top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

{ПРОГРАММИРОВАНИЕ}

ПАТТЕРНЫ
ПРОЕКТИРОВАНИЯ

ПОРОЖДАЮЩИЕ ПАТТЕРНЫ
ПРОЕКТИРОВАНИЯ

СТРУКТУРНЫЕ
ПАТТЕРНЫ

ПАТТЕРНЫ
ПОВЕДЕНИЯ

Урок №3

Паттерны поведения

Содержание

1. Понятие паттерна поведения

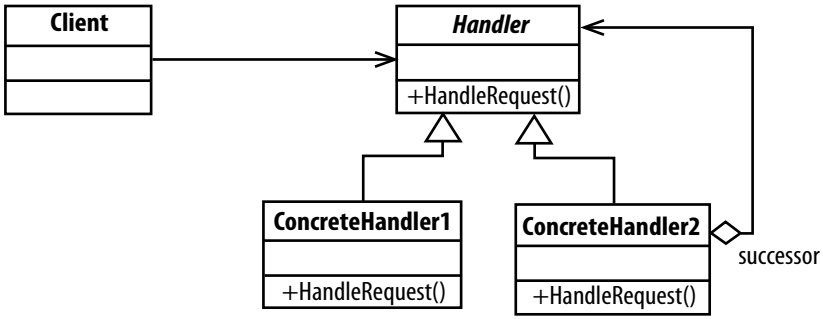
Паттерны поведения (поведенческие паттерны) как видно из названия служат для управления различными вариантами поведения системы объектов (классов). В этом уроке мы рассмотрим некоторые из данных паттернов. В проекте, который идет с уроком вы найдете код всех паттернов.

2. Паттерн Chain Of Responsibility

Данный паттерн предназначен для того чтобы позволять объекту отправлять команду, не имея информации об объекте(-ах), получающих ее. Важно отметить, что команда передается группе объектов, которая часто является частью более крупной структуры.

Каждый объект цепочки может обрабатывать, передавать полученную команду следующему объекту в цепи или делать и то, и другое.

Рассмотрим UML диаграмму для данного паттерна:



В данной диаграмме следующие участники:

Handler

- Определяет интерфейс для обработки запросов.

ConcreteHandler

- Обрабатывает запрос, предназначенный конкретному исполнителю.

- Если ConcreteHandler может обработать запрос он это делает, иначе запрос переадресовывается отправителю.

Client (*ChainApp*)

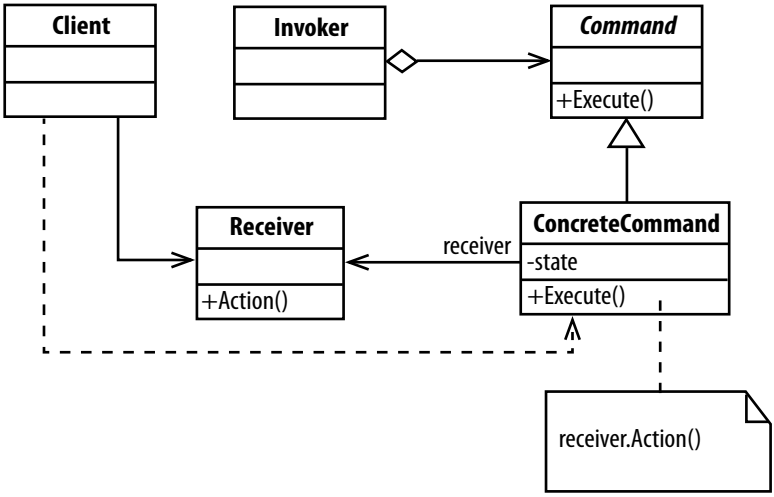
- Инициирует запрос к объекту ConcreteHandler, находящемуся в цепочке.

Пример кода, который демонстрирует работу паттерна прикреплен к pdf-файлу данного урока.

3. Паттерн Command

Паттерн Command инкапсулирует команды в некотором объекте. Инкапсулирование, таким образом, позволяет выполнять различные манипуляции, например такие как: управление выбором и последовательностью исполнения команд, возможность постановки команд в очередь, отмена команд и т.д.

Рассмотрим UML диаграмму для данного паттерна.



В данной диаграмме следующие участники:

Command

- Определяет интерфейс для исполнения операции.

ConcreteCommand

- Определяет связывание между объектом-получателем (Receiver) и действием.

- Реализует исполнение путем вызова соответствующих операций Receiver.

Client

- Создает объект **ConcreteCommand** и устанавливает его получателя.

Invoker

- Запрашивает команду выполнить некоторый запрос.

Receiver

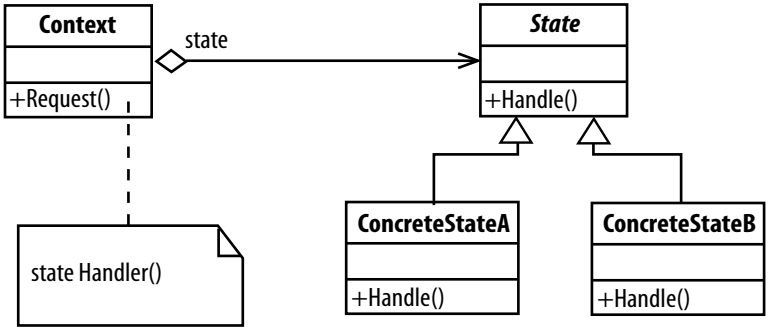
- Знает, как выполнить операции связанные с обработкой запроса.

