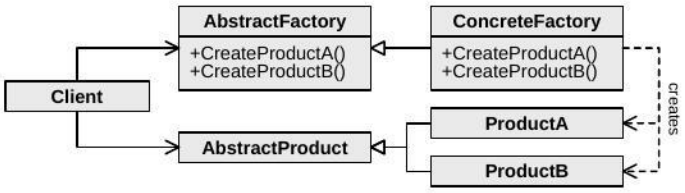
Design Patterns Cheat Sheet

Creational Patterns

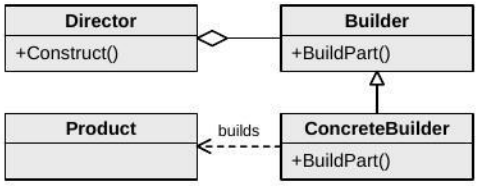
Abstract Factory

Provides an interface for creating families of related or dependent objects without specifying their concrete classes



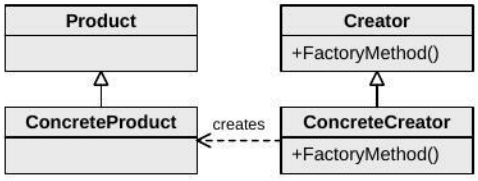
Builder

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



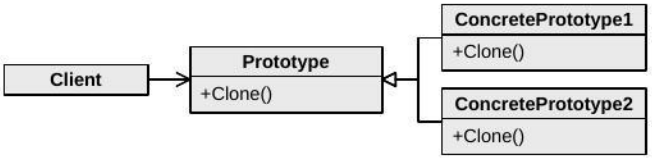
Factory Method

Defines an interface for creating an object but let subclasses decide which class to instantiate



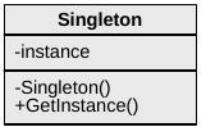
Prototype

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



Singleton

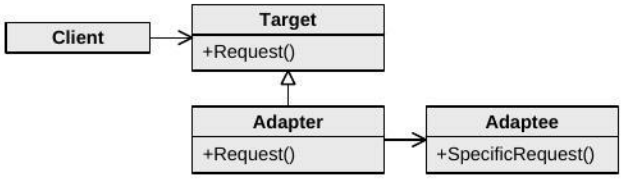
Ensure a class only has one instance and provide a global point of access to it



Structural Patterns

Adapter

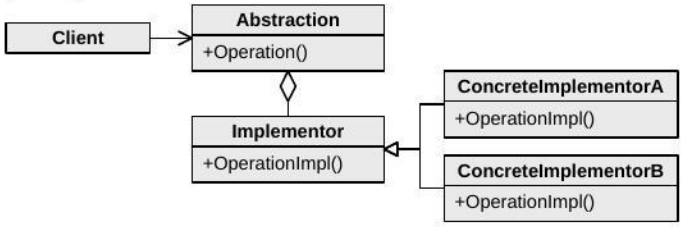
Converts the interface of a class into another interface clients expect



Structural Patterns (cont'd)

