1. **Назначение структурных паттернов?**

Решают задачи компоновки системы на основе классов и объектов.

механизмы: Наследование и Композиция

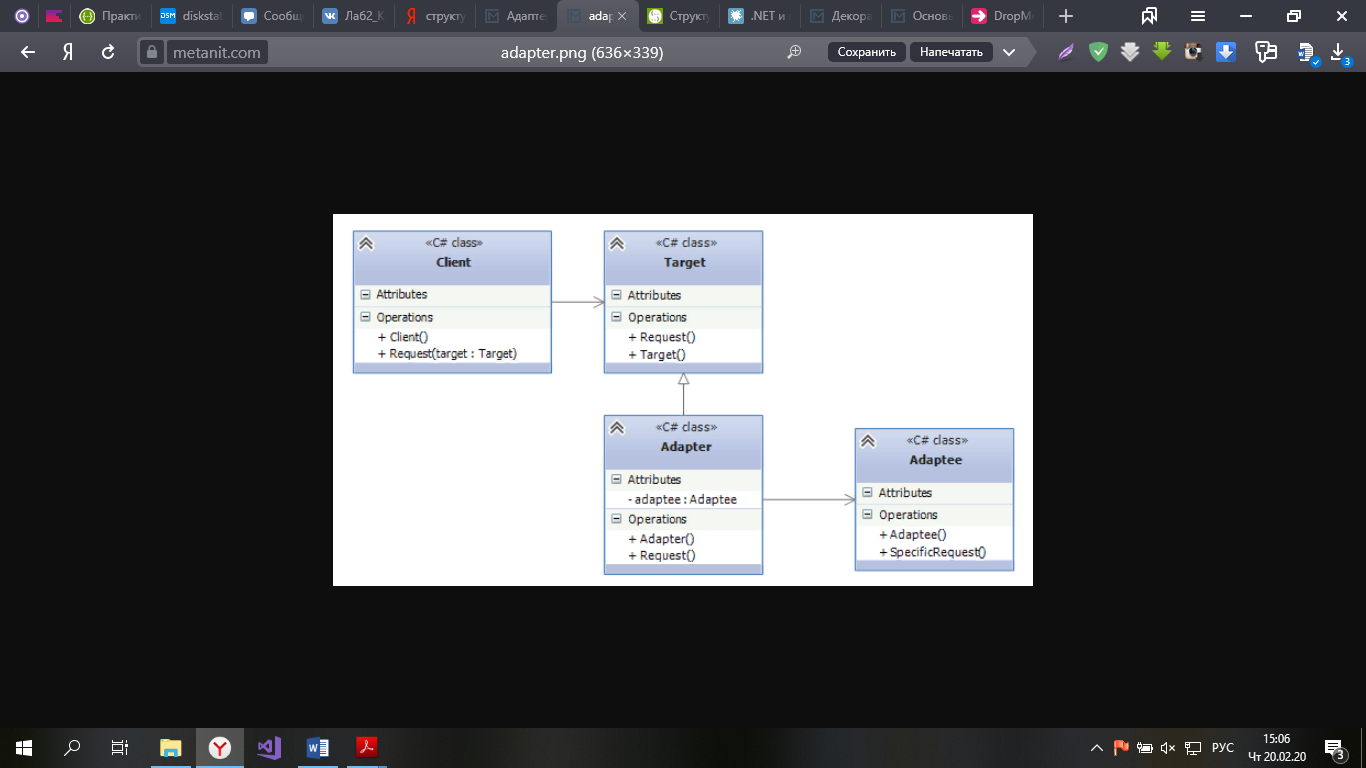
Компонует объекты в древовидные структуры для представления иерархий «часть — целое». Позволяет клиентам единообразно трактовать индивидуальные и составные объекты.

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

Паттерн Адаптер (Adapter) предназначен для преобразования интерфейса одного класса в интерфейс другого. Благодаря реализации данного паттерна мы можем использовать вместе классы с несовместимыми интерфейсами.

