SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT, NAGPUR - 440013

DESIGN PATTERNS

V SEMESTER SECTION A,B

COURSE COORDINATOR: RINA DAMDOO

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ADAPTER DESIGN PATTERNS

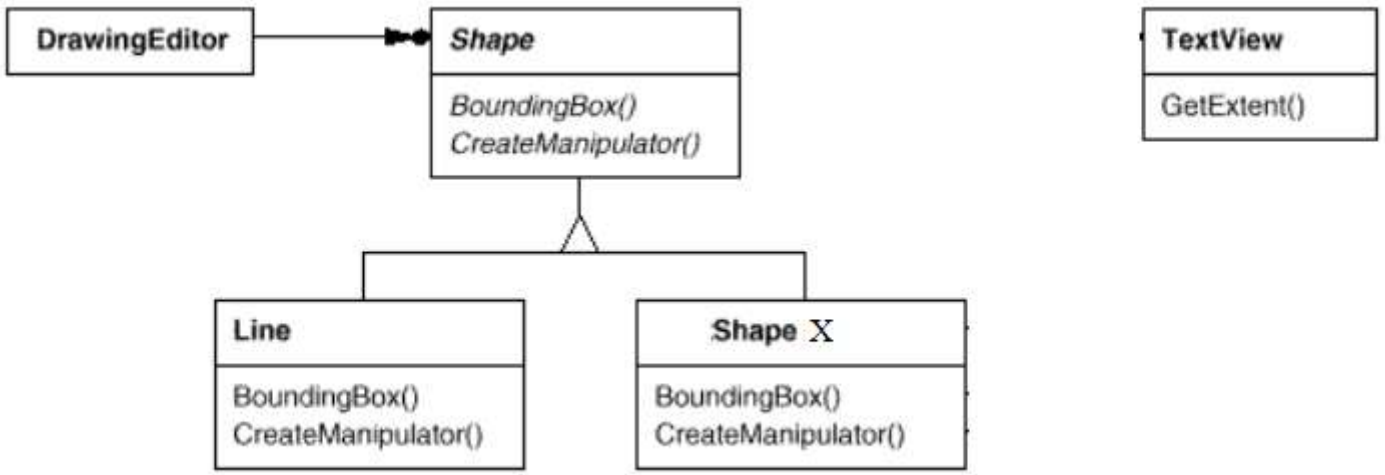
- **Intent**

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

Also Known As

Wrapper

ADAPTER EXAMPLE



We can define TextShape so that it adapts the TextView interface to Shapes. We can do this in one of two ways:

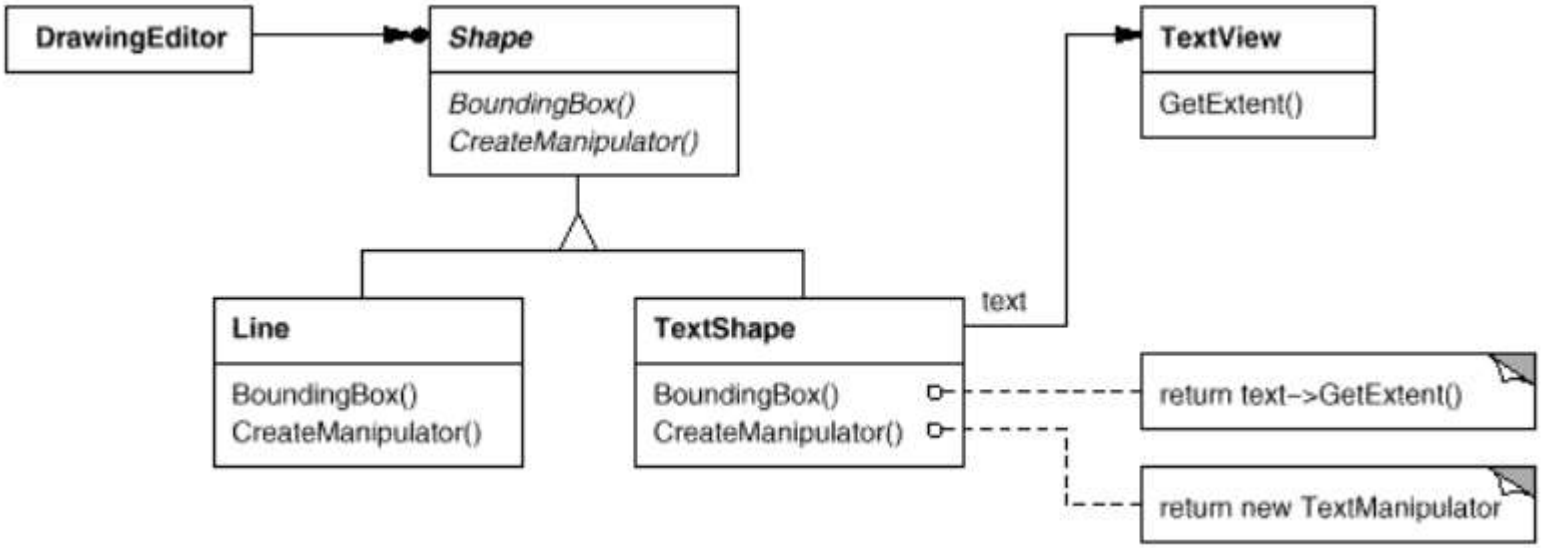
(1) by inheriting Shape's interface and TextView's implementation or

(2) by composing a TextView instance within a TextShape and implementing TextShape in terms of TextView's interface.

These two approaches correspond to the class and object versions of the Adapter pattern. We call TextShape an adapter.

