Design Patterns

Thierry Sans

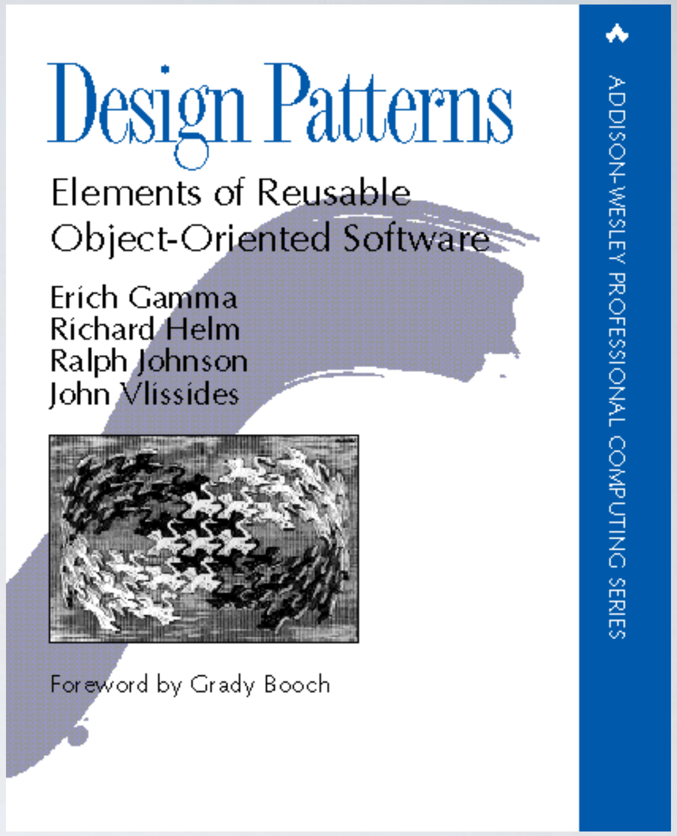
with slides from Anya Tafliovich

Definition

A design pattern is a general description of the solution to a well-established problem using an arrangement of classes and objects

- ➡ Describe the shape of code rather than the details
- ✓ They are not specific to any one programming language
- ✓ Implementation differs between programming languages

Original Proposal


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Original Gang of Four book in 1995 described 23 design patterns at first

➡ More design patterns have been added since

Other ressources online

- http://www.oodesign.com/
- https://sourcemaking.com/design_patterns
- https://en.wikipedia.org/wiki/Software_design_pattern

