МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**Факультет программной инженерии и компьютерной техники**

**Лабораторная работа No2**

по дисциплине

“Архитектура программных систем”

Студент:

Гуменник Петр Олегович

Группа P3333

Преподаватель:

Перл Иван Андреевич

Санкт-Петербург, 2025

## Шаблоны GoF

1. Observer

#### Сценарий 1: Система контроля за состоянием складской техники

Погрузчики, конвейеры, упаковочные станции, все устройства рассылают уведомления при перегреве, поломке, критическом уровне заряда.

**Ограничения**:

* Огромное количество устройств в одном цеху создаёт высокую частоту событий, что может вызвать задержки при обработке уведомлений.
* При сбоях в сети некоторые уведомления могут не дойти, поэтому нужна гибкая система ретрансляции или логирования.
* Если устройство не «отписывается» при выводе из эксплуатации, в системе сохранится «призрак», нагромождающий логику.

#### Сценарий 2: Платформа интеллектуальных уведомлений об изменениях курса криптовалют

Подписчики: различные боты и трейдеры; при изменении цены на определённый процент они должны получить сигнал.

**Ограничения**:

* Частые колебания курса и большое число наблюдателей сильно загружают систему оповещений.
* Нужен механизм тонкой фильтрации, чтобы не отправлять уведомления при минимальных изменениях стоимости.
* Отсутcтвие своевременной «отписки» ботов, которые давно не торгуют, приводит к высокому объёму «лишних» уведомлений.

1. Decorator

#### Сценарий 1: Расширение функционала личного кабинета клиента

Базовый «Личный кабинет», поверх которого идут декораторы: чат с поддержкой, бонусная программа, управление подписками, статистика покупок.

**Ограничения**:

* При слишком большом количестве «декораторов», взаимодействующих друг с другом, растёт риск коллизий (напр. бонусы не совмещаются с определёнными подписками).
* Может быть сложно отследить, какой декоратор внёс изменения в логику кабинета, если декораторов много.
* Расход памяти и CPU может увеличиваться при глубоких цепочках декораторов.

#### Сценарий 2: Тонкая настройка объёмного звука в VR-играх

Базовый объект: аудиопоток. Декораторы: эхо, реверберация, 3D-звук, шумоподавление, динамическая компрессия.

**Ограничения**:

* Важно соблюдать порядок применения декораторов: если сначала наложить шумоподавление, а потом реверберацию, итоговый звук может получиться сильно искажённым.
* Каждый декоратор увеличивает нагрузку на процессор, а в VR-средах критичен реальный FPS.
* Трудно тестировать все возможные комбинации эффектов: их может быть слишком много.

#### Сценарий 3: Интернет-магазин с системой скидок и доп. услуг

Базовая цена товара может быть «декорирована» бонусными баллами, сезонными скидками, страховками, подарочными упаковками.

**Ограничения**:

* Накопление скидок может превратиться в «двойные» или «тройные» скидки, нужно точно определять приоритет наложения декораторов.
* Некоторые декораторы (например, страховка) могут конфликтовать с другими (бесплатная подарочная упаковка), требуя доп. проверок.
* Сложно объяснить логику конечной стоимости покупателю, когда она пересчитана кучей декораторов (требуется визуализировать цепочку декораторов).

## Шаблоны GRASP

1. Creator

#### Сценарий 1: Создание API-ключей для интеграций

«KeyManager» генерирует разные типы ключей (временные, постоянные, ограниченные правами), выступая Creator.

**Ограничения**:

* При появлении новых алгоритмов генерации (например, более безопасные методы) придётся обновлять Creator, что может затронуть много кода.
* Нужно строго следить за безопасностью хранения ключей, Creator может стать «узким местом», если в нём сосредоточена вся логика шифрования.
* Разные типы ключей (например, OAuth-токен vs. ключ с ограниченными правами) могут потребовать разных параметров, затрудняя единый интерфейс создания.

#### Сценарий 2: Автоматическая генерация ботов в чат-платформе

При подключении нового корпоративного клиента создаётся бот с индивидуальным набором сценариев, расписанием и языковыми моделями.

**Ограничения**:

* Creator (например, «BotFactory») рискует «обрасти» сложной бизнес-логикой, если для каждого клиента нужно множество индивидуальных настроек.
* Быстрое масштабирование: если нужно «сотни» ботов в один момент, нужно оптимизировать процесс создания и инициализации.
* Сложно поддерживать обратную совместимость, когда появляются новые типы ботов с особыми функциями.

1. Controller

#### Сценарий 1: Модуль обработки заказов для агрегатора доставки

«OrderController» принимает запрос от пользователя, проверяет ближайшие рестораны/магазины, формирует заказ, вызывает курьеров, обрабатывает оплату.

**Ограничения**:

* Большой объём входящих заказов может приводить к перегрузке, если не реализовать распределённую архитектуру.
* При добавлении новых типов товаров (например, лекарства со специальным режимом доставки) придётся дорабатывать логику.
* Высокая связность с модулями расчёта стоимости, логистикой, оплатой, требуется поддерживать множество внешних API.

#### Сценарий 2: Административная панель для онлайн-университета

«UniversityController» управляет созданием курсов, зачислением студентов, расписанием вебинаров, учётом успеваемости.

**Ограничения**:

* Если не переносить бизнес-логику в сервисы, контроллер рискует превратиться в «гиганта» с сотнями методов.
* Постоянные изменения учебных программ (новые курсы, новые форматы занятий) заставляют дорабатывать контроллер.

## Вывод

Применение шаблонов GoF и GRASP позволяет грамотно распределять обязанности, обеспечивая гибкую и легко расширяемую архитектуру. Однако следует учитывать возможные ограничения: нагрузку при массовых уведомлениях (Observer), непредсказуемые эффекты при множественном наложении декораторов (**Decorator**), риски «раздутия» логики при создании объектов (Creator) и «божественного» контроллера (Controller). Главное использовать паттерны по делу, чтобы не усложнять проект сверх меры.