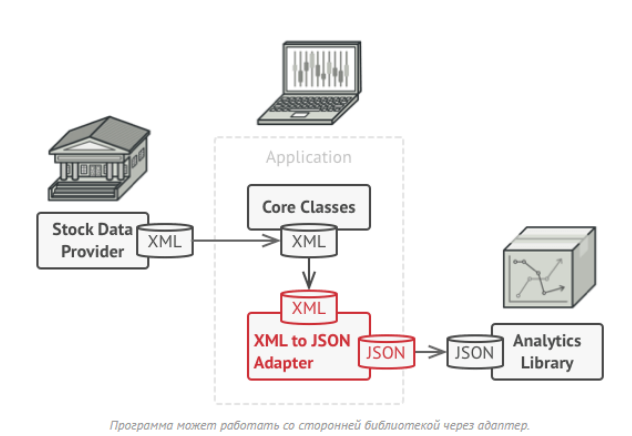
1. Назначение структурных паттернов?

решают задачи компоновки системы на основе классов и объектов

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



есть сторонняя библиотека кода. Классы из этой библиотеки не обладают необходимыми для клиента интерфейсами. Нужно адаптировать классы для использования клиентом.

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

Класс адаптера класса является наследником адаптируемого класса

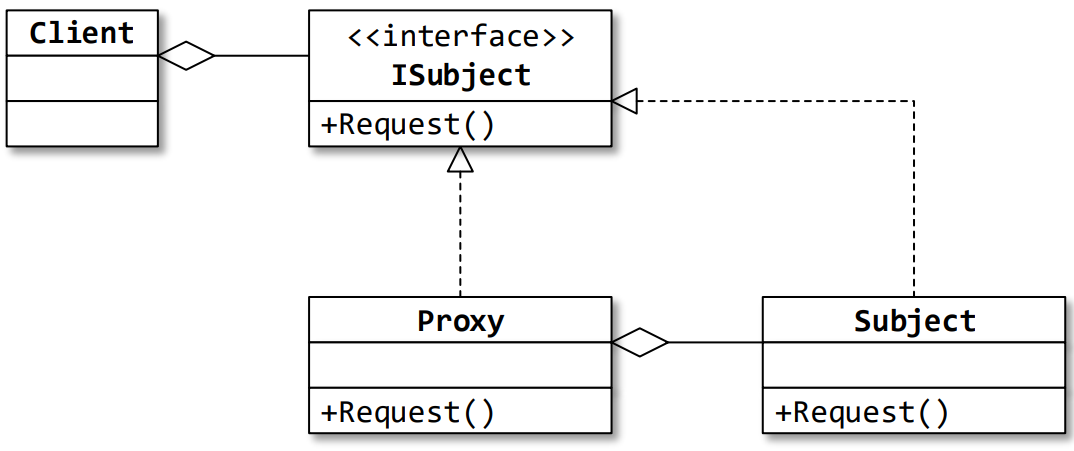
Класс адаптера объекта агрегирует адаптируемый объект.

1. Назначение и принцип организации паттерна Декоратор.

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.

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



суррогат или заместитель другого объекта и контролирует доступ к нему –> контролирует доступ к заданному объекту, перехватывая все вызовы к этому объекту предоставляет дополнительный уровень косвенности (распределенного, управляемого или интеллектуального доступа). являясь "оберткой" реального компонента, защищает его от излишней сложности.

* 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.

1. В чем разница между паттернами Decorator и Adapter?

Декоратор добавляет к классу новые возможности, адаптер обеспечивает взаимодействие классов с различными интерфейсами

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

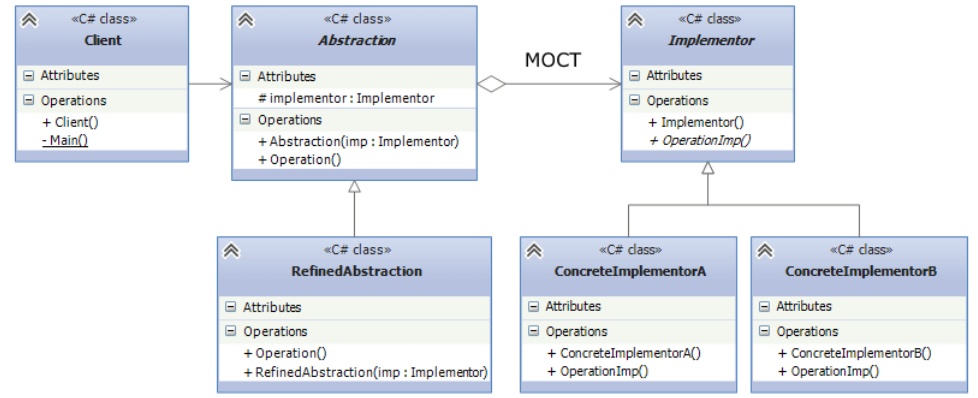
Паттерн Компоновщик (Composite) объединяет группы объектов в древовидную структуру по принципу "часть-целое и позволяет клиенту одинаково работать как с отдельными объектами, так и с группой объектов.

