Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт Информационных Технологий и Анализа Данных

Кафедра вычислительной техники

**Название работы** – “Разработка приложения с оконным графическим пользовательским интерфейсом”

Отчет по лабораторной работе “Лабораторная работа №5”

по дисциплине Объектно-ориентированное программирование

Вариант 24

Выполнил

Студент, номер группы ИСМб-19-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Д. Солопов

Дата: \_\_\_\_\_\_\_\_\_\_

Принял

Должность \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.В. Маланова

Дата: \_\_\_\_\_\_\_\_\_\_

Иркутск 2021 г.

Содержание

[1 Введение 3](#_Toc65099688)

[2 Постановка задач 4](#_Toc65099689)

[3 Проектирование графического интерфейса 5](#_Toc65099690)

[4 Проектирование классов 20](#_Toc65099691)

[5 Таблица тестов 25](#_Toc65099692)

[6 Результаты тестов 28](#_Toc65099693)

[7 Листинг исходного кода 42](#_Toc65099694)

[8 Заключение 56](#_Toc65099695)

1 Введение

Современный мир невозможно представить без компьютерных технологий. Одной из самых важных частей каждого компьютера является набор программ, которые управляют работой компьютера и предоставляют возможности в различных сферах человеческой деятельности. Например, графические редакторы, среды разработки (IDE), системы автоматизированного проектирования (САПР), системы моделирования, инженерные программы, бухгалтерские программы и многие другие.

Каждая программа предоставляет тот или иной функционал, который может быть использован пользователем для достижения тех или иных целей. Современная программа предоставляет пользователю возможность пользоваться функционалом посредством пользовательского графического интерфейса (GUI – Graphical User Interface). Однако, у программы может быть несколько интерфейсов: графический и (или) консольный. Консольный интерфейс программы не содержит элементов управления. Например, таких как кнопка (Button) и флажки (CheckBox). От графического интерфейса зависит то, как удобно будет пользоваться функционалом программы. Например, можно создать такой интерфейс управления, что функционал будет доступен при нажатии на определённую кнопку, а можно создать такой, что для вызова того же функционала потребуется большое число нажатий на кнопки, различные переходы и возможно пропись команд в специально созданный для этого элемент (TextField).

В данной работе реализуется графический интерфейс, предоставляющий пользователю широкий функционал для решения задач согласно индивидуальному варианту.

2 Постановка задач

**Цель работы**

Получение общего представления о шаблоне программирования Модель/Вид/Контроллер и его практического применения.

Получение практического опыта работы с базовыми графическими библиотеками java.

**Задача**

a. Определите, является ли одна строка подстрокой другой.

b. Найти длину указанного слова в предложении.

c. Выведите из строки, содержащей слова, разделенные пробелами и запятыми, центральное слово (если в предложении два центральных слова, выведите оба).

3 Проектирование графического интерфейса

Графический интерфейс был спроектирован на основе библиотеки JavaFX. Интерфейс состоит из одного окна, расположенных определённых образом таких управляющих элементов как: Label, TextField, Button, TextArea и ComboBox. Управляющие элементы в свою очередь определённым образом были расположены в таких панелях компоновки, как VBox (все элементы располагаются вертикально) и HBox (все элементы располагаются горизонтально). Интерфейс был спроектирован с учётом возможных

Для упрощения организации и работы с интерфейсом в JavaFX был использован язык разметки FXML, который создан на основе XML. Также, была использована программа SceneBuilder, позволяющая корректно создавать файл расширения \*.fxml, содержащий теги определяющие внешний вид интерфейса. Любой файл \*.fxml, содержащий в себе разметку интерфейса, можно добавить на окно, реализация данного подключения будет представлена в исходном коде программы.

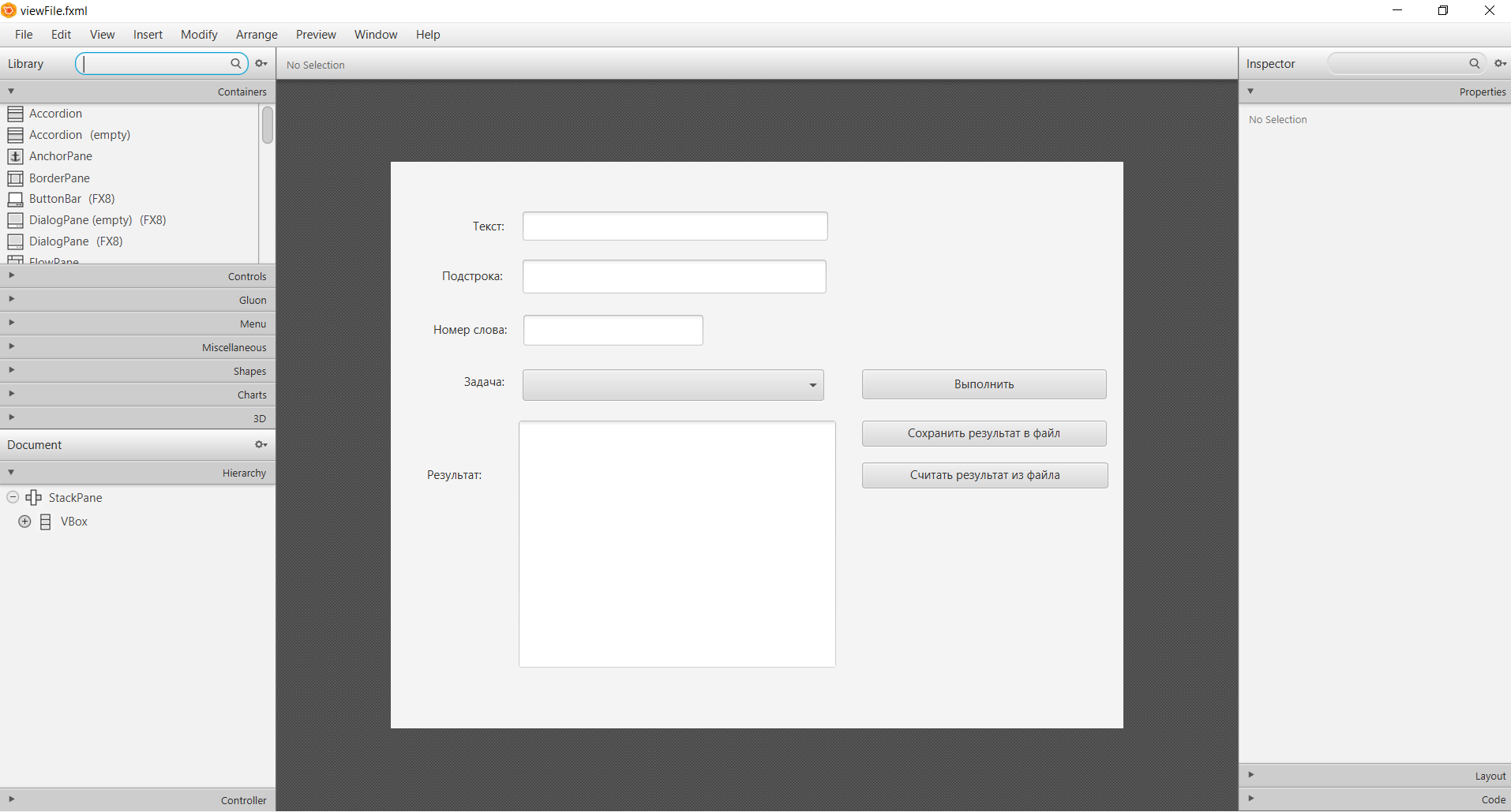


Рисунок 1 – Проектирование интерфейса в SceneBuilder

С помощью SceneBuilder был спроектирован интерфейс и в файл viewFile.fxml были добавлены теги компонентов, их расположение на окне, уникальные идентификаторы элементов управления, размеры и многое другое.

Содержимое файла viewFile.fxml:

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ComboBox?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TextArea?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.StackPane?>

<?import javafx.scene.layout.VBox?>

<StackPane fx:id="root" xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="programGUI.Controller">

<children>

<VBox prefHeight="574.0" prefWidth="742.0">

<children>

<HBox prefHeight="50.0" prefWidth="742.0" />

<HBox prefHeight="29.0" prefWidth="742.0">

<children>

<Label alignment="CENTER\_RIGHT" prefHeight="71.0" prefWidth="115.0" text="Текст:" textAlignment="CENTER" />

<HBox prefHeight="30.0" prefWidth="18.0" />

<TextField id="\_txtBase" alignment="CENTER" prefHeight="30.0" prefWidth="309.0" />

</children>

</HBox>

<HBox prefHeight="19.0" prefWidth="742.0" />

<HBox prefHeight="34.0" prefWidth="742.0">

<children>

<Label alignment="CENTER\_RIGHT" prefHeight="47.0" prefWidth="113.0" text="Подстрока:" />

<HBox prefHeight="34.0" prefWidth="20.0" />

<TextField id="\_txtSubstring" alignment="CENTER" prefHeight="34.0" prefWidth="308.0" />

</children>

</HBox>

<HBox prefHeight="21.0" prefWidth="742.0" />

<HBox prefHeight="31.0" prefWidth="742.0">

<children>

<Label alignment="CENTER\_RIGHT" prefHeight="39.0" prefWidth="118.0" text="Номер слова:" />

<HBox prefHeight="31.0" prefWidth="16.0" />

<TextField id="\_txtNumberWord" alignment="CENTER" prefHeight="48.0" prefWidth="182.0" />

</children>

</HBox>

<HBox prefHeight="24.0" prefWidth="742.0" />

<HBox prefHeight="33.0" prefWidth="742.0">

<children>

<Label alignment="CENTER\_RIGHT" prefHeight="26.0" prefWidth="115.0" text="Задача:" />

<HBox prefHeight="26.0" prefWidth="18.0" />

<ComboBox id="\_cmbTask" prefHeight="32.0" prefWidth="305.0" />

<HBox prefHeight="34.0" prefWidth="38.0" />

<Button id="\_btnStart" mnemonicParsing="false" prefHeight="30.0" prefWidth="248.0" text="Выполнить" />

</children>

</HBox>

<HBox prefHeight="18.0" prefWidth="742.0" />

<HBox prefHeight="250.0" prefWidth="742.0">

<children>

<Label alignment="CENTER" prefHeight="110.0" prefWidth="130.0" text="Результат:" />

<TextArea id="\_txtResult" prefHeight="250.0" prefWidth="322.0" />

<HBox prefHeight="250.0" prefWidth="26.0" />

<VBox prefHeight="250.0" prefWidth="265.0">

<children>

<Button id="\_btnSave" mnemonicParsing="false" prefHeight="26.0" prefWidth="248.0" text="Сохранить результат в файл" />

<VBox prefHeight="16.0" prefWidth="266.0" />

<Button id="\_btnRead" mnemonicParsing="false" prefHeight="26.0" prefWidth="249.0" text="Считать результат из файла" />

</children>

</VBox>

</children>

</HBox>

</children>

</VBox>

</children>

</StackPane>

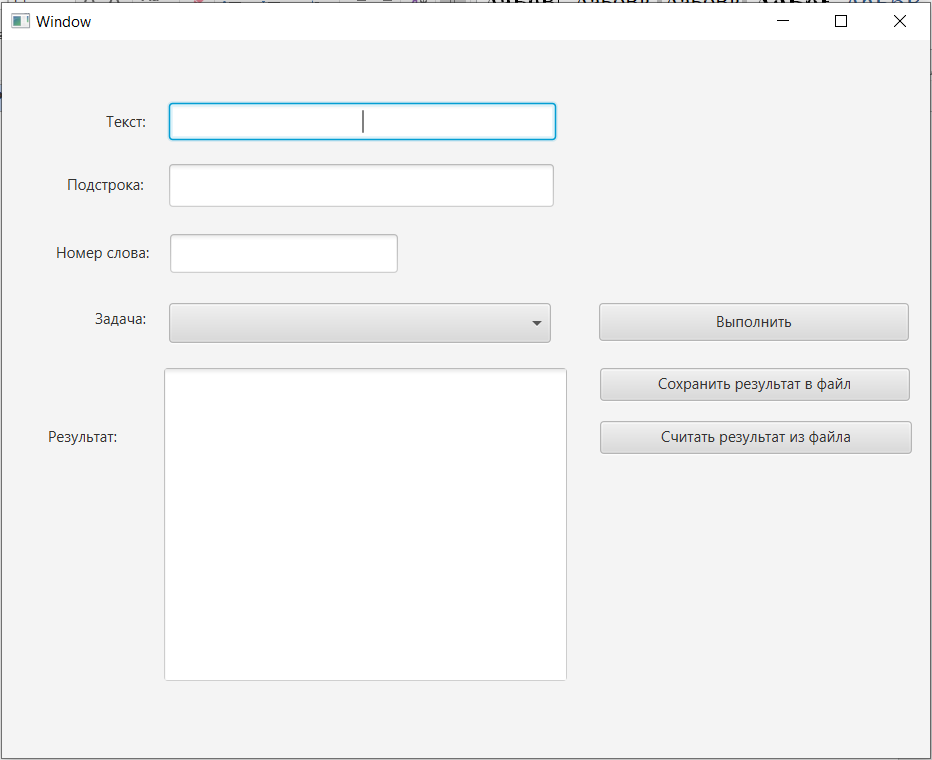


Рисунок 2 – Приложение с спроектированным графическим интерфейсом

На рисунке 2 представлен графический интерфейс приложения. Для решения той или иной задачи необходимо записать строку в текстовое поле (TextField), перед которым стоит элемент управление Label, с названием “Текст”.

