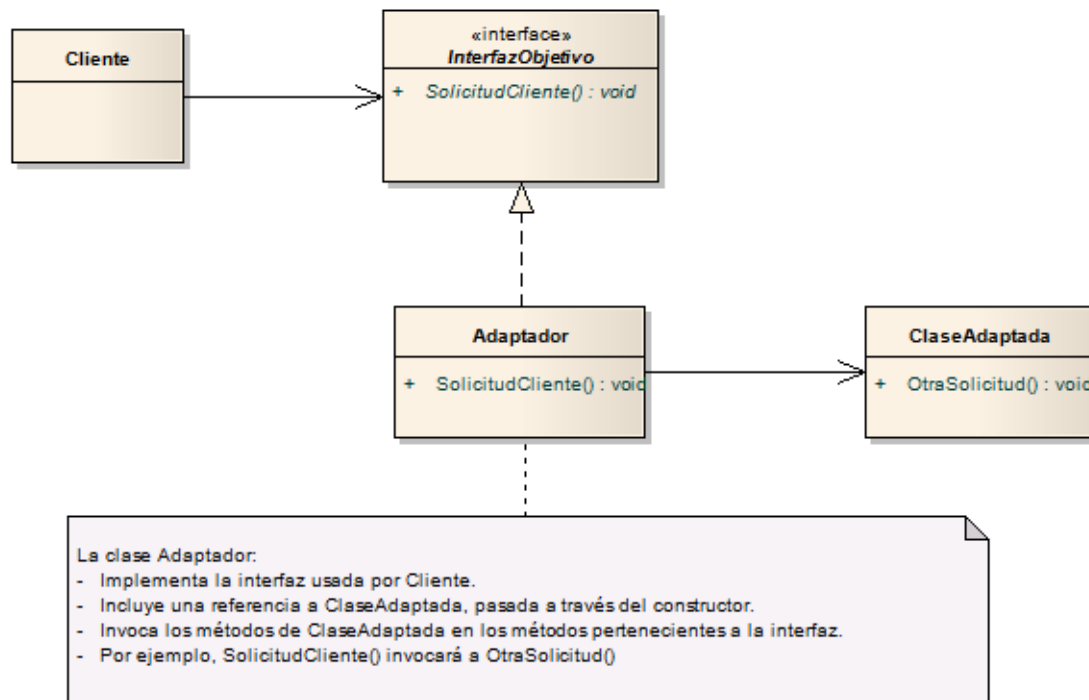


PATRÓN ADAPTER

Objetivo:

«Convertir la interfaz de una clase en otra interfaz que el cliente espera. Adapter consigue que trabajen juntas clases que de otro modo no podrían.»



El patrón *Adapter* es un patrón estructural. Los patrones estructurales se basan en la forma en la que un conjunto de clases se relacionan entre sí para proporcionar una funcionalidad compleja, proporcionando una estructura para conseguir lograr ese objetivo.

La filosofía de *Adapter*, es la de establecer una capa intermedia que permita comunicarse a dos clases que de otro modo no podrían hacerlo, realizando una *adaptación* de la interfaz de la clase que proporciona el servicio a la que la solicita.

Representándolo de una forma visual, imaginemos que tenemos electrodomésticos y otras herramientas que usan distintos voltajes en la electricidad, como 110 voltios, 220 voltios, etc. En el ejemplo un objeto que pertenece a la clase *Taladro*, que para poder funcionar necesita el método *Voltaje110V()* de una clase que implemente la interfaz *IEnchufeIngles*. Por tanto, el taladro está preparado para hacer uso de esa interfaz y espera recibir como valor de retorno dicho voltaje:



Sin embargo, resulta que la clase que tenemos en el otro extremo no implementa esta interfaz, sino otra llamada *IEnchufeEuropeo()* que dispone de un método capaz de devolver a nuestro taladro un voltaje de 220V, cuyo nombre es *Voltaje220V()*. La funcionalidad es parecida, pero no la esperada, y la interfaz que el taladro espera utilizar no es compatible con el elemento del subsistema. **Nos encontramos con un claro problema de compatibilidad.**



Para que el taladro pueda comunicarse con *IEnchufeEuropeo*, será necesario modificar uno de los dos extremos para que la interfaz de comunicación coincida, pero si hiciéramos esto, nuevamente estaríamos rompiendo el **principio abierto/cerrado** del que hablamos con anterioridad, amén de que cambiando esta interfaz haríamos que de repente dejaran de compilar todas aquellas clases que la estuvieran utilizando hasta el momento.