ADAPTER EXAMPLE

Adaptee



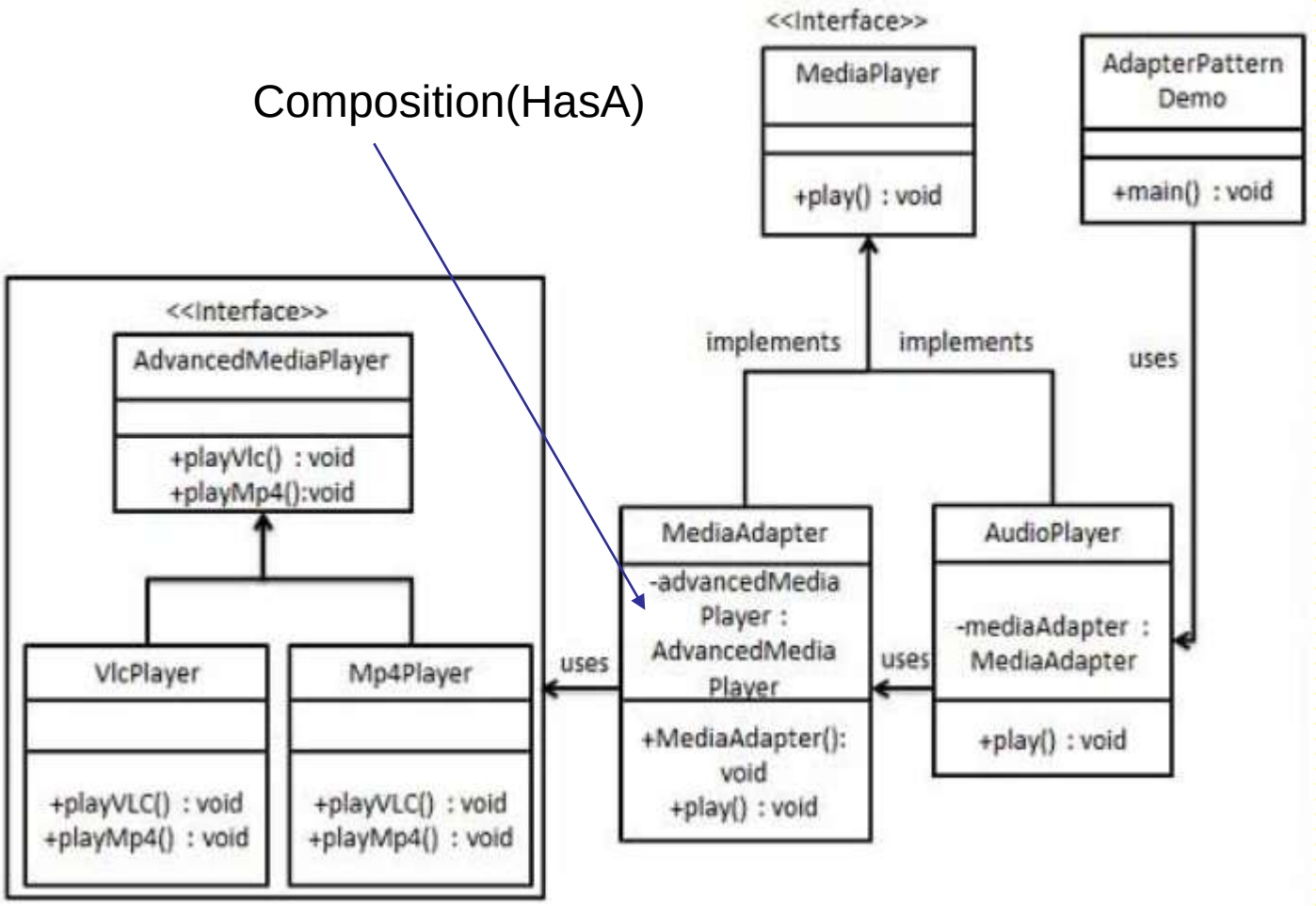
Adapter

ADAPTER DESIGN PATTERNS

Problem Statement:

Target

Composition(HasA)



