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

ОТЧЕТ

по учебной практике

Тема: Кратчайшие пути в графе. Алгоритм Дейкстры.

Студент гр. 7304 _____ Абдульманов Э.М

Студентка гр. 7304 _____ Каляева А.В

Студент гр. 7304 _____ Комаров А.О

Руководитель _____ Жангиров Т.Р.

Санкт-Петербург

2019

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Абдульманов Э.М. группы 7304
Студентка Каляева А.В. группы 7304
Студент Комаров А.О группы 7304

Тема практики: Кратчайшие пути в графе. Алгоритм Дейкстры.

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.

Алгоритм: Дейкстры.

Сроки прохождения практики: 01.07.2019 – 14.07.2019

Дата сдачи отчета: 00.07.2019

Дата защиты отчета: 00.07.2019

Студент	_____	Абдульманов Э.М
Студентка	_____	Каляева А.В
Студент	_____	Комаров А.О
Руководитель	_____	Жангиров Т.Р.

АННОТАЦИЯ

В ходе выполнения задания учебной практики была реализована программа, предназначенная для нахождения кратчайшего пути в графе. Поиск кратчайшего пути осуществляется с использованием алгоритма Дейкстры. Разработанная программа детально показывает этапы работы алгоритма при построении кратчайшего пути. Программа была написана на языке программирования Java, в среде разработки IntelliJ Idea.

SUMMARY

During the execution of educational practice was written a program to find the shortest path in the graph. Shortest path search is written using Dijkstra`s algorithm. The developed program demonstrates the steps of the algorithm. The program was written in Java programming language. The program was written using the IntelliJ Idea development environment.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В ходе выполнения задания учебной практики требуется разработать программу для поиска кратчайшего пути в графе с помощью алгоритма Дейкстры.

Алгоритм Дейкстры - алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Данный алгоритм находит кратчайшие пути от одной из вершин графа до всех остальных вершин. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании и технологиях, например, его используют протоколы маршрутизации OSPF и IS-IS.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. Требования к вводу исходных данных

На вход алгоритму подается взвешенный ориентированный граф. Именем вершины могут быть буквы (как строчные, так и прописные), а так же строки и цифры.

1.1.2. Требования к визуализации

Пользовательский интерфейс представляет собой диалоговое окно, содержащее набор кнопок, предназначенных для управления состоянием программы.

