УО «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра информационных систем и технологий

**Лабораторная работа № 2**

Применение структурных паттернов проектирования

**Выполнила:**   
Студентка II курса

Хохрякова А.С.

Минск 2020

Ответы на вопросы:

1. Назначение структурных паттернов?

Решают задачи компоновки системы наоснове классов и объектов.

Структурные паттерны показывают различные способы построения связей между объектами.

2. Нарисуете диаграмму классов и поясните принцип работы паттерна Adapter. В каких случаях надо его применять.

Преобразует интерфейс одного класса в интерфейс другого, который ожидают клиенты.

•Когда необходимо использовать имеющийся класс, но его интерфейс не соответствует потребностям;

•Когда надо использовать уже существующий класс совместно с другими классами, интерфейсы которых не совместимы;

3. В чем разница между адаптером класса и адаптером объекта?

•Адаптер Класс использует наследование и может обернуть только класс. Он не может обернуть интерфейс,

так как по определению он должен быть производным от некоторого базового класса.

•Объектный Адаптер использует состав и может обернуть классы или интерфейсы, или оба.

Он может сделать это, так как он содержит, как частный, инкапсулированный член, класс или интерфейс объект она оборачивается.

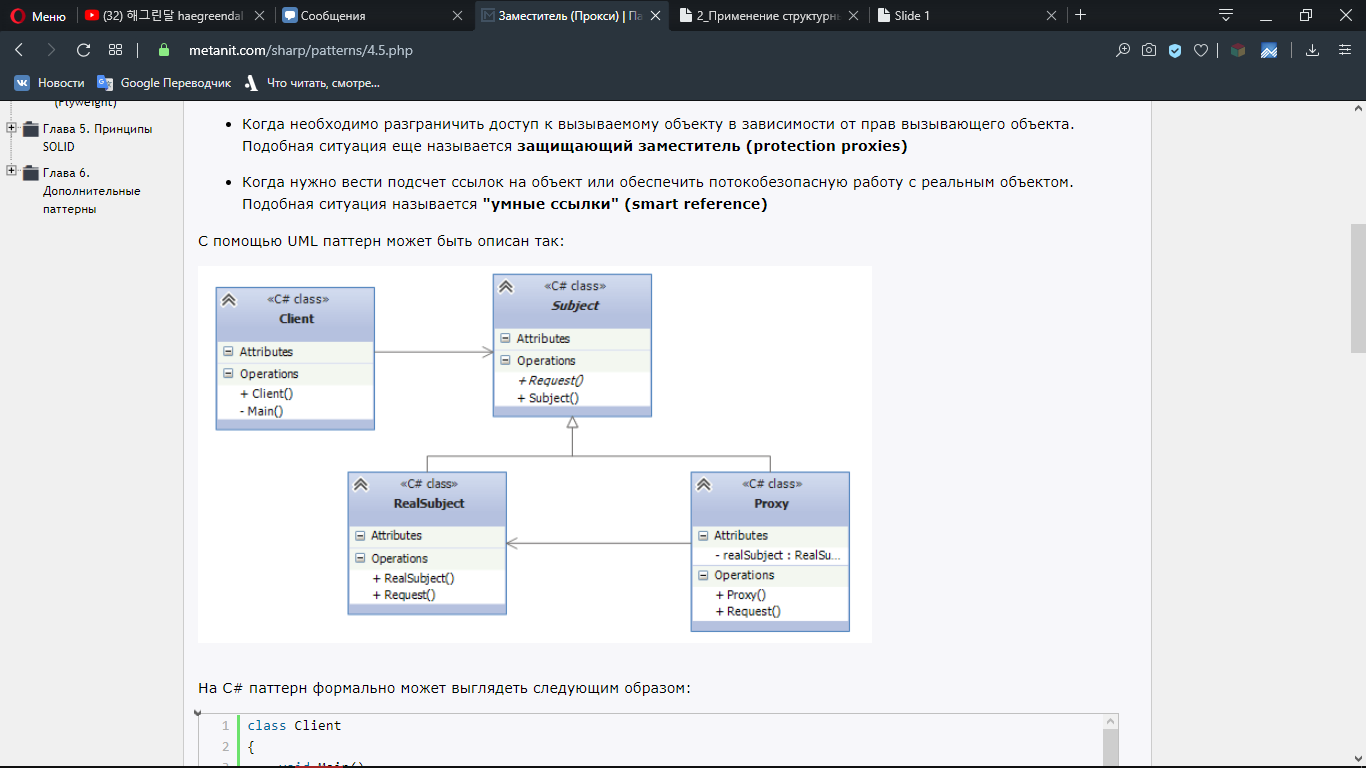
4. Назначение и принцип организации паттерна Декоратор.

Назначение: добавляет к объекту новую функциональность – новое состояние или поведение. Является динамической альтернативой наследованию.

Декоратор – обёртка для декорируемого объекта и реализует тот же интерфейс, что и класс декорируемого объекта.

Декоратор добавляет свой код до, после или вместо вызовов методов декорируемого объекта.

5. Нарисуйте диаграмму классов для паттерна Proxy. Поясните его назначение. Какие разновидности proxy вы знаете?



Заместитель — это структурный паттерн проектирования, который позволяет подставлять вместо реальных объектов специальные объекты-заменители.

Эти объекты перехватывают вызовы к оригинальному объекту, позволяя сделать что-то до или после передачи вызова оригиналу.

Разновидности:

•Удалённый заместитель (remote proxy) обеспечивает связь с замещаемым объектом, который находится в другом адресном пространстве.

•Виртуальный заместитель (virtual proxy) реализует создание замещаемого объекта только тогда, когда он действительно необходим.

•Защищающий заместитель (protection proxy) проверяет, имеет ли вызывающий объект необходимые для выполнения запроса права.

6. В чем разница между паттернами Decorator и Adapter?

•Декоратор также оборачивает некоторый класс и предоставляет такой же или расширенный интерфейс.

Иногда декоратор называют "умным заместителем" (smart proxy).

Т.е. декоратор может притворяться оригинальным классом и при этом расширять его функциональность.

•Адаптер также оборачивает некоторый класс, но при этом предоставляет другой интерфейс.

Т.е. используется в случаях, когда есть класс с нужными данными и поведением, но с неподходящим интерфейсом.

7. В чем суть паттерна Composite? Поясните в каких случаях его надо применять?

Компоновщик — это структурный паттерн проектирования, который позволяет сгруппировать множество объектов в древовидную структуру,

а затем работать с ней так, как будто это единичный объект.

Use case:

•нужно представить иерархию объектов вида часть-целое;

•хотите, чтобы клиенты единообразно

трактовали составные и индивидуальные

объекты.

пример: кнопка-визуальный компонентформа; XML-элемент и XML-документ,

папка- файл

8. Расскажите о паттерне Facade?

Фасад — это структурный паттерн проектирования, который предоставляет простой интерфейс к сложной системе классов, библиотеке или фреймворку.

Предоставляет унифицированный интерфейс вместо набора интерфейсов некоторой подсистемы.

Фасад определяет интерфейс более высокого уровня, который упрощает использование подсистемы.

9. В каких случаях надо применять Bridge? Поясните на диаграмме классов как он применятся

Мост — это структурный паттерн проектирования, который разделяет один или несколько классов на две отдельные иерархии — абстракцию и реализацию,

позволяя изменять их независимо друг от друга.