Software Design Techniques and Mechanisms

Topic: Structural Design Patterns

Presenter: Drumea Vasile

UNIVERSITATEA TEHNICĂ A MOLDOVEI
fondată
1964

Chișinău 2021

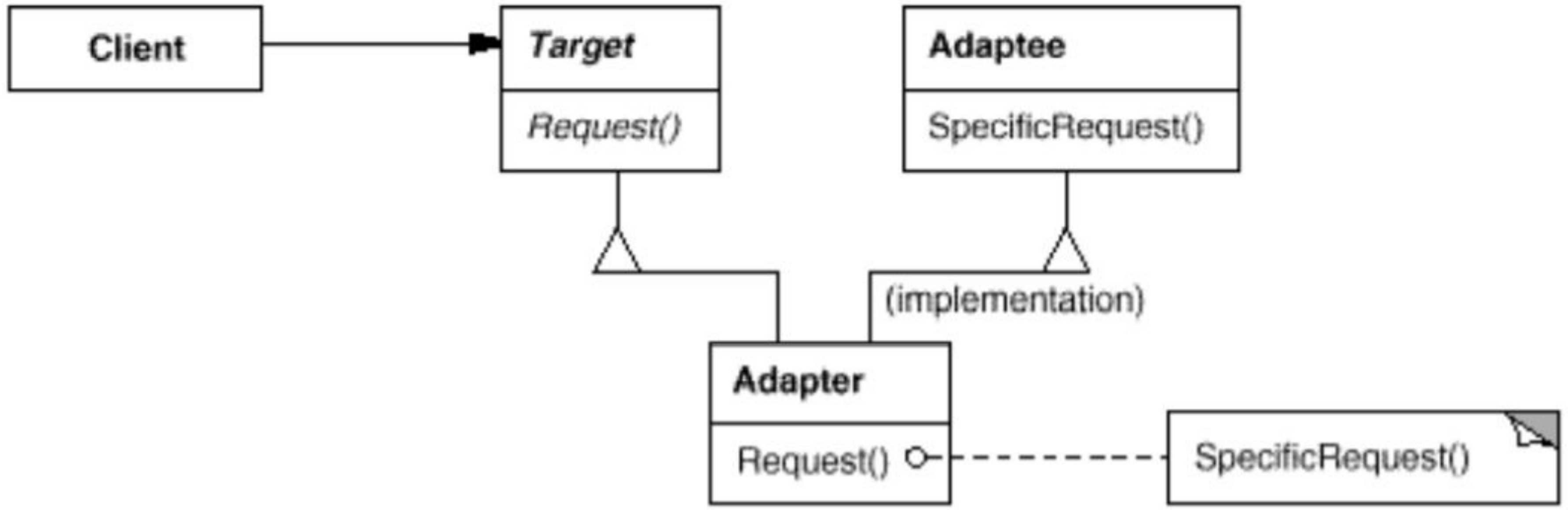
Overview

- The Structural Design Patterns are concerned with the ways of composing classes and objects into complex structures.
- This group can be divided into 2 groups:
  - Structural Class patterns (using inheritance)
  - Structural Object patterns (using composition)

