

## **Fachada**

**Nombre:**

Fachada

**Clasificación del patrón:**

Estructural

**Intención:**

Provee una interfaz unificada para un conjunto de interfaces en un subsistema. Fachada define un alto nivel de interfaces que hace el subsistema fácil de usar.

**Otros nombres:**

Facade

**Motivación:**

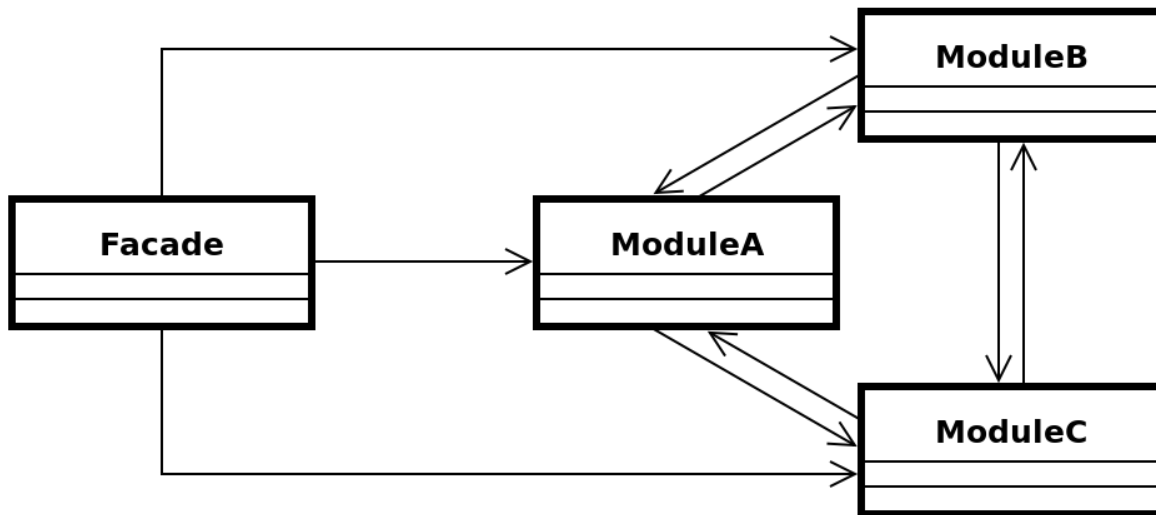
Uno de los objetivos comunes en cuanto al desarrollo de software consiste en reducir y minimizar las comunicaciones entre subsistemas (acoplamiento), una forma de conseguirlo es precisamente usando Fachada para suministrar una interfaz simple ampliando la facilidad del sistema.

**Aplicabilidad:**

Se debe usar cuando:

- Se quiere avituallar una interfaz simple a un sistema complejo.
- Se busca hacer el sistema más fácil para nuevos usuarios, pero viejos usuarios quieran mantener su aplicación, se da un poderoso acceso al subsistema.
- Se piensa abastecer de nuevas y poderosas interfaces a diferentes niveles de usuario.
- Se reducen en gran medida las dependencias entre clientes y las clases que usan.

## Estructura:



## Participantes:

- Fachada: Delegan la petición del cliente a sus respectivos objetos en los subsistemas.
- Clase de subsistemas: Implementan la funcionalidad del sistema.

## Colaboraciones:

- Los Clientes se comunican con el subsistema por medio de la Fachada que envía las peticiones a los objetos encargados.
- Los clientes que usan la Fachada no necesitan tener acceso directo a cada objeto con el que interactúan.

## Ventajas:

- Aisla clases de las clases más complejas. Lo cual reduce el número de objetos con los que el cliente interactúa y también reduce la dependencia en la compilación de sistemas grandes.
- Existe un débil acoplamiento entre las clases del cliente y las clases del subsistema, porque las clases del subsistema pueden cambiar sin afectar las clases de los clientes.
- La Fachada no impide que los clientes usen las clases del subsistema.
- Para funciones de bajo nivel y estructuras de datos, se proporciona una interfaz orientada a objetos reduciendo los errores de programación.