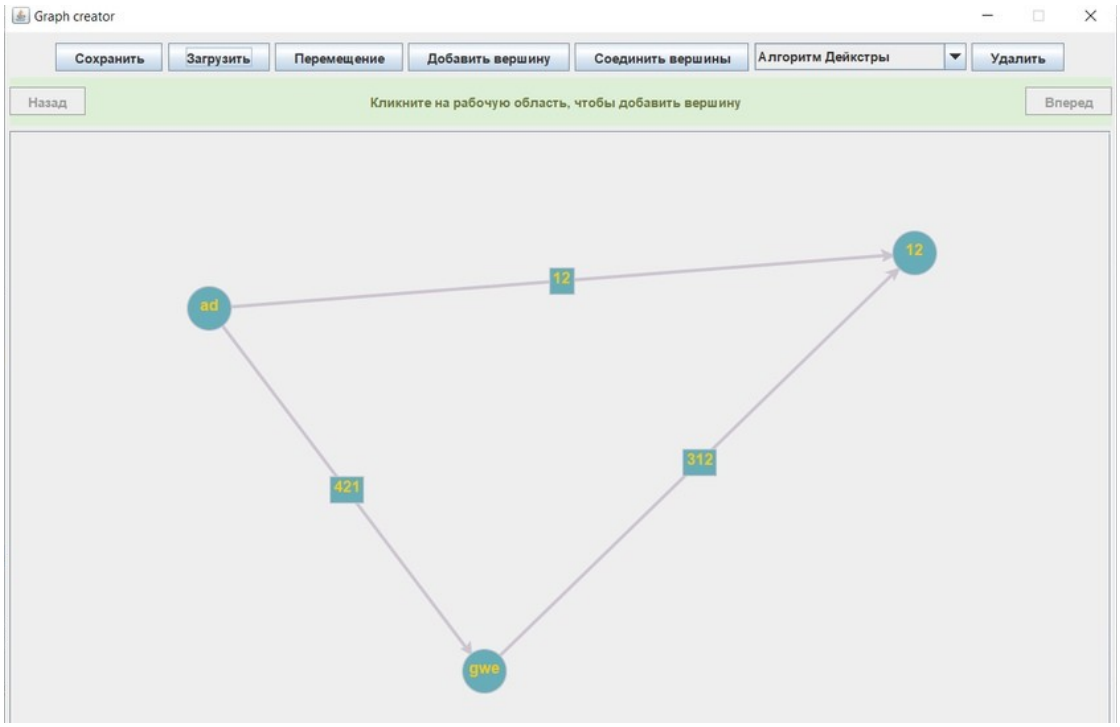

Рис. 1 Пример пользовательского интерфейса

Диалоговое окно состоит из:

- Рабочей области для построения графа.
- Кнопки «Сохранить», которая позволяет пользователю сохранить созданный в рабочей области граф в виде .txt файла.

- Кнопки «Загрузить», которая позволяет пользователю загрузить граф, для которого необходимо применить алгоритм Дейкстры, в виде .txt файла.
- Кнопки «Перемещение», которая позволяет перемещать вершины созданного графа внутри рабочей области.
- Кнопки «Добавить вершину», которая создает вершину графа в рабочей области после клика мышью. Вновь созданной вершине пользователь должен дать имя, допустимое для данной программы.
- Кнопки «Соединить вершины», которая позволяет создать направленное ребро между двумя вершинами и задать его вес.
- Кнопки «Удалить», которая удаляет выбранный пользователем элемент графа.
- Кнопки «Вперед», которая отображает следующую итерацию алгоритма.
- Кнопки «Назад», которая отображает предыдущую итерацию алгоритма.

1.2. Уточнение требований после сдачи прототипа

- Необходимо добавить кнопку, отвечающую за старт работы алгоритма, на построенном графе.
- Необходимо добавить кнопку, отвечающую за отображение результата работы алгоритма Дейкстры.
- Необходимо добавить окно логирования.

