Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Архитектура программных систем»

**Отчет**

По лабораторной работе №2

Выполнил:

Манжиков Никита Сергеевич

*Р33102*

Преподаватель:

*Перл И. А.*

Санкт-Петербург, 2023 г.

Задание:

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

Сделать предположение о возможных ограничениях, к которым можем привести использование шаблона в каждом описанном случае. Обязательно выбрать шаблоны из обоих списков.

GoF шаблон “Прототип” (Prototype)

**1. Медицинские документы:**

Сценарий 1: Копирование медицинского шаблона

1)Врачам часто приходится заполнять различные формы медицинских документов. Используя шаблон "Прототип", можно создать прототип заполненного документа, который затем может быть скопирован для создания новых документов с теми же основными данными.

2)В случае изменения форматов или стандартов для медицинских документов, можно использовать прототип для создания новой версии документа, а затем адаптировать его под новые требования.

Ограничения:

* В случае сложных медицинских данных, которые требуют специфичных обработок, прототип может оказаться недостаточно гибким.

Для разработки системы управления медицинскими документами можно рассмотреть несколько шаблонов проектирования, учитывая требования к медицинским данным и их сложную структуру. Давайте рассмотрим два подхода:

Предметная область: Игровые персонажи

* **Сценарий 1:** Разработка компьютерной игры, где игровые персонажи могут иметь различные визуальные стили, экипировку и характеристики. Использование прототипа для быстрого создания и тестирования различных вариантов персонажей без необходимости создания каждого с нуля. Например, можно создать прототип воина с базовыми параметрами, а затем создавать его модификации с разными внешними и игровыми характеристиками.
* **Сценарий 2:** Расширение функциональности игры с добавлением новых классов персонажей. Прототип позволяет быстро создать базовый шаблон нового класса и тестировать его в игровом окружении, что экономит время разработки.
* **Сценарий 3:** Внедрение новой системы анимаций для персонажей. Прототип может служить основой для тестирования различных анимаций и их комбинаций без необходимости внесения изменений в основной код.

*Ограничения:* Прототип может столкнуться с ограничениями, если игра имеет сложную логику взаимодействия персонажей, требующую подробного исследования в реальной среде игрового процесса.

**2 Шаблон проектирования: Компоновщик (Composite)**

* **Сценарий 1:** Создание иерархии классов для представления различных типов медицинских документов (например, анализы, история болезни, рецепты) как компонентов. Компоновщик позволит создавать составные объекты, представляющие целые медицинские пакеты.
* **Сценарий 2:** Расширение системы для поддержки версионности медицинских документов. Компоновщик обеспечит удобство работы с различными версиями, сохраняя структуру компонентов.

*Ограничения:* Шаблон "Компоновщик" может быть избыточен, если нет потребности в иерархической структуре медицинских данных.

Для создания сайта выборности предметов на семестр для вуза можно рассмотреть несколько шаблонов проектирования, учитывая специфику задачи. Давайте рассмотрим два подхода:

1. **Шаблон проектирования: Стратегия (Strategy)**

* **Сценарий 1:** Разработка системы выбора предметов, где каждый алгоритм выбора (например, приоритет по специализации, график занятости) представляет собой отдельную стратегию. Это позволит легко добавлять и изменять стратегии выбора предметов без изменения основного кода.

*Ограничения:* Шаблон "Стратегия" может усложнить структуру системы, если количество стратегий становится слишком велико.

Представим, у нас есть система управления транспортным потоком в городе. Различные стратегии могут использоваться для оптимизации движения различных видов транспорта в зависимости от текущих условий, таких как плотность движения, погодные условия и события на дорогах.

**Применение шаблона "Стратегия":**

1. **Стратегия для оптимизации городского транспорта:**
   * **Описание стратегии:** Разработка стратегии, которая оптимизирует движение общественного транспорта, учитывая расписание, маршруты и текущий объем пассажиропотока.
   * **Применение:** В периоды пиковой загрузки, такая стратегия может использоваться для увеличения числа транспортных средств или изменения маршрутов для более эффективного обслуживания пассажиров.
2. **Стратегия для управления светофорами:**
   * **Описание стратегии:** Разработка стратегии управления светофорами, которая реагирует на текущий объем транспорта, минимизируя время простоя и улучшая пропускную способность перекрестков.
   * **Применение:** В зависимости от плотности движения на перекрестках, система может переключаться между различными стратегиями, такими как адаптивное управление светофорами или предпочтение определенным направлениям.

# Шаблон проектирования: Наблюдатель (Observer)

Сценарий 1: Реализация системы, в которой студенты могут подписываться на уведомления об изменениях в расписании или доступности предметов. Наблюдатель позволяет легко управлять подписчиками и рассылать им актуальную информацию.

Сценарий 2: Внедрение автоматических уведомлений о конфликтах в выборе предметов. Наблюдатель позволяет быстро интегрировать новые оповещения без изменения основного кода.

Ограничения: Шаблон "Наблюдатель" может привести к избыточному уровню уведомлений, если не управлять аккуратно списками подписчиков.

Выводы:

В процессе изучения шаблонов программирования GoF и GRASP, я убедился в их ценности и важности в разработке программного обеспечения. Глубокий взгляд на эти шаблоны позволил мне сформулировать следующие выводы:

В первую очередь, шаблоны представляют собой не только инструмент для оптимизации кода, но и мощное средство для улучшения читаемости и понимания кодовой базы. Разбивая сложные структуры и алгоритмы на управляемые компоненты, шаблоны делают код более доступным для других разработчиков. Это способствует обмену знаний в команде и облегчает поддержку кода в долгосрочной перспективе.