

The background is a dark blue gradient with a starry texture. On the left side, there are several overlapping circular elements. A prominent one is a large circle with a scale around its perimeter, marked with numbers from 140 to 260 in increments of 10. Other circles are partially visible, some with dashed lines and arrows, suggesting a technical or scientific theme.

# ПАТТЕРНЫ В ПРОГРАММИРОВАНИИ

# Singleton

**Type:** Creational

**What it is:**

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

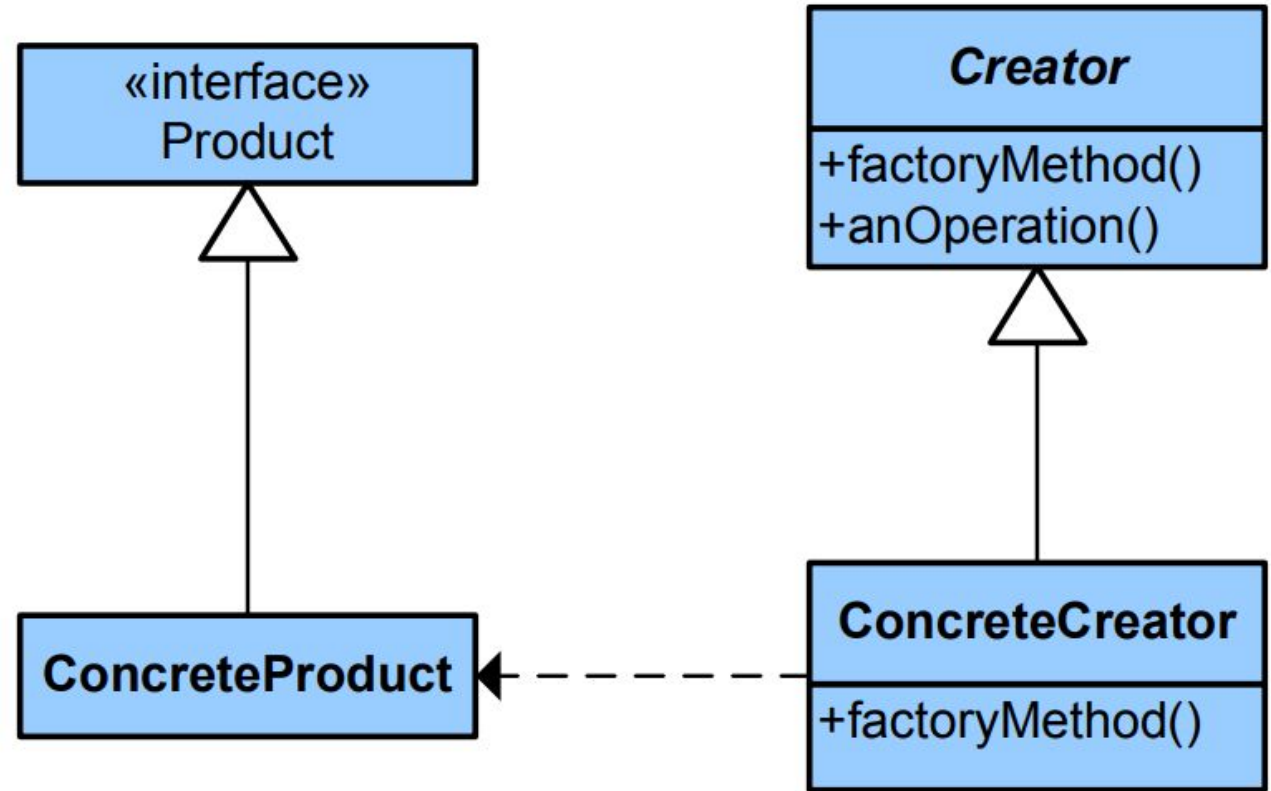
| Singleton                                   |
|---|
| -static uniqueInstance<br>-singletonData    |
| +static instance()<br>+SingletonOperation() |

