Tarea Extraclase II

Algoritmos y Estructuras de Datos I

Patrones de Diseño

Instituto Tecnológico de Costa Rica

Prof. Diego Noguera Mena

Jose Alejandro Chavarria Madriz

Natalia Gonzalez Bermudez

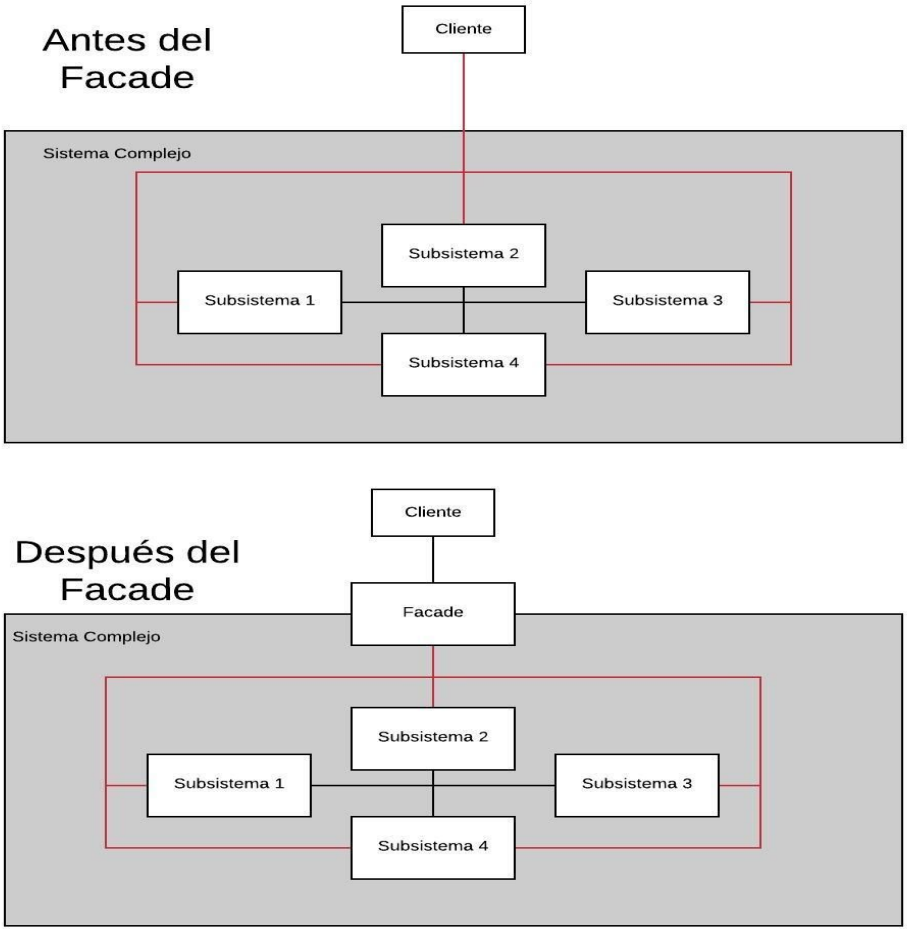
Patrones de Diseño

Façade

Propósito

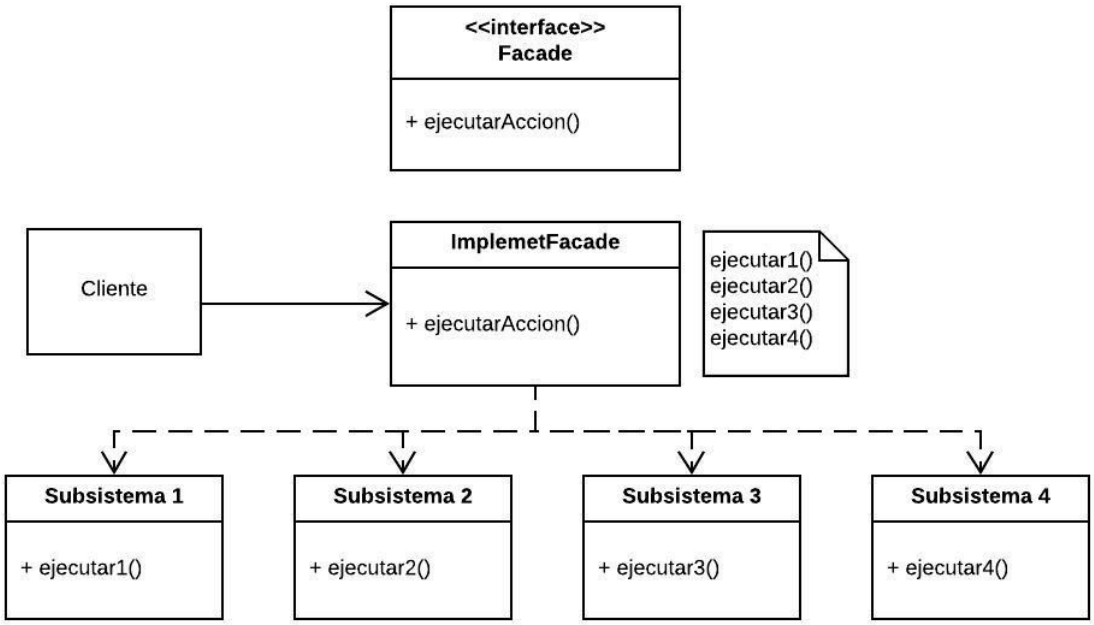
Este patrón tiene como propósito la implementación de una interfaz de muy alto nivel que unifique un conjunto de interfaces de un sistema más complejo. Lo que se quiere lograr con el patrón es hacer que un sistema con varios subsistemas se vuelva más fácil de utilizar. Es una forma de reducir la dependencia y comunicación entre módulos.[1]