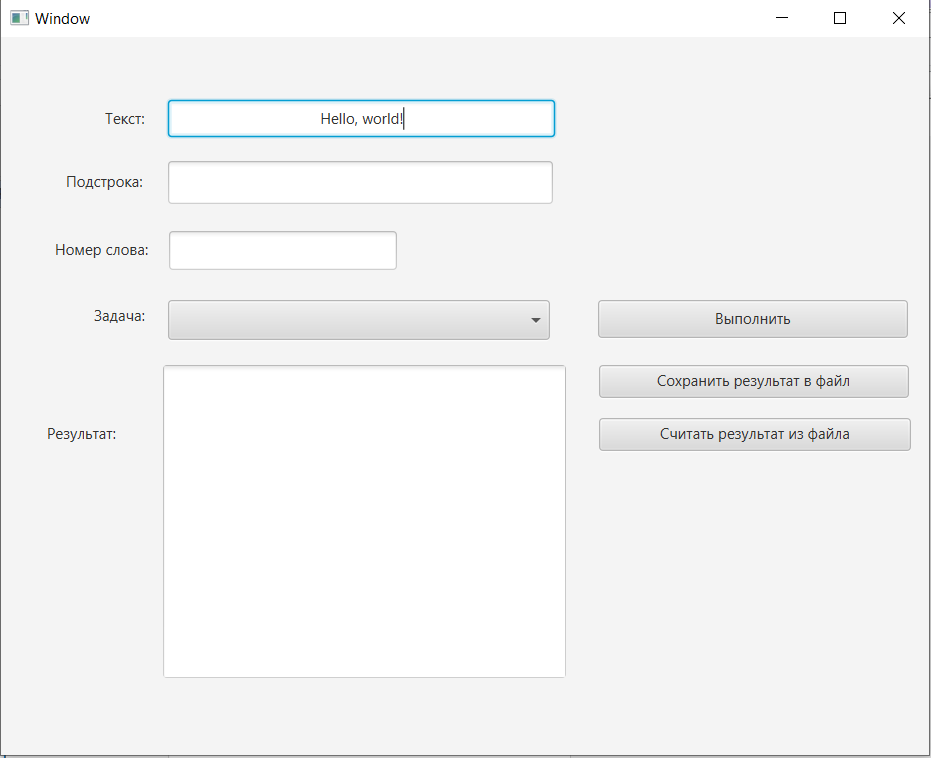


Рисунок 3 – Запись текста в текстовое поле

Далее, необходимо выбрать задачу, которую необходимо решить. Для этого необходимо выбрать один из элементов выпадающего списка, для решения той или иной задачи. Стоит отметить, что каждая задача требует различных входных данных.

Все задачи требуют, чтобы текстовое поле напротив элемента управления Label с названием “Текст:” было заполнено.

Задача “Определить, является ли подстрока другой строки” требует, чтобы текстовое поле напротив элемента управления Label с названием “Подстрока:” было заполнено.

Задача “Найти длину указанного слова в предложении” требует, чтобы текстовое поле напротив элемента управления Label с названием “Номер слова:” содержала в себе число (в случае не корректности числа – будет выведена ошибка).

Задача “Определить центральное слово” требует лишь заполнения текстового поля напротив элемента управления Label с название “Текст:”.

В случае, если пользователь введёт или забудет ввести входные данные, которые не будут удовлетворять требованиям решения задач будет выведена ошибка и никаких действий, по отношению к решению задач, предпринято не будет.

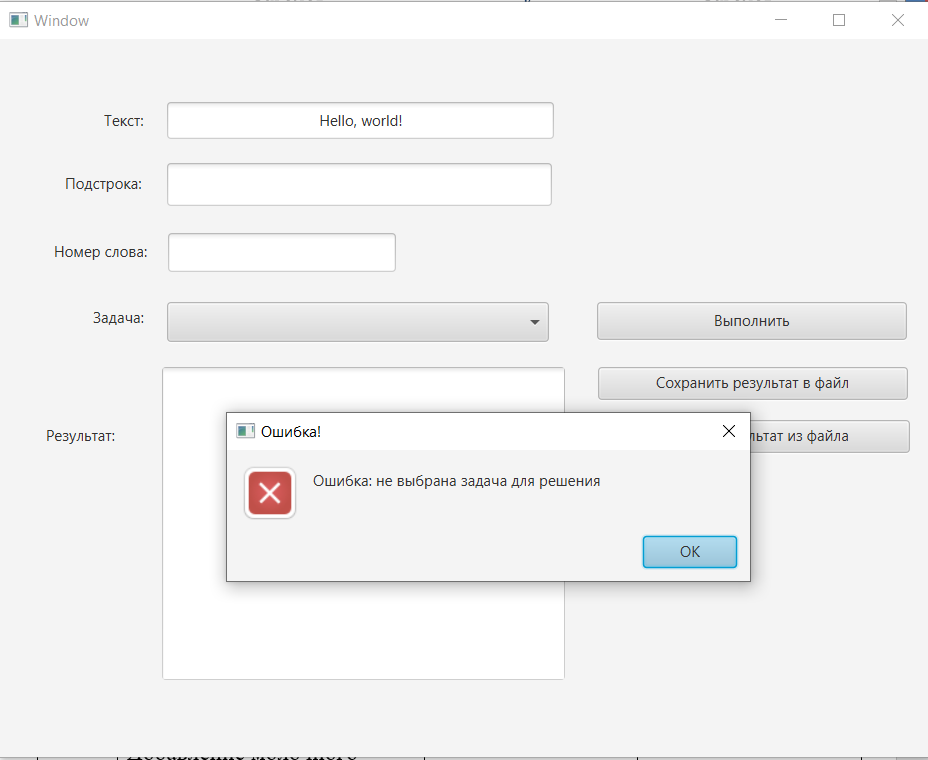


Рисунок 4 – Нажатие кнопки “Выполнить”, когда не выбрана ни одна задача для решения

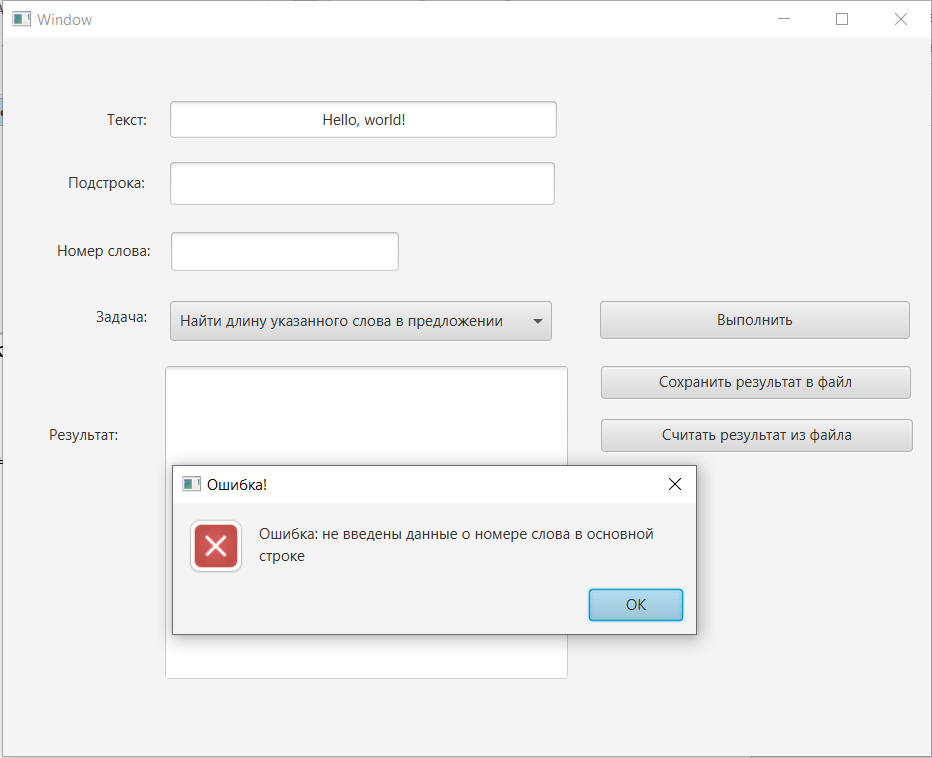


Рисунок 5 – Попытка решения задания, когда входные данные не удовлетворяют требованиям задачи

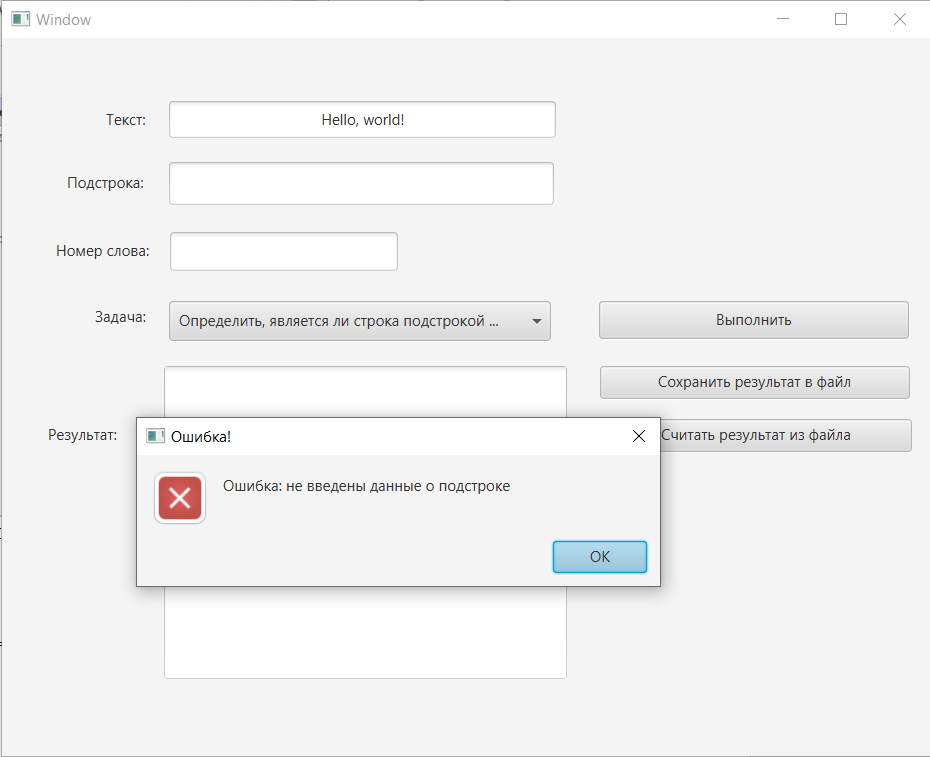


Рисунок 6 - Попытка решения задания, когда входные данные не удовлетворяют требованиям задачи

В случае успешного решения задания, пользователь увидит текст в элементе управления TextArea, перед которым находится элемент управления Label с названием “Результат:”.