Adapter Pattern

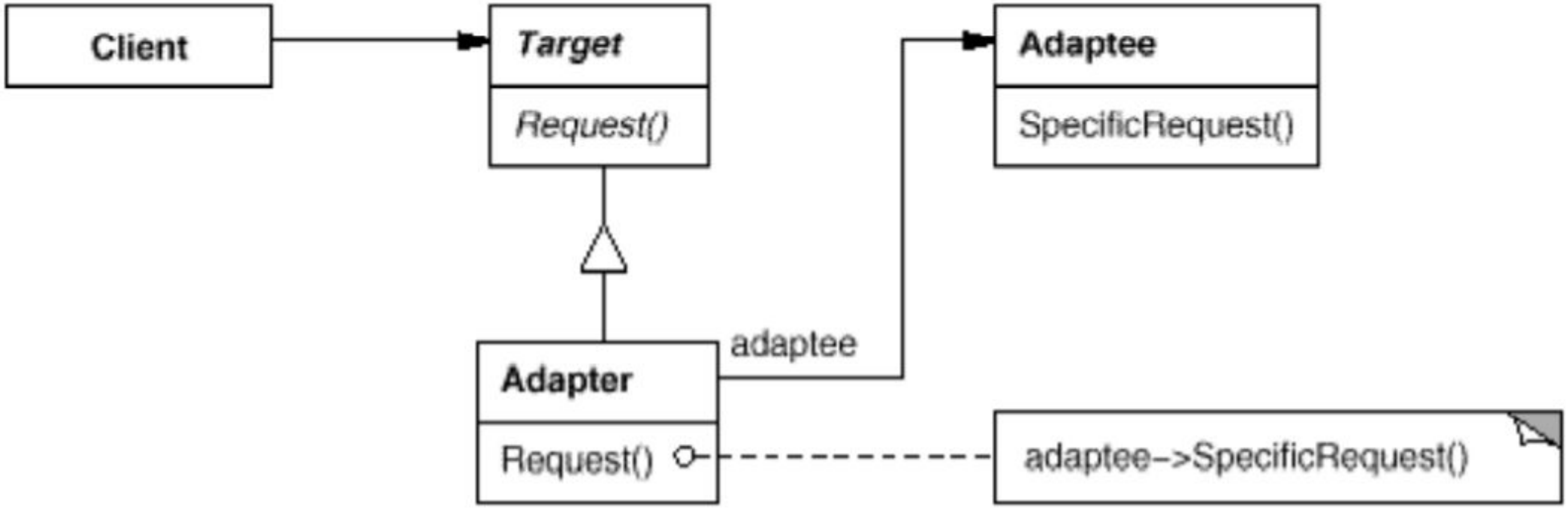
- It lets classes work together, that couldn’t otherwise because of incompatible parent classes.
- Aka Wrapper.

