Structural Design Pattern

Hung Tran

Fpt software

October 26, 2021

Outline

Structural Pattern Overview

How classes and objects are composed fo form larger structure.

- **Adapter**: Convert the interface of a class into another interface.
- **Bridge**: Decouple an abstraction from its implementation.
- **Composite**: Compose objects into tree structure.
- **Decorator**: Attach additional responsibilities to an object dynamically.
- **Facade**: Provide a unified interface to a set of interfaces.
- **Flyweight**: Use sharing to support large numbers of fine-grained objects efficiently.
- **Proxy**: Provide a surrogate or placeholder for another object to control access to it.

Why we need Adapter Design Pattern?

-
-
-
-

Class wrapper

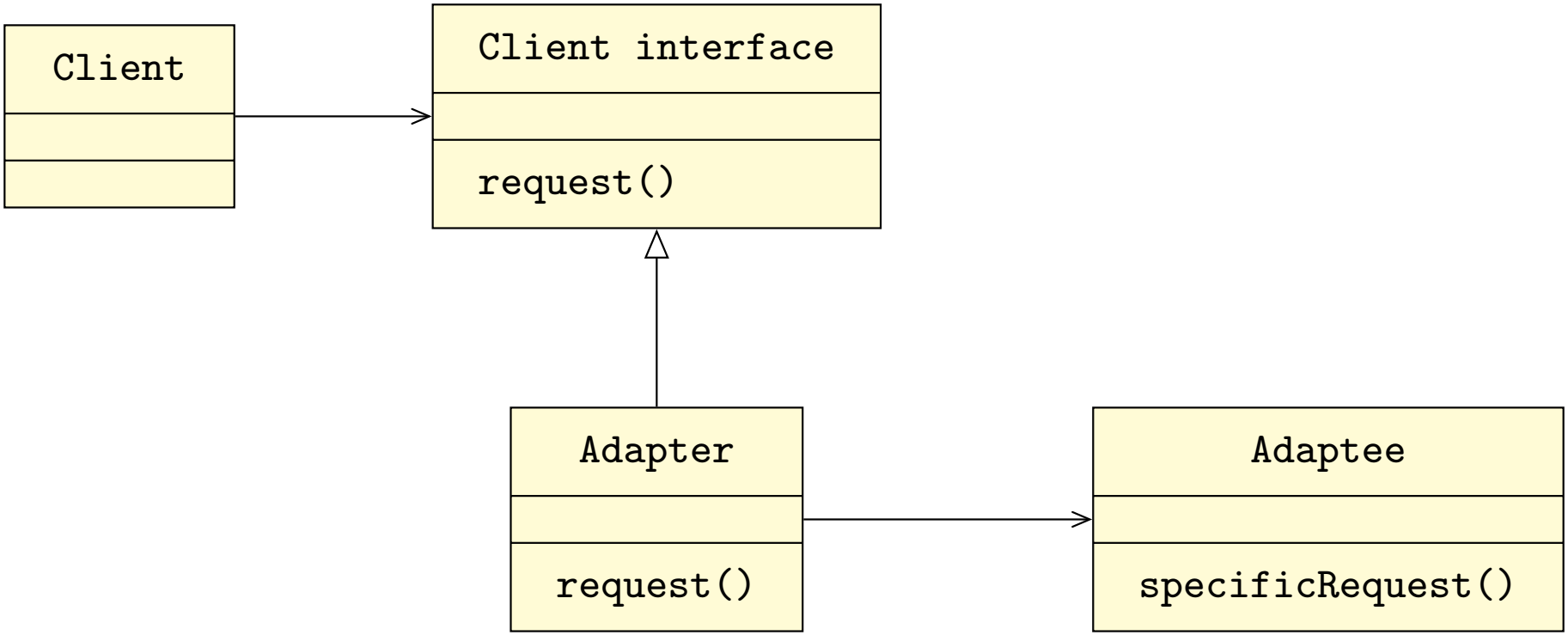
The Intent of Adapter Design Pattern

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

How to implement Adapter Design Pattern?

-
-
-

Structure of Adapter Pattern



Basic implementation

Where to use?

When only one instance should be use because:

- multiple instances cause data corruption.
- managing global state or shared state.
- multiple instances are not required.

Thank You!