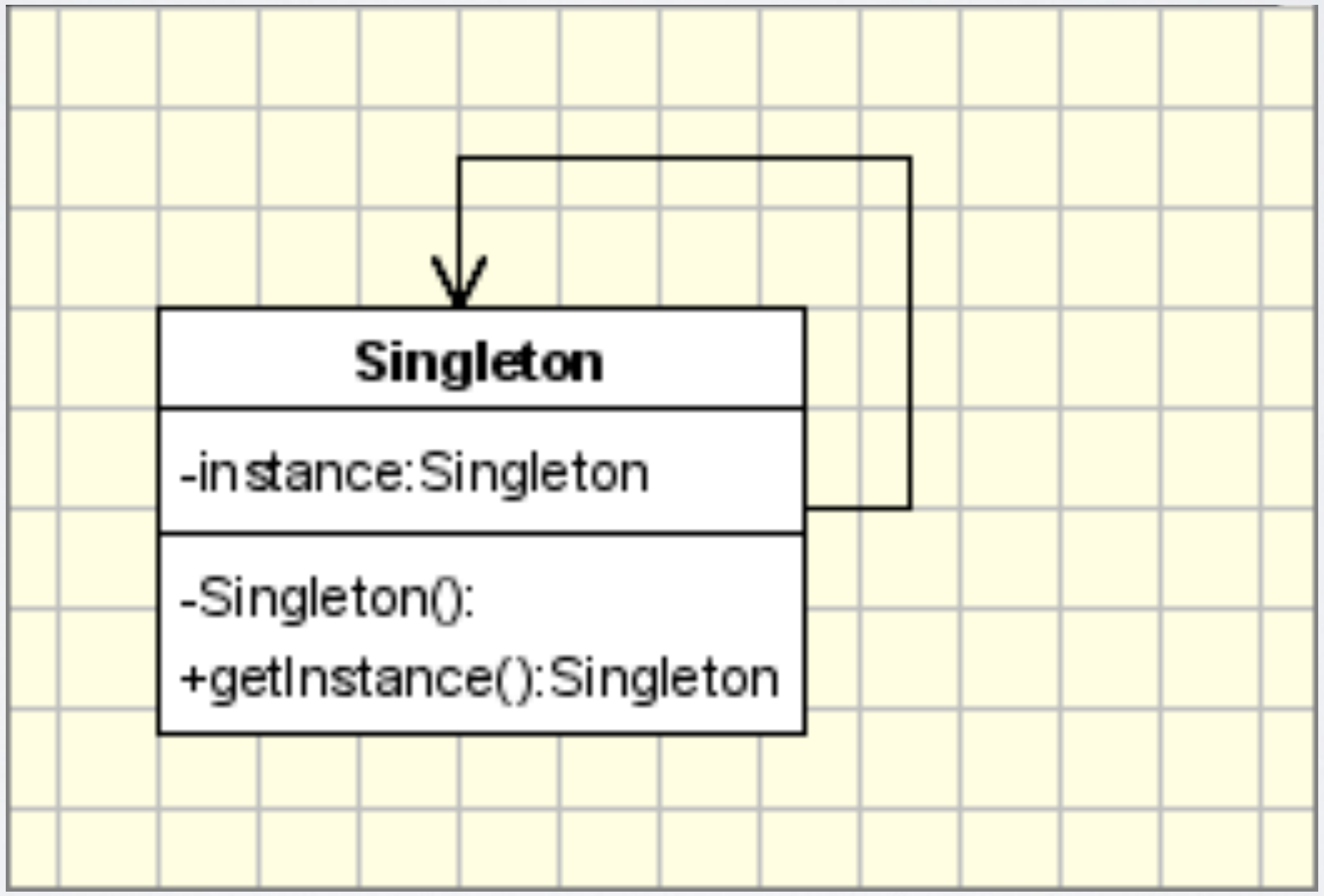
Pattern Families

- Creational Patterns
- Structural Patterns
- Behavioral Patterns
- Concurrency Patterns (not in the original Gang of Four book)

Creational Patterns

Singleton

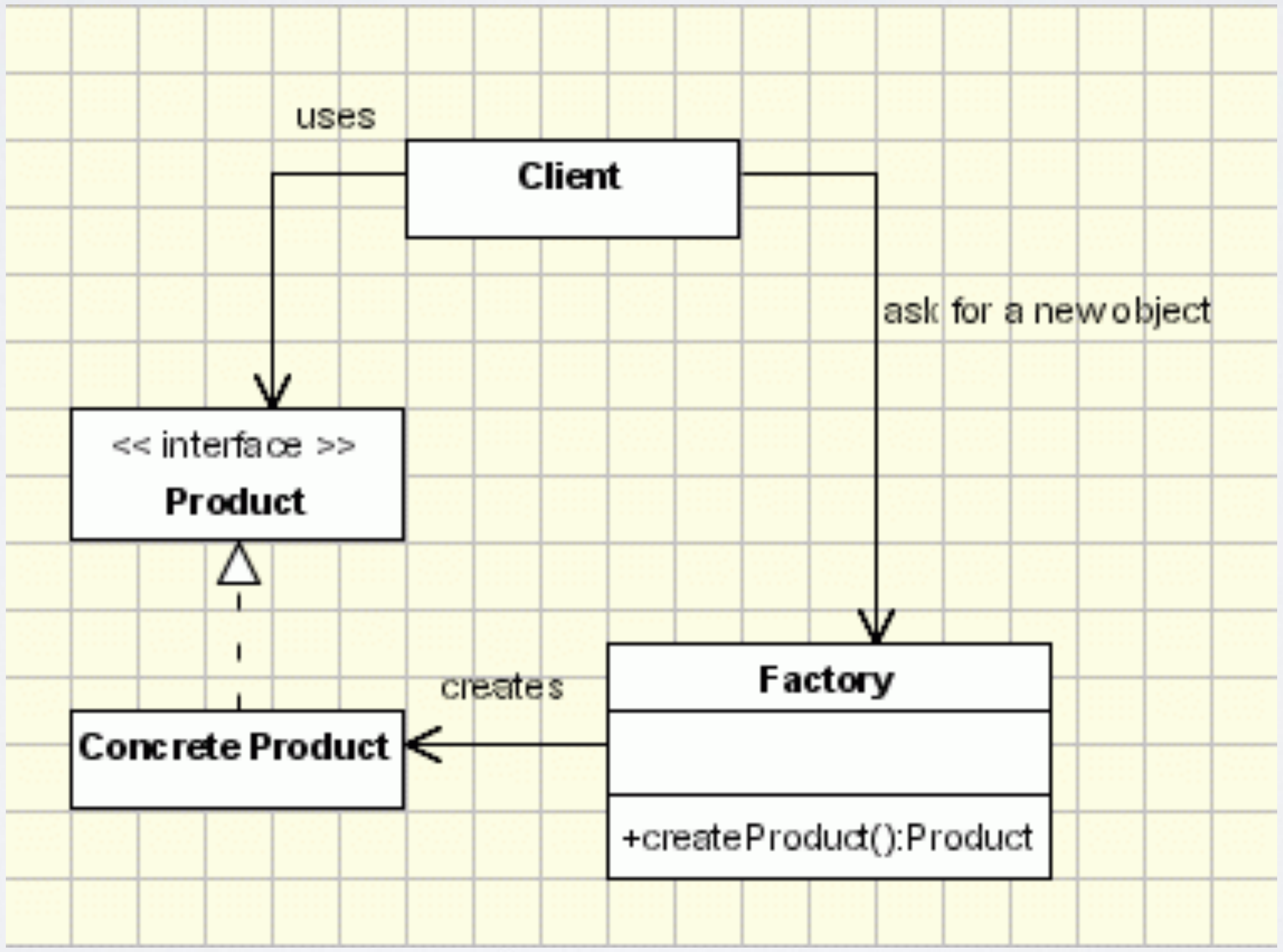
Ensure that only one instance of a class is created and provide a global access point to the object



