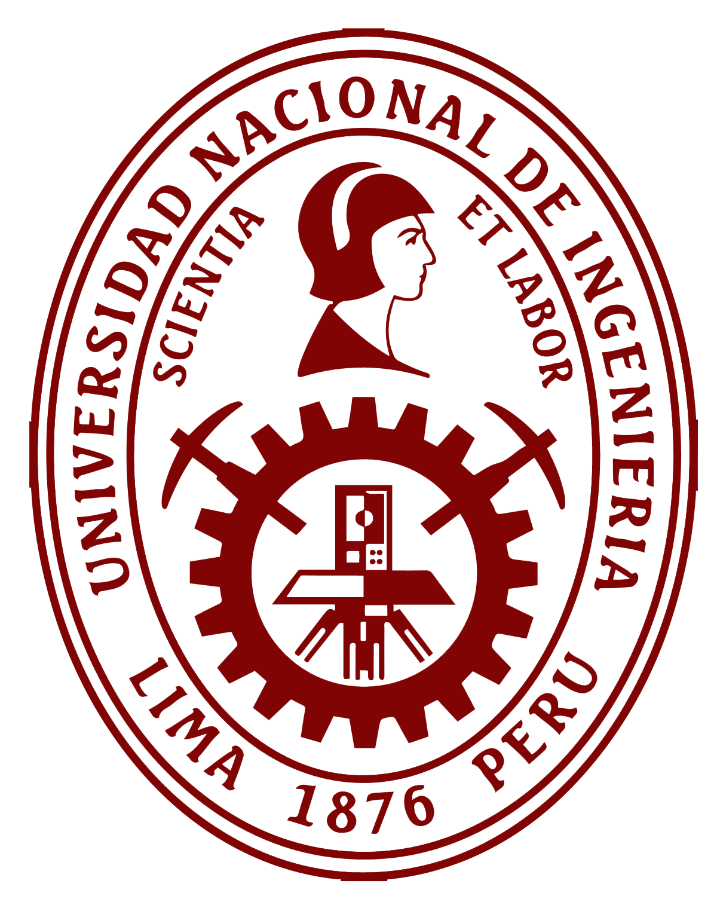
**UNIVERSIDAD NACIONAL DE INGENIERÍA**

**TRABAJO DE INVESTIGACIÓN**

**PATRÓN FACADE**

**ESTUDIANTE: ALEXANDER TORRES MAÑUECO**

**DOCENTE: GUSTAVO CORONEL**

**LIMA-PERÚ**

**2020**

**Dedicatoria**

A mi madre con mucho amor y cariño le dedico todo mi esfuerzo y trabajo puesto parala realización de este trabajo.

**Índice**

Contenido

[1.Definicion 4](#_Toc32955630)

[2.Componentes 4](#_Toc32955631)

[3.Diagrama de secuencia 5](#_Toc32955632)

[4.Caso Desarrollado 7](#_Toc32955633)

[5.Conclusiones 9](#_Toc32955634)

