ЗАДАНИЕ
на курсовую работу по дисциплине
«Информатика и программирование»

1. Тема работы:

«Разработка программного продукта «Редактор».

2. Срок сдачи работы: « 01 » сентября 2024 г.
3. Цель работы:

Разработка программного продукта «Редактор».

4. Формулировка задания:

Программа должна представлять собой оконное приложение, которое позволяет загружать и просматривать текстовые файлы с возможностью поиска и замены слов и выражений.

В рамках курсовой работы должны быть реализованы:

1) Дружественный графический интерфейс программы. Интуитивно понятное управление.
2) Возможность загрузки текстовых файлов (*.txt).
3) Разработка общего алгоритма программы (создание формы приложения с полем для отображения содержимого загружаемого файла, кнопки поиска/замены).
4) Разработка алгоритма поиска и замены слов и выражений.
5) В случае обнаружения совпадения программа должна выдавать окно с информацией о количестве совпадений.
6) Меню «О программе», содержащее вкладки «Справка», «О разработчике».

5. Содержание отчета:
    1) Титульный лист
    2) Задание на курсовую работу
    3) Оглавление
    4) Введение
    5) Разработка библиотеки классов
        - Диаграмма классов
        - Выбор языка программирования
        - Реализация классов
    6) Тестирование
    7) Заключение
    8) Список использованных источников
    9) Приложение А Диаграмма классов
    10) Приложение Б Листинг программы

Оглавление

Введение

Современные офисные программы закладывались в 70-е годы. Прежде всего, в это время активно разрабатывались текстовые редакторы [2].

Текстовые редакторы совершенствовались, оснащались все новыми возможностями и на определенном этапе получили статус так называемых текстовых процессоров. Термин «текстовый процессор» обычно применяется для обозначения программ, которые позволяют осуществлять более сложную обработку текста, чем текстовые редакторы [2].

Основное назначение текстовых редакторов – создавать текстовые файлы, редактировать тексты, просматривать их на экране, изменять формат текстового документа, распечатывать его на принтере [2].

При выборе текстового редактора решающее значение имеет набор функциональных возможностей, предоставляемых программой [1].

Набираемый на компьютере текст воспроизводится на экране дисплея в рабочем поле редактора. Специальный значок – курсор указывает то место на экране, на которое пользователь в данный момент может указывать воздействие (создавать, изменять символы и т.д.) с помощью редактора [2].

Формат файла определяет способ содержания текста в файле и отражается его оригинальным расширением. Существуют универсальные и оригинальные форматы текстовых файлов. Наиболее распространенным форматом текстового файла является TXT – универсальный формат простого текста, не предусматривает форматирования текста, применяется для создания текстовых документов, которые должны читаться в различных системных средах [2].

В настоящее время для подготовки документов на персональном компьютере используются различные текстовые редакторы. Их развитие идет, с одной стороны, по пути расширения функциональных возможностей и, с другой – по пути обеспечения удобства работы с ними [1].

Целью данной работы является получение навыков самостоятельной разработки программного продукта в соответствии с принципами объектно-ориентированного программирования [3].

Исходя из формулировки задания, в данном случае будет разработано приложение, которое по своим функциональным возможностям можно отнести к простейшему текстовому редактору с ограниченным функционалом (отсутствие возможности изменять формат текстовых данных, распечатки на принтере и т.п.).

1. Разработка библиотеки классов

1.1 Диаграмма классов

Для проектирования объектной модели разрабатываемого приложения использовался язык UML (Unified Modeling Language) – унифицированный язык моделирования. Язык UML включает 13 видов диаграмм, среди которых на первом месте в списке – диаграмма классов [3].

Диаграмма классов (прил. А, рис. 1) – это графическое представление набора элементов, изображенного в виде связного графа вершин (классов) и путей (связей).

Кроме того, в данном случае с целью структурирования файлов приложения и удобства его разработки и сопровождения использовались следующие папки-пакеты (“package”): *model, view* и *controller*.

Данная структура приложения является отражением классического шаблона проектирования «Model-View-Controller» и находит свое отражение в диаграмме классов в виде визуального объединения классов или их групп в рамках того или иного пакета.

1.2 Выбор языка программирования

С целью выбора языка программирования был проведен анализ функциональных требований к разрабатываемому программному продукту.

Исходя из требований задания, к основным функциональным требованиям можно отнести следующие:

1) возможность открытия и сохранения текстовых файлов в формате *.txt;
2) возможность редактирования (создания, изменения, удаления) текста;
3) возможность поиска и замены символов, слов и выражений;
4) дружественный графический интерфейс программы и интуитивно понятное управление; к этому же разделу можно отнести создание формы приложения с полем для отображения содержимого загружаемого файла, кнопки поиска/замены, окно с информацией о количестве совпадений, меню «О программе», вкладки «Справка», «О разработчике».

Рекомендованными в рамках дисциплины языками являются C++, C#, Java или какой-либо другой объектно-ориентированный язык высокого уровня, позволяющий разрабатывать независимые оконные приложения.

Исходя из анализа функциональных требований, выбираемый язык должен обладать следующими возможностями:

- взаимодействие с потоками ввода/вывода (п. 1 функциональных требований);
- редактирование (создание, изменение, удаление) строковых данных (п. 2 и 3 функциональных требований);
- графический интерфейс пользователя – GUI (п. 4 функциональных требований).

Ни одно из представленных требований к используемому языку программирования не является уникальным. По крайней мере любой из основных рекомендуемых языков (C++, C# и Java) в полной мере обладают данными возможностями. Кроме того, указанные языки являются одними из наиболее распространенных, что в перспективе позволяет надеяться на долгосрочную совместимость, меньшую трудоемкость и более легкую поддержку, сопровождение и, при необходимости, доработку программного продукта [4]. По крайней мере – в сравнении с менее распространенными языками.

При выборе из трех основных рекомендуемых языков программирования непосредственно функциональные требования самого программного продукта отходят на второй план (в основном – за счет сравнительной простоты и нетребовательности приложения) и не могут служить основными критериями выбора.

В данном случае для разработки приложения был выбран язык Java по следующим причинам.

1. Полная независимость будущего Java-приложения от операционной системы и оборудования, что позволит выполнить приложение

на любом устройстве, для которого существует соответствующая виртуальная машина [4].

2. Устранение большинства ошибок будет происходить на этапе компиляции, то есть, фактически – на ранних стадиях разработки программы [4].

3. Потенциальное отсутствие ошибок, связанных с распределением памяти (выделением, освобождением, поиском и устранением утечек), за счет использования сборщика мусора (garbage collector) для освобождения незанятой памяти [4].

4. Упрощенной обработкой при возникновении исключительных ситуаций (деление на ноль, попытка открыть несуществующий файл и т.п.) с помощью встроенных объектно-ориентированных средств обработки [4].

1.3 Реализация классов

При написании данного приложения использовались 5 классов, для размещения которых с целью структурирования файлов приложения и удобства его разработки и сопровождения использовались следующие папки-пакеты: *model, view* и *controller* [6].

Класс *Main* (*package main.view*) – основной стартовый класс приложения. Также является основным классом-представителем компонента представления. *Представление* (View) отвечает за отображение данных предметной области (*модели*) пользователю, реагируя на изменения *модели* [6].

Класс *Main* состоит из двух полей и 5 методов.

К полям относятся следующие:

- *tabPanelMain* – является графическим элементом типа *TabPane*; центральный элемент графического интерфейса, представляющий из себя панель вкладок;
- *numTab* – является счетчиком открытых вкладок, использующийся для изменения имени вкладки при создании новой вкладки.

К методам относятся следующие.

1. Основной метод выполнения приложения *main()* – расширяет класс Application.

2. Основной метод, отвечающий за построение визуального интерфейса приложения – *start()*. При этом структура приложения извлекается из файла *Main.fxml*, стиль отображения – из файла *main.css*.

