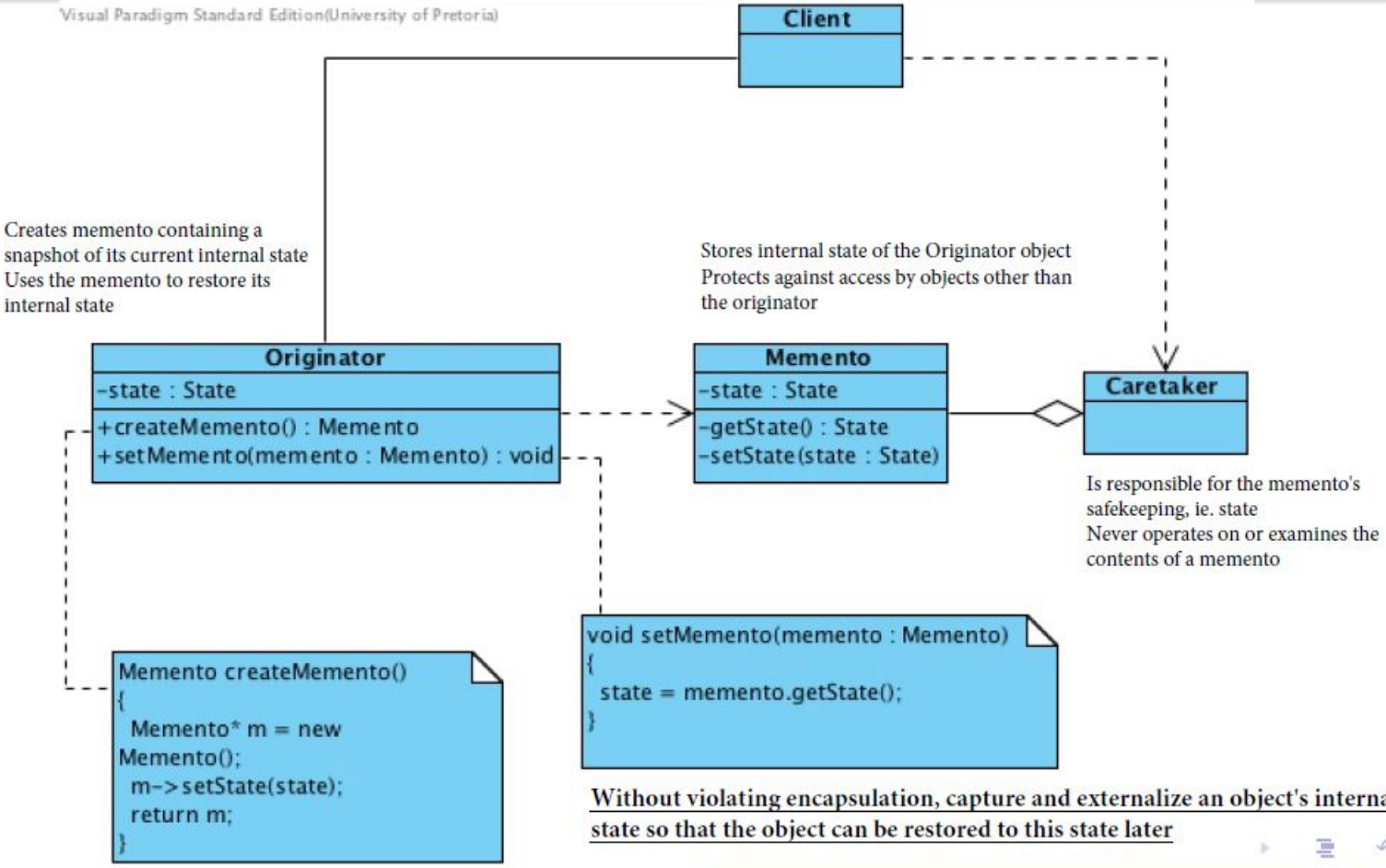
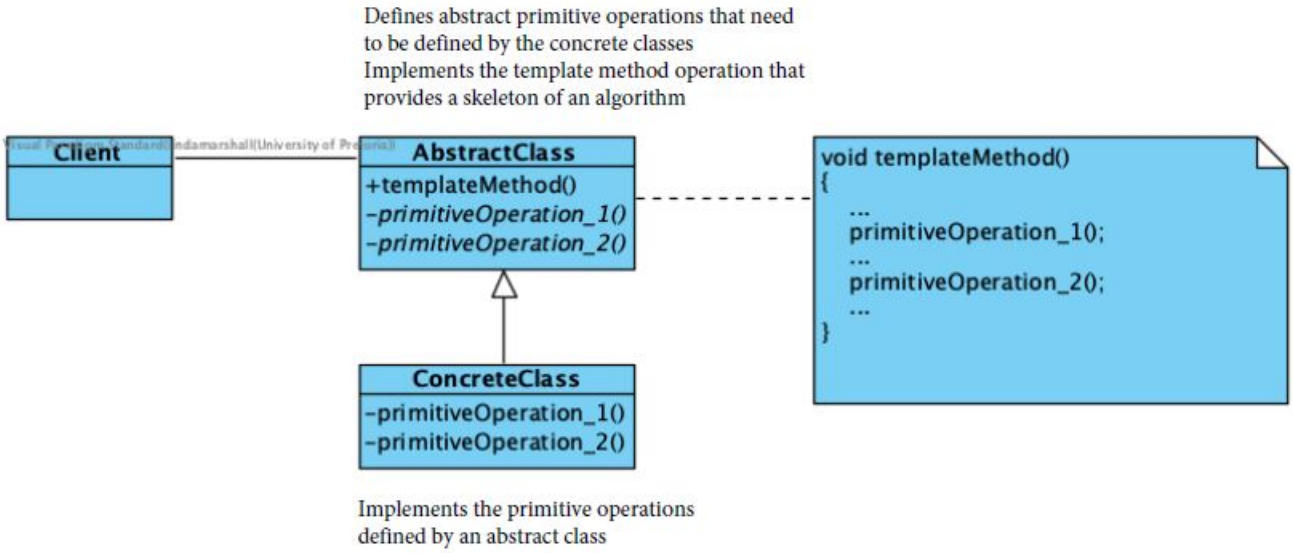
MEMENTO



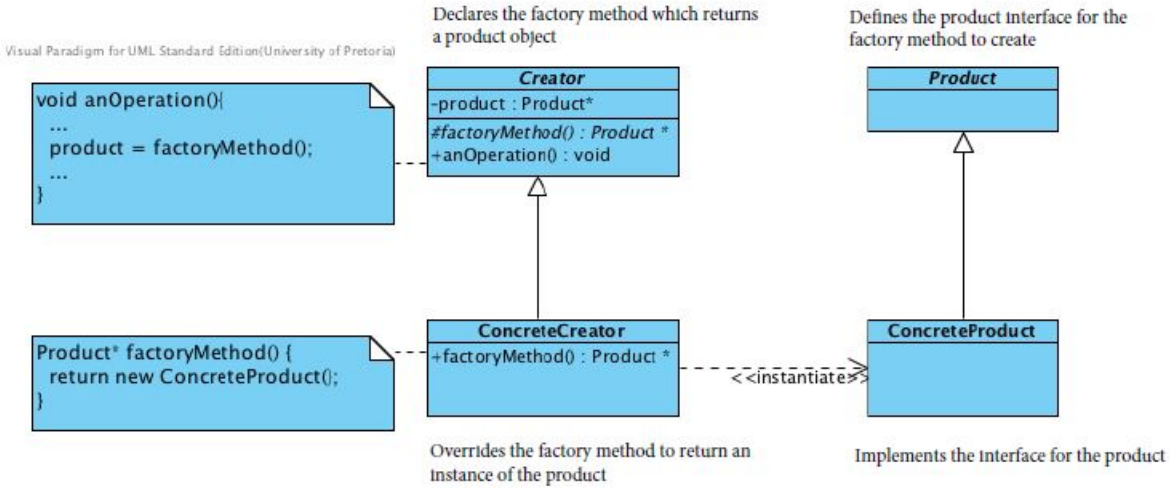
Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later

