Lecture #6 on Object-Oriented Modeling

Thierry Géraud

EPITA Research and Development Laboratory (LRDE)

2006

Outline

Outline

Outline

Outline

Definition

Design patterns represent solutions to problems that arise when developing software within a particular context.

Patterns facilitate:

- modularity
- reusability
- efficient talk about software design!

see also: `http://en.wikipedia.org/wiki/Design_Patterns`

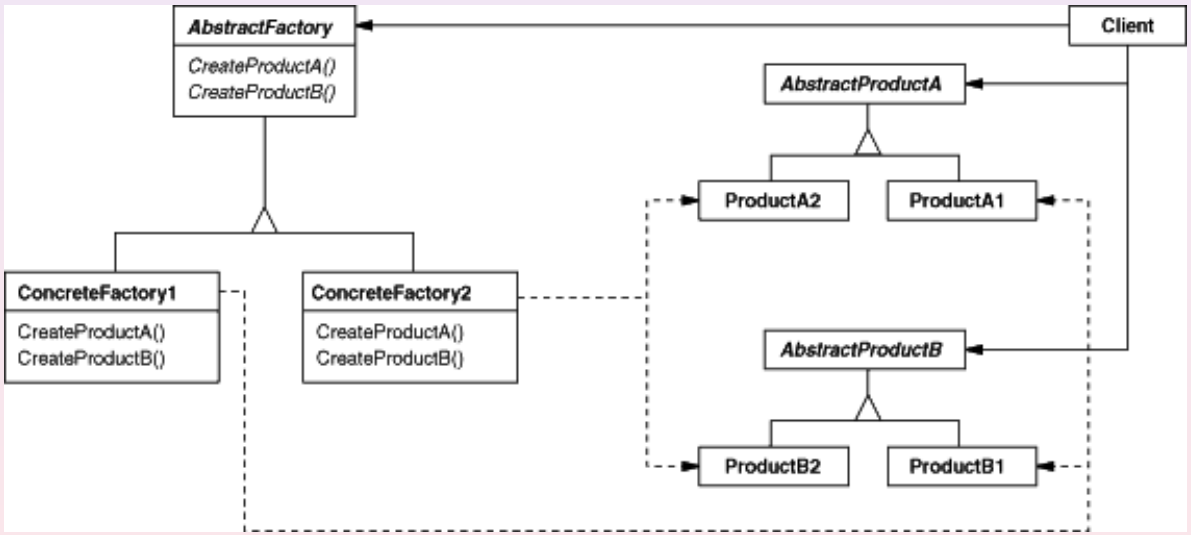
Design Patterns Space

- **Creational patterns**
  deal with initializing and configuring classes and objects.
- **Structural patterns**
  deal with decoupling interface and implementation of classes and objects.
- **Behavioral patterns**
  deal with dynamic interactions among societies of classes and objects.

see also: `http://hillside.net/`

Abstract Factory

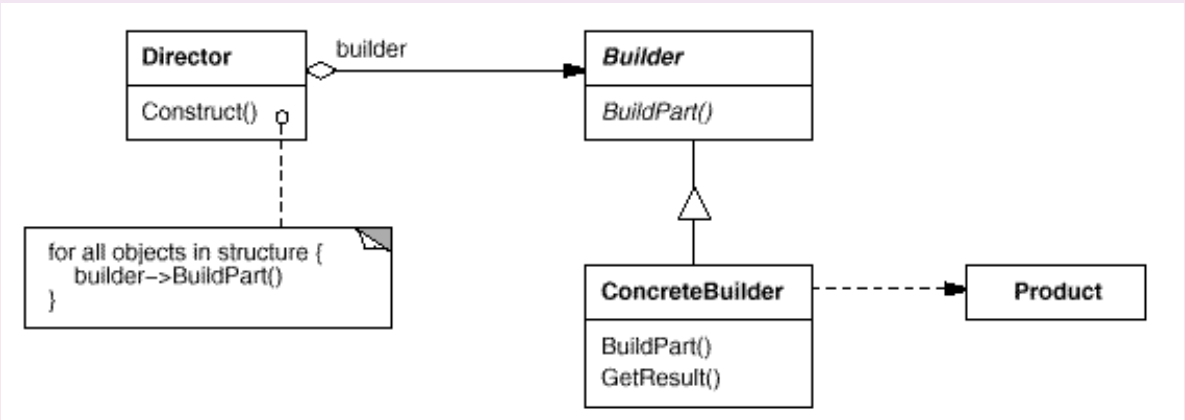
Provide an interface for creating families of related or dependent objects without specifying their concrete classes.



see also: `http://en.wikipedia.org/wiki/Abstract_factory_pattern`

Builder

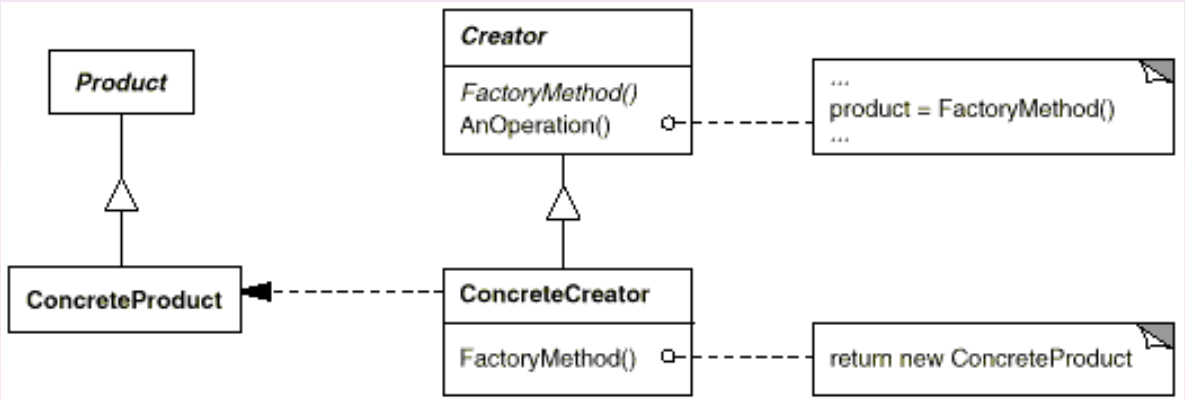
Separate the construction of a complex object from its representation.



