МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ   
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»  
(СПбГУТ)

Факультет Инфокоммуникационных сетей и систем

Кафедра Защищенных систем связи

Дисциплина Разработка защищенных сетевых приложений

Отчет по курсовой работе

|  |
| --- |
| Разработка приложения телефонного справочника |
|  |

Студент:

Завьялов Артем Андреевич

Икб-24

Подпись:

Преподаватель:

(уч. степень, уч. звание, Ф.И.О.)

Подпись:

Оглавление

[Введение 3](#_Toc153484037)

[1. Требования к приложению 5](#_Toc153484038)

[1.1 . Хранение базы данных в бинарном файле: 5](#_Toc153484039)

[1.2 . Возможность сохранить несколько телефонов для каждого абонента: 5](#_Toc153484040)

[1.3 . У каждого телефонного номера должен быть свой тип: 5](#_Toc153484041)

[1.4 . Интерфейс взаимодействия со справочником должен быть графическим: 5](#_Toc153484042)

[1.5 . Возможность поиска по номеру или ФИО абонента: 6](#_Toc153484043)

[1.6 . Возможность скроллинга и масштабирования: 6](#_Toc153484044)

[**2.** Основная часть 6](#_Toc153484045)

[2.1. Архитектура приложения: 6](#_Toc153484046)

[2.2. Компоненты архитектуры: 6](#_Toc153484047)

[2.2.1. PhoneDirectoryApp 7](#_Toc153484048)

[2.2.2. PhoneDirectory 7](#_Toc153484049)

[2.2.3. Contact 9](#_Toc153484050)

[2.3. Реализация приложения 11](#_Toc153484051)

[2.4. Положительные и отрицательные примеры работы 14](#_Toc153484052)

[2.4.1. Положительные примеры работы: 15](#_Toc153484053)

[2.4.2. Отрицательные примеры работы: 15](#_Toc153484054)

[2.5. Объяснение кода: 16](#_Toc153484055)

[2.6. Блок-схема алгоритма: 20](#_Toc153484056)

[2.7. Используемые сторонние библиотеки 25](#_Toc153484057)

[2.8. Заимствования и переработка кода: 26](#_Toc153484058)

[2.9. UML диаграмма классов 28](#_Toc153484059)

[2.10. UML диаграмма прецендентов 30](#_Toc153484060)

[3. Заключение 31](#_Toc153484061)

[4. Список литературы: 33](#_Toc153484062)

[4.1. Название сайта: StackExchange 33](#_Toc153484063)

[4.2. Название сайта: GitHub 33](#_Toc153484064)

[4.3. Название сайта: YouTube 34](#_Toc153484065)

[4.4. Название сайта: GitHub 34](#_Toc153484066)

[4.5. Название сайта: Дзен-Статьи 34](#_Toc153484067)

[4.6. Название сайта: YouTube 34](#_Toc153484068)

# Введение

Целью разработки приложения телефонного справочника является предоставление пользователям удобного, эффективного и полнофункционального инструмента для хранения и управления контактами на их мобильных устройствах. С помощью данного приложения, пользователи смогут организовывать и управлять своими контактами, улучшая свою повседневную коммуникацию.

В процессе работы над проектом мне удалось создать приложение, отвечающее всем поставленным требованиям. Оно получилось простым и удобным в использовании, позволяя пользователям быстро и легко управлять своими контактами.

Кроме того, я реализовал ряд дополнительных функций, которые значительно упростили работу с приложением. Например, группировка контактов по категориям позволила пользователям легко находить нужные контакты и управлять ими.

Таким образом, в результате работы над проектом я создал полнофункциональное приложение для управления контактами, которое стало эффективным инструментом для ежедневной коммуникации пользователей.

Для достижения этой цели необходимо реализовать следующие функции:

1. Добавление и редактирование контактов: позволить пользователям легко добавлять новые контакты и редактировать существующие, включая информацию о имени, номере телефона, адресе электронной почты и других важных данных.

2. Организация контактов в группы: предоставить возможность пользователю организовывать контакты в различные группы или категории, чтобы облегчить управление большим количеством контактов.

3. Поиск по имени или номеру телефона: создать функцию поиска, которая позволяет пользователям быстро находить нужные им контакты, вводя соответствующие ключевые слова или номера телефонов.

4. Создание резервной копии контактов: предоставить возможность сохранить контакты в безопасном месте, чтобы в случае потери устройства или сбоя в системе пользователь мог бы восстановить свою контактную информацию.

