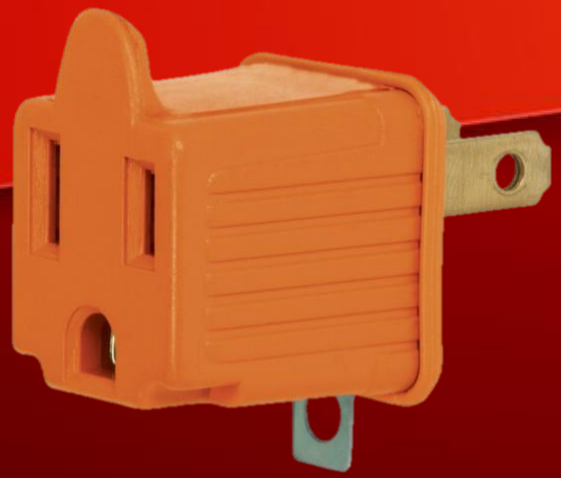
Patrón de diseño Adapter.

Noemí Esther Flores Pardo.



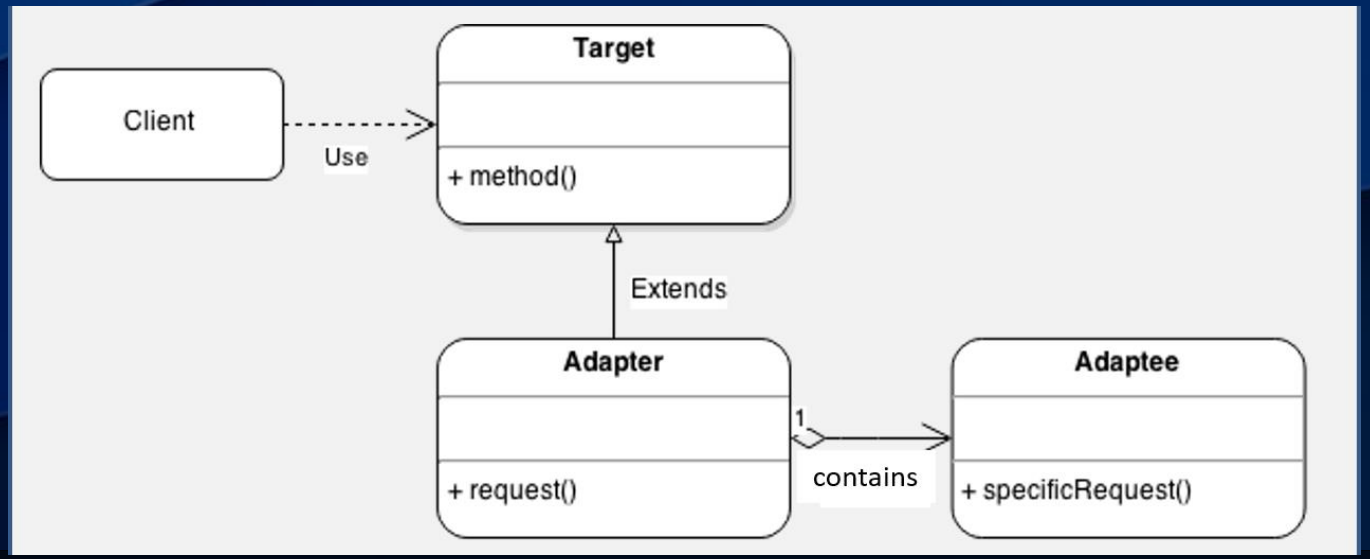
Adapter.

Hay un enchufe de tres patas pero donde se conectará solo se puede usar un enchufe de dos, se necesita insertar la clavija en el sin modificar ninguna de las dos.
Solución: Usar un adaptador.

¿Para qué sirve y cuándo usarlo?

- Es utilizado cuando se tienen interfaces incompatibles, pero a pesar de serlo tienen una funcionalidad similar.
- Este patrón es implementado cuando se desea hacer igual la forma de trabajar de estas interfaces incompatibles, para esto se crea una clase intermedia que funciona como un adaptador.
- Esta clase adaptador proporcionará los métodos para interactuar con la interface incompatible.

Componentes.



Client

Actor que interactua con el Adapter.

Target

Interface que nos permitirá homogenizar la forma de trabajar con las interfaces incompatibles, esta interface es utilizada para crear los Adapter.

Adapter

Representa la implementación del Target, el cual tiene la responsabilidad de mediar entre el Client y el Adaptee. Oculta la forma de comunicarse con el Adaptee.

Adaptee

Representa la clase con interface incompatible.

Bridge

Aunque son parecidos tienen objetivos diferentes ya que Bridge está pensado para separar una interfaz de su implementación, mientras que Adapter cambia la interfaz de un objeto existente.

Decorator

Adapter modifica la interfaz de un objeto, pero mantiene su funcionalidad, Decorator permite que la interfaz sea igual pero mejora su funcionalidad.

Facade

Es una alternativa a Adaptador cuando en lugar de llamar a un solo objeto se necesita llamar a varios.

Proxy

Es similar a Adaptador en el sentido en que proporcionan una interfaz, con la diferencia en que Proxy ofrece la misma interfaz que la clase a la que se llama.

Patrones con los que se relaciona.

Ejemplo del mundo real.

- Mediante la implementación del patrón de diseño *Adapter* crearemos un adaptador que nos permite interactuar de forma homogénea entre dos API bancarías, las cuales nos permite aprobar créditos personales, sin embargo, las dos API proporcionadas por los bancos cuenta con interfaces diferentes y aunque su funcionamiento es prácticamente igual, las interfaces expuestas son diferentes, lo que implica tener dos implementaciones diferentes para procesar los préstamos con cada banco. Mediante este patrón crearemos un adaptador que permitirá ocultar la complejidad de cada implementación del API, exponiendo una única interface compatible con las dos API proporcionadas, además que dejáramos el camino preparado por si el día de mañana llegara una nueva API bancaría.