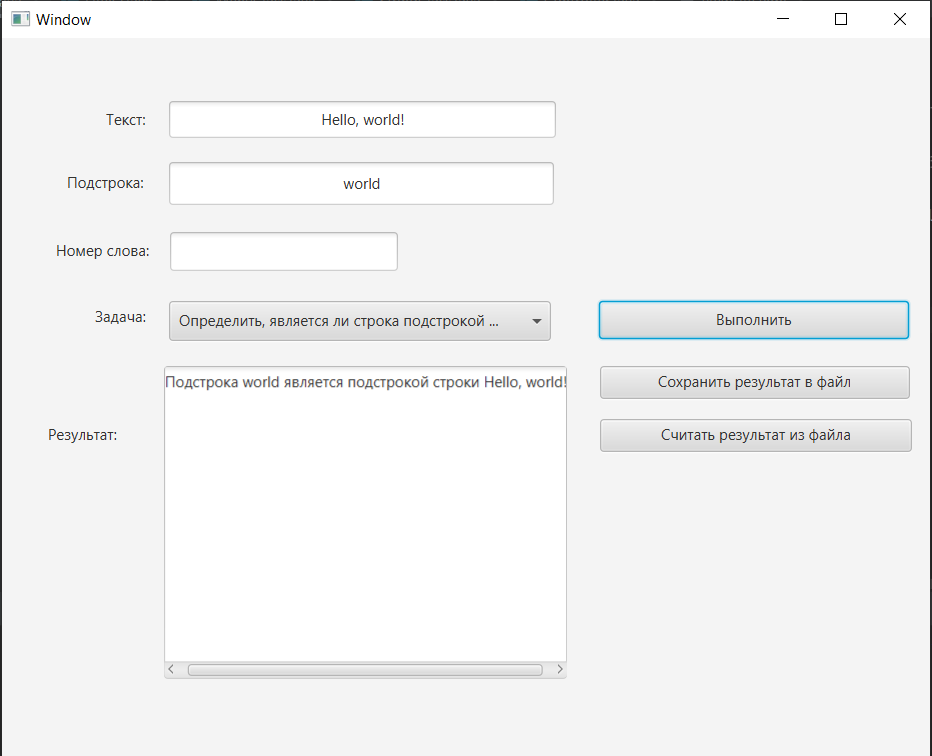


Рисунок 7 – Решение задачи о принадлежности подстроки строке

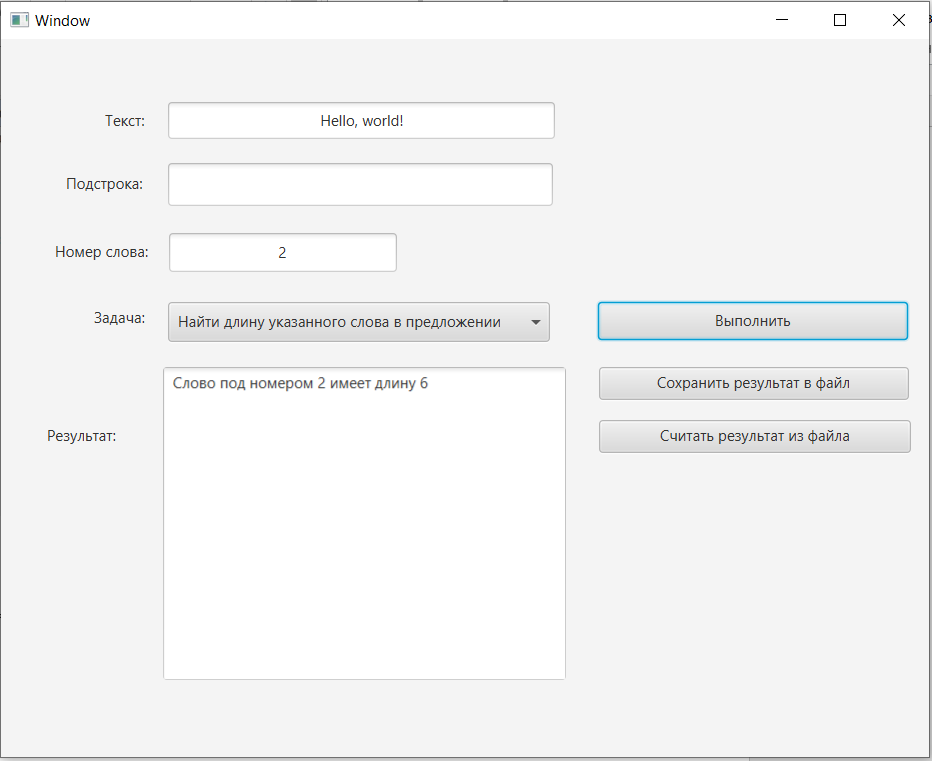


Рисунок 8 – Решение задачи о длине указанного слова в предложении

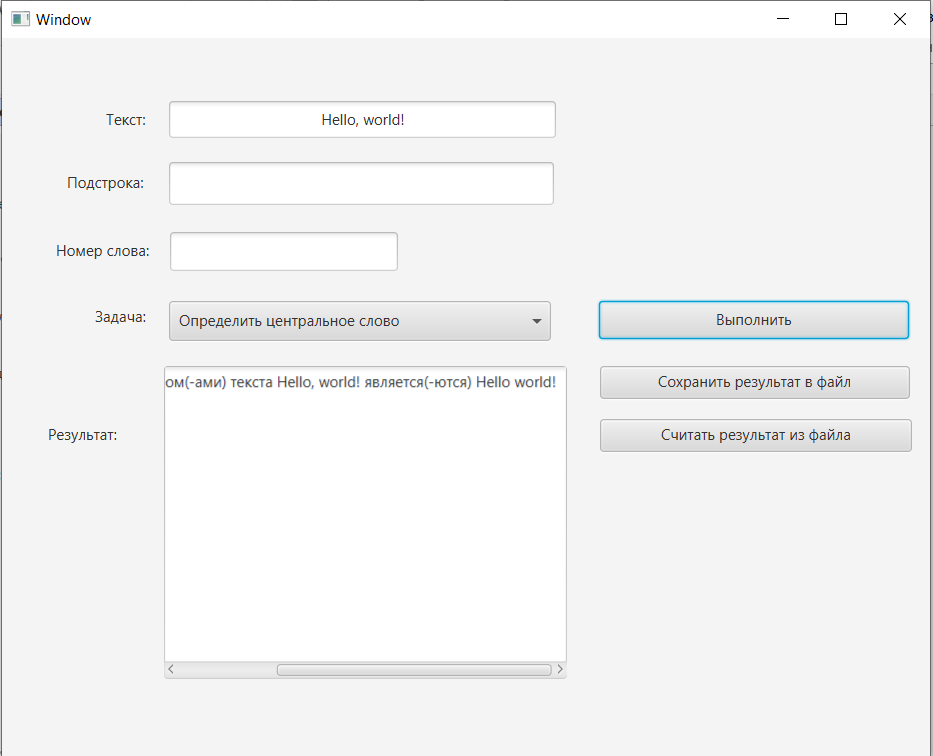


Рисунок 9 – Решение задачи о нахождении центрального слова в предложении

На рисунке 9 видно, что при объемном тексте элемент управления TextArea предоставляет горизонтальный ползунок, с помощью которого можно последовательно рассмотреть текст задачи. Возможно, что данной программе на вход поступит достаточно большая строка, вывод которой будет не возможен в силу технических ограничений, поэтому ощущается необходимость предоставления горизонтального ползунка, для чтения объёмного текста.

Для того, чтобы сохранить данные в результат достаточно нажать на кнопку “Сохранить результат в файл”, и выбрать файл для записи в новом открывшемся окне с названием “Сохранение данных”.

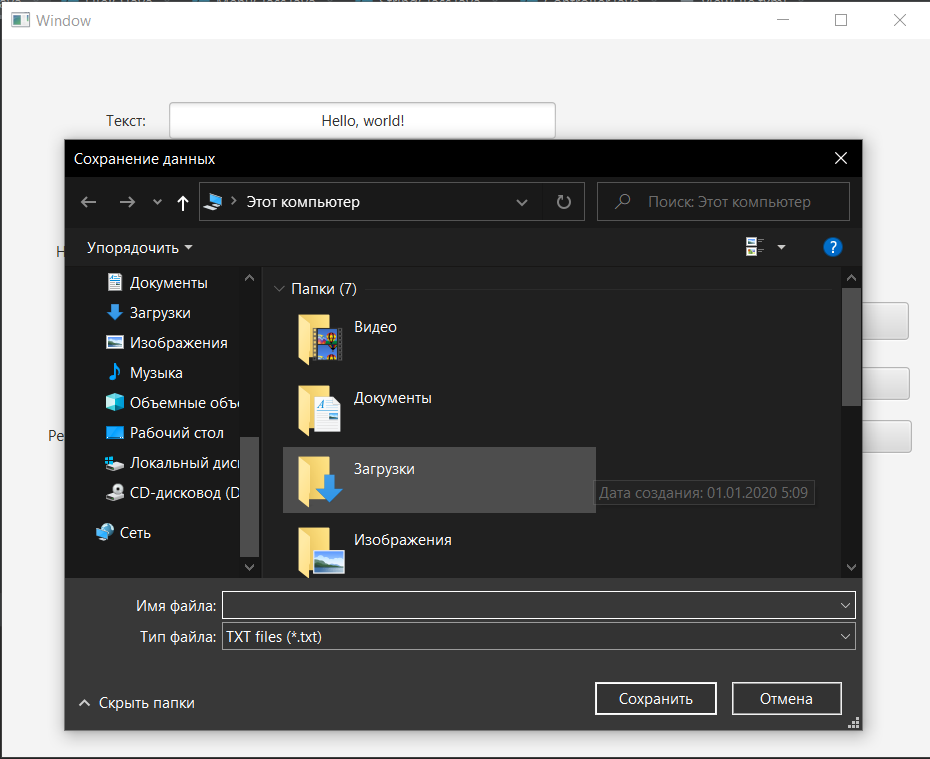


Рисунок 10 – Сохранение данных

После выбора файла и нажатия на кнопку “Сохранить”, программа выведет уведомление о сохранении данных или не сохранении (в случае ошибки).

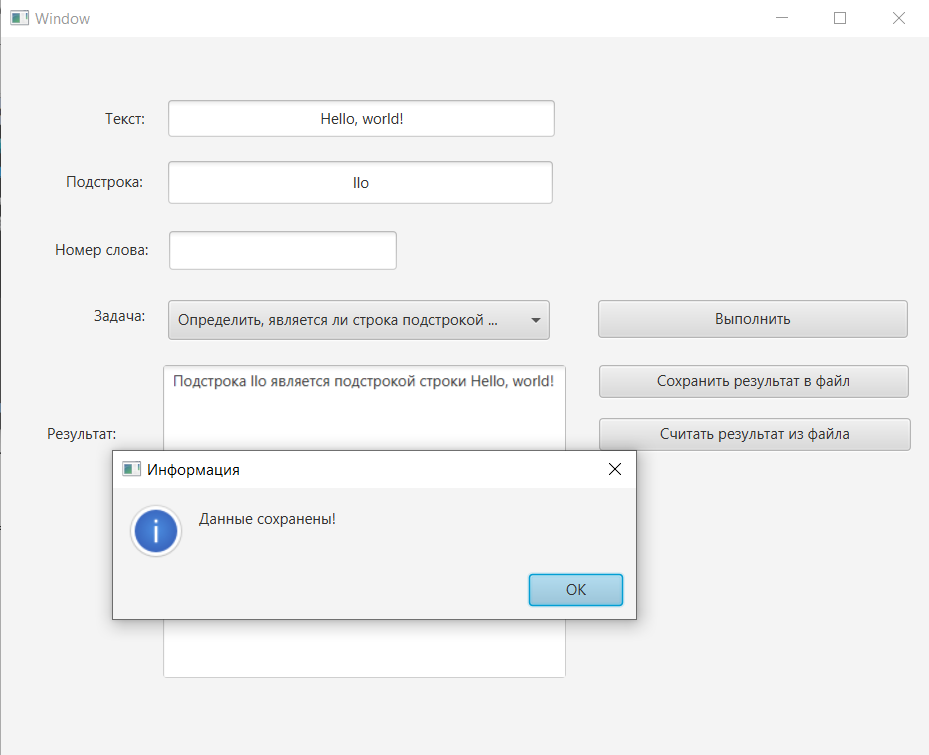


Рисунок 11 – Сохранение данных

Для различных задач данные сохраняются определённым образом. Например, для задачи “Определить, является ли строка подстрокой другой строки” в файле последовательно будет сохранена следующая информация: номер задания (1-3, согласно индивидуальному варианту: 1-a, 2-b, 3-c), основная строка и подстрока. Для задачи “Найти длину указанного слова в предложении” в файле будет последовательно сохранена следующая информация: номер задания, основная строка и номер слова, длину которого необходимо найти. Для задачи “Определить центральное слово” в файле будет находиться только номер задания и основная строка.

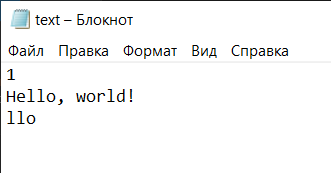


Рисунок 12 – Сохранённые данные задачи “Определить, является ли строка подстрокой другой строки”

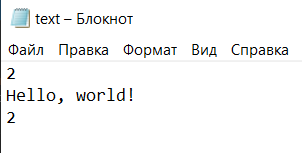


Рисунок 13 – Сохранённые данные задачи “Найти длину указанного слова в предложении”

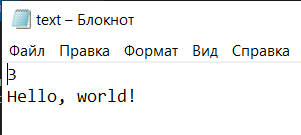


Рисунок 14 – Сохранённые данные задачи “Определить центральное слово”

Для считывания данных из файла, содержащий корректные данные для решения той или иной задачи, необходимо нажать на кнопку “Считать результат из файла” и выбрать определённый файл.