Также в данном методе задаются имя приложения в рамке окна («Блокнот»), иконка приложения в рамке, стартовый размер приложения, стартовое расположение приложения на экране (по центру).

Кроме этого, подключается прослушиватель событий сочетания горячих клавиш для управления приложением с клавиатуры (впоследствии все клавиши будут синхронизированы с кнопками визуального интерфейса).

3. Метод-геттер *getTabPanelMain()* – используется классом GeneralController для управления вкладками приложения.

4. Метод-геттер getNumTab() – используется для получения номера, необходимого для заголовка при создании новой вкладки.

5. Метод-сеттер getNumTab() – используется для изменения (увеличения) номера, необходимого для заголовка при создании новой вкладки.

Классы FmxlController, GeneralController и KeyCode (*package main.controller*) являются классами-представителями компонента контроллера. В данном случае все классы *контроллера* (Controller) участвую в интерпретации действия пользователя, оповещая *модель* о необходимости изменений [6].

Класс *FmxlController* состоит из двух полей и 10 методов.

Все 10 методов по сути являются обработчиками событий при нажатии на кнопки и пункты меню (визуальные компоненты интерфейса приложения, определенные в классе *Main.fxml*).

Каждый из методов в свою очередь вызывает соответствующий метод в классе *GeneralController*, который по факту и является не только основным классом-контроллером, определяющим необходимые действия приложения,

но и классом, синхронизирующим действия горячих клавиш, определенных в классе *Main*, и действия элементов интерфейса файла *Main.fxml*.

При этом два поля *toolBar* и *viewPanelButton* необходимы для передачи в составе метода *viewPanelButton()* в класс *GeneralController*.

Класс *GeneralController* также состоит из двух полей и 10 методов.

Оба поля (*selectionModel* и *currentTab*) являются глобальными переменными класса, которые неоднократно используются в методах класса. Поля представляют из себя модель выбора вкладок на панели (только одна вкладка в текущий момент времени) и собственно – текущую вкладку.

К методам относятся следующие.

1. Метод создания новой вкладки *create()*.
2. Метод открытия текстового файла *.txt *open()*.
3. Метод сохранения текстового файла *.txt *save()*.
4. Метод закрытия текущей вкладки *close()*.
5. Метод выхода из приложения (закрытия) *exit()*.
6. Метод открытия окна поиска *search()*.
7. Метод открытия окна замены *change()*.
8. Метод скрытия/отображения элемента инферфейса панели кнопок *viewPanelButton()*.
9. Метод открытия окна помощи *help()*, структурно реализованного с помощью файла *help.fxml*, стилистически определенного с помощью файла *help.css*.
10. Метод открытия окна *devInfo()*, содержащего сведения о разработчике.

Класс *KeyCode* является enum-классом, являющимся локальной реализацией класса *javafx.scene.input.KeyCode*, и содержащим сведения только о клавишах, использующихся в приложении в качестве горячих.

Класс *TextTabModel* (*package main.model*) – основной класс-представитель компонента модели. Модель (Model) предоставляет данные

предметной области представлению и реагирует на команды контроллера, изменяя свое состояние [6].

Класс *TextTabModel* состоит из 5 полей, конструктора и 8 методов.

Данный класс расширяет стандартный JavaFX класс Tab. Расширение стандартного класса было выполнено таким образом, чтобы каждый документ класса представлял из себя не просто отдельную вкладку, а отдельный текстовый документ с возможностью редактирования, поиска и замены текста.

Для реализации указанных функций были введены 5 переменных:

1) textArea – текстовая область для ввода текста;
2) mainText – основной текст, содержащийся в текстовой области;
3) templateSearch – шаблон (символ, слово или текст) для поиска;
4) templateChange – шаблон (символ, слово или текст) для замены;
5) count – счетчик для подсчета количества совпадений при поиске или количества произведенных замен.

Конструктор TextTabModel(String fileName) принимает текстовое поле *fileName*, которое генерируется в методе *GeneralController.create()* при создании новой вкладки или в методе *GeneralController.open()* при открытии уже существующего текстового файла и добавляется в качестве заголовка в создаваемую конструктором вкладку.

Также в конструкторе уже существующей переменной textArea присваивается значение нового текстового поля, устанавливается размер шрифта, текстовое поле добавляется на создаваемую вкладку, после чего на текстовое поле перемещается фокус курсора.

К методам относятся:

1) getTextArea() – метод-геттер поля textArea (при сохранении, поиске и замене текста);
2) setTextArea() – метод-сеттер поля textArea (при открытии файла);
3) setCount() – метод-сеттер поля count (при поиске и замене);

4) setNullCount() – метод-сеттер поля для приведения к нулевому значению после выполненного поиска или замены;
5) getCount() – метод-геттер (при поиске и замене);
6) setNewMainText() – метод-сеттер для передачи в текстовое поле нового текста после замены;
7) setTemplateSearch() – метод-сеттер для передачи шаблона поиска из поля окна в экземпляр модели;
8) setTemplateChange() – метод-сеттер для передачи шаблона замены из поля окна в экземпляр модели.

2. Тестирование

Тестирование программного обеспечения выполняется для демонстрации работоспособности программы, а также для выявления ошибок и дефектов [5].

При этом необходимо отметить, что большая часть ошибок и дефектов (в основном – синтаксических и семантических) была устранена на этапе отладки программного кода с помощью встроенных средств среды разработки и исходя из опыта написания программного кода.

Непосредственно тестирование состоит из ряда процедур, основной из которых является составление набора тестовых сценариев, образующих тест-сьют.

Тестовые сценарии в данном случае основаны на функциональных требованиях к системе и представляют из себя упрощённый вариант модульного тестирования, сводящийся к тестированию всех методов разработанной библиотеки классов.

При этом для элементов приложения, принадлежащих пакетам *view* и *controller* использовалось ручное тестирование графического интерфейса пользователя (табл. 2.1), а для единственного класса *TextTabModel* пакета *model* проводилось как ручное тестирование, так и автоматическое тестирование с использованием фреймоворка JUnit.

Таблица 2.1 – Чек-лист ручного тестирования графического интерфейса пользователя

	Описание теста	pass / fail
	Основное окно	
1	Открытие основного окна при запуске приложения	pass
2	Расположение основного окна по центру экрана	pass
3	Соответствие иконки системного значка приложения установленному (блокнот с ручкой)	pass
4	Соответствие строки заголовка приложения (Блокнот)	pass
5	Наличие кнопок управления окном (сворачивание, изменение размера, закрытие)	pass
6	Наличие строки меню (Файл, Поиск, Вид, О программе)	pass
7	Наличие панели кнопок управления с соответствующими фоновыми картинками (7 кнопок)	pass
8	Наличие панели закладок в основном окне	pass