TEMPLATE METHOD



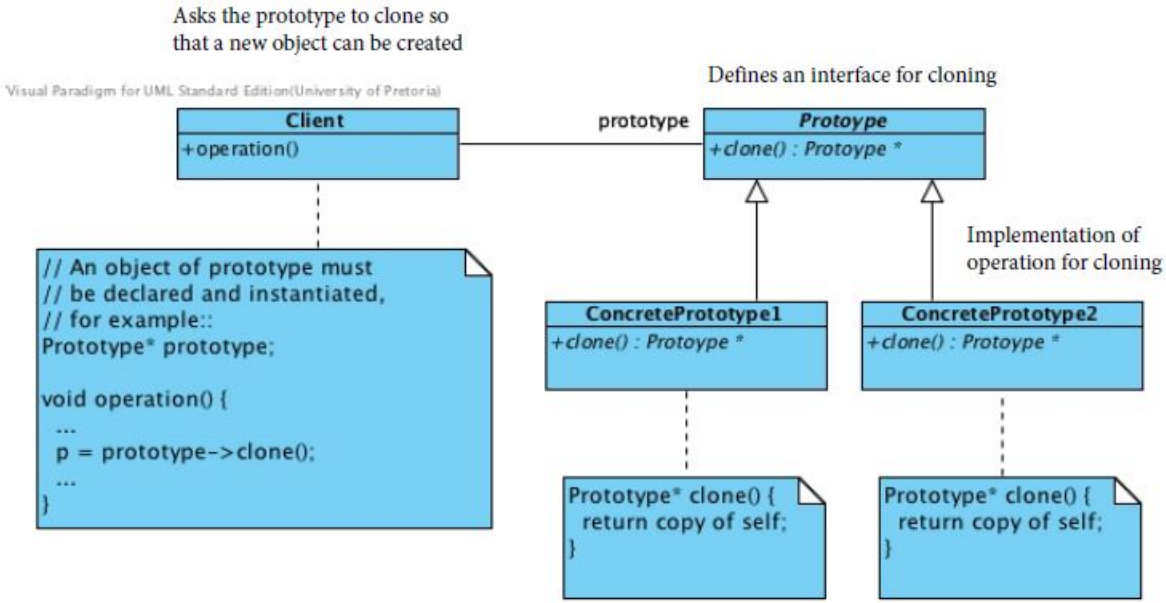
Define a skeleton of an algorithm in an operation, deferring some steps to subclasses. Lets subclasses redefine certain steps of an algorithm without changing the algorithms structure.

FACTORY METHOD



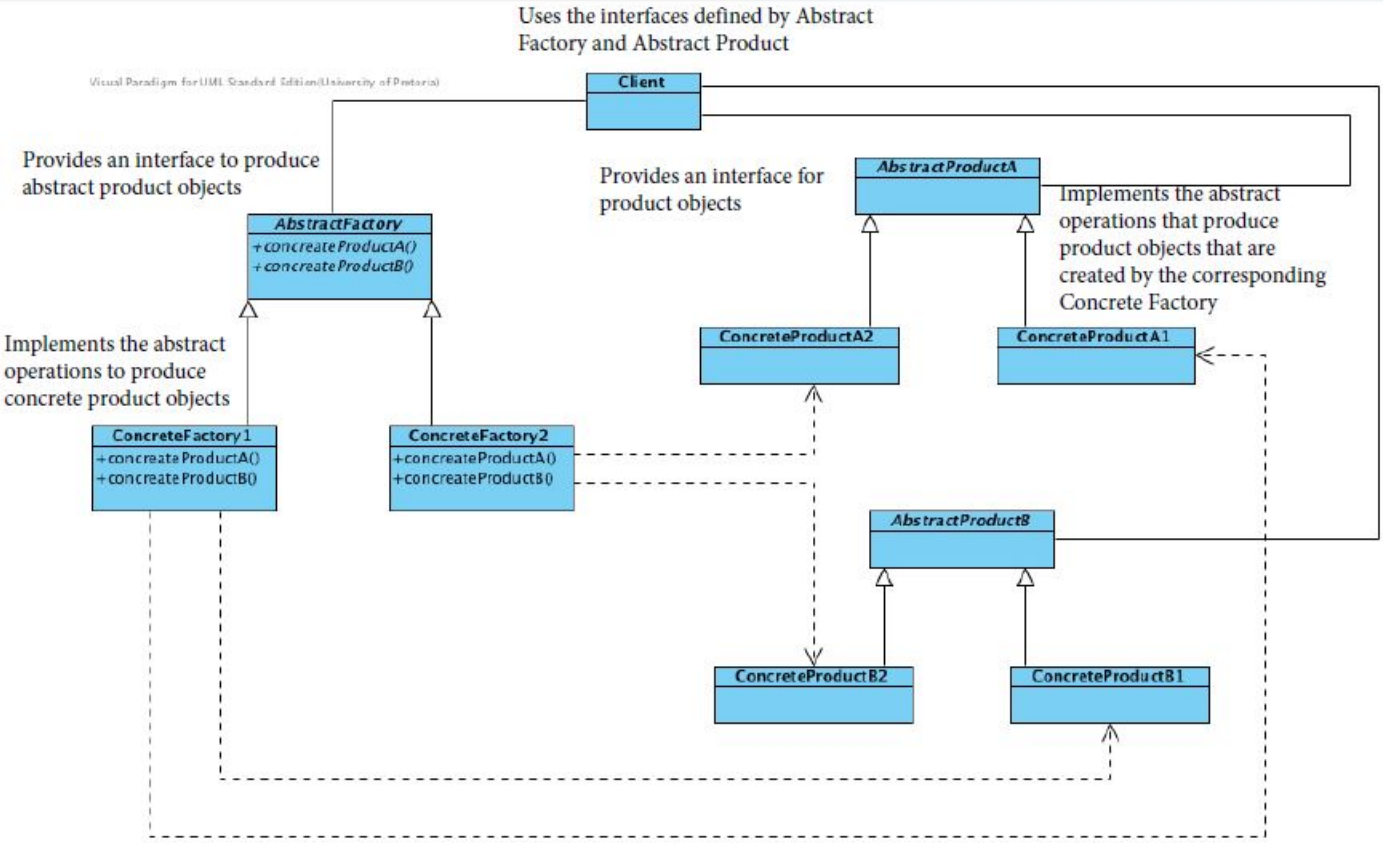
Define an interface for creating an object but let subclasses decide which class to instantiate. Factory method lets a class defer instantiation to subclasses.

