Patrones de diseño

¿Qué es un patrón?

- Un patrón de diseño describe una situación o problema habitual que nos encontramos durante el desarrollo de una aplicación, así como la forma en la que este debe ser resuelto.

- El grupo de cuatro desarrolladores GoF, elaboraron un catálogo que recoge 23 patrones de diseño

- Divididos en tres grupos

Grupos de patrones

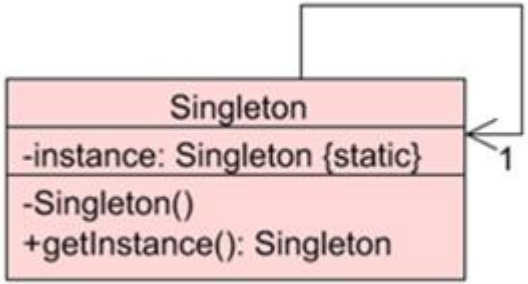
- **Creacionales.** Tratan sobre la forma de crear objetos, abstrayendo de la manera en la que estos son creados e inicializados: Factory, Builder, Singleton,…

- **Estructurales.** Establecen cómo como combinar clases y objetos para formar nuevas estructuras y proporcionar nuevas funciones: Adapter, Decorator, Façade, Composite,…

- **De comportamiento.** Centrados en los algoritmos y en el reparto de responsabilidades entre las clases para, entre otras cosas, evitar el acoplamiento entre los objetos: Observer, Strategy, Mediator,…

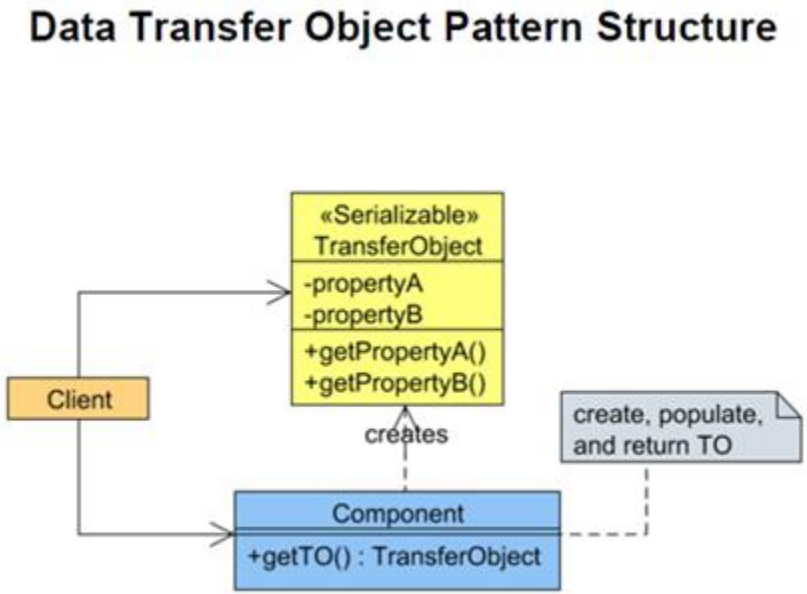
Patrón Singleton

- Se enfoca en el problema de cómo crear una única instancia de una clase y todos los clientes utilicen dicha instancia.



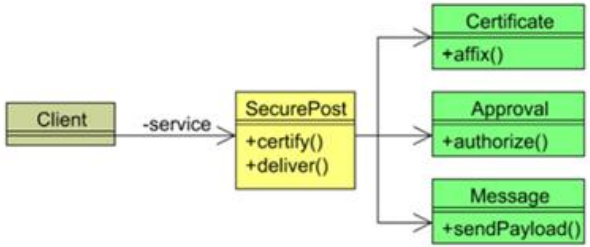
Patrón DTO

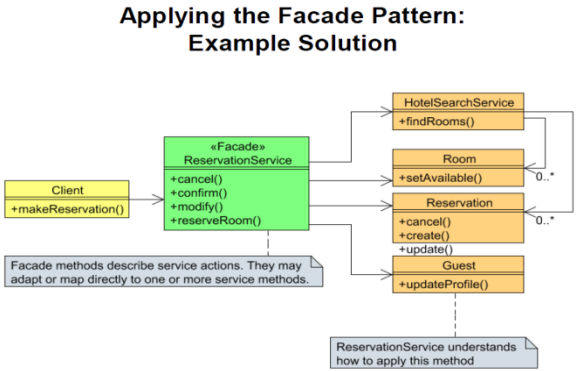
- Data Transfer Object. Problema de cómo enviar un paquete de datos desde una capa a otra: encapsulando los datos en un JavaBean



