**ЗАМЕСТИТЕЛЬ** - 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. Чтобы заставить клиента использовать прокси, часто используется фабрика (перед возвращением можно упаковать реальный объект в заместителя). С помощью java.util.reflect.Proxy можно динамически создавать заместителей

ПРИМЕНЕНИЕ:

- ленивая инициализация (виртуальный прокси): когда есть тяжёлый объект, грузящий данные из файловой системы или базы данных

- отложенное копирование: задерживает фактическое копирование объекта до момента выполнения операций с копией. Разновидность виртуального прокси

- защита доступа (защищающий прокси): когда в программе есть разные типы пользователей, и вам хочется защищать объект от неавторизованного доступа. Например, если ваши объекты — это важная часть операционной системы, а пользователи — сторонние программы (хорошие или вредоносные)

- локальный запуск сервиса (удалённый прокси): когда настоящий сервисный объект находится на удалённом сервере

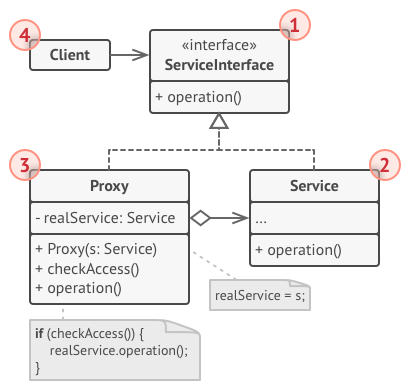
- логирование запросов (логирующий прокси): когда требуется хранить историю обращений к сервисному объекту

- кеширование объектов (кеширующий прокси): когда нужно кешировать результаты запросов клиентов и управлять их жизненным циклом

- выполенение дополнительных действий («умная» ссылка): когда нужно выполнить дополнительные действия при обращении к объекту

- потокобезопасный доступ (синхронизирующий прокси): предоставляет безопасный доступ к объекту из нескольких потоков

- упрощение интерфейса (упрощающий прокси): скрывает сложность и управляет доступом к сложному набору классов. Иногда называется "фасадным заместителем"

РЕАЛИЗАЦИЯ:

**1. ИНТЕРФЕЙС СЕРВИСА**: определяет общий интерфейс для сервиса и заместителя. Благодаря этому, объект заместителя можно использовать там, где ожидается объект сервиса. Например, "сложное вычисление"

**2. СЕРВИС**: содержит полезную бизнес-логику. Например, "калькулятор"

**3. ЗАМЕСТИТЕЛЬ**: хранит ссылку на объект сервиса. После того как заканчивает свою работу (например, инициализацию, логирование, защиту или другое), он передаёт вызовы вложенному сервису. Может сам отвечать за создание и удаление объекта сервиса. Например, "заместитель калькулятора"

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

ПЛЮСЫ:

- позволяет контролировать сервисный объект незаметно для клиента

- может работать, даже если сервисный объект ещё не создан

- может контролировать жизненный цикл служебного объекта

МИНУСЫ:

- усложняет код программы из-за введения дополнительных классов

- увеличивает время отклика от сервиса