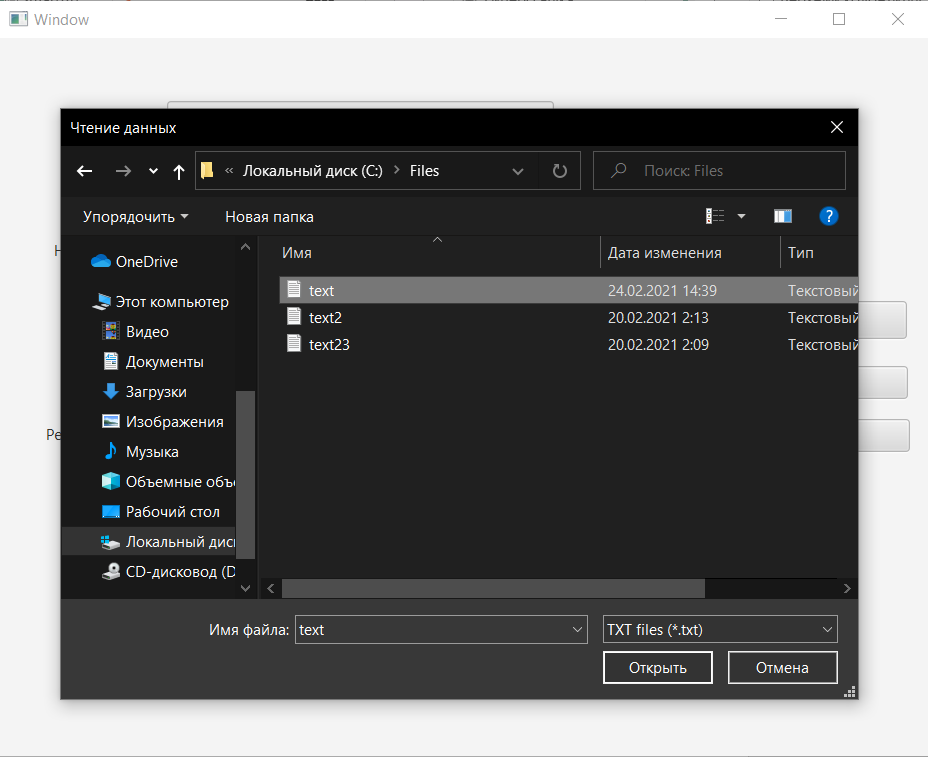


Рисунок 15 – Считывание данных для решения задачи

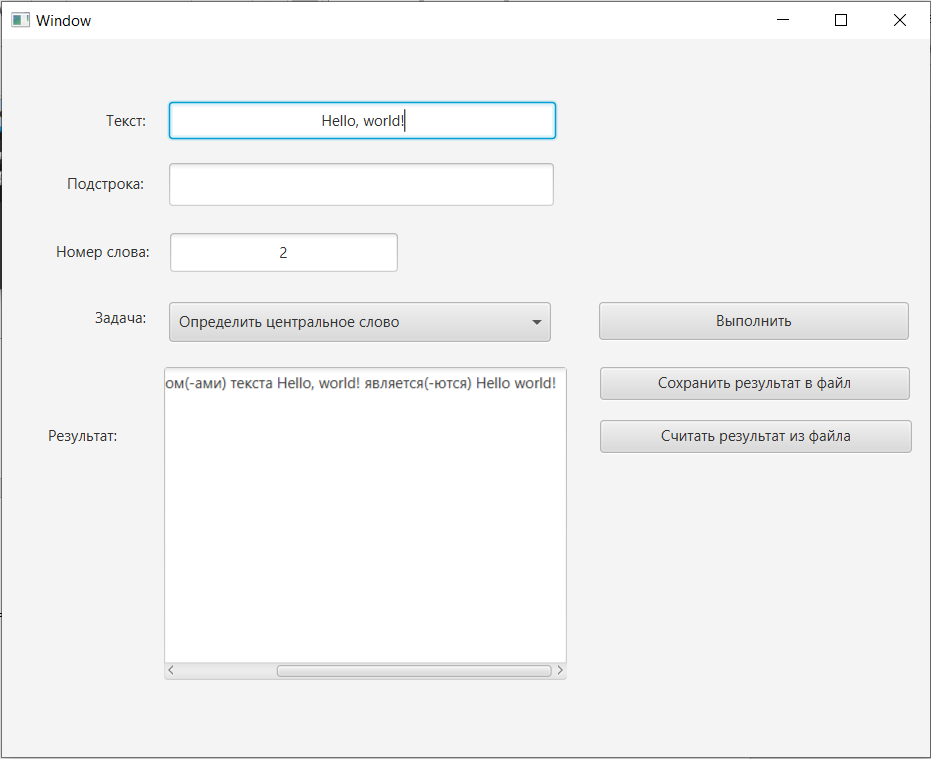


Рисунок 16 – Результат считывания данных из файла

Таким образом, данный интерфейс предоставляет удобный и не сложный функционал относительно решения задач согласно индивидуальному варианту. Интерфейс позволяет выводить информацию о результатах решения в элемент управления TextArea, или сохранять входные данные в файл, а затем считывать их из файла с возможностью решения задач относительно входных данных, полученных после считывания из файла.

4 Проектирование классов

Таблица 1 – Описание методов класса и спецификация локальных переменных

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя** | **Спецификатор** | **Входные данные** | **Назначение** | **Тип** | **Выходные данные** | **Назначение** | **Тип** |
| **Класс StringClass** | | | | | | | | |
| 1 | StringClass | public | - | - | - | - | - | - |
| 2 | StringClass | public | \_text | Строка, для копирование в поле класса text | String | text | Строка, для работы объекта класса | String |
| 3 | StringClass | public | obj | Объект, для копирования | StringClass | text | Строка, для работы объекта класса | String |
| 4 | setText | public | \_text | Строка, для копирование в поле класса text | String | text | Строка, для работы объекта класса | String |

Продолжение таблицы 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | getText | public | - | - | - | text | Строка, для работы объекта класса | String |
| 6 | substringInString | public | substring | Строка для поиска | String | flag | Результат поиска | boolean |
| 7 | lengthDefineWord | public | number | Порядковый номер слова | int | lengthWord | Длина искомого слова | int |
| 8 | centerWords | public | - | - | - | results | Результат поиска | String |
| **Класс FileIO** | | | | | | | | |
| 1 | checkFilePath | private | filePath | Полный путь к файлу | String | check | Результат проверки | boolean |
| 2 | inputDataInFile | public | filePath  data  create | Полный путь к файлу  Данные для записи  Флаг для создания файла | String  String  boolean | - | - | - |
| 3 | outputDataFromFile | public | filePath | Полный путь к файлу | String | data | Результат считывания данных из файла | String |

Продолжение таблицы 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Main** | | | | | | | | |
| 1 | MessageShow | public | type  title  message | Функция, генерирующая сообщения | Alert.AlertType  String  String | Сообщение | Вывод определённого сообщения | - |
| 2 | ComboBoxValidate | public | - | Проверка корректности данных элемента управления ComboBox | - | flag | true – проверка прошла успешно, false – в противном случае | boolean |
| 3 | start | public | primaryStage | Инициализация интерфейса графического приложения | Stage | - | - | - |
| 4 | main | public | args | Точка входа | String[] | - | - | - |

Реализация взаимодействия пользователя с графическим интерфейсом и подключение \*.fxml файла, для инициализации окна элементами управления происходит в классе Main. Данный класс имеет функцию stage(), которая была переопределена при помощи директивы @Override от базового класса Application. В данной функции осуществляется настройка интерфейса, согласно жизненному циклу приложения, написанному на JavaFX.

Этапы жизненного цикла приложения, написанного на JavaFX, представлены методами класса Application, которые можно переопределять.

**Этапы:**

1. init(): инициализирует приложение до его запуска. Метод не должен использоваться для создания графического интерфейса или отдельных его частей.

2. start(Stage stage): здесь определяется графический интерфейс.

3. stop(): вызывается после закрытия приложения.

На рисунке 17 представлена диаграмма классов графического приложения.

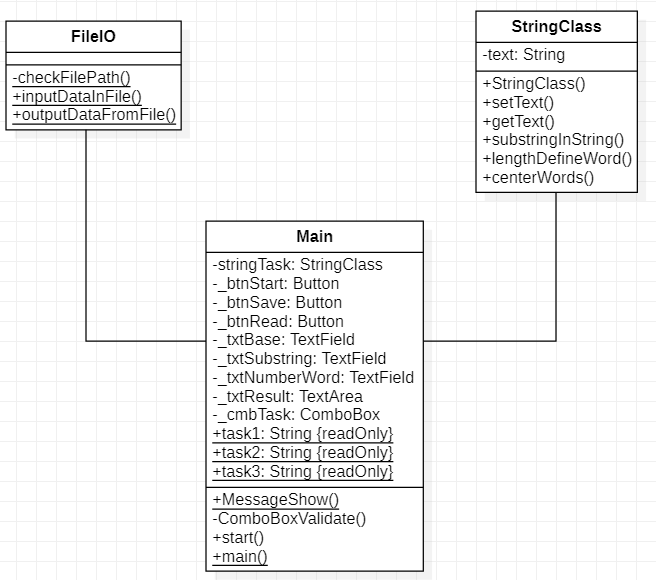


Рисунок 17 – Диаграмма классов графического приложения

**Описание класса FileIO:**

Данный класс содержит в себе три статических функции, лишь две из которых доступны для общего использования. Статическая функция checkFilePath() используется внутри двух других статических функций класса FileIO, для проверки корректности полных путей к файлам. Статическая функция inputDataInFile() записывает данные в файл определённым образом, а функция outputDataFromFile() считывает все данные из файла.

**Описание класса StringClass:**

Данные класс содержит одно закрытое поле типа String, характеризующее данные основного текста – text. Данное поле используется всеми функциями, которые реализованы в данном классе (основной текст используется для решения задач). Функции getText() и setText() – это геттер и сеттер атрибута text. Функция substringInString() определяет, является ли входная строка подстрокой строки text. Функция lengthDefineWord() определяет размер слова в предложении text, по указанному номеру, а функция centerWords() возвращает центральное слово, найденное в атрибуте класса text.

**Описание класса Main:**

В данном классе происходит реализация графического пользовательского интерфейса. Класс является наследником класса Application, который предоставляет возможность запуска графических приложений с помощью библиотеки JavaFX. На рисунке 17 представлена полное описание класса Main. Класс имеет достаточно много атрибутов, таких как кнопки, выпадающий список, строки, поле с типом StringClass, текстовые поля и строковые константы, которые являются статическими (для возможности использования их в статических функциях). Класс имеет 4 функции, две из которых статические. Статическая функция MessageShow() генерирует сообщение и выводит его на экран, согласно заданному набору параметров. Функция ComboBoxValidate() проверяет корректность значения выбранного в выпадающем списке ComboBox. Функция start() настраивает пользовательский интерфейс и описывает реакцию элементов управления на действия пользователей. Функция main() запускает приложение.

5 Таблица тестов

Таблица 2 – Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Решение задания a: строка является подстрокой основной строки | Текст: Hello, world!  Подстрока: Hello | Подстрока Hello является подстрокой строки Hello, world! (рис. 18) |
| 2 | Решение задания a: строка не является подстрокой основной строки | Текст: Hello  Подстрока: ajf8@)9- | Подстрока ajf8@)9- не является подстрокой строки Hello (рис. 19) |
| 3 | Решение задания b | Текст: abcd nbdh, nher, hehrd  Номер слова: 4 | Слово под номером 4 имеет длину 5 (рис. 20) |
| 4 | Решение задания c | Текст: ab dc bf | Центральным словом(-ами) текста ab dc bf является(-ются) dc (рис. 21) |
| 5 | Сохранение данных, для решения задания a | Текст: He Iu Hij H  Подстрока: Iu | (рис. 22-23) |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | Сохранение данных для решения задания b | Текст: Hello, world!  Номер слова: 2 | (рис. 24-25) |
| 7 | Сохранение данных для решения задания c | Текст: Roy, Car, Woler, Woler | Центральным словом(-ами) текста Roy, Car, Woler, Woler является(-ются) Car Woler (рис. 26-27) |
| 8 | Решение задания b: ошибка – не корректное числовое значение | Текст: hu iiiao 9123j af  Номер слова: ha2 | (рис. 28) |
| 9 | Считывание данных для задания a, когда в файле находились данные для задания c | (рис. 29) | (рис. 30) |
| 10 | Считывание данных для задания a, когда в файле находились данные для задания a | (рис. 31) | Подстрока JGo не является подстрокой строки ABC def JGI jo (рис. 32) |
| 11 | Решение задания a: ошибка -не введена подстрока | Текс: ABC def JGI jo  Подстрока: | (рис. 33) |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| 12 | Решение задания a: ошибка – не введена основная строка | Текст:  Подстрока: hell | (рис. 34) |
| 13 | Сохранение данных: отмена сохранения данных задания b | Текст: GUI Console  Номер слова: 2 | (рис. 35) |
| 14 | Считывание данных: отмена считывания данных задания c | - | (рис. 36) |
| 15 | Сохранение данных: ошибка – нет данных для записи (не выбрана задача и поля не заполнены) | - | (рис. 37) |

