Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

Высшего образования

*Факультет Программной Инженерии и Компьютерной Техники*

**Лабораторная работа 2 по АПС**

Группа: P3316

Выполнил:

Сиразетдинов А.Н.

Проверил:

Перл И. А.

Г. Санкт-Петербург

2024

Оглавление

[Задание 3](#_Toc187533318)

[Посетитель (visitor, GoF) 4](#_Toc187533319)

[Описание 4](#_Toc187533320)

[Примеры использования 4](#_Toc187533321)

[Фабричный метод (Factory Method, GoF) 5](#_Toc187533322)

[Описание 5](#_Toc187533323)

[Примеры использования 5](#_Toc187533324)

[Посредник (Indirection, grasp) 6](#_Toc187533325)

[Описание 6](#_Toc187533326)

[Примеры использования 6](#_Toc187533327)

# 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.

# Посетитель (visitor, GoF)

## Описание

Поведенческий паттерн проектирования, который позволяет добавлять новые операции к существующим объектам, не изменяя их классы. Паттерн достигается благодаря тому, что операции выносятся в отдельный объект — «посетителя», а сами объекты предоставляют метод для его принятия.

## Примеры использования

1. Есть информационная система сотрудников. Наша цель – построить дерево зарплат по отделам и сотрудникам. Имеется абстрактный класс и две реализации – сотрудник и отдел. Визитор будет иметь два разных метода под каждый класс, который будет в зависимости от вызванной функции по-разному производить операцию создания дерева и еще сохранять промежуточное состояние

Преимущества: чтобы создать дополнительный функционал, нам не нужно захламлять реализации абстрактного класса. Нам не требуются проверки и приведения типов

Ограничения: при добавлении новой реализации класса, нам нужно обновлять все визиторы, что проблематично для больших систем.

1. Информационная система – система для 2д моделирования. В ней можно создавать сложные фигуры из простых компонентов (квадраты, треугольники итп). И можно сделать большое количество операций с сложной фигурой – посчитать площадь, периметр, отрендерить ее

Преимущества: паттерн очень удобен, когда количество реализаций абстрактного класса конечно и мало, а количество вариантов визиторов огромен

Ограничения: будет проблематично хранить состояние для такой системы, ведь фигуры могут накладываться, скорее всего нужно будет не абстрагировать поведение а реализовывать оптимизированные алгоритмы

# Фабричный метод (Factory Method, GoF)

## Описание

порождающий паттерн проектирования, который предоставляет интерфейс для создания объектов в суперклассе, но позволяет подклассам изменять тип создаваемых объектов. Основная идея заключается в том, чтобы делегировать создание объектов дочерним классам.

## Примеры использования

1. Мы хотим сделать логгирование в информационной системе. Для этого мы можем сделать абстрактный Logger с методом log и абстрактный LoggerFactory который будет возвращать Logger. Затем в зависимости от требований мы можем реализовать несколько различных LoggerFactory, которые будут возвращать разные классы Logger, например JsonLogger, StdoutLogger, CsvLogger

Преимущества: удобно для расширения (в том числе внешними разработчиками), потому что они могут добавлять функциональность не изменяя основной код. Чтобы в приложении изменить логгер, будет достаточно изменить LoggerFactory

Ограничения: мы не можем изменять параметры конкретного логгера, потому что мы получаем его абстрактно, а для передачи параметра нужно их передавать в конструктор LoggerFactory.

1. Мы разрабатываем мультиплатформенный фреймворк для создания приложений, но сталкиваемся с проблемой, что кнопки в Android и Ios различаются по дизайну. Чтобы в реализации интерфейса не привязываться к операционной системе, мы будем использовать фабричный метод с фабрикой для конкретной ОС.

Преимущества: в коде мы не привязаны к ОС, и манипулируем абстрактной кнопкой

Ограничения: скорее всего абстрактная кнопка будет с очень сильно урезанным функционалом, потому что будет реализовывать только общие параметры кнопки операционных систем

# Посредник (Indirection, grasp)

## Описание

Поведенческий паттерн проектирования, который обеспечивает централизованное взаимодействие между множеством объектов. Вместо прямого взаимодействия объектов друг с другом, они общаются через посредника, что уменьшает их взаимные зависимости.

## Примеры использования

1. Информационная система – система сигнализации. У нас есть много датчиков, и есть системы оповещения. Чтобы абстрагировать зависимость систем оповещения от систем сигнализации мы вводим объект посредник который будет зависеть от датчиков и в случае срабатывания их будет вызывать системы оповещения

Преимущества: локализуем зависимости в одном классе, абстрагируем все системы оповещения

Ограничения: посредник может быть слишком объемным и потребуются дополнительные абстракции (бог-объект)

1. Информационнная система – фреймворк для ORM. Мы вводим посредник (EntityManager), который будет зависеть от DataSource и репозиториев. Это позволяет нам убрать зависимость репозиториев от датасурсов

Ограничения: посредник получается объемным, и для небольших информационных систем он может быть чрезмерно функциональным