Частное учреждение образования

«Колледж бизнеса и права»

УТВЕРЖДАЮ

Ведущий

методист колледжа

\_\_\_\_\_\_\_\_\_\_\_ Е.В. Паскал

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2021

|  |  |
| --- | --- |
| Специальность: 2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Технология разработки программного обеспечения» |

**Лабораторная работа № 23**

**Инструкционно-технологическая карта**

Тема: Создание программной системы, использующей поведенческие шаблоны

Цель: Научиться создавать программные системы, используя поведенческие шаблоны

Время выполнения: 2 часа

**Контрольныевопросы**

1. Назовите и опишите поведенческие шаблоны проектирования

**Теоретические сведения для выполнения работы**

Поведенческие шаблоны (англ. behavioral patterns) — шаблоны проектирования, определяющие алгоритмы и способы реализации взаимодействия различных объектов и классов.

Шаблоны поведения рассматривают вопросы о связях между объектами и распределением обязанностей между ними. Для этого могут использоваться механизмы, основанные как на наследовании, так и на композиции.

**Шаблон «Цепочка обязанностей»**

Шаблон «Цепочка обязанностей» (англ. Chain of Responsibility) позволяет избежать жесткой зависимости отправителя запроса от его получателя, при этом запрос может быть обработан несколькими объектами. Объекты-получатели связываются в цепочку. Запрос передается по этой цепочке, пока не будет обработан.

Используется в ситуациях, когда имеется поток запросов и переменное число "обработчиков" этих запросов. Необходимо эффективно обрабатывать запросы без жесткой привязки к их обработчикам, при этом запрос может быть обработан любым обработчиком.

Вводит конвейерную обработку для запроса с множеством возможных обработчиков.

Обычно, для реализации шаблона используется объектно-ориентированный связанный список с рекурсивным обходом. При большом количестве возможных обработчиков запроса рекурсивный обход не рекомендуется, так как может привести к переполнению стека вызовов.

Инкапсулирует элементы по обработке запросов внутри абстрактного "конвейера". Клиенты передают свои запросы на вход этого конвейера.

Шаблон связывает в цепочку объекты-получатели, а затем передает запрос-сообщение от одного объекта к другому до тех пор, пока не достигнет объекта, способного его обработать. Число и типы объектов-обработчиков заранее неизвестны, они могут настраиваться динамически. Механизм связывания в цепочку использует рекурсивную композицию, что позволяет использовать неограниченное число обработчиков. За счёт этого упрощаются взаимосвязи между объектами. Вместо хранения ссылок на всех кандидатов-получателей запроса, каждый отправитель хранит единственную ссылку на начало цепочки, а каждый получатель имеет единственную ссылку на своего преемника - последующий элемент в цепочке.

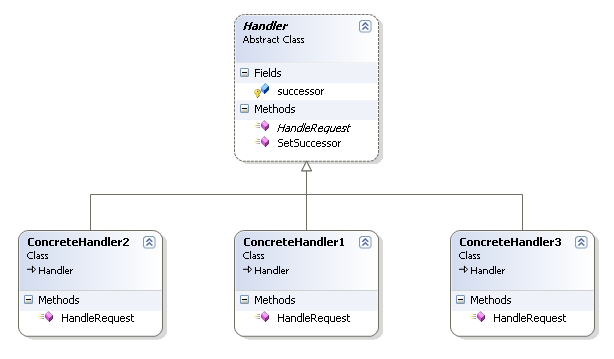


Рисунок 1 Диаграмма классов

**Шаблон «Команда»**

Шаблон «Команда» (англ. Command) —используется в случаях, когда необходима реализация какого-либо действия. Объект команды заключает в себе само действие и его параметры.

Используется в следующих случаях:

Система управляется событиями. При появлении такого события (запроса) необходимо выполнить определенную последовательность действий;

Необходимо параметризировать объекты выполняемым действием, ставить запросы в очередь или поддерживать операции отмены (undo) и повтора (redo) действий;

Нужен объектно-ориентированный аналог функции обратного вызова в процедурном программировании.