PROTOTYPE



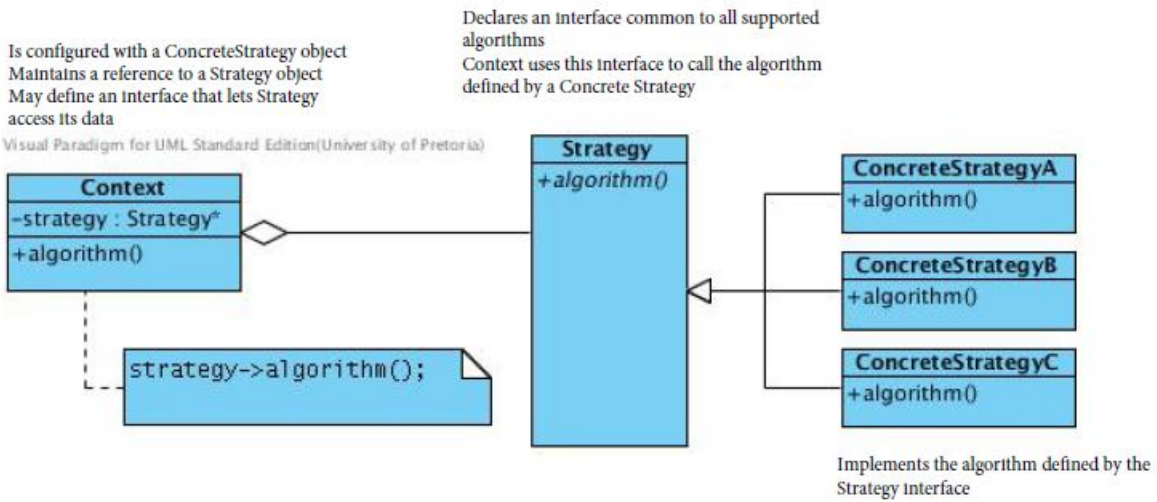
Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype

ABSTRACT FACTORY



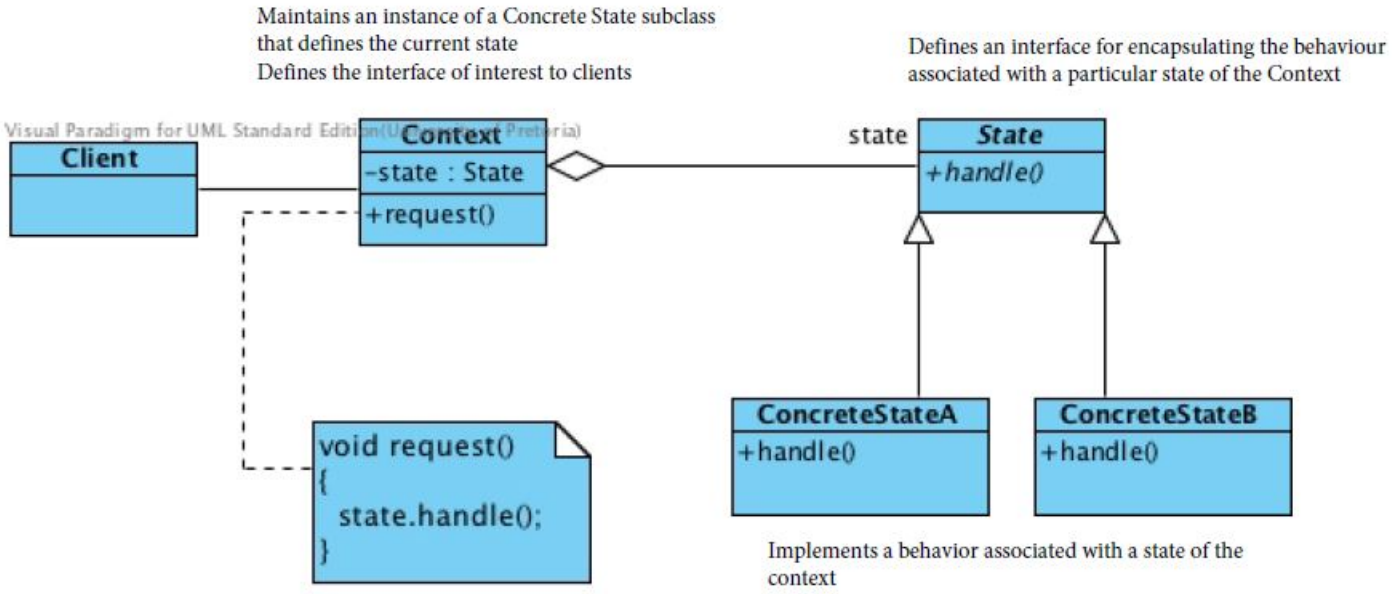
Provide an interface for creating families of related or dependent objects without specifying the concrete classes

STRATEGY



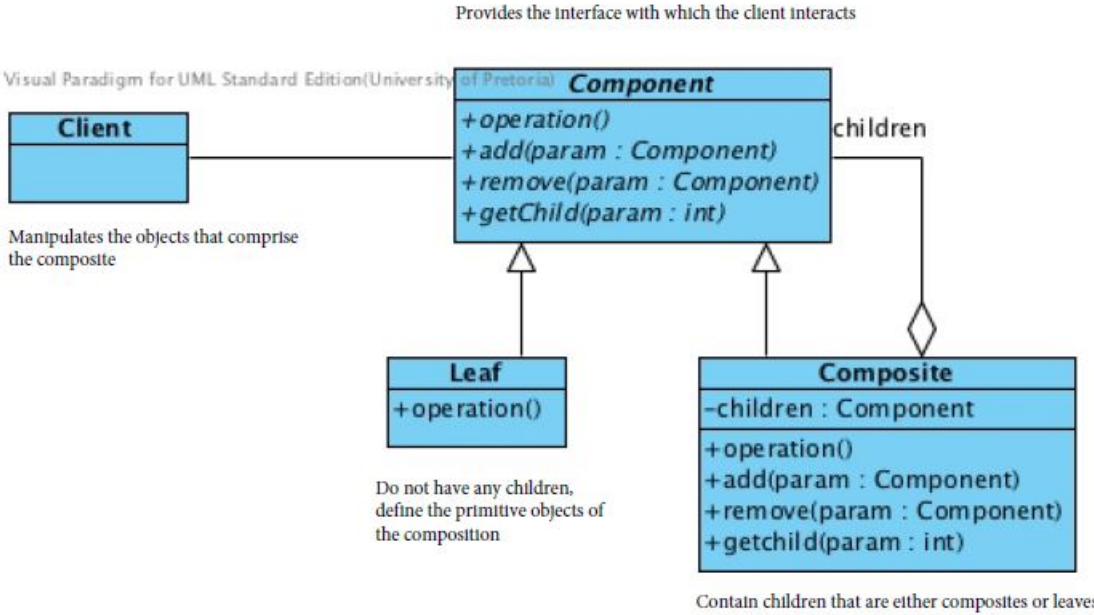
Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it