* 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

1. Расскажите о паттерне Facade?

Фасад (Facade) представляет шаблон проектирования, который позволяет скрыть сложность системы с помощью предоставления упрощенного интерфейса для взаимодействия с ней.

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



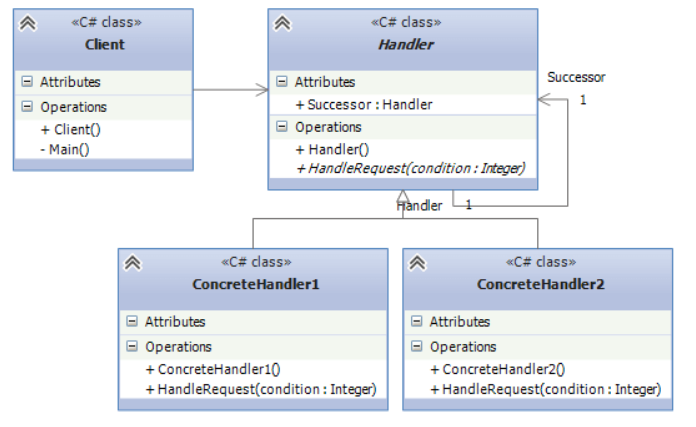
* Когда надо избежать постоянной привязки абстракции к реализации
* Когда наряду с реализацией надо изменять и абстракцию независимо друг от друга. То есть изменения в абстракции не должно привести к изменениям в реализации

1. Назначение паттернов поведения?

Паттерны поведения (behavioral patterns) используются для организации, управления и объединения различных вариантов поведения объектов.

1. Нарисуете диаграмму классов и поясните принцип работы паттерна Chain of responsibility. В каких случаях надо его применять?

Цепочка Обязанностей (Chain of responsibility) - поведенческий шаблон проектирования, который позволяет избежать жесткой привязки отправителя запроса к получателю. Все возможные обработчики запроса образуют цепочку, а сам запрос перемещается по этой цепочке. Каждый объект в этой цепочке при получении запроса выбирает, либо закончить обработку запроса, либо передать запрос на обработку следующему по цепочке объекту.



* Когда имеется более одного объекта, который может обработать определенный запрос
* Когда надо передать запрос на выполнение одному из нескольких объект, точно не определяя, какому именно объекту
* Когда набор объектов задается динамически

1. Назначение и принцип организации паттерна Command. Поясните как он связан с конечными автоматами

Паттерн "Команда" (Command) позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. Этот объект запроса на действие и называется командой. При этом объекты, инициирующие запросы на выполнение действия, отделяются от объектов, которые выполняют это действие.

1. Как реализовать паттерн Observer?

interface IObservable

{

void AddObserver(IObserver o);

void RemoveObserver(IObserver o);

void NotifyObservers();

}

class ConcreteObservable : IObservable

{

private List<IObserver> observers;

public ConcreteObservable()

{

observers = new List<IObserver>();

}

public void AddObserver(IObserver o)

{

observers.Add(o);

}

public void RemoveObserver(IObserver o)

{

observers.Remove(o);

}

public void NotifyObservers()

{

foreach (IObserver observer in observers)

observer.Update();

}

}

interface IObserver

{

void Update();

}

class ConcreteObserver :IObserver

{

public void Update()