# 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.

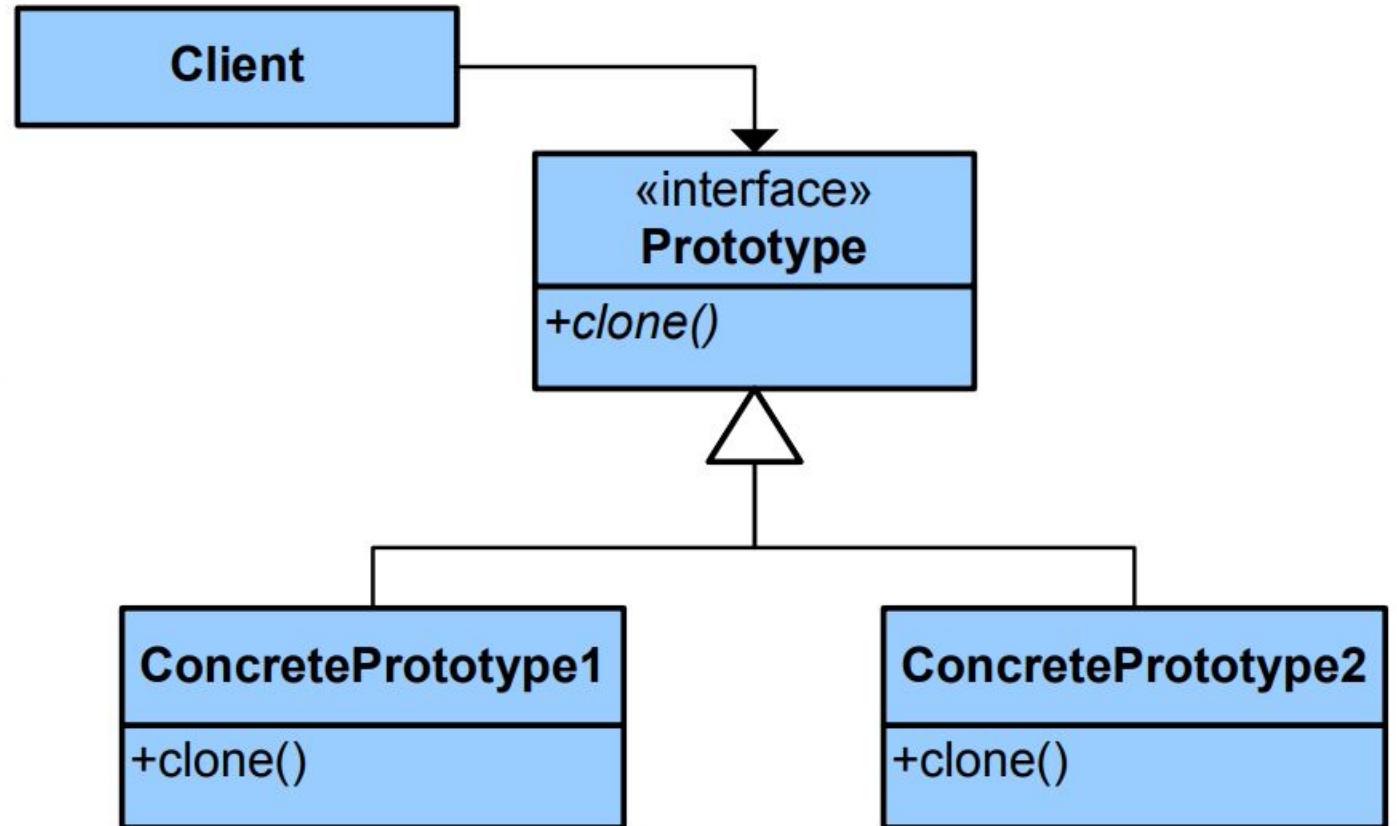


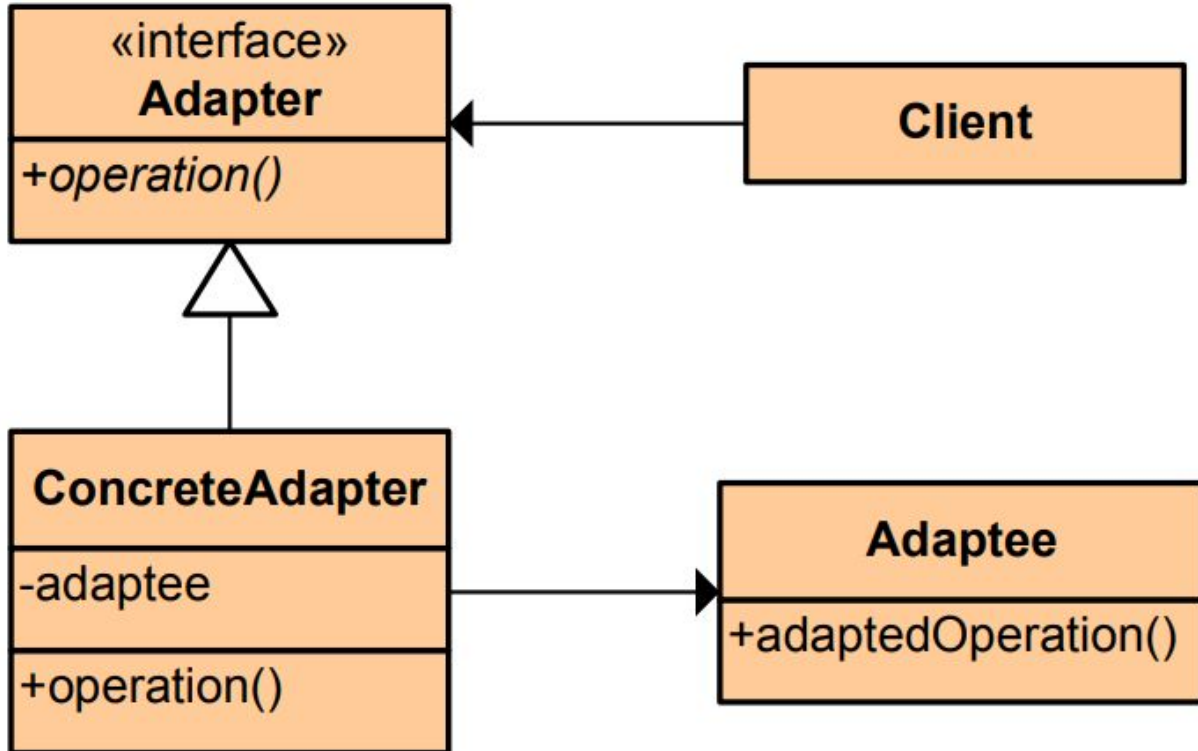
# Prototype

**Type:** Creational

**What it is:**