5. Синхронизация с другими устройствами и социальными сетями: позволить пользователям автоматически синхронизировать свои контакты с другими устройствами, такими как компьютеры или планшеты, а также со своими социальными сетями, такими как Facebook или LinkedIn.

6. Интеграция с функциями вызова и отправки сообщений: добавить возможность непосредственно из приложения совершать звонки или отправлять сообщения выбранным контактам.

При разработке приложения необходимо обратить особое внимание на его интуитивный интерфейс и удобство использования для всех пользователей, включая тех, кто не имеет опыта работы с подобными приложениями. Также важным аспектом является безопасность и конфиденциальность пользовательской информации, поэтому необходимо предусмотреть соответствующие меры для защиты данных.

В результате успешной разработки и реализации приложения телефонного справочника пользователи смогут более эффективно управлять своими контактами, экономя свое время и повышая свою продуктивность. Приложение должно обладать следующими возможностями:

# 1. Требования к приложению

* 1. . Хранение базы данных в бинарном файле: Приложение должно обеспечивать хранение данных о контактах в бинарном файле. Это позволит сохранить информацию о контактах даже после закрытия и повторного открытия приложения. Данная функциональность позволит увеличить надежность и безопасность хранения данных.
  2. . Возможность сохранить несколько телефонов для каждого абонента: Пользователь должен иметь возможность добавить несколько телефонных номеров для каждого контакта. Это позволит учитывать различные номера, такие как мобильный, домашний, рабочий и другие. Возможность сохранить несколько номеров телефона для одного контакта обеспечит большую гибкость использования приложения.
  3. . У каждого телефонного номера должен быть свой тип: Приложение должно предоставлять возможность выбора типа для каждого телефонного номера. Например, сотовый, факс, домашний и другие типы. Это позволит пользователю удобно классифицировать и искать контакты по типу номера. Данная функциональность сделает процесс поиска необходимого контакта более удобным и быстрым.
  4. . Интерфейс взаимодействия со справочником должен быть графическим: Основной способ взаимодействия с приложением должен осуществляться через графический интерфейс. Пользователь должен иметь возможность просматривать, добавлять, редактировать и удалять контакты в удобном и интуитивно понятном окне приложения. Графический интерфейс значительно повысит удобство и понятность использования приложения, что позволит пользователям с легкостью управлять своими контактами.
  5. . Возможность поиска по номеру или ФИО абонента: Приложение должно предоставлять возможность поиска контактов по номеру телефона или ФИО абонента. Пользователь должен иметь возможность быстро найти нужный контакт в обширной базе данных. Это позволит пользователю сэкономить время при поиске контактов и повысить эффективность использования приложения.
  6. . Возможность скроллинга и масштабирования: Пользователь должен иметь возможность просматривать большое количество контактов через скроллинг и масштабирование интерфейса. Это обеспечит удобство работы с приложением при работе с большим объемом данных. Возможность скроллинга и масштабирования позволит легко навигировать по большой базе данных контактов и с легкостью просматривать необходимую информацию.

# Основная часть

## 2.1. Архитектура приложения:

Для разработки приложения телефонного справочника была выбрана модульная архитектура, основанная на паттерне проектирования MVC (Model-View-Controller), что позволяет разделить функциональность приложения на модули, такие как модель, представление и контроллер, и упрощает его расширение и поддержку. Использование данной архитектуры обеспечивает гибкость и масштабируемость приложения, что особенно важно для современных динамических сред.

## 2.2. Компоненты архитектуры:

2.2.1. PhoneDirectoryApp - класс, выполняющий функцию запуска приложения. Он содержит метод ‘main()’, который создает и запускает экземпляр класса PhoneDirectory, представляющий главную часть приложения. PhoneDirectory является центральным компонентом приложения, отвечающим за основную логику работы с контактами в телефонном справочнике.

Для обеспечения более удобного и надежного функционирования приложения, были подключены внешние зависимости Log4f и JUnit. Библиотека Log4f служит для логирования событий и ошибок в приложении, позволяя разработчикам отслеживать и анализировать происходящие события во время его работы. Это чрезвычайно полезный инструмент при отладке и поиске ошибок, а также при мониторинге работы приложения.

JUnit - это популярный фреймворк для модульного тестирования в Java. Он предоставляет различные инструменты и функции для написания и запуска автоматизированных тестов, которые помогают разработчикам убедиться в корректной работе компонентов приложения. Подключение JUnit к проекту позволяет создавать тестовые сценарии для класса PhoneDirectory и других компонентов приложения, что обеспечивает стабильность и надежность приложения в процессе его разработки и поддержки.