Estructura



En el diagrama se puede observar una representación del Facade. Antes de implementarlo el cliente tenía acceso a todos los módulos del sistema complejo y tenía que interactuar con cada uno de ellos. Después de implementar el facade el cliente solo interactúa con el mismo y este se encarga de comunicarse con los subsistemas que deben realizar la operación solicitada por el cliente.[5]

Un ejemplo básico del diagrama de clases del facade es el siguiente:



En el diagrama se ve que el cliente ejecuta una acción y el facade se encarga de acceder a los sistemas que involucran esta acción y ejecuta todas.
Las colaboraciones de un facade son las siguientes:

- Los clientes que envían sus solicitudes al objeto facade y este envía lo necesario a los objeto del subsistema que son los que realizan el trabajo real.
- Los clientes que utilizan el facade no acceden directamente a los subsistemas.

Contexto

Caso 1:

Cuando se quiere tomar un sistema complejo y otorgarle al cliente una vista simplificada del mismo para facilitar su uso y que además tenga un solo punto de acceso al sistema y no pueda acceder a todos los subsistemas.[1]

Caso 2:

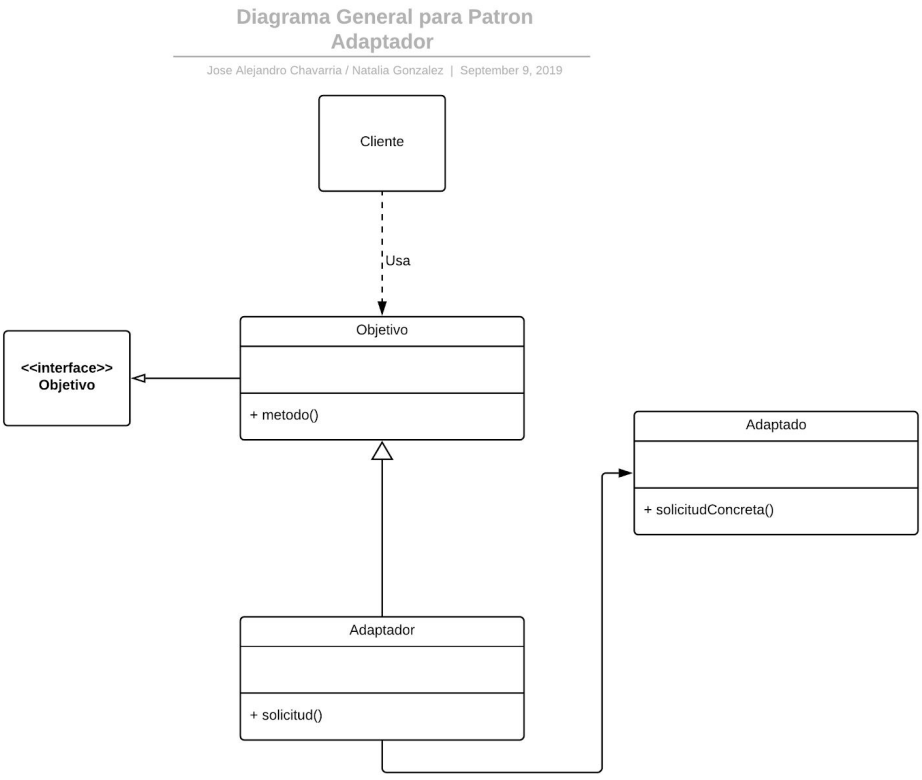
Se utiliza también cuando tenemos varias partes independientes de un sistema y el cliente necesita interactuar con algunas de ellas.[1]

Adapter

Propósito

Es un patrón de diseño que permite la cooperación entre clases, que de otra manera no podrían ser compatibles. La idea detrás de este patrón es convertir la interfaz de una clase en otra, que es la que el cliente espera.[1]

Estructura



En el diagrama se aprecia la estructura general para el patrón adaptador.[2][3] Primeramente es importante ver que el cliente es quien inicia la secuencia al utilizar la clase Objetivo. Al utilizar esta clase lo hace de tal forma que requiere de la clase Adaptado. Como no es posible que el Adaptado coopere con el Objetivo se procede a implementar una clase Adaptador, que por herencia es una subclase de Objetivo y utiliza una clase Adaptado como atributo. Así mediante el método solicitud(), por orden

de Objetivo, ejecuta solicitud Concreta() de la Adaptado, que es el proceso que se desea realizar en primer lugar. De esta forma se logra utilizar a la clase Adaptado a través de una clase Objetivo.

Contexto

Caso 1:

Se tienen una clase cuya interfaz es distinta a la que de desea utilizar. Aquí se puede aplicar el patrón para adaptar dicha clase a la interfaz deseada.[1]

Caso 2:

Se tienen clases que pueden ser reutilizadas pero no fueron pensadas originalmente para cooperar, por lo que sus interfaces son distintas. Nuevamente para poder acoplar estas clases se utiliza un adaptador. [1]

Referencias

Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*.[1]

"The Adapter design pattern - Structure and Collaboration". *w3sDesign.com*. Retrieved 9-09-2019. [2]

Blancarte, O. (2016). *Reactive Programming*. Retrieved from ADAPTER-PATRÓN ESTRUCTURAL: https://reactiveprogramming.io/books/design-patterns/es/catalog/adapter

GeeksfoGeeks. (2019). *GeeksforGeeks*. Retrieved from Facade Design Pattern | Introduction.[4]

MitoCode. (2018, Jul 7). *Curso de Patrones de diseño - 6 Facade*. Retrieved from https://www.youtube.com/watch?v=6dYwdDbhpwQ[5]

Link al repositorio

https://github.com/natigb/Tarea-Extraclase-2