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

ОТЧЕТ

по учебной практике

Тема: Топологическая сортировка

Студент гр. 9381 Прашутинский К.И.

Студент гр. 9381 Николаев А.А.

Руководитель Фирсов М.А.

Санкт-Петербург

2021

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Прашутинский К.И. группы 9381

Студент Николаев А.А. группы 9381

Тема практики: Топологическая сортировка.

Задание на практику:

Командная итеративная разработка визуализатора алгоритма(ов) на Java с графическим интерфейсом.

Алгоритм: Топологическая сортировка.

Сроки прохождения практики: 1.07.2021 – 12.07.2021

Дата сдачи отчета: 12.07.2021

Дата защиты отчета: ??.07.2021

Студент		Прашутинский К.И.
Студент		Николаев А.А.
Руководитель		Иванов И.И.

АННОТАЦИЯ

Целью текущей учебной практики является разработка GUI приложения для топологической сортировки для заданного графа.

Программа разрабатывается на языке Java командой из двух человек, каждый из которых имеет определенную специализацию. Сдача и показ проекта на определенном этапе выполнения осуществляется согласно плану разработки.

SUMMARY

The goal of the current training practice is to develop a GUI application for topological sorting for a given graph.

The program is developed in Java by a team of two people, each of whom has a specific specialization. The delivery and display of the project at a certain stage of implementation is carried out according to the development plan.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью текущей учебной практики является разработка GUI приложения для топологической сортировки для заданного графа.

Программа предоставляет пользователю структурированный интерфейс для создания и изменения графа. Во время визуализации работы алгоритма Топологической сортировки пользователь может либо сразу вывести результат, либо пошагово идти по алгоритму. Каждый шаг алгоритма имеет графическое отображение.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

Программа представляет собой визуализацию алгоритма Топологической сортировки для ориентированного графа без циклов.

1.1.1. Требования к вводу исходных данных

Программа предоставляет пользователю следующий интерфейс для создания графа:

- Чтения графа из файла
- Изменения текущего графа

1.1.2. Требования к визуализации

Графический интерфейс предоставляет пользователю следующие методы для изменения графа:

- Добавить вершину
- Удалить вершину
- Переместить вершину
- Удалить вершину

Пользователь имеет возможность применить алгоритм Топологической сортировки для построенного графа, если он удовлетворяет условию отсутствия циклов. Алгоритм Топологической сортировки нахождения для заданного графа имеет два способа реализации:

- Вывод результата алгоритма
- Пошаговая работа алгоритма

Если выбран вариант пошаговой визуализации алгоритма, пользователю предоставляется возможность самому управлять последовательность выполнения алгоритма: он может либо «откатиться» к предыдущей итерации, либо пройти к следующей.

Таким образом, программа предлагает следующий интерфейс для прохода по алгоритму:

- Вперед (не доступен на последнем шаге)
- Назад (не доступен на первом шаге)

1.1.3. Внешний вид приложения

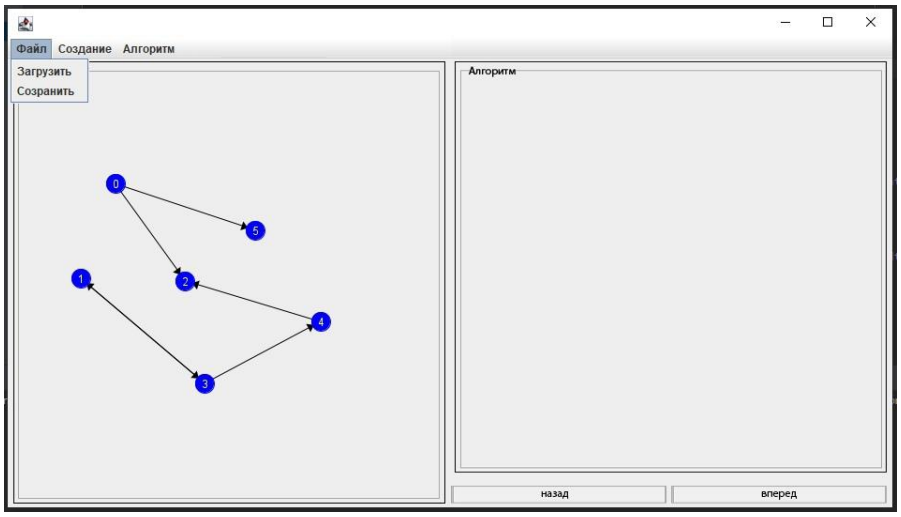

Рис. 1 – Внешний вид приложения (1)

