Patrones de Diseño: Patrones Estructurales. Tema 4-2: Adapter

Descripción del patrón

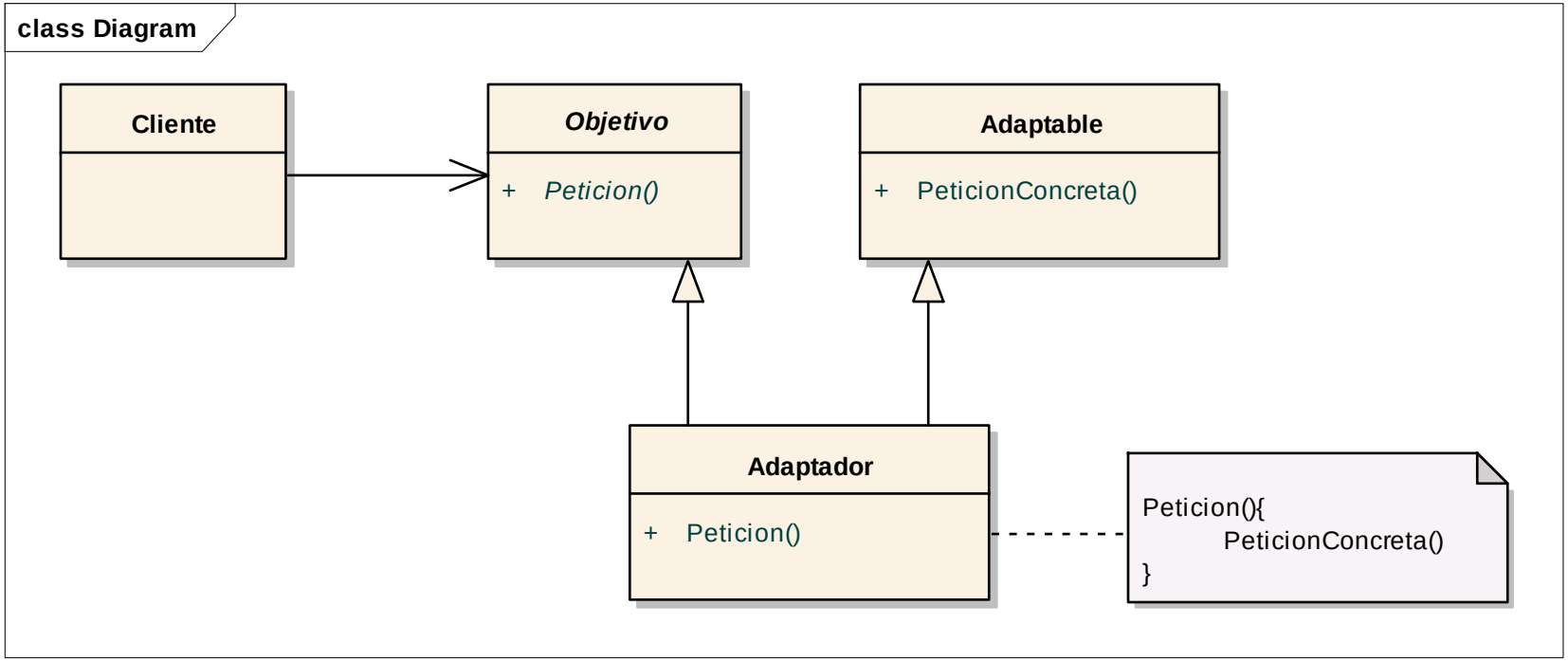
- **Nombre:**
  - Adaptador
  - También conocido como Wrapper (encapsulador)
- **Propiedades:**
  - Tipo: estructural
  - Nivel: clase
- **Objetivo o Propósito:**
  - Sirve de intermediario entre dos clases, convirtiendo las interfaces de una clase para que pueda ser utilizada por otra. Permite que cooperen clases que tienen interfaces incompatibles.

Aplicabilidad

- Use el patrón Adapter cuando:
  - Se quiere usar una clase existente y su interfaz no concuerda con la que se necesita.
  - Deba realizar una traducción entre interfaces de varios objetos.
  - Un objeto tiene que actuar como intermediario para un grupo de clases, y solo es posible saber en tiempo de ejecución qué clase será utilizada.
  - Se quiere crear una clase reutilizable que coopere con clases no relacionadas o que no han sido previstas, es decir, clases que no tienen porque tener interfaces compatibles.
  - Es necesario usar varias subclases existentes, pero no resulta práctico adaptar su interfaz heredando de cada una de ellas (adaptador de objetos).

