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

ОТЧЕТ

по учебной практике

Тема: Кратчайшие пути в графе.

Алгоритм Дейкстры.

Студент гр. 1303 _____ Смирнов Д.Ю.

Студент гр. 1303 _____ Иевлев Е.А.

Студент гр. 1303 _____ Чернуха В.В.

Руководитель _____ Шестопалов Р.П.

Санкт-Петербург
2023

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Смирнов Д.Ю. группы 1303

Студент Иевлев Е.А. группы 1303

Студент Чернуха В.В. группы 1303

Тема практики: Выполнение мини-проекта на языке *Kotlin*.

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на *Kotlin* с графическим интерфейсом.

Алгоритм: Поиск кратчайших путей в графе. Алгоритм Дейкстры.

Сроки прохождения практики: 30.06.2023 – 13.07.2023

Дата сдачи отчета: 12.07.2023

Дата защиты отчета: 12.07.2023

Студент		Смирнов Д.Ю.
Студент		Иевлев Е.А.
Студент		Чернуха В.В.
Руководитель		Шестопалов Р.П.

АННОТАЦИЯ

Результатом выполнения учебной практики является готовое приложение с графическим интерфейсом, наглядно показывающее работу алгоритма Дейкстры на графе с неотрицательными рёбрами. Содержание отчета включает в себя следующие части: спецификация (требования) программы, распределение ролей в команде и план выполнения проекта, описание алгоритма, основных классов и структур данных, тестирование, вывод по результатам работы.

SUMMARY

The result of the training practice is a ready-made application with a graphical interface that clearly shows the operation of Dijkstra's algorithm on a graph with non-negative edges. The content of the report includes the following parts: specification (requirements) of the program, distribution of roles in the team and project execution plan, description of the algorithm, main classes and data structures, testing, and conclusion based on the results of the work.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью практической работы является создание графического приложения, наглядно демонстрирующего работу алгоритма Дейкстры на графе. Для этого необходимо реализовать алгоритм, графический интерфейс и связать их между собой. Далее необходимо протестировать работу алгоритма.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Требования к входным данным

Программе подаётся на вход неориентированный граф с неотрицательными весами ребер и начальная вершина. Вершины будут имеет название из одного или двух символов. Алгоритм может работать как на связном графе так и на несвязном графе. В конце работы алгоритма рядом с каждой вершиной будет отображатся минимальный вес пути от начальной вершины до этой, если путь между ними существует, иначе символ бесконечности.

1.2. Требования к визуализации

Пользовательский интерфейс программы представлен на рисунке 1.

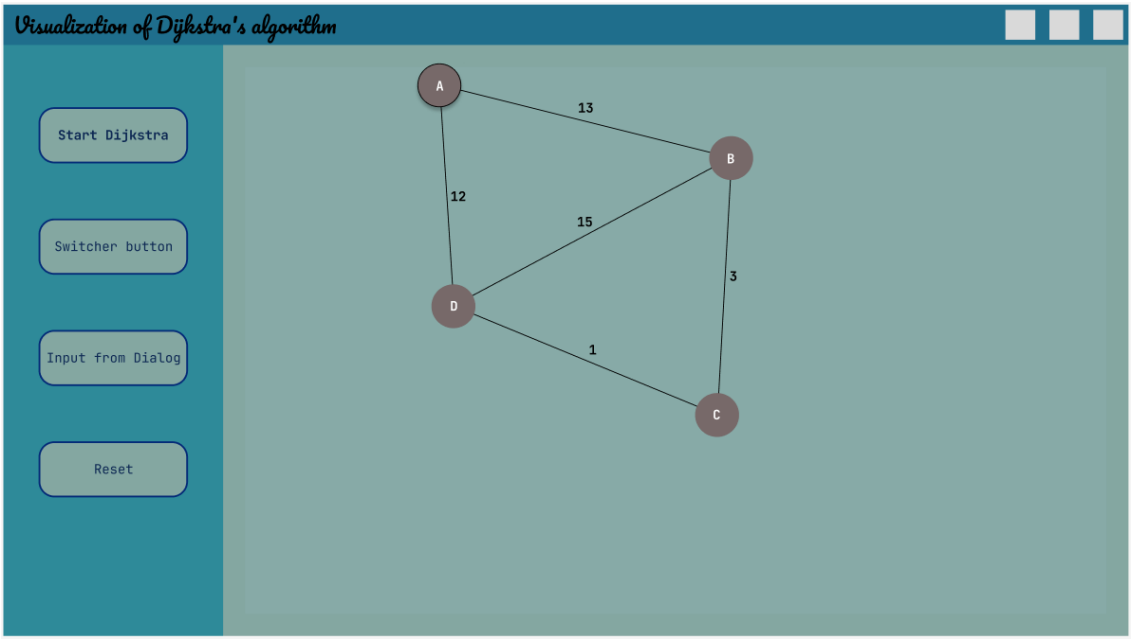

Рисунок 1 - Пользовательский интерфейс программы

Назначение кнопок в верхней панели приложения:

- Левая кнопка — открывает диалоговое окно с информацией о приложении.
- Средняя кнопка — сворачивает приложение.
- Правая кнопка — закрывает приложение.