source: *http://www.oodesign.com/*

Factory

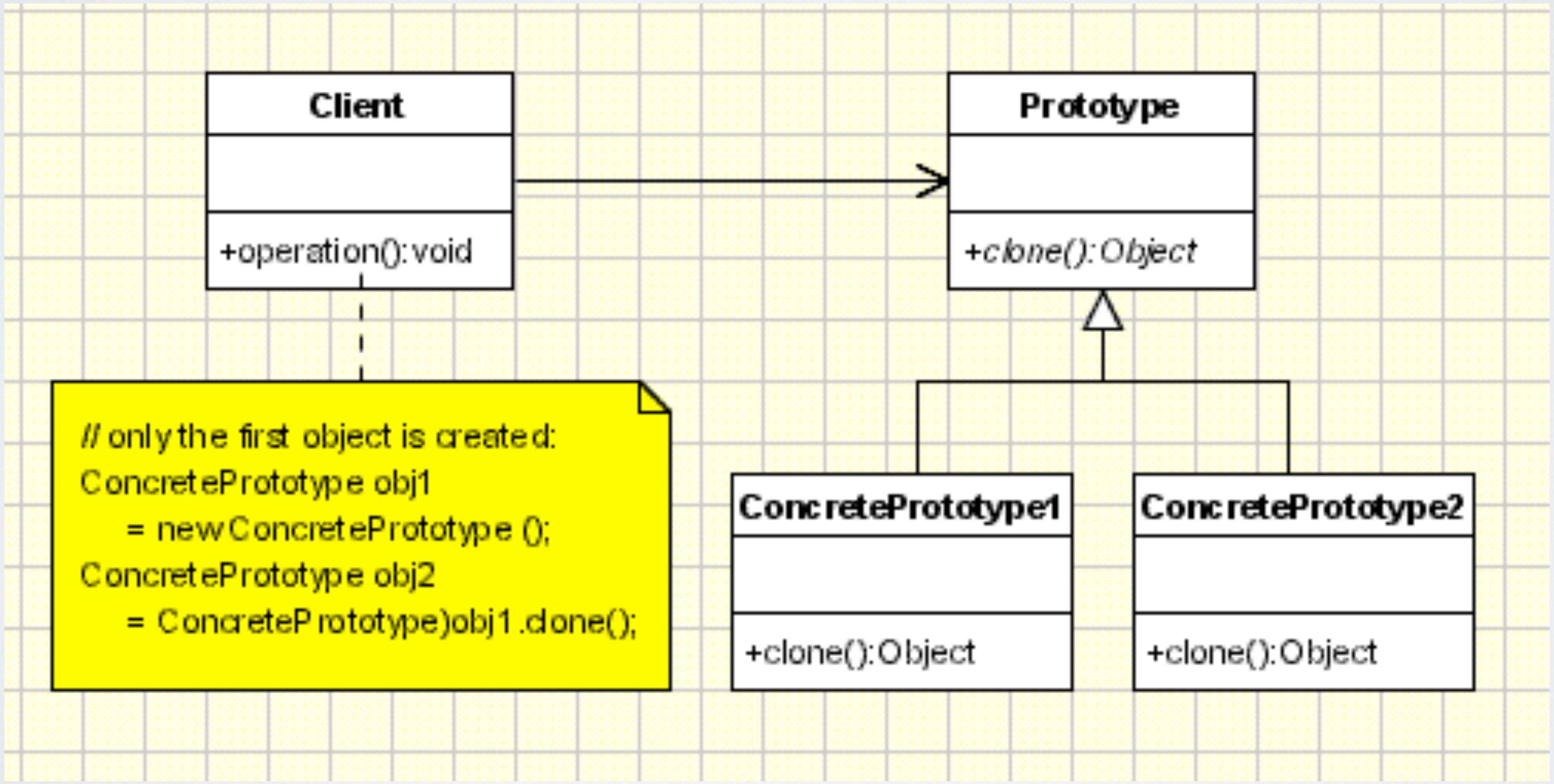
Creates objects without exposing the instantiation logic to the client and refers to the newly created object through a common interface