9	Наличие стартовой закладки (Новый_1)	pass
10	Наличие текстового поля на стартовой закладке	pass
11	Наличие фокуса курсора на текстовом поле	pass
	Строка меню	
12	Открытие подменю при нажатии на пункт меню Файл (Новый, Открыть…, Сохранить…, Закрыть, разделитель, Выход)	pass
13	Открытие подменю при нажатии на пункт меню Поиск (Найти…, Заменить…)	pass
14	Открытие подменю при нажатии на пункт меню Вид (Панель инструментов – чек бокс с установленной галочкой)	pass
15	Открытие подменю при нажатии на пункт меню О программе (Справка, О разработчике)	pass
16	Добавление новой вкладки со следующим порядковым номером при выборе пункта меню Файл/Новый	pass
17	Появление нативного окна операционной системы Открытие при выборе пункта меню Файл/Открыть…	pass
18	Появление нативного окна операционной системы Сохранение при выборе пункта меню Файл/Сохранить… (при наличии вкладок)	pass
19	Открытие типового окна предупреждения при выборе пункта меню Файл/ Сохранить… (при отсутствии вкладок)	pass
20	Закрытие активной вкладки при выборе пункта меню Файл/Закрыть (при наличии вкладок)	pass
21	Открытие типового окна предупреждения при выборе пункта меню Файл/ Закрыть (при отсутствии вкладок)	pass
22	Завершение работы приложения при выборе пункта меню Файл/Выход	pass
23	Открытие окна приложения Поиск при выборе пункта меню Поиск/Найти…	pass
24	Открытие типового окна предупреждения при выборе пункта меню Файл/ Найти… (при отсутствии вкладок)	pass
25	Открытие окна приложения Замена при выборе пункта меню Поиск/Заменить…	pass
26	Открытие типового окна предупреждения при выборе пункта меню Файл/ Заменить… (при отсутствии вкладок)	pass
27	Исчезновение/появление панели инструментов (кнопок) при изменении состояния чек-бокса пункта меню Вид/Панель инструментов	pass
28	Открытие окна Справка при выборе пункта меню О программе/Справка	pass
29	Открытие окна Сведения о разработчике при выборе пункта меню О программе/О Разработчике	pass
	Панель инструментов	
30	Добавление новой вкладки со следующим порядковым номером при нажатии на первую кнопку (документ с плюсом)	pass
31	Появление нативного окна операционной системы Открытие при нажатии на вторую кнопку (папка с документом)	pass
32	Появление нативного окна операционной системы Сохранение при нажатии на третью кнопку (дискета)	pass
33	Закрытие активной вкладки нажатии на четвертую кнопку (белый крест на фоне красного круга)	pass
	Открытие окна приложения Поиск нажатии на пятую кнопку (лупа на	pass

	фоне документа)	
34	Открытие окна приложения Замена нажатии на шестую кнопку (две стрелочки на зеленом фоне)	pass
35	Открытие окна Справка нажатии на седьмую кнопку (знак вопроса на синем фоне)	pass
	Горячие клавиши	
36	Добавление новой вкладки со следующим порядковым номером при нажатии на комбинацию клавиш Alt+N	pass
37	Появление нативного окна операционной системы Открытие при нажатии на комбинацию клавиш Alt+O	pass
38	Появление нативного окна операционной системы Сохранение при нажатии на комбинацию клавиш Alt+S	pass
39	Закрытие активной вкладки нажатии на комбинацию клавиш Alt+W	pass
40	Открытие окна Поиск нажатии на комбинацию клавиш Alt+F	pass
41	Открытие окна Замена нажатии на комбинацию клавиш Alt+J	pass
42	Открытие окна Справка нажатии на клавишу F1	pass
	Работа приложения с файлами	
43	Открытие в новой вкладке текстового файла *.txt с передачей в качестве имени вкладки имени файла	pass
44	Сохранение текста текущей вкладки в файл *.txt с изменением имени вкладки на имя файла	pass
	Вкладки их текстовые поля	
45	Переключение между вкладками кликом мышки на заголовок вкладки	pass
46	Закрытие вкладок кликом мышки на крестик рядом с именем вкладки	pass
47	Отображение введенного ранее текста на каждой вкладке при переключении между вкладками	pass
48	Отображение вертикальной и горизонтальной полос прокрутки при превышении размерами текста размеров окна	pass
	Окно поиска	
49	Расположение окна поиска по центру экрана	pass
50	Соответствие иконки системного значка приложения установленному (блокнот с ручкой)	pass
51	Соответствие строки заголовка окна (Поиск)	pass
52	Наличие стандартной кнопки закрытия окна	pass
53	Наличие поля ввода шаблона поиска	pass
54	Наличие кнопок управления Найти и Отмена	pass
55	Открытие типового окна информации при нажатии кнопки Найти с выводом количества найденных совпадений	pass
56	Открытие типового окна информации при нажатии клавиши Enter с выводом количества найденных совпадений	pass
57	Закрытие окна Поиск при нажатии кнопки Отмена	pass
58	Закрытие окна Поиск при нажатии клавиши Esc	pass
	Окно замены	
59	Расположение окна замены по центру экрана	pass
60	Соответствие иконки системного значка приложения установленному (блокнот с ручкой)	pass
61	Соответствие строки заголовка окна (Замена)	pass
62	Наличие стандартной кнопки закрытия окна	pass
63	Наличие поля ввода шаблона поиска	pass
64	Наличие поля ввода шаблона замены	pass

65	Наличие кнопок управления Заменить и Отмена	pass
65	Открытие типового окна информации при нажатии кнопки Заменить с выводом количества произведенных замен	pass
66	Открытие типового окна информации при нажатии клавиши Enter с выводом количества произведенных замен	pass
67	Закрытие окна Замена при нажатии кнопки Отмена	pass
68	Закрытие окна Замена при нажатии клавиши Esc	pass
	Окно справки	
69	Расположение окна справки по центру экрана	pass
70	Соответствие иконки системного значка приложения установленному (блокнот с ручкой)	pass
71	Соответствие строки заголовка окна (Справка)	pass
72	Наличие стандартной кнопки закрытия окна	pass
73	Наличие справочной информации (наименование пункта меню, горячих клавиш, иконки кнопок, описание функций)	pass
74	Закрытие окна Справка при нажатии клавиши Esc	pass
	Окно сведений о разработчике	
75	Расположение окна сведений о разработчике по центру экрана	pass
76	Соответствие иконки системного значка приложения установленному (блокнот с ручкой)	pass
77	Соответствие строки заголовка окна (Сведения о разработчике)	pass
78	Наличие стандартной кнопки закрытия окна	pass
79	Наличие кнопки управления ОК	pass
80	Наличие сведений о разработчике	pass
81	Закрытие окна при нажатии клавиши ОК	pass
82	Закрытие окна при нажатии клавиши Esc	pass

Данный чек-лист является результирующим и отражает отсутствие ошибок в тестируемых параметрах. При этом на предыдущих этапах были выявлены и устранены несоответствия в пунктах 50, 60, 61 и 76. Ошибки были связаны с отсутствием необходимых строк кода.

Таблица 2.2 – Тест-кейс класса *TextTabModel* с использованием фреймоворка JUnit

ID	Входные данные	Ожидаемый результат выполнения теста	Результат выполнения теста	Вывод
1	**Получение данных из текстовой области текущей вкладки**			
	getTextAreaTest	getTextAreaTest	getTextAreaTest	pass
2	**Передача данных в текстовую область текущей вкладки**			
	setTextAreaTest	setTextAreaTest	setTextAreaTest	pass
3	**Передача данных в счетчик количества совпадений / замен**			
	88888	5	5	pass
4	**Установка нулевого значения в счетчик количества совпадений / замен**			
	0	0	0	pass
5	**Получение данных из счетчика количества совпадений / замен**			
	0	0	0	pass
6	**Передача в текстовую область обновленных данных после выполнения замены**			

	Change, Search	Change	Change	pass
7	**Получение данных шаблона поиска, введенных пользователем**			
	setTemplateSearch	setTemplateSearch	setTemplateSearch	pass
8	**Получение данных шаблона замены, введенных пользователем**			
	setTemplateChange	setTemplateChange	setTemplateChange	pass

Данный тест-сьют также отражает отсутствие ошибок в тестируемых параметрах.

Заключение

В результате выполненной работы в полном объеме достигнута главная цель работы – получены навыки самостоятельной разработки программного продукта в соответствии с принципами объектно-ориентированного программирования на примере реализации простейшего текстового редактора.

Также в ходе реализации основной цели был выполнен ряд вспомогательных промежуточных задач:

1) освоены современные инструментальные среды, используемые в разработке объектно-ориентированных программ (IntelliJ IDEA, Visual Paradigm);
2) построена объектная модель предметной области;
3) разработана диаграмма классов и интерфейсов;
4) выполнена собственно программная реализация;
5) проведено тестирование и отладка программы;
6) выполнены подготовка и написание отчета.