STATE



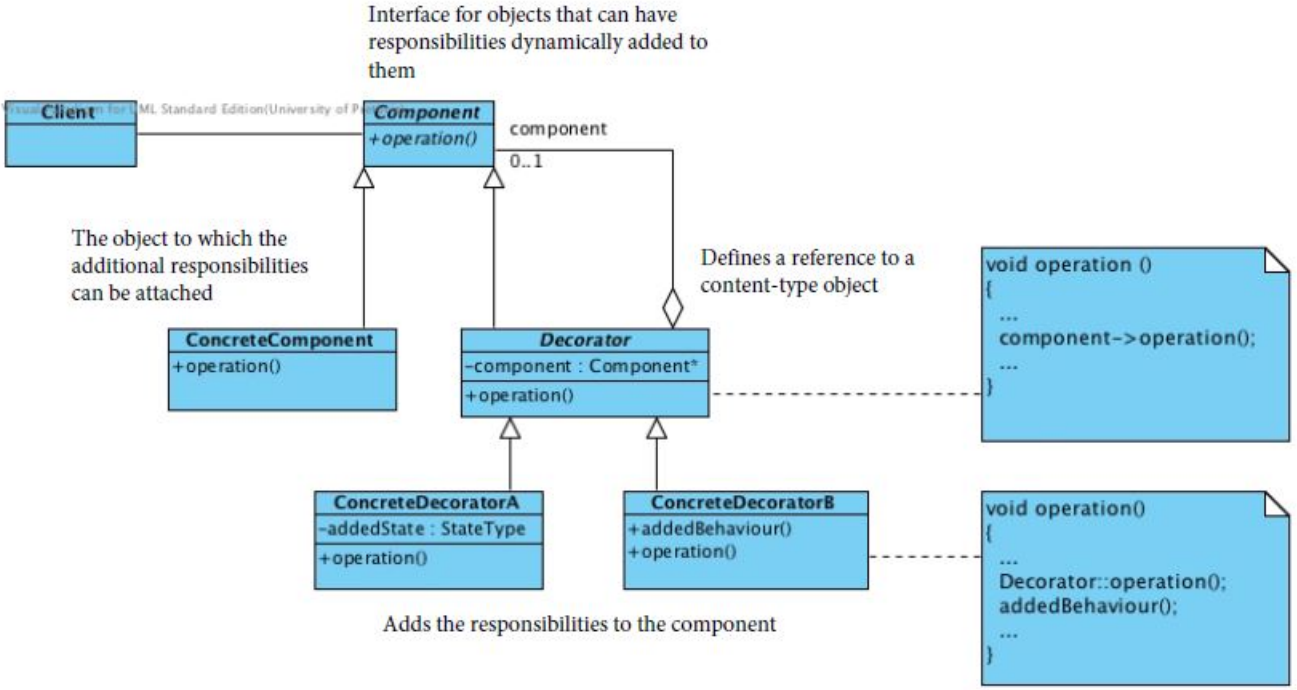
Allows an object to alter its behavior when its internal state changes. The object will appear to change its class.

COMPOSITE



Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly

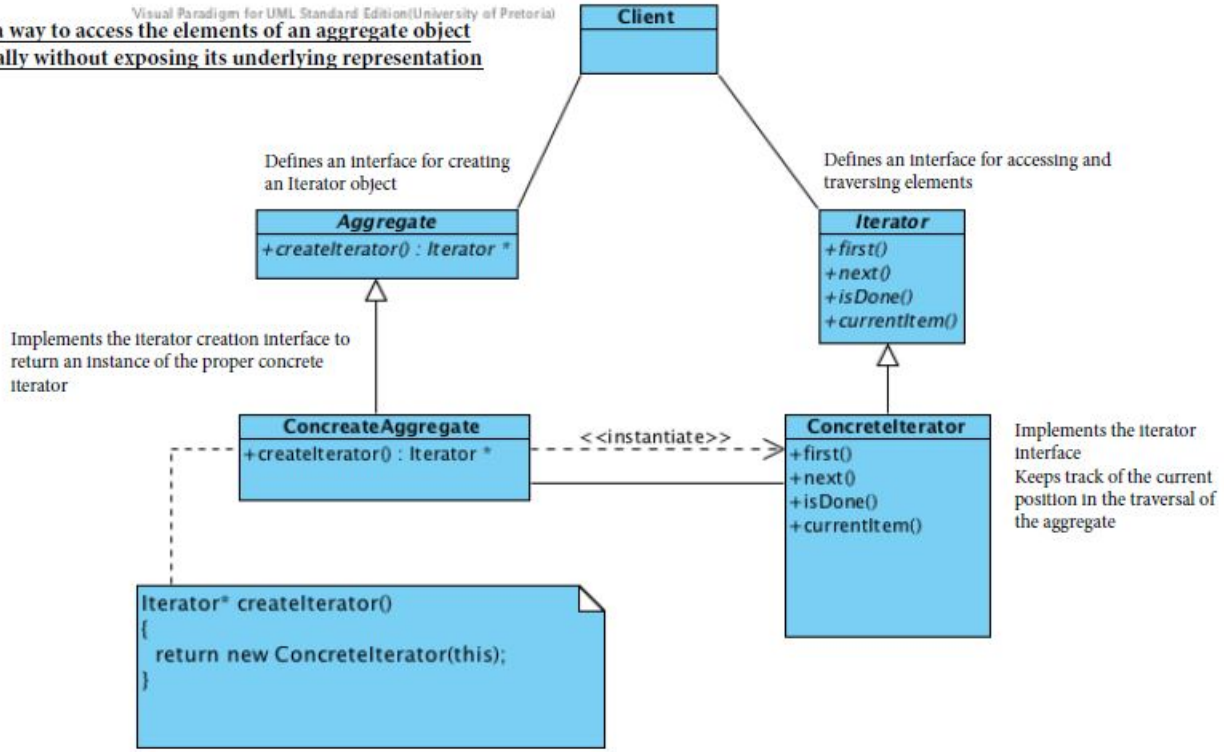
DECORATOR



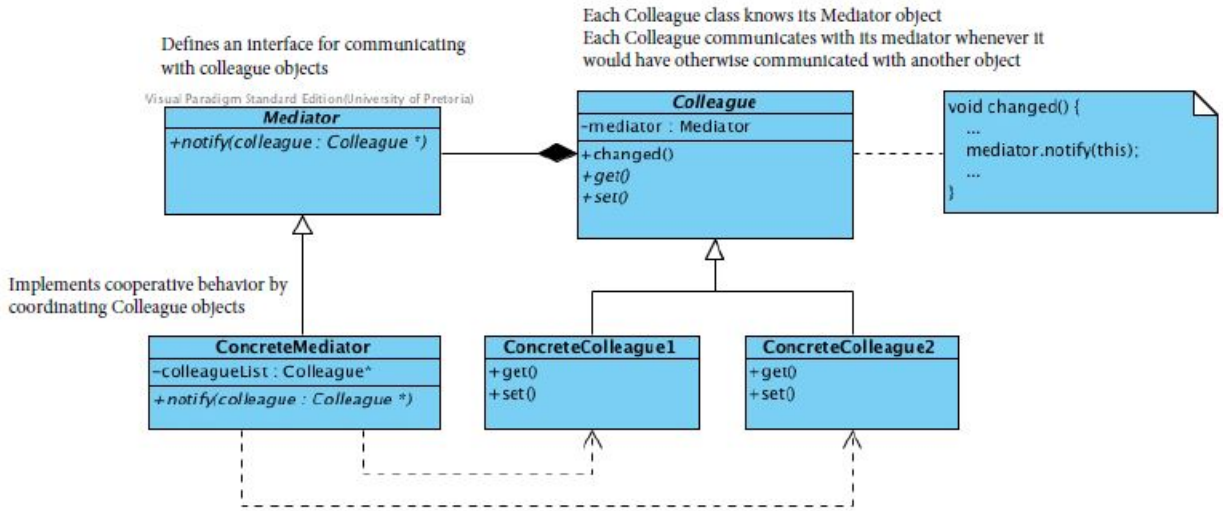
Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality

ITERATOR

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

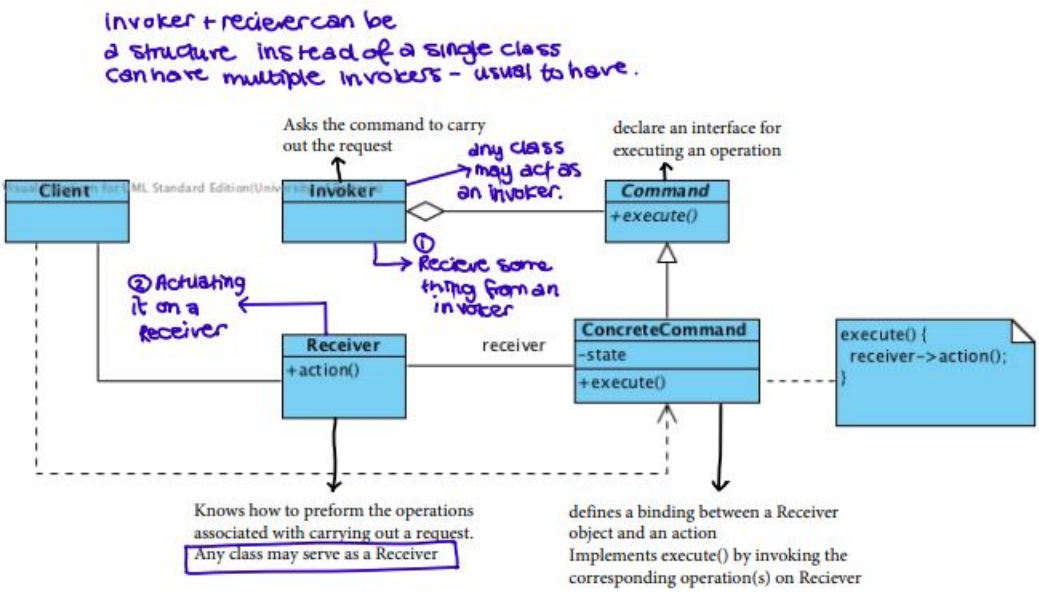


MEDIATOR



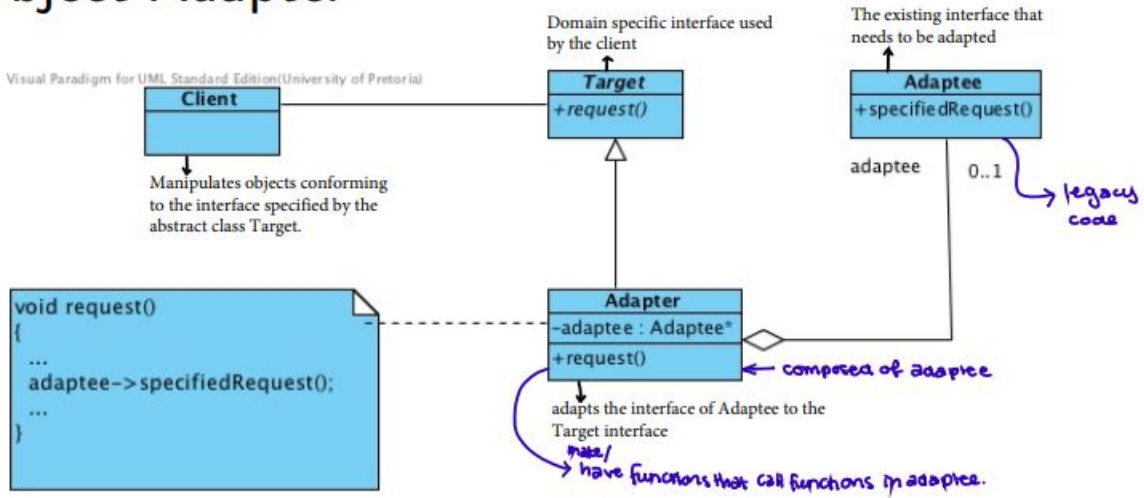
Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly and it lets you vary the interaction independently

COMMAND
Object behavioural



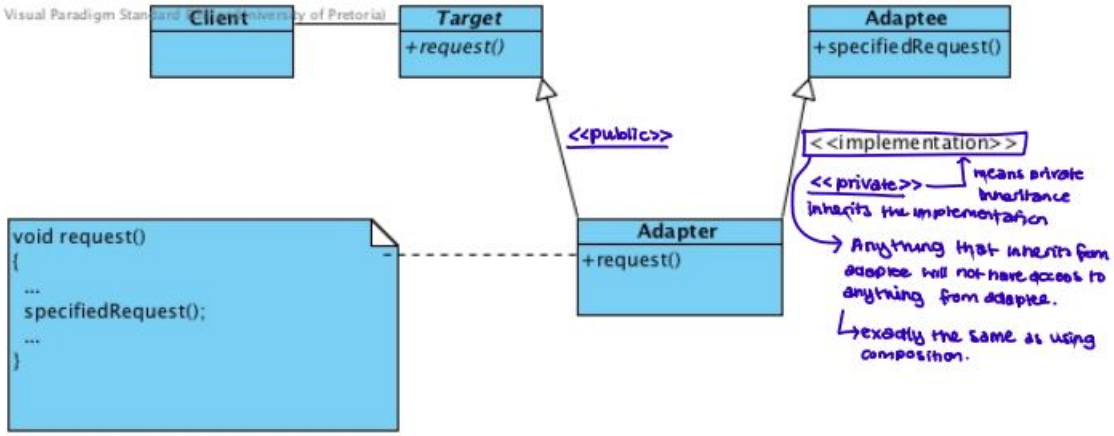
"Encapsulate a request as an object, thereby letting you parameterise clients with different requests, queue or log requests, and support undoable operations

Object Adapter



Convert an interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.

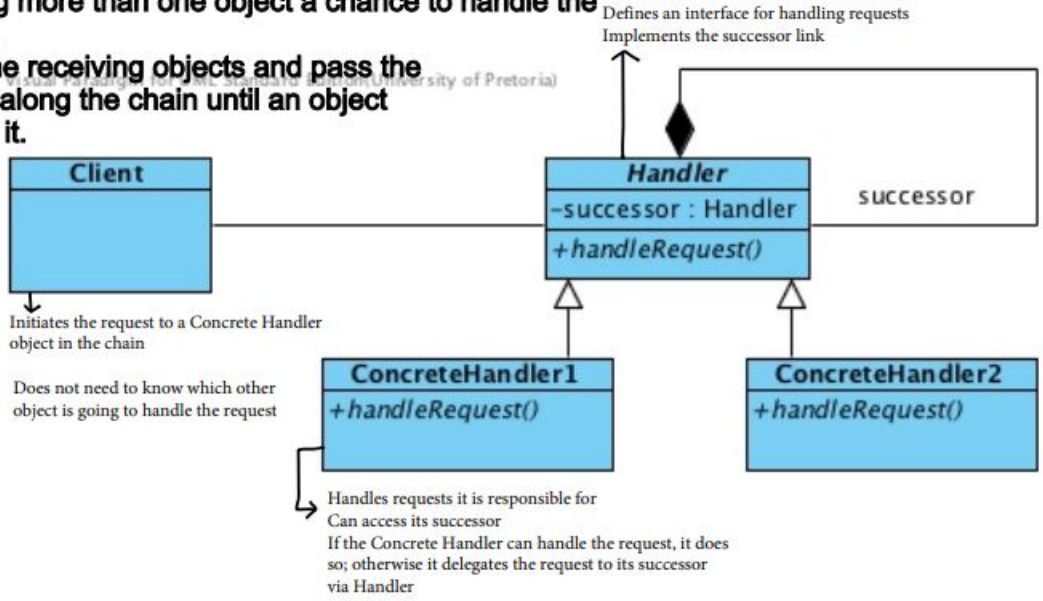
Class Adapter



Convert an interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.