The UML Diagram for Class Adapter



UNIVERSITATEA TEHNICĂ A MOLDOVEI

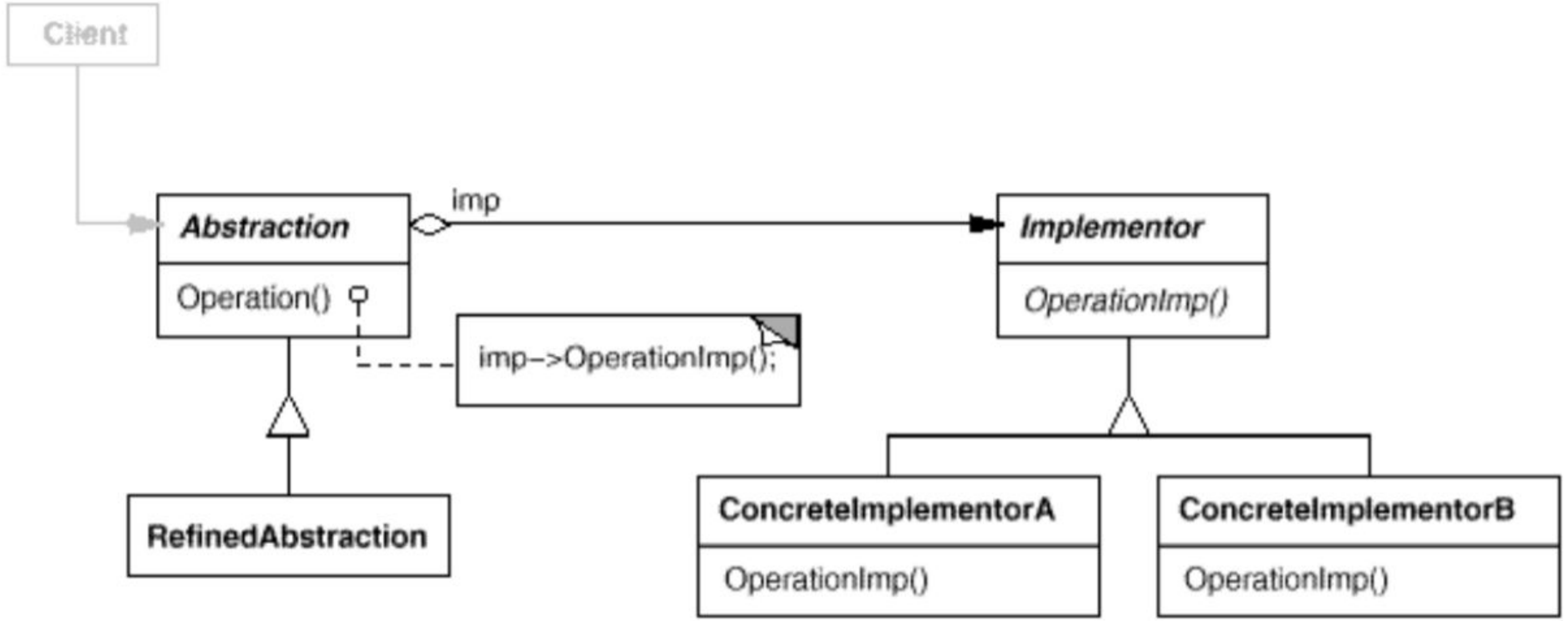
The UML Diagram for Object Adapter



Bridge Pattern

- Separates an object’s abstraction from its implementation so that these two levels could vary independently.
- Separate an inheritance hierarchy into 2 smaller hierarchies and using composition to connect them, this acting as the bridge between them.
- Use it when you have 2 orthogonal dimensions.