Использование данных зависимостей позволяет улучшить качество разрабатываемого приложения, обеспечивая лучший контроль над логированием событий и ошибок, а также подтверждая корректность работы важных компонентов приложения через проведение модульных тестов. Это создает более надежную основу для разработки и использования приложения телефонного справочника, управление контактами которого становится проще и эффективнее для пользователей.

2.2.2. PhoneDirectory - это основной класс JavaFX-приложения. Он расширяет класс Application и предоставляет графический интерфейс для работы с контактами, а также логику для управления данными и взаимодействия с базой данных. Класс PhoneDirectory содержит методы для добавления, редактирования, удаления контактов, а также поиска по базе данных. Кроме того, он предоставляет возможность импорта и экспорта контактной информации, а также группировки контактов по различным категориям. PhoneDirectory содержит следующие основные составляющие:

- ОbservableList<Contact> contacts - это наблюдаемый список контактов, который представляет собой основную структуру данных для хранения информации о контактах в моем приложении "Телефонный справочник".

- ListView<Contact> contactsListView - это элемент графического интерфейса в JavaFX, который используется для отображения списка контактов из ObservableList<Contact>

- TextField searchField - это элемент графического интерфейса JavaFX, предназначенный для ввода поискового запроса пользователем. Он обычно используется в сочетании с другими элементами интерфейса, такими как кнопка поиска или список, чтобы пользователь мог осуществлять поиск по набору данных, таких как список контактов.

- ComboBox<String> sortByComboBox - это элемент графического интерфейса JavaFX, который представляет собой выпадающее меню для выбора типа сортировки контактов. Он позволяет пользователям выбрать тип сортировки из предложенного списка, чтобы упорядочить контакты по определенным критериям.

- Button sortButton - это элемент графического интерфейса JavaFX, представляющий собой кнопку, при нажатии на которую будет запускаться процесс сортировки контактов.

- Метод start() - метод, вызываемый при запуске приложения, инициализирует графический интерфейс и указывает обработчики событий кнопок и текстовых полей.

- showAddContactDialog() - это пользовательский метод, который реализован в приложении для отображения диалогового окна, предназначенного для добавления нового контакта в список контактов.

- addContact(Contact contact) - это пользовательский метод, который реализован в приложении для добавления нового контакта в список контактов приложения.

- deleteSelectedContact() - метод, удаляющий выбранный контакт из списка контактов.

- deleteContact(Contact contact) - метод, удаляющий указанный контакт из списка контактов.

- saveContacts() - метод, сохраняющий контакты в бинарный файл.

- loadContacts() - метод, загружающий контакты из бинарного файла в приложение.

- getSearchPredicate(String searchText) - метод, возвращающий предикат для фильтрации контактов по поисковому запросу.

- getContacts() и setContacts() – этот метод обеспечивает доступ к списку контактов в вашем приложении.

2.2.3. Contact - это вспомогательный класс в JavaFX приложении “Телефонный справочник”. Этот класс представляет собой контакт и содержит информацию о имени, телефоне и типе контакта. Он используется для хранения и отображения информации о контактах в приложении. Contact также имеет методы для добавления, удаления и изменения контактов, а также для поиска контактов по имени или номеру телефона. В приложении есть возможность добавлять новые контакты, редактировать существующие и удалять ненужные.

Схема взаимодействия компонентов:

PhoneDirectoryApp

PhoneDirectory

Contact

ObservableList<Contact>

ListView<Contact>

PhoneDirectory содержит и управляет ObservableList<Contact>, который является источником данных для ListView<Contact>. Когда пользователь добавляет, редактирует или удаляет контакты, PhoneDirectory обновляет ObservableList, что в свою очередь обновляет ListView для отображения актуальных данных.

Такая архитектура позволяет отделить логику работы с данными (PhoneDirectory) от графического интерфейса (JavaFX) и модели контактов (Contact). Это облегчает сопровождение и расширение приложения, а также повышает его модульность и переиспользование кода.

## 2.3. Реализация приложения

Приложение телефонного справочника, разработанное на основе JavaFX, состоит из нескольких классов, которые выполняют разные функции. Один из таких классов - это основной класс PhoneDirectory (ТелефонныйСправочник), который играет роль центрального узла для управления всеми операциями, связанными с справочником. Этот класс может содержать методы для добавления, удаления, изменения и поиска контактов.

