

Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Рязанов Иван Дмитриевич

**ОРГАНИЗАЦИЯ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ.
СТРУКТУРНЫЙ ПАТТЕРН «ЗАМЕСТИТЕЛЬ»**

Лабораторная работа

студента образовательной программы «Программная инженерия»
по направлению подготовки 09.03.04 Программная инженерия

Руководитель
к.т.н., доцент кафедры Информационных технологий в бизнесе НИУ ВШЭ-Пермь

А.В. Кычкин

Пермь, 2020 год

Оглавление

Глава 1. Паттерн «Заместитель»	3
Глава 2. Проектирование и реализация	7
2.1 Проектирование	7
2.2 Реализация	7

Глава 1. Паттерн «Заместитель»

Название и классификация паттерна. Заместитель - паттерн, структурирующий объекты, который предоставляет объект, который контролирует доступ к другому объекту, перехватывая все вызовы.

Назначение. Управление ресурсоемкими объектами, при котором нет необходимости создавать экземпляры таких объектов до момента их реального использования. Проксу является суррогатом другого объекта и управляет доступом к нему.

Заместитель это объект, интерфейс которого идентичен интерфейсу реального объекта. При первом запросе клиента заместитель создает реальный объект, сохраняет его адрес и затем отправляет запрос этому реальному объекту. Все последующие запросы просто переадресуются инкапсулированному реальному объекту.

Применимость. Существуют ситуации, когда можно использовать паттерн Проксу:

1. Виртуальный прокxu является заместителем объектов, создание которых обходится дорого. Реальный объект создается только при первом запросе/доступе клиента к объекту.
2. Удаленный прокxu предоставляет локального представителя для объекта, который находится в другом адресном пространстве.
3. Защитный прокxu контролирует доступ к основному объекту. "Суррогатный" объект предоставляет доступ к реальному объекту, только вызывающий объект имеет соответствующие права.
4. Интеллектуальный прокxu выполняет дополнительные действия при доступе к объекту.

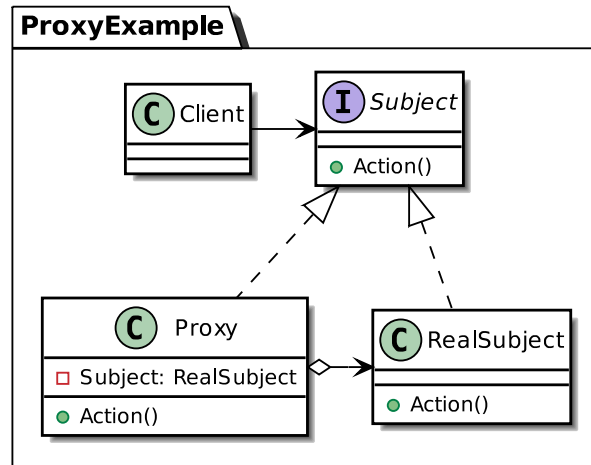


Рис. 1.1. Диаграмма классов паттерна «Заместитель»

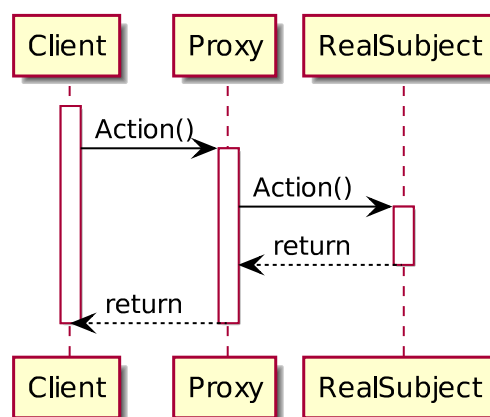


Рис. 1.2. Диаграмма последовательности паттерна «Заместитель»

Участники

1. Проху - заместитель:

- хранит ссылку, которая позволяет заместителю обратиться к реальному субъекту. Объект класса Проху может обращаться к объекту класса Subject, если интерфейсы классов RealSubject и Subject одинаковы;
- предоставляет интерфейс, идентичный интерфейсу Subject, так что заместитель всегда может быть подставлен вместо реального субъекта;
- контролирует доступ к реальному субъекту и может отвечать за его создание и удаление;
- прочие обязанности зависят от вида заместителя:
- *удаленный заместитель* отвечает за кодирование запроса и его аргументов и отправление закодированного запроса реальному субъекту в другом адресном пространстве;
- *виртуальный заместитель* может кэшировать дополнительную информацию о реальном субъекте, чтобы отложить его создание.
- *защищающий заместитель* проверяет, имеет ли вызывающий объект необходимые для выполнения запроса права;

2. Subject - субъект: определяет общий для RealSubject и Проху интерфейс, так что класс Проху можно использовать везде, где ожидается RealSubject;

3. RealSubject - реальный субъект: определяет реальный объект, представленный заместителем.

Отношения Проху при необходимости переадресует запросы объекту RealSubject. Детали зависят от вида заместителя.

Плюсы и минусы Плюсы:

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

Минусы:

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

Области применения

1. Системы доставки контента (кеширование важно).
2. ПО, требующее авторизацию пользователей.

Глава 2. Проектирование и реализация

2.1. Проектирование

Для реализации был выбран 3 вариант:

«Программа для обеспечения оплаты подписок на сервисы.»

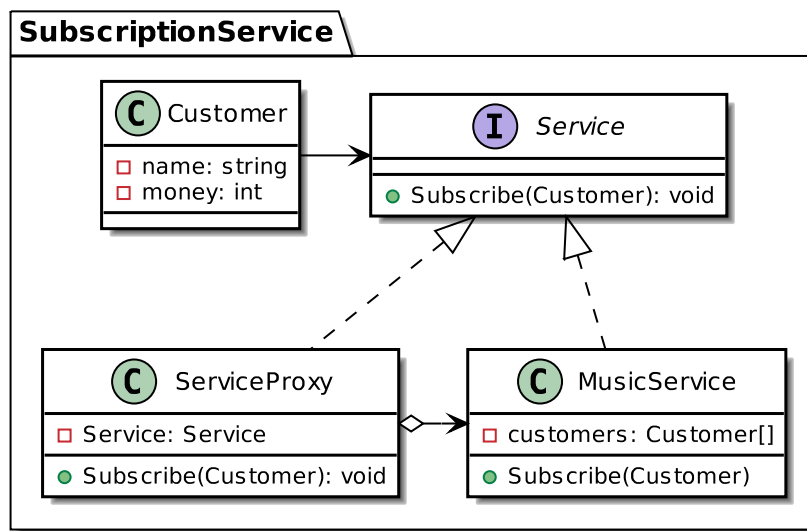


Рис. 2.1. Диаграмма классов

Участники

1. **Service** — сервис, на который можно подписаться.
2. **MusicService** — реализация сервиса в виде сервиса предоставления музыки.
3. **ServiceProxy** — заместитель, который контролирует оплату перед подпиской на сервис.
4. **Customer** — пользователь.

2.2. Реализация

Реализация паттерна «Заместитель» находится в git-репозитории по ссылке: github.com