Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 «Программная инженерия» –

Системное и прикладное программное обеспечение

**Отчёт**

**По лабораторной работе №2**

**По архитектуре программных систем**

Выполнил:

студент 3 курса

Батманов Даниил Евгеньевич

Группа: Р3307

Принял:

Пёрл Иван Андреевич

Отчёт принят «\_\_»\_\_\_\_\_2024 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

г. Санкт-Петербург, 2024

Оглавление

[Задание 3](#_Toc187000556)

[Выбранные шаблоны 4](#_Toc187000557)

[1. GoF: Стратегия (Strategy) 4](#_Toc187000558)

[2. GoF: Наблюдатель (Observer) 4](#_Toc187000559)

[3. GRASP: Контроллер (Controller) 5](#_Toc187000560)

[4. GRASP: Перенаправление (Indirection) 5](#_Toc187000561)

[Систематизирующая таблица 6](#_Toc187000562)

# 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.

# Выбранные шаблоны

1. GoF: Стратегия (Strategy)
2. GoF: Наблюдатель (Observer)
3. GRASP: Контроллер (Controller)
4. GRASP: Перенаправление (Indirection)

# 1. GoF: Стратегия (Strategy)

**Описание шаблона:**

Стратегия позволяет определить семейство алгоритмов, инкапсулировать их и делать взаимозаменяемыми. Это помогает избегать жесткой привязки к конкретным реализациям.

**Сценарии:**

1. **Реализация различных методов оплаты в интернет-магазине**
   1. *Проблема:* В магазине нужно поддерживать несколько способов оплаты (карты, PayPal, криптовалюты и т. д.).
   2. *Решение:* Создать интерфейс PaymentStrategy и несколько реализаций (CardPayment, PayPalPayment и т. д.). Клиент выбирает стратегию оплаты во время оформления заказа.
   3. *Ограничения:*
      1. Повышенная сложность структуры кода из-за добавления новых классов.
      2. Трудности в управлении стратегиями, если их слишком много.
2. **Настройка поведения персонажа в игре**
   1. *Проблема:* Персонажи могут атаковать, защищаться, лечиться или выполнять другие действия, и поведение должно меняться динамически.
   2. *Решение:* Создать общий интерфейс для поведения и реализовать различные стратегии (AggressiveBehavior, DefensiveBehavior, HealingBehavior).
   3. *Ограничения:*
      1. Дополнительная память на хранение множества объектов-стратегий.
      2. Сложность управления связями между стратегиями и внешними условиями игры.

# 2. GoF: Наблюдатель (Observer)

**Описание шаблона:**

Наблюдатель используется для установки зависимости «один-ко-многим» между объектами, чтобы при изменении состояния одного объекта автоматически обновлялись все зависимые.

**Сценарии:**

1. **Обновление интерфейса пользователя при изменении данных**
   1. *Проблема:* В приложении данные хранятся в модели, а интерфейс должен автоматически обновляться при их изменении.
   2. *Решение:* Модель реализует интерфейс «наблюдаемый объект», а интерфейсные элементы выступают наблюдателями, подписанными на изменения модели.
   3. *Ограничения:*
      1. Высокая зависимость между наблюдателями и наблюдаемым.
      2. Возможные проблемы производительности при большом числе наблюдателей.
2. **Оповещение клиентов о скидках в системе рассылок**
   1. *Проблема:* Интернет-магазин должен уведомлять подписчиков о новых скидках и акциях.
   2. *Решение:* Подписчики (наблюдатели) получают уведомления от сервиса скидок (наблюдаемый объект).
   3. *Ограничения:*
      1. Потенциальные проблемы с асинхронным уведомлением при большом числе подписчиков.
      2. Зависимость подписчиков от формата данных, предоставляемых системой.

# 3. GRASP: Контроллер (Controller)

**Описание шаблона:**

Контроллер отвечает за обработку событий, поступающих от пользователя или системы, и делегирует выполнение необходимых операций другим объектам.

**Сценарии:**

1. **Обработка запросов в системе интернет-банкинга**
   1. *Проблема:* Разные виды операций (переводы, оплата счетов, запросы выписок) требуют корректного управления.
   2. *Решение:* Контроллер выступает связующим звеном между пользовательским интерфейсом и бизнес-логикой, направляя запросы соответствующим модулям.
   3. *Ограничения:*
      1. Увеличение сложности контроллера, если он обрабатывает слишком много задач.
      2. Возможность превращения в «божественный объект» при неправильной декомпозиции.
2. **Управление процессом регистрации пользователей**
   1. *Проблема:* Регистрация может включать валидацию данных, отправку писем и создание учетной записи.
   2. *Решение:* Контроллер управляет последовательностью вызовов необходимых сервисов (валидация, отправка, создание).
   3. *Ограничения:*
      1. Высокая ответственность контроллера за разные аспекты процесса.
      2. Потенциальная проблема при внесении изменений в процесс, если логика не делегируется.

# 4. GRASP: Перенаправление (Indirection)

**Описание шаблона:**

Шаблон направлен на использование промежуточного объекта для управления взаимодействием между другими объектами. Это позволяет ослабить связь между объектами, делая систему более гибкой и масштабируемой.

**Сценарии:**

1. **Прокси-сервер для управления запросами**
   1. *Проблема:* Система должна обрабатывать сетевые запросы и направлять их к нужным серверам, скрывая сложность внутренней структуры.
   2. *Решение:* Внедрить объект-посредник (прокси), который перенаправляет запросы на основе правил маршрутизации.
   3. *Ограничения:*
      1. Дополнительная задержка из-за введения посредника.
      2. Усложнение конфигурации при увеличении числа серверов.
2. **Обработка платежей через сторонние сервисы**
   1. *Проблема:* Приложение работает с несколькими платежными системами (Stripe, PayPal и т. д.), и нужно обеспечить единый интерфейс для взаимодействия.
   2. *Решение:* Ввести объект перенаправления, который определяет, какой сервис использовать для конкретного платежа.
   3. *Ограничения:*
      1. Повышенная сложность от поддержки и настройки логики перенаправления.
      2. Возможные ошибки при неправильной обработке специфики платежных систем.
3. **Логгирование действий в приложении**
   1. *Проблема:* В приложение нужно встроить централизованную систему логирования, которая отправляет логи в файл, базу данных или внешний сервис.
   2. *Решение:* Использовать объект перенаправления для маршрутизации логов в зависимости от типа события (например, ошибки отправлять в базу, а информационные сообщения — в текстовый файл).
   3. *Ограничения:*
      1. Задержки при записи логов из-за добавления дополнительного слоя.
      2. Усложнение отладки, если логика перенаправления содержит ошибки.

# Систематизирующая таблица

|  |  |  |
| --- | --- | --- |
| **Шаблон** | **Пример использования** | **Ограничения** |
| *Стратегия* | Методы оплаты в магазине, поведение персонажа в игре | Усложнение структуры кода, управление множеством стратегий. |
| *Наблюдатель* | Обновление UI, оповещение о скидках | Высокая зависимость, проблемы производительности при большом числе наблюдателей. |
| *Контроллер* | Регистрация пользователей, обработка банковских операций | Перегрузка логики, превращение в «божественный объект», сложность изменений. |
| *Перенаправление* | Прокси-сервер, платежи, логгирование | Дополнительная задержка, усложнение конфигурации и отладки. |