6 Результаты тестов

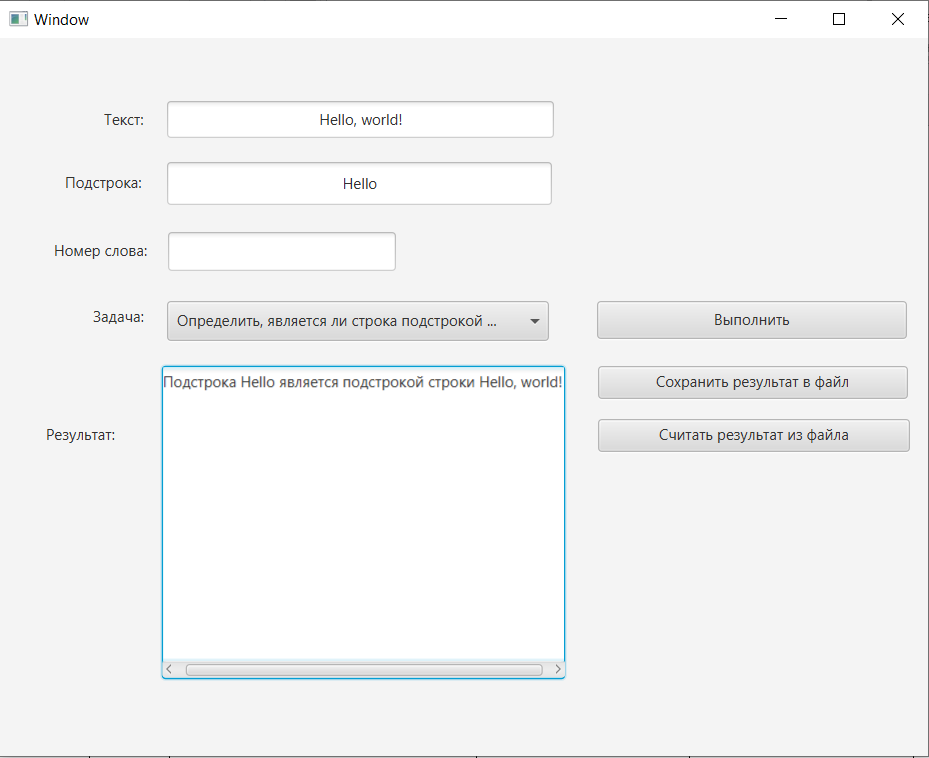


Рисунок 18 – Результат теста №1 из таблицы тестов №2

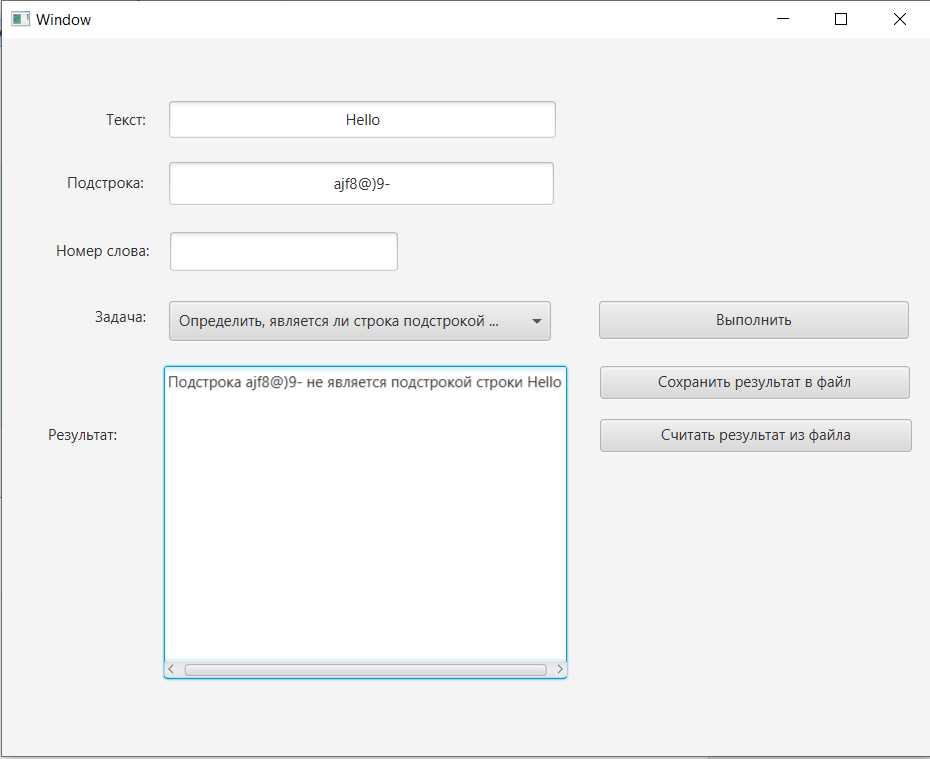


Рисунок 19 – Результат теста №2 из таблицы тестов №2

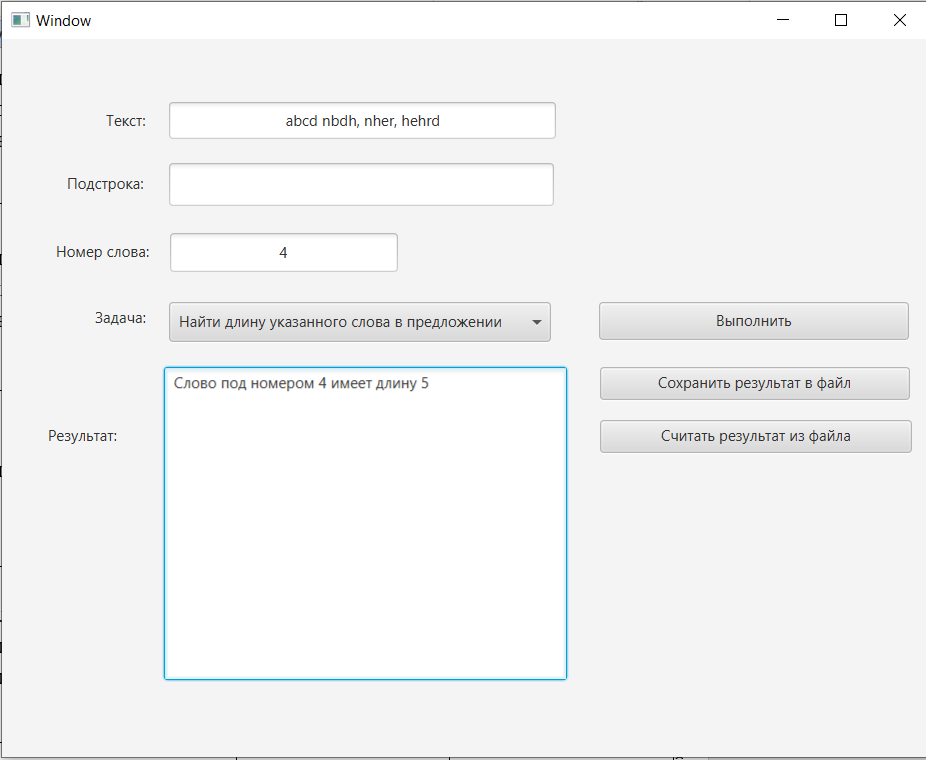


Рисунок 20 – Результат теста №3 из таблицы тестов №2

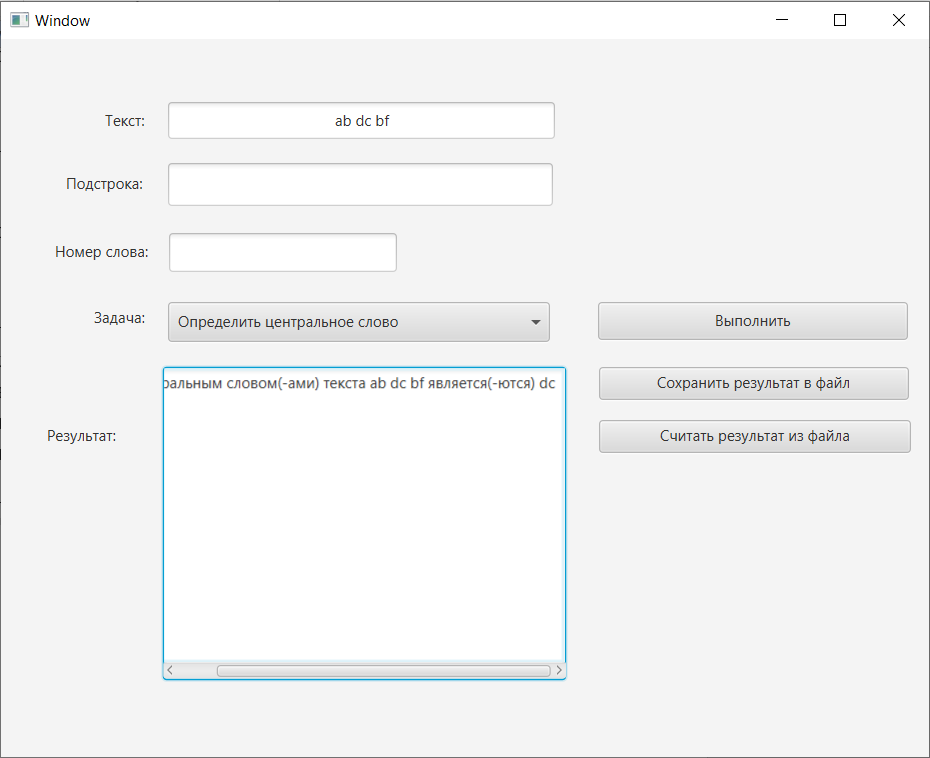


Рисунок 21 – Результат теста №4 из таблицы тестов №2

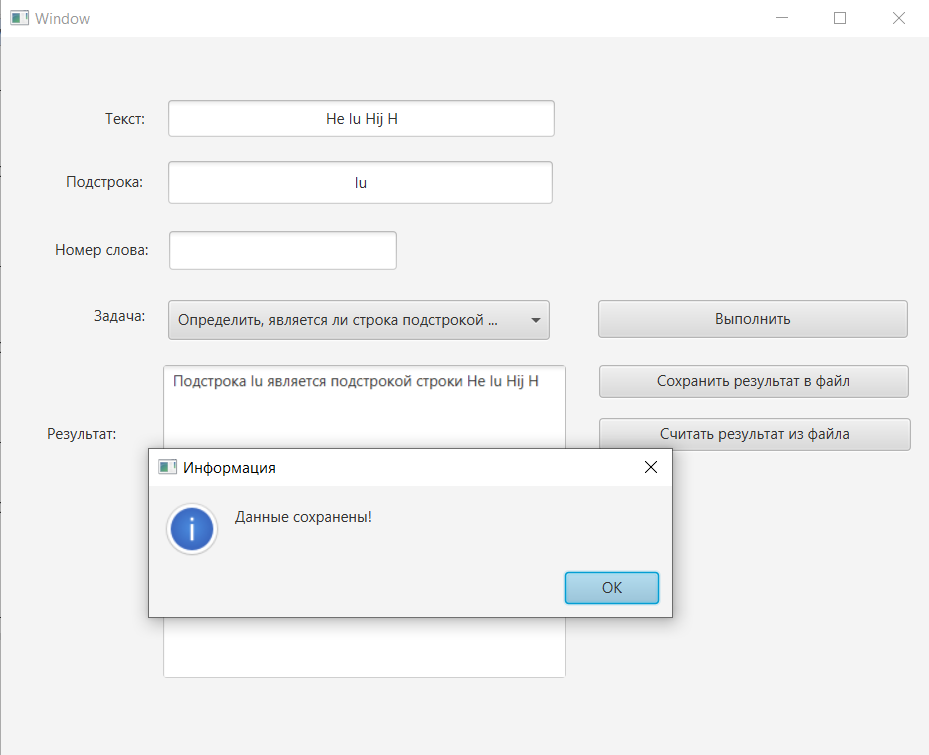


Рисунок 22 – Сохранение данных в файл (тест №5 из таблицы тестов №2)

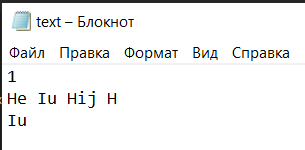


Рисунок 23 – Содержимое файла text.txt после сохранения данных в файл (тест №5 из таблицы тестов №2)

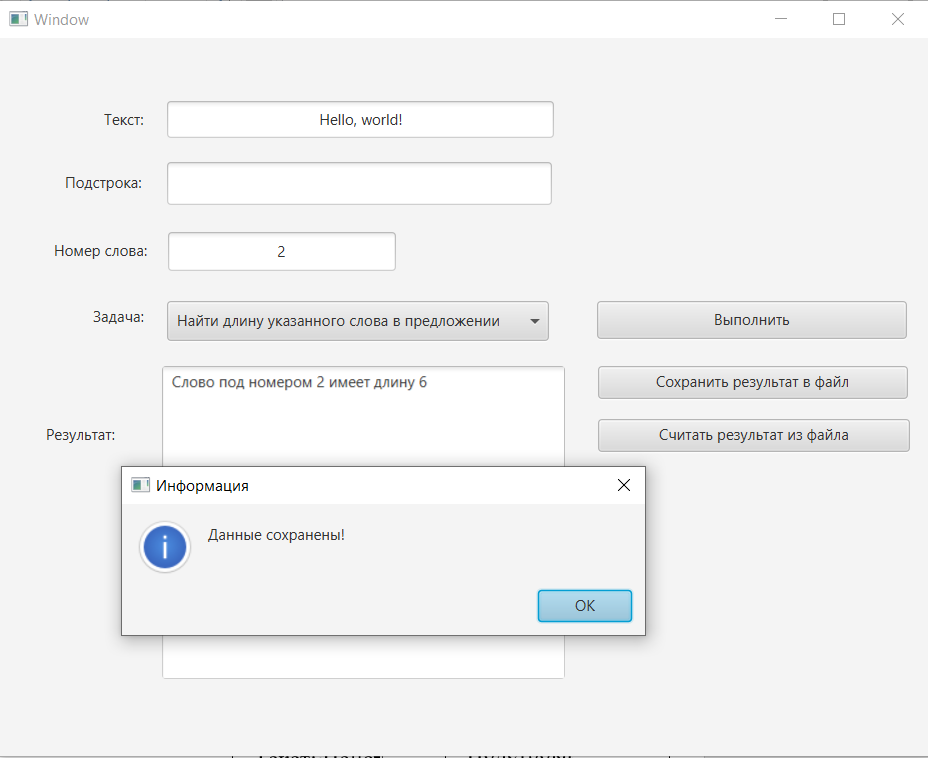


Рисунок 24 - Сохранение данных в файл (тест №6 из таблицы тестов №2)

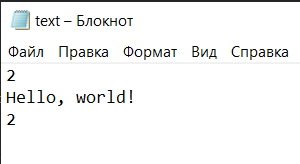


Рисунок 25 - Содержимое файла text.txt после сохранения данных в файл (тест №6 из таблицы тестов №2)