**Когда надо использовать Адаптер?**

* 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



1. **В чем разница между адаптером класса и адаптером объекта?**

Класс адаптера является наследником адаптируемого класса.

Класс адаптера агрегирует адаптируемый объект.

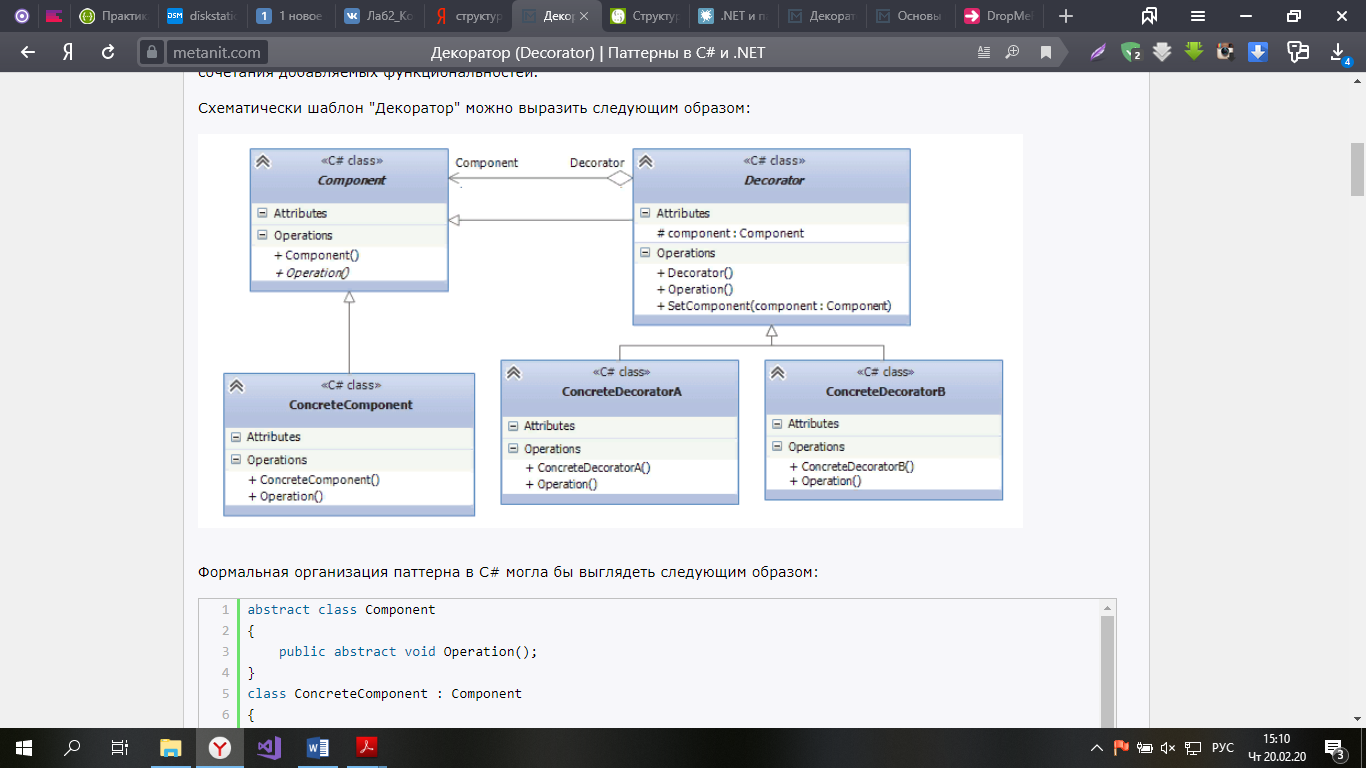
**4. Назначение и принцип организации паттерна Декоратор.**

Декоратор (Decorator) позволяет динамически подключать к объекту дополнительную функциональность.

Для определения нового функционала в классах нередко используется наследование. Декораторы же предоставляет наследованию более гибкую альтернативу, поскольку позволяют динамически в процессе выполнения определять новые возможности у объектов.

