**Заместитель (proxy)**

**Суть паттерна**

**Заместитель** — это структурный паттерн проектирования, который позволяет подставлять вместо реальных объектов специальные объекты-заменители. Эти объекты перехватывают вызовы к оригинальному объекту, позволяя сделать что-то *до* или *после* передачи вызова оригиналу.

**Проблема**

Рассмотрим такой пример: у вас есть внешний ресурсоёмкий объект, который нужен не все время, а изредка.

Мы могли бы создавать этот объект не в самом начале программы, а только тогда, когда он кому-то реально понадобится. Каждый клиент объекта получил бы некий код отложенной инициализации. Но, вероятно, это привело бы к множественному дублированию кода.

В идеале, этот код хотелось бы поместить прямо в служебный класс, но это не всегда возможно. Например, код класса может находиться в закрытой сторонней библиотеке.

**Решение**

Паттерн Заместитель предлагает создать новый класс-дублёр, имеющий тот же интерфейс, что и оригинальный служебный объект. При получении запроса от клиента объект-заместитель сам бы создавал экземпляр служебного объекта и переадресовывал бы ему всю реальную работу. Но теперь имеется возможность поместить в класс заместителя какую-то промежуточную логику, которая выполнялась бы до (или после) вызовов этих же методов в настоящем объекте. А благодаря одинаковому интерфейсу, объект-заместитель можно передать в любой код, ожидающий сервисный объект.

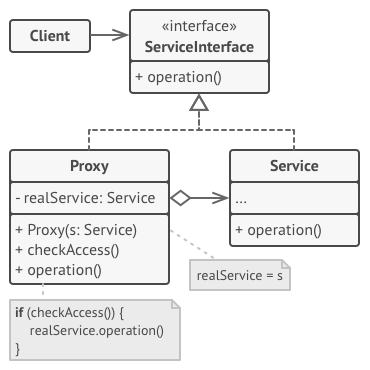
**Структура**

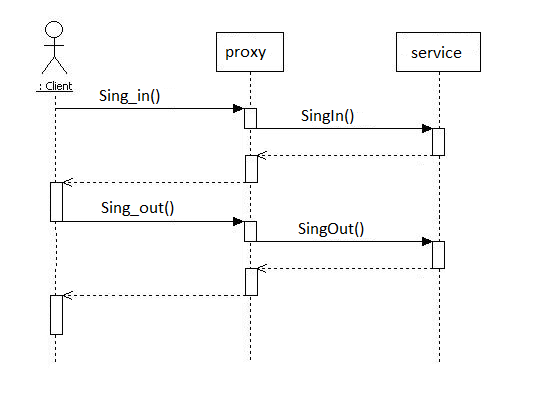
**Интерфейс сервиса** определяет общий интерфейс для сервиса и заместителя. Благодаря этому, объект заместителя можно использовать там, где ожидается объект сервиса.

**Сервис** содержит полезную бизнес-логику.

**Заместитель** хранит ссылку на объект сервиса. После того как заместитель заканчивает свою работу (например, инициализацию, логирование, защиту или другое), он передаёт вызовы вложенному сервису. Заместитель может сам отвечать за создание и удаление объекта сервиса.

**Клиент** работает с объектами через интерфейс сервиса. Благодаря этому, его можно «одурачить», подменив объект сервиса объектом заместителя.





**Преимущества:**

* Позволяет контролировать сервисный объект незаметно для клиента.
* Может работать, даже если сервисный объект ещё не создан.
* Может контролировать жизненный цикл служебного объекта.

**Недостатки:**

* Усложняет код программы из-за введения дополнительных классов.
* Увеличивает время отклика от сервиса.