Ejemplo de patrón facade

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

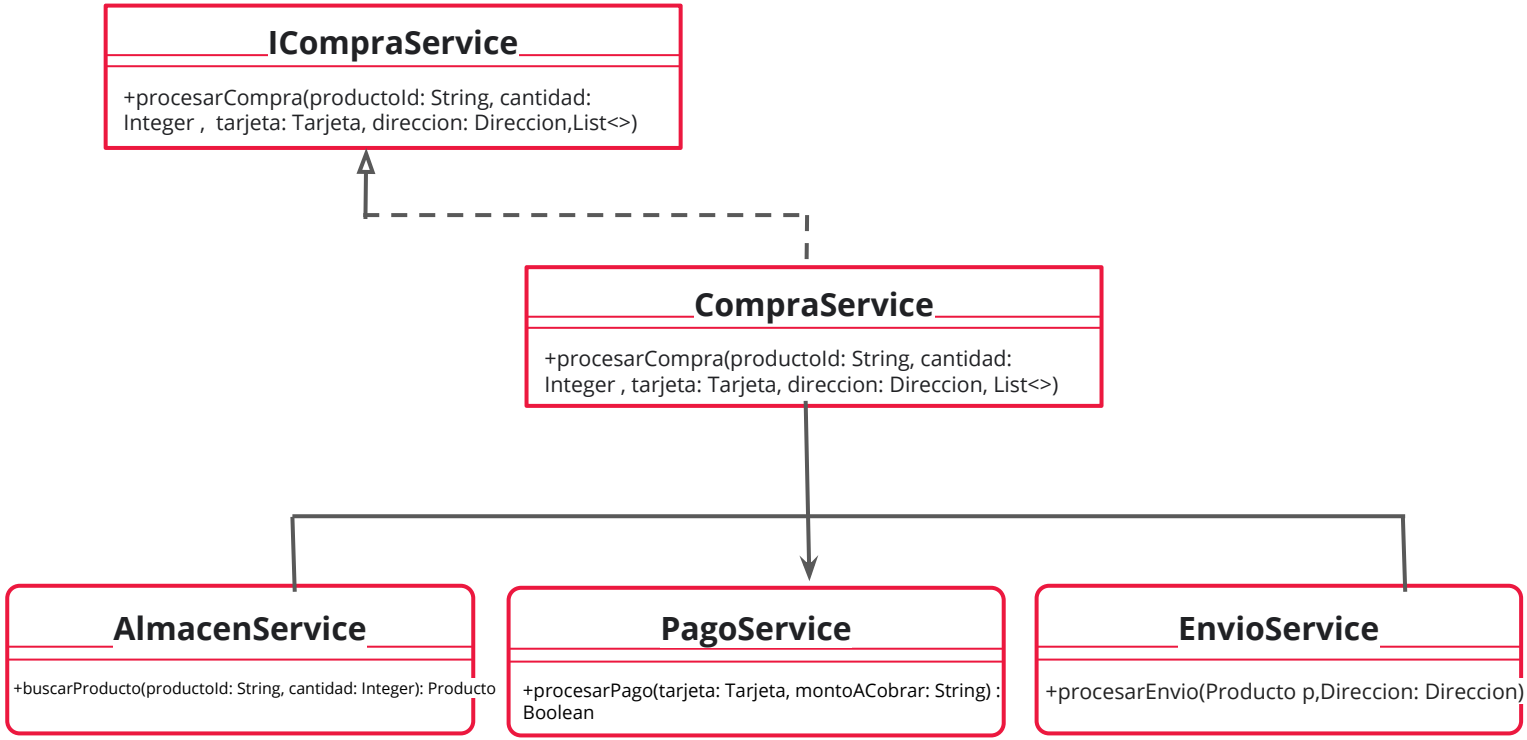
Ejemplo de patrón facade

Ahora imaginemos una implementación...

Supongamos que tenemos que diseñar un sistema para un e-commerce. Nuestro cliente nos pide que al momento de efectuar la venta del producto, nuestro sistema debería realizar una serie de pasos, por ejemplo: pedir el producto al almacén, acreditar el pago y enviar el pedido.

Veamos cómo podemos resolver este problema aplicando este patrón.

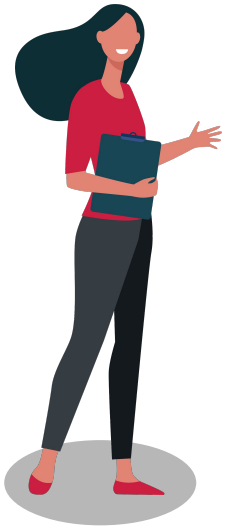
Solución


ICompraService
+procesarCompra(productoId: String, cantidad: Integer , tarjeta: Tarjeta, direccion: Direccion,List<>)
CompraService
+procesarCompra(productoId: String, cantidad: Integer , tarjeta: Tarjeta, direccion: Direccion, List<>)
AlmacenService
+buscarProducto(productoId: String, cantidad: Integer): Producto
PagoService
+procesarPago(tarjeta: Tarjeta, montoACobrar: String) : Boolean
EnvioService
+procesarEnvio(Producto p,Direccion: Direccion)