source: *http://www.oodesign.com/*

Prototype

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



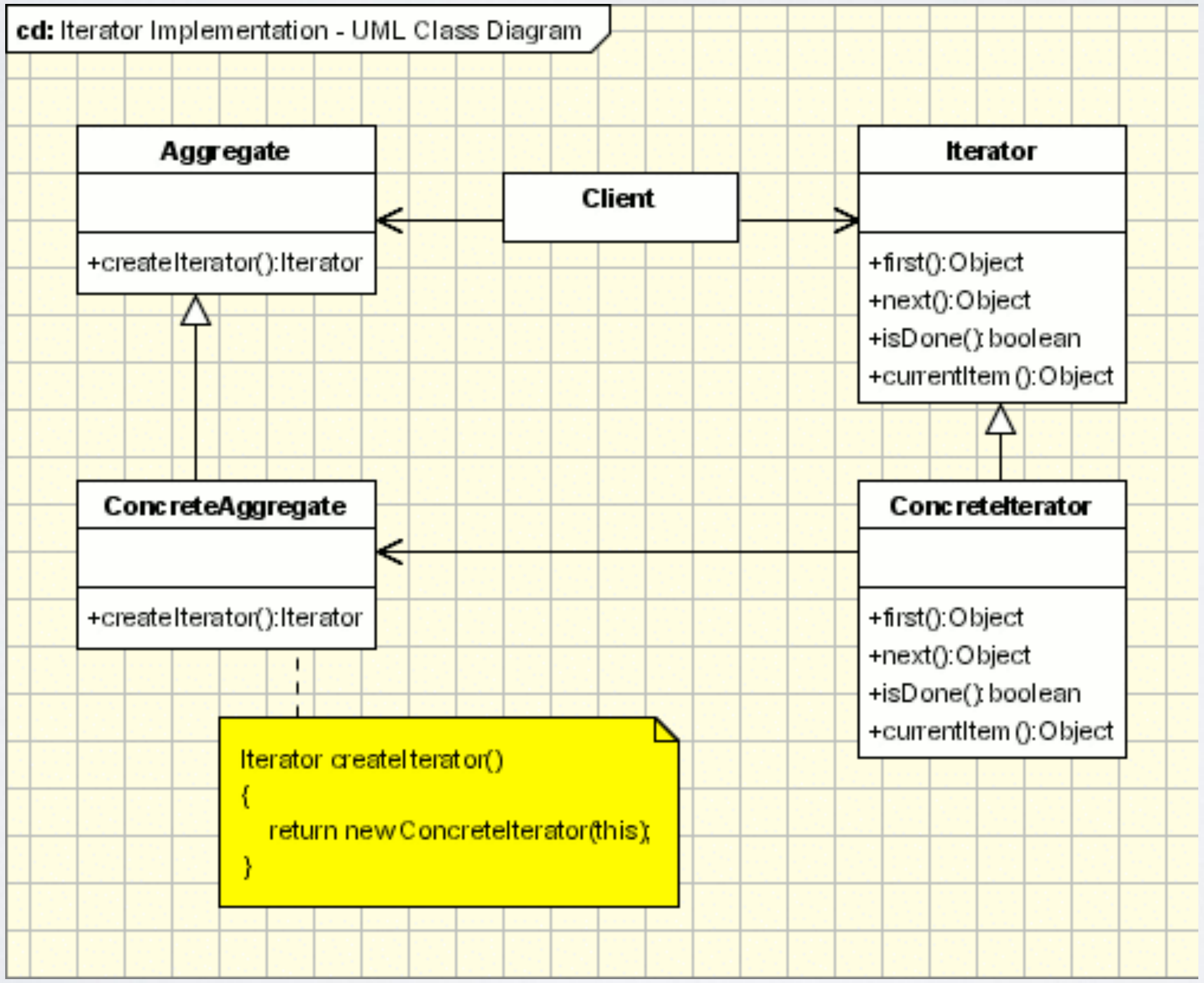
```
// only the first object is created:
ConcretePrototype obj1
    = new ConcretePrototype ();
ConcretePrototype obj2
    = ConcretePrototype)obj1.clone();
```

source: *http://www.oodesign.com/*

Behavioral Patterns

Iterator

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

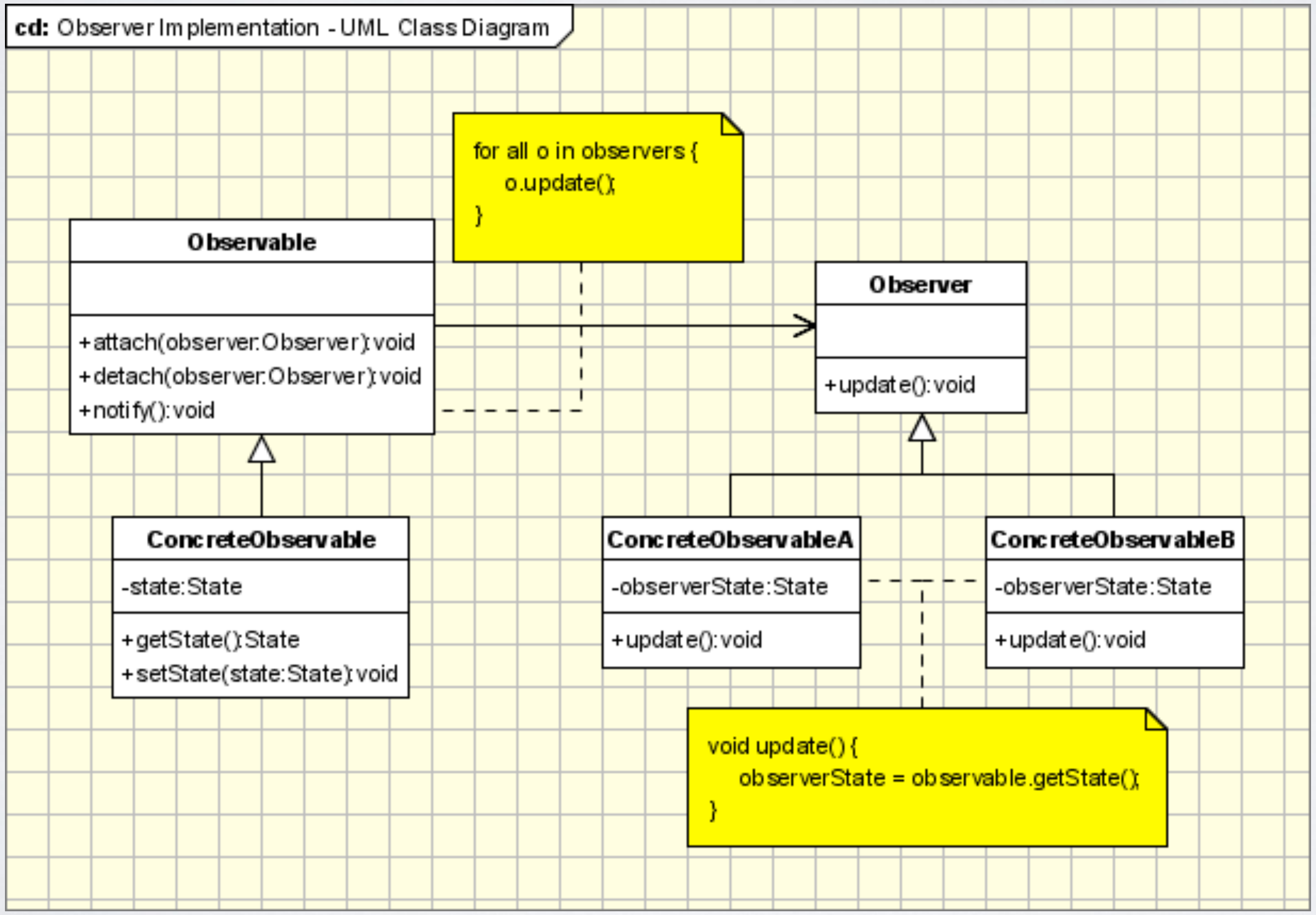


```
Iterator createIterator()
{
    return new ConcreteIterator(this);
}
```