Шаблон преобразовывает запрос на выполнение действия в отдельный объект-команду. Такая инкапсуляция позволяет передавать эти действия другим функциям и объектам в качестве параметра, приказывая им выполнить запрошенную операцию. Команда – это объект, поэтому над ней допустимы любые операции, что и над объектом.

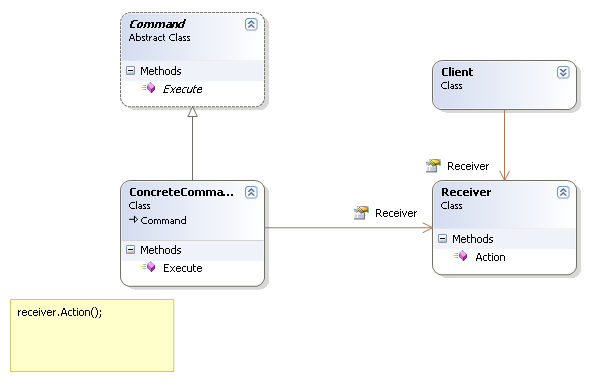


Рисунок 2 Реализация шаблона «Команда»

Клиент создаёт команду и передаёт ей экземпляр класса получателя команды. Затем выполняет метод команды «Execute», который выполняет конкретное действие получателя команды.

**Шаблон «Интерпретатор»**

Интерпретатор (англ. Interpreter) решает часто встречающуюся, но подверженную изменениям, задачу.

Если в некоторой, хорошо определенной области периодически случается некоторая проблема и эта область может быть описана некоторым «языком», то проблема может быть легко решена с помощью «интерпретирующей машины».

Шаблон определяет грамматику простого языка для проблемной области, представляет грамматические правила в виде языковых предложений и интерпретирует их для решения задачи. Для представления каждого грамматического правила шаблон «Интерпретатор» использует отдельный класс. А так как грамматика, как правило, имеет иерархическую структуру, то иерархия наследования классов хорошо подходит для ее описания.

Если грамматика очень сложная, должны использоваться другие методики.

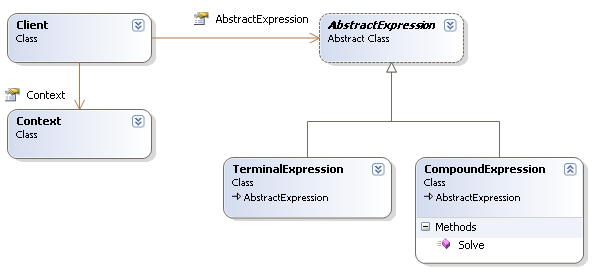


Рисунок 3 Диаграмма классов

Шаблон «Интерпретатор» моделирует проблемную область с помощью рекурсивной грамматики. Каждое грамматическое правило может быть либо составным (CompoundExpression - правило ссылается на другие правила) либо терминальным (TerminalExpression - листовой узел в структуре ”дерево”).

**Шаблон «Итератор»**

Шаблон «Итератор» (англ. Iterator, Cursor) — представляет собой объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из объектов, входящий в состав агрегации.

Составной объект, такой как список, должен предоставлять способ доступа к его элементам без раскрытия своей внутренней структуры. Более того, иногда нужно перебирать элементы списка различными способами, в зависимости от конкретной задачи.

Перебор элементов выполняется объектом итератора, а не самой коллекцией. Это упрощает интерфейс и реализацию коллекции, а также способствует более логичному распределению обязанностей.

Особенностью полноценно реализованного итератора является то, что код, использующий итератор, может ничего не знать о типе итерируемого агрегата.

Абстракция шаблона имеет основополагающее значение для технологии, называемой «обобщенное программирование». Эта технология четко разделяет такие понятия как «алгоритм» и «структура данных». Мотивирующие факторы: способствование компонентной разработке, повышение производительности и снижение расходов на управление.

