

2

(VII)

Patrón Adapter

(Patrones de diseño)

Diseño del Software

Grado en Ingeniería Informática del Software

Curso 2022-2023

Adapter (Adaptador)

- Patrón estructural (versión tanto de objetos como de clases)
- Propósito:

Convierte la interfaz de una clase en otra que es la que esperan los clientes. Permite que trabajen juntas clases que de otro modo no podrían por tener interfaces incompatibles.

- También conocido como:
 - Wrapper (Envoltorio)

Motivación

- **Supongamos que estamos haciendo un editor de dibujo**
 - La abstracción fundamental es el objeto gráfico (**Shape**), que puede dibujarse a sí mismo
 - Define una subclase para cada tipo de objeto gráfico: **LineShape, PolygonShape...**
- **Supongamos que, para implementar una subclase **TextShape** (bastante más compleja que las anteriores) queremos echar mano de una clase **TextView** que nos proporciona la biblioteca gráfica**

¿Cómo lo hacemos?

- **La interfaz de `TextView` no tendrá nada que ver con la de `Shape`**

- Le faltarán algunas operaciones, otras las tendrá pero con otro nombre, o bien recibirán parámetros de otro tipo...

- **Podríamos cambiar la clase `TextView` para que implemente dicha interfaz**

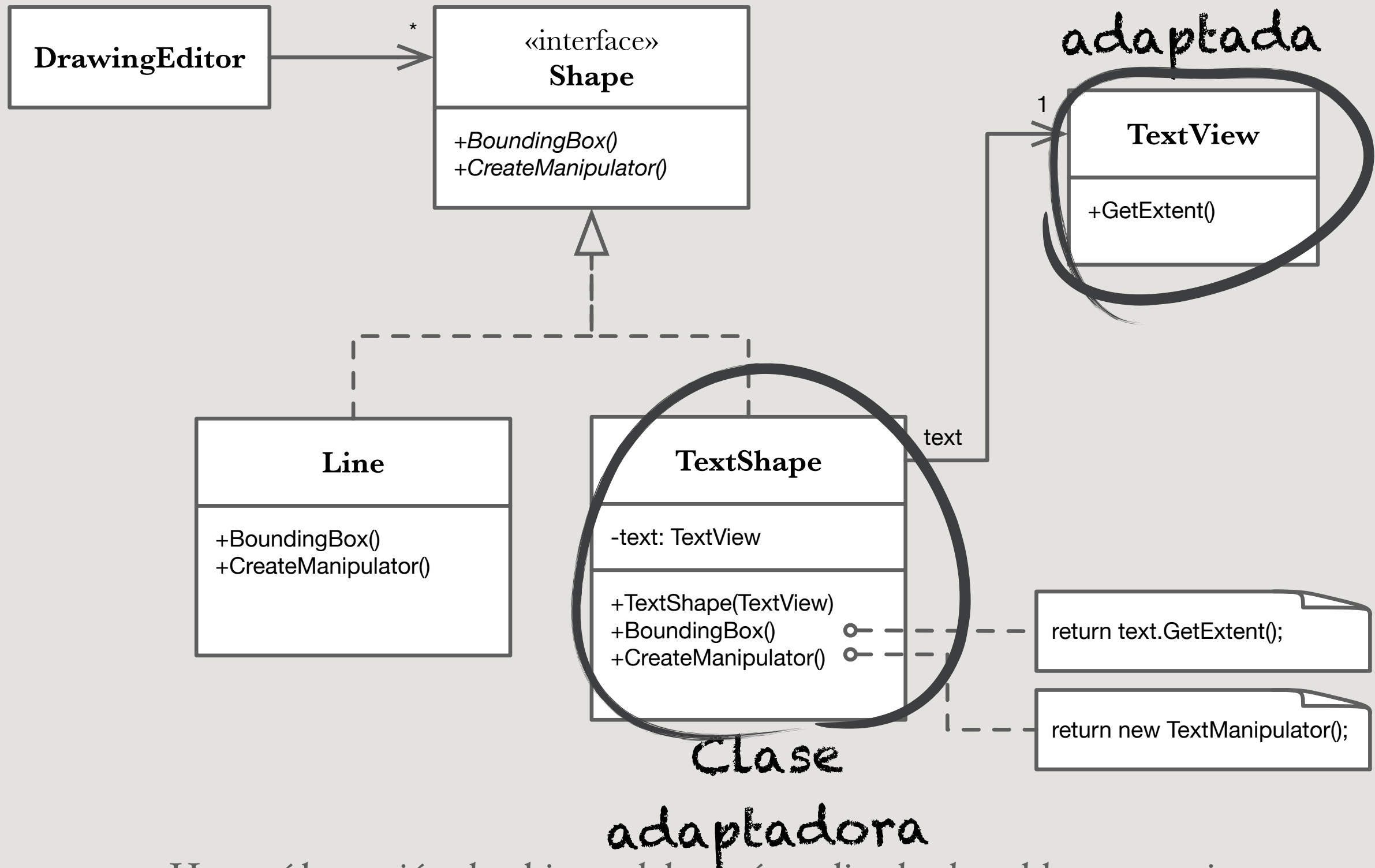
- Pero eso normalmente no es una opción

- ¿Podemos cambiar el código fuente de `JTextField` en Swing para reutilizarlo en nuestra aplicación?
 - ¿Tendría sentido hacerlo?

Solución

- En vez de eso, creamos una clase **TextShape** que adapte la interfaz de **TextView** a la de **Shape**
- Dos opciones:
 - Heredando la interfaz de **Shape** y la implementación de **TextView** (versión de clases)
 - Mediante composición de objetos, haciendo que **TextShape** delegue en una instancia de **TextView** (versión de objetos)

Solución



He aquí la versión de objetos del patrón aplicada al problema anterior

Solución

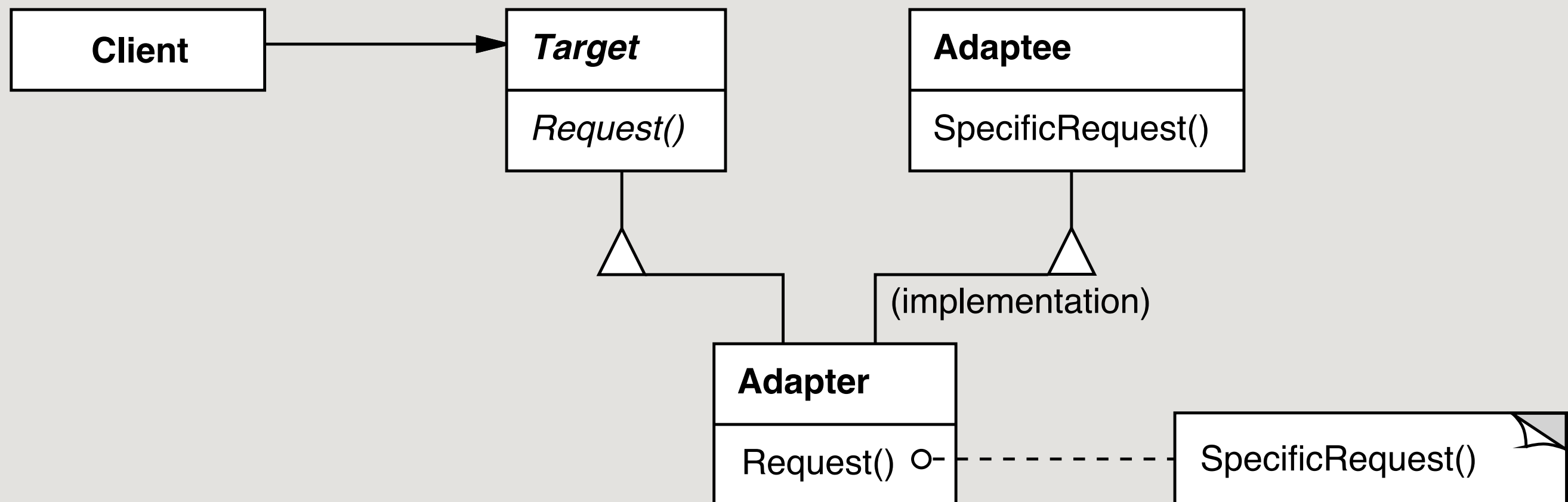
- **Naturalmente, muchas veces el adaptador tendrá que implementar funcionalidad que la clase adaptada no hace**
 - Por ejemplo, en el caso anterior, arrastrar la forma de texto a otra posición
 - ▶ (Por medio del apropiado objeto **TextManipulator**)