source: *http://www.oodesign.com/*

Observer

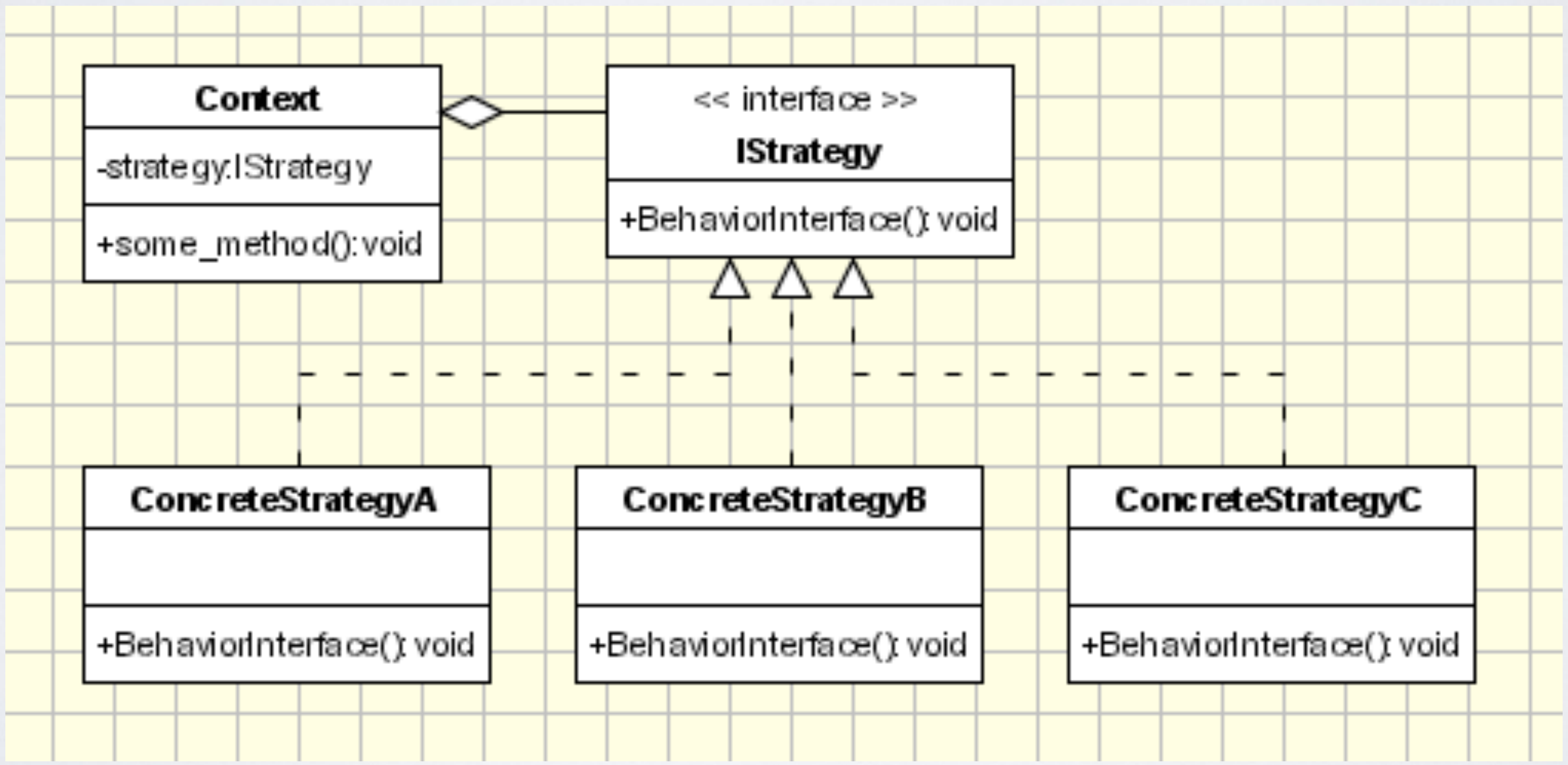
Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically



source: *http://www.oodesign.com/*

Strategy

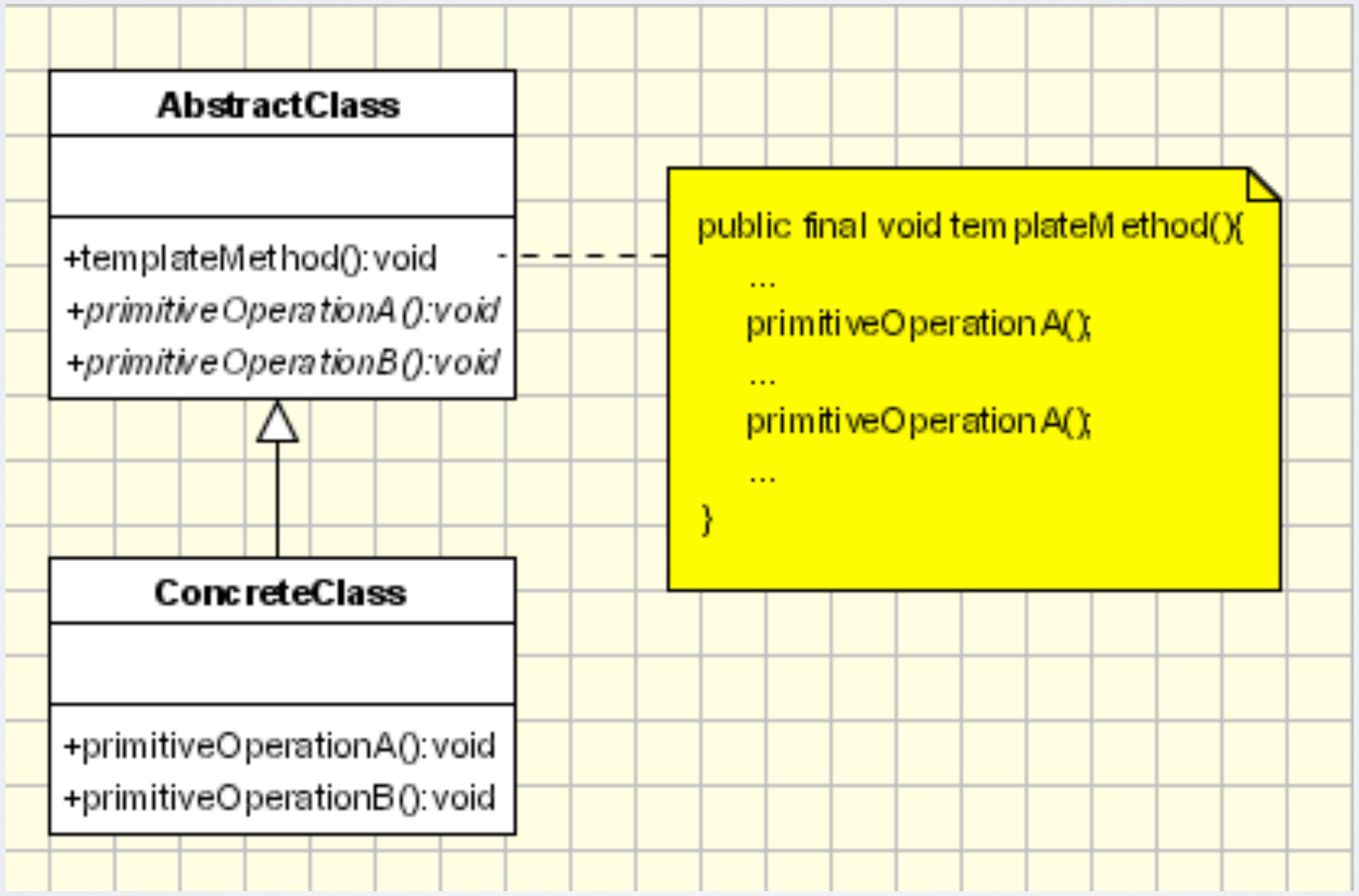
Define a family of algorithms, encapsulate each one, and make them interchangeable



source: *http://www.oodesign.com/*

Template Method

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


AbstractClass
+templateMethod():void
+primitiveOperationA():void
+primitiveOperationB():void
ConcreteClass
+primitiveOperationA():void
+primitiveOperationB():void

