Design Defects and Restructuring

LECTURE 10

SAT, NOV 14, 2020

Structural Patterns

Adapter

Bridge

Composite

Decorator

Façade

Flyweight

Proxy

Decorator

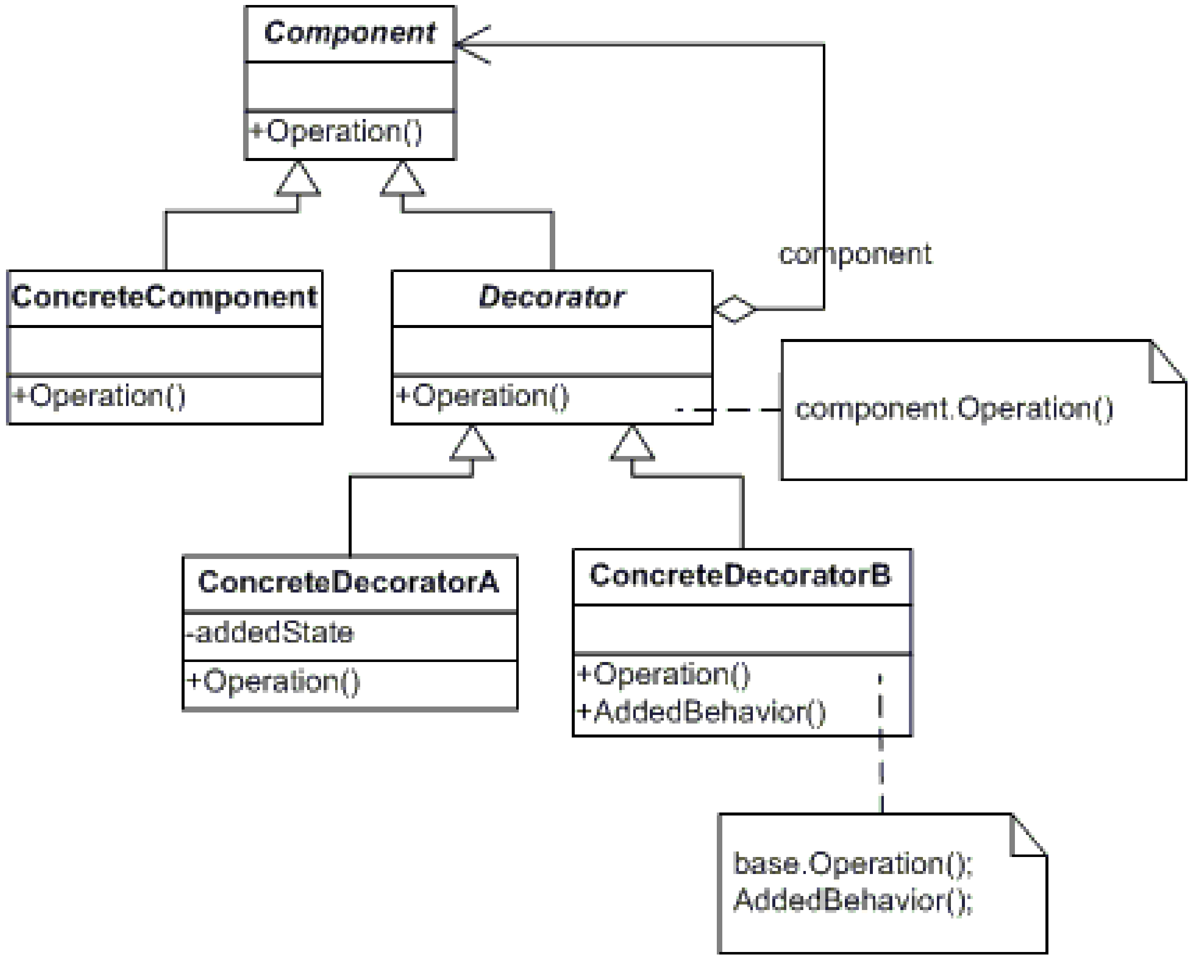
Intent

- Attach additional responsibilities to an object dynamically
- Decorators provide a flexible alternative to sub-classing for extending functionality

Applicability

- To add responsibilities to individual objects dynamically and transparently without affecting other objects
- For responsibilities that can be withdrawn
- When extension by sub-classing is impractical
  - Sometimes a large number of independent extensions are possible and would produce an explosion of subclasses to support every combination
  - Or a class definition may be hidden or otherwise unavailable for sub-classing