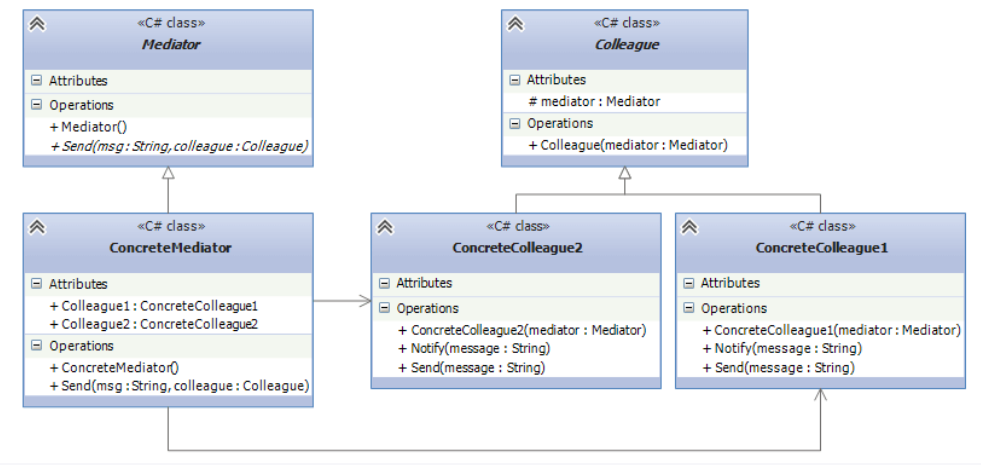
{

}

}

1. Нарисуйте диаграмму классов для паттерна Mediator. Поясните его назначение

представляет такой шаблон проектирования, который обеспечивает взаимодействие множества объектов без необходимости ссылаться друг на друга. Тем самым достигается слабосвязанность взаимодействующих объектов.



1. В чем разница между паттернами Mediator и Facade?

Фасад используется для отделения клиента от реализации, посредник – при большом количестве связей между объектами

1. В чем суть паттерна Memento? Поясните на примере.

Паттерн Хранитель (Memento) позволяет выносить внутреннее состояние объекта за его пределы для последующего возможного восстановления объекта без нарушения принципа инкапсуляции.

1. Расскажите о паттерне Visitor?

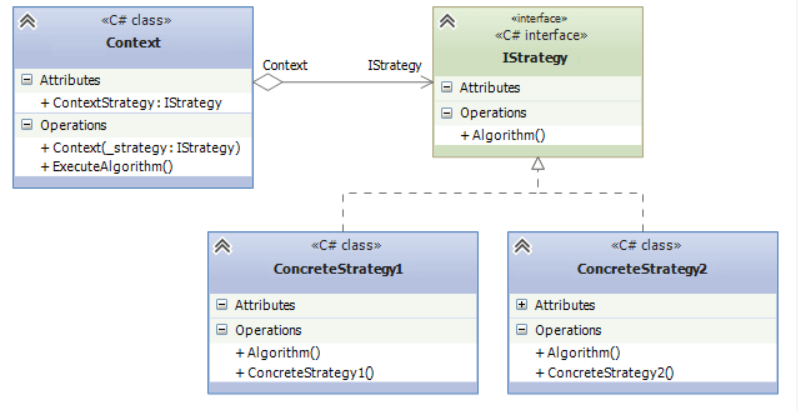
Паттерн Посетитель (Visitor) позволяет определить операцию для объектов других классов без изменения этих классов.

При использовании паттерна Посетитель определяются две иерархии классов: одна для элементов, для которых надо определить новую операцию, и вторая иерархия для посетителей, описывающих данную операцию.

1. В каких случаях надо применять Null object?

Нулевой объект – это объект с заданным нейтральным поведением, использование которого позволяет заменить условные операторы (проверки на null) полиморфизмом.

1. Поясните на диаграмме классов как реализовать Strategy.



1. Перечислите и поясните принципы проектирования SOLID.

Принцип единственной ответственности

The Single Responsibility Principle (SRP)

Каждый компонент должен иметь одну и только одну причину для изменения

Принцип открытости/закрытости

The Open Closed Principle (OCP)

Сущности программы должны быть открыты для расширения, но закрыты для изменения.

Принцип подстановки Лисков

The Liskov Substitution Principle (LSP)

класс S может считаться подклассом T, если замена объектов T на объекты S не приведет к изменению работы программы.

Должна быть возможность вместо базового типа подставить любой его подтип.

Принцип разделения интерфейса

The Interface Segregation Principle (ISP)\

Клиенты не должны вынужденно зависеть от методов, которыми не пользуются.

Принцип инверсии зависимостей

The Dependency Inversion Principle (DIP)

Модули верхнего уровня не должны зависеть от модулей нижнего уровня. И те и другие должны зависеть от абстракций.

Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.