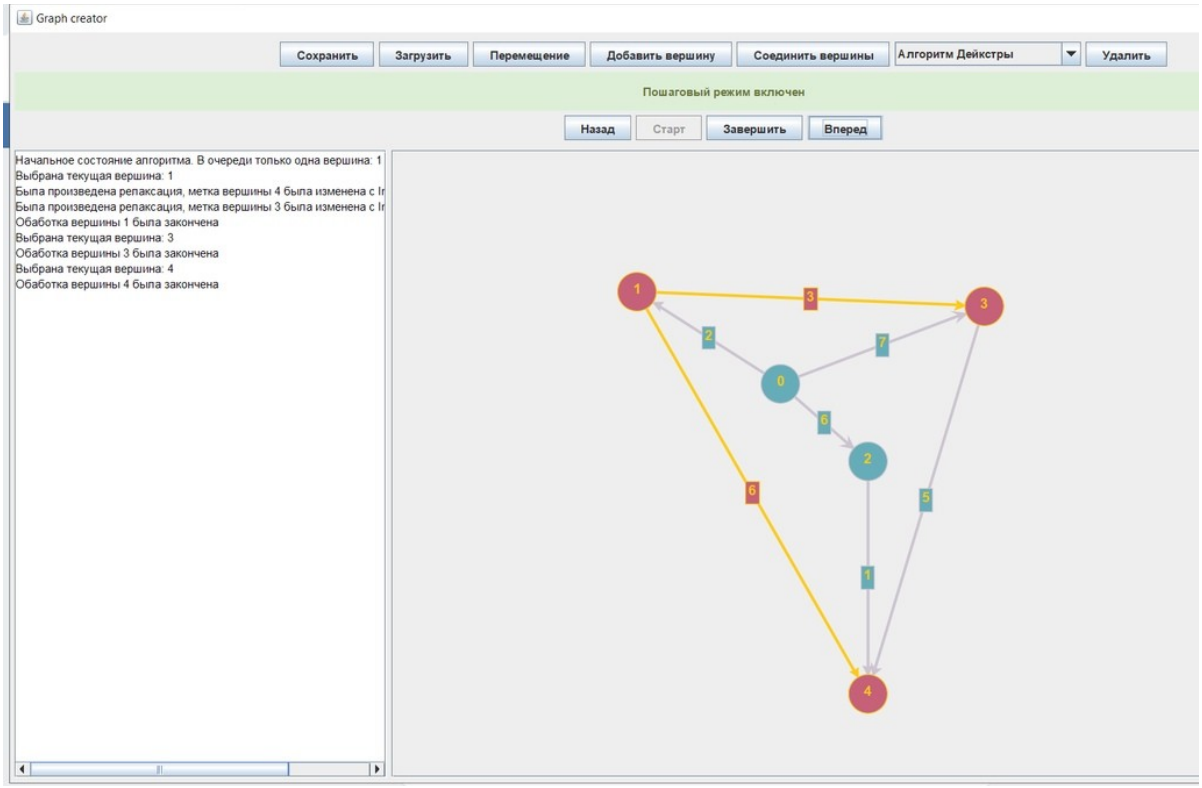

Рис. 2 Пример пользовательского интерфейса после уточнения требований

1.3. Уточнение требований после сдачи 1-ой версии программы.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

- Прототип: демонстрация окна пользовательского интерфейса программы без функционала. Создание UML диаграмм. Срок выполнения: к 4 июля.
- Итерация 1: написан код для работы алгоритма. Реализованы кнопки, отвечающие за отрисовку графа. Срок выполнения: к 6 июля.
- Итерация 2: исправлены возможные недочеты в коде алгоритма. Реализованы кнопки пользовательского интерфейса, отвечающие за модификацию текущего состояние графа. Написан код, для демонстрации пошаговой работы алгоритма. Срок выполнения: к 8 июля.
- Финальная версия программы: исправлены возможные недочеты проекта, закончено оформление отчета. Срок выполнения: к 11 июля.

2.2. Распределение ролей в бригаде

Абдульманов Э. - архитектура программы.

Каляева А. – реализация алгоритма.

Комаров А. – визуализация работы алгоритма.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Архитектура программы

- Архитектура данного приложения построена на основе шаблона проектирования MVC (Model View Controller). Суть данного шаблона заключается в том, что любое действие пользователя обрабатывается в контроллере, который имеет доступ к модели. Контроллер, в свою очередь, изменяет состояние модели. View отображает текущее состояние модели. Таким образом, выделив 3 интерфейса (GraphCreatorController, GraphCreator View, GraphCreatorModel) разрабатывать данное приложение можно независимо в трех «плоскостях».