Разработанный в ходе выполнения работы программный продукт полностью работоспособен и может использоваться в качестве текстового редактора при выполнении простейших заявленных функций – создания, открытия, редактирования и сохранения текстовых файлов в формате *.txt.

При этом с учетом специфики сферы использования программного продукта в настоящее время пользователям доступны как большое количество относительно простых и свободно распространяемых текстовых редакторов (некоторые из которых предустанавливаются вместе с операционными системами), так и значительное количество более технически сложных и функциональных текстовых процессоров.

Исходя из изложенного, разработанный программный продукт не имеет самостоятельной экономической, научной и социальной значимости.

Список использованных источников

1. Беляева Т.М. и др. Информатика и математика: учебник и практикум для вузов. Москва, Издательство Юрайт, 2024. 402 с.
2. Коваленко Т.А., Сирант О.В., Знаткова Г.Ю. «Информатика» Часть I: учебное пособие. Самара, ПГУТИ, 2018. 118 с.
3. Морозова Ю.В. Информатика и программирование: методические указания к выполнению курсовой работы для студентов заочной формы обучения с применением ДОТ (уровень бакалавриата). Томск: ФДО ТУСУР, 2018. 30 с.
4. Морозова Ю.В. Объектно-ориентированный анализ и программирование: учебное пособие. Томск: Эль Контент, 2018. 140 с.
5. Морозова Ю.В. Тестирование и контроль качества программного обеспечения: учебное пособие. Томск: ФДО, ТУСУР, 2023. 303 с.
6. Обобщенный Model-View-Controller. Каркас на основе шаблона проектирования MVC в исполнении Generic Java и C# [Электронный ресурс] // Сайт: rsdn.org. – Режим доступа: http:// rsdn.org/article/patterns/generic-mvc.xml (дата обращения: 24.08.2024).

Приложение А

(обязательное)

Диаграмма классов

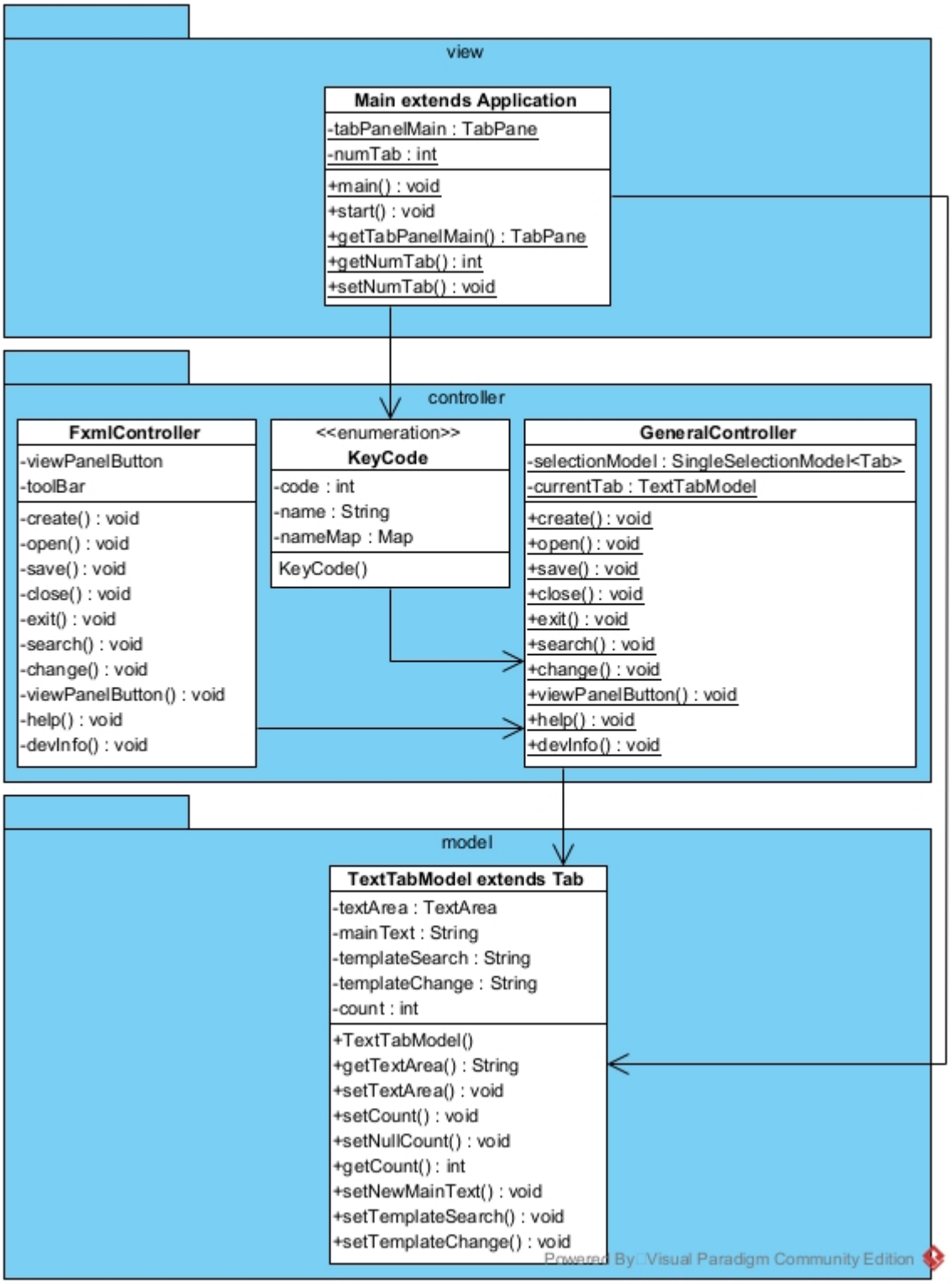

Рисунок 1 – Диаграмма классов разрабатываемого приложения

Приложение Б

(обязательное)