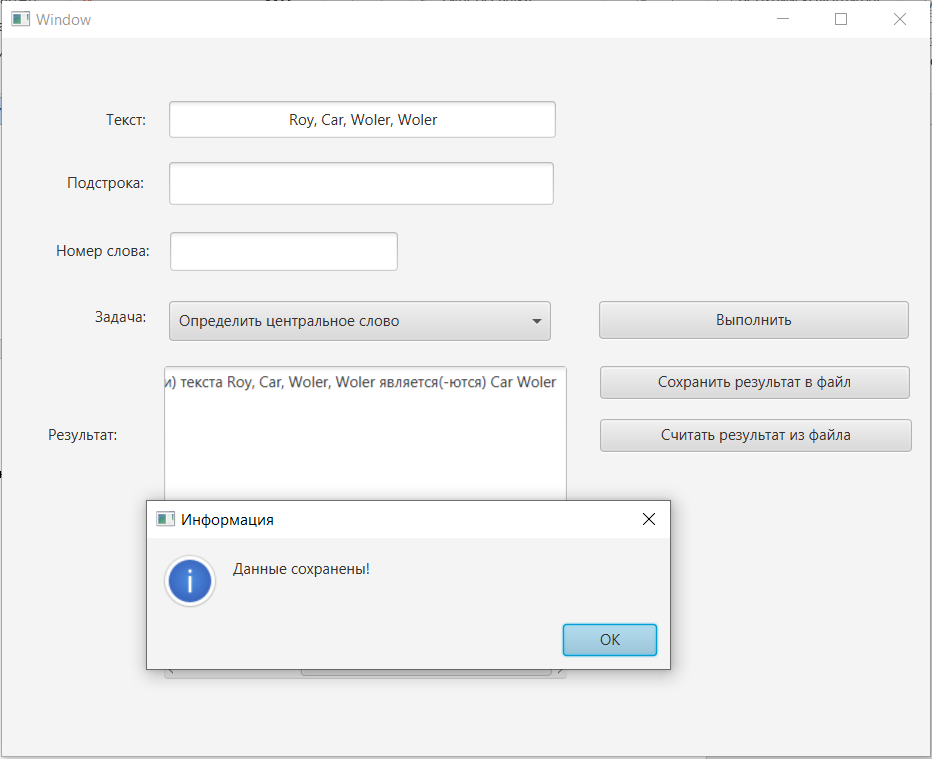


Рисунок 26 - Сохранение данных в файл (тест №7 из таблицы тестов №2)

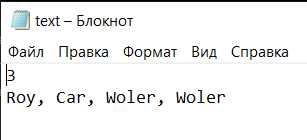


Рисунок 27 - Содержимое файла text.txt после сохранения данных в файл (тест №7 из таблицы тестов №2)