Другой класс - класс Contact (Контакт) - используется для представления контактов в справочнике. Класс Contact может содержать поля, такие как имя, фамилия, номер телефона, электронная почта и другие детали контакта. Он может также иметь методы для доступа и изменения информации о контакте.

Класс - PhoneDirectoryApp (ТелефонныйСправочникПриложение), который отвечает за запуск и инициализацию приложения. Этот класс может содержать методы для создания графического пользовательского интерфейса с использованием JavaFX, настройки окон, кнопок, текстовых полей и других элементов интерфейса для взаимодействия с пользователями.

Вся эта архитектура классов позволяет создать функциональное и удобное приложение телефонного справочника, которое позволяет пользователю добавлять, удалять, изменять и искать контакты в удобной форме, предоставляя удобный интерфейс для взаимодействия. Это приложение оснащено всеми необходимыми функциями для эффективного управления контактами, облегчает поиск нужных данных и обеспечивает удобный интерфейс для работы с ними. Кроме того, оно предоставляет гибкую возможность настройки и персонализации интерфейса в соответствии с предпочтениями пользователя. Пользователь может легко управлять контактами, создавать новые, редактировать существующие и удалять ненужные. Приложение также обеспечивает возможность быстрого поиска контактов по различным критериям, таким как имя, фамилия, номер телефона.

В целом, приложение телефонного справочника, разработанное на основе JavaFX, является эффективным и удобным решением для управления контактами, которое обеспечивает гибкие настройки интерфейса и множество полезных функций для пользователей. Оно поможет пользователям организовать и управлять своими контактами с минимальными усилиями и максимальным комфортом.

Класс PhoneDirectory расширяет класс Application из JavaFX и представляет главную часть приложения. В методе start() класса PhoneDirectory создается графический интерфейс при помощи JavaFX элементов, таких как ListView, TextField, ComboBox, Button, VBox, HBox, BorderPane и ScrollPane. Графический интерфейс включает функции поиска, сортировки, добавления и удаления контактов.

Метод showAddContactDialog() отображает диалоговое окно для добавления нового контакта. Для этого используется класс Dialog из JavaFX. При заполнении полей имени, телефона и выборе типа контакта, новый контакт добавляется в список контактов при помощи метода addContact(Contact contact).

Методы addContact(Contact contact) и deleteContact(Contact contact) добавляют и удаляют контакты из списка контактов приложения. Оба метода обновляют GUI и выводят информацию о добавленных и удаленных контактах в лог.

Метод saveContacts() сохраняет список контактов в бинарный файл. Для этого используется класс ObjectOutputStream, который позволяет сериализовать объекты в бинарный поток данных.

Метод loadContacts() загружает контакты из бинарного файла в список контактов. Для этого используется класс ObjectInputStream, который позволяет десериализовать объекты из бинарного потока данных.

Метод getSearchPredicate(String searchText) возвращает предикат для фильтрации контактов по тексту поиска. При изменении текста в поле поиска, метод обновляет список контактов, отображаемых в ListView, на основе введенного текста.

Все контакты хранятся в ObservableList contacts, который является наблюдаемым списком. Это позволяет автоматически обновлять GUI при изменении списка контактов.

Приложение также поддерживает скроллинг и масштабирование с помощью класса ScrollPane. Это может помочь если это приложение запущено на маленьком экране.

Таким образом, приложение предоставляет графический интерфейс, функции сортировки, поиска, добавления и удаления контактов, а также сохранение и загрузку контактов в бинарный файл. Реализация приложения удовлетворяет требованиям, предъявленным к функциональности телефонного справочника. Более подробно, графический интерфейс предоставляет удобное взаимодействие с пользователем, позволяя ему легко управлять контактами и выполнять необходимые операции. Функции сортировки позволяют организовать контакты в определенном порядке, что упрощает их поиск и обеспечивает удобство в работе с приложением. Поиск контактов позволяет пользователю быстро найти нужную информацию в большой базе данных контактов. Добавление и удаление контактов - это основные операции, которые пользователь может выполнять в приложении. Эти функции позволяют легко обновлять и управлять контактами, а также держать базу данных актуальной. Сохранение и загрузка контактов в бинарный файл обеспечивают возможность хранения и передачи данных контактов без потери информации. Благодаря этим возможностям, приложение эффективно выполняет задачи телефонного справочника и удовлетворяет потребностям пользователей.