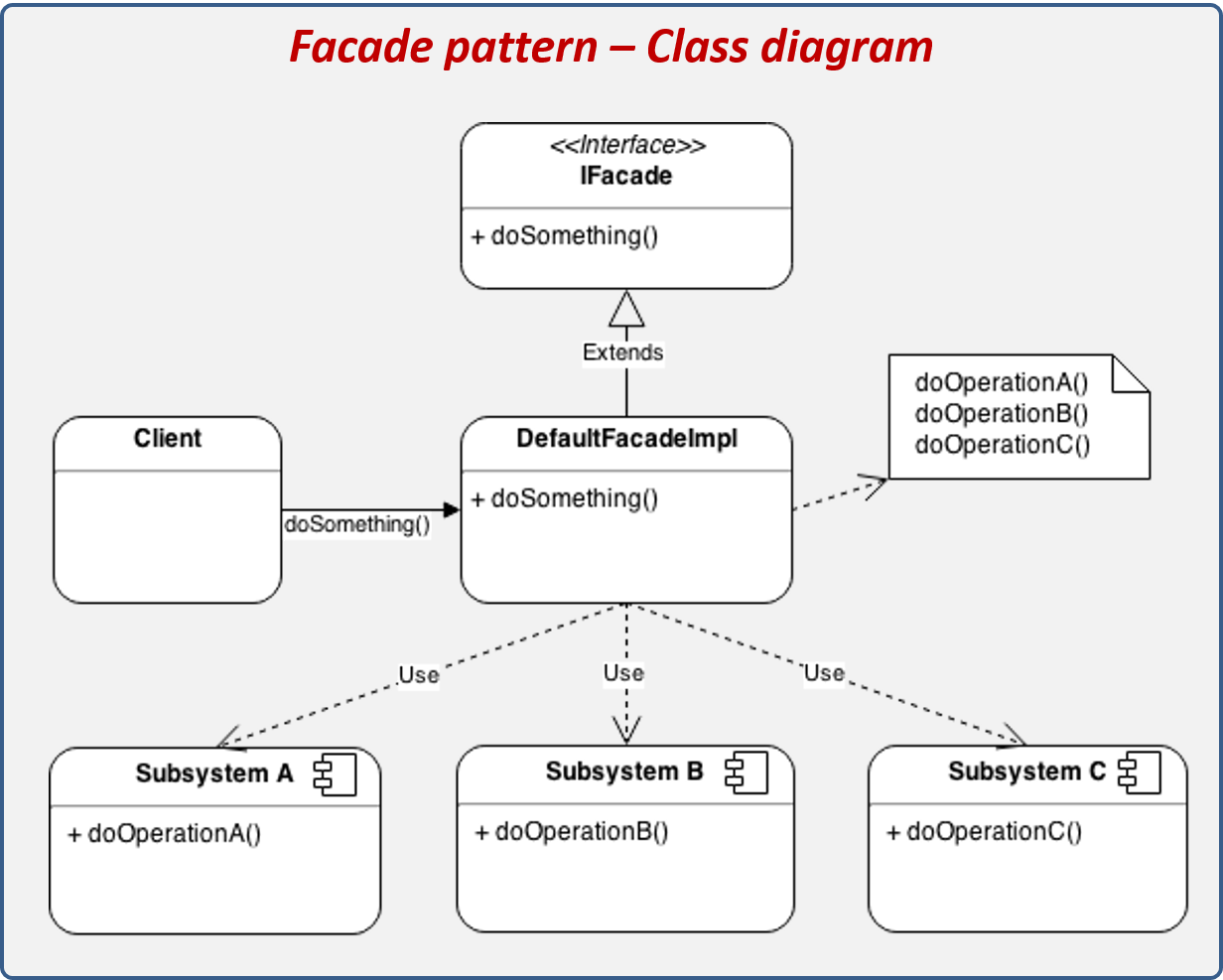
[6.Recomendaciones 9](#_Toc32955635)

# 1.Definicion

**Fachada** (*Facade*) es un tipo de [patrón de diseño](https://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o) estructural. tiene la característica de ocultar la complejidad de interactuar con un conjunto de subsistemas proporcionando una interface de alto nivel, la cual se encarga de realizar la comunicación con todos los subsistemas necesarios. La fachada es una buena estrategia cuando requerimos interactuar con varios subsistemas para realizar una operación concreta ya que se necesita tener el conocimiento técnico y funcional para saber qué operaciones de cada subsistema tenemos que ejecutar y en qué orden, lo que puede resultar muy complicado cuando los sistemas empiezan a crecer demasiado. Otro escenario proclive para su aplicación surge de la necesidad de desacoplar un sistema de sus clientes y de otros subsistemas, haciéndolo más independiente, portable y reutilizable (esto es, reduciendo dependencias entre los subsistemas y los clientes).