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



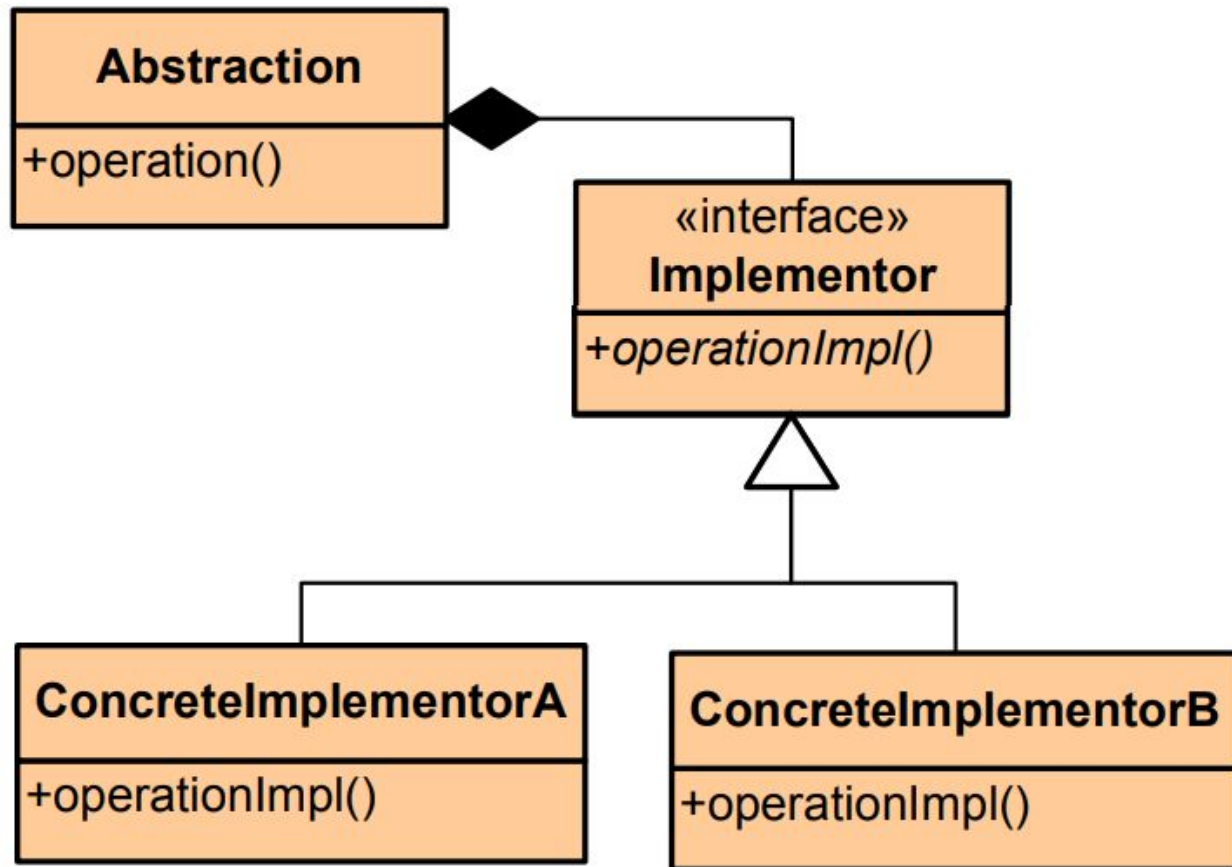


# Adapter

**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.



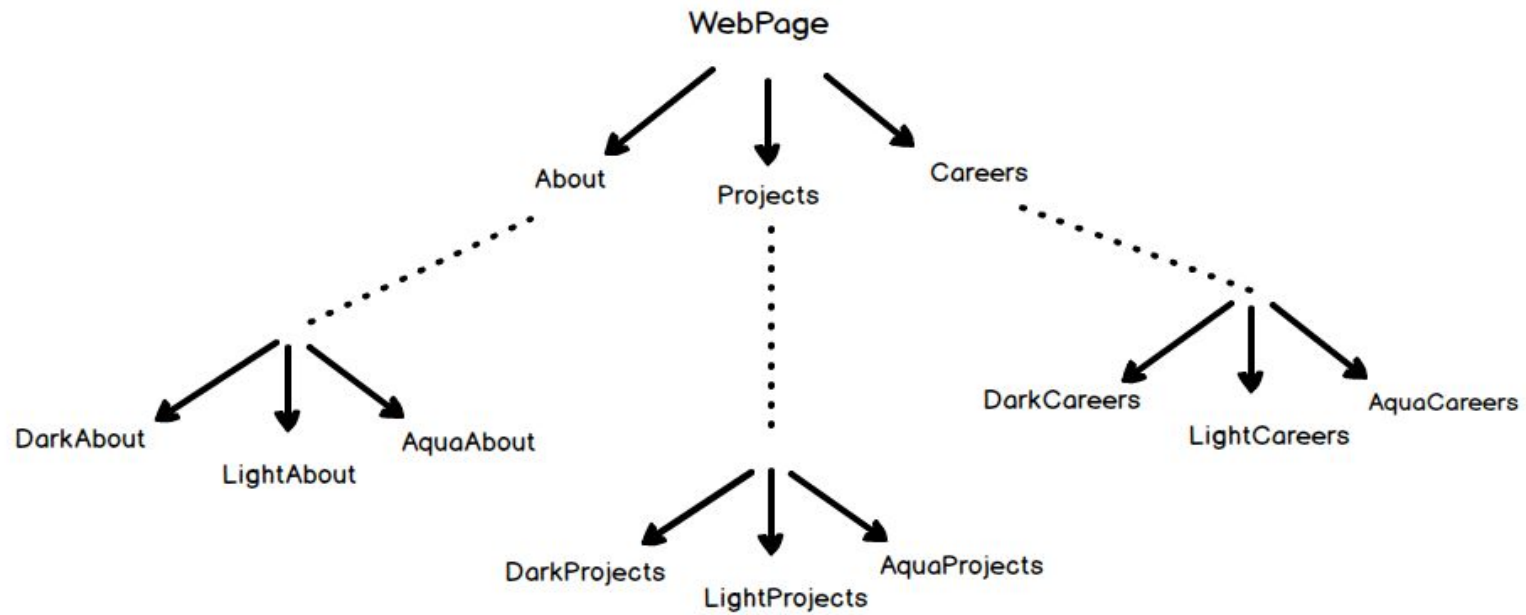