Листинг программы

```
package main.view;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.TabPane;
import javafx.scene.image.Image;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.stage.Stage;
import main.controller.GeneralController;

import java.io.IOException;
import java.util.Objects;

public class Main extends Application {
    private static TabPane tabPanelMain;
    private static int numTab = 0;
    public static void main(String[] args) {
        launch(args);
    }
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader loader = new
FXMLLoader(getClass().getResource("Main.fxml"));
        Parent root = loader.load();

        Scene scene = new Scene(root);

scene.getStylesheets().add(Objects.requireNonNull(getClass().get
Resource("main.css")).toExternalForm());
        scene.addEventHandler(KeyEvent.KEY_PRESSED, (event) -> {
            if (event.isControlDown()) {
                if (event.getCode() == KeyCode.N) {
                    GeneralController.create();
                }
                if (event.getCode() == KeyCode.O) {
                    GeneralController.open();
                }
                if (event.getCode() == KeyCode.S) {
                    GeneralController.save();
                }
                if (event.getCode() == KeyCode.W) {
```

```
                    GeneralController.close();
                }
                if (event.getCode() == KeyCode.F) {
                    GeneralController.search();
                }
                if (event.getCode() == KeyCode.J) {
                    GeneralController.change();
                }
            }
            if (event.getCode() == KeyCode.F1) {
                try {
                    GeneralController.help();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        });
        stage.setScene(scene);

        tabPanelMain = (TabPane) scene.lookup("#tabPanelMain");
        GeneralController.create();

        stage.setTitle("Блокнот");
        stage.getIcons().add(new
Image("/main/view/img/frame.png"));
        stage.setWidth(1024);
        stage.setHeight(768);
        stage.centerOnScreen();
        stage.show();
    }
    public static TabPane getTabPanelMain() {
        return tabPanelMain;
    }
    public static int getNumTab() {
        return numTab;
    }
    public static void setNumTab(int changer) {
        numTab = numTab + changer;
    }
}
```

```
package main.controller;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.*;
import java.io.*;

public class FxmlController {
    @FXML
    private CheckMenuItem viewPanelButton;

    @FXML
    private ToolBar toolBar;

    @FXML
    private void create(ActionEvent event) {
        GeneralController.create();
    }
    @FXML
    private void open(ActionEvent event) {
        GeneralController.open();
    }
    @FXML
    private void save(ActionEvent event) {
        GeneralController.save();
    }
    @FXML
    private void close(ActionEvent event) {
        GeneralController.close();
    }
    @FXML
    private void exit(ActionEvent event) {
        GeneralController.exit();
    }
    @FXML
    private void search(ActionEvent event) {
        GeneralController.search();
    }
    @FXML
    private void change(ActionEvent event) {
        GeneralController.change();
    }
    @FXML
    private void viewPanelButton(ActionEvent event) {
        GeneralController.viewPanelButton(viewPanelButton,
toolBar);
    }
    @FXML
    private void help(ActionEvent event) throws IOException {
        GeneralController.help();
    }
    @FXML
    private void devInfo(ActionEvent event) {
```

```
            GeneralController.devInfo();
        }
    }
```

```
package main.controller;

import javafx.application.Platform;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.GridPane;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.util.Pair;

import main.model.TextTabModel;
import main.view.Main;

import java.io.*;
import java.util.Optional;

public class GeneralController {
    private static final SingleSelectionModel<Tab>
selectionModel = Main.getTabPanelMain().getSelectionModel();
    private static TextTabModel currentTab;
    public static void create() {
        Main.setNumTab(1);
        TextTabModel newTab = new TextTabModel("Новый_" +
Main.getNumTab());
        Main.getTabPanelMain().getTabs().add(newTab);

selectionModel.select(Main.getTabPanelMain().getTabs().size() -
1);
        Main.getTabPanelMain().requestFocus();
    }
    public static void open() {
        try {
            FileChooser fileChooser = new FileChooser();
            fileChooser.setInitialDirectory(new File("C:\\"));
            fileChooser.getExtensionFilters().addAll(
                    new FileChooser.ExtensionFilter("Текстовые
файлы", "*.txt"),
                    new FileChooser.ExtensionFilter("Все файлы",
"*.*"));
            File openFile = fileChooser.showOpenDialog(null);
            FileReader reader = new FileReader(openFile);
            BufferedReader tempBuf = new BufferedReader(reader);
            TextTabModel newTab = new
TextTabModel(openFile.getName());
            Main.getTabPanelMain().getTabs().add(newTab);
            newTab.setTextArea(tempBuf.readLine());

selectionModel.select(Main.getTabPanelMain().getTabs().size() -
```

```
1);
        } catch (IOException | NullPointerException ignored) {
        }
    }
    public static void save() {
        if (Main.getTabPanelMain().getTabs().size() > 0) {
            try {
                FileChooser fileChooser = new FileChooser();
                fileChooser.setInitialDirectory(new
File("C:\\"));
                fileChooser.getExtensionFilters().addAll(
                        new
FileChooser.ExtensionFilter("Текстовые файлы", "*.txt"),
                        new FileChooser.ExtensionFilter("Все
файлы", "*.*"));
                fileChooser.setInitialFileName("new" + ".txt");
                File savedFile =
fileChooser.showSaveDialog(null);

                currentTab = (TextTabModel)
selectionModel.getSelectedItem();
                FileWriter writer = new FileWriter(savedFile);
                writer.write(currentTab.getTextArea());
                currentTab.setText(savedFile.getName());
                writer.close();
            } catch (IOException | IndexOutOfBoundsException |
NullPointerException ignored) {
            }
        } else {
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("Внимание!");
            alert.setHeaderText("Ошибка запрашиваемого
действия");
            alert.setContentText("Для выполнения операции
сохранения текста в виде файла\n" +
                    "откройте имеющийся файл или создайте новую
вкладку");
            Stage stage = (Stage)
alert.getDialogPane().getScene().getWindow();
            stage.getIcons().add(new
Image(GeneralController.class.getResource("frame.png").toString(
)));
            alert.showAndWait();
        }
    }
    public static void close() {
        if (Main.getTabPanelMain().getTabs().size() > 0) {

Main.getTabPanelMain().getTabs().remove(Main.getTabPanelMain().g
etSelectionModel().getSelectedItem());

selectionModel.select(Main.getTabPanelMain().getTabs().size() -
1);
```

```
            if (Main.getTabPanelMain().getTabs().size() > 0) {

Main.getTabPanelMain().getSelectionModel().getSelectedItem().get
Content().requestFocus();
            }
        } else {
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("Внимание!");
            alert.setHeaderText("Ошибка запрашиваемого
действия");
            alert.setContentText("Все вкладки закрыты.\n" +
                    "Для выполнения операции закрытия вкладки\n"
+
                    "откройте имеющийся файл или создайте новую
вкладку");
            Stage stage = (Stage)
alert.getDialogPane().getScene().getWindow();
            stage.getIcons().add(new
Image(GeneralController.class.getResource("frame.png").toString(
)));
            alert.showAndWait();
        }
    }
    public static void exit() {
        System.exit(0);
    }
    public static void search() {
        if (Main.getTabPanelMain().getTabs().size() > 0) {
            Dialog<String> dialog = new Dialog<>();
            dialog.setTitle("Поиск");

            Stage stage = (Stage)
dialog.getDialogPane().getScene().getWindow();
            stage.getIcons().add(new
Image(GeneralController.class.getResource("frame.png").toString(
)));

            dialog.setHeaderText("Введите символ, слово или
выражение для поиска");

            ButtonType changeButton = new ButtonType("Найти",
ButtonBar.ButtonData.OK_DONE);
            ButtonType chancelButton = new ButtonType("Отмена",
ButtonBar.ButtonData.CANCEL_CLOSE);

dialog.getDialogPane().getButtonTypes().addAll(changeButton,
chancelButton);

            GridPane grid = new GridPane();
            grid.setHgap(10);
            grid.setVgap(10);
            grid.setPadding(new Insets(10, 10, 10, 10));
```

```
            TextField searchString = new TextField();
            searchString.setPrefWidth(300);

            grid.add(new Label("Найти:"), 0, 0);
            grid.add(searchString, 1, 0);

            dialog.getDialogPane().setContent(grid);

            Platform.runLater(() ->
searchString.requestFocus());

            dialog.setResultConverter(dialogButton -> {
                if (dialogButton == changeButton) {
                    return searchString.getText();
                }
                return null;
            });

            Optional<String> result = dialog.showAndWait();
            result.ifPresent(input -> {
                currentTab = (TextTabModel)
selectionModel.getSelectedItem();
                currentTab.setTemplateSearch(input);
                currentTab.setCount();

                Alert alert = new
Alert(Alert.AlertType.INFORMATION);
                alert.setTitle("Результаты поиска");
                alert.setHeaderText(null);
                alert.setContentText("Найдено совпадений:  " +
currentTab.getCount());
                Stage stage_2 = (Stage)
alert.getDialogPane().getScene().getWindow();
                stage_2.getIcons().add(new
Image(GeneralController.class.getResource("frame.png").toString(
)));
                alert.showAndWait();
                currentTab.setNullCount();
            });
        } else {
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("Внимание!");
            alert.setHeaderText("Ошибка запрашиваемого
действия");
            alert.setContentText("Для выполнения операции поиска
текста\n" +
                    "откройте имеющийся файл или создайте новую
вкладку");
            Stage stage = (Stage)
alert.getDialogPane().getScene().getWindow();
            stage.getIcons().add(new
Image(GeneralController.class.getResource("frame.png").toString(
)));
```

```
            alert.showAndWait();
        }
    }
    public static void change() {
        if (Main.getTabPanelMain().getTabs().size() > 0) {
            Dialog<Pair<String, String>> dialog = new
Dialog<>();
            dialog.setTitle("Замена");

            Stage stage = (Stage)
dialog.getDialogPane().getScene().getWindow();
            stage.getIcons().add(new
Image(GeneralController.class.getResource("frame.png").toString(
)));

            dialog.setHeaderText("Введите символ, слово или
выражение для замены");

            ButtonType changeButton = new ButtonType("Заменить",
ButtonBar.ButtonData.OK_DONE);
            ButtonType chancelButton = new ButtonType("Отмена",
ButtonBar.ButtonData.CANCEL_CLOSE);

dialog.getDialogPane().getButtonTypes().addAll(changeButton,
chancelButton);

            GridPane grid = new GridPane();
            grid.setHgap(10);
            grid.setVgap(10);
            grid.setPadding(new Insets(10, 10, 10, 10));

            TextField searchString = new TextField();
            searchString.setPrefWidth(300);
            TextField changeString = new TextField();
            changeString.setPrefWidth(300);

            grid.add(new Label("Найти:"), 0, 0);
            grid.add(searchString, 1, 0);
            grid.add(new Label("Заменить на:"), 0, 1);
            grid.add(changeString, 1, 1);

            dialog.getDialogPane().setContent(grid);

            Platform.runLater(() ->
searchString.requestFocus());

            dialog.setResultConverter(dialogButton -> {
                if (dialogButton == changeButton) {
                    return new Pair<>(searchString.getText(),
changeString.getText());
                }
                return null;
            });
```

```
            Optional<Pair<String, String>> result =
dialog.showAndWait();
            result.ifPresent(input -> {
                currentTab = (TextTabModel)
selectionModel.getSelectedItem();
                currentTab.setTemplateSearch(input.getKey());
                currentTab.setTemplateChange(input.getValue());
                currentTab.setCount();
                currentTab.setNewMainText();

                Alert alert = new
Alert(Alert.AlertType.INFORMATION);
                alert.setTitle("Результаты замены");
                alert.setHeaderText(null);
                alert.setContentText("Выполнено замен:  " +
currentTab.getCount());
                Stage stage_2 = (Stage)
alert.getDialogPane().getScene().getWindow();
                stage_2.getIcons().add(new
Image(GeneralController.class.getResource("frame.png").toString(
)));
                alert.showAndWait();
                currentTab.setNullCount();
            });
        } else {
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("Внимание!");
            alert.setHeaderText("Ошибка запрашиваемого
действия");
            alert.setContentText("Для выполнения операции замены
текста\n" +
                    "откройте имеющийся файл или создайте новую
вкладку");
            Stage stage = (Stage)
alert.getDialogPane().getScene().getWindow();
            stage.getIcons().add(new
Image(GeneralController.class.getResource("frame.png").toString(
)));
            alert.showAndWait();
        }
    }
    public static void viewPanelButton(CheckMenuItem
viewPanelButton, ToolBar toolBar) {
        if (!viewPanelButton.isSelected()) {
            toolBar.setVisible(false);
            toolBar.setLayoutY(-25);
        } else {
            toolBar.setVisible(true);
            toolBar.setLayoutY(25);
        }
    }
    public static void help() throws IOException {
```

```
        Stage newStage = new Stage();
        FXMLLoader loader = new
FXMLLoader(GeneralController.class.getResource("Help.fxml"));

        Scene scene = new Scene(loader.load());
        newStage.setScene(scene);
        newStage.setTitle("Справка");
        newStage.getIcons().add(new
Image("/main/view/img/frame.png"));
        newStage.centerOnScreen();

        newStage.addEventHandler(KeyEvent.KEY_PRESSED, (event) -
> {
            if (event.getCode() == KeyCode.ESCAPE) {
                newStage.close();
            }
        });
        newStage.show();
    }
    public static void devInfo() {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Сведения о разработчике");
        alert.setHeaderText("ФГБОУ ВО \"ТОМСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ\n" +
                "СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»
(ТУСУР)\"\n\n" +
                "Факультет дистанционного образования\n\n" +
                "Кафедра автоматизации обработки информации
(АОИ)");
        alert.setContentText("Синичев Дмитрий Николаевич (группа
з-423П5-5)");
        Stage stage = (Stage)
alert.getDialogPane().getScene().getWindow();
        stage.getIcons().add(new
Image(GeneralController.class.getResource("frame.png").toString(
)));
        alert.showAndWait();
    }
}
```

```
package javafx.scene.input;

import java.util.HashMap;
import java.util.Map;

public enum KeyCode {
    ESCAPE(0x1B, "Esc"),
    F(0x46, "F"),
    J(  0x4A, "J"),
    N(0x4E, "N"),
    O(0x4F, "O"),
    S(0x53, "S"),
    W(0x57, "W"),
    F1(0x70, "F1");

    private final int code;
    private final String name;
    private final Map nameMap = new
HashMap(KeyCode.values().length);

    KeyCode(int code, String name) {
        this.code = code;
        this.name = name;
        for (KeyCode c : KeyCode.values()) {
            nameMap.put(c.name, c);
        }
    }
}
```

```
package main.model;

import javafx.application.Platform;
import javafx.scene.control.Tab;
import javafx.scene.control.TextArea;

public class TextTabModel extends Tab {
    private TextArea textArea;
    private String mainText;
    private String templateSearch;
    private String templateChange;
    private int count = 0;

    public TextTabModel(String fileName) {
        super(fileName);
        this.textArea = new TextArea();
        this.textArea.setFont(javafx.scene.text.Font.font(20));
        this.setContent(textArea);
        Platform.runLater(() -> textArea.requestFocus());
    }

    public String getTextArea() {
        return textArea.getText();
    }
    public void setTextArea(String text) {
        textArea.setText(text);
    }
    public void setCount() { // searchInText
        mainText = textArea.getText();
        for (int position = 0; position < mainText.length(); ) {
            if (mainText.indexOf(templateSearch, position) >= 0)
{
                count++;
                position = mainText.indexOf(templateSearch,
position) + 1;
            } else {
                break;
            }
        }
    }
    public void setNullCount() {
        count = 0;
    }
    public int getCount() {
        return count;
    }
    public void setNewMainText() {
        String newMainText = mainText.replaceAll(templateSearch,
templateChange);
        textArea.setText(newMainText);
    }
    public void setTemplateSearch(String input) {
        templateSearch = input;
```

```
    }
    public void setTemplateChange(String input) {
        templateChange = input;
    }
}
```