see also: `http://en.wikipedia.org/wiki/Builder_pattern`

Factory Method

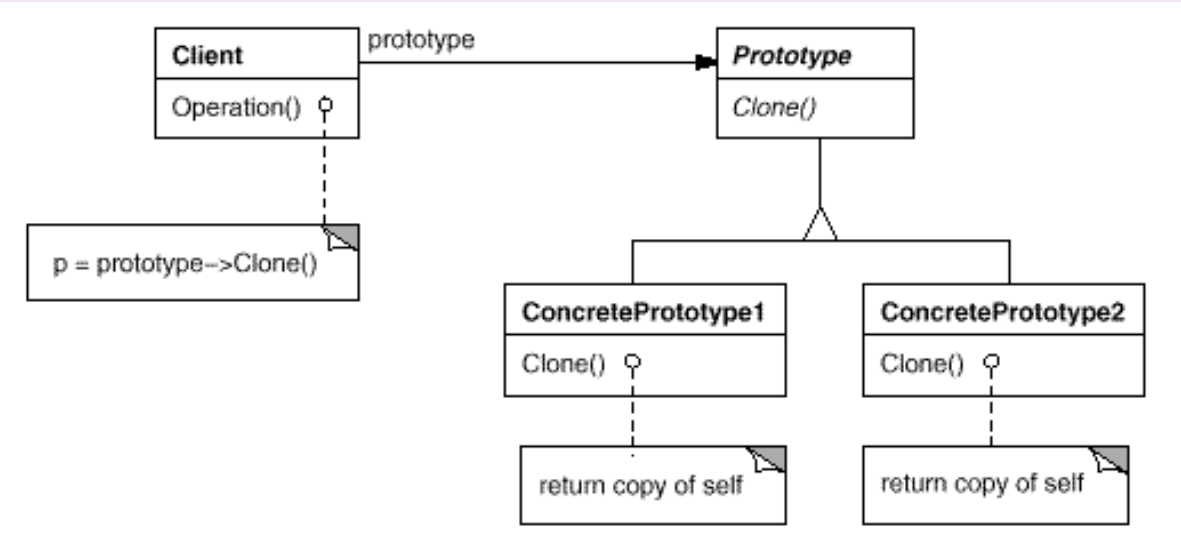
Define an interface for creating an object, but let subclasses decide which class to instantiate.



see also: `http://en.wikipedia.org/wiki/Factory_method_pattern`

Prototype

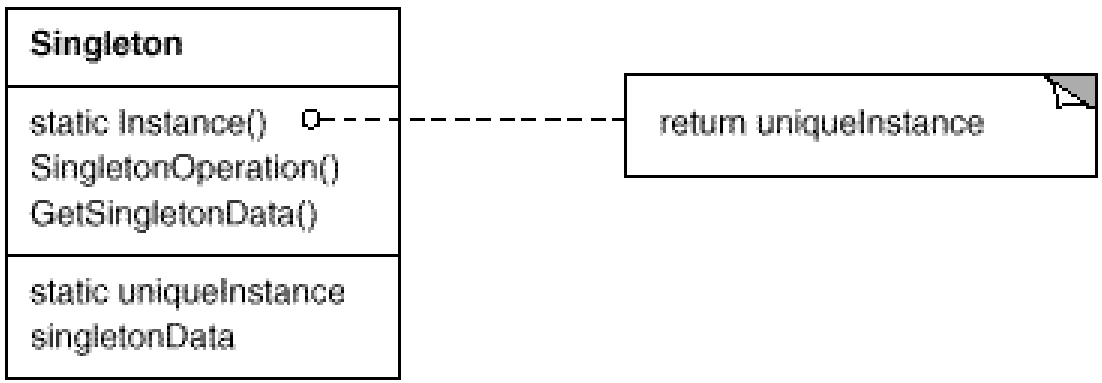
Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.



see also: `http://en.wikipedia.org/wiki/Prototype_pattern`

Singleton

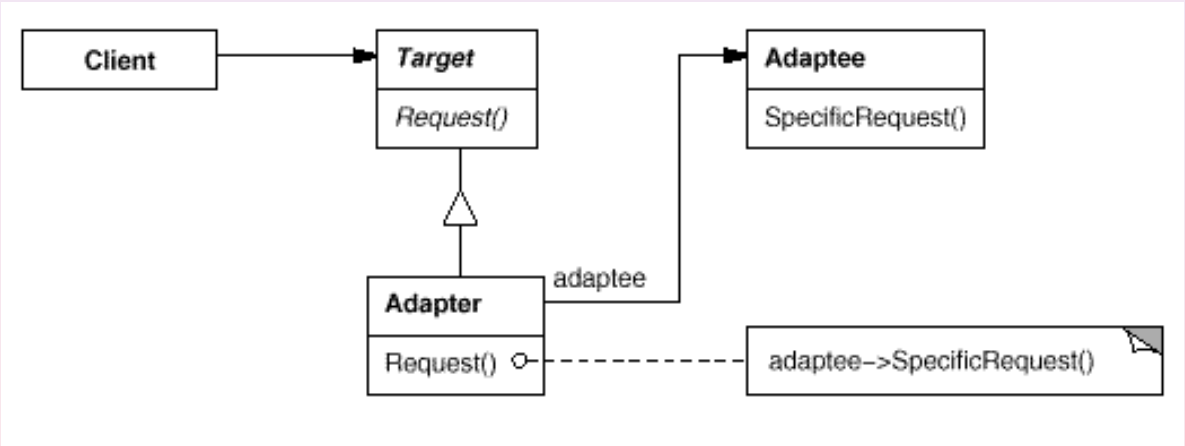
Ensure a class only has one instance, and provide a global point of access to it.