## Bridge

**Type:** Structural

**What it is:**

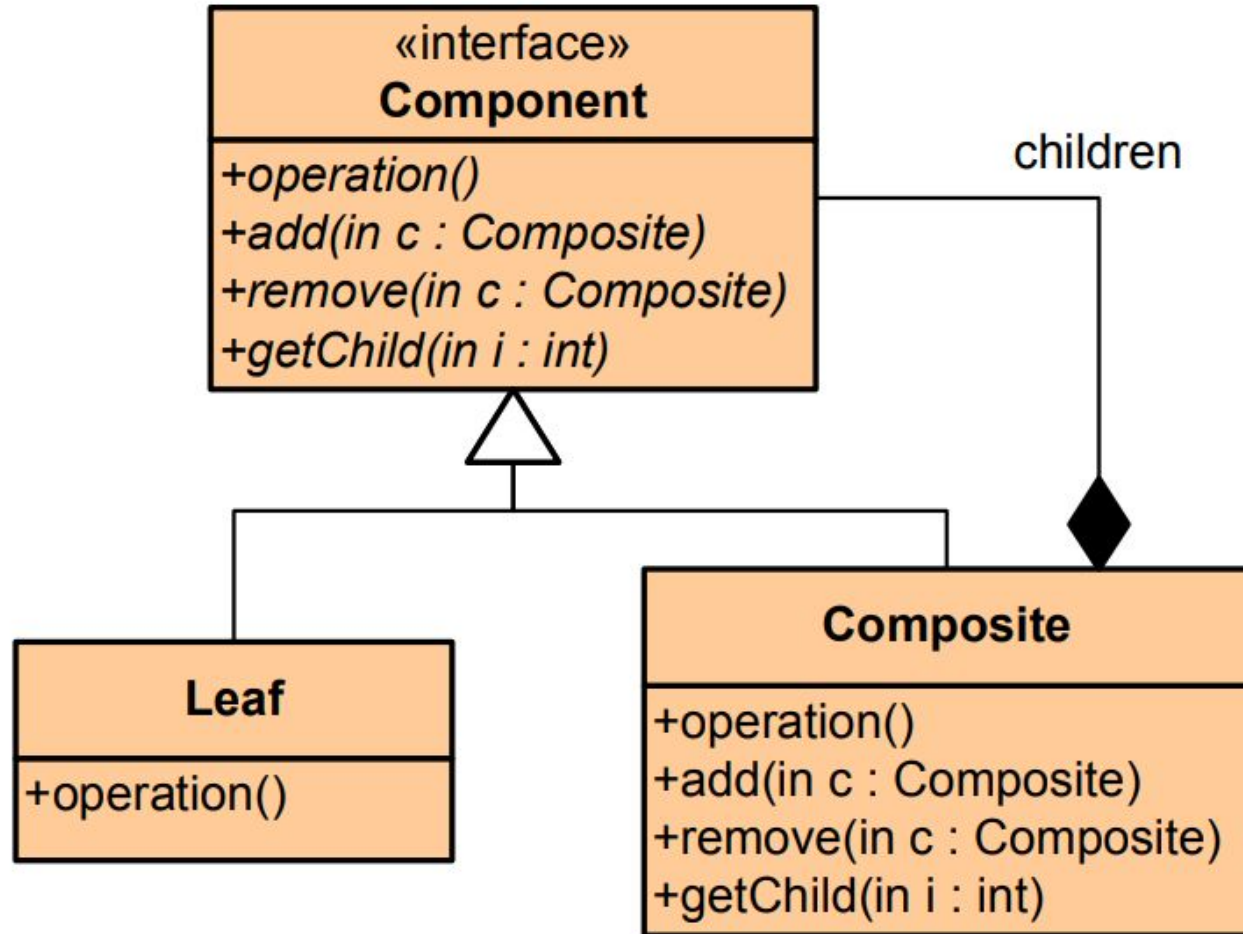
Decouple an abstraction from its implementation so that the two can vary independently.

