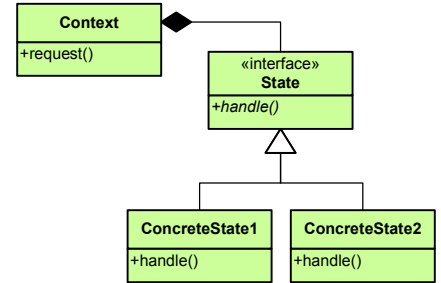
What it is: Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.



State

Type: Behavioral

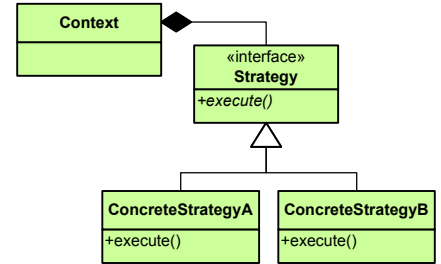
What it is: Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.



Strategy

Type: Behavioral

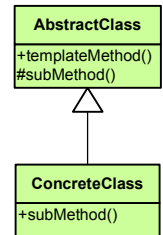
What it is: Define a family of algorithms, encapsulate each one, and make them interchangeable. Lets the algorithm vary independently from clients that use it.



Template Method

Type: Behavioral

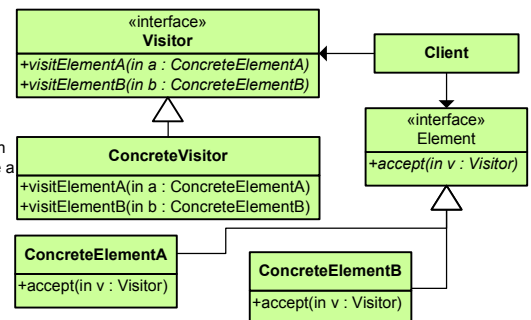
What it is: Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

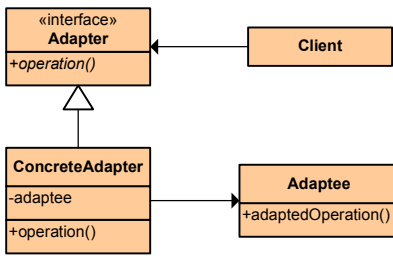


Visitor

Type: Behavioral

What it is: Represent an operation to be performed on the elements of an object structure. Lets you define a new operation without changing the classes of the elements on which it operates.

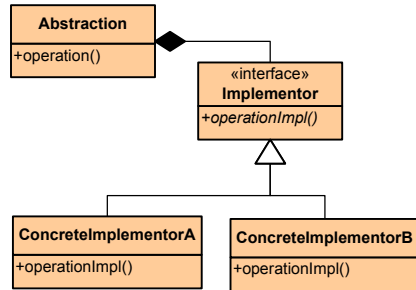




Adapter

Type: Structural

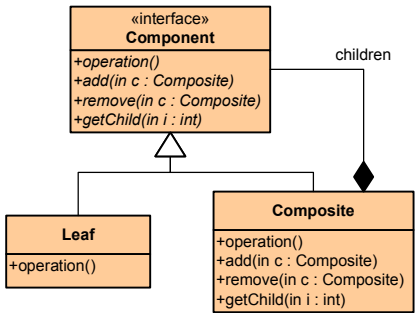
What it is: Convert the interface of a class into another interface clients expect. Lets classes work together that couldn't otherwise because of incompatible interfaces.



Bridge

Type: Structural

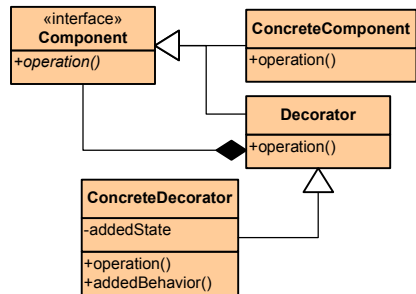
What it is: Decouple an abstraction from its implementation so that the two can vary independently.