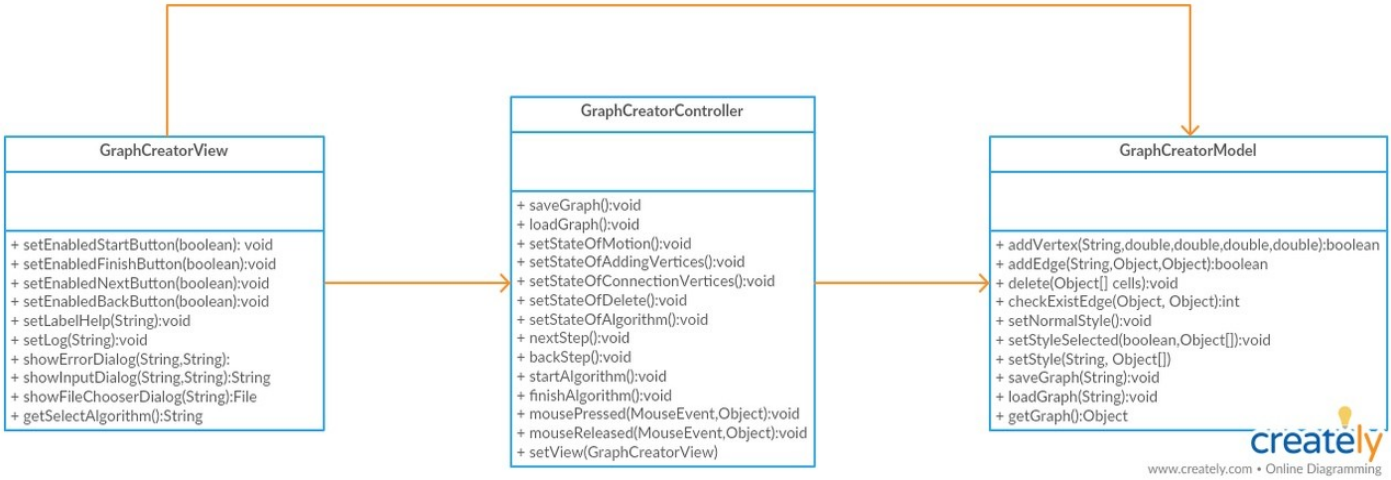

Рис.3 UML диаграмма

- Для того, чтобы пользователь мог добавлять, перемещать, соединять вершины, а так же удалять элементы графа был применен шаблон проектирования State(Состояние). Пользователь имеет 2 рычага управления – нажать и отпустить кнопку мыши. В разные моменты времени эти два рычага выполняют разные действия: добавить вершину, удалить элемент графа и так далее. Чтобы не писать большие условные конструкции был применен шаблон проектирования State. Были выделены следующие состояния:

DeleteState, MoveState, AddVertexState, ConnectionVertexState, AlgorithmShortestWayState. Контекстом для данных состояний стал контроллер, который имеет поле currentState и обрабатывает действия кнопки мыши. Смена состояний происходить путем обработки нажатий на соответствующие кнопки.