see also: `http://en.wikipedia.org/wiki/Singleton_pattern`

Adapter

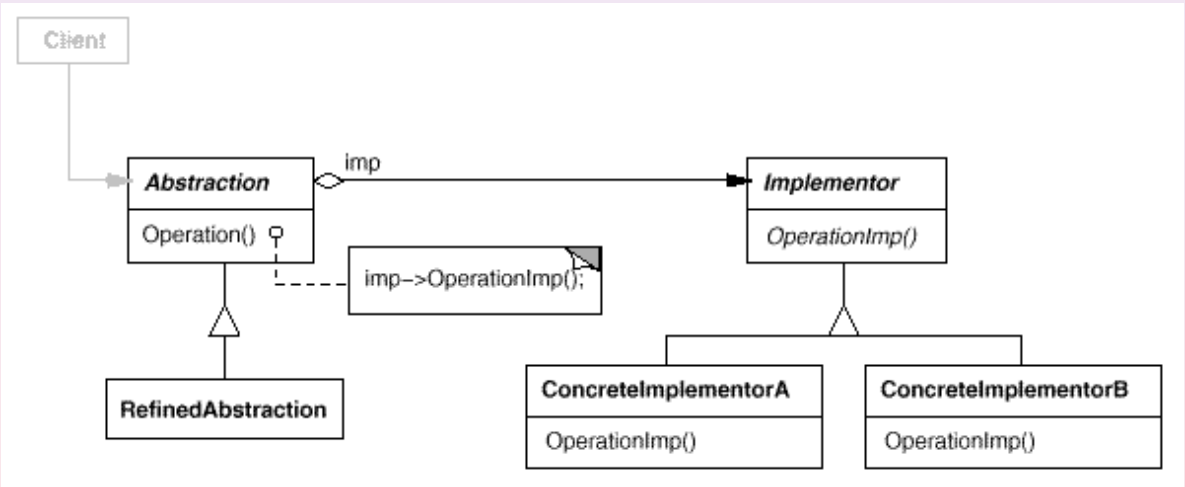
Convert the interface of a class into another interface clients expect.



see also: `http://en.wikipedia.org/wiki/Adapter_pattern`

Bridge

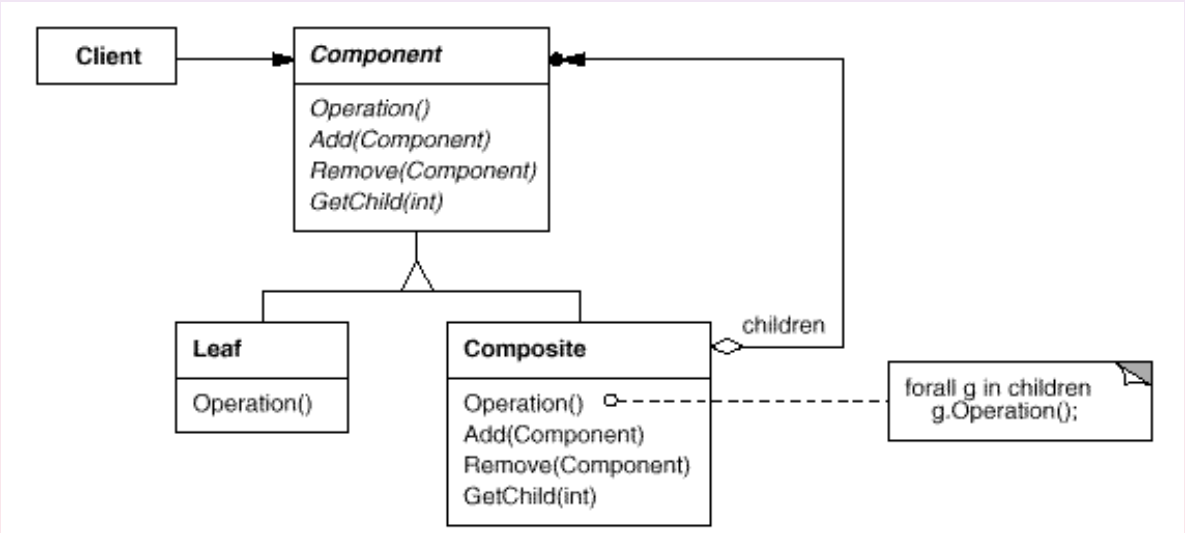
Decouple an abstraction from its implementation so that the two can vary independently.



see also: `http://en.wikipedia.org/wiki/Bridge_pattern`

Composite

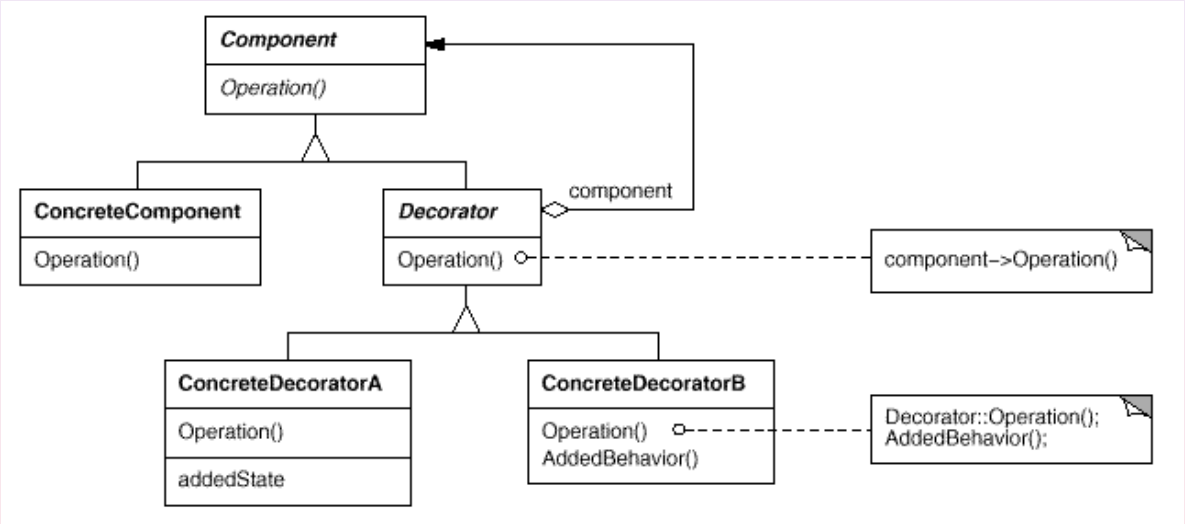
Compose objects into tree structures to represent part-whole hierarchies.