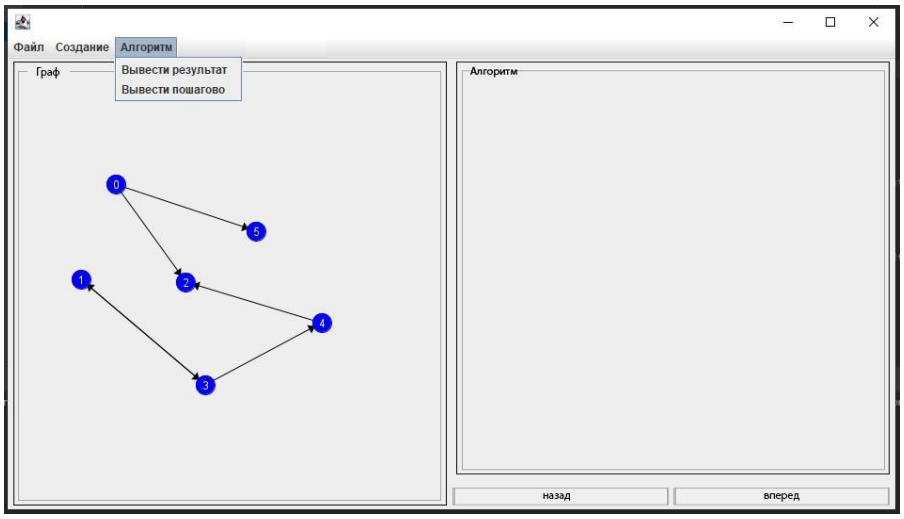

Рис. 2 – Внешний вид приложения (2)

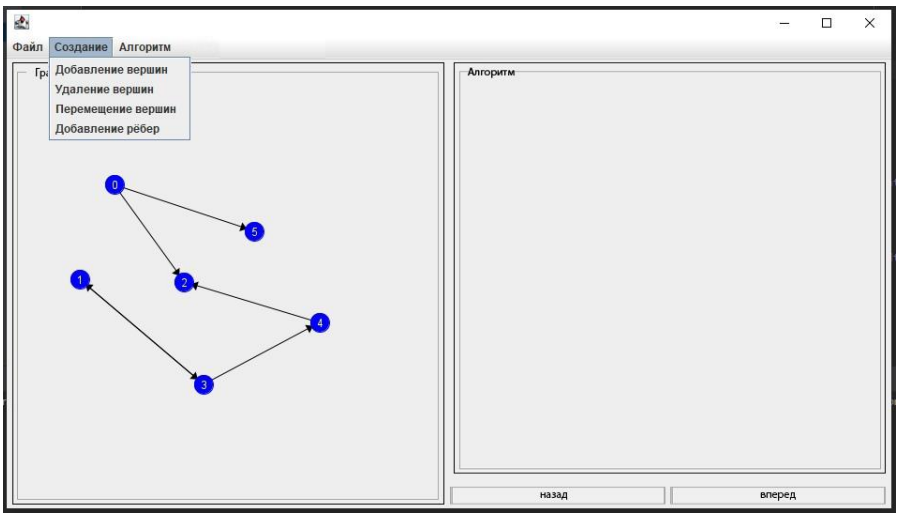

Рис. 3 – Внешний вид приложения (3)

1.2. Уточнение требований после проверки.

1.2.1. После загрузки прототипа.

- Заголовок окна - название алгоритма. (Сделано)
- Не "Пояснение", а "Пояснения". В поле должно помещаться несколько строк. (Сделано)
- Уменьшить начальный размер окна: сейчас после запуска нижняя часть окна оказалась скрыта под панелью задач (а если она слева, а не снизу, то кусок окна справа оказывается за пределами экрана). (Сделано)
- Над окном создания графа добавьте 4 радиокнопки, соответствующие 4 пунктам меню "Создание". (Сделано)

1.2.2. После загрузки 1-й версии.

- Уточнения отсутствуют

1.2.3. После загрузки 2-й версии.