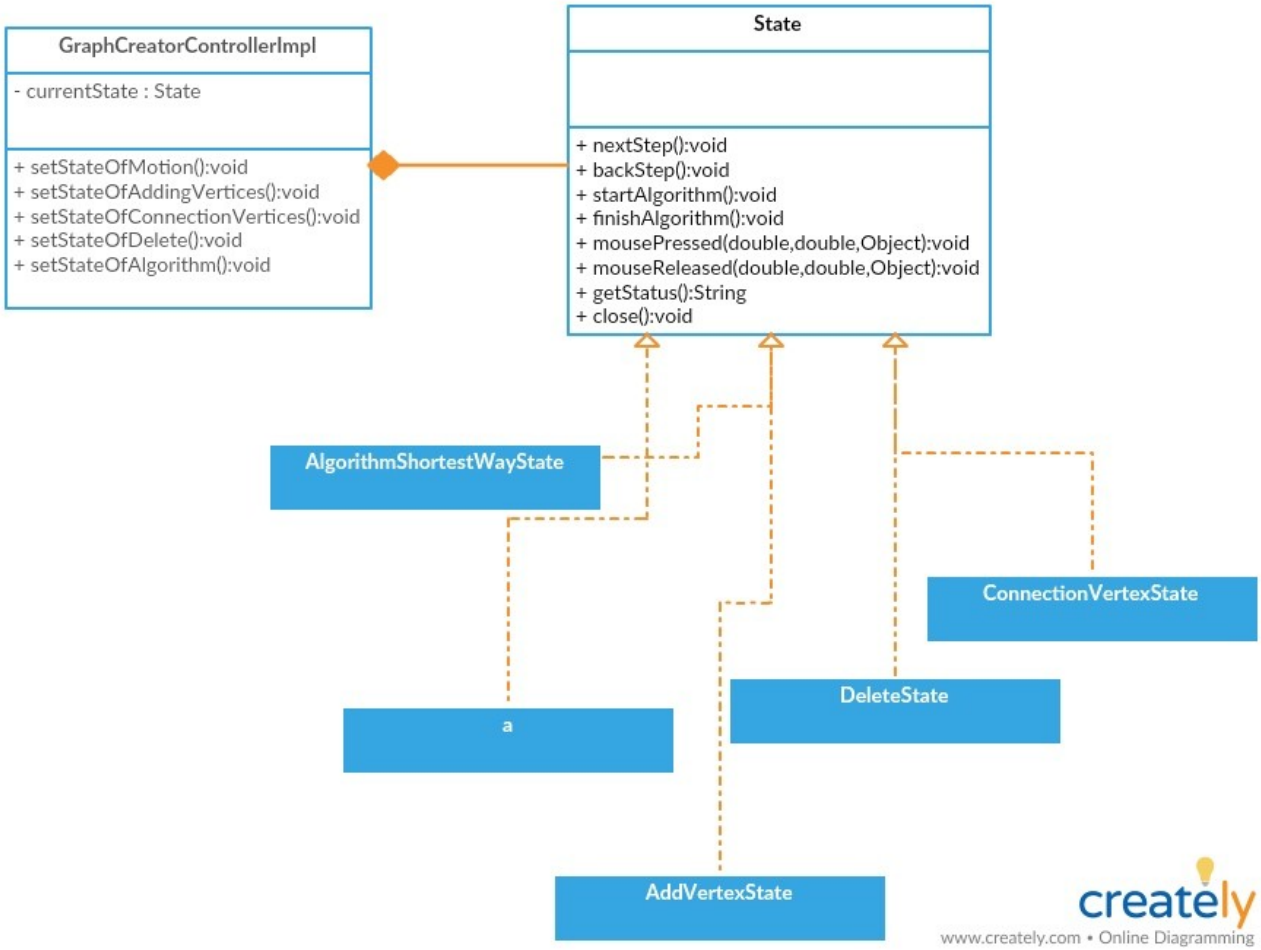

Рис. 4 UML диаграмма

- Следующая задача заключалась в том, чтобы разработать алгоритм таким образом, чтобы он не зависел от его отображения. Первая идея была в том, чтобы вынести реализацию алгоритма в другой модуль и написать его так, будто выполняется консольное приложение. Вторая идея была в том, чтобы применить шаблон проектирование Adapter. Прежде чем запустить алгоритм, данные

преобразовываются в необходимый формат, алгоритм выполняется, а затем данные обратно преобразовываются для отображения.