### Desventajas:

- El uso excesivo del patrón nos guía a un sistema poco funcional, con muchas capas y errores del funcionamiento.

### Implementación:

- Se puede hacer una reducción extra del acoplamiento haciendo a la Fachada una clase abstracta que permita escoger entre diferentes implementaciones del subsistema.
- En Java se facilita la definición de clases privadas a un subsistema.

### Código de ejemplo:

- Retirada de efectivo de un cajero automático.

```
1 public class Autenticacion{
2     /* ... */
3     public boolean leerTarjeta(){}
4     public String introducirClave(){}
5     public boolean comprobarClave(String clave){}
6     public Cuenta obtenerCuenta(){}
7     public void alFallar(){}
8 }
9
10 public class Cajero{
11     /* ... */
12     public int introducirCantidad(){}
13     public boolean tieneSaldo(int cantidad){}
14     public int expedirDinero{}
15     public String imprimirTicket(){}
16 }
17
18 public class Cuenta{
19     /* ... */
20     public double comprobarSaldoDisponible(){}
21     public boolean bloquearCuenta(){}
22     public boolean desbloquearCuenta{}
23     public void retirarSaldo(int cantidad){}
24     public boolean actualizarCuenta(){}
25     public void alFallar(){}
26 }
```

- Fachada:

```

1 public class FachadaCajero{
2     private Autenticacion autenticacion = new Autenticacion();
3     private Cajero cajero = new Cajero();
4     private Cuenta cuenta = null;
5
6     public void introducirCredenciales(){
7         boolean tarjeta_correcta = autenticacion.leerTarjeta();
8         if(tarjeta_correcta){
9             String clave = autenticacion.introducirClave();
10            boolean clave_correcta = autenticacion.comprobarClave(clave);
11            if(clave_correcta){
12                cuenta = autenticacion.obtenerCuenta();
13                return;
14            }
15        }
16        autenticacion.alFallar();
17    }
18
19    public void sacarDinero(){
20        if(cuenta != null){
21            int cantidad = cajero.introducirCantidad();
22            int tiene_dinero = cajero.tieneSaldo(cantidad);
23            if(tiene_dinero){
24                boolean hay_saldo_suficiente = ((int)cuenta.comprobarSaldoDisponible()) >= cantidad;
25                if(hay_saldo_suficiente){
26                    cuenta.bloquearCuenta();
27                    cuenta.retirarSaldo(cantidad);
28                    cuenta.actualizarCuenta();
29                    cuenta.desbloquearCuenta();
30
31                    cajero.expedirDinero();
32                    cajero.imprimirTicket();
33                }
34                else{
35                    cuenta.alFallar();
36                }
37            }
38        }
39    }
40 }

```

- Clase principal:

```

1 public static void main(String[] args){
2
3     FachadaCajero cajero_automatiko = new FachadaCajero();
4
5
6
7     cajero_automatiko.introducirCredenciales();
8
9     cajero_automatiko.sacarDinero();
10
11 }

```

### Usos conocidos:

- Varias librerías de Software, como Font y Graphics en el caso de Java.

### Patrones relacionados:

- Singleton

## **Bibliografía:**

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

Patrones de Diseño (XI): Patrones Estructurales - Facade [Página web]. (s.f.). Ubicación  
[https://programacion.net/articulo/patrones\\_de\\_diseno\\_xi\\_patrones\\_estructurales\\_facade\\_1014](https://programacion.net/articulo/patrones_de_diseno_xi_patrones_estructurales_facade_1014).

Sánchez, Giovanni. Patrones. (Mayo de 2009). Facade. Consultado en:  
<http://giovanni-sanchez.blogspot.com/2009/05/facade.html>.

Junta de Andalucía. (s.f). Fachada. Marco de Desarrollo de la Junta de Andalucía.  
<http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/189>.