

Design Defects and Restructuring

Lecture 9

Sat, Nov 20, 2021

Creational Patterns

- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton

Structural Patterns

- Adapter
- Bridge
- Composite
- Decorator
- Façade
- Flyweight
- Proxy

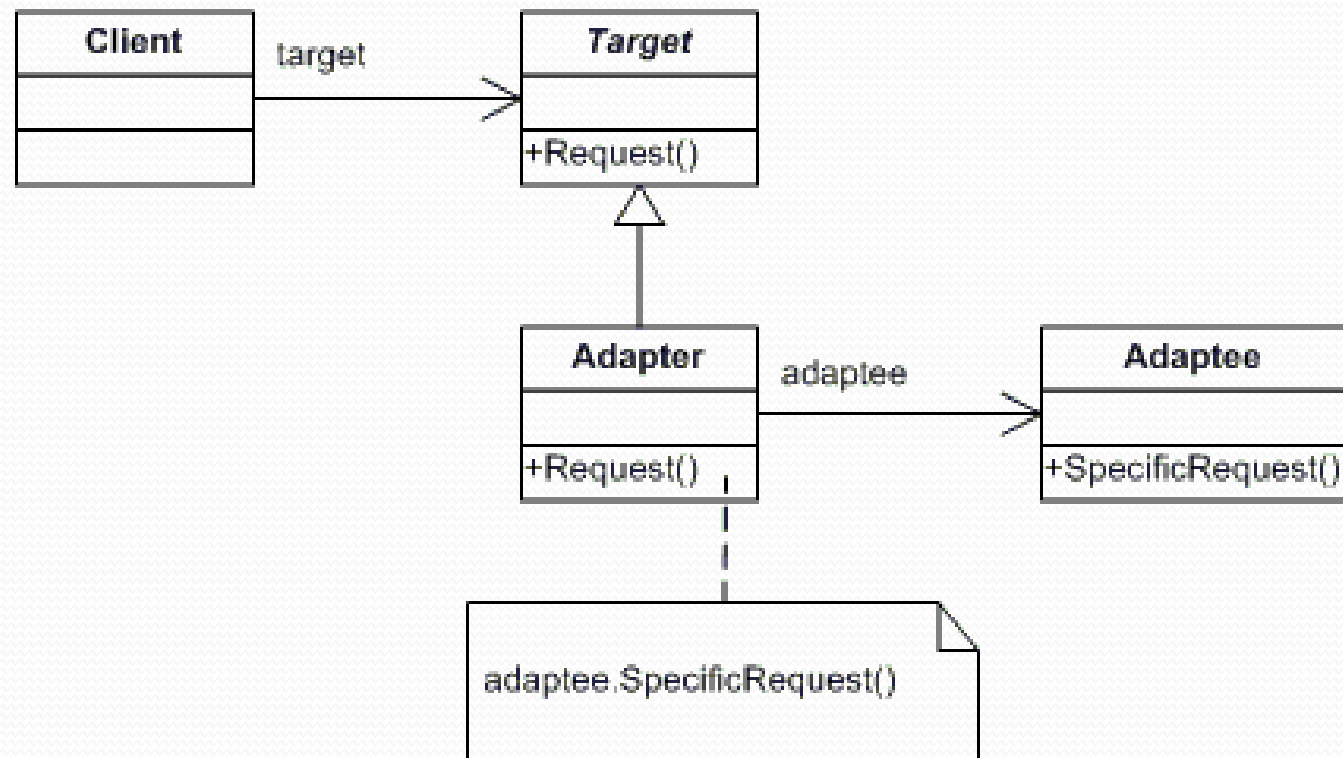
Adapter

- Intent
 - Convert the interface of a class into another interface clients expect
 - Adapter lets classes work together that could not otherwise because of incompatible interfaces

Adapter

- Applicability
 - You want to use an existing class, and its interface does not match the one you need
 - You want to create a reusable class that cooperates with unrelated or unforeseen classes, that is, classes that do not necessarily have compatible interfaces
 - You need to use several existing subclasses, but it is impractical to adapt their interface by sub-classing each one
 - An object adapter can adapt the interface of its parent class