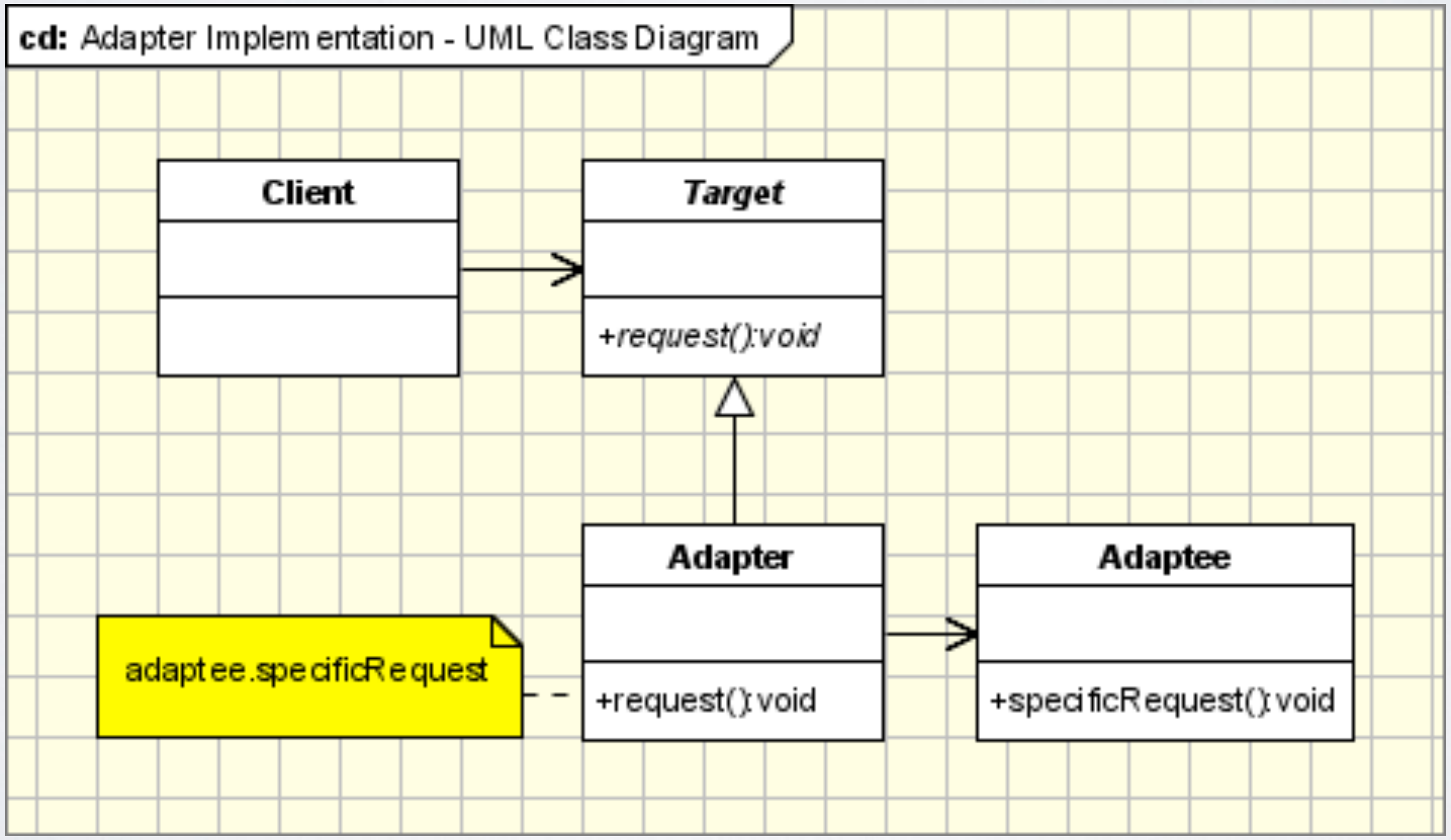
```
public final void templateMethod(){
    ...
    primitiveOperationA();
    ...
    primitiveOperationA();
    ...
}
```

source: *http://www.oodesign.com/*

Structural Patterns

Adapter

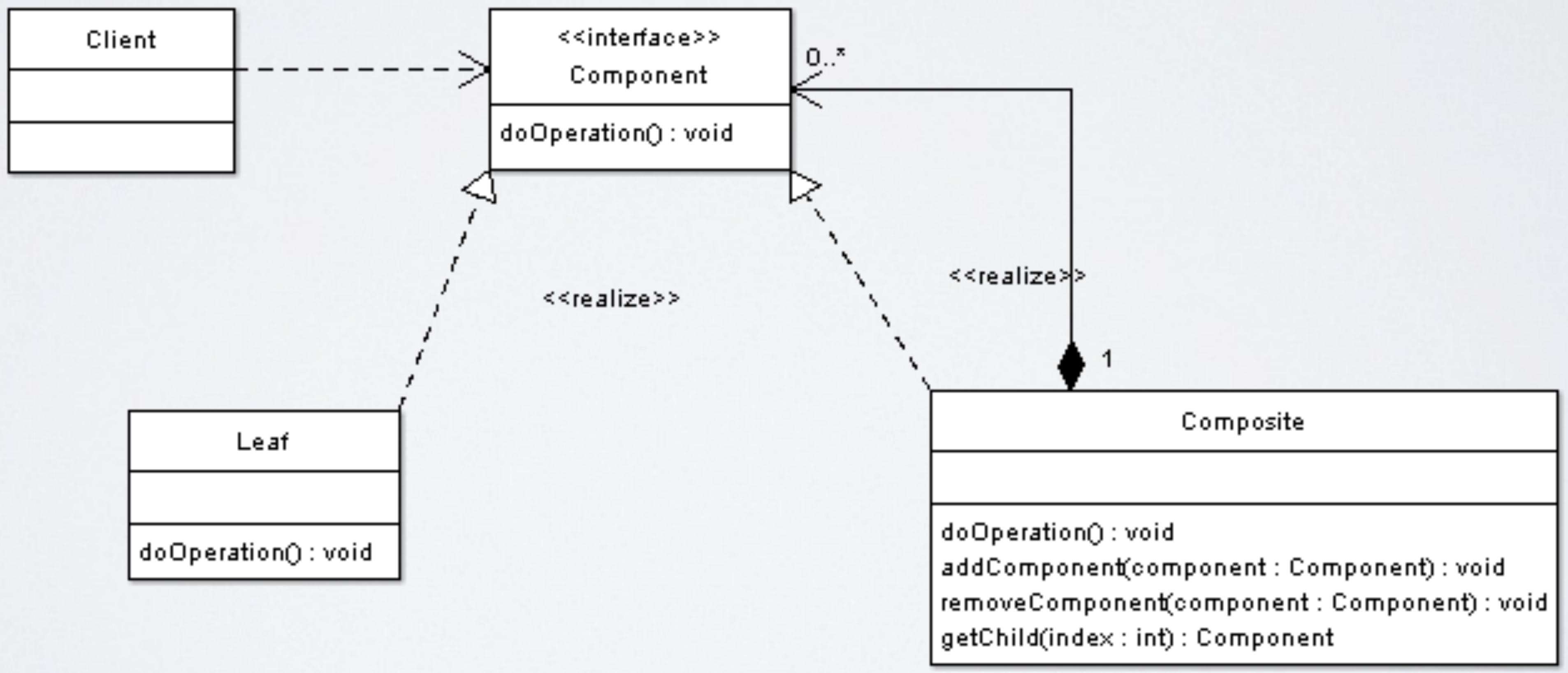
Convert the interface of a class into another interface clients expect