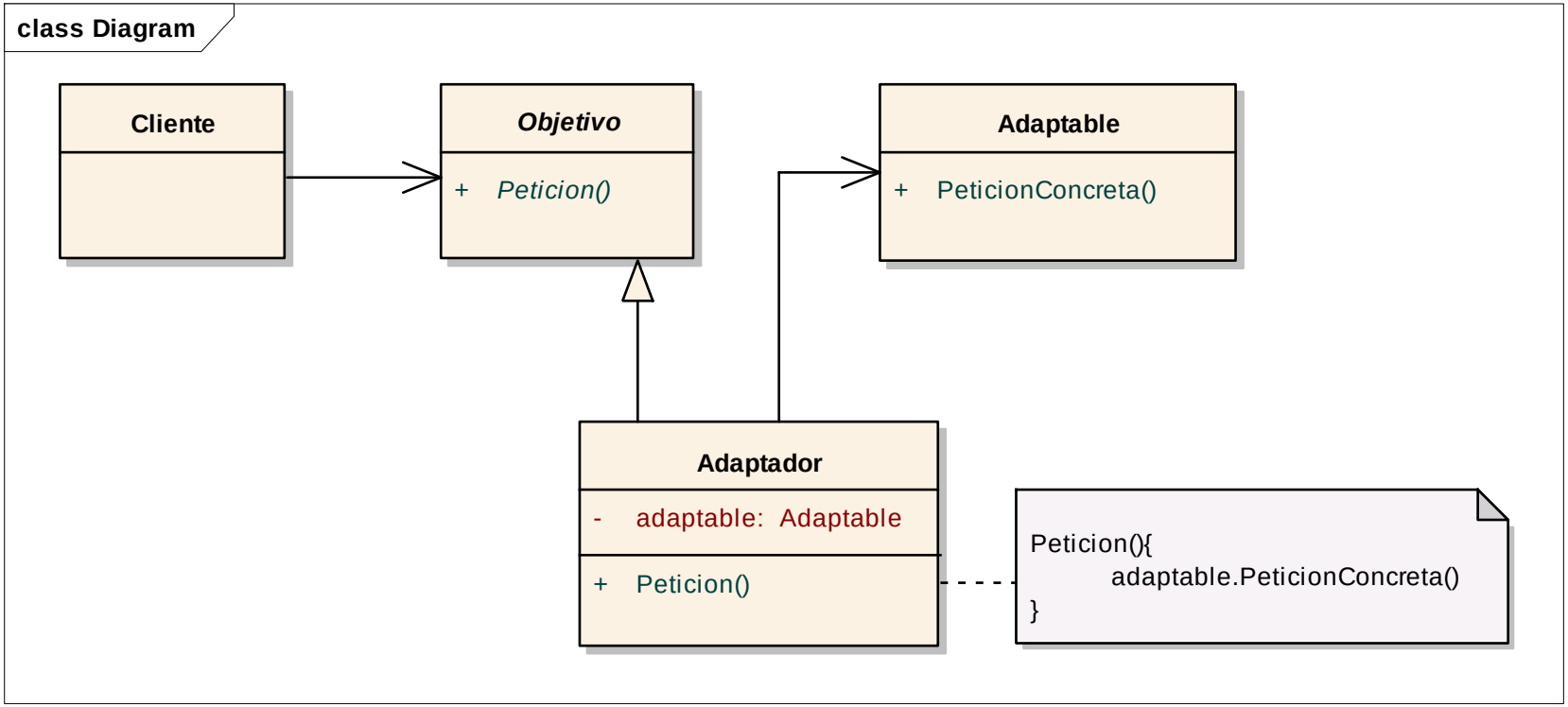
Estructura

Un adaptador de clases usa la herencia múltiple para adaptar una interfaz a otra:

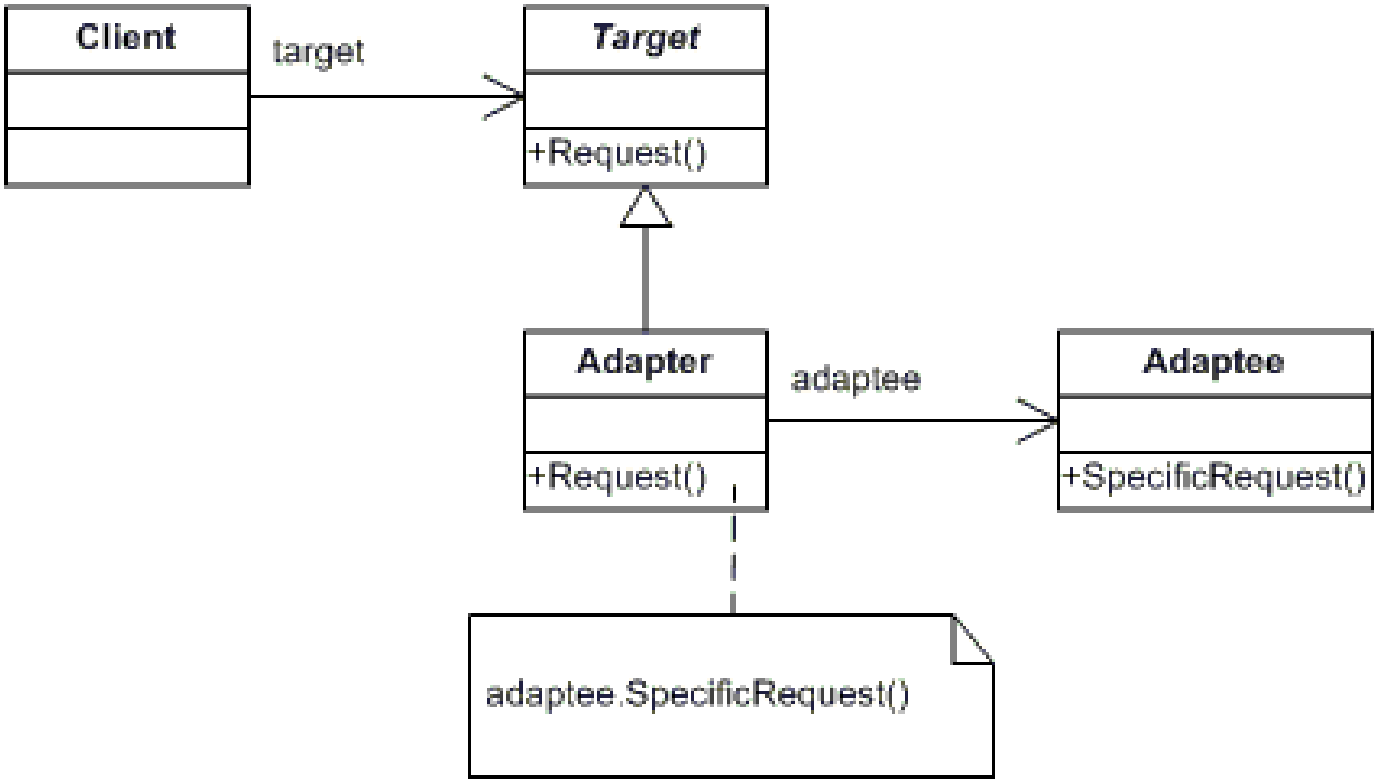


Estructura

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



Estructura


Client
target
Target
+Request()
Adapter
adaptee
Adaptee
+Request()
+SpecificRequest()
adaptee.SpecificRequest()

Estructura. Participantes

- **Objetivo**: Define la interfaz específica del dominio que usa el Cliente.
- **Cliente**: Colabora con objetos que se ajustan a la interfaz Objetivo.
- **Adaptable**: Define una interfaz existente que necesita ser adaptada.
- **Adaptador**: Adapta la interfaz de Adaptable a la interfaz Objetivo.

Estructura. Variaciones

- **Variaciones del patrón:**
- **Un adaptador y múltiples adaptables.** Dependiendo del diseño del sistema el adaptador puede añadir funcionalidad a todos los Adaptables a la vez.
- **Adaptadores no basados en interfaz.** Se da en las situaciones en las que no se puede utilizar interfaces, por ejemplo, si se reciben componentes completos que no implementan ninguna interfaz.
- **Una capa de interfaz entre el invocador y el adaptador, y otra entre el adaptador y el adaptable.** La capa entre el invocador y el adaptador permite que se puedan introducir fácilmente nuevos adaptadores en el sistema en tiempo de ejecución. Y entre el adaptador y el adaptable, hace que los adaptables sean cargados dinámicamente en tiempo de ejecución.