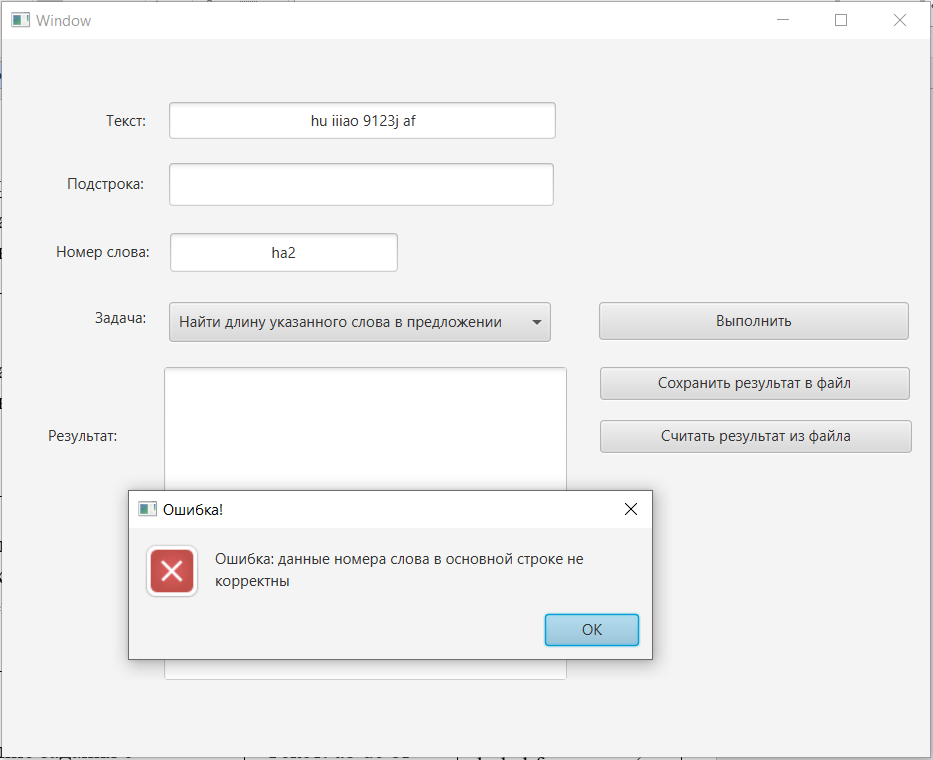


Рисунок 28 – Результат теста №8 из таблицы тестов №2

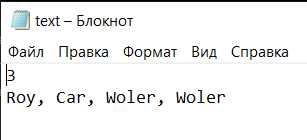


Рисунок 29 – Содержимое файла text.txt, до выполнения считывания данных из файла

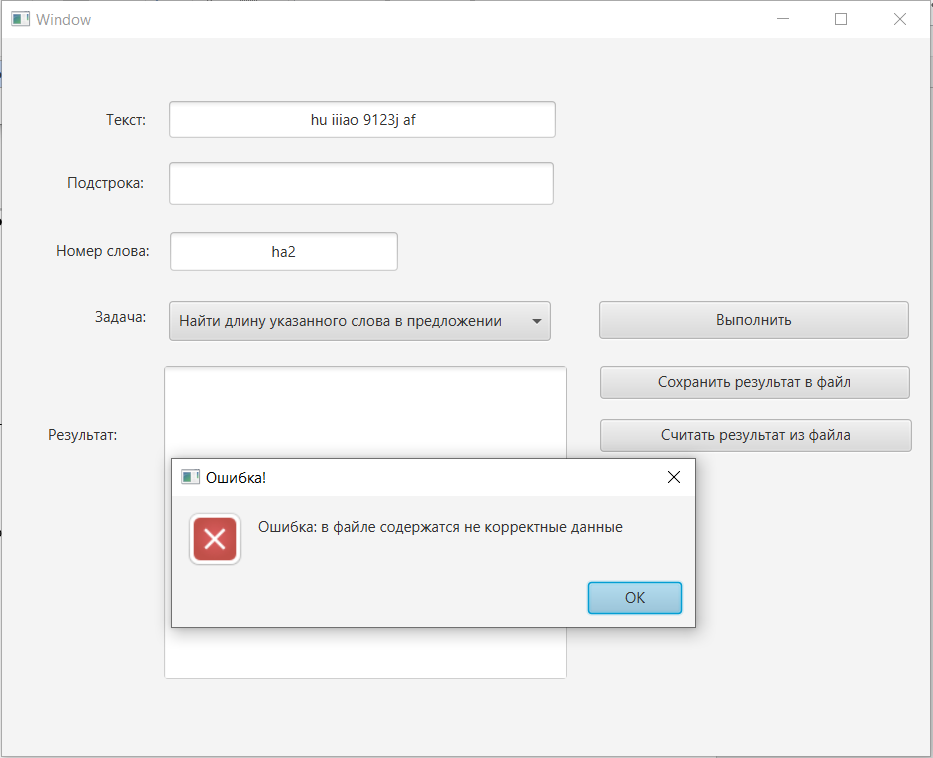


Рисунок 30 – Результат теста №9 из таблицы тестов №2

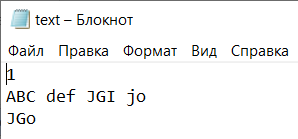


Рисунок 31 - Содержимое файла text.txt, до выполнения считывания данных из файла

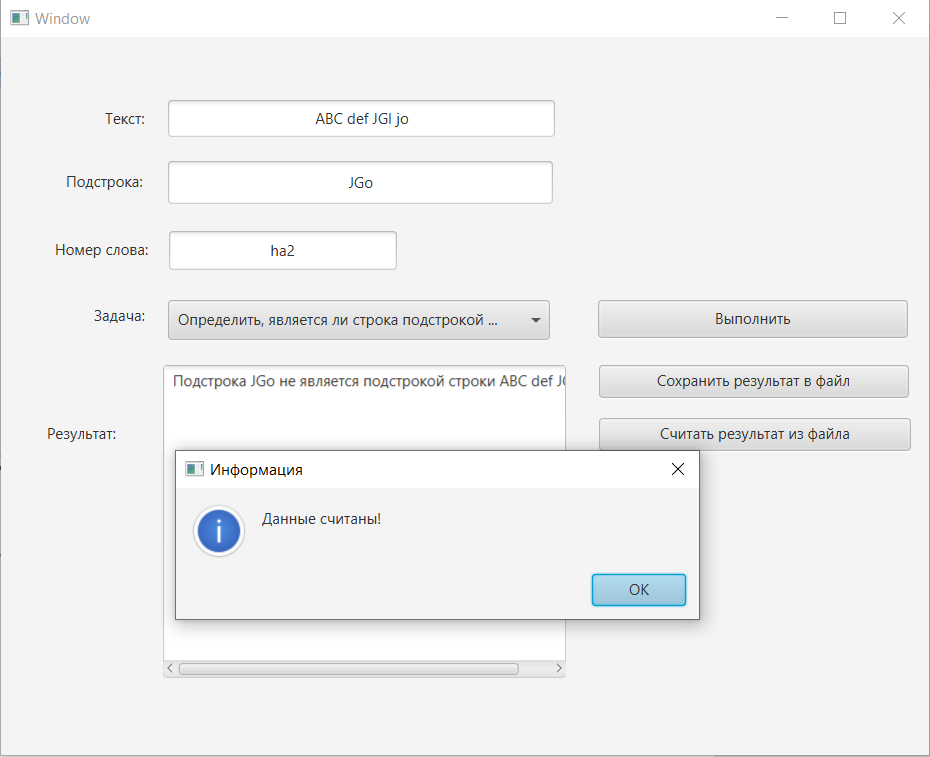


Рисунок 32 – Результат теста №10 из таблицы тестов №2

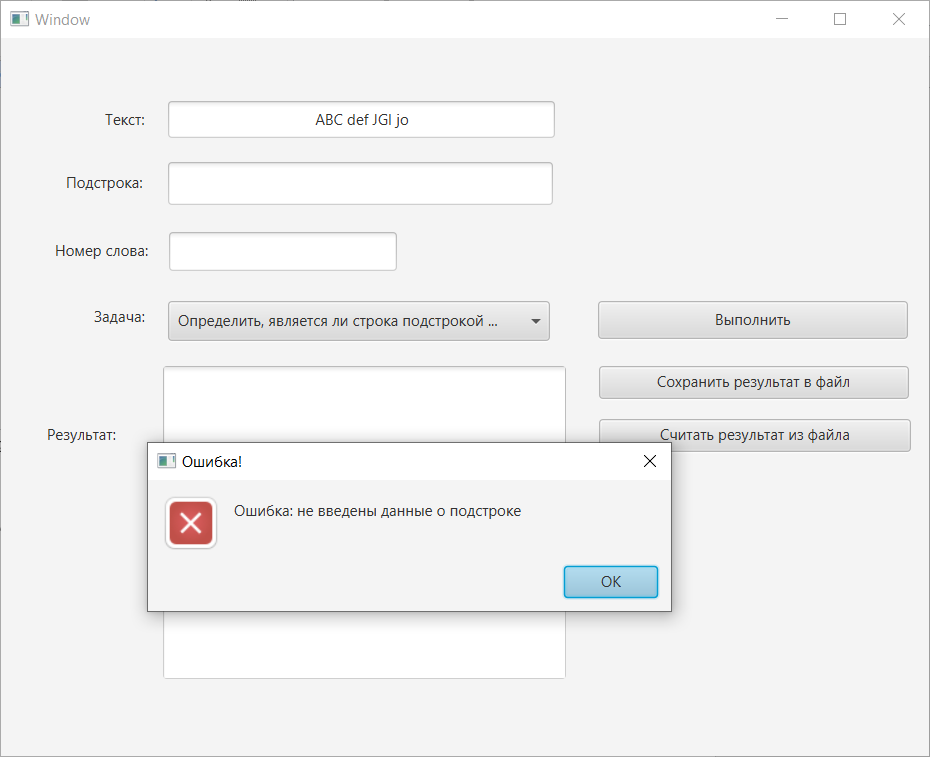


Рисунок 33 – Результат теста №11 из таблицы тестов №2

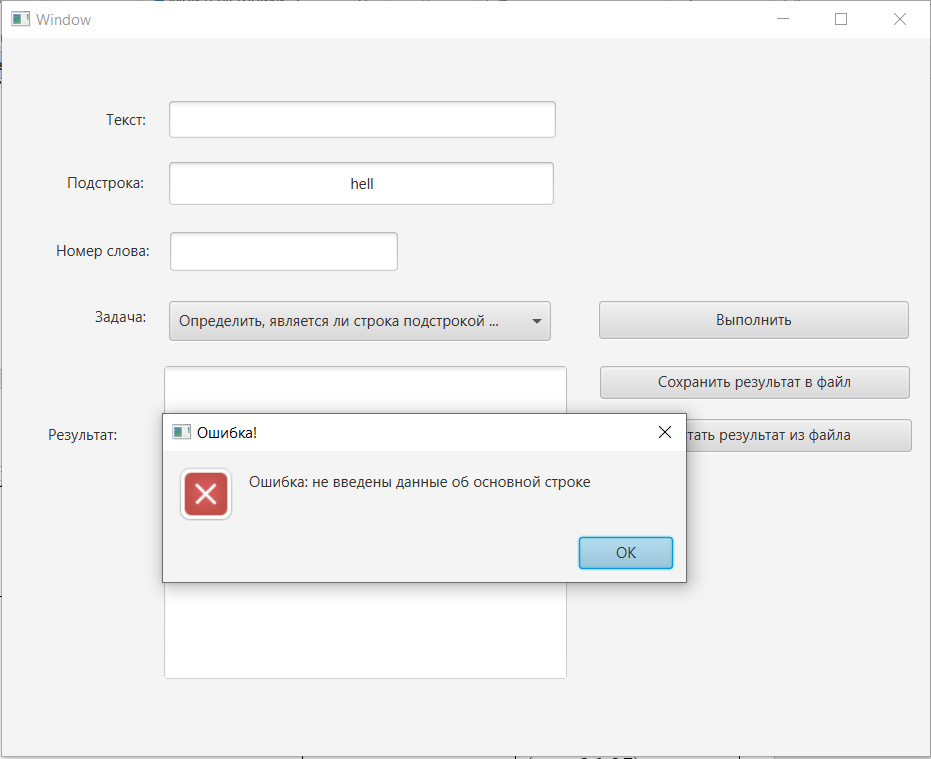


Рисунок 34 – Результат теста №12 из таблицы тестов №2

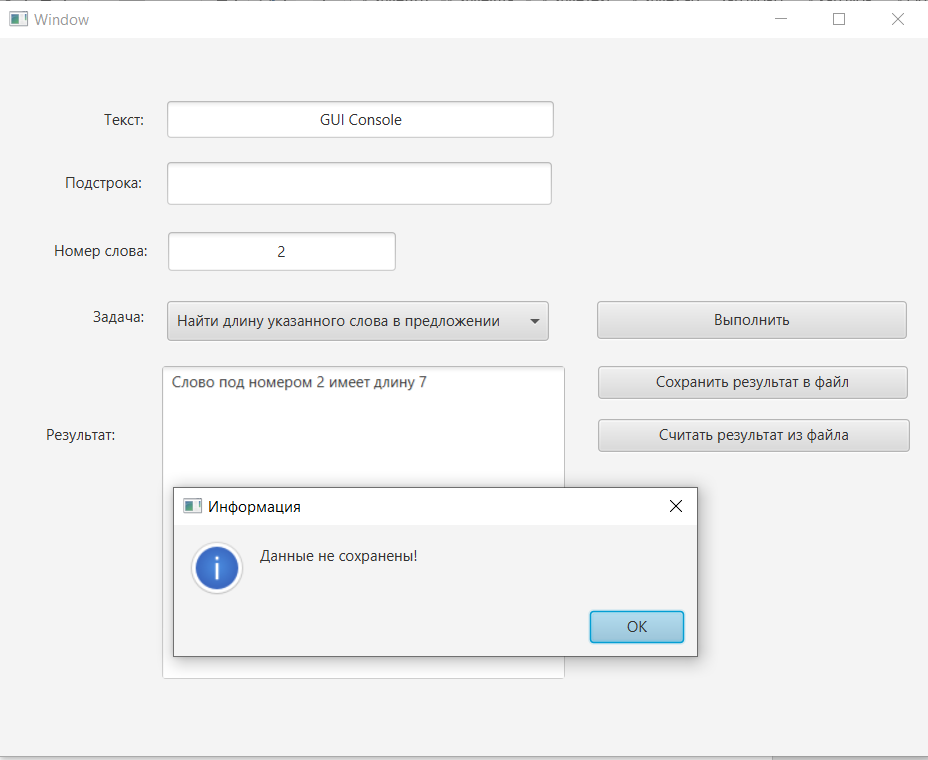


Рисунок 35 – Результат теста №13 из таблицы тестов №2

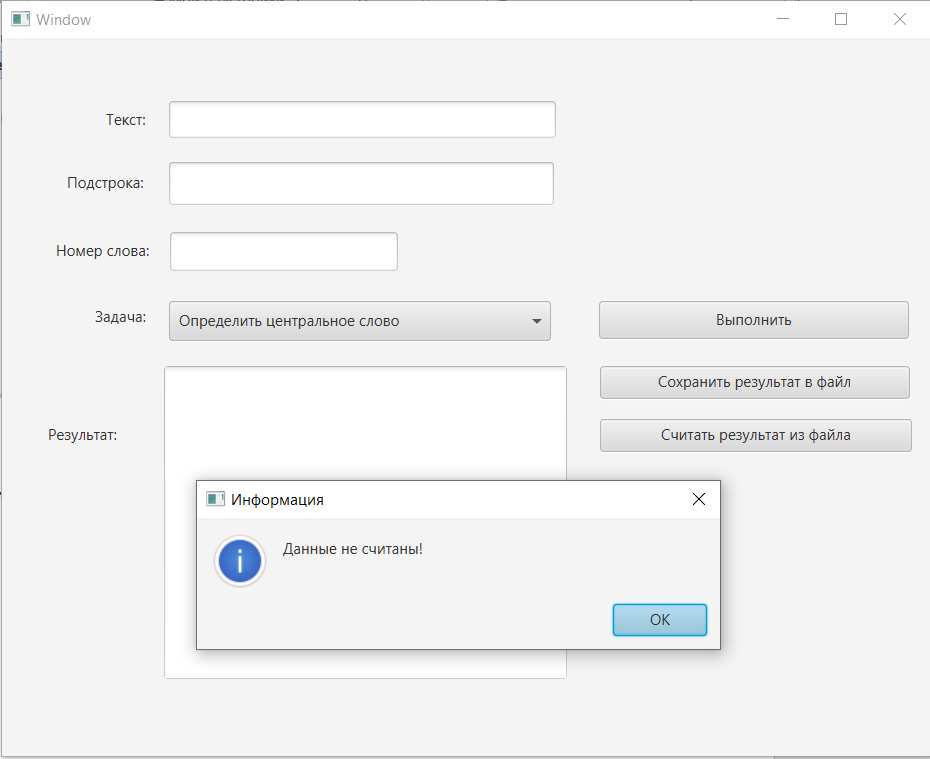


Рисунок 36 – Результат теста №14 из таблицы тестов №2

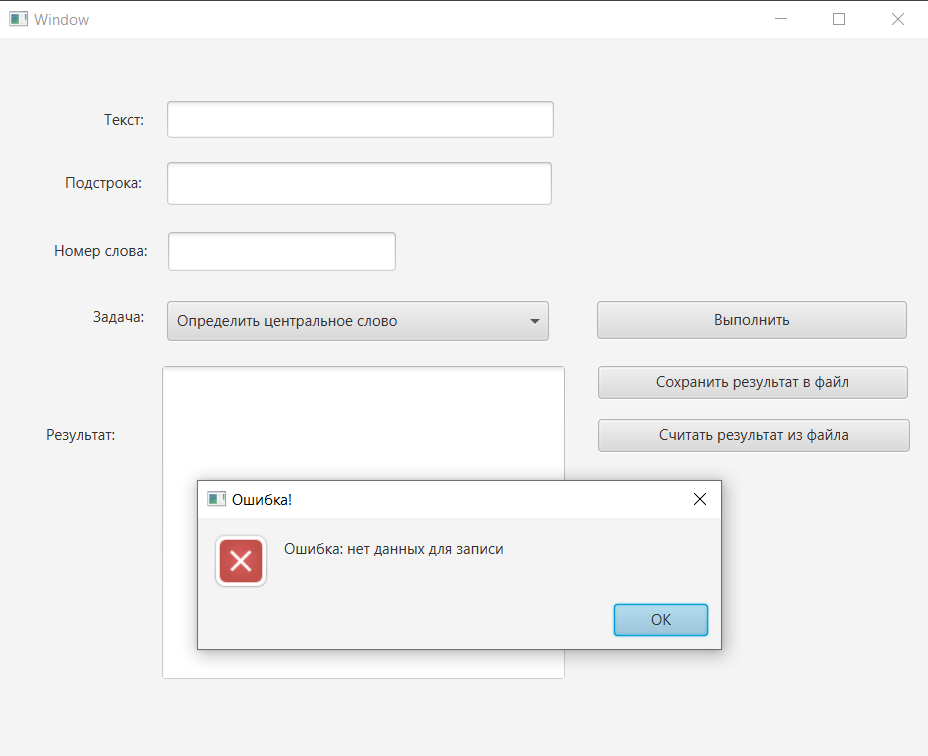


Рисунок 37 – Результат теста №15 из таблицы тестов №2

7 Листинг исходного кода

**Класс FileIO:**

package programGUI;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

public class FileIO {

private static boolean checkFilePath(String filePath){

boolean check = false;

if((filePath.length() == 0)

|| (filePath.charAt(0) == ':')

|| (filePath.indexOf(':') != filePath.lastIndexOf(':'))

|| (filePath.indexOf(':') == (-1))

|| ((filePath.indexOf('\\') != (-1)) && (filePath.indexOf('/') != (-1)))

|| ((filePath.indexOf('\\') == (-1)) && (filePath.indexOf('/') == (-1)))

|| ((filePath.indexOf('\\') != (-1)) && filePath.indexOf('\\') < filePath.indexOf(':'))

|| ((filePath.indexOf('/') != (-1)) && filePath.indexOf('/') < filePath.indexOf(':'))

|| (filePath.indexOf('.') != filePath.lastIndexOf('.'))

)

return check;

char[] symbols = {'|', '?', '\*', '<', '>'};

for(char symbol : symbols)

if(filePath.indexOf(symbol) != (-1))

return check;

check = true;

return check;

}

public static void inputDataInFile(String filePath, String data, boolean create) throws Exception{

if((!checkFilePath(filePath)))

throw new Exception("\nОшибка: не корректный путь к файлу. Дальнейшая работа с записью в файл не возможна." +

" Работа программы будет переведена в штатный режим. \n");

if(create == true){

String[] cellFilePath = filePath.replace('\\', '/').split("\\/");

String directory = "";

for(int i = 0; i < (cellFilePath.length-1); i++)

directory += cellFilePath[i] + "\\";

File file = new File(filePath);

if((!file.exists()) && (!(new File(directory).mkdir())) && (!file.createNewFile()))

throw new Exception("\nОшибка: файла не существует и файл нельзя создать. Дальнейшая работа с записью в файл не возможна." +

" Работа программы будет переведена в штатный режим. \n");

}

try(FileWriter fileWriter = new FileWriter(filePath, false)){

fileWriter.write(data);

}catch(Exception ex){

throw new Exception("\nОшибка: файла не существует. Дальнейшая работа с записью в файл не возможна." +

" Работа программы будет переведена в штатный режим. \n");

}

}

public static String outputDataFromFile(String filePath) throws Exception{

if((!checkFilePath(filePath)))

throw new Exception("\nОшибка: не корректный путь к файлу. Дальнейшая работа с чтением из файла не возможна." +

" Работа программы будет переведена в штатный режим. \n");

String data = "";

try(FileReader fileReader = new FileReader(filePath)){

int c = (-1);

while((c = fileReader.read()) != (-1))

data += (char)c;

}catch(Exception ex){

throw new Exception("\nОшибка: не удалось считать данные из файл. Дальнейшая работа с чтением из файла не возможна." +

" Работа программы будет переведена в штатный режим. \n");

}

return data;

}

}

**Класс StringClass:**

package programGUI;

public class StringClass {

private String text;

public StringClass(){ //Конструктор по умолчанию

text = "";

}

public StringClass(String \_text){ //Конструктор с параметром

\_text = \_text.trim();

text = \_text;

}

public StringClass(StringClass obj){ //Конструктор копирования

text = obj.text;

}

public void setText(String \_text){ //Метод для задания данных

\_text = \_text.trim();

text = \_text;

}

public String getText(){ //Метод для взятия данных

return text;

}

public boolean substringInString(String substring) throws Exception{

/\*

a. Определите, является ли одна строка подстрокой другой.

\*/

substring = substring.trim();

boolean flag = false;

for(int i = 0; (i < text.length()) && (!flag); i++)

if(text.charAt(i) == substring.charAt(0)){

int count = 0;

for(int k = 0; k < substring.length(); k++)

if(substring.charAt(k) == text.charAt(k + i))

count++;

if(count == substring.length())

flag = true;

}

return flag;

}

public int lengthDefineWord(int number) throws Exception{

/\*

b. Найти длину указанного слова в предложении.

\*/

int counter = 1;

int lengthWord = 0;

for(int i = 0; i < text.length(); i++){

if((i >= 1) && ((text.charAt(i-1) == ' ') || (text.charAt(i-1) == ','))

&& ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

counter++;

if((counter == number) && ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

lengthWord++;

}

return lengthWord;

}

public String centerWords () throws Exception{

//c. Выведите из строки, содержащей слова, разделенные пробелами и запятыми, центральное слово

// (если в предложении два централь-ных слова, выведите оба).