La solución adecuada consistirá en **adaptar** los requerimientos de la interfaz de la clase cliente (taladro) a los servicios que la clase servidora (sistema eléctrico europeo) puede ofrecer. Es decir... como indica el propio nombre del patrón, necesitaremos un **adaptador** que haga coincidir la entrada de un elemento con la salida del otro. Este será el objetivo del patrón *Adapter*.

Lo más adecuado será **codificar una nueva clase** que implemente la interfaz original (*IEnchufeIngles*) y que posea una propiedad cuyo tipo será el de **la clase que se pretende adaptar**. Es decir la clase adaptador contendrá una

referencia a una instancia de una clase que implemente *IEnchufeEuropeo*. De este modo, el taladro hará una llamada a la clase que espera (*IEnchufeIngles.Voltaje110V()*) y de forma interna se invocará el método *Voltaje220V()* mediante la instancia de *EnchufeEuropeo*.

Codificando un Adaptador

Un adaptador no sólo se encarga de transformar una interfaz en otra: también puede realizar otro tipo de operaciones, principalmente de transformación. En el ejemplo de los enchufes, además de transformar una interfaz en otra (los polos del enchufe), podría también realizar otras operaciones, como por ejemplo transformar la diferencia de potencial del voltaje de salida de 220 a 110, ya que de lo contrario correríamos el riesgo de quemar el dispositivo.

Comenzamos con las interfaces *IEnchufeIngles* e *IEnchufeEuropeo* y dos clases que las implementen, *EnchufeBritanico* y *EnchufeEuropeo*.

1. Ver las interfaces y clases antes mencionadas.

Ahora codificaremos la clase *Taladro*, que posee un enchufe inglés (es decir, posee una referencia a una interfaz *IEnchufeIngles*) para recibir la corriente eléctrica.

2. Ver la clase Taladro.

Lo ideal sería utilizar la clase *EnchufeBritanico*, que es lo que nuestro taladro espera...

3. Ver en el Program, la

// 1era parte, el taladro funciona a 110v entonces instanciamos el enchufe inglés.

... y vemos que obtenemos un voltaje adecuado, entre 103 y 117 voltios, aceptable para el taladro.

```
C:\Ort2021\Programacion 2\Patron ADAPTER\ProyectoPatronAdapter\PatronAdapter\bin\Debug\net5.0\PatronAdapter.exe
El taladro esta funcionando con voltaje de 113 Voltios
El taladro esta funcionando con voltaje de 115 Voltios
El taladro esta funcionando con voltaje de 109 Voltios
El taladro esta funcionando con voltaje de 109 Voltios
El taladro esta funcionando con voltaje de 104 Voltios
El taladro esta funcionando con voltaje de 108 Voltios
El taladro esta funcionando con voltaje de 109 Voltios
El taladro esta funcionando con voltaje de 109 Voltios
El taladro esta funcionando con voltaje de 114 Voltios
El taladro esta funcionando con voltaje de 107 Voltios
El taladro esta funcionando con voltaje de 109 Voltios
El taladro esta funcionando con voltaje de 108 Voltios
El taladro esta funcionando con voltaje de 113 Voltios
El taladro esta funcionando con voltaje de 104 Voltios
El taladro esta funcionando con voltaje de 106 Voltios
El taladro esta funcionando con voltaje de 103 Voltios
El taladro esta funcionando con voltaje de 112 Voltios
El taladro esta funcionando con voltaje de 105 Voltios
El taladro esta funcionando con voltaje de 109 Voltios
El taladro esta funcionando con voltaje de 108 Voltios
El taladro esta funcionando con voltaje de 116 Voltios
El taladro esta funcionando con voltaje de 104 Voltios
El taladro esta funcionando con voltaje de 108 Voltios
El taladro esta funcionando con voltaje de 109 Voltios
El taladro esta funcionando con voltaje de 106 Voltios
El taladro esta funcionando con voltaje de 111 Voltios
El taladro esta funcionando con voltaje de 113 Voltios
El taladro esta funcionando con voltaje de 107 Voltios
El taladro esta funcionando con voltaje de 106 Voltios
```