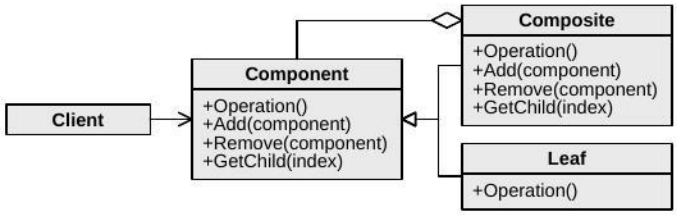
Bridge

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



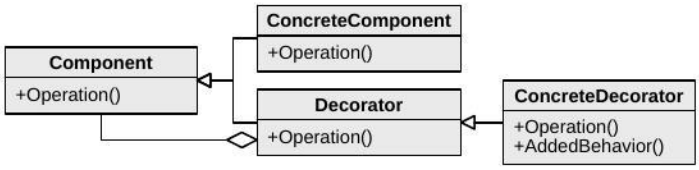
Composite

Composes objects into tree structures to represent part-whole hierarchies



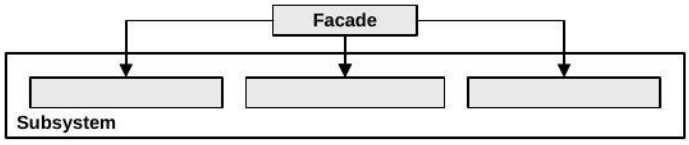
Decorator

Attaches additional responsibilities to an object dynamically



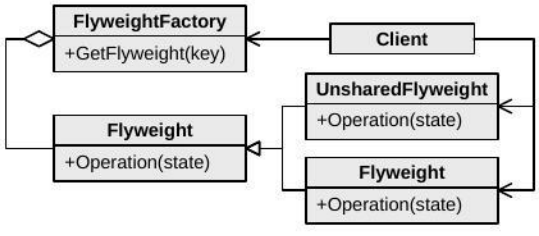
Facade

Provides a unified interface to a set of interfaces in a subsystem



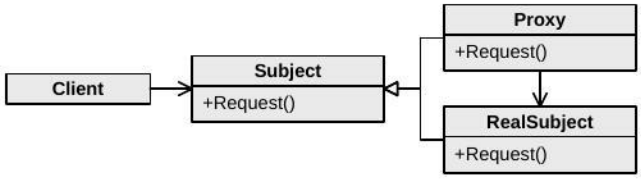
Flyweight

Uses sharing to support large numbers of fine-grained objects efficiently



Proxy

Provides a surrogate or placeholder for another object to control access to it

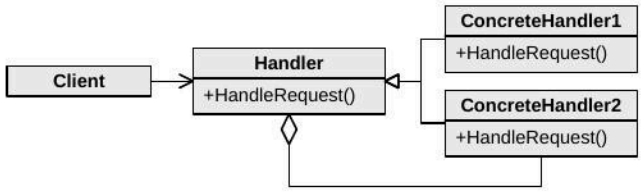


Design Patterns Cheat Sheet

Behavioral Patterns

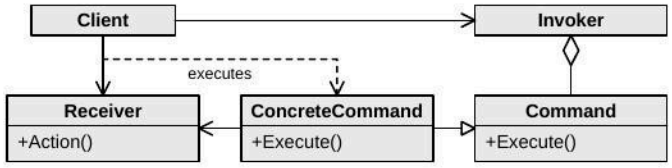
Chain of Responsibility

Avoids coupling the sender of a request to its receiver by giving more than one object a chance to handle the request



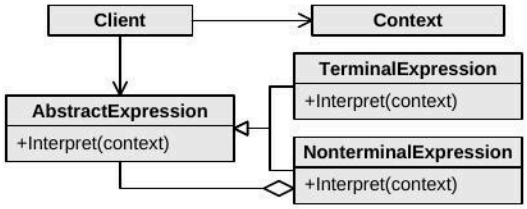
Command

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



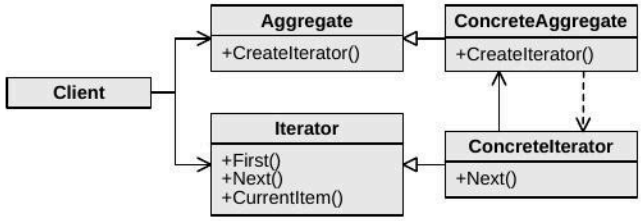
Interpreter

Given a language, defines a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language



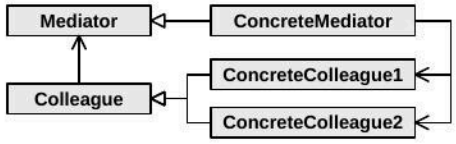
Iterator

Given a language, defines a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language



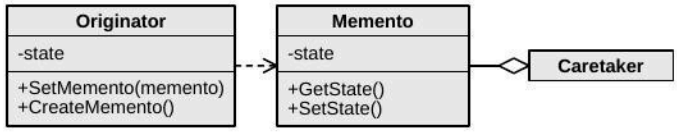
Mediator

Defines an object that encapsulates how a set of objects interact



Memento

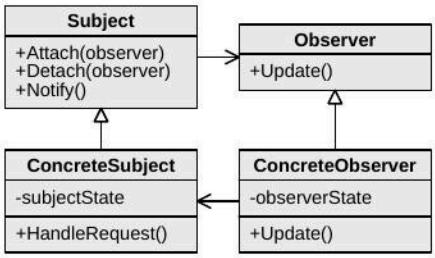
Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later



Behavioral Patterns (cont'd)

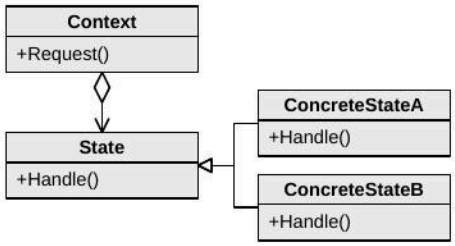
Observer

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



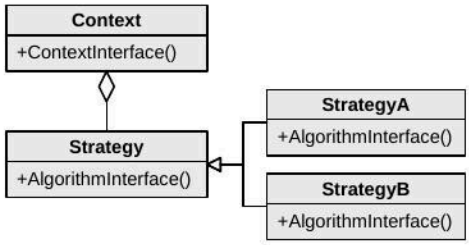
State

Allows an object to alter its behavior when its internal state changes



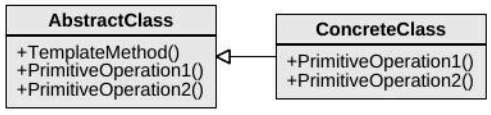
Strategy

Defines a family of algorithms, encapsulate each one, and make them interchangeable



TemplateMethod

Defines the skeleton of an algorithm in an operation, deferring some steps to subclasses



Visitor

Represents an operation to be performed on the elements of an object structure