Decorator



Component
+Operation()
component
ConcreteComponent
+Operation()
Decorator
+Operation()
component.Operation()
ConcreteDecoratorA
-addedState
+Operation()
ConcreteDecoratorB
+Operation()
+AddedBehavior()
base.Operation();
AddedBehavior();

Façade

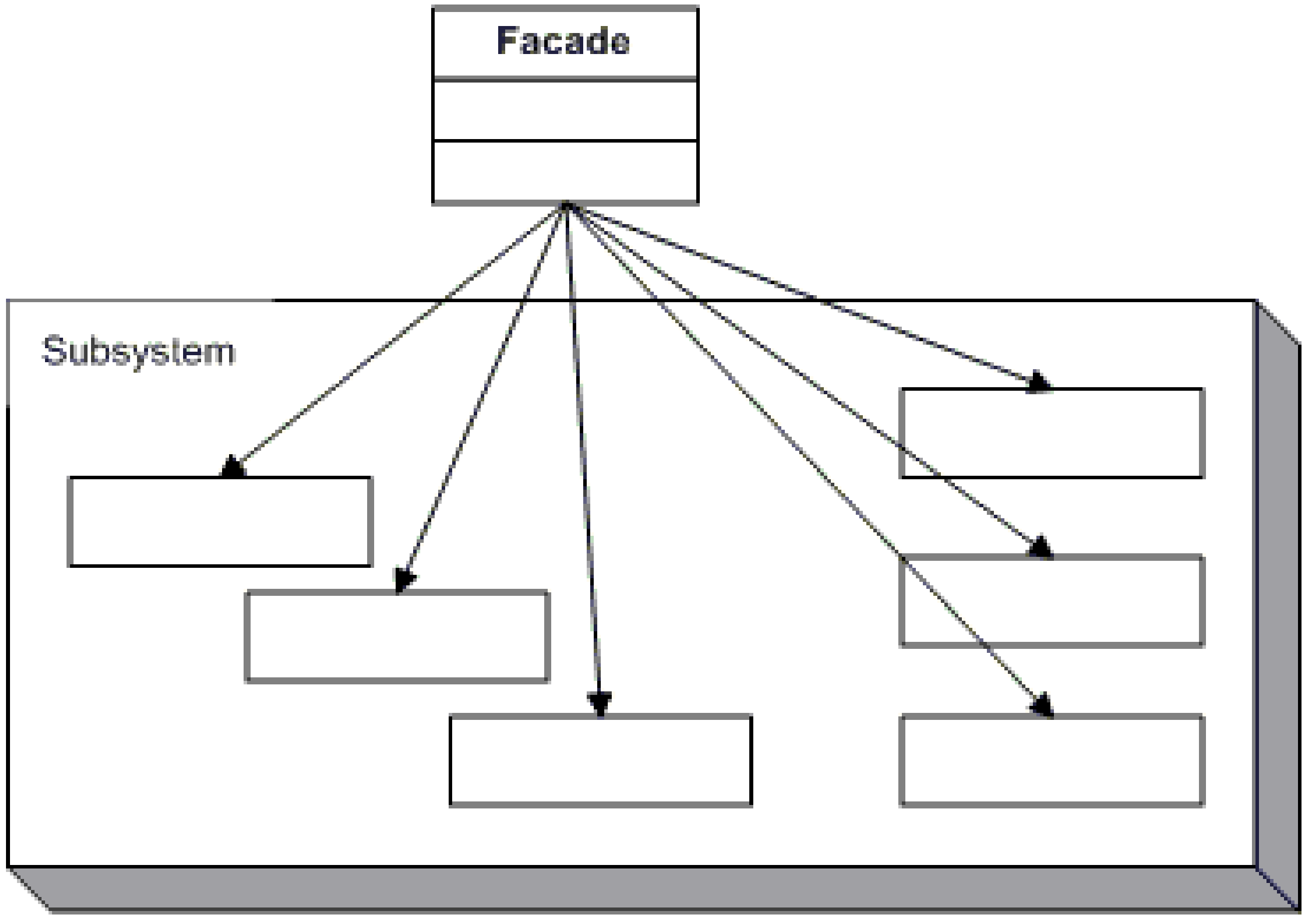
Intent

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

Applicability

- You want to provide a simple interface to a complex subsystem
- There are many dependencies between clients and the implementation classes of an abstraction
  - Introduce a facade to decouple the subsystem from clients and other subsystems, thereby promoting subsystem independence and portability
- You want to layer your subsystems