CHAIN OF RESPONSIBILITY

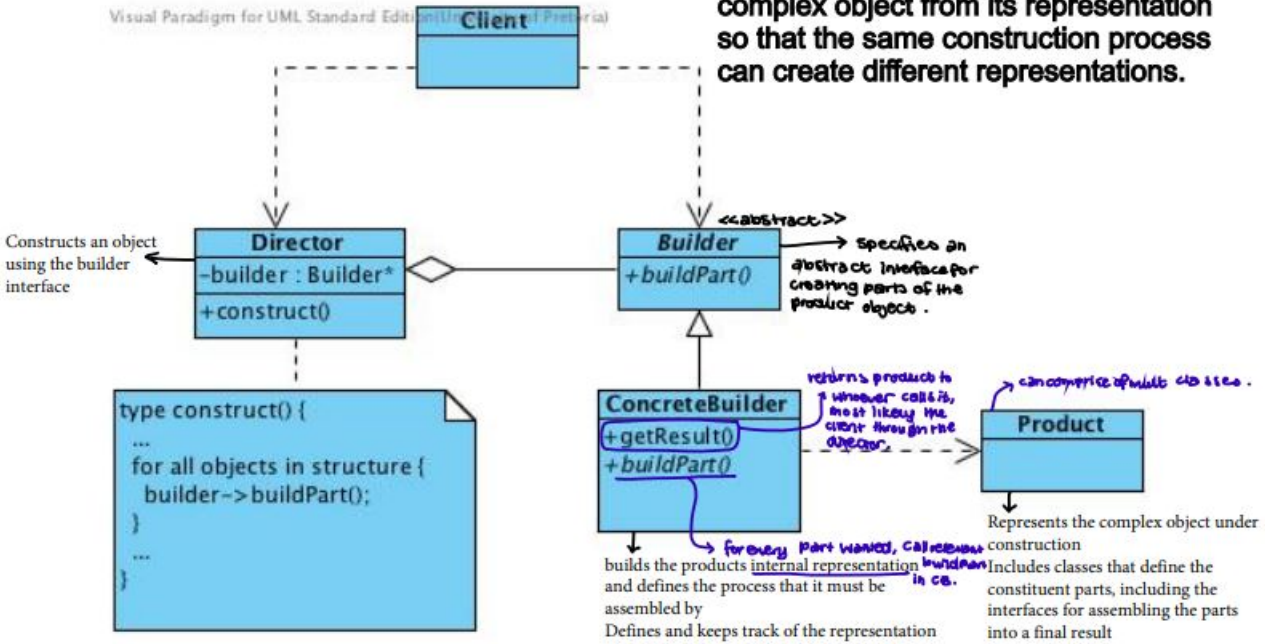
Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request.
Chain the receiving objects and pass the request along the chain until an object handles it.



BUILDER

Creation, Delegation

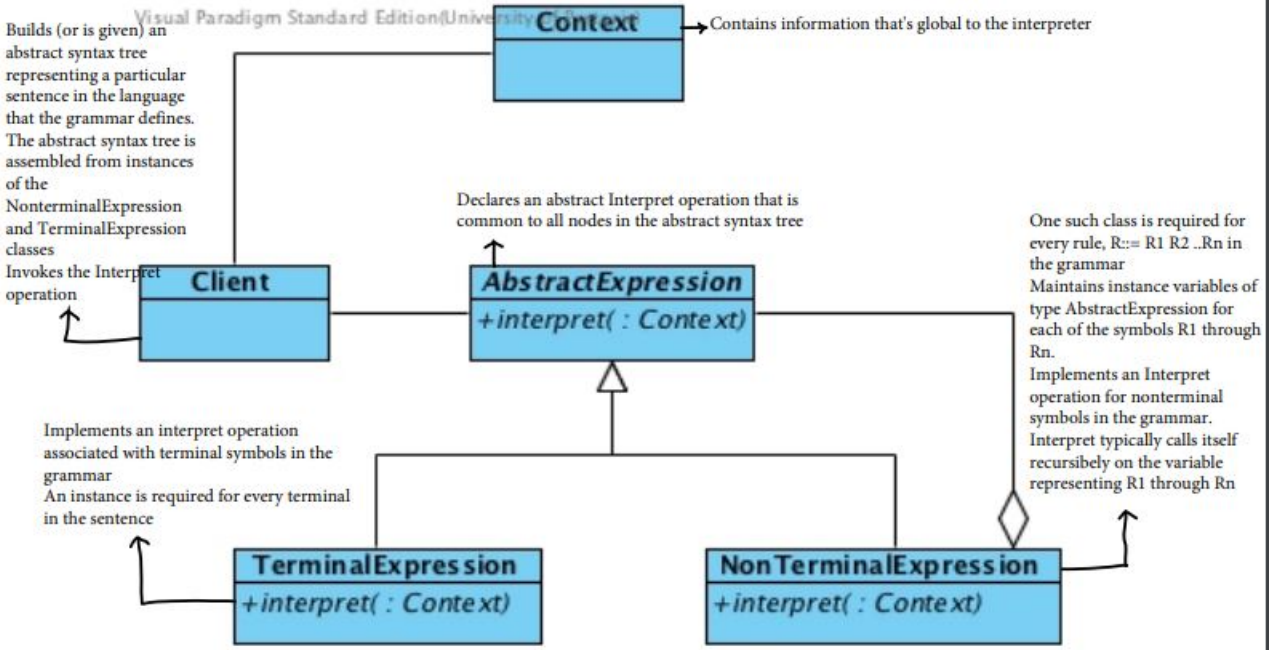
Separate the construction of a complex object from its representation so that the same construction process can create different representations.