## Without Bridge



## With Bridge





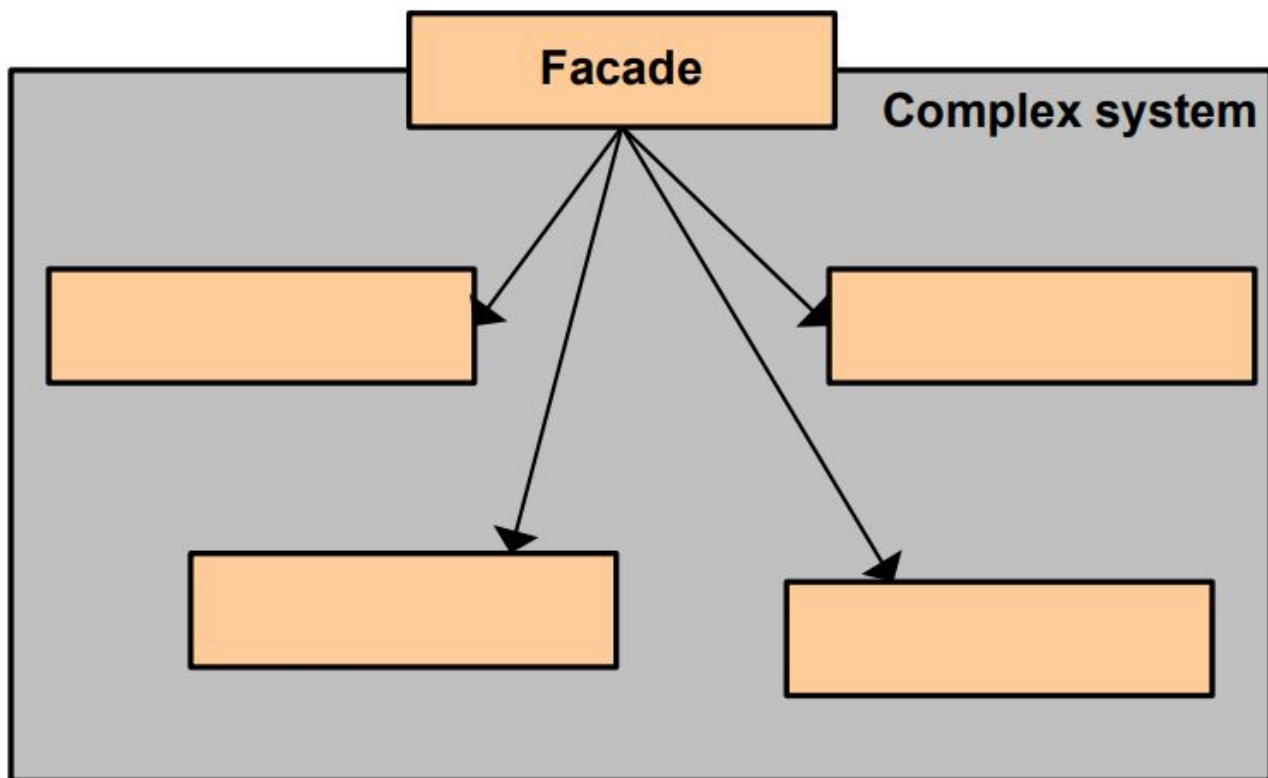
# Composite

**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.





# Facade

**Type:** Structural

**What it is:**

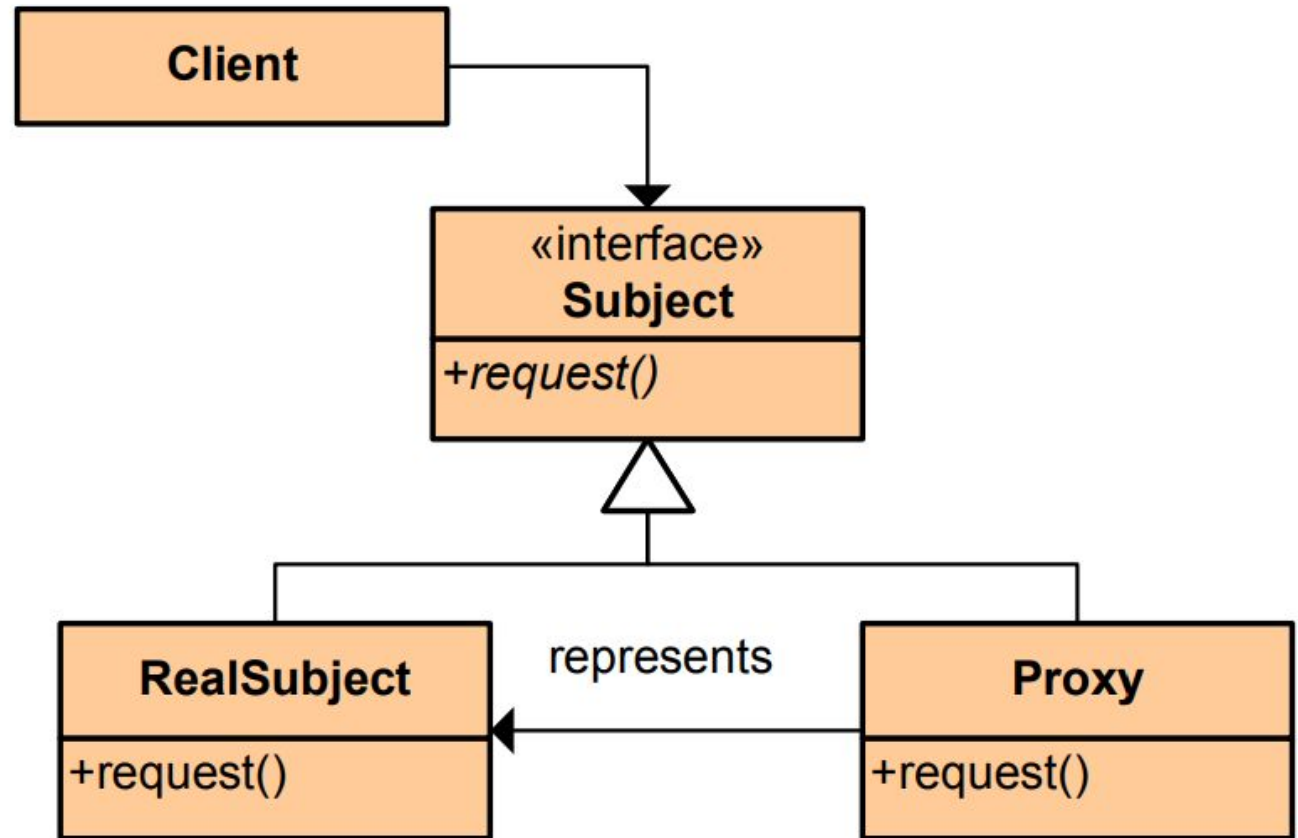
Provide a unified interface to a set of interfaces in a subsystem. Defines a high-level interface that makes the subsystem easier to use.

