**PATRÓN DE DISEÑO FACADE**

**Descripción del patrón:**

Busca simplificar el sistema, desde el punto de vista del cliente, 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.

**Ventajas:**

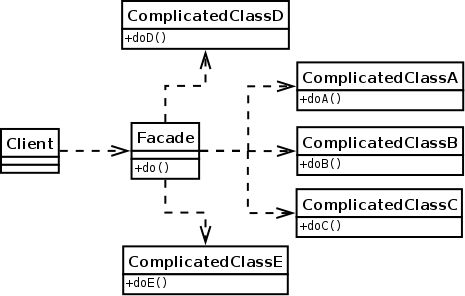
* Oculta a los clientes de la complejidad del subsistema y lo hace fácil de usar.
* Favorece un acoplamiento débil entre el subsistema y sus clientes, consiguiendo que los cambios de las clases del sistema sean transparentes a los clientes.
* Facilita la división en capas y reduce dependencias de compilación.
* No se impide el acceso a las clases del sistema.

**Desventajas:**

* Creamos clases para funcionalidad ya existente.

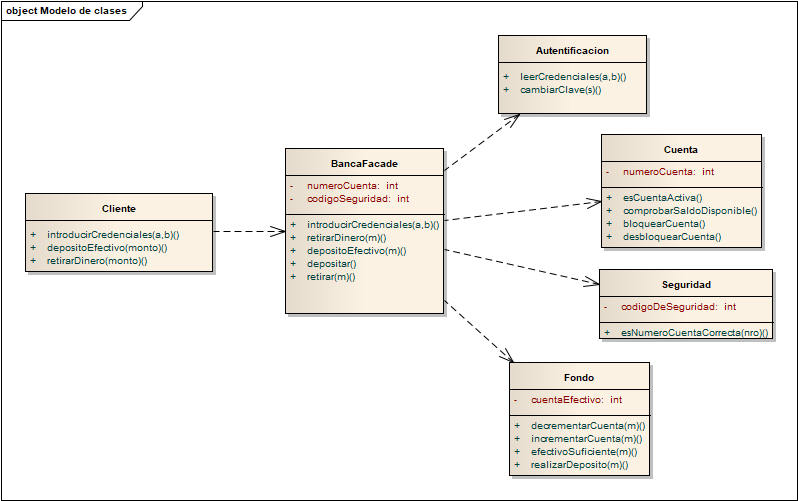
Lo más importante de todo es que este patrón se debe aplicar en las clases más representativas y no en las específicas.

**Diagrama genérico:**

****

Client: Representa al sistema que quiere hacer uso de la clase compleja o el conjunto de susbsistemas mediante la fachada.  
  
Facade: Clase fachada que trata de ofrecer la funcionalidad que demanda el cliente mediante una interfaz sencilla donde, internamente, utiliza las clases complejas.  
  
ComplicatedClassX: Conjunto de clases que se necesitan utilizar y a las que se pretende dar un punto de acceso sencillo mediante la fachada.

**Problema**:  
  
Un ejemplo muy común en la vida real: sacar dinero de la cuenta del banco. Para realizar un retiro de efectivo en un cajero automático, se requiere un gran número de operaciones para garantizar la seguridad y la integridad.

****

**Cuervos

UNIVERSIDAD AUTÓNOMA

**“GABRIEL RENÉ MORENO”**

*FACULTAD DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN*

*Y TELECOMUNICACIONES*

**

**ARQUITECTURA DE SOFTWARE**

**ESTUDIANTE** :Rigoberto Tapia Corpa

**DOCENTE** : Ing. Josué Obed Veizaga Gonzales

**FECHA** : 22 / 11 / 2019

**SANTA CRUZ – BOLIVIA**