Sin embargo, recordemos que el patrón *Adapter* no tiene sentido en este escenario, sino cuando el objeto del que disponemos no cumple con la interfaz esperada, es decir: en nuestro sistema, la clase **EnchufeBritanico** no existe. La clase del subsistema al que necesitamos conectarnos no implementa la interfaz *IEnchufeBritanico*, sino que se trata de un objeto de la clase **EnchufeEuropeo**. La situación, por lo tanto, es esta:

4. Ver en el Program, la

// 2da parte, intentamos usar el taladro de 110v con 220v.

COMENTE el código de la 1era parte!!!!

```
// 2da parte, intentamos usar el taladro de 110v con 220v
```

```
// Instanciamos enchufe y taladro
```

```
IEnchufeEuropeo enchufe = new EnchufeEuropeo();
```

```
Taladro taladro = new Taladro(enchufe);
```

```
// Encendemos el taladro
```

```
taladro.Encender();
```

(variable local) IEnchufeEuropeo enchufe

CS1503: Argumento 1: no se puede convertir de 'PatronAdapter.IEnchufeEuropeo' a 'PatronAdapter.IEnchufeIngles'

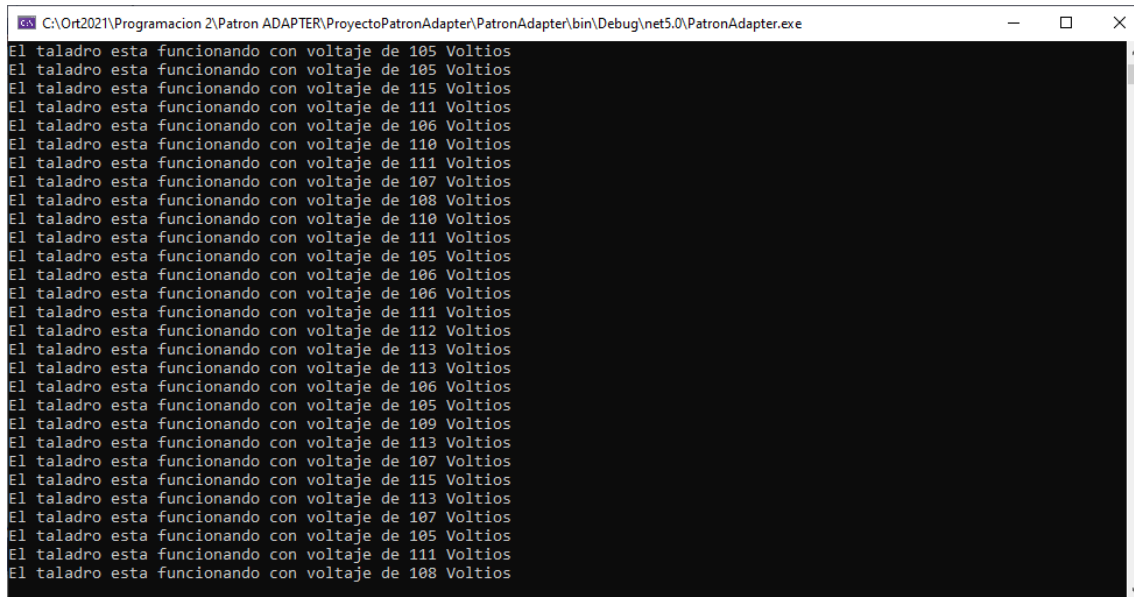
[Mostrar posibles correcciones](#) (Alt+Entrar o Ctrl+.)

