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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Объектно-ориентированное программирование»

ТЕМА: ОБРАБОТЧИК ПОСЛЕДОВАТЕЛЬНОСТИ.

Студент гр. 7304 Моторин Е.В.

Преподаватель Размочаева Н.В

Санкт-Петербург

2019

Цель работы:

Реализовать программу обработчика последовательности, которые представляет из себя параллельно последовательный граф с одним входом и одним выходом.

1. Задача:

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.

Минимальные требования к заданию:

1. По 2 типа обработчика для каждой комбинации входа/выхода
2. Обработчик должен работать с двумя стандартными типами
3. Обработчик должен работать с одним пользовательским типом

Дополнительное задание (выбрать минимум одно):

1. Замена одного функционального элемента набором функциональных элементов
2. Реализация обработчика с несколькими входами и выходами. Добавляется возможность обработки разных последовательностей одного типа
3. Функциональный элементы могут преобразовывать тип объектов.

2. Ход работы:

2.1. В качестве дополнительного задания был реализован GUI с использованием средств QT.

2.2. Был реализован паттерн адаптер для конвертации формата JSON в граф.

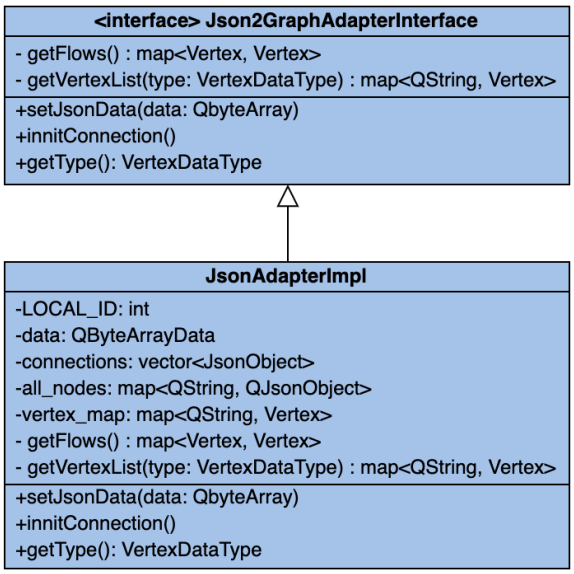

Рис. 1 (UML-диаграмма JSON адаптер)

2.3. Реализован паттерн снимок для хранения состояний графа. Снимок позволяет запоминать каждое из состояний графа и возвращаться к ним.

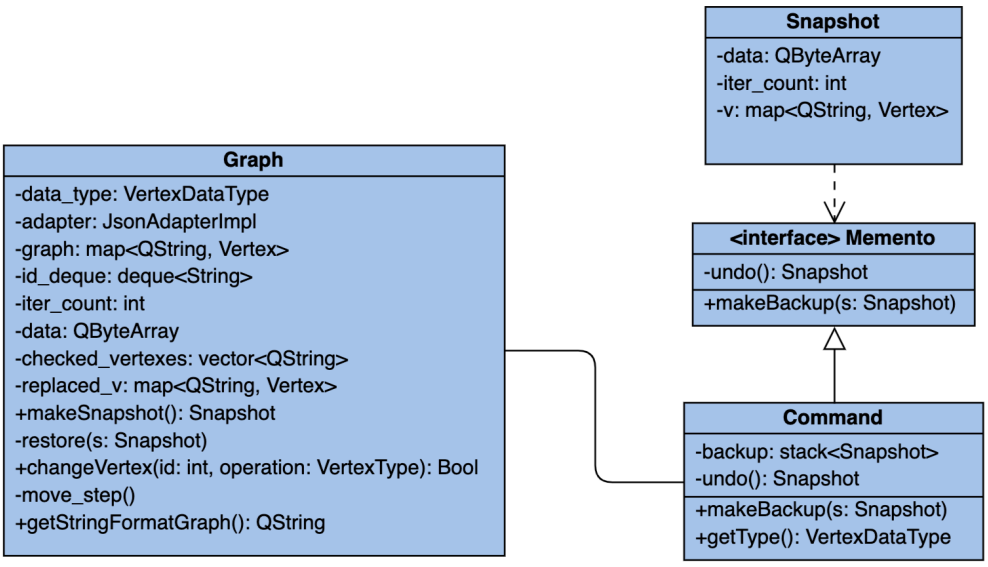

Рис. 2 (UML-диаграмма паттерна Снимок)