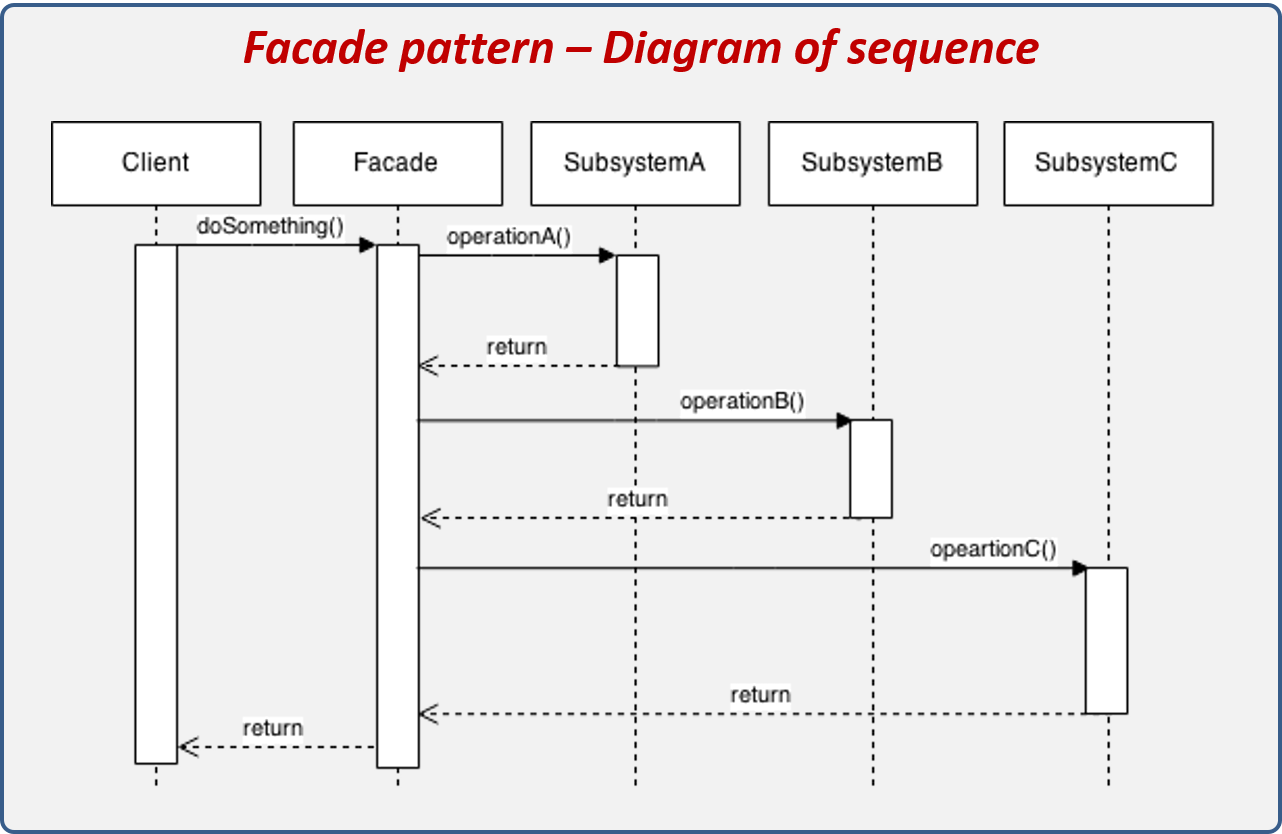
# 2.Componentes

* **IFacade (Interfaz):**Proporciona una interface de alto nivel que oculta la complejidad de interactuar con varios sistemas para realizar una operación.
* **Client (Cliente):**Sistema o evento que interactúa con la fachada.
* **DefaultFacadeImpl (Fachada):**Representa la implementación de IFacade y se encarga de comunicarse con todos los subsistemas.
* **Subsystems (Subsistema):**Representa módulos o subsistemas que exponen interfaces para comunicarse con ellos.