Main.fxml

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

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<BorderPane fx:id="borderPane" maxHeight="-Infinity" maxWidth="-
Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="555.0" prefWidth="786.0"
xmlns="http://javafx.com/javafx/11.0.14-internal"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="main.controller.FxmlController">
   <top>
       <AnchorPane BorderPane.alignment="CENTER">
           <MenuBar prefHeight="10.0" prefWidth="800.0"
AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
               <Menu mnemonicParsing="false" text="Файл">
                   <MenuItem mnemonicParsing="false"
onAction="#create" text="Новый" />
                   <MenuItem mnemonicParsing="false"
onAction="#open" text="Открыть..." />
                   <MenuItem mnemonicParsing="false"
onAction="#save" text="Сохранить..." />
                   <MenuItem mnemonicParsing="false"
onAction="#close" text="Закрыть" />
                   <SeparatorMenuItem />
                   <MenuItem mnemonicParsing="false"
onAction="#exit" text="Выход" />
               </Menu>
               <Menu mnemonicParsing="false" text="Поиск">
                   <MenuItem mnemonicParsing="false"
onAction="#search" text="Найти..." />
                   <MenuItem mnemonicParsing="false"
onAction="#change" text="Заменить..." />
               </Menu>
               <Menu mnemonicParsing="false" text="Вид">
                   <CheckMenuItem fx:id="viewPanelButton"
mnemonicParsing="false" onAction="#viewPanelButton"
selected="true" text="Панель инструментов" />
               </Menu>
               <Menu mnemonicParsing="false" text="О программе">
                   <MenuItem mnemonicParsing="false"
onAction="#help" text="Справка" />
                   <MenuItem mnemonicParsing="false"
onAction="#devInfo" text="О разработчике" />
               </Menu>
           </MenuBar>
           <ToolBar fx:id="toolBar" layoutY="25.0"
prefHeight="34.0" prefWidth="800.0">
               <Button id="createButton" onAction="#create"
prefHeight="40.0" prefWidth="40.0" />
```