Назначение кнопок главного меню:

- Кнопка “*Start Dijkstra*” - запускает работу алгоритма, если была выбрана начальная вершина, иначе выдает сообщение об ошибке.
- Кнопка “*Switcher button*” - переключает режим работы рабочей области
- Кнопка “*Input from Dialog*” - предоставляет возможность ввода графа через диалоговое окно в формате
- Кнопка “*Reset*” - очищает рабочую область (удаляет граф).

Справа от главного меню расположена рабочая область, которая в зависимости от режима позволяет интерактивно расставлять вершины и соединять их ребрами. При нажатии правой кнопкой по вершине, предоставляется возможность её переименования и удаления через контекстное меню. Таким же образом через правую кнопку мыши можно удалить ребро или изменить его вес.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЯ РОЛЕЙ В КОМАНДЕ

2.1. План разработки

1. Согласование спецификации и плана разработки - по 3 июля.
2. Разработка прототипа программы - приложение с графическим интерфейсом без реализации основной логики, консольный вариант алгоритма Дейкстры, написание основных структур данных - по 5 июля.
3. Разработка первой версии программы (бета) - Обработка нажатий кнопок главного меню приложения, написание обработчиков рабочей области, написание пошагового алгоритма Дейкстры - по 7 июля
4. Разработка второй версии программы - Обработка исключений, написание логики ввода графа через диалоговое окно, добавление пошаговой визуализации алгоритма Дейкстры - по 10 июля.
5. Написание отчета - по 12 июля

2.2. Распределение ролей в команде

Чернуха В.В. - алгоритм Дейкстры, тестирование

Иевлев Е.А. - Макет приложения, *GUI*, обработка исключительных состояний

Смирнов Д.Ю. - Структуры данных, связь графической части программы с основной логикой

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Использованные структуры данных

Все вершины отображаемые на рабочей области описаны классом *Vertex*. Данная класс хранит в себе координаты центра окружности, название вершины, список инцидентных ребер, а также необходимую информацию для алгоритма Дейкстры такую как: текущая стоимость пути в данную вершину и метка вершины (посещена / не посещена).

Ребра графа в свою очередь описываются классом *Edge*. Данный класс содержит в себе ссылки на две вершины и вес ребра.

3.2. Основные методы

В классе *GraphController* можно выделить несколько основных методов:

- *preInitAlgorithm()* - выполняет пре инициализацию алгоритма Дейкстры.
- *makeStep()* - выполняет один шаг алгоритма, если это возможно.
- *afterAlgorithm()* - восстанавливает исходное состояние графа до запуска решения.
- *setHandlersVertex()* - устанавливает обработчики событий для вершины.
- *createVertex()* - позволяет создавать вершину на координатах события.
- *SetHandlersEdge()* - устанавливает обработчики событий для ребра.
- *edgeCreation()* - создает ребро связывающее две вершины.
- *drawGraph()* - отвечает за рисование графа на рабочей области.
- *clear()* - очищает рабочую область от ребер и вершин.

UML-диаграмма классов представлена в приложении А.

3.3. Графический интерфейс

Для разработки графического интерфейса была выбрана платформа *JavaFX*, поддерживающая *CSS* и *FXML*, а также обладающая инструментом

JavaFX Scene Builder. Поскольку используя инструмент *Scene Builder* можно создавать *FXML* файлы без непосредственного написания кода, а просто дизайнерского проектирования, а также простой последующей интеграции их в код программы, и добавления логики виджетам, была выбрана именно эта платформа.

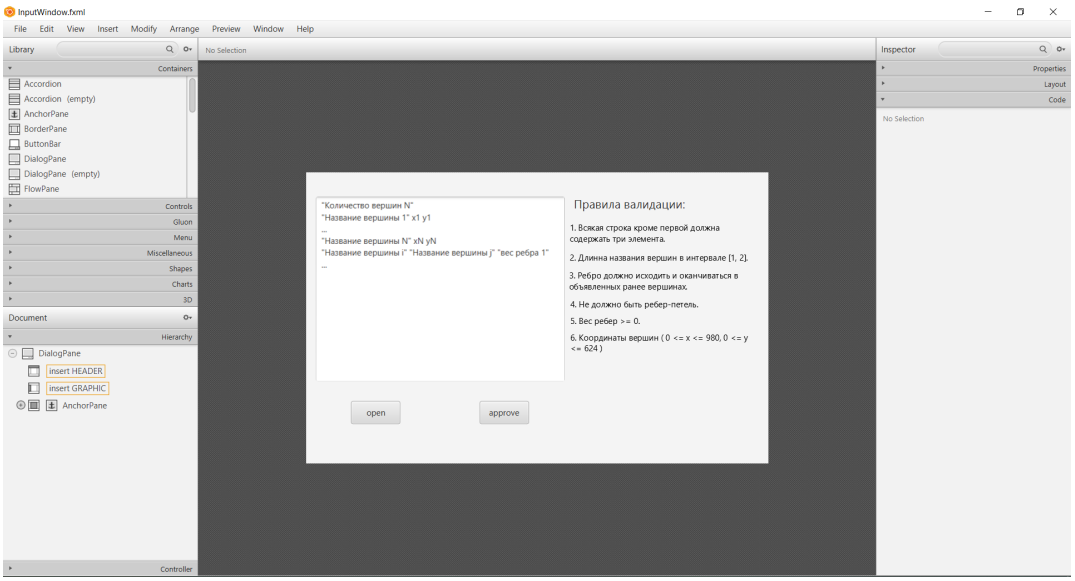

Рис. 2 - написание окна ввода в JavaFX Scene Builder

В JavaFX для связывания логики приложения с соответствующими элементами интерфейса, описанными в FXML файле, используются классы контроллеры.