see also: `http://en.wikipedia.org/wiki/Composite_pattern`

Decorator

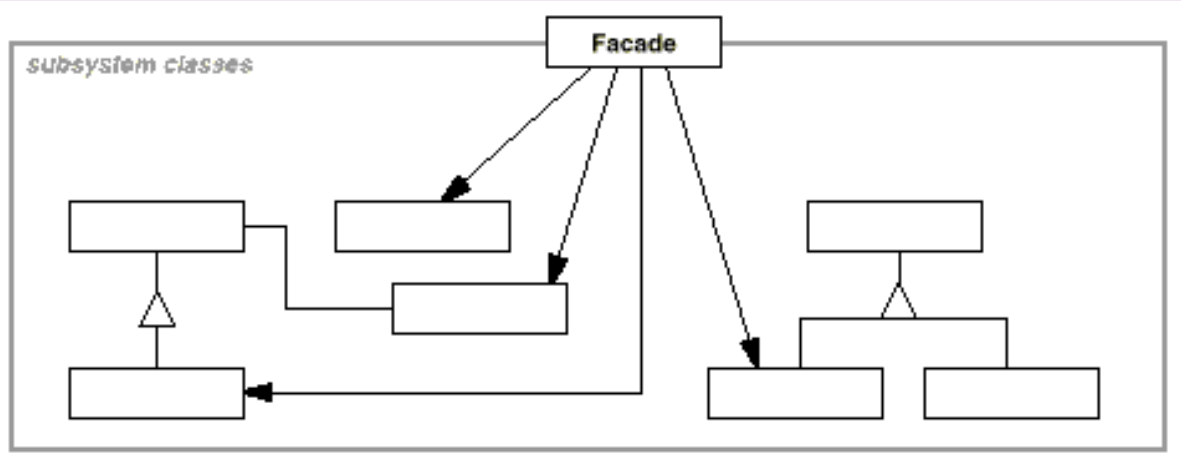
Attach additional responsibilities to an object dynamically.



see also: `http://en.wikipedia.org/wiki/Decorator_pattern`

Facade

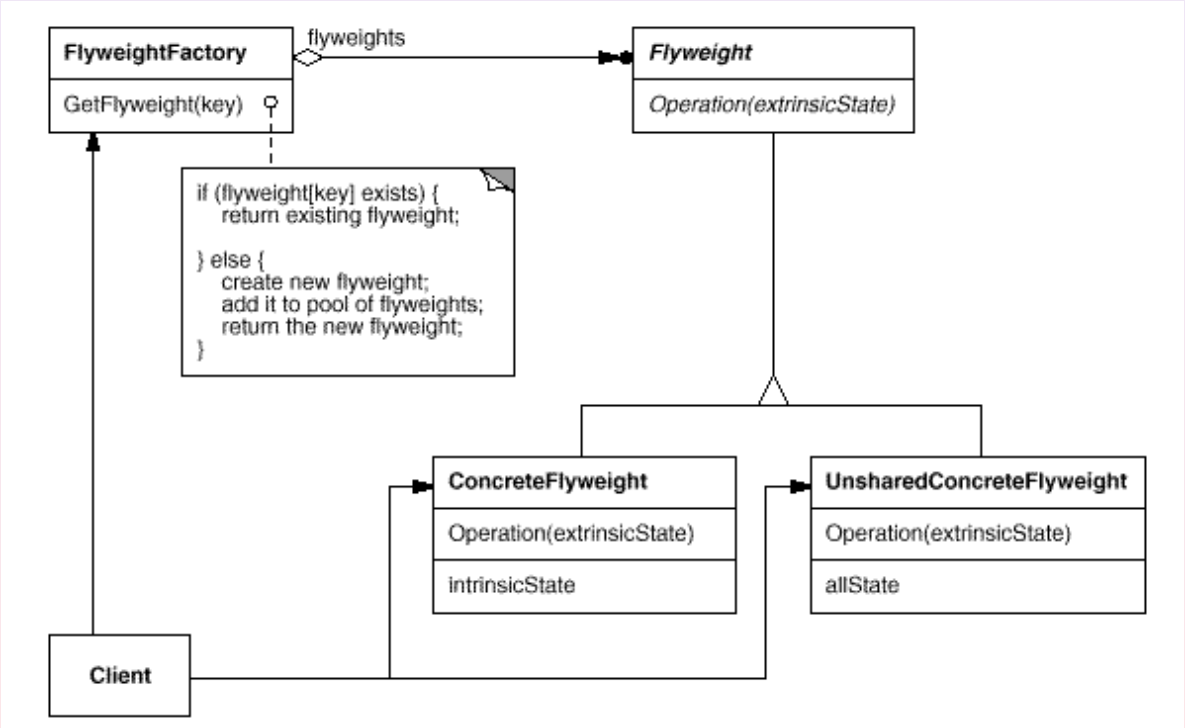
Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.