String results = "";

int countWords = 1;

for(int i = 1; i < text.length(); i++)

if(((text.charAt(i - 1) == ' ') || (text.charAt(i-1) == ','))

&& ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

countWords++;

int number = (int)(countWords / 2), counter = 1;

if((countWords % 2) != 0)

number++;

for(int i = 0; i < text.length(); i++){

if((i >= 1) && ((text.charAt(i-1) == ' ') || (text.charAt(i-1) == ','))

&& ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

counter++;

if((counter == number) && ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

results += text.charAt(i);

}

if((countWords % 2) == 0){

results += ' ';

counter = 1;

number++;

for(int i = 0; i < text.length(); i++){

if((i >= 1) && ((text.charAt(i-1) == ' ') || (text.charAt(i-1) == ','))

&& ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

counter++;

if((counter == number) && ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

results += text.charAt(i);

}

}

return results;

}

}

**Класс Main:**

package programGUI;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.event.Event;

import javafx.event.EventHandler;

import javafx.event.EventType;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Group;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.FlowPane;

import javafx.scene.layout.StackPane;

import javafx.stage.FileChooser;

import javafx.stage.Stage;

import javafx.stage.WindowEvent;

import javax.swing.event.DocumentEvent;

import java.io.File;

import java.rmi.server.ExportException;

public class Main extends Application {

private StringClass stringTask = null; //экземпляр объекта, решающего задачи

private Button \_btnStart = null; //экземпляр объекта кнопки выполнения задания

private Button \_btnSave = null; //экземпляр объекта кнопки сохранения результата выполнения задания в файл

private Button \_btnRead = null; //экземпляр объекта кнопки чтения результатов выполнения задания из файла

private TextField \_txtBase = null; //текстовое поле (основной текст, относительно которого решается задача

private TextField \_txtSubstring = null; //текстовое поле для подстроки

private TextField \_txtNumberWord = null;//текстовое поле для номера слова в предложении

private TextArea \_txtResult = null; //текстовое поле для вывода результата

private ComboBox \_cmbTask = null; //экземпляр объекта ComboBox, для выбора решаемой задачи

//Задания:

public static final String task1 = "Определить, является ли строка подстрокой другой строки";

public static final String task2 = "Найти длину указанного слова в предложении";

public static final String task3 = "Определить центральное слово";

//Вывод сообщения

public static void MessageShow(Alert.AlertType type, String title, String message){

Alert alert = new Alert(type);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.show();

}

private boolean ComboBoxValidate(){

if((\_cmbTask.getValue() == null) || ((!\_cmbTask.getValue().toString().equals(task1))

&& (!\_cmbTask.getValue().toString().equals(task2))

&& (!\_cmbTask.getValue().toString().equals(task3)))){

return false;

}

return true;

}

@Override

public void start(Stage primaryStage) throws Exception{

Parent root = FXMLLoader.load(getClass().getResource("viewFile.fxml")); //загрузка fxml файла с вёрсткой интерфейса

Scene scene = new Scene(root); //установка для сцены объектов

primaryStage.setScene(scene); //установка для основного контейнера сцены

primaryStage.setTitle("Window"); //установка названия окна

stringTask = new StringClass(); //определение экземпляра объекта, для решения заданий

ComboBox cmbTask = (ComboBox) scene.lookup("#\_cmbTask");

cmbTask.getItems().addAll(task1, task2, task3); //добавления решаемых задач в ComboBox

//нахождение элементов на вёрстке

\_btnStart = (Button) scene.lookup("#\_btnStart");

\_btnSave = (Button) scene.lookup("#\_btnSave");

\_btnRead = (Button) scene.lookup("#\_btnRead");

\_txtSubstring = (TextField) scene.lookup("#\_txtSubstring");

\_txtNumberWord = (TextField) scene.lookup("#\_txtNumberWord");

\_txtBase = (TextField) scene.lookup("#\_txtBase");

\_txtResult = (TextArea) scene.lookup("#\_txtResult");

\_cmbTask = (ComboBox) scene.lookup("#\_cmbTask");

\_txtResult.editableProperty().set(false); //установка режима "только чтение" для элемента TextArea

\_cmbTask.valueProperty().addListener(new ChangeListener<String>() { //обработка события изменения значения в ComboBox

@Override

public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {

\_txtResult.setText("");

}

});

\_btnStart.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if(!ComboBoxValidate()){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не выбрана задача для решения");

return;

}

if(\_txtBase.getText().length() == 0){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не введены данные об основной строке");

return;

}

stringTask.setText(\_txtBase.getText());

if(\_cmbTask.getValue().toString().equals(task1)){

if(\_txtSubstring.getText().length() == 0){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не введены данные о подстроке");

return;

}

try{

\_txtResult.setText("Подстрока " + \_txtSubstring.getText() +

((stringTask.substringInString(\_txtSubstring.getText()))? " является" : " не является")

+ " подстрокой строки " + \_txtBase.getText());

}catch (Exception e){

e.printStackTrace();

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: данная подстрока задана не корректно");

}

}else if(\_cmbTask.getValue().toString().equals(task2)){

if(\_txtNumberWord.getText().length() == 0){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: не введены данные о номере слова в основной строке");

return;

}

try{

\_txtResult.setText("Слово под номером " + \_txtNumberWord.getText() +

" имеет длину " + stringTask.lengthDefineWord(Integer.valueOf(\_txtNumberWord.getText())));

}catch (Exception e){

e.printStackTrace();

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: данные номера слова в основной строке не корректны");

}

}else if(\_cmbTask.getValue().toString().equals(task3)){

try{

\_txtResult.setText("Центральным словом(-ами) текста " + \_txtBase.getText().toString()

+ " является(-ются) " + stringTask.centerWords().toString());

}catch (Exception e){

e.printStackTrace();

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: основная строка не корректна, не возможно определить центральное слово");

}

}

}

});

\_btnRead.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if(!ComboBoxValidate()){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: нет данных для чтения");

return;

}

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Чтение данных");

FileChooser.ExtensionFilter extFilter =

new FileChooser.ExtensionFilter("TXT files (\*.txt)", "\*.txt");

fileChooser.getExtensionFilters().add(extFilter);

try{

File file = fileChooser.showOpenDialog(primaryStage);

if (file != null) {

String dataCmb = \_cmbTask.getValue().toString();

String data = FileIO.outputDataFromFile(file.getAbsolutePath());

if(data.length() == 0){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: данные отсутствуют в файле");

}

String[] strings = data.split("\n");

int number = Integer.valueOf(strings[0]);

if(((dataCmb.equals(task1) || dataCmb.equals(task2)) && (strings.length != 3))

|| ((number > 3) || (number < 1))){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: в файле содержатся не корректные данные");

return;

}

stringTask.setText(strings[1]);

if((number == 1) && (dataCmb.equals(task1))){

\_txtBase.setText(strings[1]);

\_txtSubstring.setText(strings[2]);

\_txtResult.setText("Подстрока " + \_txtSubstring.getText() +

((stringTask.substringInString(\_txtSubstring.getText()))? " является" : " не является")

+ " подстрокой строки " + \_txtBase.getText());

}else if((number == 2) && (dataCmb.equals(task2))){

\_txtBase.setText(strings[1]);

\_txtNumberWord.setText(Integer.valueOf(strings[2]).toString());

\_txtResult.setText("Слово под номером " + \_txtNumberWord.getText() +

" имеет длину " + stringTask.lengthDefineWord(Integer.valueOf(\_txtNumberWord.getText())));

}else if((number == 3) && (dataCmb.equals(task3))){

if(strings.length != 2){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: в файле содержатся не корректные данные");

return;

}

\_txtBase.setText(strings[1]);

\_txtResult.setText("Центральным словом(-ами) текста " + \_txtBase.getText().toString()

+ " является(-ются) " + stringTask.centerWords().toString());

}

else{

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: в файле содержится данные о результатах другой задачи");

stringTask.setText("");

return;

}

MessageShow(Alert.AlertType.INFORMATION, "Информация", "Данные считаны!");

}else{

MessageShow(Alert.AlertType.INFORMATION, "Информация", "Данные не считаны!");

}

}catch (Exception e){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: невозможно открыть данный файл");

e.printStackTrace();

}

}

});

\_btnSave.addEventHandler(MouseEvent.MOUSE\_CLICKED, new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

String dataCmb;

if((\_txtResult.getText().length() == 0) || (!ComboBoxValidate())

|| ((dataCmb = \_cmbTask.getValue().toString()).length() == 0)

|| (\_txtBase.getText().length() == 0)

|| (dataCmb.equals(task1) && \_txtSubstring.getText().length() == 0)

|| (dataCmb.equals(task2) && \_txtNumberWord.getText().length() == 0)){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: нет данных для записи");

return;

}

FileChooser fileChooser = new FileChooser();//Класс работы с диалогом выборки и сохранения

fileChooser.setTitle("Сохранение данных");//Заголовок диалога

FileChooser.ExtensionFilter extFilter =

new FileChooser.ExtensionFilter("TXT files (\*.txt)", "\*.txt");

fileChooser.getExtensionFilters().add(extFilter);

try{

File file = fileChooser.showSaveDialog(primaryStage);

if (file != null) {

String data = "";

if(dataCmb.equals(task1)){

data += "1\n" + \_txtBase.getText() +

"\n" + \_txtSubstring.getText();

}

else if(dataCmb.equals(task2)){

data += "2\n" + \_txtBase.getText() +

"\n" + \_txtNumberWord.getText();

}

else if(dataCmb.equals(task3)){

data += "3\n" + \_txtBase.getText();

}

FileIO.inputDataInFile(file.getAbsolutePath(), data, false);

MessageShow(Alert.AlertType.INFORMATION, "Информация", "Данные сохранены!");

}else{

MessageShow(Alert.AlertType.INFORMATION, "Информация", "Данные не сохранены!");

}

}catch (Exception e){

MessageShow(Alert.AlertType.ERROR, "Ошибка!", "Ошибка: невозможно открыть данный файл");

e.printStackTrace();

}

}

});

primaryStage.show(); //вывод основного окна приложения на экран

}

public static void main(String[] args) {

launch(args);

}

}

**Файл разметки viewFile.fxml:**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ComboBox?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TextArea?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.StackPane?>

<?import javafx.scene.layout.VBox?>

<StackPane fx:id="root" xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="programGUI.Controller">

<children>

<VBox prefHeight="574.0" prefWidth="742.0">

<children>

<HBox prefHeight="50.0" prefWidth="742.0" />

<HBox prefHeight="29.0" prefWidth="742.0">

<children>

<Label alignment="CENTER\_RIGHT" prefHeight="71.0" prefWidth="115.0" text="Текст:" textAlignment="CENTER" />

<HBox prefHeight="30.0" prefWidth="18.0" />

<TextField id="\_txtBase" alignment="CENTER" prefHeight="30.0" prefWidth="309.0" />

</children>

</HBox>

<HBox prefHeight="19.0" prefWidth="742.0" />

<HBox prefHeight="34.0" prefWidth="742.0">

<children>

<Label alignment="CENTER\_RIGHT" prefHeight="47.0" prefWidth="113.0" text="Подстрока:" />

<HBox prefHeight="34.0" prefWidth="20.0" />

<TextField id="\_txtSubstring" alignment="CENTER" prefHeight="34.0" prefWidth="308.0" />

</children>

</HBox>

<HBox prefHeight="21.0" prefWidth="742.0" />

<HBox prefHeight="31.0" prefWidth="742.0">

<children>

<Label alignment="CENTER\_RIGHT" prefHeight="39.0" prefWidth="118.0" text="Номер слова:" />

<HBox prefHeight="31.0" prefWidth="16.0" />

<TextField id="\_txtNumberWord" alignment="CENTER" prefHeight="48.0" prefWidth="182.0" />

</children>

</HBox>

<HBox prefHeight="24.0" prefWidth="742.0" />

<HBox prefHeight="33.0" prefWidth="742.0">

<children>

<Label alignment="CENTER\_RIGHT" prefHeight="26.0" prefWidth="115.0" text="Задача:" />

<HBox prefHeight="26.0" prefWidth="18.0" />

<ComboBox id="\_cmbTask" prefHeight="32.0" prefWidth="305.0" />

<HBox prefHeight="34.0" prefWidth="38.0" />

<Button id="\_btnStart" mnemonicParsing="false" prefHeight="30.0" prefWidth="248.0" text="Выполнить" />

</children>

</HBox>

<HBox prefHeight="18.0" prefWidth="742.0" />

<HBox prefHeight="250.0" prefWidth="742.0">

<children>

<Label alignment="CENTER" prefHeight="110.0" prefWidth="130.0" text="Результат:" />

<TextArea id="\_txtResult" prefHeight="250.0" prefWidth="322.0" />

<HBox prefHeight="250.0" prefWidth="26.0" />

<VBox prefHeight="250.0" prefWidth="265.0">

<children>

<Button id="\_btnSave" mnemonicParsing="false" prefHeight="26.0" prefWidth="248.0" text="Сохранить результат в файл" />

<VBox prefHeight="16.0" prefWidth="266.0" />

<Button id="\_btnRead" mnemonicParsing="false" prefHeight="26.0" prefWidth="249.0" text="Считать результат из файла" />

</children>

</VBox>

</children>

</HBox>

</children>

</VBox>

</children>

</StackPane>

8 Заключение

Были получены общие представления о шаблоне программирования Модель/Вид/Контроллер и его практического применения. Был получен практический опыт работы с базовыми графическими библиотеками java.

Было реализовано графическое приложение с помощью библиотеки JavaFX, отлажено и протестировано. Реализованная программа справляется с заданными задачами и в достаточной мере предоставляет функционал, который может быть использован для решения задач.