МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Объектно-ориентированное программирование»

Тема: Паттерны

Студент гр. 7304 _____ Сергеев И.Д.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2019

Цель работы.

Исследование паттернов проектирования и их реализация на языке программирования C ++.

Задание

Вариант 2. Обработчик последовательности.

Реализовать обработчик последовательности объектов стандартного (int, string, double, char) типа. Обработчик представляет собой ориентированный параллельно-последовательный граф с одним входом и выходом. Каждая вершина графа является функциональным элементом, выполняющим какое-то действие над объектом.

Функциональные элементы могут быть различаться по количеству входов и выходов: 1 вход – 1 выход; 1 вход – 2 выхода; 2 входа – 1 выход: 2 входа – выхода. Вход обработчика может быть элементом только с 1 входов, а выход обработчика может быть элементом только с 1 выходом. Причем, функциональные элементы с 2 входами выполняются только когда объекты поступили на оба входа. Например, могут быть следующие функциональные элементы: умножение числа на другое число n; определение более длинной строки; расчет целой части и остатка от деления числа на число n.

Такт – одна итерация работы всего обработчика. За один такт выполняется каждый функциональный элемент, передает объект дальше и принимает следующий. Таким образом, работа обработчика происходит по принципе конвейера.

При запуске программы, набор обработчиков заранее известен, но должна быть возможность построения обработчика. Также должна быть возможность замены одного функционального элемента на другой в ходе работы обработчики, причем состояние должно сохраняться.

Также необходимо предусмотреть возможность отката на один такт назад.

Экспериментальные результаты.

1. Использование как минимум один паттерна каждого типа для решения задачи (порождающие, структурные, поведенческие). (подробности ниже)
2. Сохранение логов хода работы программы в файл. Реализовал 3 вида сохранения

- Без сохранения логов
- Моментальное сохранение (информацию о событии сразу записывается в файл)
- Сохранение с кэшированием (информация сохраняется в кэш, при заполнении кэша происходит запись в файл и очистка кэша)

3. UML-диаграммы классов.
4. Реализация своего класса исключений их обработка.

Опциональные пункты:

- Сохранение и загрузка состояния программы в файл. Реализовал для 2 расширений файлов (txt, бинарный файл и т.д.).
- Реализация GUI. При создании GUI необходимо разделять логику интерфейса и бизнес-логику.

Минимальные требования к заданию:

- По 2 типа обработчика для каждой комбинации входа/выхода
- Обработчик должен работать с двумя стандартными типами (int и double)
- Обработчик должен работать с одним пользовательским типом (Coordinate)

Дополнительное задание:

- Замена одного функционального элемента набором функциональных элементов

- Реализация обработчика с несколькими входами и выходами. Добавлялся возможность обработки разных последовательностей одного типа
- Функциональный элементы могут преобразовывать тип объектов.

Реализованные кнопки

- next – переключение на следующую в графе вершину.
- Prev – переключение на предыдущую в графе вершину
- Init – инициализация графа из файла, который выбирает пользователь
- newCurr – создание новой операции на текущей вершине.
- Change – изменение операции текущей вершины.
- newStart – создание стартовой вершины, без файла.
- NewNext – создание новой следующей вершины.
- Swap – переключение между операциями н текущей вершине.
- Openlog – открытие лог-фала.
- Save – обычное сохранение в лог-файл.
- CacheSave – сохранение в буфер данных.
- openGraph – открыть текущее отображение вершин графа.

Паттерны проектирования, используемые в лабораторной работе

Порождающие

Одиночка (Singleton)

Одиночка — это порождающий паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.

В этом проекте он используется в классе myInterface. Это нужно, чтобы в программе был гарантированно 1 экземпляр обработчика. Также это необходимо, чтобы люди, которые захотят расширить код программы не использовали 2 класс интерфейса и обработчика.

Структурные

Заместитель (Proxy)

Заместитель — это структурный паттерн проектирования, который позволяет подставлять вместо реальных объектов специальные объекты-заменители. Эти объекты перехватывают вызовы к оригинальному объекту, позволяя сделать что-то до или после передачи вызова оригиналу.

В этом проекте он используется для обеспечения возможности кэширования для сохранения в файл информации. CacheObj является заменой Simple obj, он содержит в себе буфер, который заполняется по мере нажатия кнопки cachesave, и вызывает функцию записи до определенного значения счетчика нажатий. Это помогает уменьшить количество вызовов функции записи, что занимает много времени. Если пользователь хочет сохранить все обычно, то ему следует нажать кнопку simplesave, которая сохранит текущую информацию в файл.

Декоратор (Decorator)

Декоратор – структурный паттерн проектирования, который позволяет динамически добавлять объектам новую функциональность, оборачивая их в полезные “обертки”.

В данном случае он используется в создании узла графа в графическом интерфейсе. Паттерн добавляет к уже имеющимся операции и вводу ID каждого узла. Это позволяет не задумываться о дописывании ID узла каждый раз, когда мы добавляем новую вершину.

Поведенческие

Снимок (Memento)

Снимок — это поведенческий паттерн проектирования, который позволяет сохранять и восстанавливать прошлые состояния объектов, не раскрывая подробностей их реализации.

Класс Snapshot хранит в себе информацию о текущей вершине, которая при надобности может заменить текущую. Используя этот шаблон, Модель может сохранять состояния обработчиков, выполнять такт, резервные копии после этого могут быть использованы для возврата такта в любое время. Объект класса создается при инициализации последовательности. Снимок делается при нажатии кнопки next и используется пользователем при нажатии prev. Это гарантирует откат операции на 1 такт.

Выводы

В результате этой лабораторной работы я изучил шаблоны проектирования и использовал их для реализации задачи построения последовательности функциональных элементов. Были реализованы 3 паттерна на языке c++ такие как: Singleton, Decorator, Memento и Proxy.