После загрузки FXML файла (определяющего графический дизайн интерфейса) и отвечающего за него контроллер, в месте программы, где нужно вызвать окно, задаются его параметры размещения, после чего специальным методом вызывается открытие этого окна.

По такому принципу реализованы все пользовательские диалоговые окна (окна alert встроены и вызываются соответствующими методами).

Язык *CSS* позволяет стилизовать пользовательский интерфейс приложения, что являлось плюсом при выборе платформы для работы с *GUI*.

В программе реализовано два *FXML* файла - *main* (описание главного окна *GUI*) и *InputWindow* (описание собственного диалогового окна для ввода

данных), также есть файл *main.css*, настраивающий стили главного окна приложения.

4. ТЕСТИРОВАНИЕ

1. При создании 26 вершин новые вершины щелчком мыши не создаются (т.к. по умолчанию щелчком мыши создаются вершины с одной из букв английского алфавита, соответственно при 26 вершинах все буквы уже заняты)

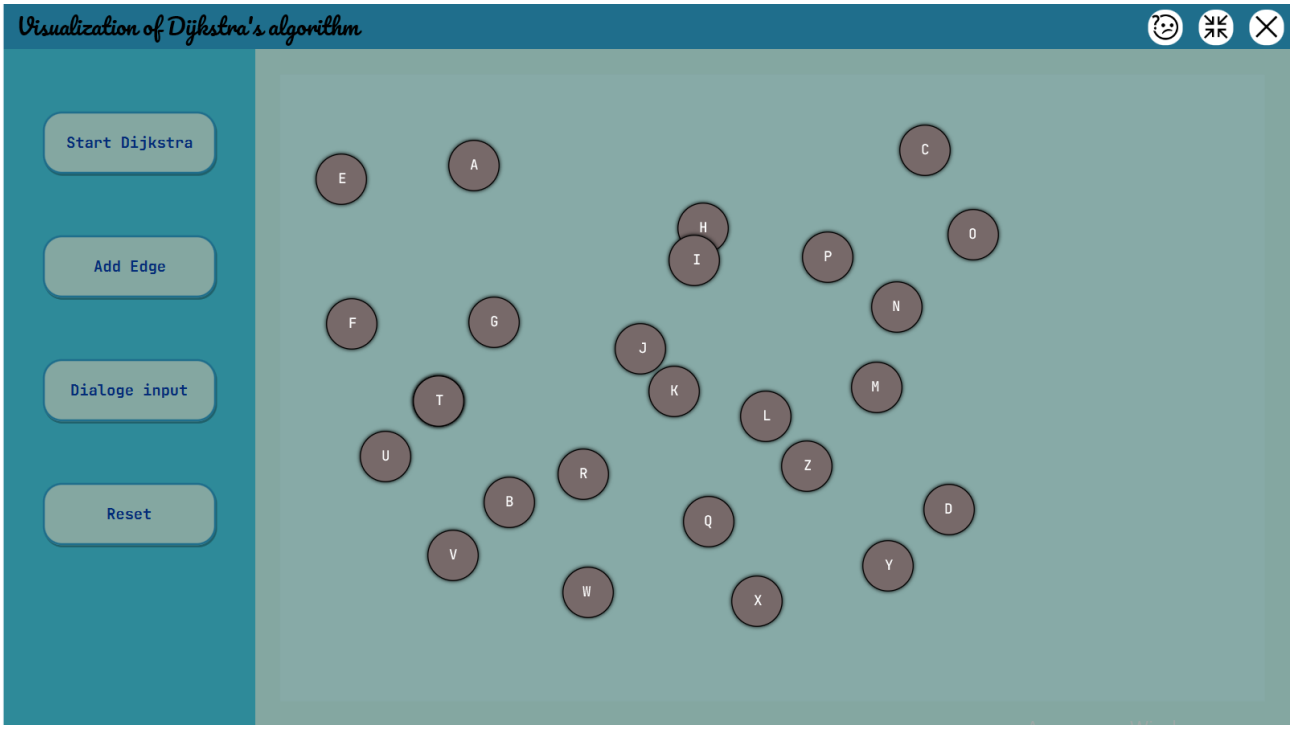

Рис. 3 - щелчком мыши созданы 26 вершин

2. При попытке назвать вершину строкой состоящей из больше чем 2 символов вылетает предупреждение о том что так делать нельзя

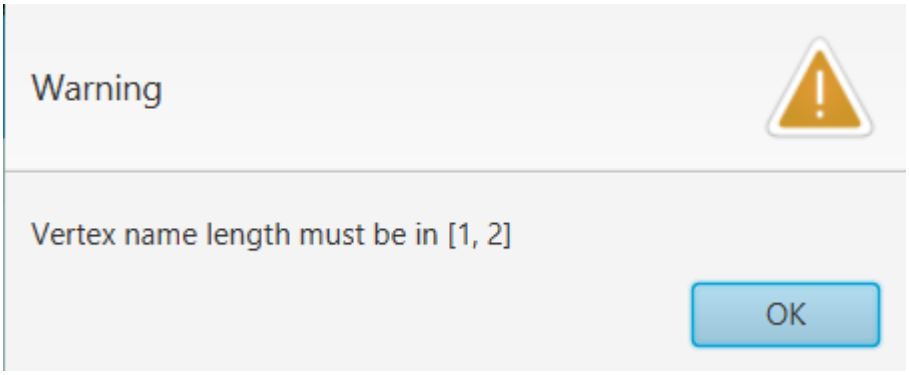

Рис. 4 - предупреждение о превышение длины имени

3. При попытке назвать вершину именем которое уже есть то также появляется предупреждение что так делать не стоит.