```
                <Button id="openButton" onAction="#open"
prefHeight="40.0" prefWidth="40.0" />
                <Button id="saveButton" onAction="#save"
prefHeight="40.0" prefWidth="40.0" />
                <Button id="closeButton" onAction="#close"
prefHeight="40.0" prefWidth="40.0" />
                <Separator orientation="VERTICAL"
prefHeight="35.0" />
                <Button id="searchButton" onAction="#search"
prefHeight="40.0" prefWidth="40.0" />
                <Button id="changeButton" onAction="#change"
prefHeight="40.0" prefWidth="40.0" />
                <Separator orientation="VERTICAL"
prefHeight="35.0" />
                <Button id="helpButton" onAction="#help"
prefHeight="40.0" prefWidth="40.0" />
            </ToolBar>
        </AnchorPane>
    </top>
    <center>
        <TabPane fx:id="tabPanelMain">
        </TabPane>
    </center>
</BorderPane>
```

main.css

```
TabPane {
    -fx-background-color: lightgray;
}

.button {
    -fx-background-repeat: no-repeat;
    -fx-background-position: center;
}

#createButton {
    -fx-background-image: url('/main/view/img/create.png');
}

#openButton {
    -fx-background-image: url('/main/view/img/open.png');
}

#saveButton {
    -fx-background-image: url('/main/view/img/save.png');
}

#closeButton {
    -fx-background-image: url('/main/view/img/close.png');
}

#searchButton {
    -fx-background-image: url('/main/view/img/search.png');
}

#changeButton {
    -fx-background-image: url('/main/view/img/change.png');
}

#helpButton {
    -fx-background-image: url('/main/view/img/help.png');
}
```

Help.fxml

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

<?import javafx.scene.layout.*?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.control.ScrollPane?>

<ScrollPane xmlns="http://javafx.com/javafx"
            prefHeight="580.0" prefWidth="630.0"
            stylesheets="@help.css">
    <GridPane >
        <columnConstraints>
            <ColumnConstraints minWidth="110" prefWidth="110"
                               maxWidth="Infinity"
halignment="RIGHT">
            </ColumnConstraints>
            <ColumnConstraints minWidth="80" prefWidth="80"
                               maxWidth="Infinity"
hgrow="ALWAYS">
            </ColumnConstraints>
            <ColumnConstraints minWidth="40" prefWidth="40"
                               maxWidth="Infinity"
halignment="RIGHT">
            </ColumnConstraints>
            <ColumnConstraints minWidth="375" prefWidth="375"
                               maxWidth="Infinity"
hgrow="ALWAYS">
            </ColumnConstraints>
        </columnConstraints>

        <Label text="Файл" GridPane.rowIndex="0"
GridPane.columnIndex="0"
               GridPane.halignment="LEFT"
               style="-fx-font-weight: bold">
        </Label>
        <Label text="вкладка меню, позволяющая создать, открыть,
сохранить и закрыть файлы"
               GridPane.rowIndex="0" GridPane.columnIndex="1"
               GridPane.columnSpan="3"
               style="-fx-font-weight: bold">
        </Label>

        <Label text="Создать" GridPane.rowIndex="1"
GridPane.columnIndex="0"
               GridPane.halignment="LEFT">
        </Label>
        <Label text="CTRL + N" GridPane.rowIndex="1"
GridPane.columnIndex="1">
        </Label>
        <ImageView GridPane.rowIndex="1"
```

```
GridPane.columnIndex="2"
                    GridPane.halignment="CENTER">
            <Image
                    url="/main/view/img/create.png"
                    backgroundLoading="true"
            />
        </ImageView>
        <Label text="создать новый файл" GridPane.rowIndex="1"
GridPane.columnIndex="3">
        </Label>

        <Label text="Открыть" GridPane.rowIndex="2"
GridPane.columnIndex="0"
               GridPane.rowSpan="1" GridPane.columnSpan="1"
               GridPane.halignment="LEFT">
        </Label>
        <Label text="CTRL + O" GridPane.rowIndex="2"
GridPane.columnIndex="1"
               GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>
        <ImageView GridPane.rowIndex="2"
GridPane.columnIndex="2"
                   GridPane.rowSpan="1" GridPane.columnSpan="1"
                   GridPane.halignment="CENTER">
            <Image
                    url="/main/view/img/open.png"
                    backgroundLoading="true"
            />
        </ImageView>
        <Label text="открыть имеющийся файл"
GridPane.rowIndex="2" GridPane.columnIndex="3"
               GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>

        <Label text="Сохранить" GridPane.rowIndex="3"
GridPane.columnIndex="0"
               GridPane.rowSpan="1" GridPane.columnSpan="1"
               GridPane.halignment="LEFT">
        </Label>
        <Label text="CTRL + S" GridPane.rowIndex="3"
GridPane.columnIndex="1"
               GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>
        <ImageView GridPane.rowIndex="3"
GridPane.columnIndex="2"
                   GridPane.rowSpan="1" GridPane.columnSpan="1"
                   GridPane.halignment="CENTER">
            <Image
                    url="/main/view/img/save.png"
                    backgroundLoading="true"
            />
        </ImageView>
        <Label text="сохранить новый или открытый ранее файл"
```

```
GridPane.rowIndex="3" GridPane.columnIndex="3"
                GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>

        <Label text="Закрыть" GridPane.rowIndex="4"
GridPane.columnIndex="0"
                GridPane.rowSpan="1" GridPane.columnSpan="1"
                GridPane.halignment="LEFT">
        </Label>
        <Label text="CTRL + W" GridPane.rowIndex="4"
GridPane.columnIndex="1"
                GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>
        <ImageView GridPane.rowIndex="4"
GridPane.columnIndex="2"
                    GridPane.rowSpan="1" GridPane.columnSpan="1"
                    GridPane.halignment="CENTER">
            <Image
                    url="/main/view/img/close.png"
                    backgroundLoading="true"
            />
        </ImageView>
        <Label text="закрыть вкладку текущего файла"
GridPane.rowIndex="4" GridPane.columnIndex="3"
                GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>

        <Label text="Выход" GridPane.rowIndex="5"
GridPane.columnIndex="0"
                GridPane.rowSpan="1" GridPane.columnSpan="1"
                GridPane.halignment="LEFT">
        </Label>
        <Label text="ALT + F4" GridPane.rowIndex="5"
GridPane.columnIndex="1"
                GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>
        <ImageView GridPane.rowIndex="5"
GridPane.columnIndex="2"
                    GridPane.rowSpan="1" GridPane.columnSpan="1"
                    GridPane.halignment="CENTER">

        </ImageView>
        <Label text="выход из приложения" GridPane.rowIndex="5"
GridPane.columnIndex="3"
                GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>

        <Label text="Поиск" GridPane.rowIndex="6"
GridPane.columnIndex="0"
                GridPane.halignment="LEFT"
                style="-fx-font-weight: bold">
        </Label>
        <Label text="вкладка меню, позволяющая найти или
```

```
заменить часть текста"
                GridPane.rowIndex="6" GridPane.columnIndex="1"
                GridPane.columnSpan="3"
                style="-fx-font-weight: bold">
        </Label>

        <Label text="Найти..." GridPane.rowIndex="7"
GridPane.columnIndex="0"
                GridPane.halignment="LEFT">
        </Label>
        <Label text="CTRL + F" GridPane.rowIndex="7"
GridPane.columnIndex="1">
        </Label>
        <ImageView GridPane.rowIndex="7"
GridPane.columnIndex="2"
                   GridPane.halignment="CENTER">
            <Image
                    url="/main/view/img/search.png"
                    backgroundLoading="true"
            />
        </ImageView>
        <Label text="найти символ, слово или выражение"
GridPane.rowIndex="7" GridPane.columnIndex="3">
        </Label>

        <Label text="Заменить..." GridPane.rowIndex="8"
GridPane.columnIndex="0"
                GridPane.rowSpan="1" GridPane.columnSpan="1"
                GridPane.halignment="LEFT">
        </Label>
        <Label text="CTRL + J" GridPane.rowIndex="8"
GridPane.columnIndex="1"
                GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>
        <ImageView GridPane.rowIndex="8"
GridPane.columnIndex="2"
                   GridPane.rowSpan="1" GridPane.columnSpan="1"
                   GridPane.halignment="CENTER">
            <Image
                    url="/main/view/img/change.png"
                    backgroundLoading="true"
            />
        </ImageView>
        <Label text="заменить символ, слово или выражение"
GridPane.rowIndex="8" GridPane.columnIndex="3"
                GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>

        <Label text="Вид" GridPane.rowIndex="9"
GridPane.columnIndex="0"
                GridPane.halignment="LEFT"
                style="-fx-font-weight: bold">
        </Label>
```

```
        <Label text="вкладка меню, позволяющая изменить
отображение панели кнопок и строки состояния"
               GridPane.rowIndex="9" GridPane.columnIndex="1"
               GridPane.columnSpan="3"
               style="-fx-font-weight: bold">
        </Label>

        <Label text="Панель кнопок" GridPane.rowIndex="10"
GridPane.columnIndex="0"
               GridPane.halignment="LEFT">
        </Label>
        <Label GridPane.rowIndex="10" GridPane.columnIndex="1">
        </Label>
        <ImageView GridPane.rowIndex="10"
GridPane.columnIndex="2"
                   GridPane.halignment="CENTER">
        </ImageView>
        <Label text="вкл/выкл режима отображения панели кнопок"
GridPane.rowIndex="10" GridPane.columnIndex="3">
        </Label>

        <Label text="О программе" GridPane.rowIndex="12"
GridPane.columnIndex="0"
               GridPane.halignment="LEFT"
               style="-fx-font-weight: bold">
        </Label>
        <Label text="вкладка меню, позволяющая получить
дополнительную информацию о работе программы и разработчике"
               GridPane.rowIndex="12" GridPane.columnIndex="1"
               GridPane.columnSpan="3"
               style="-fx-font-weight: bold">
        </Label>

        <Label text="Справка" GridPane.rowIndex="13"
GridPane.columnIndex="0"
               GridPane.halignment="LEFT">
        </Label>
        <Label text="F1" GridPane.rowIndex="13"
GridPane.columnIndex="1">
        </Label>
        <ImageView GridPane.rowIndex="13"
GridPane.columnIndex="2"
                   GridPane.halignment="CENTER">
            <Image
                    url="/main/view/img/help.png"
                    backgroundLoading="true"
            />
        </ImageView>
        <Label text="справочная информация об органах управления
программой" GridPane.rowIndex="13" GridPane.columnIndex="3">
        </Label>

        <Label text="О разработчике" GridPane.rowIndex="14"
```

```
GridPane.columnIndex="0"
                GridPane.rowSpan="1" GridPane.columnSpan="1"
                GridPane.halignment="LEFT">
        </Label>
        <Label GridPane.rowIndex="14" GridPane.columnIndex="1"
                GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>
        <ImageView GridPane.rowIndex="14"
GridPane.columnIndex="2"
                    GridPane.rowSpan="1" GridPane.columnSpan="1"
                    GridPane.halignment="CENTER">
        </ImageView>
        <Label text="краткая информация о разработчике"
GridPane.rowIndex="14" GridPane.columnIndex="3"
                GridPane.rowSpan="1" GridPane.columnSpan="1">
        </Label>
    </GridPane>
</ScrollPane>
```