# Proxy

**Type:** Structural

**What it is:**

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

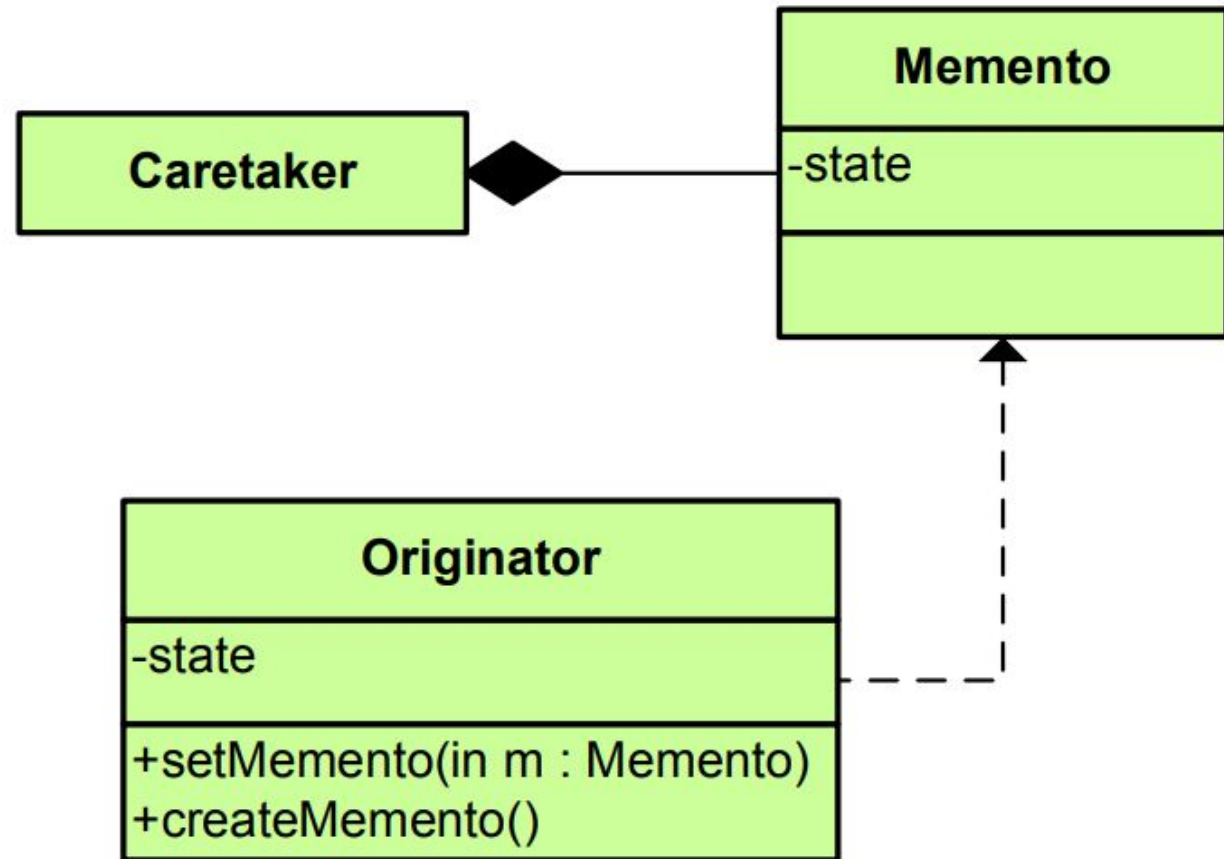


# Memento

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.