- Перемещение вершин работает, только если перемещать их медленно. (Сделано)
- Кнопки "Назад" и "Вперёд" должны быть неактивны в том случае, когда их нажатие не имеет эффекта. (Сделано)

- Пояснения должны быть выделяемыми и копируемыми. (Сделано)
- По ходу алгоритма удаляемая вершина вместе с инцидентными ей рёбрами должна перекрашиваться. (Не сделано)
- При наведении курсора на вершину в ходе выполнения алгоритма должна всплывать подсказка с текущим количеством вхождений. (Сделано)
- После нажатия "Вывести результат" должна быть возможность делать шаги назад. (Сделано)
- В меню Алгоритм добавьте пункт "Заново", возвращающий состояние программы в начало. (Сделано)

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

- Согласование
    - Распределение ролей
    - Составление спецификации
- Прототип (06.07.2021)
    - Создание прототипа пользовательского интерфейса
    - Разработка интерфейса позволяющего построить граф
- 1-я Итерация (07.07.2021)
    - Реализация алгоритма и вывод результата работы алгоритма в приложение
- 2-я Итерация (09.07.2021)
    - Добавление возможность пошагового исполнения алгоритма
    - Добавление возможности возврата исполнения алгоритма к предыдущим шагам
    - Добавление возможности записи/чтения графа из файла
    - Тестирование приложения

2.2. Распределение ролей в бригаде

1. Николаев Александр (9381) - Разработка пользовательского интерфейса отвечающего за создание (Кнопки создания графа, редактирования, работы алгоритма, привязка к этим компонентам функционал). Визуализация графа.
2. Прашутинский Кирилл (9381) - Создание графа, как некоторой структуры данных и реализация самого алгоритма. Визуализация работы алгоритма.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Структуры данных

Пакет resources

Алгоритм топологической сортировки, предназначен для обработки графа. Хранение графа осуществляется при помощи класса Graph. Граф содержит коллекцию вершин, которые представляют собой объекты класса Node. Каждая вершина хранит список указателей на инцидентные этой вершине вершины, свой ID, который также является и её именем и положение на экране.

Класс UserMeta отвечает за хранение настроек программы и отслеживание действий пользователя. Она хранит поле editMode, типа перечисления EditMode, которое содержит текущий выбранный пользователем режим редактирования (Создать вершину, удалить вершину, переместить вершину, создать ребро, ничего не выбрано).

Пакет UI

Класс MainWindow является главным классом в программе. Он отвечает за GUI и выводит на экран пользователя все компоненты программы.

Класс DrawPanel extends JPanel - реализует рисование графа, а AlgorithmPanel extends JPanel - реализует вывод всей информации об алгоритме пользователю (пояснение к каждому шагу итерации, таблица, содержащая информацию о вершинах, необходимую для выполнения алгоритма и отрисовка полученного графа на каждом шаге итерации). Класс UIStyleSetting отвечает за хранение некоторых констант программы, таких как цвет и радиус вершины, цвет и размер имени вершины, цвет и размер ребра, а также настройки сглаживания и т.п.

3.2. Основные методы

В Graph метод saveGraph() записывает граф в файл для того, чтобы его можно было потом заново загрузить использовать при последующих запусках программы.

В Graph метод loadGraph() создает граф на основе сохраненного файла.

В Graph метод Algorithm() - алгоритм топологической сортировки.

В Graph метод addNode() - позволяет пользователю добавить вершину в граф.

В Graph метод deleteNode() - позволяет пользователю удалить вершину из графа.

В Graph метод move() - позволяет пользователю переместить вершину в графе.

В Graph метод addEdge() - позволяет пользователю добавить ребро в графе.