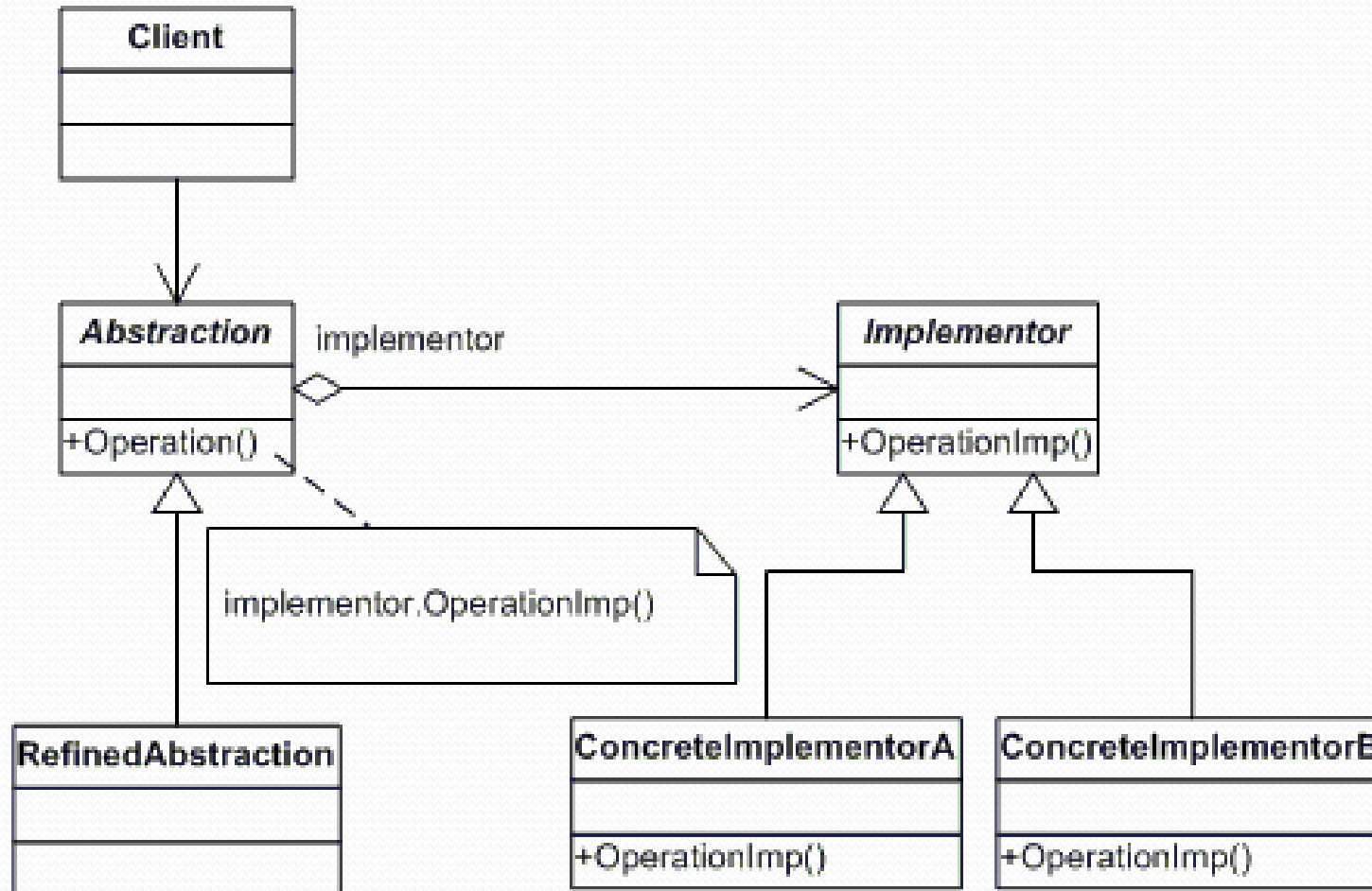
Adapter



Bridge

- Intent
 - Decouple an abstraction from its implementation so that the two can vary independently
- Applicability
 - You want to avoid a permanent binding between an abstraction and its implementation
 - Both the abstractions and their implementations should be extensible by sub-classing
 - Changes in the implementation of an abstraction should have no impact on clients