Façade



Flyweight

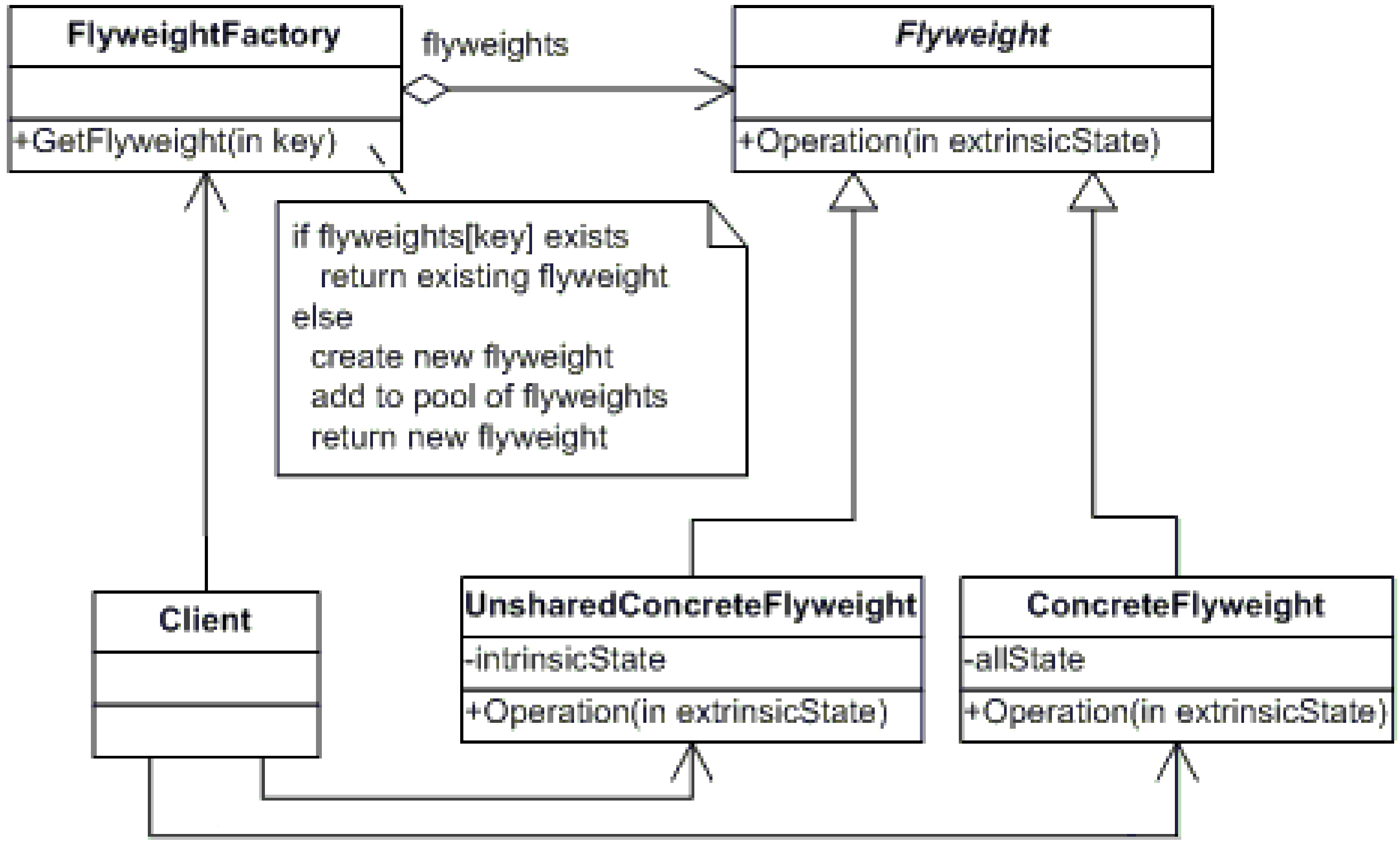
Intent

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

Applicability

- An application uses a large number of objects
- Storage costs are high because of the sheer quantity of objects
- Most object state can be made extrinsic
- Many groups of objects may be replaced by relatively few shared objects once extrinsic state is removed
- The application does not depend on object identity
  - Since flyweight objects may be shared, identity tests will return true for conceptually distinct objects

Flyweight



FlyweightFactory
+GetFlyweight(in key)
flyweights
Flyweight
+Operation(in extrinsicState)
if flyweights[key] exists
return existing flyweight
else
create new flyweight
add to pool of flyweights
return new flyweight
Client
UnsharedConcreteFlyweight
-intrinsicState
+Operation(in extrinsicState)
ConcreteFlyweight
-allState
+Operation(in extrinsicState)