Implementación de las clases del diagrama UML

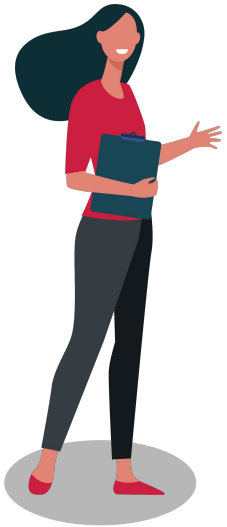
```
public interface ICompraService {

   public void procesarCompra(String productoId,
      Integer cantidad, Tarjeta tarjeta,
      Direccion direccion, List<Producto> productos);
}
```

Código de la interface **ICompraService**



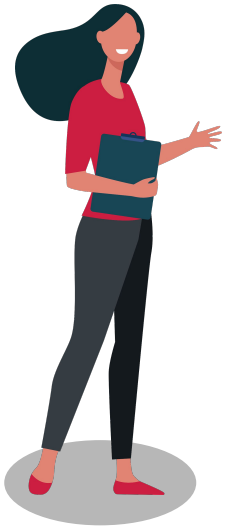
```
public class CompraService implements ICompraService {

    private AlmacenService almacenService;
    private PagoService pagoService;
    private EnvioService envioService;

    public CompraService() {

        this.almacenService = new AlmacenService();
        this.pagoService = new PagoService();
        this.envioService = new EnvioService();

    }
```

Código de la clase **CompraService (nuestro facade)**



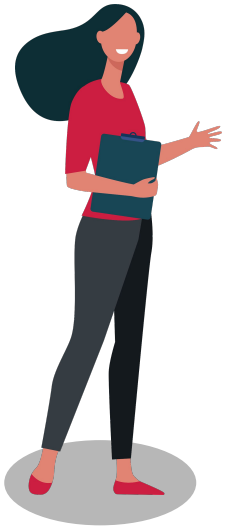
```
public void procesarCompra(String productoId,
                Integer cantidad, Tarjeta tarjeta,
                Direccion direccion, List<Producto>
                productos) {

   Producto prod;
   almacenService.setProductos(productos);
   prod = almacenService.buscarProducto(productoId,cantidad);
   if(prod != null){
      double montoCobrar = producto.getValor() * cantidad;
       if(pagoService.procesarPago(tarjeta,montoCobrar)){
          envioService.procesarEnvio(producto,direccion);
        }
     }
 }
}
```

Código de la clase **CompraService (nuestro facade)**



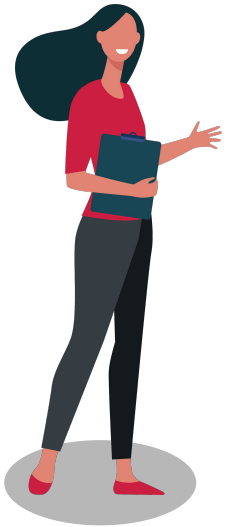
```java
public class AlmacenService {
   private List<Producto> productos;

   public void setProductos(List<Producto> productos) {
       this.productos = productos;
  }
```

Código de la clase **AlmacenService (subsistema)**



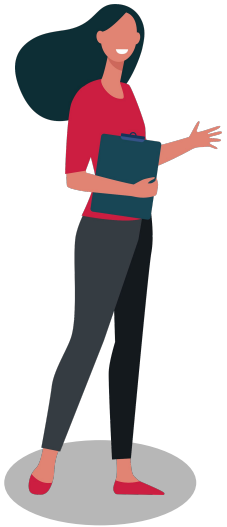
```
public Producto buscarProducto(String productoId,
                               Integer cantidad) {
    Producto producto = null;

    for (Producto p : this.productos) {
       if (p.getProductoId().equals(productoId) &&
                p.getCantidad() >= cantidad){

            producto = p;
            p.setCantidad(p.getCantidad() - cantidad);
            producto.setCantidad(cantidad);
        }
     }
     return producto;
 }
```

Código de la clase **AlmacenService (subsistema)**



```
Public class PagoService {

   public Boolean procesarPago(Tarjeta tarjeta,double montoACobrar){
       Boolean pagoRealizado = Boolean.FALSE;
       if(tarjeta != null && tarjeta.getNumerosFrente() != null &&
                           tarjeta.getCodSeguridad() != null)

           System.out.println("Procesando el pago por "+
                                   montoACobrar);
            pagoRealizado = TRUE;

       return pagoRealizado;
   }

}
```

Código de la clase **PagoService (subsistema)**



```java
public class EnvioService {

   public void procesarEnvio(Producto producto,
                             Direccion direccion){

       System.out.println("Enviando producto a " +
             direccion.getCalle() +" "+ direccion.getNro() +","+
             direccion.getBarrio());
   }
}
```

Código de la clase **EnvioService (subsistema)**

Simulamos un caso en el método main para ponerlo a prueba.

```
public class Prueba {
   public static void main(String[] args) {
       List<Producto> productos = new ArrayList<>();
       Producto productoUno = new Producto("1",5,1000,"Mouse");
       Producto productoDos = new Producto("2",5,3000,"Teclado");
       productos.add(productoUno);
       productos.add(productoDos);
       Tarjeta tarjeta  = new Tarjeta("1111222233334444","012","2025/07/09");
       Direccion direccion = new Direccion("Av Monroe","860","1428","CABA","Capital federal");

       ICompraService compraService = new CompraService();
       compraService.procesarCompra("1",2,tarjeta,direccion, productos);
   }
}
```

Conclusión

Mediante una **interface** definimos cómo el cliente se deberá comunicar con nuestro sistema, definimos la clase **CompraService** que actuará de **fachada**, recibiendo las peticiones y comunicándose con los subsistemas para que en conjunto, se complete la compra.


DigitalHouse>
Coding School