Adapter

Nombre:
Adapter

Clasificación del patrón:
Estructural

Intención:
Convierte la interfaz de una clase en otra interfaz esperada por el cliente. Adapter permite que clases que no podrían trabajar juntas debido a que tienen interfaces incompatibles, lo hagan.

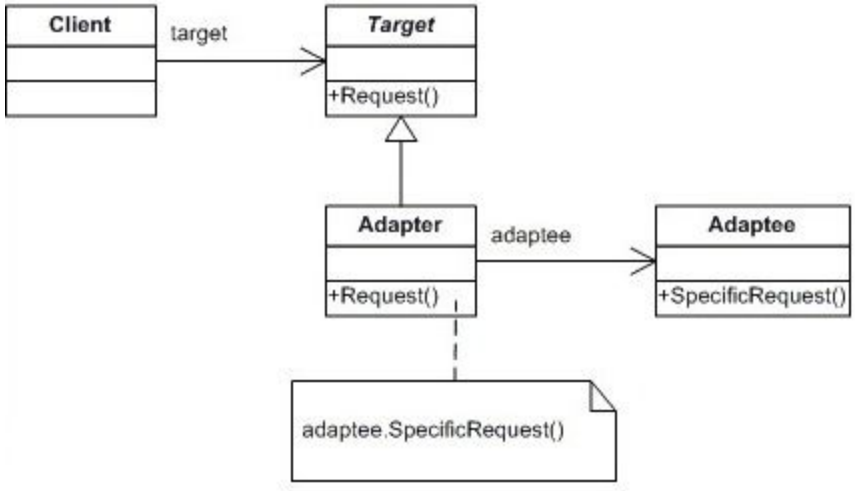
Otros nombres:
Wrapper, Adaptador, Envoltorio

Motivación:
En el mundo de hoy es habitual el uso de adaptadores (cargadores, tipos de enchufes, transformadores, etc.). En el mundo del software algunas veces un conjunto de clases no se puede reutilizar debido a que la interfaz no concuerda con el dominio que la aplicación requiere. Allí nace Adapter para facilitar esa reutilización y permite que no se deba modificar a gran escala el código.

Aplicabilidad:
Se debe usar cuando:

- Usar una clase existente que tenga una interfaz no compatible.
- El objeto adapter usa varias subclases para la adaptación.
- La clase adapter invalida algunos de los comportamientos de la adaptación.

Estructura:



Participantes:

- Objetivo: Define la interfaz específica que espera el Cliente.
- Cliente: Colabora con la conformación de objetos para la interfaz Objetivo.
- Inicial: Define una interfaz o clase existente que requiere adaptarse.
- Adaptador: Adapta la interfaz Inicial a la Objetivo.

Colaboraciones:

La clase Cliente llama a las operaciones sobre una instancia Adaptador. Luego, Adaptador llama a las operaciones de Inicial que llevan a cabo el pedido.

Ventajas:

- Una clase Adapter:
    - Usa el adaptador directamente llamando los métodos heredados.
    - Puede anular el comportamiento del adaptador.
- Un objeto Adapter:
    - Deja que un simple Adapter se relacione con muchos adaptadores.

Desventajas:

- Una clase Adapter no puede ser usada si las subclases de adaptación deben ser adaptadas también.
- Un Adapter de objetos no puede reemplazar el comportamiento de la adaptación.

Implementación:

- ¿Qué nivel de adaptación debe hacer el Adapter?: La cantidad de trabajo que el Adapter hace depende de la similitud de la clase Inicial y la clase Objetivo, ese trabajo puede ser simple como cambiar el nombre de las operaciones o puede encerrar todo un espectro de operaciones diferentes.
- Adaptadores Pluggables: La adaptación de la interfaz permite incorporar nuestra clase en sistemas existentes que pueden esperar diferentes interfaces para la clase.

Para implementarlo de forma correcta se aconseja crear una nueva clase que sea el Adapter, que extienda del componente existente e implemente la interfaz obligatoria.

Código de ejemplo:

- Cliente:

```
1  package estructurales.adapter.adapter02;
2
3  public class Main
4  {
5      public static void main(String[] args)
6      {
7          Adaptador conversor = new Adaptador();
8
9          conversor.ingresarPesetas( 2000 );
10         conversor.ingresarPesetas( 5000 );
11         conversor.ingresarPesetas( 1000 );
12
13         System.out.println( "Total euros: " + conversor.getTotalEuros() );
14     }
15 }
```

- Objetivo:

```
1  package estructurales.adapter.adapter02;
2   public interface IAdaptador
3  {
4      public abstract void sacarPesetas( double pesetas );
5      public abstract void ingresarPesetas( double pesetas );
6  }
```

- Adaptador:

```java
package estructurales.adapter.adapter02;
 public class Adaptador extends CajaEuros implements IAdaptador
{
    public Adaptador() {
    }
    // ------------------
    @Override
     public void sacarPesetas( double pesetas )
    {
        double euros = pesetas / 166.386;
         this.sacarEuros( euros );
    }
    // ------------------
    @Override
     public void ingresarPesetas( double pesetas )
    {
        double euros = pesetas / 166.386;
         this.ingresarEuros( euros );
    }
}
```

- Adaptable:

```java
package estructurales.adapter.adapter02;
public class CajaEuros
{
    private double euros = 0;
    // -----------------
    public CajaEuros() {
    }
    // -----------------
    public double getTotalEuros()
    {
        return this.euros;
    }
    // -----------------
     public void sacarEuros( double euros )
    {
        this.euros -= euros;
    }
    // -----------------
     public void ingresarEuros( double euros )
    {
        this.euros += euros;
    }
}
```

Usos conocidos:

- EventListener que le permite a un objeto ejecutar un evento.

- WindowsListener que extiende de EventListener

Patrones relacionados:

- Proxy
- Decorator

Bibliografía:

No específico. (No específico). GoF Design Patterns (Versión 2.1.0) [Aplicación móvil]. Descargado de: https://drive.google.com/file/d/0BywiVyFlIabXcVhGZlJBcnhWTkU/view.

Patrones de Diseño Software [Página web]. (s.f.). Ubicación https://informaticapc.com/patrones-de-diseno/adapter.php.

Junta de Andalucía. (s.f). Adaptador. Marco de Desarrollo de la Junta de Andalucía. http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/181.