Aplicabilidad

- Queremos usar una clase existente, y ésta no tiene la interfaz (es decir, el tipo) que necesitamos
- Queremos crear una clase reutilizable que coopere con clases con las que no está relacionada
 - Que no tendrán, por tanto, interfaces compatibles
- (Sólo la versión de objetos) Necesitamos usar varias subclases existentes pero sin tener que adaptar su interfaz creando una nueva subclase de cada una
 - Un adaptador de objetos puede adaptar la interfaz de su clase padre

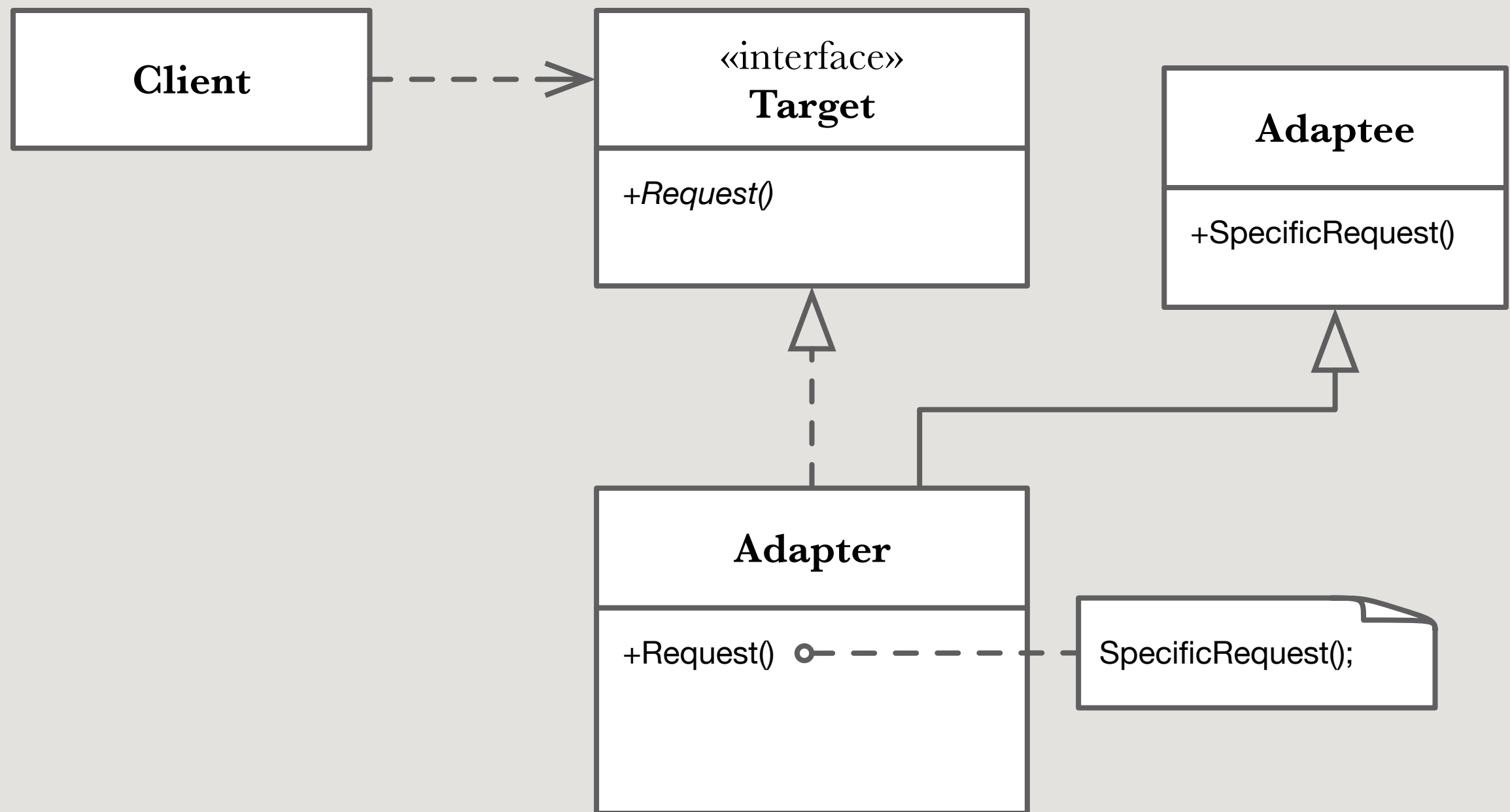
Estructura (versión de clases)