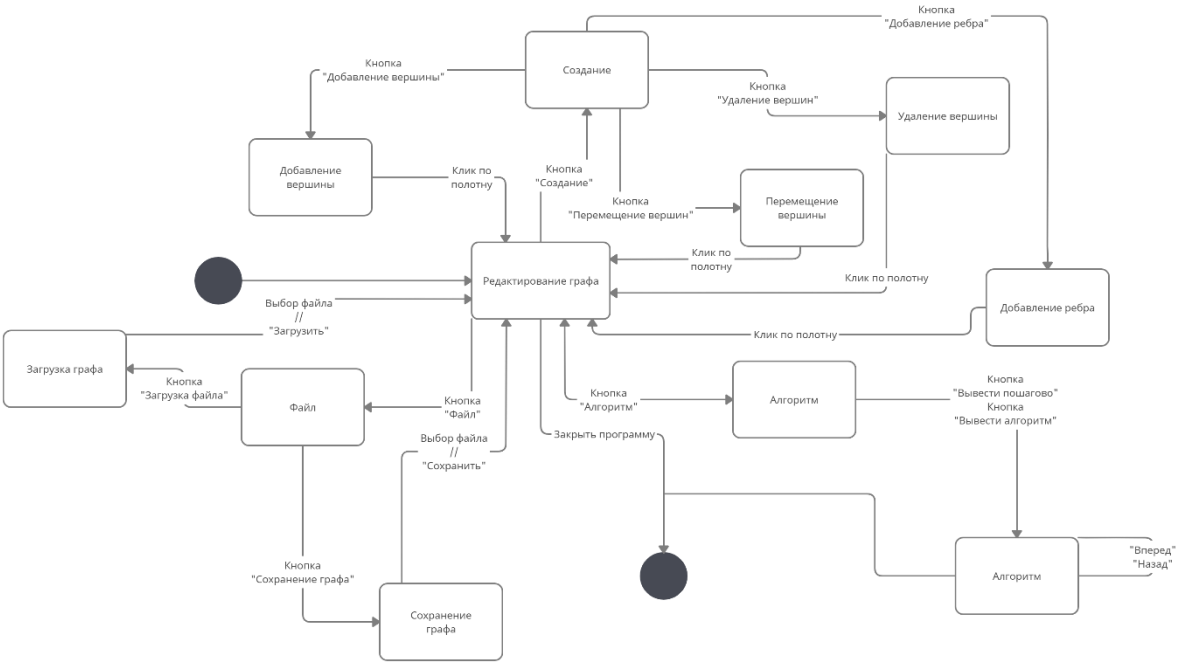

Рис. 4 – Диаграмма состояний программы

4. ТЕСТИРОВАНИЕ

4.1. План тестирования программы.

Проведение тестирования программы планируется с использованием библиотеки для модульного тестирования Junit. Объектами тестирования станут ключевые методы, используемые в алгоритме Топологической сортировки.

Объект тестирования	Данные	Результаты
Метод addEdge()	Добавленные в граф ребра.	Проверка добавления ребра в граф. Количество добавленных ребер.
Метод addNode()	Добавленные в граф вершины.	Проверка добавления вершины в граф.
Метод saveGraph()	Созданных граф из определенного числа вершин и ребер	Запись графа в файл, а затем загрузка этого же файла для проверки правильности записи графа.
Метод loadGraph()		Проверка данных в файле для создания графа
Метод deleteNode()		Проверка удаления вершины из графа.

4.2. Результаты тестирования.

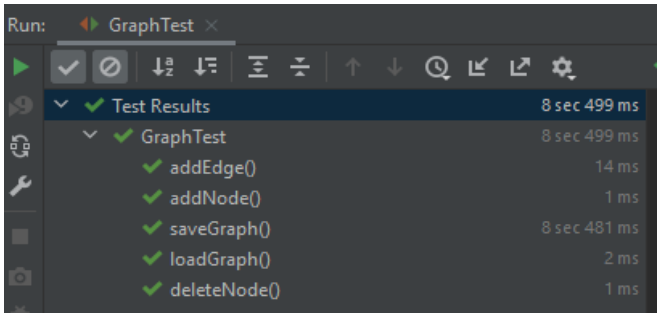

Рис. 5 – результаты тестирования

ЗАКЛЮЧЕНИЕ

Таким образом, командой была реализована работа и визуализация алгоритма Топологической сортировки. Пользователь имеет возможность задать входные данные двумя способами – вручную и с помощью загрузки из файла. Есть возможность пошагового выполнения алгоритма.

В ходе совместной работы были получены практические знания о ЯП Java, системе тестирования Junit и навыках командной разработки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Описание работы с JUnit // JUnit 5 User Guide. URL: https://junit.org/junit5/docs/current/user-guide/#writing-tests-classes-and-methods (дата обращения: 11.07.2021).
2. Описание алгоритма топологической сортировки // Wikipedia. URL: https://ru.wikipedia.org/wiki/Топологическая_сортировка (дата обращения: 3.07.2021).
3. Описание ЯП Java // Java Platform, Standard Edition 7 API Specification. URL: https://docs.oracle.com/javase/7/docs/api/overview-summary.html (дата обращения: 3.07.2021-11.07.2021).

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД АЛГОРИТМА.

URL: https://github.com/kirja1980/ETU_PR_2021