Например, если необходимо одновременно поддерживать четыре вида структур данных (массив, бинарное дерево, связанный список и хэш-таблица) и три алгоритма (сортировка, поиск и слияние), то традиционный подход потребует 12 вариантов конфигураций (четыре раза по три), в то время как обобщенное программирование требует лишь 7 (четыре плюс три).

В общем виде, структура шаблона на диаграмме выглядит следующим образом:

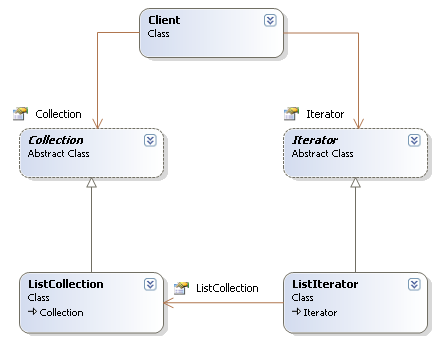


Рисунок 4 Диаграмма классов шаблона

**Шаблон «Посредник»**

Шаблон «Посредник» (англ. Mediator) —обеспечивает взаимодействие множества объектов, формируя при этом слабую связанность и избавляя объекты от необходимости явно ссылаться друг на друга.

Разбиение системы на множество объектов повышает степень повторного использования, однако, множество взаимосвязей между этими объектами приводит к обратному эффекту. Чтобы этого не допустить, необходимо инкапсулировать взаимодействия между объектами в объект-посредник. Действуя как центр связи, этот объект-посредник контролирует и координирует взаимодействие группы объектов. При этом объект-посредник делает взаимодействующие объекты слабо связанными, так как им больше не нужно хранить ссылки друг на друга – все взаимодействие идет через этого посредника. Расширить или изменить это взаимодействие можно через его подклассы.

Шаблон «Посредник» заменяет взаимодействие "все со всеми" взаимодействием "один со всеми".

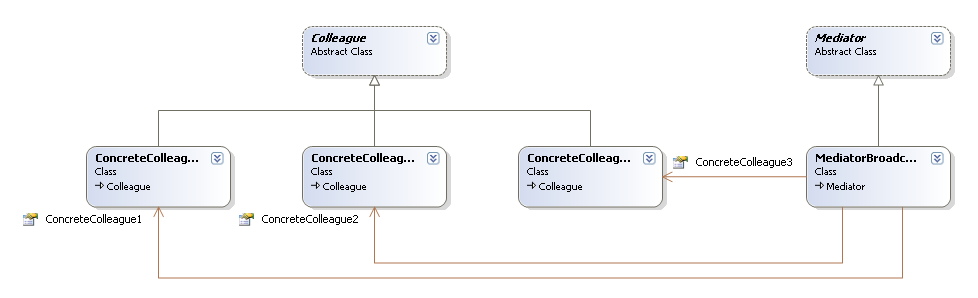


Рисунок 5 Диаграмма классов для примера использования шаблона

На данной диаграмме медиатор осуществляет взаимодействие, в виде рассылки сообщений, между сотрудниками компании.

Для того, чтобы показать удобство использования шаблона, у каждого класса, который реализует конкретного сотрудника сделаны собственные методы для отправки и получения сообщения, которых нет в родительском классе.

Конкретный класс «MediatorBroadcaster» хранит экземпляры классов конкретных сотрудников и управляет механизмом пересылки сообщений между ними. Таким образом, классам нет необходимости хранить информацию друг о друге, это делается в классе – посреднике.

**Порядок выполнения работы**

1. Изучите теоретические сведения.

2. Реализовать программу использующую один из поведенческих шаблонов по теме индивидуального задания

Преподаватель \_\_\_\_\_\_\_\_ А.С. Комаровский

Рассмотрено на заседании цикловой

комиссии ПОИТ №10

Протокол № от «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021

Председатель ЦК \_\_\_\_\_\_\_\_ В.Ю. Михалевич