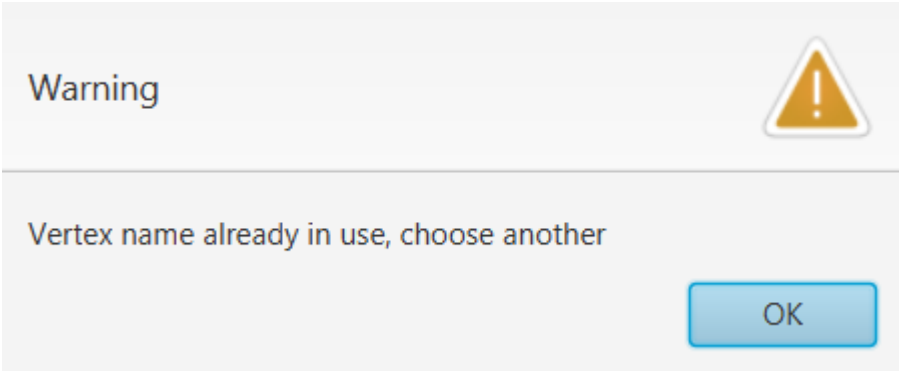

Рис. 5 - предупреждение о том, что такое имя уже есть

4. При попытке дать ребру невалидный вес то появляется предупреждение.

