**Паттерны проектирования**

**ИСиТ,ПОИБМС-2**

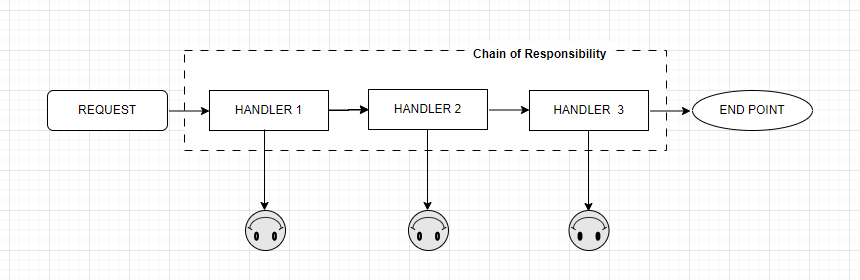
**Лекция 8**

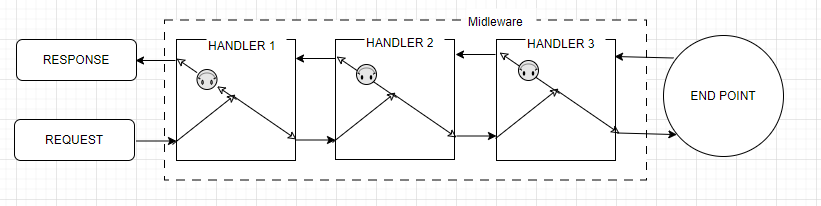
**Поведенческие паттерны**

1. **Поведенческие паттерны:** взаимодействие программ.
2. **Поведенческие паттерны:** Chain of Responsibility (цепочка обязанностей), Command (Команда), Iterator (Итератор), Mediator (Посредник), Memento (Снимок), Observer (Обозреватель), State (Состояние), Strategy (Стратегия), Template method (шаблонный метод), Visitor (Посетитель).

**Chain of Responsibility**

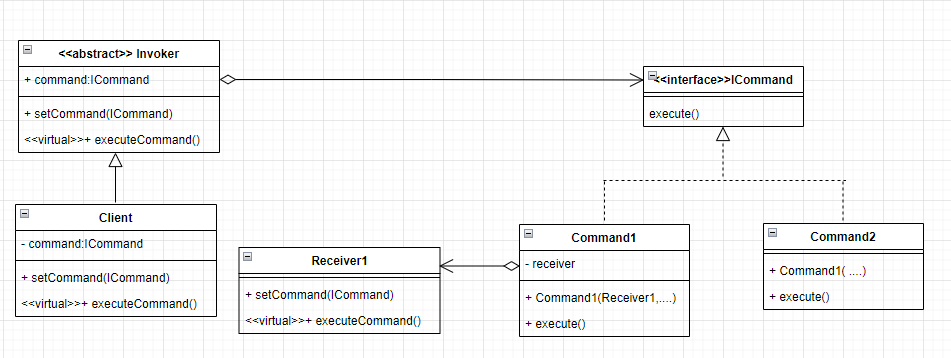
1. **Chain of Responsibility**: цепочка обработчиков запроса.

****

****

**Command**

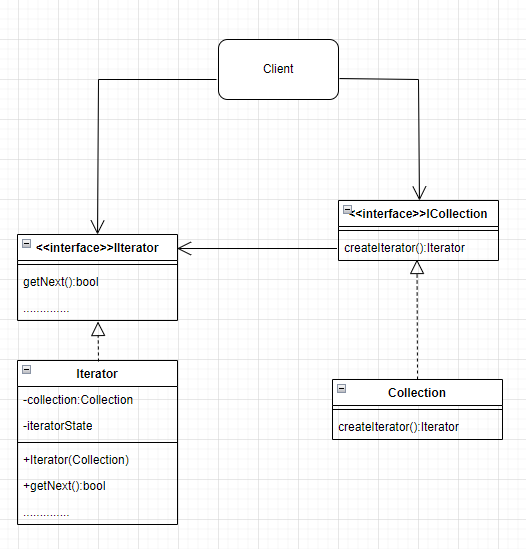
1. **Command:** превращение действия (исполняемого кода) в объект (команду).



1. **Command:** библиотека, которая предлагает набор классов Command1, Command2,…., CommandN (реализуют интерфейс ICommand), Receiver1, …(необязательные классы) и абстрактный класс Invoker; пользователь библиотеки должен разрабатывать свои классы, которые наследуют класс Invoker.