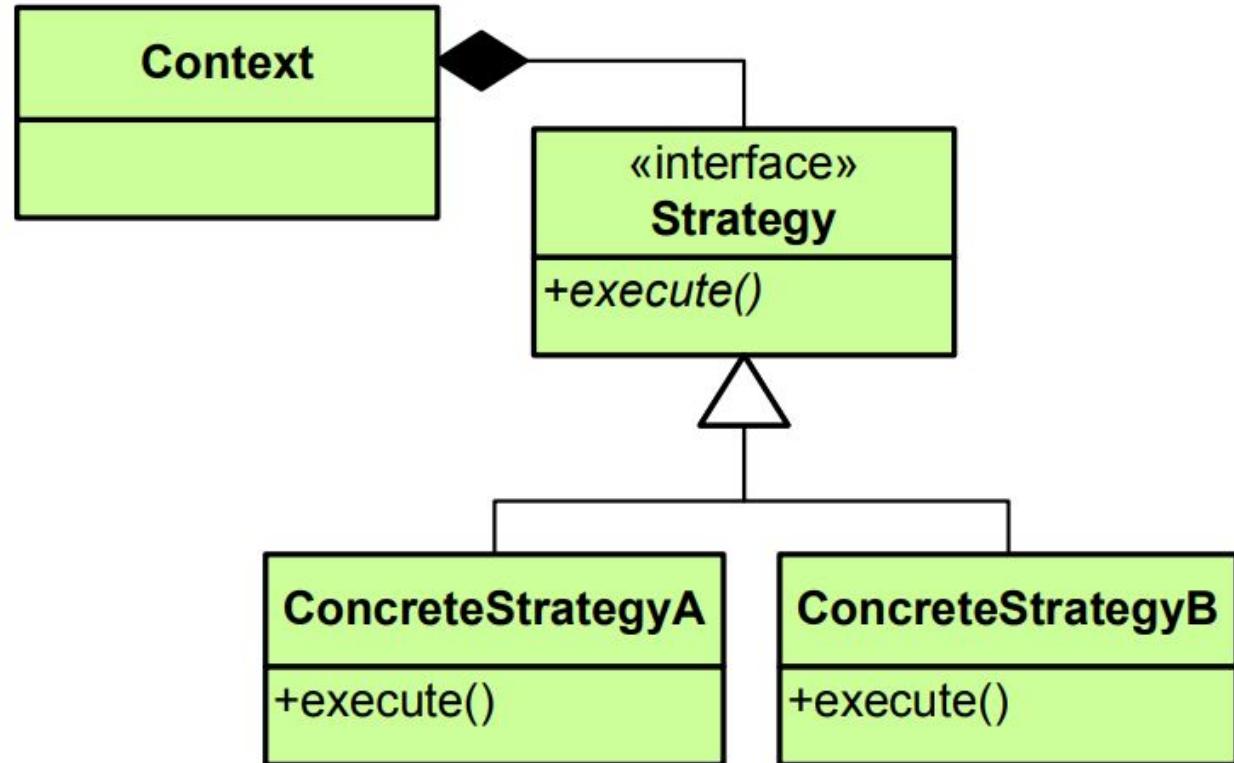


# Strategy

**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.

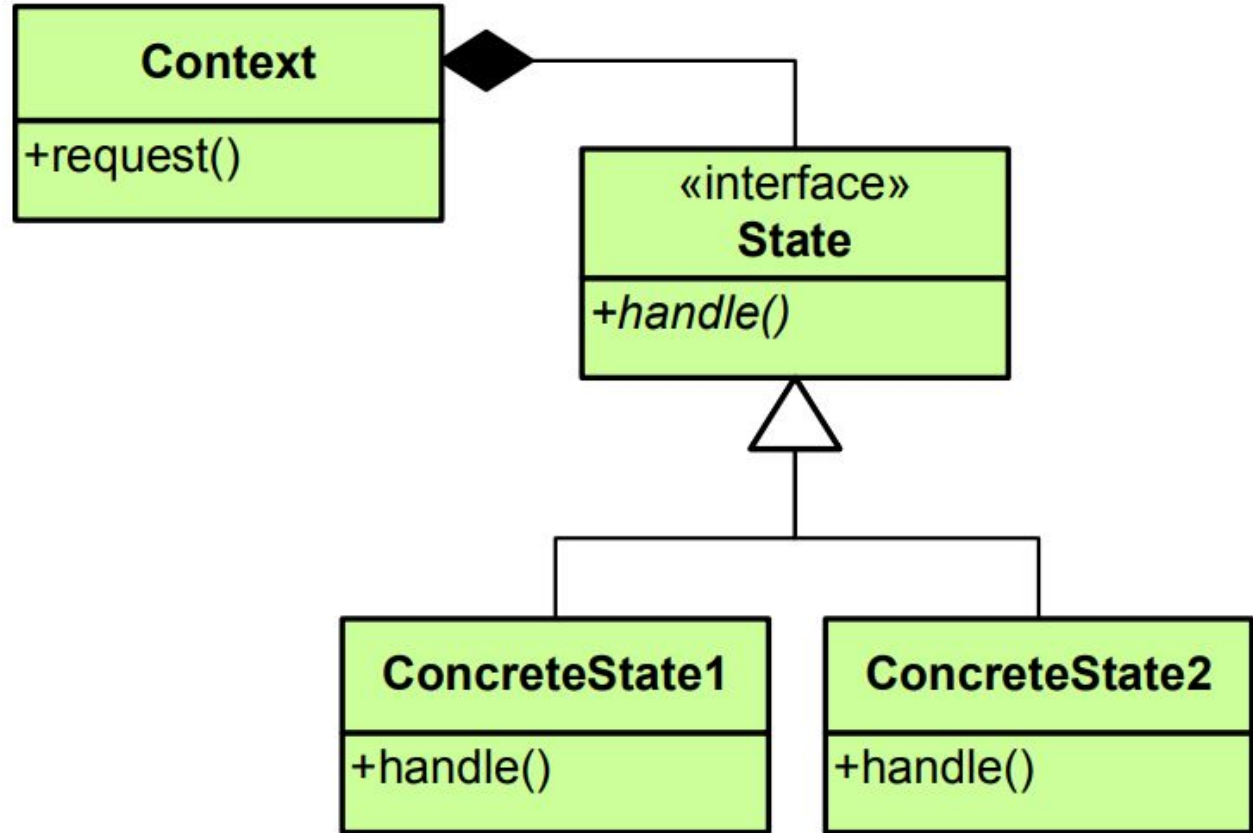


# State

**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.