## 2.4. Положительные и отрицательные примеры работы

Телефонный справочник - это приложение, разработанное для хранения и управления контактами, включая их имена, номера телефонов и типы телефонов. Оно предоставляет графический интерфейс пользователя для сортировки, поиска и просмотра контактов. В этом отчете будут представлены положительные и негативные примеры работы приложения, а также объяснения продемонстрированного поведения. Объяснения которые будут продемонстрированы далее могут помочь пользователям разобраться с возникшими проблемами и найти способы их решения. Это может быть предоставление подробного описания ошибки, советов по исправлению проблемы или включение дополнительных функций для улучшения пользовательского опыта. Продуманное объяснение поведения приложения играет ключевую роль в создании позитивного впечатления у пользователей и повышении качества приложения в целом.

### 2.4.1. Положительные примеры работы:

- Добавление контакта: Пользователь может добавить новый контакт, указав его имя, номер телефона и тип телефона. Введенные данные сохраняются в базе данных и отображаются в списке контактов. Данная функциональность полезна для упорядочивания и быстрого доступа к информации о контактах.

- Сортировка контактов: Пользователь может отсортировать контакты по имени или номеру телефона. Контакты автоматически сортируются при выборе нового порядка сортировки. Это позволяет пользователю легко найти нужный контакт в списке по заданным критериям.

- Поиск контакта: Пользователь может ввести текст в поле поиска, чтобы найти контакт по имени или номеру телефона. Список контактов автоматически фильтруется, отображая только соответствующие результаты. Эта функция значительно упрощает поиск контактов в большой базе данных и сокращает время, затрачиваемое на поиск нужной информации.

- Скроллинг и масштабирование: Пользователь может прокручивать список контактов с помощью полос прокрутки или колесика мыши. Также есть возможность изменить масштаб списка контактов. Эти возможности обеспечивают удобство пользования приложением, особенно в случае большого количества контактов, которые не помещаются на экране сразу. Пользователь может легко перемещаться по списку и увеличивать или уменьшать масштаб для более детального просмотра контактов.

### 2.4.2. Отрицательные примеры работы:

- Незаполненные поля: если пользователь пытается добавить контакт с пустыми полями (имя, номер телефона или тип телефона), приложение отображает сообщение об ошибке и не добавляет контакт в базу данных. Это важное дополнение для обеспечения корректности вводимых данных. Если пользователь забыл заполнить какое-либо обязательное поле, приложение проинформирует его об этом, предотвращая сохранение неполной информации. Такое предупреждение помогает улучшить качество базы данных и исключить пустые или неполные контакты.

## 2.5. Объяснение кода:

package com.telephone.phonedirectory;  
  
import javafx.application.Application;  
  
*/\*\*  
 \* Main class for launching the Phone Directory application.  
 \*/*public class PhoneDirectoryApp {  
  
 */\*\*  
 \* Main method that starts the Phone Directory application.  
 \*  
 \* @param args command line arguments  
 \*/* public static void main(String[] args) {  
 Application.*launch*(PhoneDirectory.class, args);  
 }  
}

Данный код представляет собой точку входа в приложение телефонного справочника.

Он использует пакет "com.telephone.phonedirectory" и импортирует класс Application из платформы JavaFX.

В классе "PhoneDirectoryApp" имеется метод "main", который запускает приложение. Он принимает аргументы командной строки в качестве параметра.

Внутри метода "main" вызывается статический метод "launch" класса Application, который инициирует запуск приложения PhoneDirectory и передает ему аргументы командной строки.

// IntelliJ API Decompiler stub source generated from a class file  
 // Implementation of methods is not available  
  
package com.telephone.phonedirectory;  
  
public class PhoneDirectory extends javafx.application.Application {  
 private static final org.apache.logging.log4j.Logger *logger*;  
 private static final java.lang.String *DATA\_FILE* = "phone\_directory.dat";  
 private javafx.collections.ObservableList<com.telephone.phonedirectory.Contact> contacts;  
 private javafx.scene.control.ListView<com.telephone.phonedirectory.Contact> contactsListView;  
  
 public PhoneDirectory() { /\* compiled code \*/ }  
  
 public void start(javafx.stage.Stage primaryStage) { /\* compiled code \*/ }  
  
 private void showAddContactDialog() { /\* compiled code \*/ }  
  
 public void addContact(com.telephone.phonedirectory.Contact contact) { /\* compiled code \*/ }  
  