**Когда следует использовать декораторы?**

* Когда надо динамически добавлять к объекту новые функциональные возможности. При этом данные возможности могут быть сняты с объекта
* Когда применение наследования неприемлемо. Например, если нам надо определить множество различных функциональностей и для каждой функциональности наследовать отдельный класс, то структура классов может очень сильно разрастись. Еще больше она может разрастись, если нам необходимо создать классы, реализующие все возможные сочетания добавляемых функциональностей.

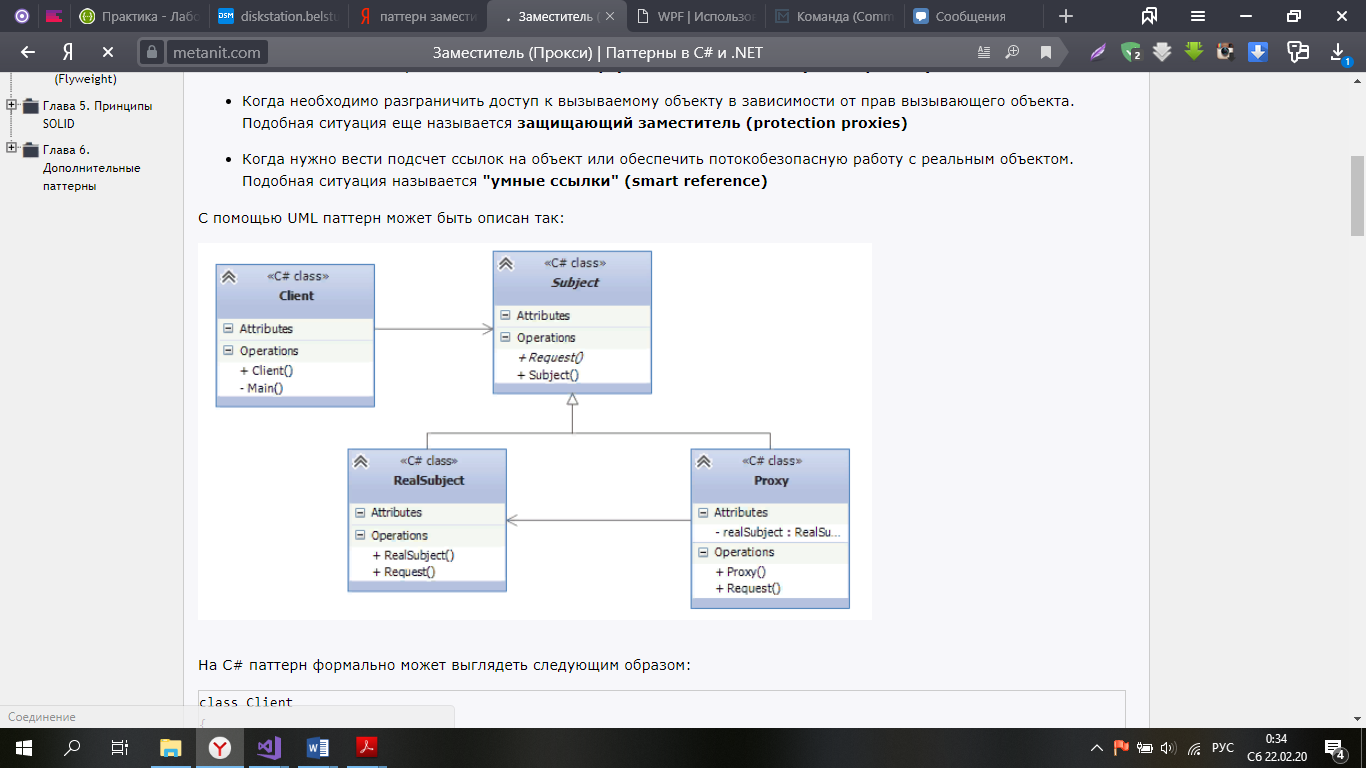


1. **Нарисуйте диаграмму классов для паттерна Proxy. Поясните его назначение. Какие разновидности proxy вы знаете?**