- Un adaptador de clases usa herencia múltiple:



La versión de clases del patrón *Adapter*, tal como aparece en el GoF

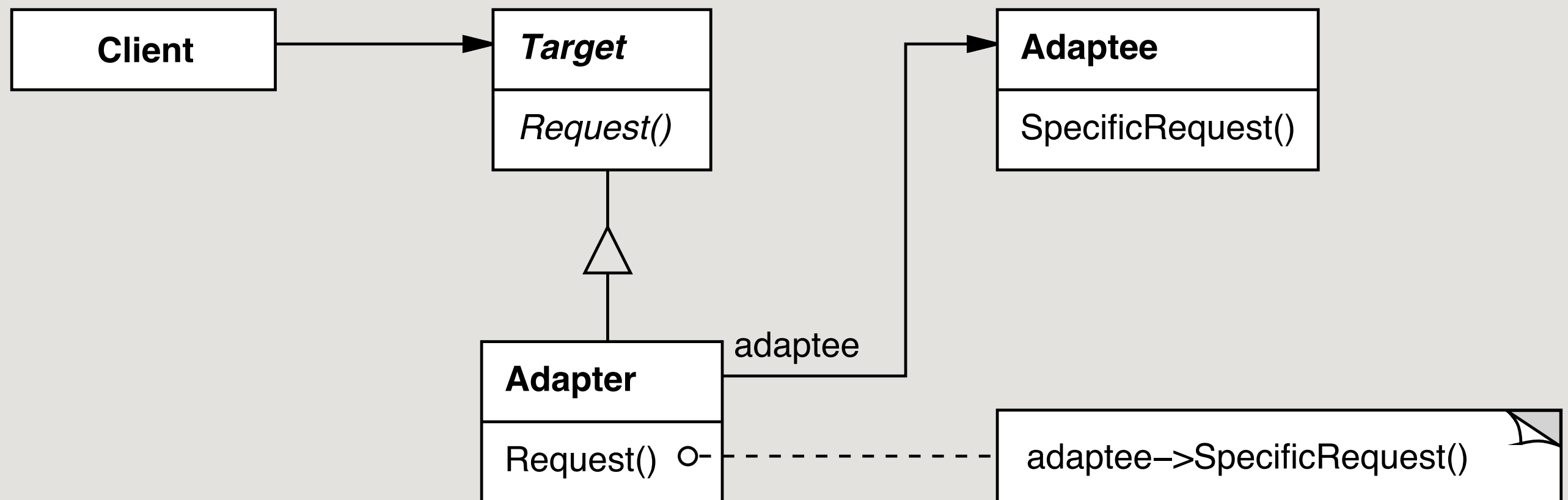
Estructura (versión de clases)