source: *http://www.oodesign.com/*

Composite

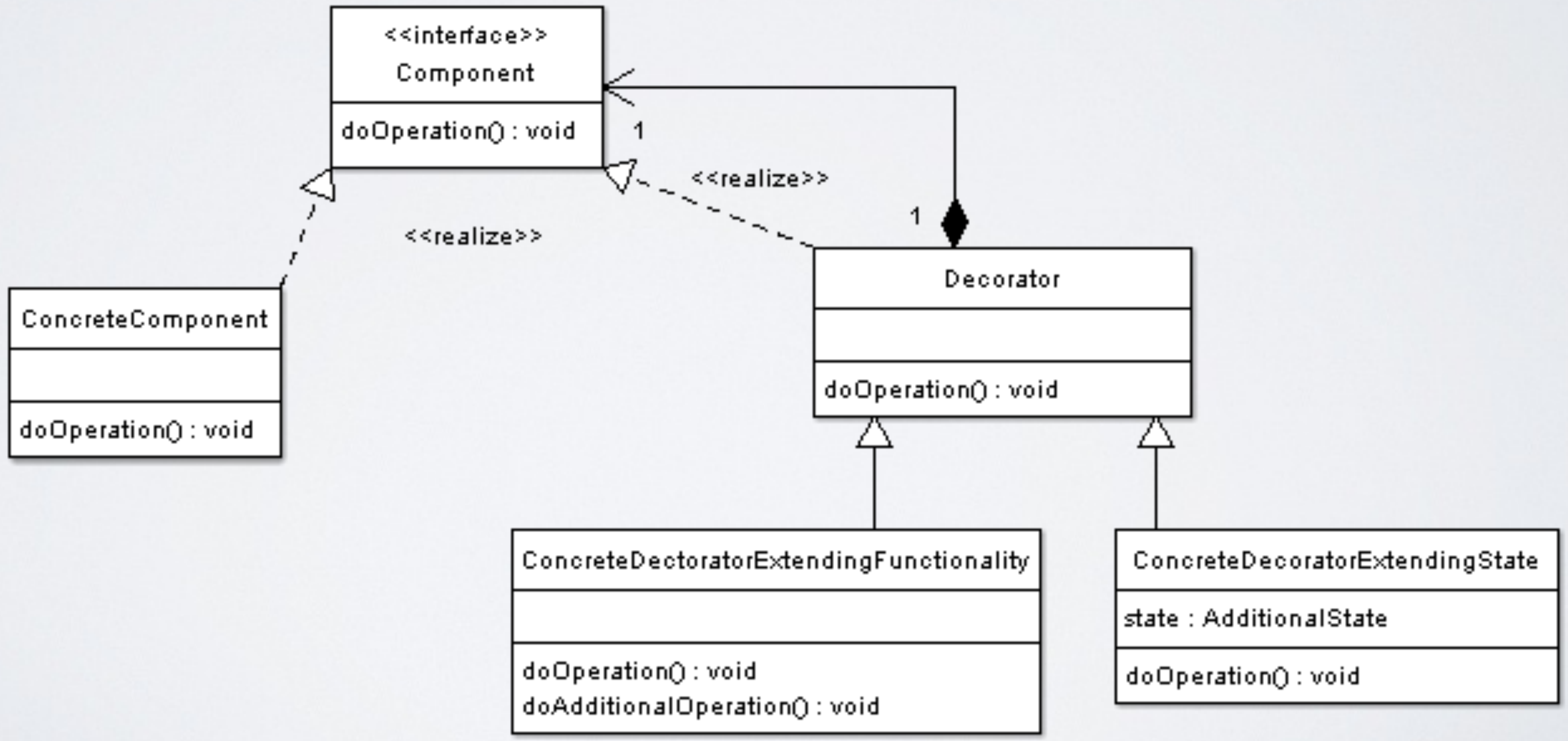
Compose objects into tree structures to represent part-whole hierarchies



source: *http://www.oodesign.com/*

Decorator

Add additional responsibilities dynamically to an object



source: *http://www.oodesign.com/*