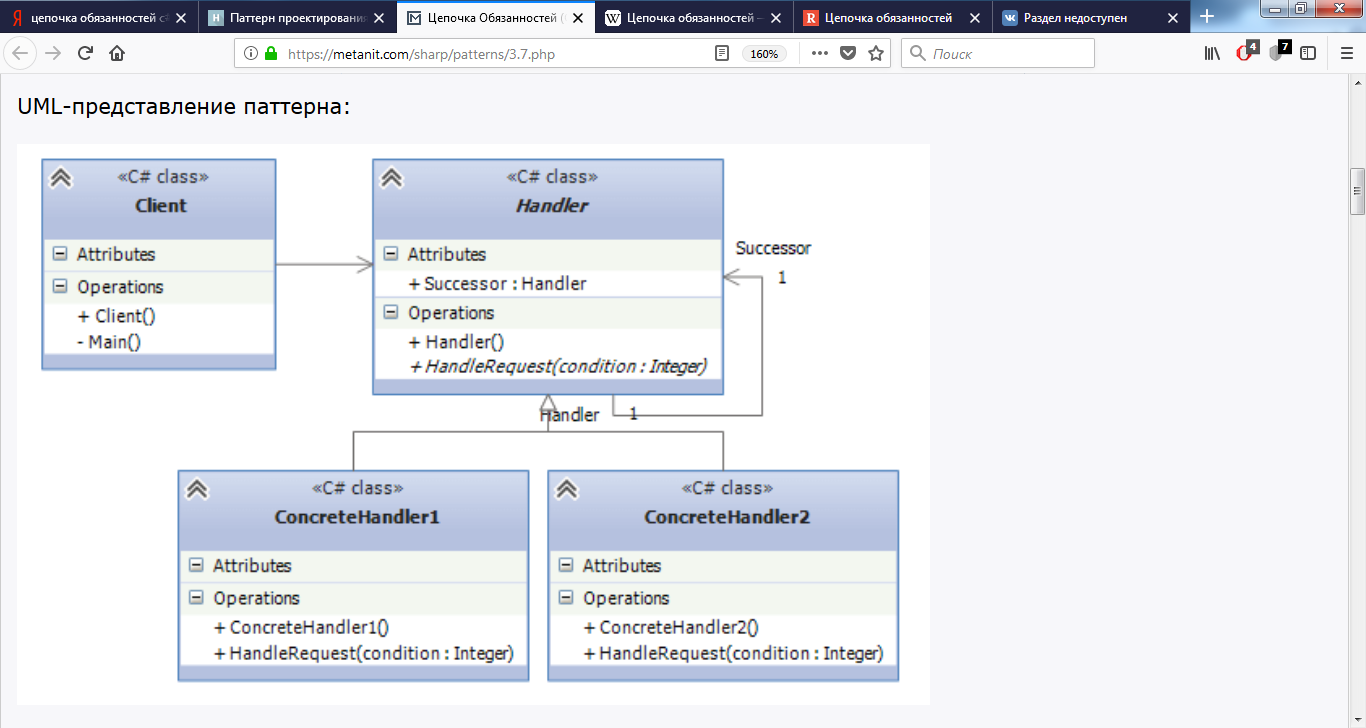
**Кукаркин И.А.**

**ПИ-15-1**

**Цепочка обязанностей**

*Цель применения паттерна* - эффективно и компактно реализовать механизм обработки потока событий/запросов/сообщений в системах с потенциально большим количеством обработчиков.

*UML-диаграмма:*



Handler: определяет интерфейс для обработки запроса. Также может определять ссылку на следующий обработчик запроса

ConcreteHandler1 и ConcreteHandler2: конкретные обработчики, которые реализуют функционал для обработки запроса. При невозможности обработки и наличия ссылки на следующий обработчик, передают запрос этому обработчику

В данном случае для простоты примера в качестве параметра передается некоторое число, и в зависимости от значения данного числа обработчики и принимают решения об обработке запроса.

Client: отправляет запрос объекту Handler.

*Используется, когда:*

* Когда программа содержит несколько объектов, способных обработать тот или иной запрос, однако заранее неизвестно какой запрос придёт и какой обработчик понадобится
* Когда важно, чтобы обработчики выполнялись один за другим в строгом порядке
* Когда набор объектов, способных обработать запрос, должен задаваться динамически.

*Особенности паттерна:*

Достоинства:

* Уменьшает зависимость между клиентом и обработчиками
* Реализует принцип единственной обязанности.
* Реализует принцип открытости/закрытости

Недостатки:

* Запрос может остаться никем не обработанным.

*Пример реализации на c#:*

class Client

{

    void Main()

    {

        Handler h1 = new ConcreteHandler1();

        Handler h2 = new ConcreteHandler2();

        h1.Successor = h2;

        h1.HandleRequest(2);

    }

}

abstract class Handler

{

    public Handler Successor { get; set; }

    public abstract void HandleRequest(int condition);

}

class ConcreteHandler1 : Handler

{

    public override void HandleRequest(int condition)

    {

        if (condition == 1)

        {

            // обработка;

        }

        else if (Successor != null)

        {

            Successor.HandleRequest(condition);

        }

    }

}

class ConcreteHandler2 : Handler

{

    public override void HandleRequest(int condition)

    {

        if (condition==2)

        {

            // обработка;

        }

        else if (Successor != null)

        {

            Successor.HandleRequest(condition);

        }

    }

}