see also: `http://en.wikipedia.org/wiki/Facade_pattern`

Flyweight

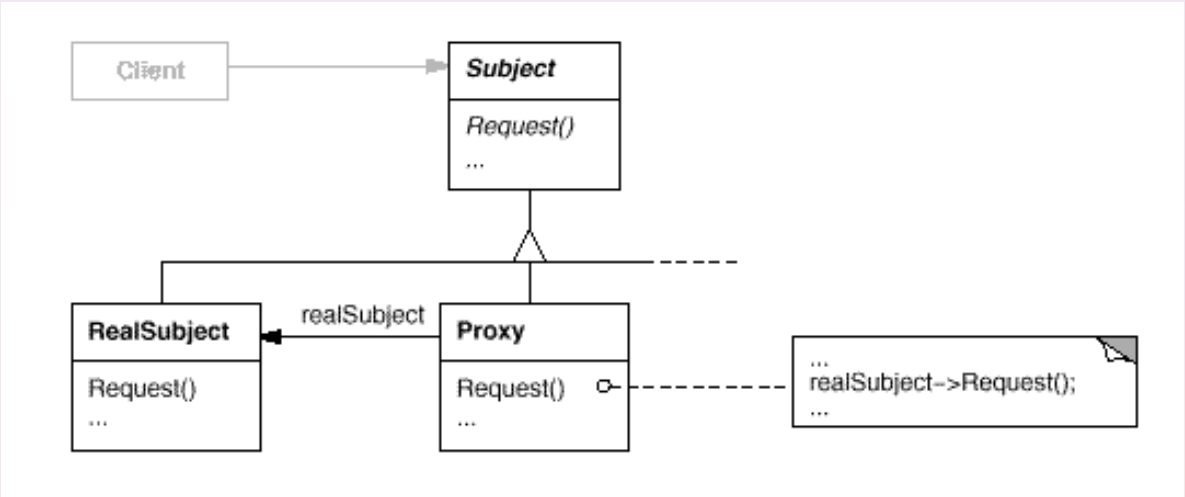
Use sharing to support large numbers of fine-grained objects.



see also: `http://en.wikipedia.org/wiki/Flyweight_pattern`

Proxy

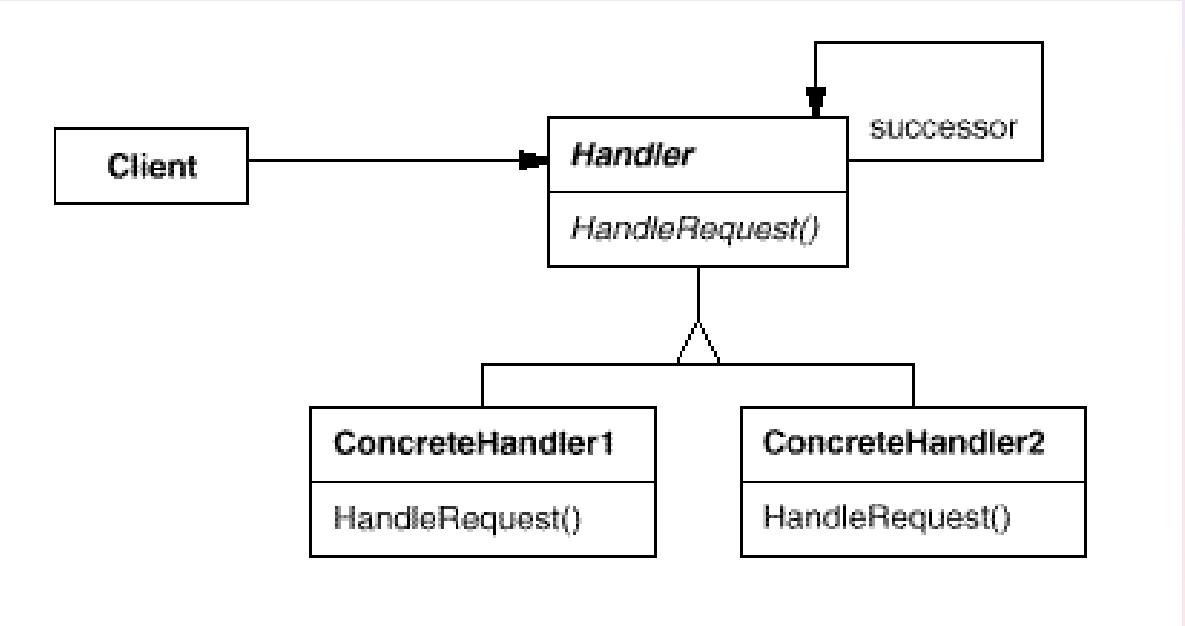
Provide a surrogate or placeholder for another object to control access to it.



see also: `http://en.wikipedia.org/wiki/Proxy_pattern`

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.

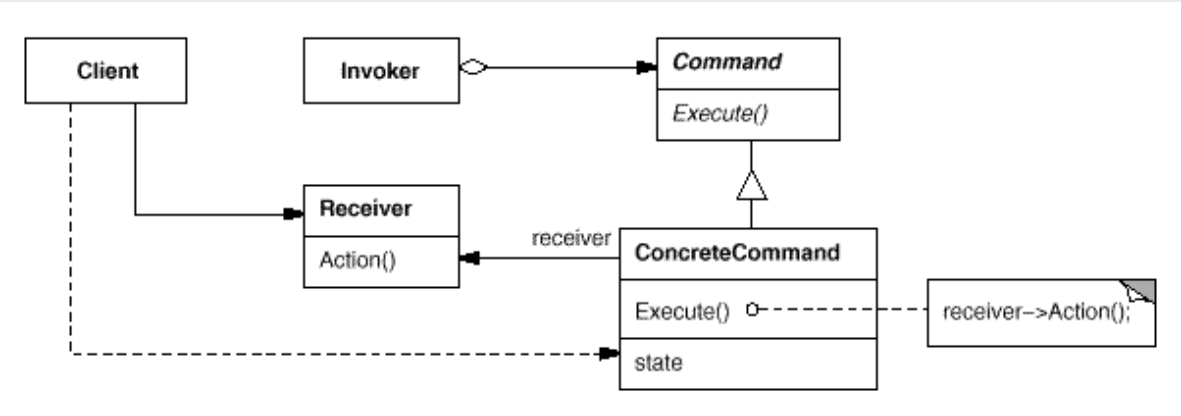


see also:

`http://en.wikipedia.org/wiki/Chain_of_responsibility_pattern`

Command

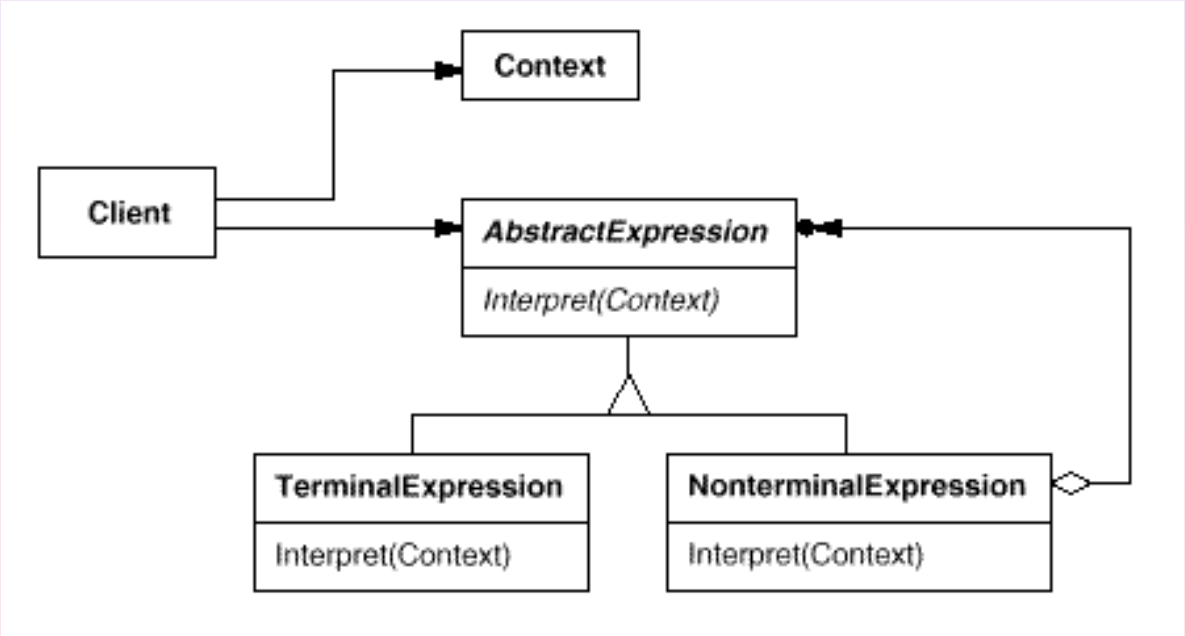
Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.



see also: `http://en.wikipedia.org/wiki/Command_pattern`

Interpreter

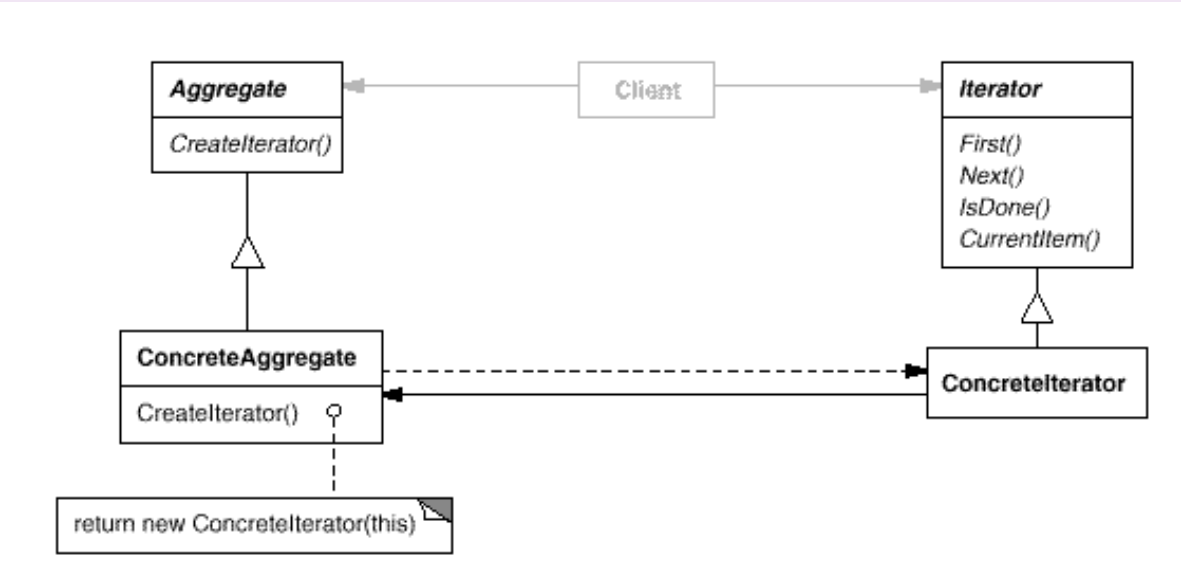
Given a language, define a represention for its grammar along with an interpreter.



see also: `http://en.wikipedia.org/wiki/Interpreter_pattern`

Iterator

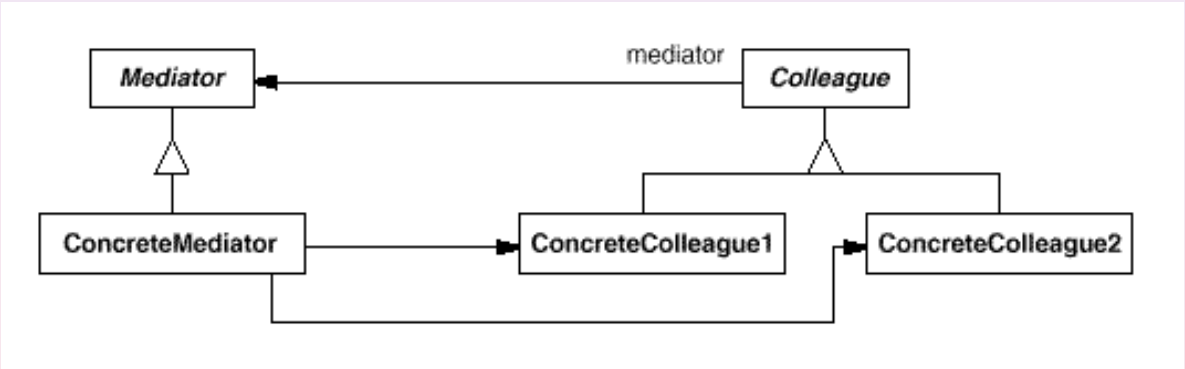
Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.