The UML Diagram for Bridge

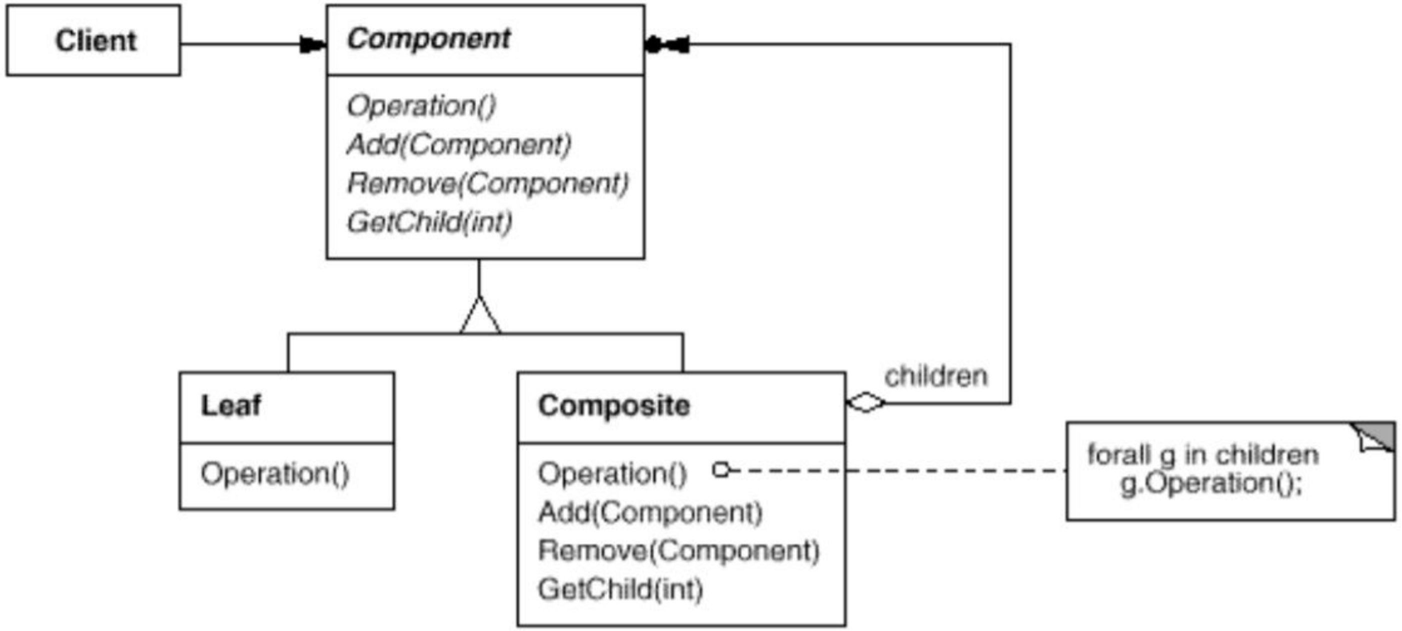


UNIVERSITATEA TEHNICĂ A MOLDOVEI

Composite Pattern

- A tree structure of simple and complex objects.
- It lets clients treat simple and complex objects uniformly.

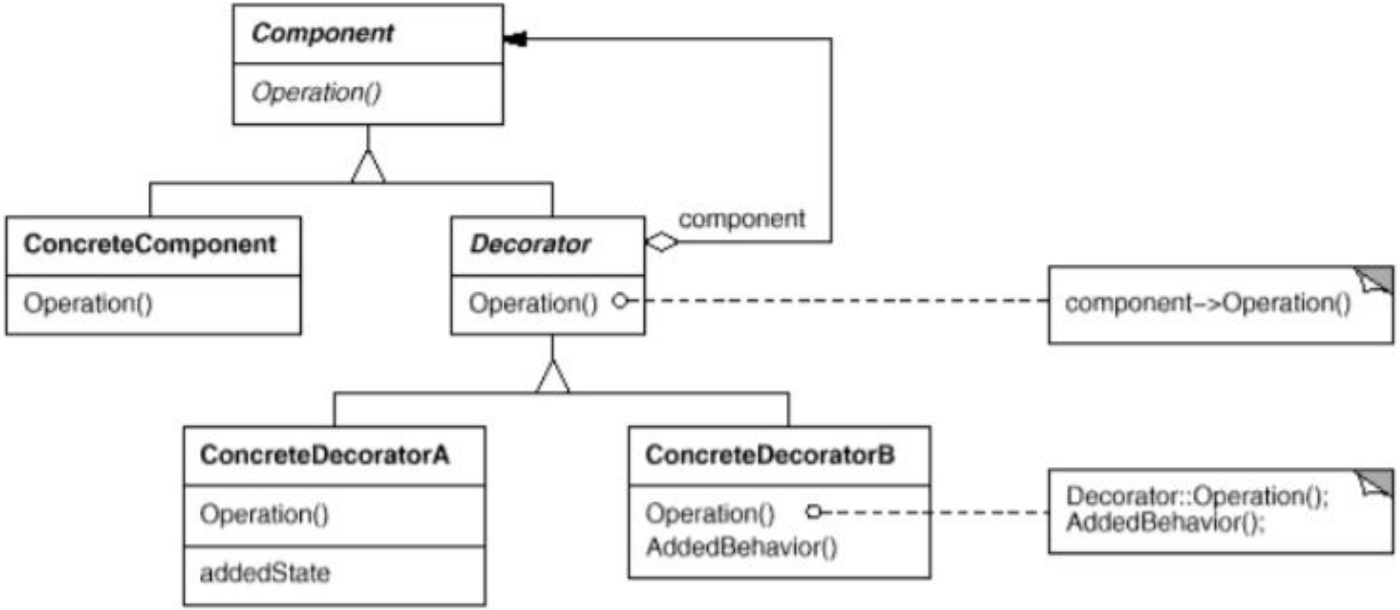
The UML Diagram for Composite



Decorator Pattern

- Add responsibilities to objects dynamically.
- Wrap the plain object into a specific decorator.