Patrón Façade

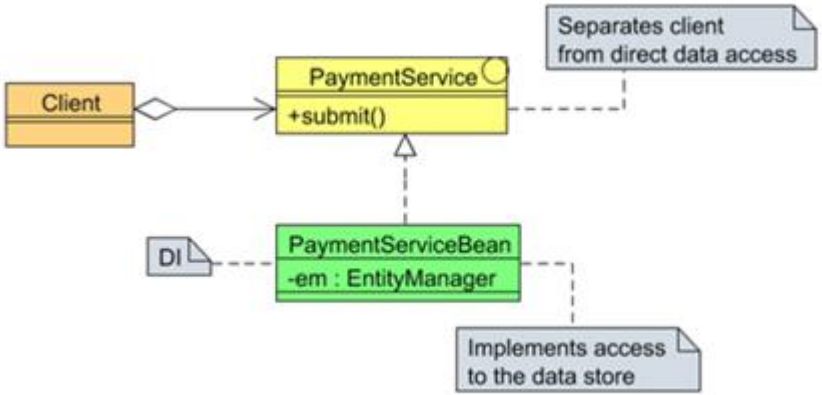
- Se encarga de definir la forma de encapsular una serie de llamadas a diferentes operaciones en una única llamada, a fin de simplificar la utilización de un proceso por parte del cliente





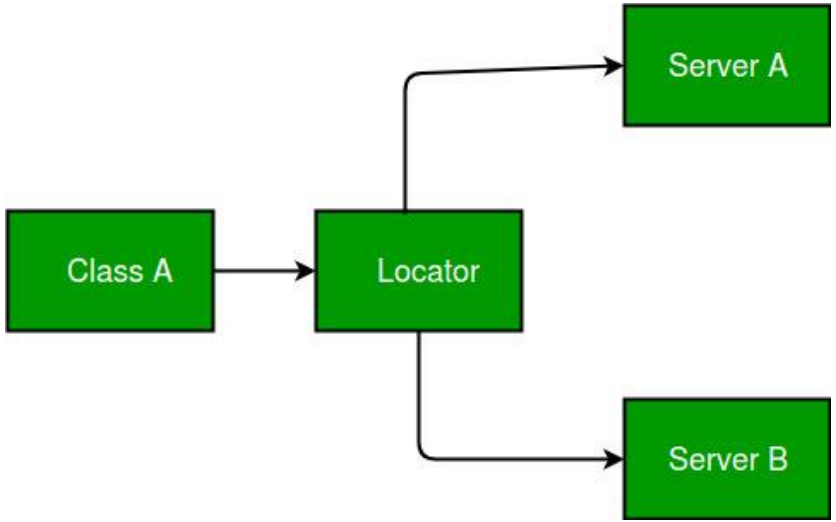
Patrón DAO

- Consiste en aislar las operaciones de acceso a datos en una capa independiente, fuera de la lógica de negocio, aislado a esta de la tecnología de acceso a datos empleada



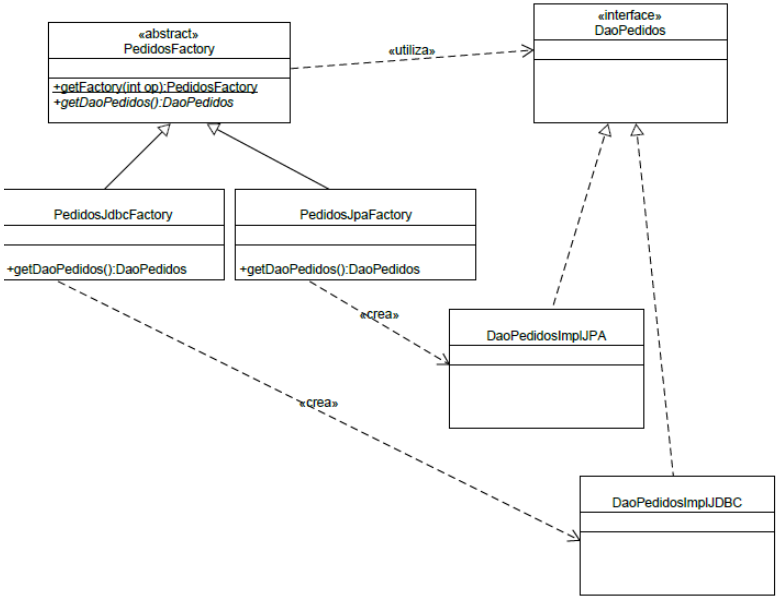
Patrón Service Locator

➢Cuando se necesita hacer uso de un determinado objeto o servicio en diferentes partes de la aplicación, las instrucciones para localizar dicho servicio se codifican en un único lugar.



Patrón Factory

- Permite, a través de una interfaz, utilizar una serie de métodos en una app cliente, abstrayéndose de la implementación de los mismos. Una clase abstracta define el método para obtener la implementación, dejando a las subclases la implementación



Patrón builder

- Consiste en utilizar un objeto simple para crear uno objeto completo, que puede requerir diferentes pasos para su creación. Adecuado cuando la clase no dispone de constructores