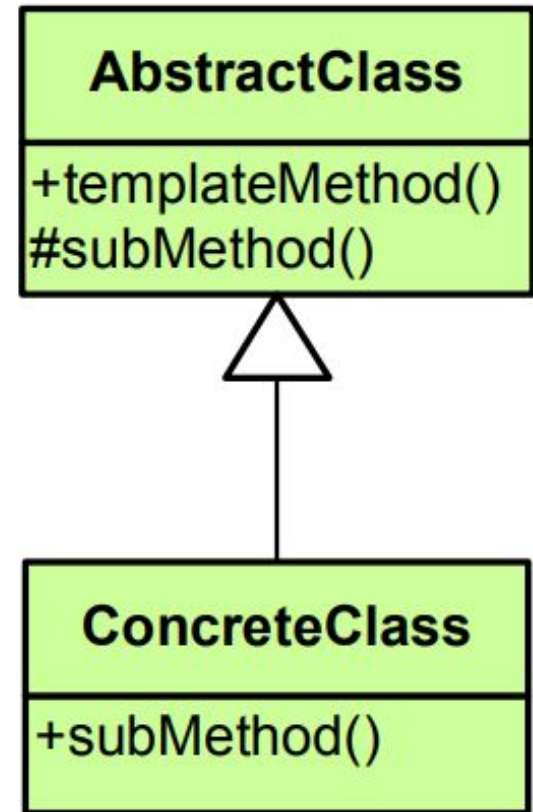


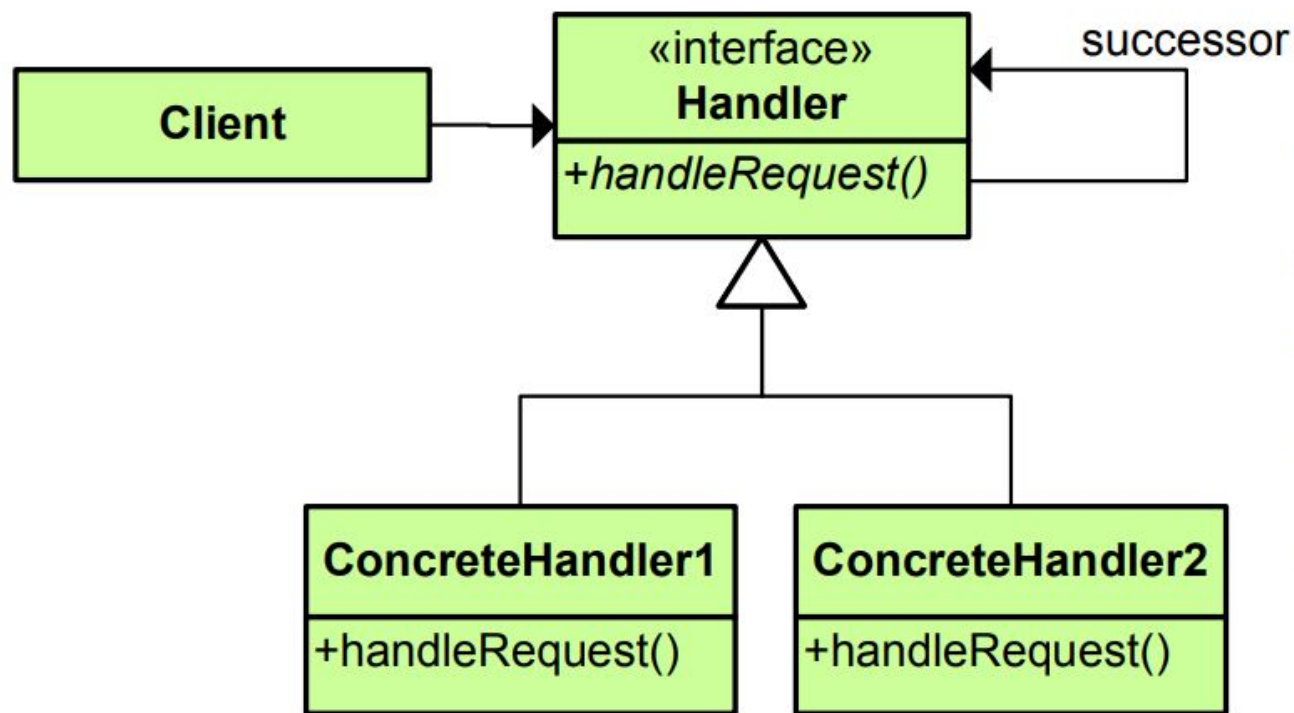
# Template Method

**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.





## 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.