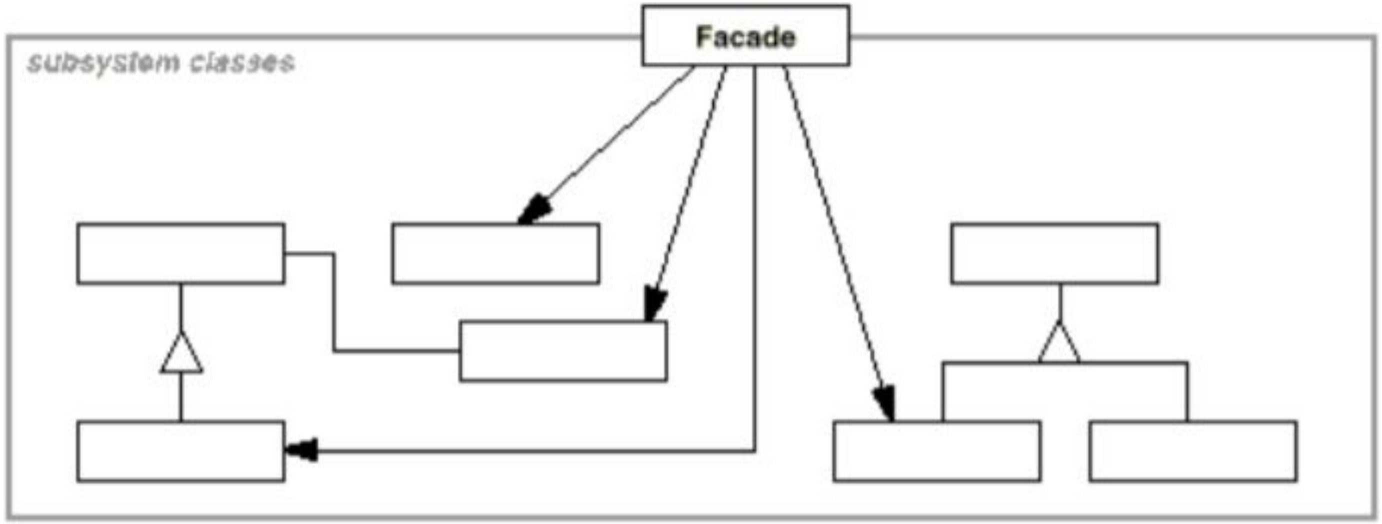
The UML Diagram for Decorator



Facade Pattern

- Provides a unified interface that represents multiple components.
- Wraps a complex sub-system with a simple abstraction.