La versión de clases del patrón *Adapter*, en UML

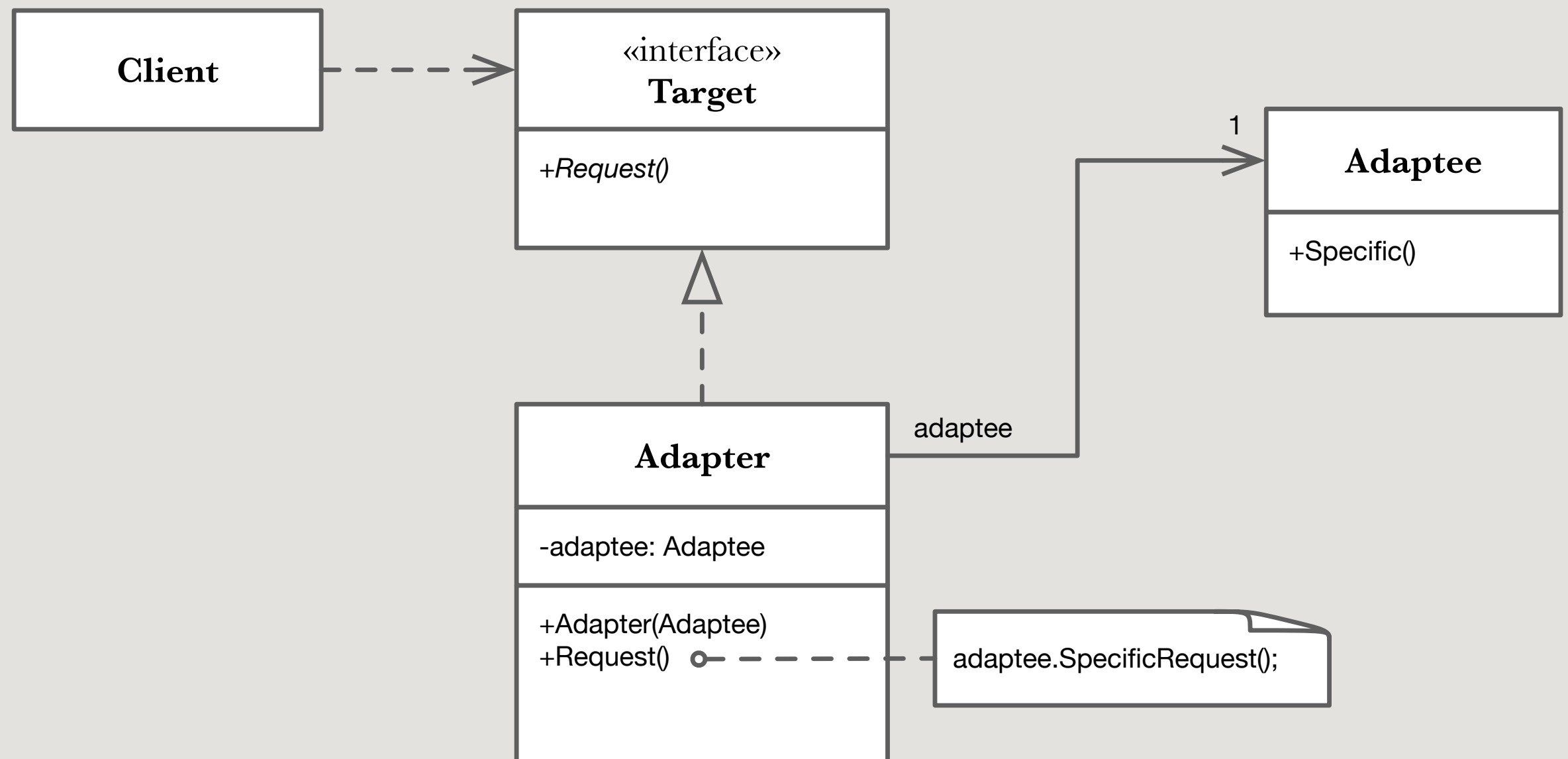
Estructura (versión de objetos)

- Un adaptador de objetos se basa en composición de objetos:



La estructura del patrón *Adapter*, tal como aparece en el GoF

Estructura (versión de objetos)



Un adaptador de objetos se basa en composición de objetos

Participantes

- ***Target (Shape)***

- Define la interfaz específica del dominio (es la que usarán los clientes)

- ***Client (DrawingEditor)***

- Colabora con objetos de tipo **Target**

- ***Adaptee (TextView)***

- La clase existente cuya interfaz necesita ser adaptada

- ***Adapter (TextShape)***

- Adapta la interfaz de **Adaptee** a la de **Target**

Consecuencias

De clases y de objetos

- **Las versiones de clases y de objetos de este patrón tienen diferentes ventajas e inconvenientes**
- **Un adaptador de clases**
 - Adapta una clase concreta a una interfaz (no se puede usar cuando queremos adaptar una clase y todas sus subclases)
 - Permite que el adaptador redefina parte del comportamiento de la clase adaptada (es una subclase de aquélla)
 - Introduce un solo objeto adicional, sin indirección

De clases y de objetos

● Un adaptador de objetos

- Permite que un único adaptador funcione no sólo con un objeto de la clase adaptada, sino de cualquiera de sus subclases
- Permite adaptar objetos existentes
- No es del tipo del objeto adaptado

Otras cuestiones

● ¿Cuánta adaptación hace el adaptador?

- Puede ir desde un mero cambio de nombres en las operaciones a tener que implementar un conjunto totalmente nuevo de operaciones, depende de lo parecidas que sean la clase que hay que adaptar y la clase objetivo

● Adaptadores «conectables»

- Es posible construir una especie de adaptación de interfaces en la propia clase, a priori