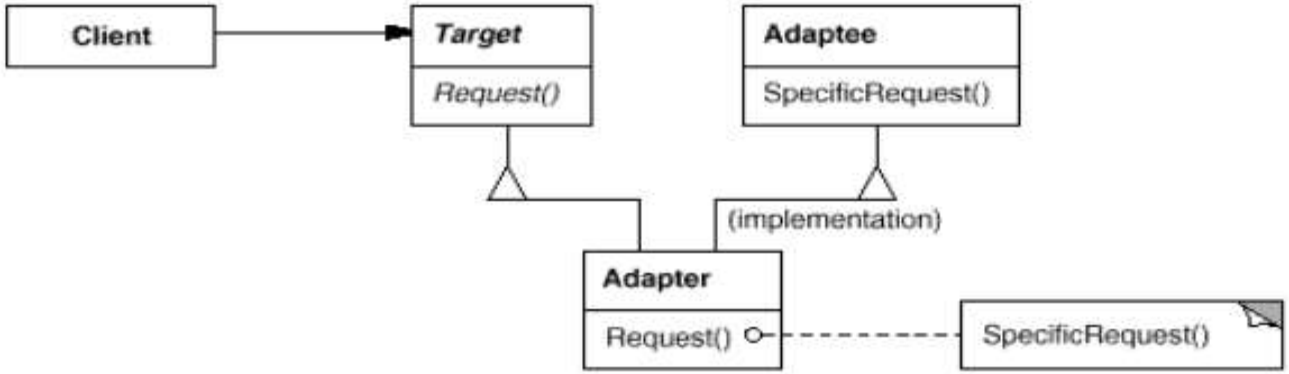
Adaptee

https://www.tutorialspoint.com/

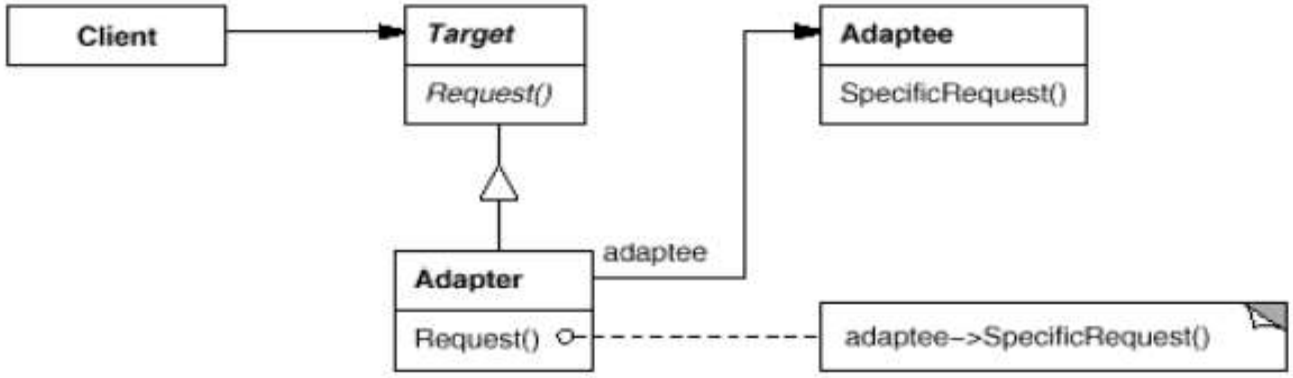
ADAPTER DESIGN PATTERNS

Structure

A class adapter uses multiple inheritance to adapt one interface to another:



An object adapter relies on object composition:



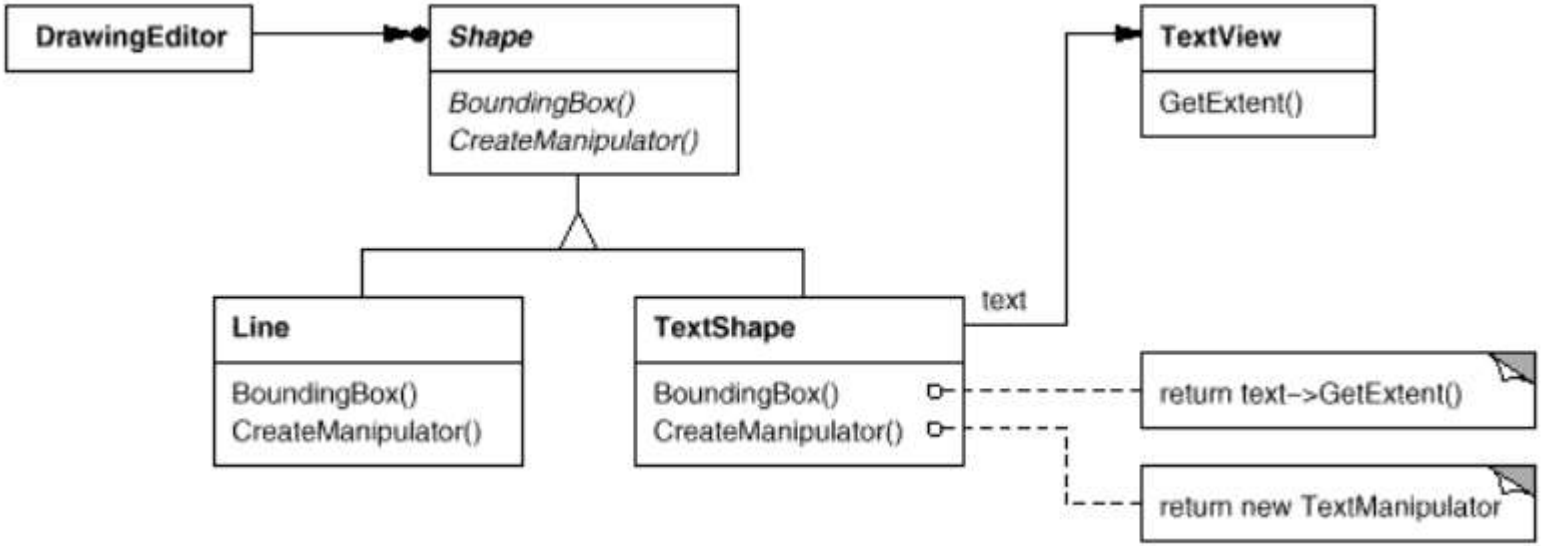
Applicability

Use the Adapter pattern when:

- 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 don't necessarily have compatible interfaces.
- (object adapter only) you need to use several existing subclasses, but it's impractical to adapt their interface by subclassing every one. An object adapter can adapt the interface of its parent class.