2.4. Реализован паттерн одиночка для доступа к логгеру, который кэширует и записывает логи в файл, также логгер реализует паттерн “Proxy”.

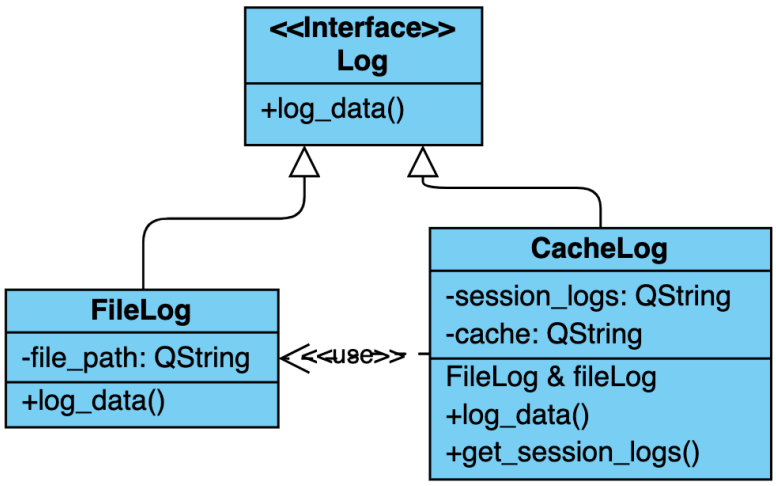

Рис. 3(UML-диаграмма Логгер)

2.5. Реализован класс обработки ошибок CustomException, он позволяет передавать ошибку и данные при которых оно возникла.

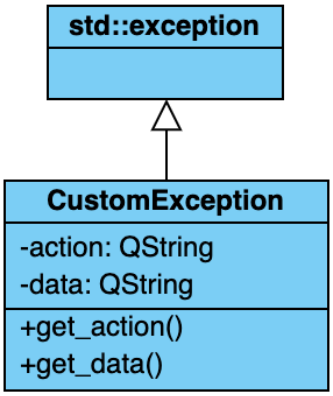

Рис. 4(UML-диаграмма CustomException)

3. Описание интерфейса:

3.1. Реализован графический интерфейс для создания графа. Данный интерфейс позволяет добавлять новые вершины графа и устанавливать связи между ними, есть возможность сохранить состояние графа в формате JSON и восстановить граф из файла, также можно возвращаться пошагово назад, просматривать логи и запустить новое окно с поэтапным отображением работы программы.

