

**FACULTAD DE INGENIERIA INDUSTRIAL Y DE SISTEMAS**

**FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA Y ELECTRONICA**

**ALUMNOS: CHÁVEZ DÍAZ WILLIAM ELISBAN**

**VALDIVIA VELASQUEZ JHAN CARLOS**

**JAVA ORIENTADA A OBJETOS**

**PROFESOR: ERIC GUSTAVO CORONEL CASTILLO**

**UNIVERSIDAD NACIONAL DE INGENIERIA**

**UNIVERSIDAD TECNOLOGICA DEL PERÚ**

**RIMAC, LIMA – PERÚ 2020**

**DEDICATORIA**

A nuestros padres por confiar

y apoyar constantemente

nuestra decisión por convertirnos en grandes

profesionales en Ingeniería de Sistemas e Informática.

**INDICE**

DEDICATORIA…………………………………………………………………………....…1

INDICE………………………………………………………………………………….…….2

PATRON DE DISEÑO DE SOFTWARE-FACADE………………………………………3

PARTES DEL PATRON………………………………………………………………….3-4

APLICACIÓN DEL PATRON……………………………………………………………....4

CONSECUENCIA DEL USO……………………………………………………...……..4-5

EJEMPLO…………………………………………………………………………………….5

CONCLUSIÓN……………………………………………………………………………….6

RECOMENDACIONES……………………………………………………………………..6

BIBLIOGRAFIA……………………………………………………………………………6-7

**PATRÓN DE DISEÑO DE SOFTWARE - FACADE**

Es el patrón que busca simplificar el sistema, proporcionando una interfaz unificada para un conjunto de subsistemas, definiendo una interfaz de nivel más alto. Esto hace que el sistema sea más fácil de usar.

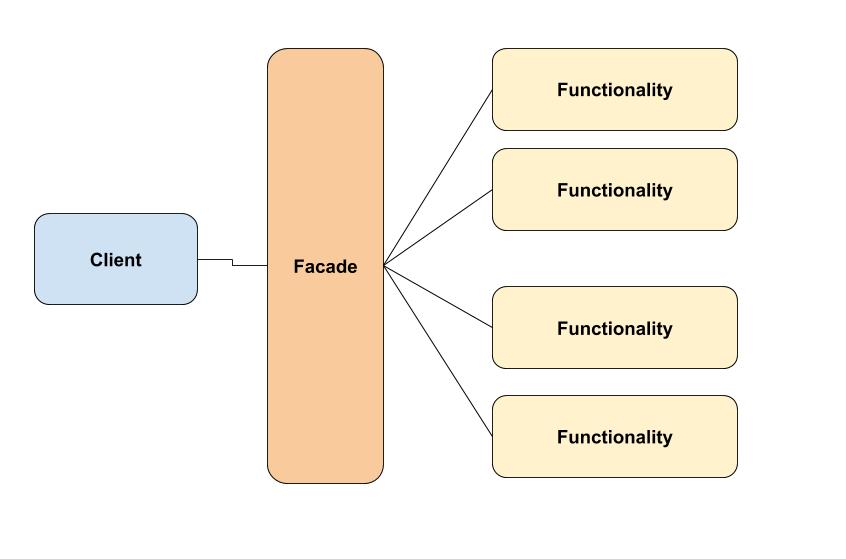
Este patrón busca reducir al mínimo la comunicación y dependencias entre subsistemas. Para ello, utilizaremos una fachada, simplificando la complejidad al cliente. El cliente debería acceder a un subsistema a través del Facade. De esta manera, se estructura un entorno de programación más sencillo, al menos desde el punto de vista del cliente (por ello se llama "fachada").

**PARTES DEL PATRÓN**

Facade dispone una interfaz simple y accesible a otros sistemas o subsistemas. De este modo simplifica la complejidad a los clientes externos exponiendo una interfaz más clara y un acceso unificado a estas funcionalidades haciendo más fácil su uso.

Tenemos entonces en este patrón tres partes:

* El Cliente que accede al Facade.
* El Facade que accede al resto de funcionalidades y las unifica o simplifica
* El resto de las funcionalidades / subsistemas, que están “atrás” del Facade.



**APLICACIN DEL PATRON**

* Cliente tiene que acceder a parte de la funcionalidad de un sistema complejo.
* Hay tareas o configuraciones muy frecuentes y es conveniente simplificar el código de uso.
* Necesitamos hacer que una librería sea más legible.
* Nuestros sistemas clientes tienen que acceder a varias APIs y queremos simplificar dicho acceso.

**CONSECUENCIA DE USO**

POSITIVAS:

* Simplifica el uso de sistemas complejos con tareas redundantes.
* Oculta al cliente la complejidad real del sistema.
* Reduce el acoplamiento entre el subsistema y los clientes.

NEGATIVAS:

* Creamos clases para funcionalidad ya existente.

**EJEMPLO**



**VIDEO EXPLICATIVO**



**ENLACE DE DESCARGA DEL PROYECTO**

**CONCLUSIÓN**

Por conclusión, lo más importante de todo es que este patrón se debe aplicar en las clases más representativas y no en las específicas. De no ser así, posiblemente no se tenga el nivel alto deseado.  
 Por aplicación, es ideal construir no demasiados objetos Facade. Sólo algunos representativos que contengan la mayoría de las operaciones básicas de un sistema.

**RECOMENDACIONES**

* Se recomienda en su totalidad tener en claro la aplicación del desarrollo en un sistema
* Se recomienda saber los tipos de soluciones del problema planteado
* Se recomienda no solo aplicar el patrón Facade si no los diversos tipos que existen
* Se recomienda practica

**BIBLIOGRAFIA**

* <https://programacion.net/articulo/patrones_de_diseno_xi_patrones_estructurales_facade_1014>
* <http://migranitodejava.blogspot.com/2011/06/facade.html>
* <https://experto.dev/patron-de-diseno-facade-en-java/>
* <https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46&Itemid=163>
* <https://devexperto.com/patrones-de-diseno-software/>