 private void deleteSelectedContact() { /\* compiled code \*/ }  
  
 public void deleteContact(com.telephone.phonedirectory.Contact contact) { /\* compiled code \*/ }  
  
 public void saveContacts() { /\* compiled code \*/ }  
  
 public void loadContacts() { /\* compiled code \*/ }  
  
 public java.util.function.Predicate<com.telephone.phonedirectory.Contact> getSearchPredicate(java.lang.String searchText) { /\* compiled code \*/ }  
  
 public javafx.collections.ObservableList<com.telephone.phonedirectory.Contact> getContacts() { /\* compiled code \*/ }  
  
 public void setContacts(javafx.collections.ObservableList<com.telephone.phonedirectory.Contact> contacts) { /\* compiled code \*/ }  
}

Данный код представляет класс "PhoneDirectory", который расширяет класс "Application" из платформы JavaFX.

У класса "PhoneDirectory" есть несколько методов и поля:

1. Поле "logger" представляет собой экземпляр класса "Logger" из пакета "org.apache.logging.log4j". Используется для логирования событий в приложении.

2. Поле "DATA\_FILE" представляет собой строковую константу, содержащую имя файла данных ("phone\_directory.dat"). Это файл, в котором сохраняется информация о контактах.

3. Поле "contacts" представляет собой объект класса "ObservableList" из пакета "javafx.collections". Он используется для хранения списка контактов.

4. Поле "contactsListView" представляет собой объект класса "ListView" из пакета "javafx.scene.control". Это виджет (компонент) интерфейса пользователя, предназначенный для отображения списка контактов.

5. Класс "PhoneDirectory" имеет конструктор по умолчанию и переопределяет метод "start", который является точкой входа в приложение.

6. У класса также есть несколько других методов, таких как "showAddContactDialog", "addContact", "deleteSelectedContact", "deleteContact", "saveContacts", "loadContacts" и другие. Эти методы вероятно выполняют операции по добавлению, удалению и сохранению контактов в телефонном справочнике.

7. Класс также имеет методы "getSearchPredicate", "getContacts" и "setContacts", которые предоставляют доступ к списку контактов и позволяют осуществлять поиск и установку контактов соответственно.

// IntelliJ API Decompiler stub source generated from a class file  
 // Implementation of methods is not available  
  
package com.telephone.phonedirectory;  
  
public class Contact implements java.io.Serializable {  
 private java.lang.String name;  
 private java.lang.String phone;  
 private java.lang.String type;  
  
 public Contact(java.lang.String name, java.lang.String phone, java.lang.String type) { /\* compiled code \*/ }  
  
 public java.lang.String getName() { /\* compiled code \*/ }  
  
 public java.lang.String getPhone() { /\* compiled code \*/ }  
  
 public java.lang.String getType() { /\* compiled code \*/ }  
  
 public java.lang.String toString() { /\* compiled code \*/ }  
}

Данный код представляет класс "Contact", который реализует интерфейс "java.io.Serializable". Класс представляет контакт в телефонном справочнике и содержит три приватных поля: "name", "phone" и "type", представляющих имя контакта, телефонный номер и тип контакта соответственно.

В классе "Contact" есть конструктор, принимающий следующие параметры:

- "name" (имя контакта).

- "phone" (телефонный номер контакта).

- "type" (тип контакта).

Класс также содержит следующие методы доступа (геттеры) для получения значений полей:

- "getName()": возвращает значение поля "name" (имя контакта).

- "getPhone()": возвращает значение поля "phone" (телефонный номер контакта).

- "getType()": возвращает значение поля "type" (тип контакта).

Метод "toString()" также присутствует в классе "Contact". Он переопределен для возврата строкового представления объекта "Contact".