**Iterator**

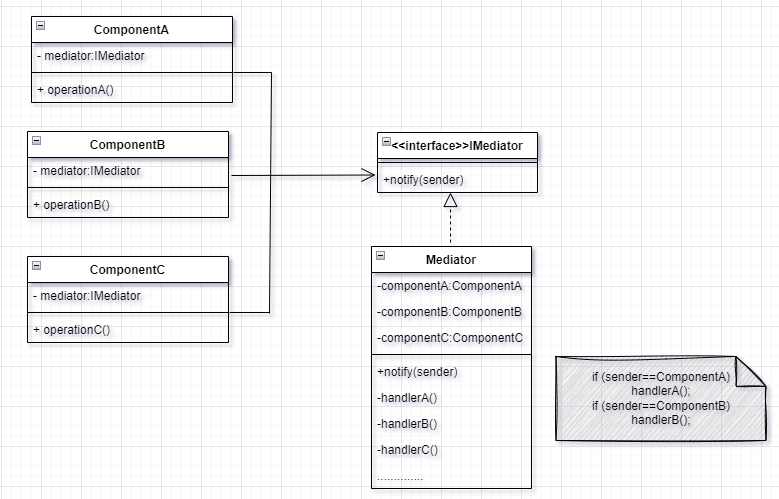
1. **Iterator:** последовательная обработка коллекции объектов.

****

1. **Iterator:** библиотека, которая предлагает наборы классов Collection1, Collection2,… (реализуют интерфейс ICollection) и Iteator1, Iterator2 (реализуют интерфейс IIterator); пользователь создает объекты класса Collection, с помощью метода createIterator (объекта Collection) создает объект класса Iterator, который позволяет последовательно перебирать объекты коллекции.

**Mediator**

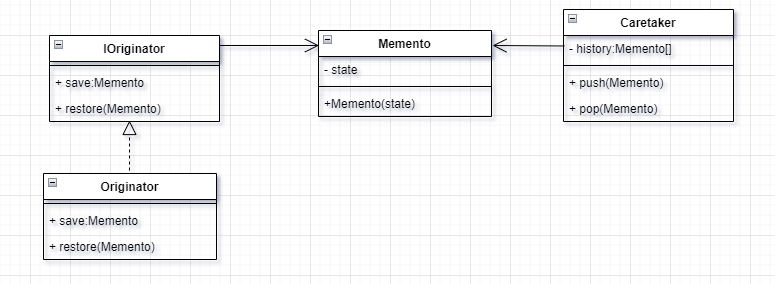
1. **Mediator:** снижение связанности классов, замена многих связей на одну

****

1. **Mediator:** библиотека предлагает набор классов ComponentA, ComponentB, … и интерфейс IMediator; пользователь библиотеки создает класс Mediator, реализующий интерфейс IMediator и при создании объектов классов ComponentA, ComponentB,… передает объект Mediator в качестве параметра конструктора; объекты ComponentA, ComponentB,… при определенных событиях уведомляют Mediator, Mediator вызывает необходимые методы объектов ComponentA, ComponentB,….

**Memento**

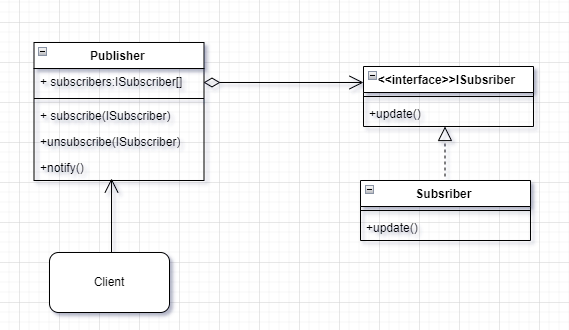
1. **Memento:** сохранение и восстановление состояния объектов

****

1. **Memento:** библиотека предоставляет классы Memento, Caretaker и интерфейс IOriginator; пользователь может создавать классы реализующие интерфейс IOriginator (классы объектов имеющие одинаковую структуру состояния), объекты Memento(Originator.save)и объекты Caretaker (стек состояний).

**Observer**

1. **Observer:**механизм издатель/подписчик(publisher/subscriber)

****

1. **Memento:** библиотека предоставляет классы Publisher и интерфейс ISubsriber; пользователь может создавать объекты классов Publisher и объекты (Subscriber) реализующие интерфейс ISubscriber; объекты Subscriber могут подписываться на события в объектах Publisher.
2. **КОНЕЦ**