

## ЛАБОРАТОРНА РОБОТА 4

Тема: Корпоративные технологии Java EE. Паттерны (шаблоны) проектирования

### Теоретическая часть

Паттерны (шаблоны) проектирования представляют собой хорошо понятные решения для распространенных проблем. Паттерны проектирования существуют в том или ином виде в каждой инженерной сфере деятельности. По-сути, паттерны проектирования представляют собой определенное описание обменивающихся информацией объектов и классов, которые адаптированы для решения общей проблемы проектирования в конкретных условиях и предназначены для решения распространенных проблем проектирования приложений. В объектно-ориентированном программировании паттерны проектирования обычно нацелены на решение задач, связанных с созданием и взаимодействием объектов, а не крупномасштабных задач, стоящих перед общей архитектурой программного обеспечения. Они обеспечивают обобщенные решения в виде шаблонов, которые могут быть применены к практическим задачам. Обычно паттерны проектирования наглядно представляют в виде диаграмм классов, демонстрирующих поведение классов и отношения между ними. Их можно разделить на три основных группы:

- Порождающие паттерны управляют созданием и инициализацией объекта, а также выбором класса;
- Паттерны поведения управляют связью, обменом сообщениями и взаимодействием между объектами;
- Структурные паттерны упорядочивают отношения между классами и объектами, обеспечивая критерии соединения и совместного использования связанных объектов для достижения желаемого поведения.

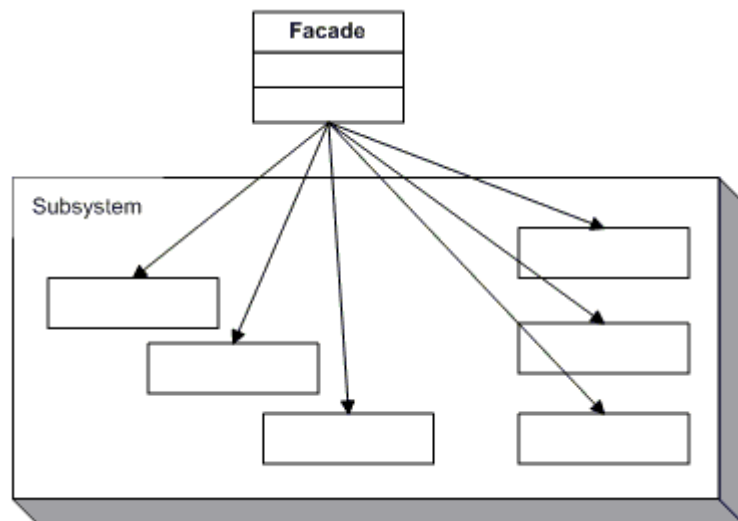
Паттерн «Фасад» - один из структурных паттернов проектирования. Его основной задачей является инкапсуляция сложной логики (бизнес-логики) в высокоуровневом интерфейсе, что облегчает использование обращения к подсистеме. Это часто осуществляется путем группировки связанных обращений к методам и вызова их последовательно из одного метода. Каждый API может рассматриваться как реализация паттерна «Фасад», так как обеспечивает простой интерфейс, скрывающий внутреннюю сложность. Любое обращение к одному из методов API приводит к вызову множества других методов из скрытой за ним подсистемы.

Паттерн «Фасад» предоставляет унифицированный интерфейс к множеству интерфейсов в некоторой подсистеме. Скрывая сложность подсистемы он предоставляет все возможности подсистемы через удобный для использования интерфейс.

Паттерн «Фасад» обычно реализуется в следующих целях и случаях:

- для обеспечения простого и унифицированного доступа к унаследованной системе управления производством;
- для создания общедоступного API к таким классам, как драйвер;
- для предоставления крупномодульного доступа к доступным сервисам, которые сгруппированы;
- для снижения количества сетевых вызовов. Фасад выполняет множество обращений к подсистеме, в то время как удаленный клиент должен выполнить одно-единственное обращение к фасаду;
- для инкапсуляции последовательности выполняемых действий и внутренних деталей приложения, чтобы обеспечить простоту и безопасность.

Диаграмма классов «Фасада» представлена на рисунке ниже и предоставляет простой интерфейс для базовой системы, инкапсулируя сложную логику.



Общая диаграмма классов паттерна «Фасад»

Реализация паттерна достаточно проста, так как не требует использования жесткой структуры или набора правил. Любой метод, обеспечивающий простой доступ к сложным последовательностям выполняемых действий, может считаться реализацией этого паттерна. Рассмотрим реализацию паттерна «Фасад» на примере стиральной машины, у которой две функции, зависящие от степени загрязненности вещей — сильная и слабая. Для каждого режима стиральная машина должна выполнить predetermined набор операций: установить температуру воды, нагреть воду, установить длительность цикла стирки, добавить стиральный порошок, добавить отбеливающее средство, добавить смягчитель ткани и т.д. Каждый режим требует различного набора инструкций (разное количество стирального порошка, более высокая/низкая температура, более долгий/короткий цикл отжима и т. д.). Простой интерфейс должен предоставить два режима стирки, скрывающих сложную выверенную логику выбора последовательности действий и величины параметров, тем самым предоставив пользователю определиться только со степенью загрязнения — сильная или слабая. Пример реализации паттерна «Фасад» применительно к описанной задаче:

```

public class WashingMachine {
    public void heavilySoiled(){
        setWaterTemperature(100);
        setWashCycleDuration(90);
        setSpinCycleDuration(10);
        addDetergent( );
        addBleach();
        addFabricSoftener();
        heatWater();
        startWash();
    }
    public void lightlySoiled(){
        setWaterTemperature(40);
        setWashCycleDuration(20);
        setSpinCycleDuration(10);
        addDetergent();
        heatWater();
        startWash();
    }
}

```

## Практическое задание

1. Используя шаблон фасад разработайте пример, скрывающий сложность математических вычислений (например: вычисление различных интегралов, функций и т.п.).
2. Используя шаблон фасад разработайте пример, скрывающий логику системы заказа и оплаты.
3. Приведите описание (анализ) разработанных примеров и демонстраций их использования в отчете.