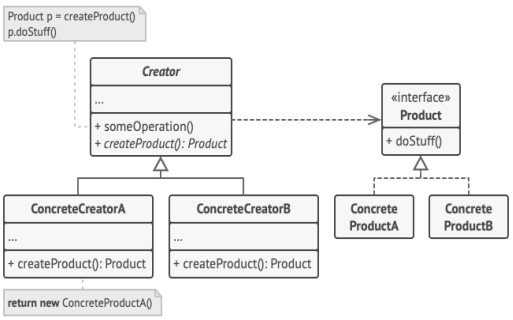
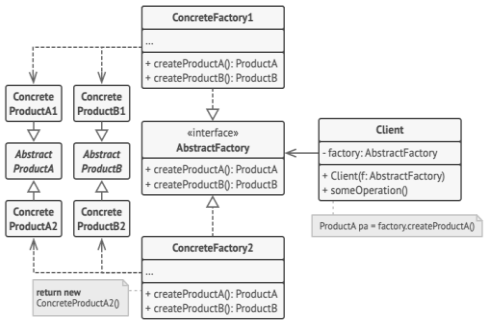


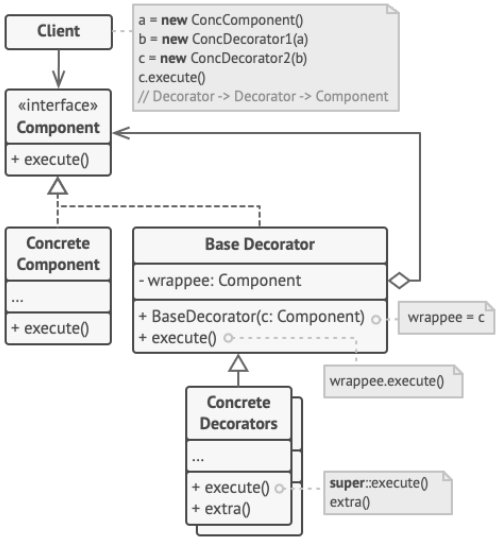
Factory Mehod

Type: Creational
Definition: provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects that will be created.



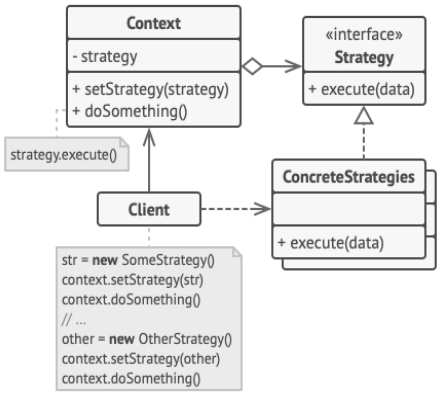
Abstract Factory

Type: Creational
Definition: produce families of related objects without specifying their concrete classes.



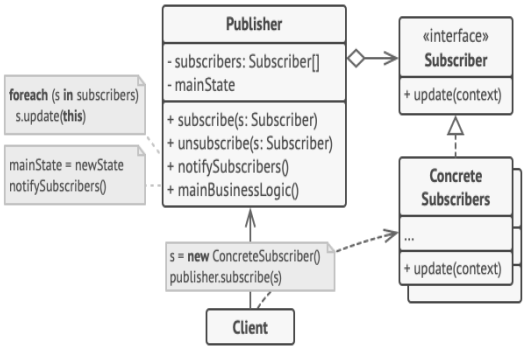
Decorator

Type: Structural
Definition: attach new behaviors to objects by placing these objects inside special wrapper objects that contain the behaviors.