see also: `http://en.wikipedia.org/wiki/Iterator_pattern`

Mediator

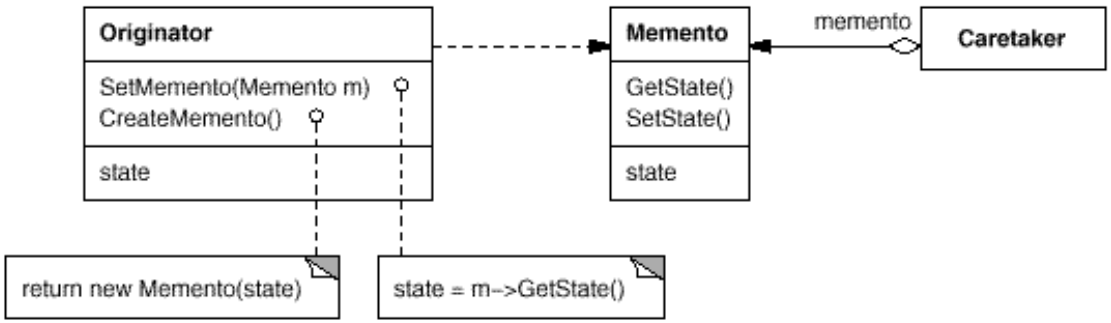
Define an object that encapsulates how a set of objects interact.



see also: `http://en.wikipedia.org/wiki/Mediator_pattern`

Memento

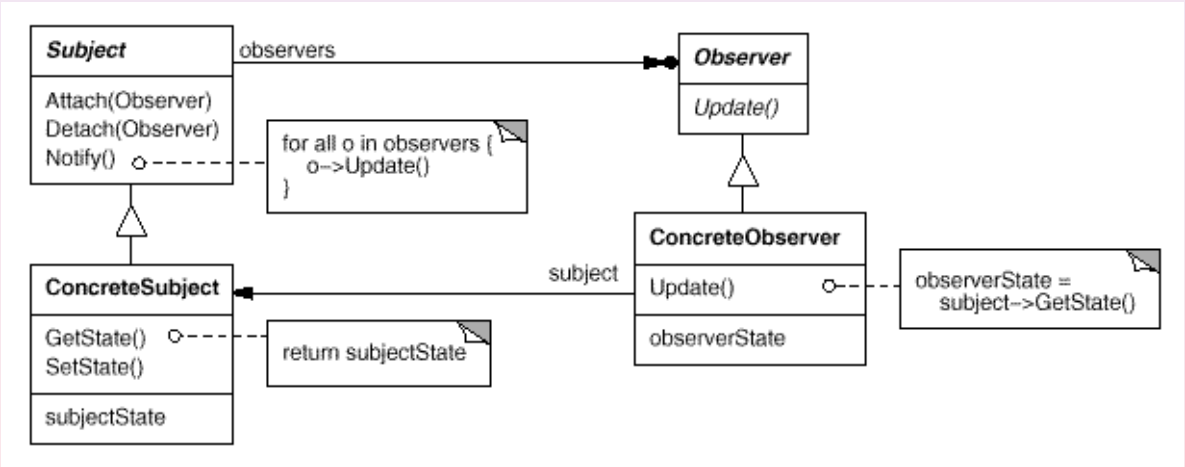
Capture and externalize an object's internal state so that the object can be restored to this state later.



see also: `http://en.wikipedia.org/wiki/Memento_pattern`

Observer

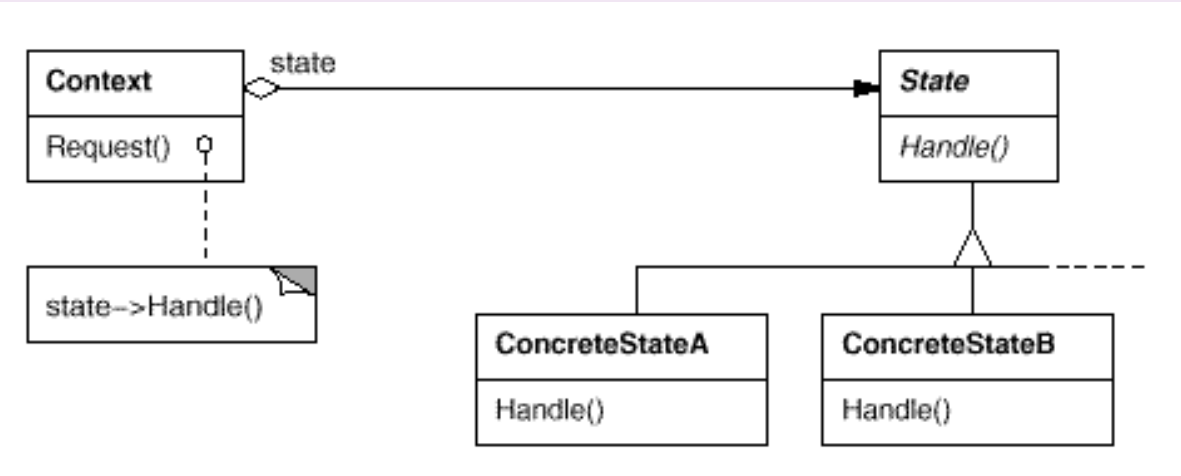
Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.