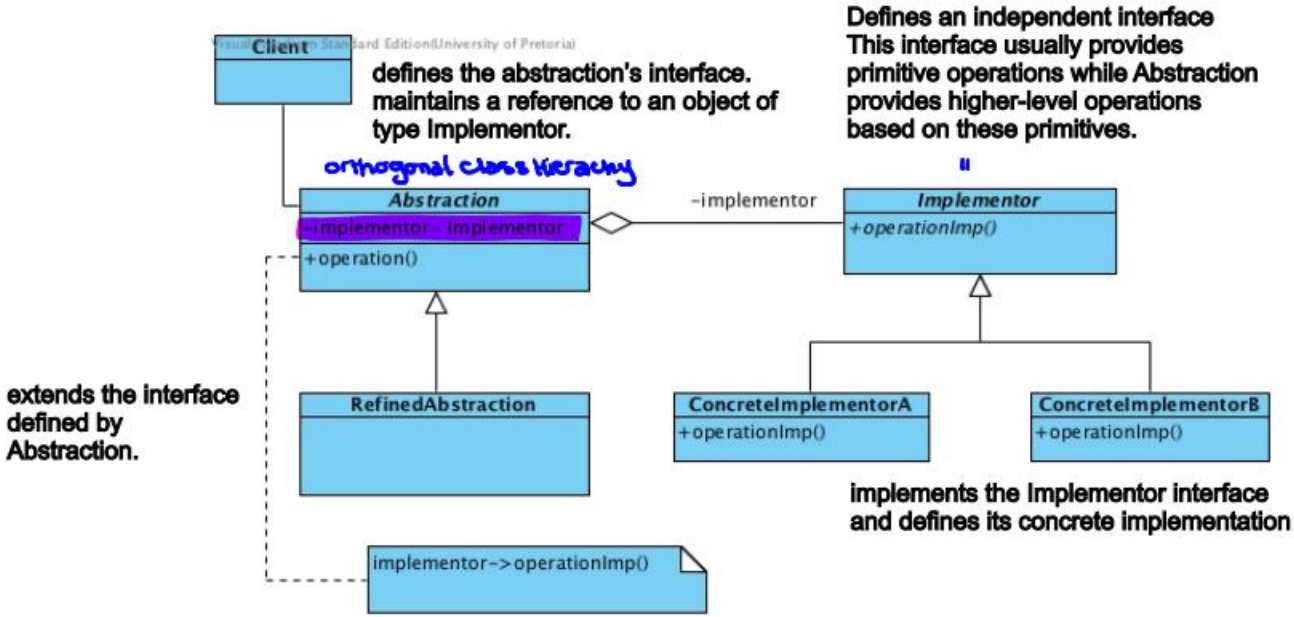
INTERPRETER

Class Behavioural

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



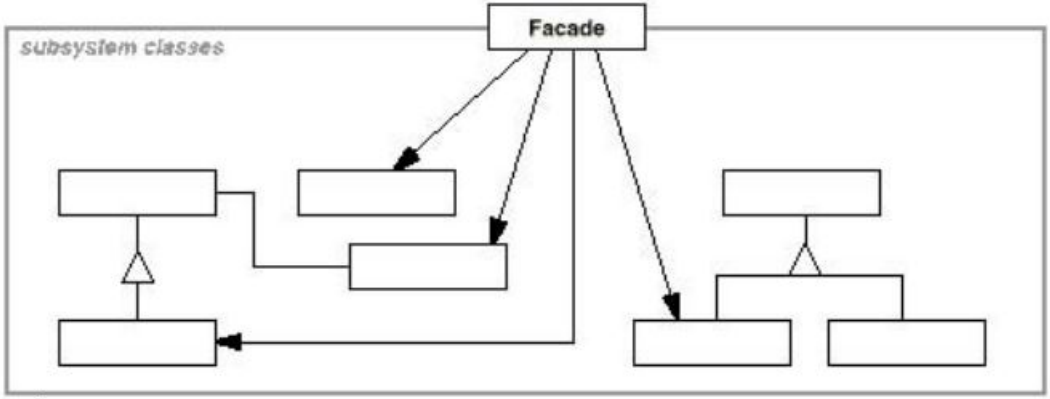
BRIDGE
Object Structural



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

FACADE
Structural - Delegation

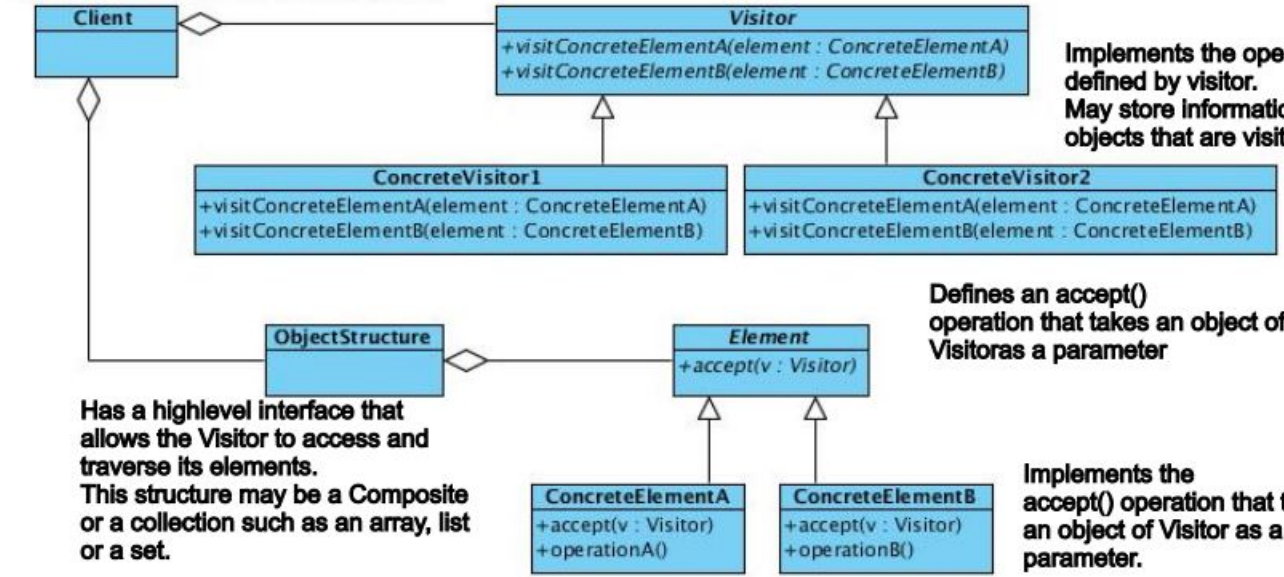
Knows which subsystem classes are responsible for a request.
Delegates client requests to appropriate subsystem objects.



Subsystem classes:
Implements subsystem functionality Handle work assigned by the Façade object.
Have no knowledge of the façade and perform operations independent of the façade.

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

each class of ConcreteElement has a visit() operation declared for it.
The operation's signature identifies the class that sends the visit() request to the visitor
The particular class is then accessed through the interface defined for it



Implements the operations defined by visitor.
May store information about objects that are visited.

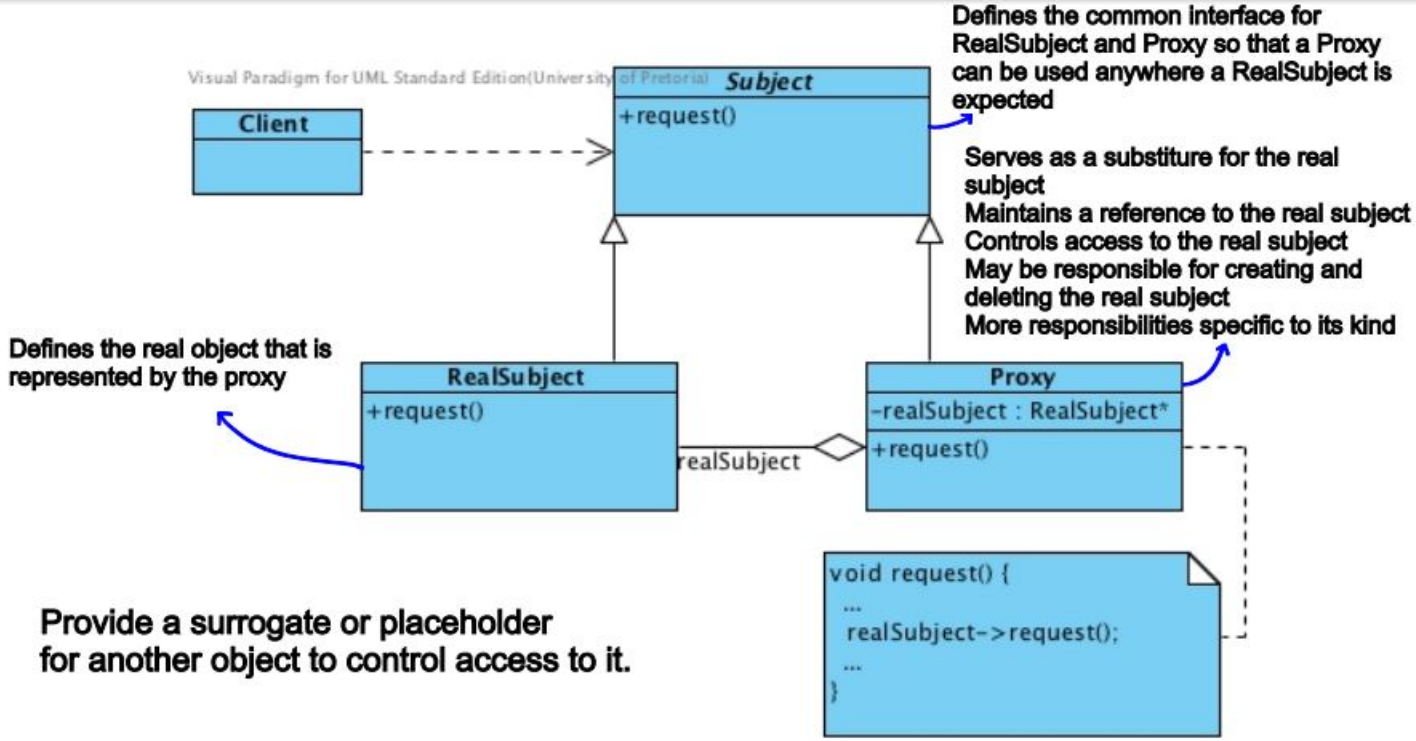
Defines an accept() operation that takes an object of Visitoras a parameter

