**Лабораторная работа №5**

**Задание Пиццерия**

Есть пиццерия и в ней есть разные пиццы с добавлением различных добавок. В зависимости от пиццы и комбинации добавок она будет месть различную стоимость. В качестве абстрактный класс Pizza, которая определяет базовую функциональность в виде свойства Name и GetCost(). Это функциональность реализуется двумя подклассами ItalianPizza и AmericanPizza, в которой есть название пиццы и ее цена.

Декоратором является абстрактный класс PizzaDecorator, который унаследован от класса Pizza и содержит ссылку на декорируемый объект Pizza. Установка декорируемого объекта происходит в конструкторе.

Отдельные функциональности – добавление томатов и сыра к пиццам реализован через одноименный класс, т.е перенаправляя метод GetCost и изменяя значение свойства Name.

Благодаря этому при создании пиццы с добавками произойдёт ее обертывание декоратором.

Если бы мы использовали наследование, то для двух пицц с двумя добавками пришлось бы создавать 8 различных классов, которые бы описывали все возможные комбинации. Декоратор более предпочтителен.

**Об паттерне декоратор (обёртка)**

Приставляет структурный шаблон проектирования, позволяет динамически подключать к объектам дополнительную информацию (функциональность).

Следует использовать, когда надо добавлять к объекту новые дополнительные данные. При этом мы можем их снять. Так же когда наследование глупо или неприемлемо использовать.

Опишу немного другие структурные шаблоны:

**Адаптер** - предназначен для преобразования интерфейса одного класса в интерфейс другого. Благодаря реализации данного паттерна мы можем использовать вместе классы с несовместимыми интерфейсами.

**Мост** - структурный шаблон проектирования, который позволяет отделить абстракцию от реализации таким образом, чтобы и абстракцию, и реализацию можно было изменять независимо друг от друга.

**Компоновщик** - объединяет группы объектов в древовидную структуру по принципу "часть-целое и позволяет клиенту одинаково работать как с отдельными объектами, так и с группой объектов.

**Фасад** - представляет шаблон проектирования, который позволяет скрыть сложность системы с помощью предоставления упрощенного интерфейса для взаимодействия с ней.

**Приспособленец** - структурный шаблон проектирования, который позволяет использовать разделяемые объекты сразу в нескольких контекстах. Данный паттерн используется преимущественно для оптимизации работы с памятью. (возможно но декоратор будет лучше)

**Заместитель** - предоставляет объект-заместитель, который управляет доступом к другому объекту. То есть создается объект-суррогат, который может выступать в роли другого объекта и замещать его.

**Для нашей задачи очевидно, что подходи только декоратор.**