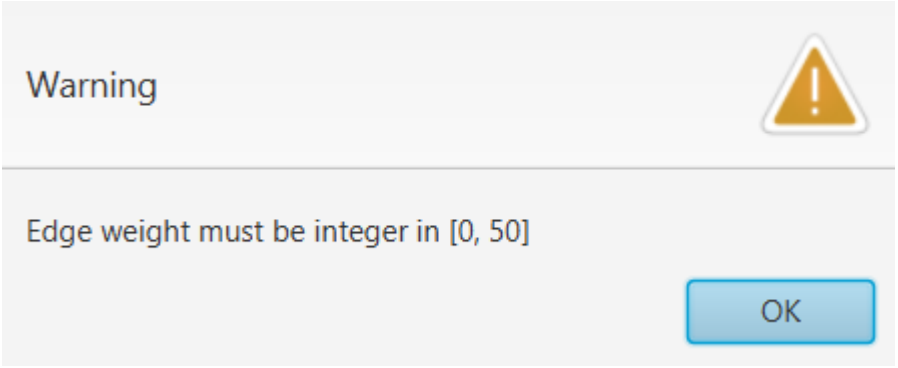

Рис. 6 - предупреждение о том, что такой вес не валиден

5. При попытке запуска алгоритма без выбранной стартовой вершины выпадет ошибка

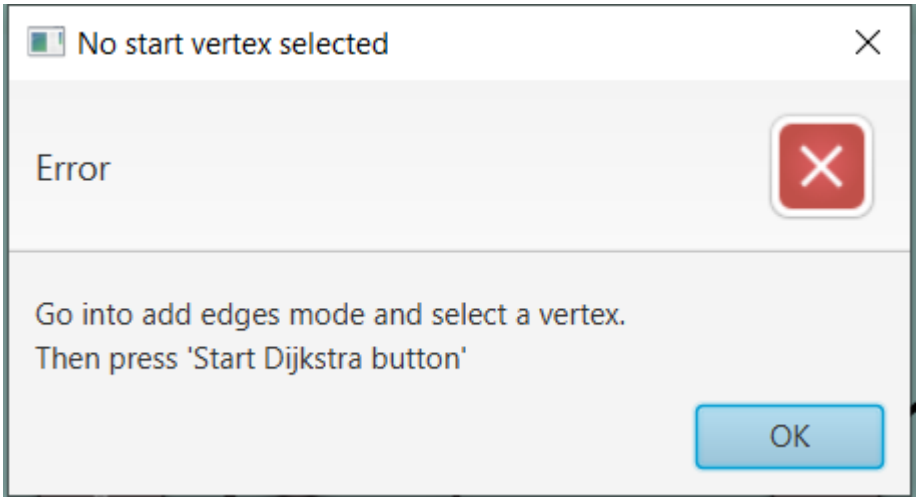

Рис. 7 - предупреждение о том, что стартовая вершина не выбрана

6. При попытке ввести пустое поле в окно ввода графа выведется предупреждение

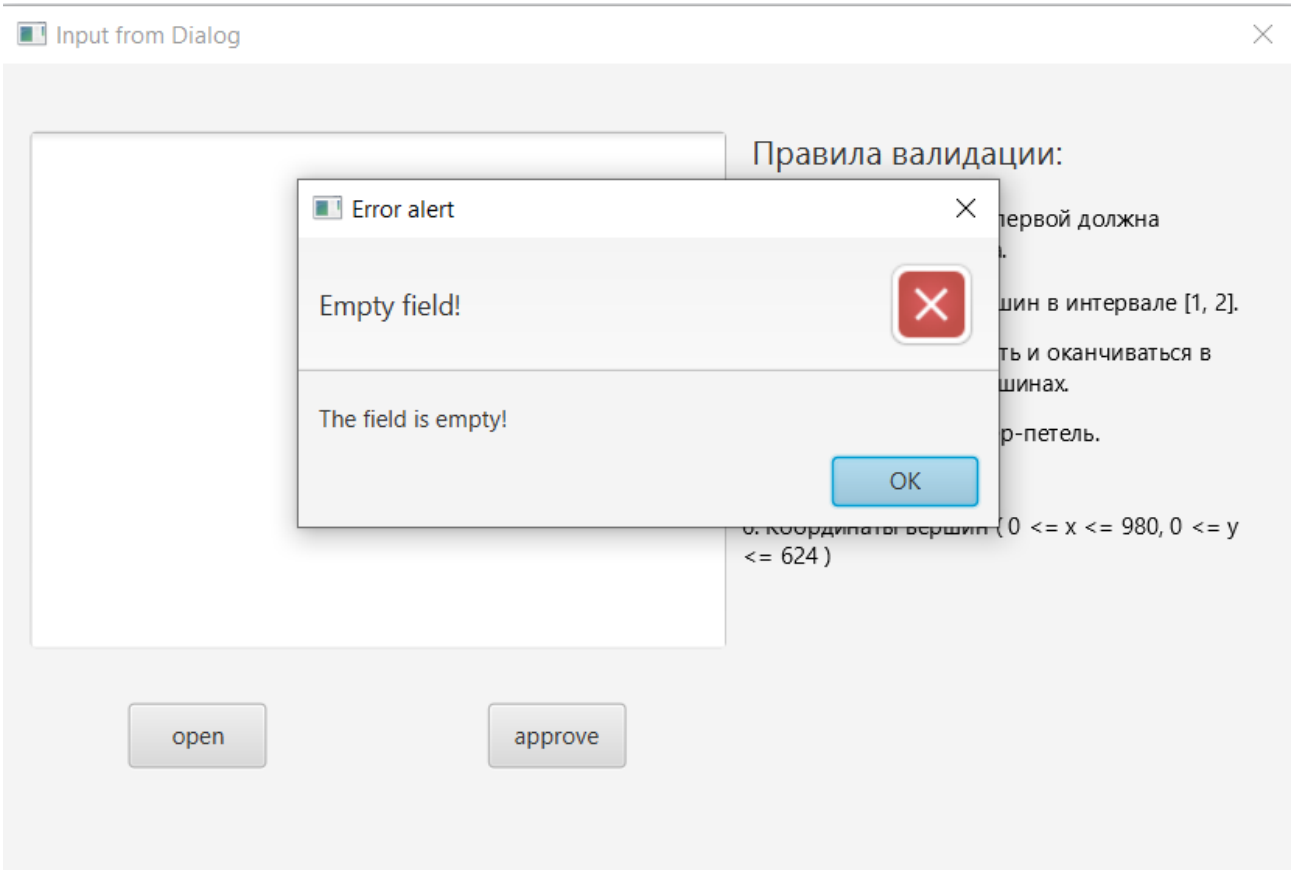

Рис. 8 - предупреждение о том, что поле пустое

7. При неверно введенных координатах выведется предупреждение

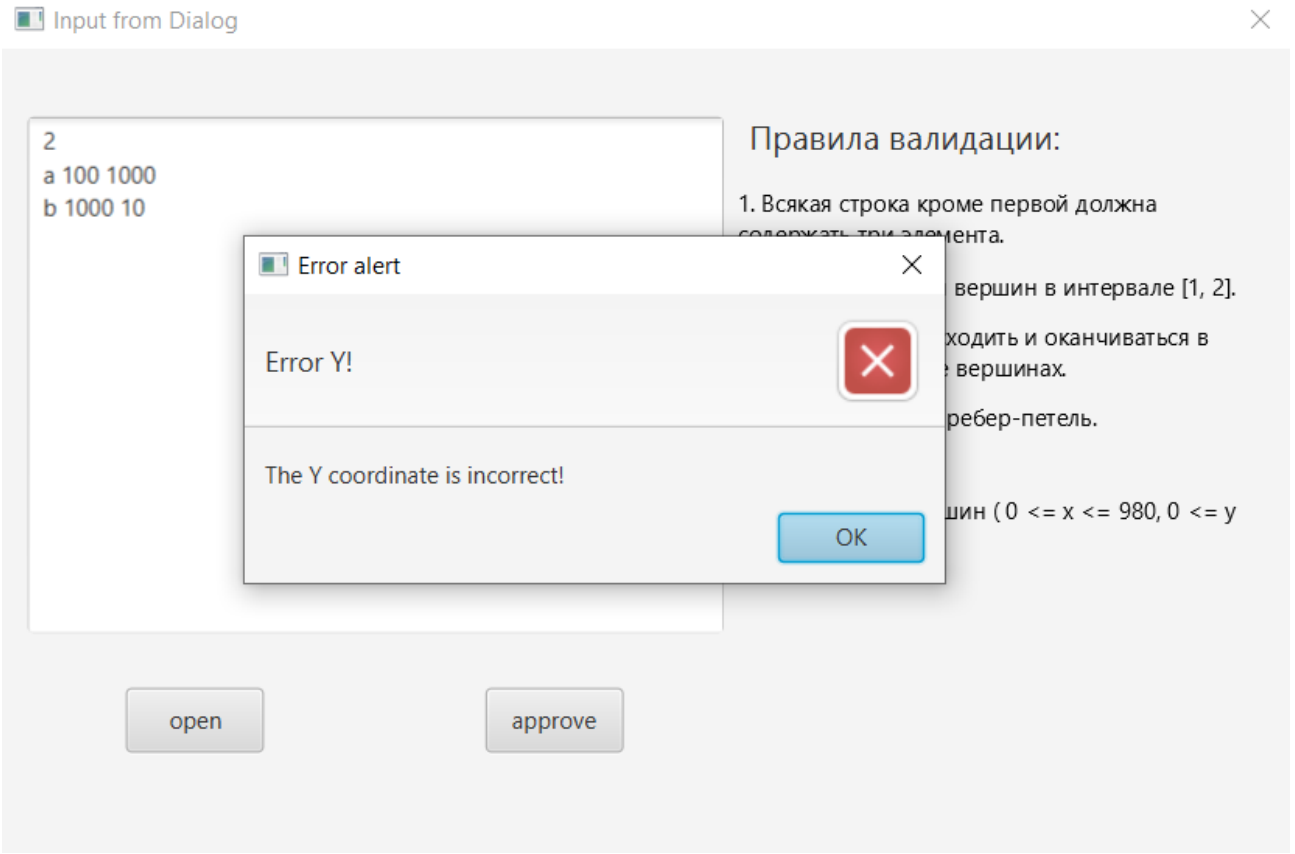

