Технический Университет Молдовы

Факультет FCIM

# Отчёт

по предмету Шаблоны проектирования

по Лабораторной работе No 2

# Тема: “Структурные шаблоны ”

Выполнил: ст.гр. TI-174: Тодосичук Денис

Проверил: преп. Булдумак Олег

Кишинёв – 2020

***Задание:***

Выбрать 5 любых шаблонов из группы Структурных шаблонов и реализовать их.

Подробно описать один шаблон.

**Адаптер** — это структурный паттерн проектирования, который позволяет объектам с несовместимыми интерфейсами работать вместе.

**Мост** — это структурный паттерн проектирования, который разделяет один или несколько классов на две отдельные иерархии — абстракцию и реализацию, позволяя изменять их независимо друг от друга.

**Компоновщик** — это структурный паттерн проектирования, который позволяет сгруппировать множество объектов в древовидную структуру, а затем работать с ней так, как будто это единичный объект.

**Декоратор** — это структурный паттерн проектирования, который позволяет динамически добавлять объектам новую функциональность, оборачивая их в полезные «обёртки».

**Фасад** — это структурный паттерн проектирования, который предоставляет простой интерфейс к сложной системе классов, библиотеке или фреймворку.

**Легковес** — это структурный паттерн проектирования, который позволяет вместить бóльшее количество объектов в отведённую оперативную память. Легковес экономит память, разделяя общее состояние объектов между собой, вместо хранения одинаковых данных в каждом объекте.

**Название/Категория/Какую проблему решает:**

**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**

Необходимо контролировать доступ к объекту, не изменяя при этом поведение клиента.

Необходимо иметь доступ к объекту так, чтобы не создавать реальные объекты непосредственно, а через другой объект, который может иметь дополнительную функциональность.

**Виды:**

**Протоколирующий прокси**: сохраняет в лог все вызовы «Субъекта» с их параметрами.

**Удалённый заместитель** (англ. *remote proxies*): обеспечивает связь с «Субъектом», который находится в другом адресном пространстве или на удалённой машине. Также может отвечать за кодирование запроса и его аргументов и отправку закодированного запроса реальному «Субъекту»,

**Виртуальный заместитель** (англ. *virtual proxies*): обеспечивает создание реального «Субъекта» только тогда, когда он действительно понадобится. Также может кэшировать часть информации о реальном «Субъекте», чтобы отложить его создание,

**Копировать-при-записи**: обеспечивает копирование «субъекта» при выполнении клиентом определённых действий (частный случай «виртуального прокси»).

**Защищающий заместитель** (англ. *protection proxies*): может проверять, имеет ли вызывающий объект необходимые для выполнения запроса права.

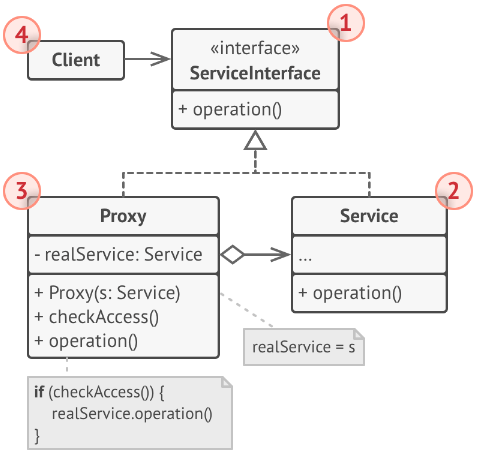
**Кэширующий прокси**: обеспечивает временное хранение результатов расчёта до отдачи их множественным клиентам, которые могут разделить эти результаты.

**Экранирующий прокси**: защищает «Субъект» от опасных клиентов (или наоборот).

**Синхронизирующий прокси**: производит синхронизированный контроль доступа к «Субъекту» в асинхронной многопоточной среде.

**«Умная» ссылка** (англ. *smart reference proxy*): производит дополнительные действия, когда на «Субъект» создается ссылка, например, рассчитывает количество активных ссылок на «Субъект».

**Диаграмма классов:**



**Участники:**

1. **Интерфейс сервиса** определяет общий интерфейс для сервиса и заместителя. Благодаря этому, объект заместителя можно использовать там, где ожидается объект сервиса.
2. **Сервис** содержит полезную бизнес-логику.
3. **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.
4. **Клиент** работает с объектами через интерфейс сервиса. Благодаря этому, его можно «одурачить», подменив объект сервиса объектом заместителя.

**Преимущества и недостатки шаблона:**

**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.

**Взаимодействие с другими шаблонами:**

Взаимодествует с шаблонами: Декоратор, Адаптер, Фасад.