see also: `http://en.wikipedia.org/wiki/Observer_pattern`

State

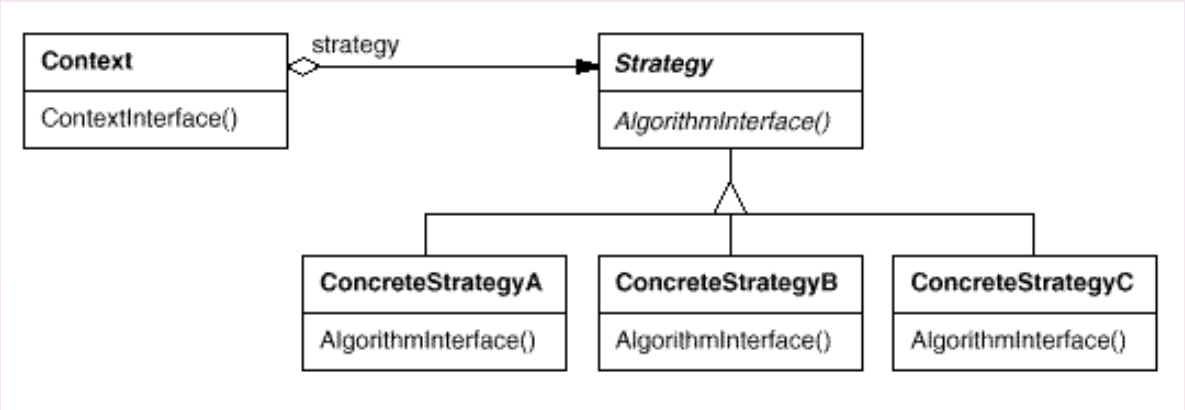
Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.



see also: `http://en.wikipedia.org/wiki/State_pattern`

Strategy

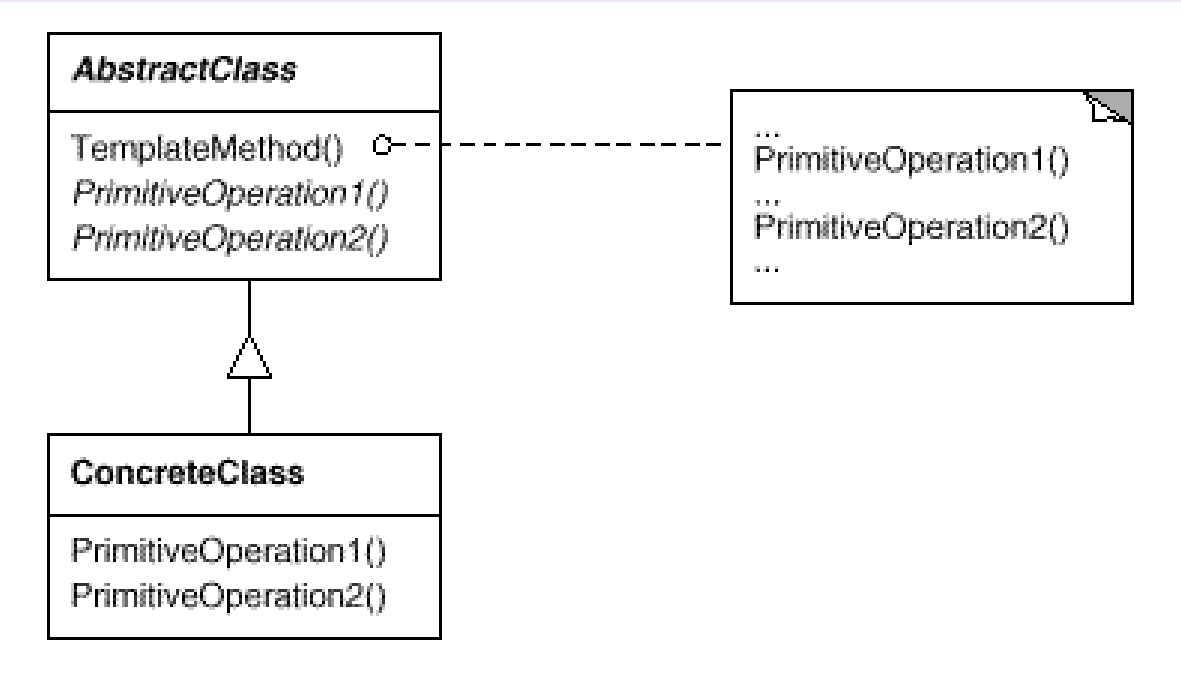
Define a family of algorithms, encapsulate each one, and make them interchangeable.



see also: `http://en.wikipedia.org/wiki/Strategy_pattern`

Template Method

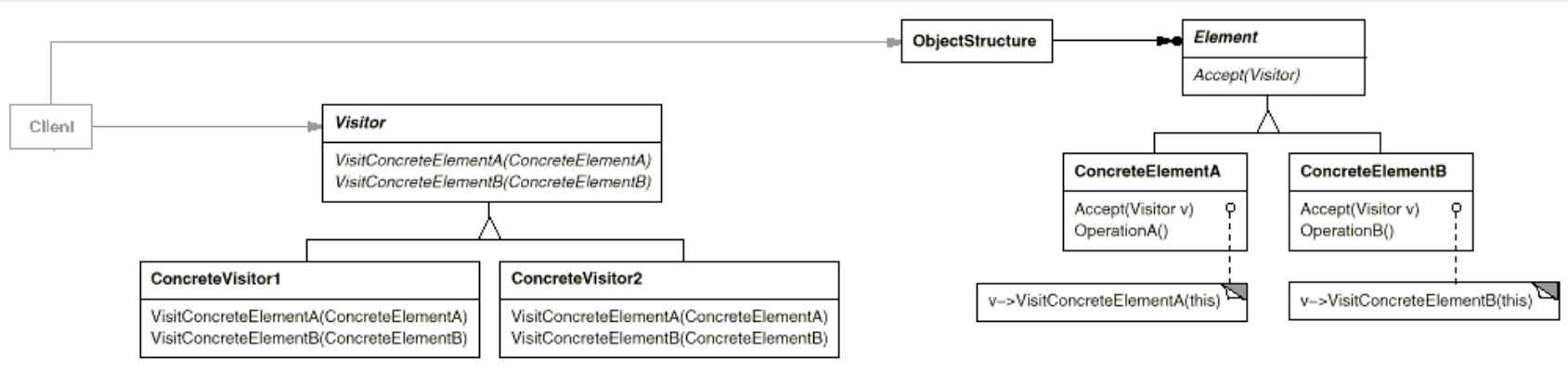
Define the skeleton of an algorithm in an operation, deferring some steps to subclasses.



see also: `http://en.wikipedia.org/wiki/Template_method_pattern`

Visitor

Represent an operation to be performed on the elements of an object structure.



see also: `http://en.wikipedia.org/wiki/Visitor_pattern`