Рис. 9 - предупреждение о том, что координаты неверны

8. При неверно указанных вершинах также выводится предупреждение

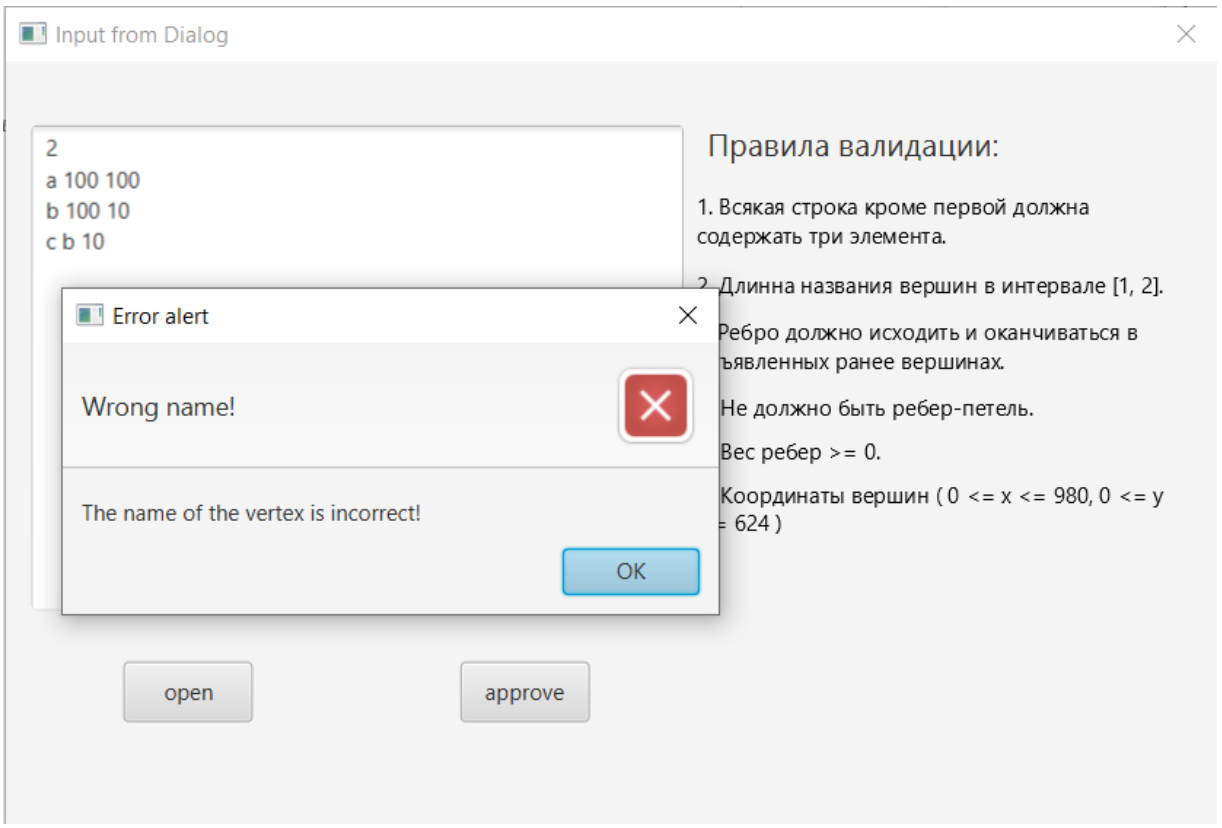

Рис. 10 - предупреждение о том, что вершина указана неверно

9. Также выполняется проверка на пустые строчки в вводе

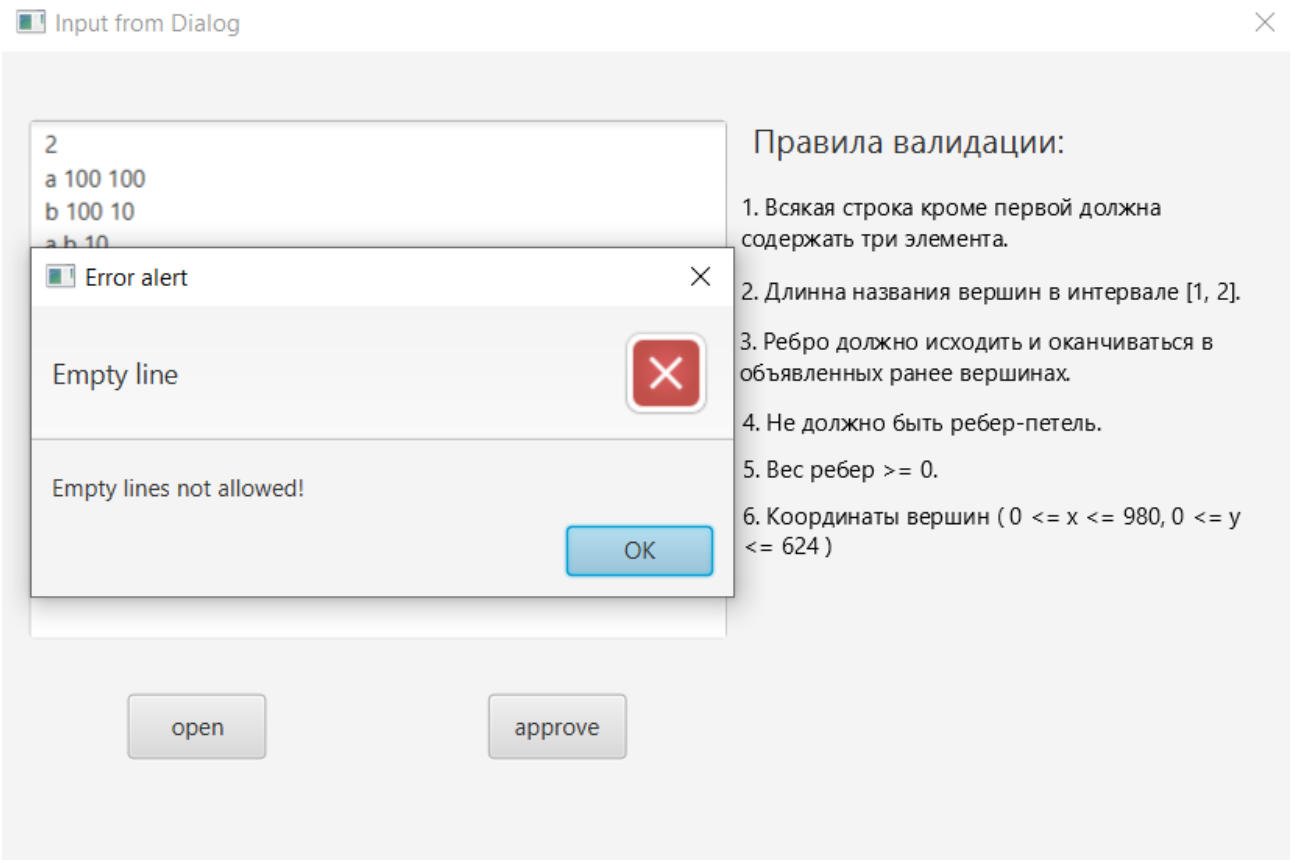

Рис. 11 - предупреждение о том, что в вводе есть пустые строки

10. При неправильном вводе веса ребра выводится предупреждение

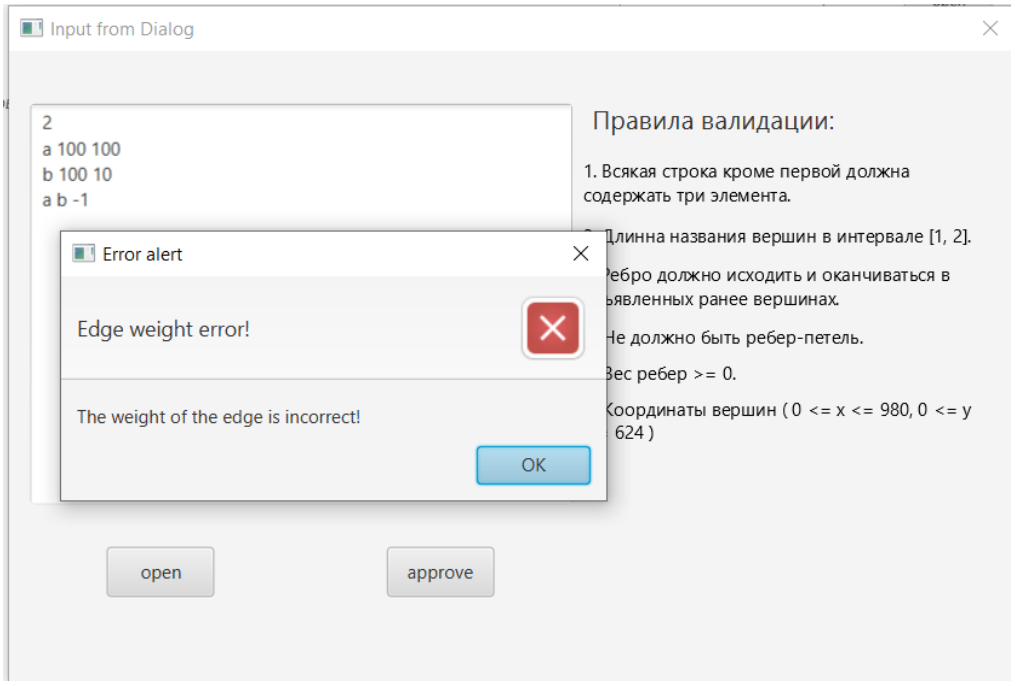

Рис. 12 - предупреждение о том, что вес ребра не валиден