Composite

Type: Structural

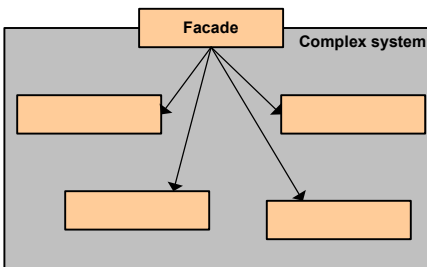
What it is: Compose objects into tree structures to represent part-whole hierarchies. Lets clients treat individual objects and compositions of objects uniformly.



Decorator

Type: Structural

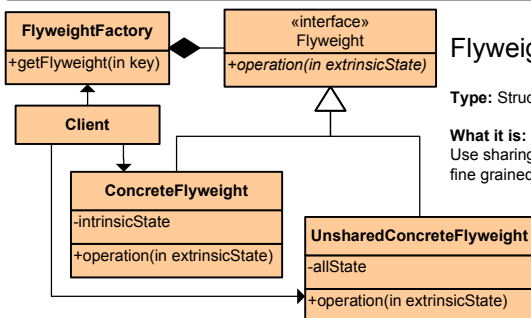
What it is: Attach additional responsibilities to an object dynamically. Provide a flexible alternative to sub-classing for extending functionality.



Facade

Type: Structural

What it is: Provide a unified interface to a set of interfaces in a subsystem. Defines a high-level interface that makes the subsystem easier to use.



Flyweight

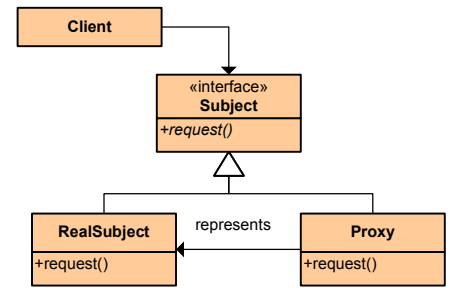
Type: Structural

What it is: Use sharing to support large numbers of fine grained objects efficiently.

Proxy

Type: Structural

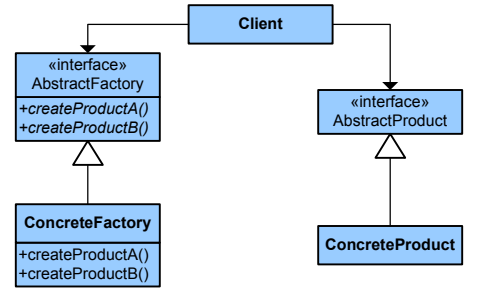
What it is: Provide a surrogate or placeholder for another object to control access to it.



Abstract Factory

Type: Creational

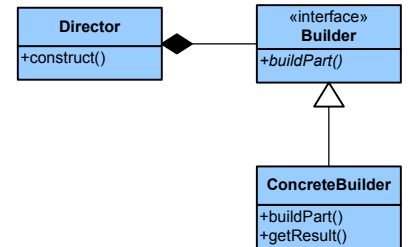
What it is: Provides an interface for creating families of related or dependent objects without specifying their concrete class.



Builder

Type: Creational

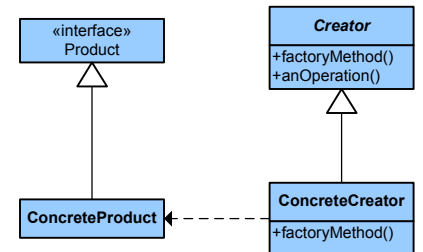
What it is: Separate the construction of a complex object from its representing so that the same construction process can create different representations.



Factory Method

Type: Creational

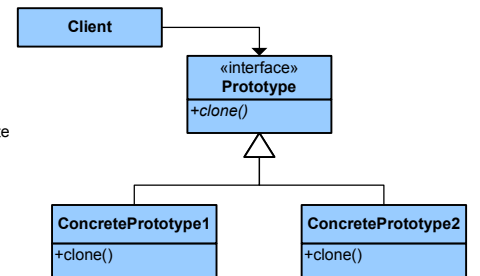
What it is: Define an interface for creating an object, but let subclasses decide which class to instantiate. Lets a class defer instantiation to subclasses.



Prototype

Type: Creational

What it is: Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.



Singleton

Type: Creational

What it is: Ensure a class only has one instance and provide a global point of access to it.