help.css

```
.label                                                    {
    -fx-font-size:                                      12;
    -fx-wrap-text:                                    true;
    -fx-padding:                                        10;
    -fx-column-halignment:                            LEFT;
}
```

```
package tests;

import javafx.application.Application;
import main.model.TextTabModel;
import main.view.Main;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;

import java.lang.reflect.Field;

import static org.junit.jupiter.api.Assertions.assertEquals;

class TextTabModelTest {
    @BeforeAll
    public static void setUp() {
        new Thread(() ->
Application.launch(Main.class)).start();
    }

    @Test
    void getTextArea() {
        String expected = "getTextAreaTest";
        TextTabModel testTab = new TextTabModel("testTab");
        testTab.setTextArea(expected);
        String actual = testTab.getTextArea();
        assertEquals(expected, actual);
    }

    @Test
    void setTextArea() {
        String expected = "setTextAreaTest";
        TextTabModel testTab = new TextTabModel("testTab");
        testTab.setTextArea(expected);
        String actual = testTab.getTextArea();
        assertEquals(expected, actual);
    }

    @Test
    void setCount() throws NoSuchFieldException {
        TextTabModel testTab = new TextTabModel("testTab");

        int expected = 5;

        testTab.setTextArea("88888");
        Field tMainText =
testTab.getClass().getDeclaredField("mainText");
        tMainText.setAccessible(true);
        testTab.setTemplateSearch("8");
        testTab.setCount();

        int actual = testTab.getCount();
        assertEquals(expected, actual);
    }
```

```
    @Test
    void setNullCount() {
        int expected = 0;
        TextTabModel testTab = new TextTabModel("testTab");
        testTab.setNullCount();
        int actual = testTab.getCount();
        assertEquals(expected, actual);
    }

    @Test
    void getCount() {
        int expected = 0;
        TextTabModel testTab = new TextTabModel("testTab");
        testTab.setNullCount();
        int actual = testTab.getCount();
        assertEquals(expected, actual);
    }

    @Test
    void setNewMainText() throws NoSuchFieldException,
IllegalAccessException {
        TextTabModel testTab = new TextTabModel("testTab");

        String expected = "Change";
        testTab.setTemplateChange(expected);

        testTab.setTemplateSearch("Search");
        Field tMainText =
testTab.getClass().getDeclaredField("mainText");
        tMainText.setAccessible(true);
        tMainText.set(testTab, (String) "Change");

        Field tTextArea =
testTab.getClass().getDeclaredField("textArea");
        tTextArea.setAccessible(true);

        testTab.setNewMainText();
        String actual = testTab.getTextArea();
        assertEquals(expected, actual);
    }

    @Test
    void setTemplateSearch() throws NoSuchFieldException,
IllegalAccessException {
        TextTabModel testTab = new TextTabModel("testTab");

        String expected = "setTemplateSearch";
        testTab.setTemplateSearch(expected);

        Field field =
testTab.getClass().getDeclaredField("templateSearch");
        field.setAccessible(true);
```

```
        String actual = (String) field.get(testTab);
        assertEquals(expected, actual);
    }

    @Test
    void setTemplateChange() throws NoSuchFieldException,
IllegalAccessException {
        TextTabModel testTab = new TextTabModel("testTab");

        String expected = "setTemplateChange";
        testTab.setTemplateChange(expected);

        Field field =
testTab.getClass().getDeclaredField("templateChange");
        field.setAccessible(true);
        String actual = (String) field.get(testTab);
        assertEquals(expected, actual);
    }
}
```