

## Taller 5 Patrones de diseño



- 1) [Igenesius/vending-machine-order-system: A project for the midterm exam in the Framework Layer Architecture course \(COMP6122\). This project is about a menu system for ordering vending machines. \(github.com\)](https://github.com/Igenesius/vending-machine-order-system)

El proyecto creado por Igenesius, recrea una maquina expendedora, tiene la particularidad que recibe tanto yenes como dólares, Ante tal particularidad se plantean dos grandes interrogantes ¿Cómo la maquina se da cuenta que moneda le llega? ¿cómo alterna entre el tipo de monedas?

El creador de este repositorio utiliza el patrón de diseño “Adapter”, que para estos casos funciona muy bien, ya que toma una interfaz no compatible e inutilizable para un tipo de objeto, en una clase comúnmente llama adapter, hace el cambio para que el objeto sea compatible con la interfaz con la que se quiere trabajar, de ahí su nombre, que en español seria “adaptador”. Así la maquina podrá recibir objetos tanto yenes como dolares sin dificultad.

```
1 package adapter;
2
3 import adapter.usdollar.USDollar;
4 import adapter.yen.Yen;
5
6 public class Currency {
7     public Currency() {
8
9     }
10
11     public double convertToUSDollar(USDollar dollar, int price) {
12         return dollar.showInDollar(price);
13     }
14
15     public double convertToYen(Yen yen, int price) {
16         return yen.showInYen(price);
17     }
18 }
```

Clase que ejecuta el “cambio”

 USDollar.java	add vendingMachineProducer folder	2 years ago
 USDollarCurrency.java	add vendingMachineProducer folder	2 years ago

En la carpeta “dollar” existen dos archivos, una que es la interfaz “USDollar”, y una clase que implementa esa interfaz y la utiliza para “darle valor”.  
Y ocurre exactamente lo mismo con los Yenes.

```
00 {  
    System.out.print("Input your currency [IDR|USD|YEN]: ");  
    currency = scan.nextLine();  
  
    if(currency.equals("IDR") || currency.equals("USD") || currency.equals("YEN")) {  
        if(currency.equals("USD")) {  
  
        }  
        else if(currency.equals("YEN")) {  
  
        }  
        currencyValidation = true;  
    }  
} while(currencyValidation == false);
```

Y dentro de el if anidado y del else if, se incorporarían los métodos de la clase “unificadora” como getprize() en este caso.

Las ventajas que tiene utilizar este patrón en este proyecto en específico son varias, ya que permite a la clase “currency” alojar cualquier tipo de moneda ya sea Yenes o Dólares, y cada moneda con su implementación diferente. Este es el sentido del patrón “Adapter”, servir como puente entre dos interfaces no compatibles entre si.

Una forma en la que se podría abordar este problema es cambiar toda la implementación de las interfaces de yenes y dólares, al tratarse de dos interfaces pequeñas sería razonable este cambio, pero en dado caso que muchas mas clases implementen esta interfaz sería muy tedioso cambiar una a una.

Una de las limitaciones que tiene este patrón podría ser la complejidad que tiene cada clase tipo adapter, es decir si se quiere asociar una interfaz no propia de un objeto con otra, se tiene que pensar en las compatibilidades de los datos que requiere esta clase. Puede que se tenga que crear mas de una clase “adapter”

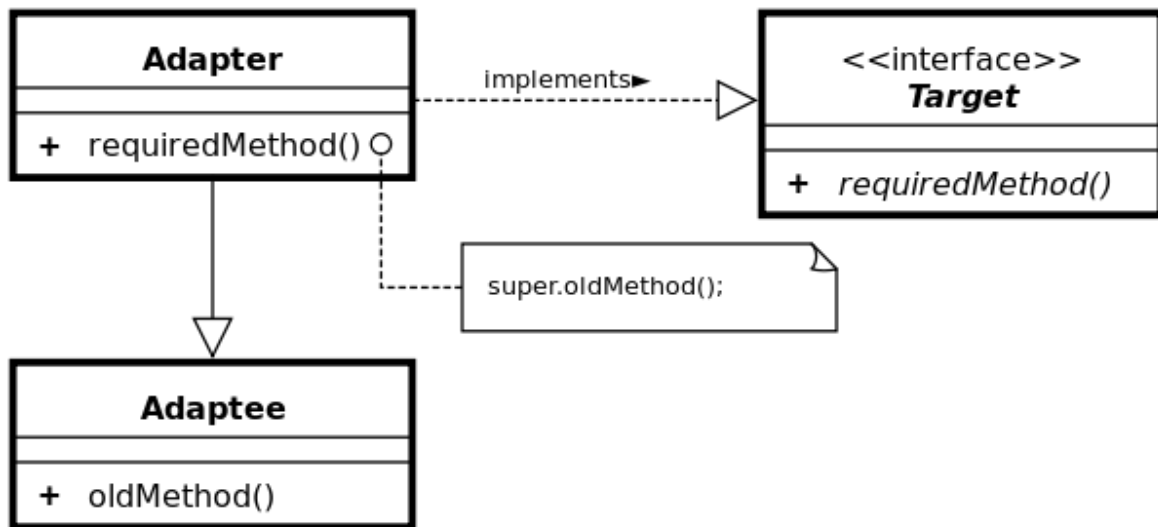


Imagen tomada de [Adaptador \(patrón de diseño\) - Wikipedia, la enciclopedia libre](#)

Esta imagen explica bastante bien el uso del patron adapter, así como sus componentes.

- Clase adapter
- Una interfaz no compatible con un objeto A
- Un objeto A