Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет Программной инженерии и компьютерной техники

**Лабораторная работа №2**

по дисциплине «Архитектура программных систем»

**Выполнил**

Ореховский А.,

группа P3417

**Преподаватель**

Перл И. А.

Санкт-Петербург,

2020

Оглавление

[Задание 3](#_Toc60010374)

[Low cohesion & High Coupling 3](#_Toc60010375)

[Adapter 3](#_Toc60010376)

[Polymorphism 3](#_Toc60010377)

[Observer 3](#_Toc60010378)

[Выводы 4](#_Toc60010379)

# Задание

Из списка шаблонов проектирования GoF и GRASP выбрать 3-4 шаблона и для каждого из них придумать 2-3 сценария, для решения которых могу применены выбранные шаблоны.

Сделать предположение о возможных ограничениях, к которым можем привести использование шаблона в каждом описанном случае.

Обязательно выбрать шаблоны из обоих списков.

# Low cohesion & High Coupling

* Микро-сервисная архитектура. Каждый сервис выполняет конкретную задачу и минимально контактирует с другими сервисами
* Набор утилит на примере GNU Binutils. Каждая утилита выполняет минимальные обязанности и почти ничего не знает о других утилитах.

Недостаток заключается в том, что найти идеальный баланс между двумя паттернами очень сложно так как их природа противоположна друг другу.

# Adapter

* Зачастую в паттерне MVC класс Model хранящийся в базе данных отличается от класса ModelView, который используется для отображения того же объекта класса Model.
* Преобразование списка объектов в его отображение. Например, в Android-e, когда мы из списка объектов делаем RecyclerView, который и будет отображаться.
* Необходимость использования стороннего класса, интерфейс которого не соответствует остальному коду приложения.

Недостаток заключается в том, что адаптер усложняет код программы из-за введения дополнительных классов.

# Polymorphism

* Использование фильтрация запрашиваемых данных через URL в REST. Например, для URL /findAll будет возвращен список всех объектов, а для URL /findAll/producer/Russia будет возвращен список только тех, которые произведены в России. Оба URL нацелены на получение списка элементов и лишь дополнительные параметры определяют альтернативную реализацию.
* Два класса, которые наследуется от класса Logger, а именно ConsoleLogger и FileLogger.
* Создание новой строки. Строка может быть создана из числа целого типа, числа типа с плавающей точкой, и набора символов разными методами одной природы.

В случае, если есть группа похожих объектов, и все они должны различным способом реагировать на одну и ту же ситуацию, те необходимо добавить и реализовать этот метод для каждого из этих объектов, что не очень хорошо если объектов скажем 200 ведь для каждого из них придется писать код ручками.

# Observer

* Любой чат имеет смысл строить так, что любой участник предпочтет подписку на обновление, постоянному поиску этого обновления. Гораздо менее затратно оповестить всех участников чата о новом сообщении, чем тратить ресурсы на выдачу всего списка сообщений, чтобы все участники сверили его с кэшированным.
* Обработка нажатий на клавиши клавиатуры/мыши по той же причине.

# Выводы

Несмотря на то, что каждый паттерн хорошо реализует определенную задачу, его использование не является обязательным. Это всего лишь рекомендация, которая позволит писать более пере используемый и поддерживаемый код. Более того паттерны не идеальны и каждый имеет свои недостатки. Использование или не использование каждого паттерна оставляется на усмотрение программиста, а некоторые и вовсе используются бессознательно.