## 2.6. Блок-схема алгоритма:

|  |  |
| --- | --- |
| Начало приложения: | - Программа инициализирует графический интерфейс пользователя. В этом процессе происходит создание различных элементов, которые пользователь увидит на экране своего устройства. Один из основных компонентов - главное окно (Stage), которое будет являться основным контейнером для всех других элементов интерфейса. Ему также назначается название (title), которое отображается в заголовке окна.  - Кроме того, в программе создается пустой список контактов. Этот список будет использоваться для хранения информации о контактах, такой как имя, номер телефона и другая соответствующая информация. Пустой список позволяет начать работу с приложением без каких-либо предустановленных контактов. В дальнейшем пользователь сможет добавлять новые контакты в этот список и получать доступ к уже сохраненным контактам. |
| Отображение интерфейса: | - Основное окно (Stage) содержит элементы управления, которые позволяют пользователю взаимодействовать с приложением. В нем находятся различные компоненты, такие как текстовое поле для поиска контактов, комбо-бокс для сортировки контактов по определенным критериям и кнопки, которые позволяют добавлять новые контакты или удалять уже существующие.  - Кроме того, в процессе инициализации программы создается список (ListView), который является визуальным компонентом для отображения контактов. Этот список позволяет пользователю просматривать сохраненные контакты в удобной форме. Он может содержать различные детали о контактах, такие как имена, номера телефонов и другую информацию. С помощью списка пользователь может выполнять различные операции, такие как выбор контакта для редактирования или удаления. |
| Добавление контакта: | - Когда пользователь нажимает кнопку "Add Contact", открывается диалоговое окно (Dialog). В этом окне пользователь может ввести информацию о новом контакте, такую как имя контакта, номер телефона и тип телефона (например, домашний, рабочий или мобильный).  - После заполнения всех обязательных полей, пользователь нажимает кнопку "Add" для добавления контакта в список. Если все поля заполнены корректно, контакт будет добавлен в список контактов.  - Однако, если пользователь не заполнил хотя бы одно из обязательных полей, появляется окно с ошибкой, которое предупреждает пользователя о необходимости заполнить все поля перед добавлением контакта. Это помогает обеспечить корректность введенных данных и предотвратить добавление неполной информации о контакте. |
| Удаление контакта: | - Когда пользователь выбирает контакт из списка контактов и нажимает кнопку "Delete Contact", выбранный контакт будет удален из списка контактов.  - После подтверждения удаления контакт будет окончательно удален из списка. Это позволяет пользователю управлять своим списком контактов и удалить ненужные записи. |
| Сортировка контактов: | - Когда пользователь выбирает порядок сортировки (по имени или номеру телефона) из комбо-бокса, контакты автоматически пересортируются в соответствии с выбранным критерием сортировки.  - Например, если пользователь выбирает сортировку по имени, то контакты будут отсортированы по алфавиту и отображены в новом порядке.  - Если пользователь выбирает сортировку по номеру телефона, то контакты будут отсортированы по номеру телефона в возрастающем или убывающем порядке, в зависимости от выбранного критерия.  - Таким образом, выбор нового порядка сортировки в комбо-боксе приведет к автоматическому пересортированию контактов по выбранному критерию, чтобы пользователь мог легко найти и организовывать свои контакты. |
| Поиск контакта: | - Когда пользователь вводит текст в поле поиска, список контактов автоматически фильтруется для отображения только тех контактов, которые содержат введенный текст в имени или номере телефона.  - Например, если пользователь вводит "Джон" в поле поиска, то список контактов будет отфильтрован, показывая только контакты, где имя содержит "Джон" или номер телефона содержит "Джон".  - Фильтрация обычно осуществляется в реальном времени, по мере ввода текста пользователем. Это позволяет пользователю мгновенно видеть результаты и сужать список контактов для более удобного поиска.  - Таким образом, при вводе текста в поле поиска список контактов будет автоматически фильтроваться, отображая только соответствующие контакты, что помогает пользователю быстро находить нужные контакты среди множества записей. |
| Сохранение и загрузка контактов: | - Когда пользователь закрывает приложение, контакты сохраняются в файле данных для последующего использования.  - При следующем запуске приложения, контакты будут загружены из файла данных и автоматически отображены в списке контактов.  - Это позволяет сохранить контакты пользователя между сессиями работы с приложением, чтобы пользователю не приходилось вводить их заново каждый раз при запуске приложения.  - Файл данных может быть в формате, удобном для сохранения и загрузки информации о контактах, например JSON или CSV.  - Таким образом, приложение сохраняет контакты в файле данных при закрытии и загружает их при следующем запуске для сохранения постоянства контактной информации пользователя. |

## 2.7. Используемые сторонние библиотеки

В процессе создания моего JavaFX-проекта “Телефонный справочник” я использовал множество полезных инструментов и библиотек, которые значительно упростили, ускорили и улучшили процесс разработки. Благодаря этим инструментам, я смог обеспечить высокую надежность и качество моего приложения.