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

**Когда использовать прокси?**

* Когда надо осуществлять взаимодействие по сети, а объект-проси должен имитировать поведения объекта в другом адресном пространстве. Использование прокси позволяет снизить накладные издержки при передачи данных через сеть. Подобная ситуация еще называется удалённый заместитель (remote proxies)
* Когда нужно управлять доступом к ресурсу, создание которого требует больших затрат. Реальный объект создается только тогда, когда он действительно может понадобится, а до этого все запросы к нему обрабатывает прокси-объект. Подобная ситуация еще называется виртуальный заместитель (virtual proxies)
* Когда необходимо разграничить доступ к вызываемому объекту в зависимости от прав вызывающего объекта. Подобная ситуация еще называется защищающий заместитель (protection proxies)
* Когда нужно вести подсчет ссылок на объект или обеспечить потокобезопасную работу с реальным объектом. Подобная ситуация называется "умные ссылки" (smart reference)



**6. В чем разница между паттернами Decorator и Adapter?**

**Decorator:**

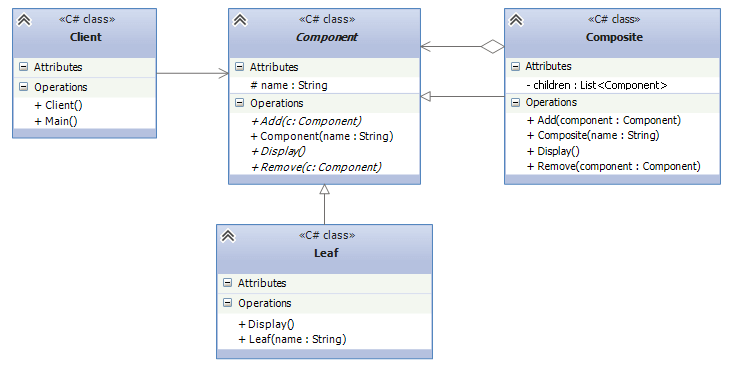
Позволяет объектам быть составленными/добавлять возможности, оборачивая их классом с тем же интерфейсом

**Aдаптер:**

Позволяет вам обернуть объект без известной реализации интерфейса, чтобы он придерживался интерфейса. Дело в том, чтобы "перевести" один интерфейс в другой.

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

Компоновщик — это относительно низкоуровневый паттерн проектирования, который лежит в основе других паттернов. Команды объединяются в составные команды, декоратор является составным объектом с одним дочерним элементом, посетитель очень часто обходит составные объекты иерархической формы.