# 3.Diagrama de secuencia

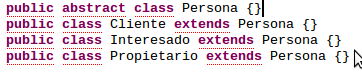
1. El ***cliente***invoca una operación de la *fachada*.
2. La ***fachada***se comunica con el ***SubsistemaA***para realizar una operación.
3. La ***fachada***se comunica con el ***SubsistemaB***para realizar una operación.
4. La ***fachada***se comunica con el ***SubsistemaC***para realizar una operación.
5. La **fachada** responde al **cliente** con el resultado de la operación.



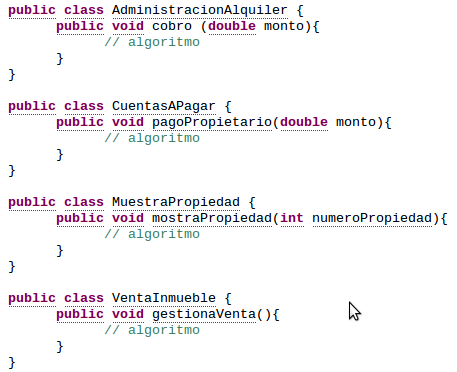
# 4.Caso Desarrollado

Un equipo de desarrolla esta realizando un software para una inmobiliaria. La inmobiliaria realiza muchos trabajos diferentes como cobro de alquiler, muestra de inmuebles, administración de consorcios, etc.

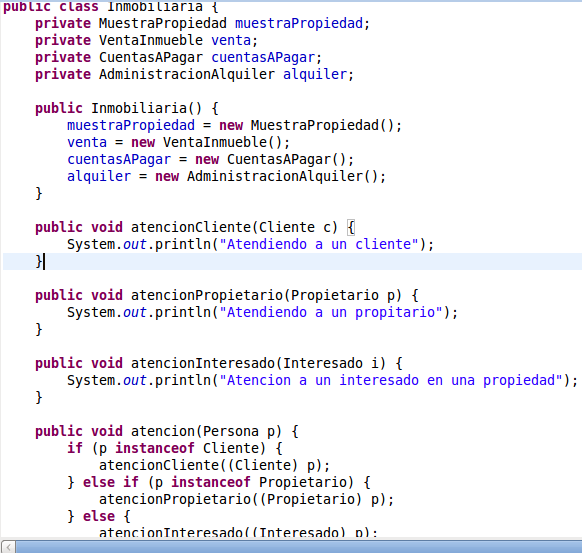
Por una cuestión de paradigma de programación orientada a objetos, no todo se realiza en una misma clase, sino que se dividen las responsabilidades en diferentes clases.



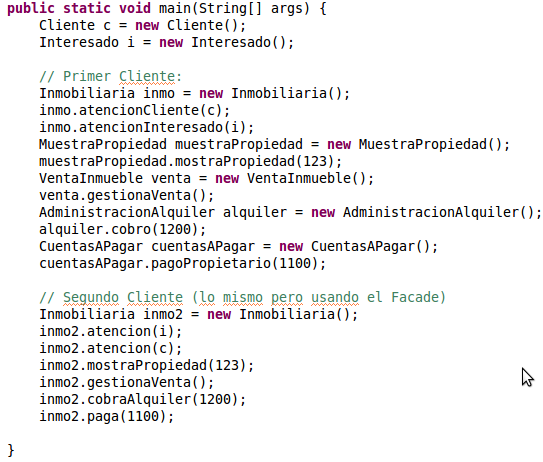
Lo mismo con los métodos principales de las diversas clases que tiene el sistema.



Ahora hacemos una clase llamada Inmobiliaria llamada Facade.



Al final tenemos dos tipos de clientes. El primero no llama al Facade y el segundo si lo utilizara. Veremos como el primero esta obligado a conocer muchos detalles de los subsistemas, mientras el segundo no.



# 5.Conclusiones

Es un patrón de arquitectura muy útil, ya que consigue que los clientes realizan acciones u operaciones sin hacer uso de los subsistemas, ocultándoles y haciéndolo fácil de usar. La interfaz es de gran ayuda ya que ayuda a que el cliente entienda de una manera más simple y rápida.

# 6.Recomendaciones

No se deben construir muchos objetos facade. Solo los más representativos que contenga la mayoría de operaciones básicas del sistema