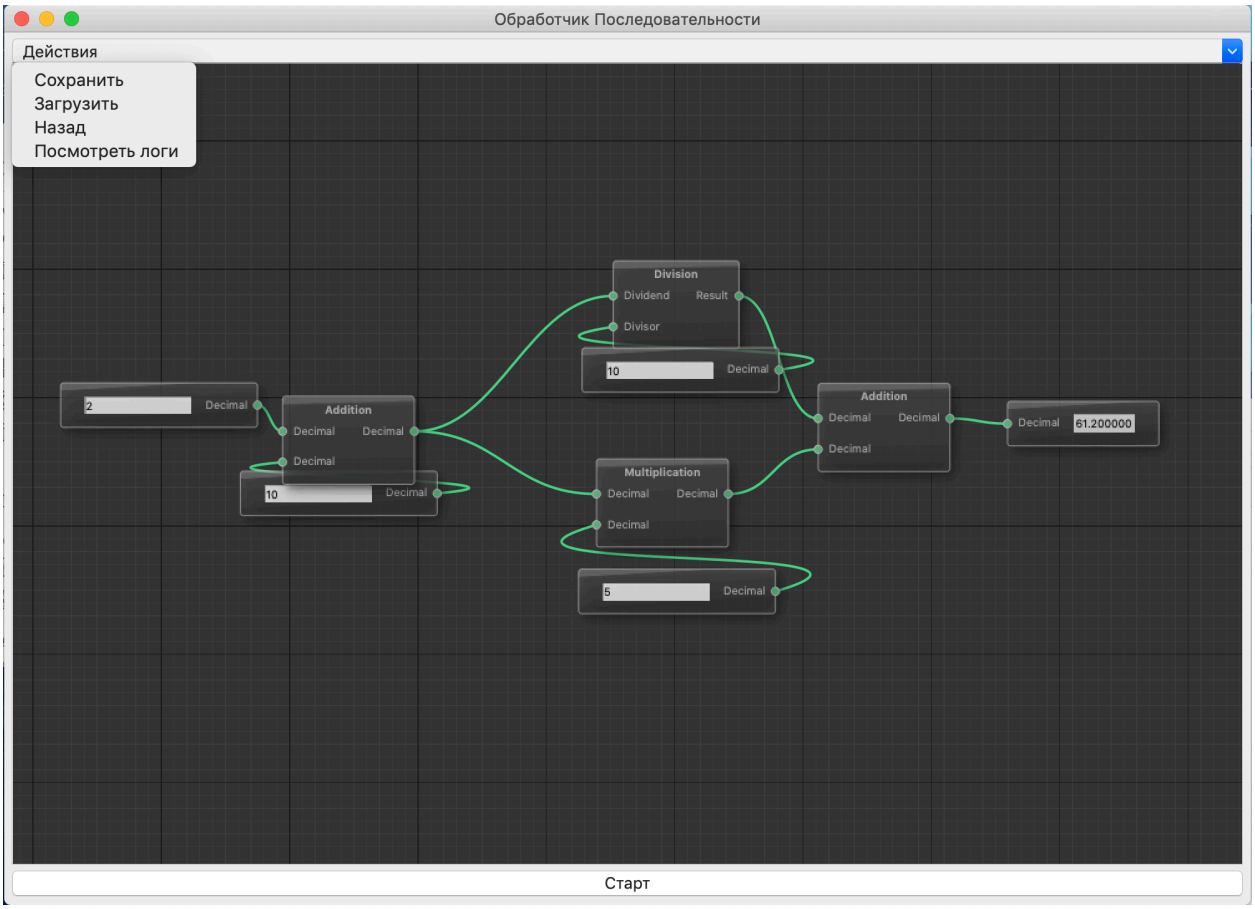

Рис. 5 (Окно составления графа)

3.2. Новые вершины добавляются с помощью всплывающего меню, которое вызывается нажатием правой кнопки мыши в поле окна.

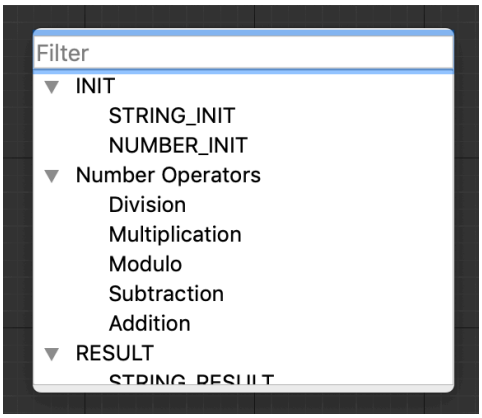

Рис. 6 (Меню добавления вершины)

3.3. В качестве дополнительного задания была реализована возможность просмотреть логи работы программы, открыть окно с логами можно через меню окна составления графа (см. рисунок 5). Также можно просматривать логи в окне поэтапного отображения работы программы (см. рисунок 8).