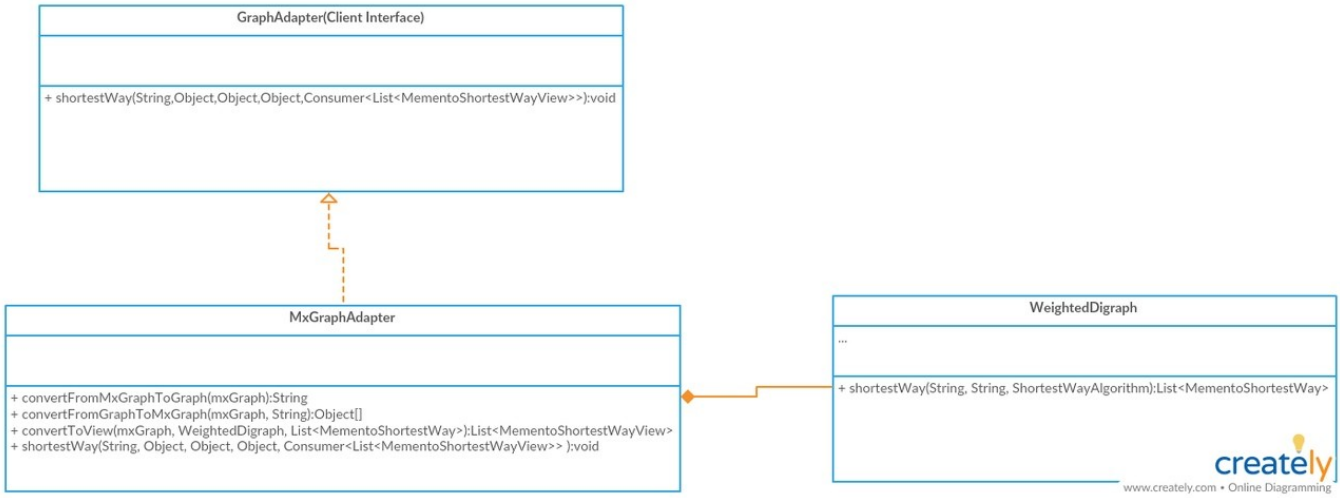

Рис.5 UML диаграмма

- Следующая проблема заключалась в реализации пошаговой визуализации работы алгоритма. Решить эту задачу помогла идея шаблона проектирования Снимок. Был создан класс MementoShortestWay, который хранит в себе текущую вершину, обработанные вершины, вершины в очереди, текущие пути до вершин. В определенные моменты алгоритма в контейнер закидываются снимки текущего состояния алгоритма. Затем, когда необходимо сделать шаг вперед просто отображается i+1 снимок.
- Так же был продуман и реализован пользовательский интерфейс.

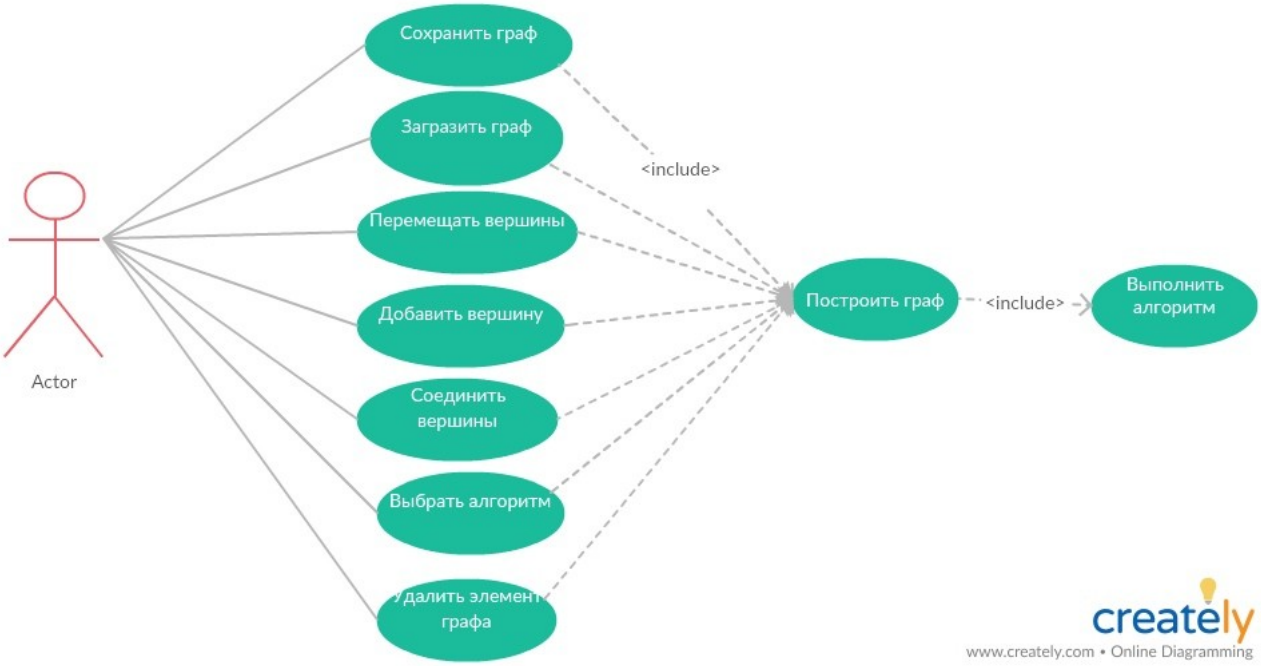

Рис.6 Use case диаграмма

3.2. Второй подраздел третьего раздела