**Лабораторная работа №3**

**Рефакторинг с использованием паттернов проектирования**

Одним из самых частых подходов при рефакторинге объектно-ориентированного кода является применение паттернов проектирования. Паттерн проектирования - повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста. Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

Существует несколько глобальных классов паттернов проектирования: порождающие, которые отвечают за безопасное и удобное создание новых объектов (в ряде случаев семейств объектов), структурные, чьей задачей является создание и поддержка иерархий классов максимально простым для программиста способом, и поведенческие, отвечающие за механизмы общения между классами. Более подробно с различными паттернами проектирования можно ознакомиться на сайте: <https://refactoring.guru/ru/design-patterns/catalog>

|  |  |
| --- | --- |
| Название | Краткое описание |
| Абстрактная фабрика | Создает семейство взаимосвязанных объектов |
| Адаптер | Преобразует интерфейс существующего класса к виду, подходящему для использования |
| Мост | Делает абстракцию и реализацию независимыми друг от друга |
| Строитель | Поэтапное создание сложного объекта |
| Цепочка обязанностей | Предоставляет способ передачи запроса по цепочке получателей |
| Команда | Инкапсулирует запрос в виде объекта |
| Компоновщик | Группирует схожие объекты в древовидные структуры |
| Декоратор | Динамически добавляет объекту новую функциональность |
| Фасад | Предоставляет унифицированный интерфейс вместо набора интерфейсов некоторой системы |
| Фабричный метод | Определяет интерфейс для создания объекта, при этом его тип определяется подклассами |
| Приспособленец | Использует разделение для поддержки множества мелких объектов |
| Интерпретатор | Для языка определяет его грамматику и интепретатор, использующий эту грамматику |
| Итератор | Предоставляет механизм обхода элементов коллекции |
| Посредник | Инкапсулирует взаимодействие между множеством объектов в объект-посредник |
| Хранитель | Сохраняет и восстанавливает состояние объекта |
| Пул объектов | Создание "затратных" объектов за счет их многократного использования |
| Наблюдатель | При изменении объекта извещает всех зависимые объекты для их обновления |
| Прототип | Создание объектов на основе прототипов |
| Заместитель | Подменяет другой объект для контроля доступа к нему |
| Одиночка | Создает единственный экземпляр некоторого класса и предоставляет к нему доступ |
| Состояние | Изменяет поведение объекта при изменении его состояния |
| Стратегия | Переносит алгоритмы в отдельную иерархию классов, делая их взаимозаменяемыми |
| Шаблонный метод | Определяет шаги алгоритма, позволяя подклассам изменить некоторые из них |
| Посетитель | Определяет новую операцию в классе без его изменения |

**Задание.**

В файле Source.cpp приведен код программы, который необходимо переписать с использованием паттернов проектирования (должно использоваться от двух и более различных паттернов).