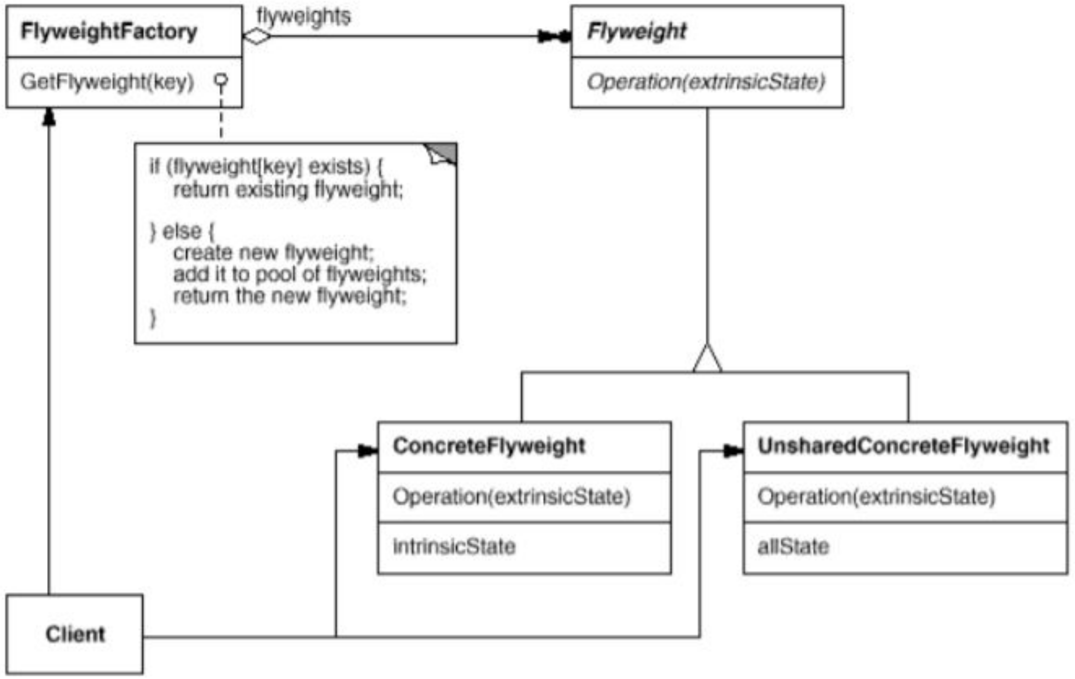
The UML Diagram for Facade



Flyweight Pattern

- Use sharing to support large numbers of fine grained objects efficiently.
- Each “flyweight” object is divided into two parts:
    - Extrinsic: state dependent part.
    - Intrinsic: state independent part.

The UML Diagram for Flyweight



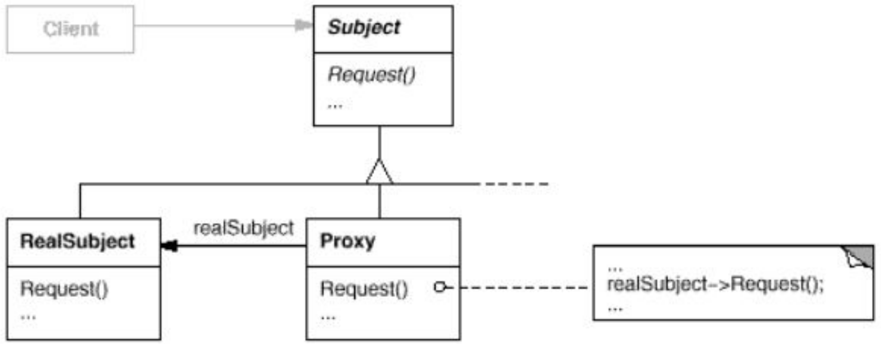
FlyweightFactory
GetFlyweight(key)
flyweights
Flyweight
Operation(extrinsicState)
if (flyweight[key] exists) {
return existing flyweight;
} else {
create new flyweight;
add it to pool of flyweights;
return the new flyweight;
}
ConcreteFlyweight
Operation(extrinsicState)
intrinsicState
UnsharedConcreteFlyweight
Operation(extrinsicState)
allState
Client

UNIVERSITATEA TEHNICĂ A MOLDOVEI

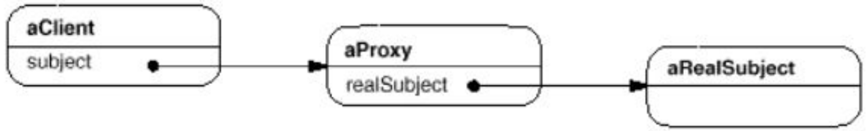
Proxy Pattern

- An object representing another object.
- Control the access to an object by wrapping it into another.
- Encapsulate the protected object in the proxy.

The UML Diagram for Proxy



Here's a possible object diagram of a proxy structure at run-time:



References

1. https://sourcemaking.com/design_patterns/structural_patterns
2. The "Gang of four", 1994, *Design Patterns: Elements of Reusable Object-Oriented Software*
3. P.S. All the diagrams are from [2].

Thanks for your attention!
Questions?