MOTIVATION

Adaptee



Adapter

PARTICIPANTS

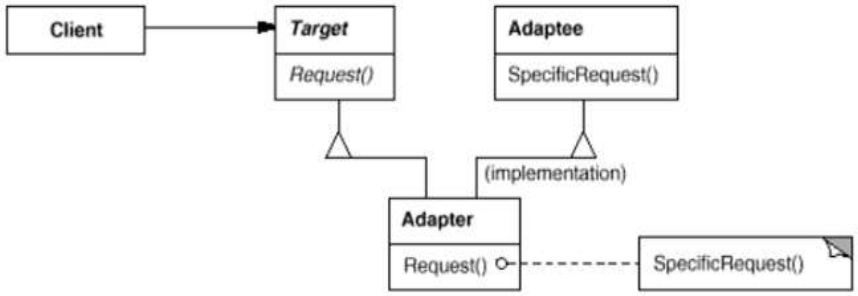
Participant	Responsibility
Target	• defines the domain-specific interface that Client uses.
Client	• collaborates with objects conforming to the Target interface.
Adaptee	• defines an existing interface that needs adapting.
Adapter	• adapts the interface of Adaptee to the Target interface.

COLLABORATIONS

- Clients call operations on an Adapter instance. In turn, the adapter calls Adaptee operations that carry out the request.

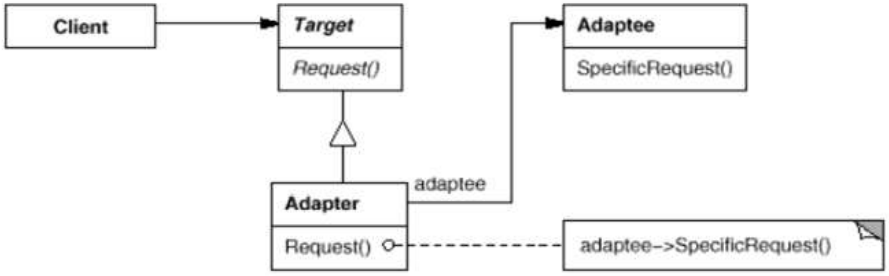
Structure

A class adapter uses multiple inheritance to adapt one interface to another:



Structure

An object adapter relies on object composition:



CONSEQUENCES

- Class and object adapters have different trade-offs.

A class adapter :

- adapts Adaptee to Target by committing to a concrete Adapter class. As a consequence, a class adapter won't work when we want to adapt a class and all its subclasses.
- lets Adapter override some of Adaptee's behavior, since Adapter is a subclass of Adaptee.
- introduces only one object, and no additional pointer indirection is needed to get to the adaptee.

CONSEQUENCES

- Class and object adapters have different trade-offs.

An Object adapter :

- Lets a single Adapter work with many Adaptees—that is, the Adaptee itself and all of its subclasses (if any).

- Makes it harder to override Adaptee behavior. It will require subclassing Adaptee and making Adapter refer to the subclass rather than the Adaptee itself.

Additional Consequences

- How much adapting does Adapter do?
- Using two-way adapters to provide transparency.

IMPLEMENTATION

Implementation Consider the following issues when applying the Adapter pattern:

- **language dependent issues are there**

Known Uses & Related Patterns

- A legacy application (legacy app) is a software program that is outdated or obsolete. Although a legacy app still works, it may be unstable because of compatibility issues with current operating systems, browsers and information technology (IT) infrastructures.
- Flipping the order of method arguments in a library's public API.
- device drivers

- **Bridge**
- **Decorator**
- **Proxy**