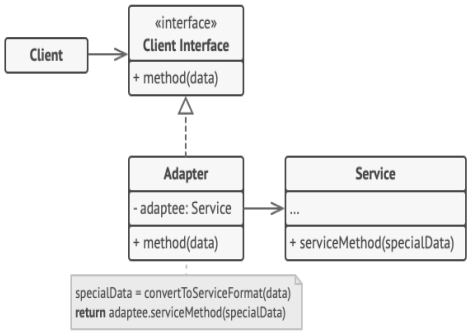
Strategy

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



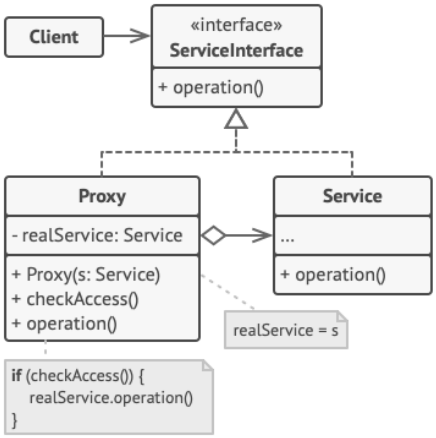
Observer

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



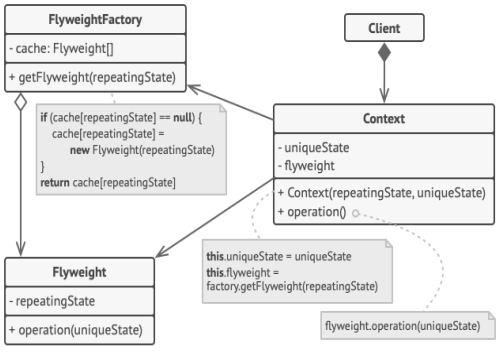
Adapter

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



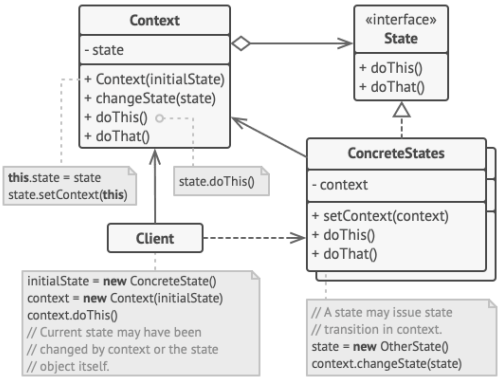
Proxy

Type: Structural
Definition: lets you provide a substitute or placeholder for another object. A proxy controls access to the original object, allowing you to perform something either before or after the request gets through to the original object.



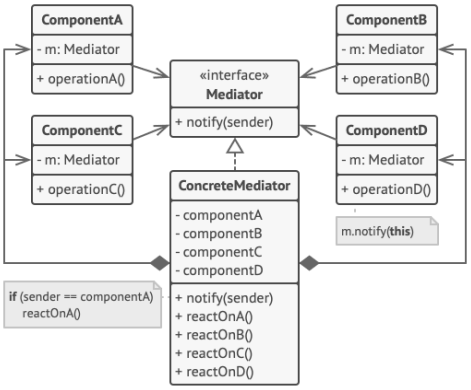
Flyweight

Type: Structural
Definition: lets you fit more objects into the available amount of RAM by sharing common parts of state between multiple objects instead of keeping all of the data in each object.



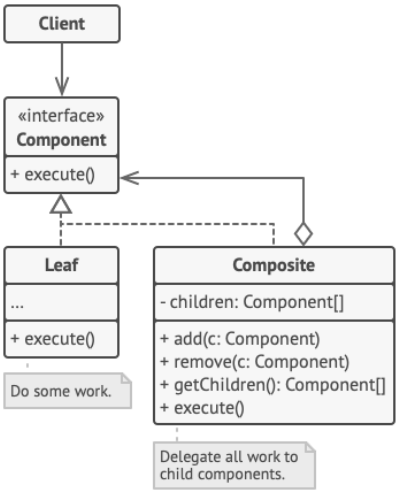
State

Type: Behavioral
Definition: lets an object alter its behavior when its internal state changes. It appears as if the object changed its class.



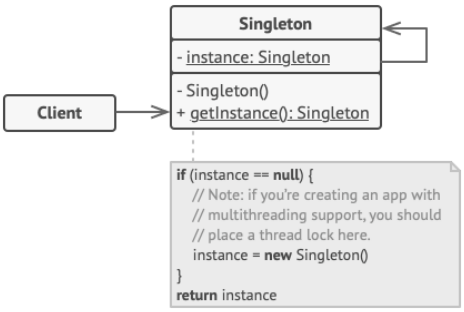
Mediator

Type: Behavioral
Definition: lets you reduce chaotic dependencies between objects. The pattern restricts direct communications between the objects and forces them to collaborate only via a mediator object.



Composite

Type: Structural
Definition: lets you compose objects into tree structures and then work with these structures as if they were individual objects.



Singleton

Type: Creational
Definition: lets you ensure that a class has only one instance, while providing a global access point to this instance.