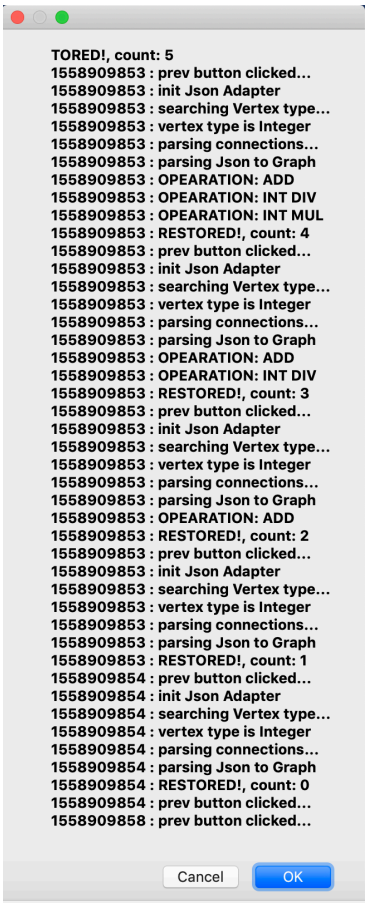

Рис. 7 (Окно с логами работы программы)

3.4. Реализованное окно с пошаговым отображением работы программы, оно включает в себя кнопки “вперед” и “назад” для прохода по графу вперед и возврату на 1 шаг назад, кнопку “Поменять вершину” для замены операции в вершине, две QScrollArea, которые отображают состояние графа на данном шаге и логи.

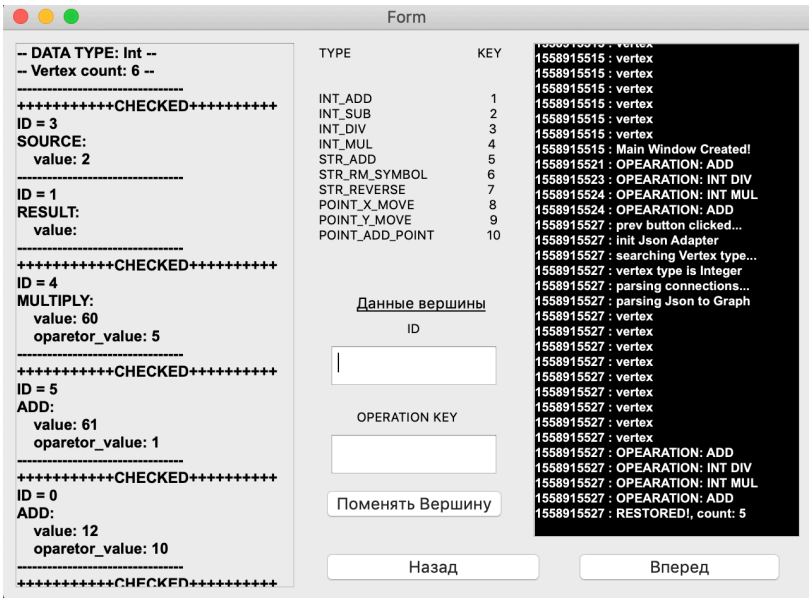

Рис. 8 (Окно пошаговой работы программы)

Вывод:

В ходе выполнения лабораторной работы были реализованы иерархии классов для графа и его компонентов, а также был применен структурный паттерн “Адаптер”, который преобразует входные данные в формате json в граф. Реализован механизм логирования с использованием структурного паттерна “Заместитель”, логи записываются в файл при переполнении буфера, логи текущей сессии дублируеются в отдельную переменную для удобного вывода в окне пошаговой статистики, также механизм логирования реализует порождающий паттерн “Одиночка” для доступа к нему из разных частей программы. Был реализован поведенческий паттерн “Снимок” для записи состояний графа и удобства возврата к ним.