Por lo tanto, crearemos una clase adaptador que cumpla el objetivo de “adaptar” el voltaje, y que:

- Implemente la interfaz *IEnchufeIngles* para ser compatible con la clase *Taladro*.
- Incluya una referencia a un *IEnchufeEuropeo*
- Implemente los métodos *IEnchufeIngles* invocando los métodos adecuados de *IEnchufeEuropeo* y realizando las operaciones de transformación adecuadas.

5. Ver el código de la clase AdapterInglesEuropeo.cs

Y en el main, ver el código de la // 3era parte, comentando las dos partes anteriores



```
C:\Ort2021\Programacion 2\Patron ADAPTER\ProyectoPatronAdapter\PatronAdapter\bin\Debug\net5.0\PatronAdapter.exe
El taladro esta funcionando con voltaje de 105 Voltios
El taladro esta funcionando con voltaje de 105 Voltios
El taladro esta funcionando con voltaje de 115 Voltios
El taladro esta funcionando con voltaje de 111 Voltios
El taladro esta funcionando con voltaje de 106 Voltios
El taladro esta funcionando con voltaje de 110 Voltios
El taladro esta funcionando con voltaje de 111 Voltios
El taladro esta funcionando con voltaje de 107 Voltios
El taladro esta funcionando con voltaje de 108 Voltios
El taladro esta funcionando con voltaje de 110 Voltios
El taladro esta funcionando con voltaje de 111 Voltios
El taladro esta funcionando con voltaje de 105 Voltios
El taladro esta funcionando con voltaje de 106 Voltios
El taladro esta funcionando con voltaje de 106 Voltios
El taladro esta funcionando con voltaje de 111 Voltios
El taladro esta funcionando con voltaje de 112 Voltios
El taladro esta funcionando con voltaje de 113 Voltios
El taladro esta funcionando con voltaje de 113 Voltios
El taladro esta funcionando con voltaje de 106 Voltios
El taladro esta funcionando con voltaje de 105 Voltios
El taladro esta funcionando con voltaje de 109 Voltios
El taladro esta funcionando con voltaje de 113 Voltios
El taladro esta funcionando con voltaje de 107 Voltios
El taladro esta funcionando con voltaje de 115 Voltios
El taladro esta funcionando con voltaje de 113 Voltios
El taladro esta funcionando con voltaje de 107 Voltios
El taladro esta funcionando con voltaje de 105 Voltios
El taladro esta funcionando con voltaje de 111 Voltios
El taladro esta funcionando con voltaje de 108 Voltios
```

Como detalle, debemos fijarnos en que tanto la clase cliente como la adaptada ignoran completamente su mutua existencia. Ambas clases están desacopladas por medio de esta clase *Adaptador*, por lo que por norma general es un patrón que sirve como ejemplo perfecto de *buenas prácticas de programación*.

¿Cuándo utilizar este patrón? Ejemplos reales

El escenario de este patrón es también bastante claro. Se utilizará en los casos en los que sea necesario utilizar una clase cuya interfaz no se adapte a los requerimientos de otra clase cliente. Podemos ver algunos ejemplos reales cuando realizamos transformaciones entre enumeraciones e iteradores, arrays y listas, etc.

El lenguaje Java, por ejemplo, ofrece los siguientes usos del patrón *Adapter*:

- [`java.util.Arrays#asList\(\)`](#)
- [`java.io.InputStreamReader\(InputStream\)`](#) (devuelve un Reader)
- [`java.io.OutputStreamWriter\(OutputStream\)`](#) (devuelve un Writer)
- [`javax.xml.bind.annotation.adapters.XmlAdapter#marshal\(\)`](#)
- [`javax.xml.bind.annotation.adapters.XmlAdapter#unmarshal\(\)`](#)