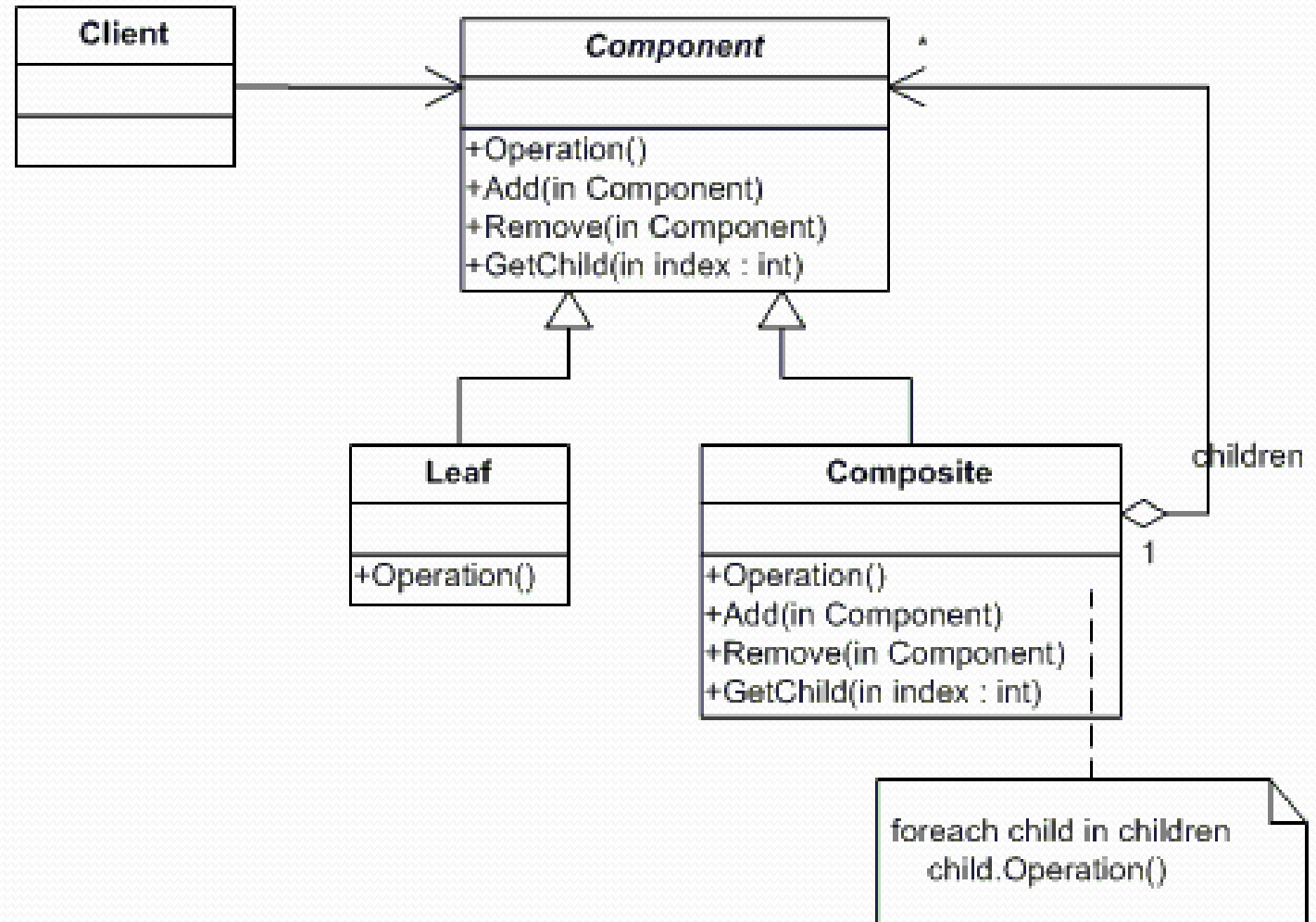
Bridge



Composite

- Intent
 - Compose objects into tree structures to represent part-whole hierarchies
 - Composite lets clients treat individual objects and compositions of objects uniformly
- Applicability
 - You want to represent part-whole hierarchies of objects
 - You want clients to be able to ignore the difference between compositions of objects and individual objects
 - Clients will treat all objects in the composite structure uniformly

Composite



Decorator

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

Decorator

- 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