**Когда использовать компоновщик?**

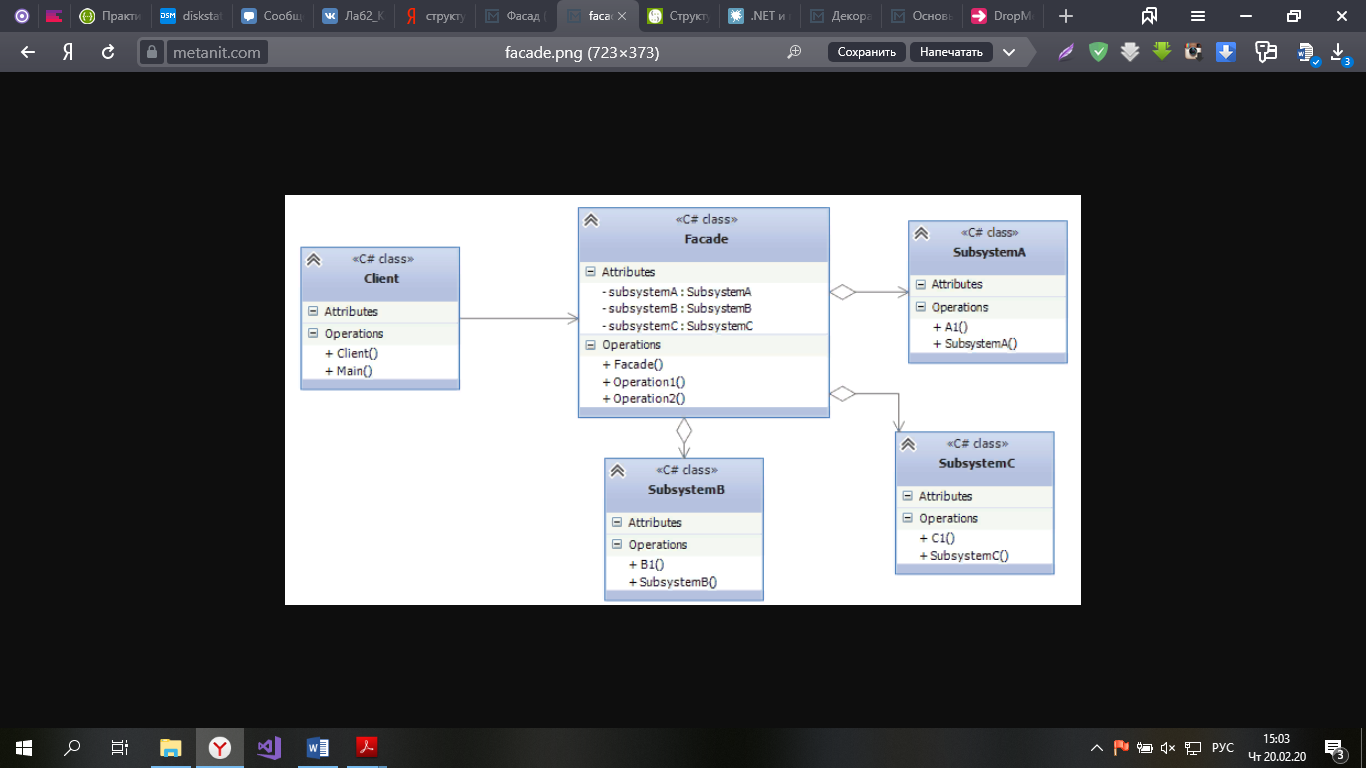
* Когда объекты должны быть реализованы в виде иерархической древовидной структуры
* Когда клиенты единообразно должны управлять как целыми объектами, так и их составными частями. То есть целое и его части должны реализовать один и тот же интерфейс

**8. Расскажите о паттерне Façade?**

Фасад (Facade) позволяет скрыть сложность системы с помощью предоставления упрощенного интерфейса для взаимодействия с ней.

**Когда использовать фасад?**

* Когда имеется сложная система, и необходимо упростить с ней работу. Фасад позволит определить одну точку взаимодействия между клиентом и системой.
* Когда надо уменьшить количество зависимостей между клиентом и сложной системой. Фасадные объекты позволяют отделить, изолировать компоненты системы от клиента и развивать и работать с ними независимо.
* Когда нужно определить подсистемы компонентов в сложной системе. Создание фасадов для компонентов каждой отдельной подсистемы позволит упростить взаимодействие между ними и повысить их независимость друг от друга.



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

Мост (Bridge) позволяет отделить абстракцию от реализации таким образом, чтобы и абстракцию, и реализацию можно было изменять независимо друг от друга.

Даже если мы отделим абстракцию от конкретных реализаций, то у нас все равно все наследуемые классы будут жестко привязаны к интерфейсу, определяемому в базовом абстрактном классе. Для преодоления жестких связей и служит паттерн Мост.

**Когда использовать данный паттерн?**

* Когда надо избежать постоянной привязки абстракции к реализации
* Когда наряду с реализацией надо изменять и абстракцию независимо друг от друга. То есть изменения в абстракции не должно привести к изменениям в реализации

Общая реализация паттерна состоит в объявлении классов абстракций и классов реализаций в отдельных параллельных иерархиях классов.