11. При попытке создать петлю выводиться ошибка

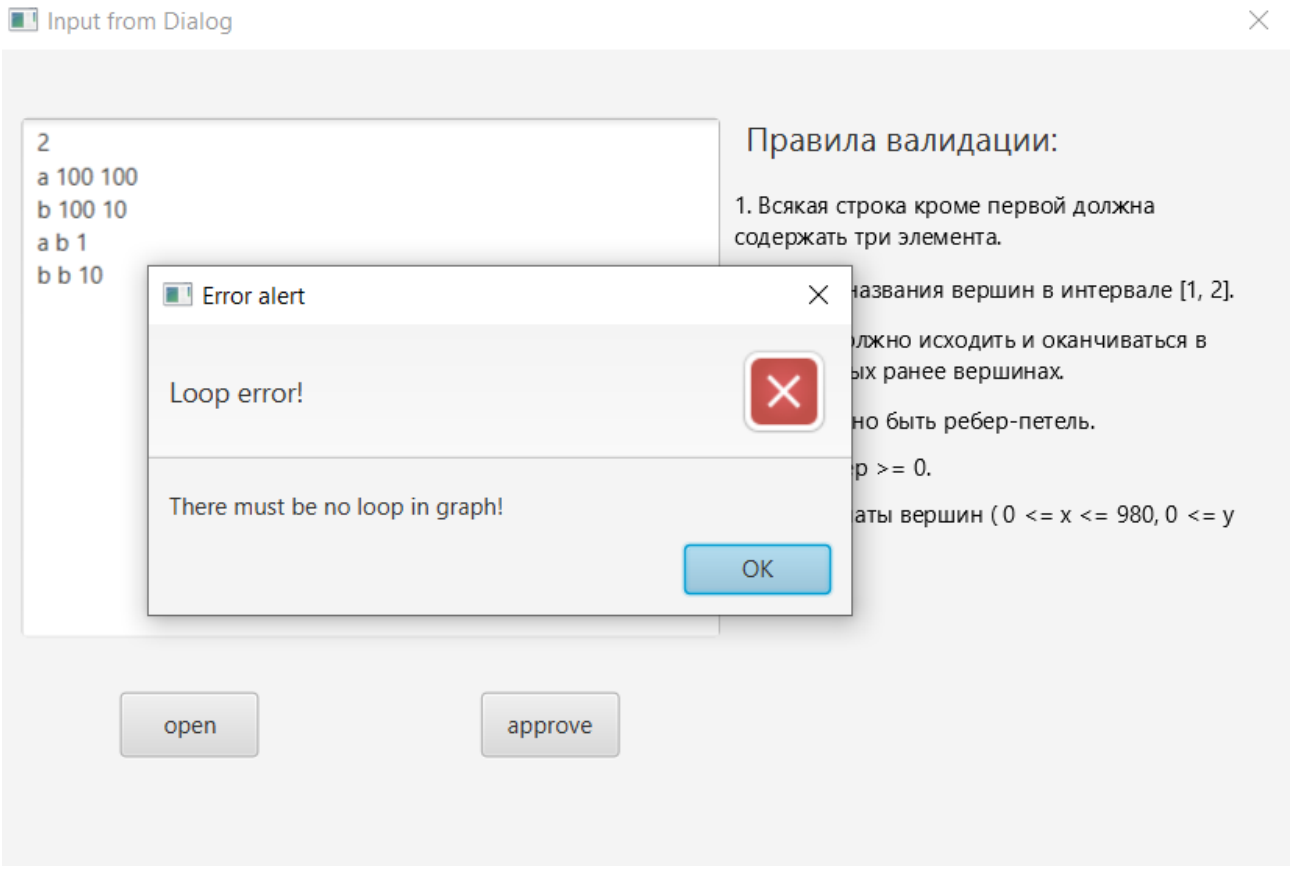

Рис. 13 - предупреждение о том, что петли не разрешены

ЗАКЛЮЧЕНИЕ

В ходе работы были составлены спецификация, план работы над проектом, были разделены роли в команде, составлена UML диаграмма и макет интерфейса, на основе которых и была создана программа. В результате выполнения практического задания был изучен процесс создания приложений с графическим интерфейсом на языке Kotlin с применением библиотеки JavaFX, было разработано приложение, позволяющее пошагово проследить работу алгоритма Дейкстры на неориентированном графе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальная документация языка Kotlin [Электронный ресурс]. URL: https://kotlinlang.org/docs/home.html (Дата обращения: 30.06.2023)
2. Официальная документация пакета JavaFX [Электронный ресурс].URL: https://docs.oracle.com/javase/8/javafx/api/toc.htm (Дата обращения: 4.07.2023)
3. Репозиторий с мини-проектом команды. URL: https://github.com/D1mitrii/LETI-educational-practice

ПРИЛОЖЕНИЕ А

UML-ДИАГРАММА