Has a highlevel interface that allows the Visitor to access and traverse its elements.
This structure may be a Composite or a collection such as an array, list or a set.

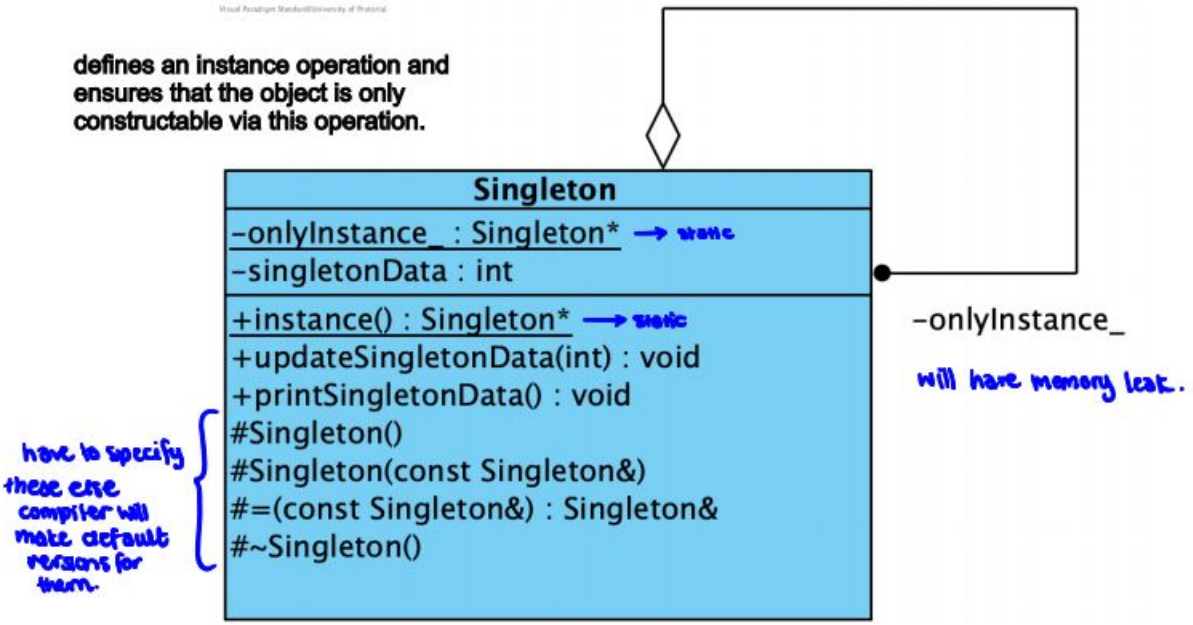
Implements the accept() operation that takes an object of Visitor as a parameter.

Represent an operation to be performed on the elements of an object structure.
Visitor lets you define a new operation without changing the classes of the elements on which it operates

PROXY
Object structural



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

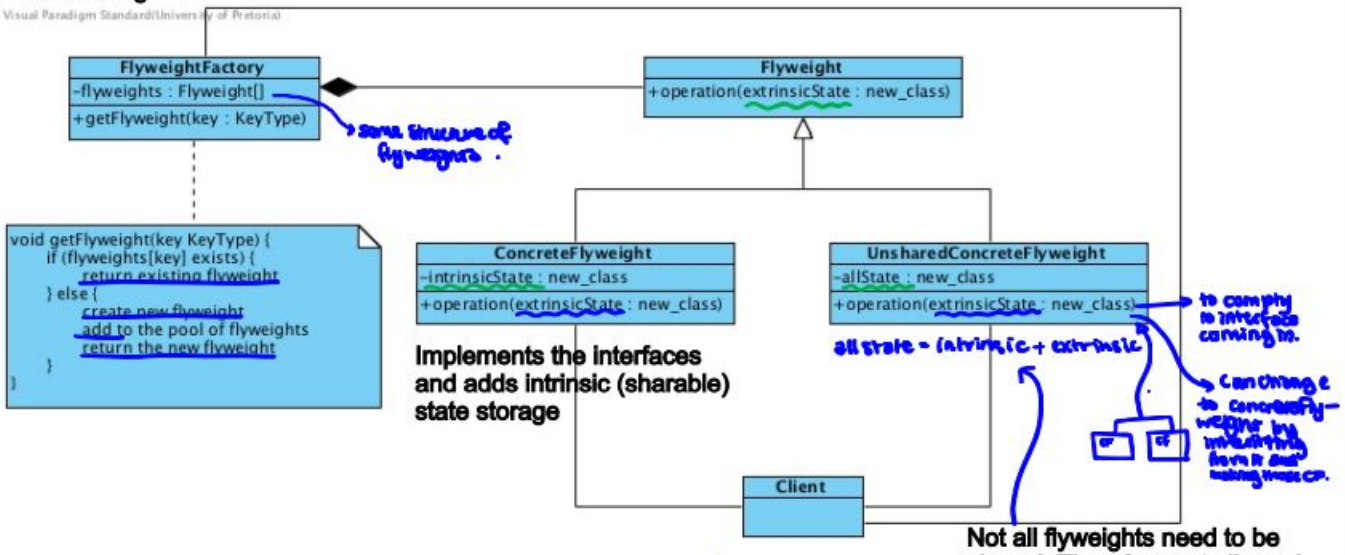


Ensure a class only has one instance, and provide a global point of access to it.

FLYWEIGHT
Object Structural

Creates an instance of a flyweight if does not exist or supplies the existing one.

Defines the interface through which flyweights are instantiated



Implements the interfaces and adds intrinsic (sharable) state storage

Not all flyweights need to be shared. Therefore not all need to store intrinsic state and add extrinsic state to represent its entire state. UnsharedConcreteFlyweight may have ConcreteFlyweight children

User sharing to support large numbers of fine-grained objects efficiently.