Пример кода, который демонстрирует работу паттерна прикреплен к pdf-файлу данного урока.

4. Паттерн State

Паттерн State заключает состояния объекта в отдельные объекты, каждый из которых расширяет общий суперкласс.

Рассмотрим UML диаграмму для данного паттерна.



В данной диаграмме следующие участники:

Context

- Определяет интерфейс для клиентов.
- Поддерживает объект наследника ConcreteState, определяющего текущее состояние.

State

- Определяет интерфейс для инкапсуляции поведения, связанного с состоянием Context.

ConcreteState

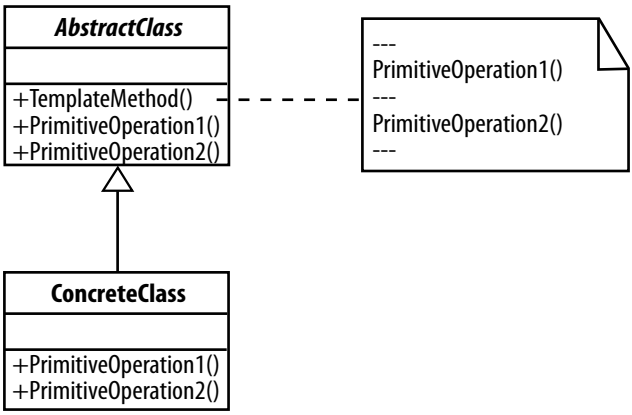
- Каждый наследник реализует поведение, связанное с состоянием Context.

Пример кода, который демонстрирует работу паттерна прикреплен к pdf-файлу данного урока.

5. Паттерн Template Method

Паттерн Template Method строится на абстрактном классе, содержащем часть логики, требуемой для исполнения задачи. Оставшаяся часть логики содержится в методах классов-потомков, которые создают свою реализацию абстрактных методов.

Рассмотрим UML диаграмму для данного паттерна:



В данной диаграмме следующие участники:

AbstractClass

- Определяет абстрактные примитивные операции, которые будут определены потомками для реализации шагов алгоритма.
- Реализует шаблонный метод, определяя скелет алгоритма. Шаблонный метод вызывает примитивные операции, определенные в AbstractClass или в других объектах.

ConcreteClass

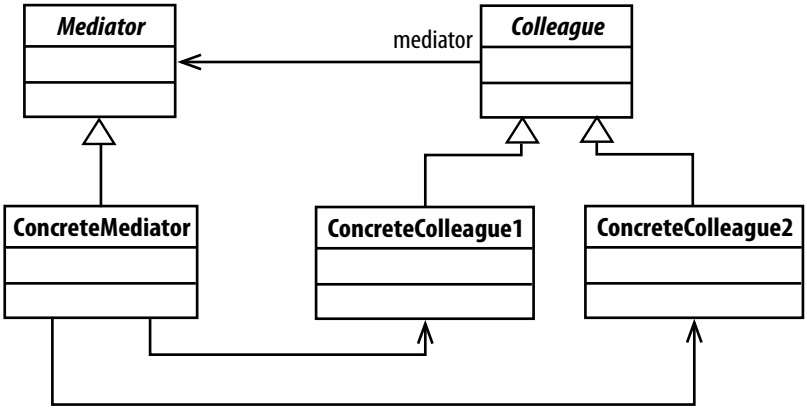
- Реализует примитивные операции, необходимые классам-потомкам для реализации алгоритмов.

Пример кода, который демонстрирует работу паттерна прикреплен к pdf-файлу данного урока.

6. Паттерн Mediator

Паттерн Mediator используется для согласования изменений состояний набора объектов с помощью одного объекта. То есть вместо раскидывания логики поведения по разным классам данный паттерн инкапсулирует логику управления изменением состояний в рамки одного класса.

Рассмотрим UML диаграмму для данного паттерна:



В данной диаграмме следующие участники:

Mediator

- Определяет интерфейс для общения с объектами Colleague.

ConcreteMediator

- Реализует совместное поведение путем координирования объектов Colleague.
- Знает и поддерживает своих Colleague.

Colleague классы

- Каждый Colleague класс знает своего Mediator.
- Каждый Colleague общается со своим медиатором.

Пример кода, который демонстрирует работу паттерна вы можете найти в папке с уроком.

В данном уроке мы привели часть паттернов поведения. Оставшиеся паттерны вам предназначены для самостоятельной проработки.

Код всех паттернов доступен для вас во вложении pdf-файла данного урока.

Экзаменационное задание

Реализуйте с использованием паттернов проектирования простейшую систему планирования задач. Должна быть возможность создания списка дел, установки приоритетов, установки дат выполнения, удаление и изменения дел. Каждому делу можно установить тег. Список дел можно загружать и сохранять в файл. Необходимо реализовать возможность поиска конкретного дела. Критерии поиска: по датам, по тегам, по приоритету и так далее.