Consecuencias

- El cliente es independiente de las clases finales que utiliza.
- Es posible utilizar la clase Adaptador para monitorizar que clases llaman a qué métodos de las clases finales.

- *Adaptador de Clase:*
  - La clase adaptadora puede redefinir y ampliar la interfaz de la clase adaptada.
  - La clase adaptadora adapta una clase Adaptable a Objetivo, pero se refiere únicamente a una clase Adaptable concreta, por lo tanto no nos servirá para todas sus subclases.

Consecuencias

- *Adaptador de Objeto:*
  - Permite que un mismo Adaptador funcione con muchos Adaptables.
  - La clase adaptadora puede utilizar todas las subclases de la clase adaptada, ya que tiene constancia de los objetos que instancia.
  - Hace que sea más difícil redefinir el comportamiento de Adaptable. Se necesitará crear una subclase de Adaptable y hacer que el Adaptador se refiera a la subclase en vez de a la clase Adaptable en sí.

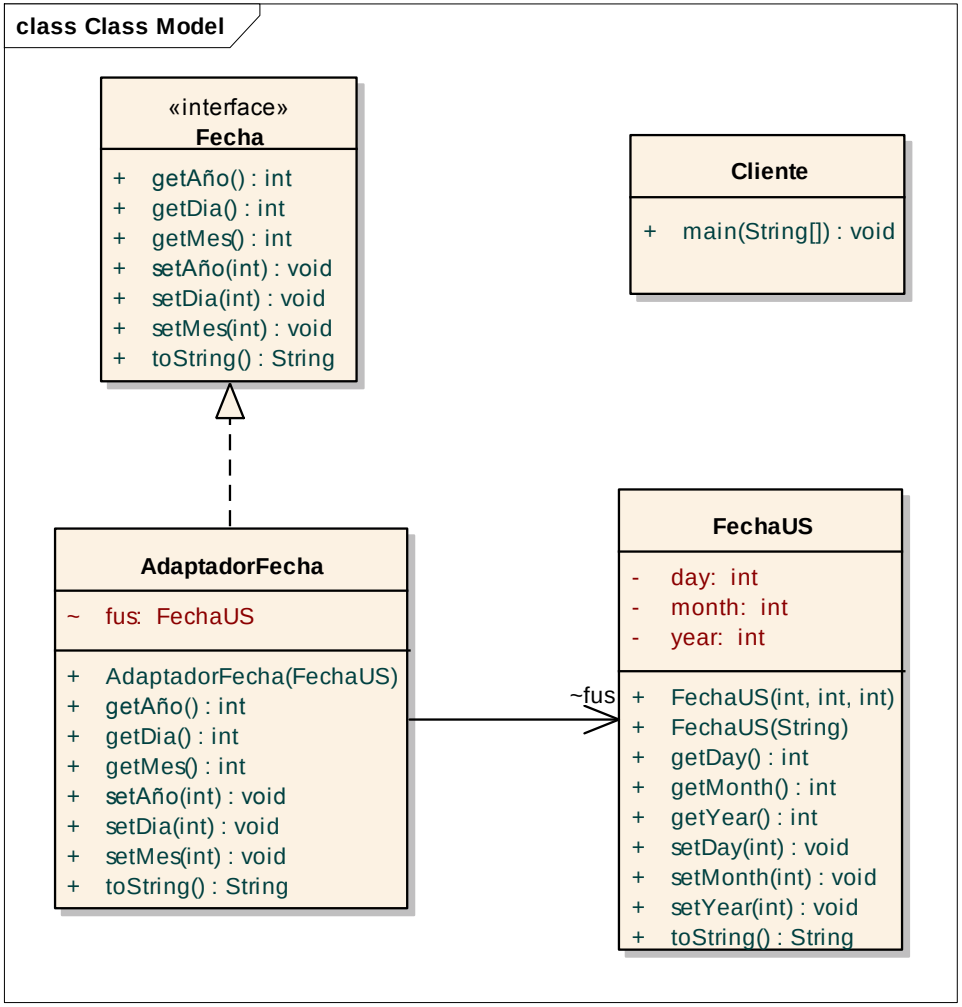
Patrones relacionados

- **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.

Código de ejemplo

Adaptador de fechas

Código de ejemplo



Código de ejemplo

- Identificamos a continuación los elementos del patrón:
  - Objetivo: Fecha.
  - Cliente: Cliente.
  - Adaptable: FechaUS.
  - Adaptador: AdaptadorFecha.