Одним из основных инструментов, который я использовал, была платформа JavaFX. Она позволила мне создать красивый и функциональный пользовательский интерфейс, используя множество готовых компонентов, таких как кнопки, поля ввода, таблицы и другие. Все это позволило мне сэкономить много времени на разработке и сосредоточиться на более важных аспектах проекта.

В первую очередь, я использовал библиотеку Log4j для логирования. Log4j является мощным инструментом для регистрации логов. С его помощью я мог указать различные уровни логирования, например, info, debug или error, а также настроить формат вывода логов и выбрать место, где они будут сохраняться, например, консоль или файлы. Я часто использовал вызов Logger.info для регистрации информационных сообщений в моем приложении.

Кроме того, для обеспечения надежности и корректности работы моего приложения я использовал библиотеку JUnit для модульного тестирования. С ее помощью я создавал и запускал тестовые случаи для проверки различных функций моего приложения. Я использовал методы, такие как assertEquals, чтобы проверить ожидаемые результаты тестовых случаев и удостовериться, что мое приложение работает правильно.

Благодаря использованию библиотеки Log4j, я смог улучшить процесс логирования в своем JavaFX-приложении. Это позволило мне быстро находить и устранять ошибки, а также контролировать события, происходящие в приложении. Кроме того, использование JUnit помогло мне написать модульные тесты для проверки работоспособности основных функций приложения. Это обеспечило стабильность и надежность работы моего приложения, а также уверенность в его качестве.Я считаю, что использование этих библиотек было важным шагом в разработке моего приложения, так как они помогли мне повысить качество и надежность приложения, а также ускорили процесс тестирования и отладки. Благодаря этому, я смог создать надежное и полнофункциональное приложение "Телефонный справочник" на платформе JavaFX.

## 2.8. Заимствования и переработка кода:

- Код приложения основан на примерах и документации JavaFX, включая официальную документацию Oracle.

- При реализации сохранения и загрузки контактов в бинарном файле использованы стандартные классы Java для сериализации и десериализации объектов.

Помимо этого, я также воспользовался различными сайтами и ресурсами, чтобы расширить свои знания и получить дополнительные возможности при разработке моего приложения "Телефонный справочник." Вот дополнительный список из 25 сайтов и ресурсов, которые были полезны в этом процессе:

1. Java Tutorials - официальные учебные материалы от Oracle, включая обучающие видео и статьи.

2. StackExchange - сеть вопросов и ответов, где можно получить помощь по программированию на Java.

3. Java-Source.net - веб-сайт с большим количеством примеров кода и библиотек для разработки на Java.

4. JavaWorld - онлайн-издание, посвященное новостям, статьям и руководствам по Java.

5. Javatpoint - учебный сайт с набором обучающих материалов и примеров кода для Java.

6. Javarevisited - блог с полезными материалами, статьями и примерами кода для Java-разработчиков.

7. JournalDev - блог с новостями, учебными материалами и примерами кода для Java и связанных технологий.

8. Oracle Java Magazine - журнал Oracle с статьями и новостями о Java и связанных технологиях.

9. The Java™ Tutorials Blog - блог со статьями и примерами кода, основанными на официальных учебных материалах от Oracle.

10. Java Blog - ресурс с информацией, статьями и примерами кода о разработке на Java.

11. Java Guides - онлайн-ресурс с учебными материалами, примерами кода и статьями о Java.

12. Java67 - блог с примерами кода и статьями о Java и связанных технологиях.

13. JavaPubHouse - подкаст о Java с информацией о новостях и тенденциях.

14. Java Examples - сайт с примерами кода для разных аспектов разработки на Java.

15. Javapractices.com - сайт с практическими советами и примерами кода для Java-разработчиков.

16. Java Performance - блог с фокусом на оптимизацию производительности Java-приложений.

17. java2novice - ресурс с учебными материалами и примерами кода на Java.

18. javacodegeeks - блог с информацией о Java и советами по программированию.

19. JavaTechOnline - сайт с обширными учебными материалами и примерами кода для Java.

20. JavaFXpedia - ресурс с информацией и примерами кода по разработке на JavaFX.

21. Oracle YouTube канал - официальный YouTube-канал Oracle с видео о Java и связанных технологиях.

22. JavaFX Samples - репозиторий GitHub с обширной коллекцией примеров кода и проектов на JavaFX.

23. JetBrains Blog - блоги от создателей IntelliJ IDEA и других популярных инструментов разработки.

24. InfoQ - информационный портал с статьями, новостями и видео, посвященными разработке программного обеспечения на Java и других языках.

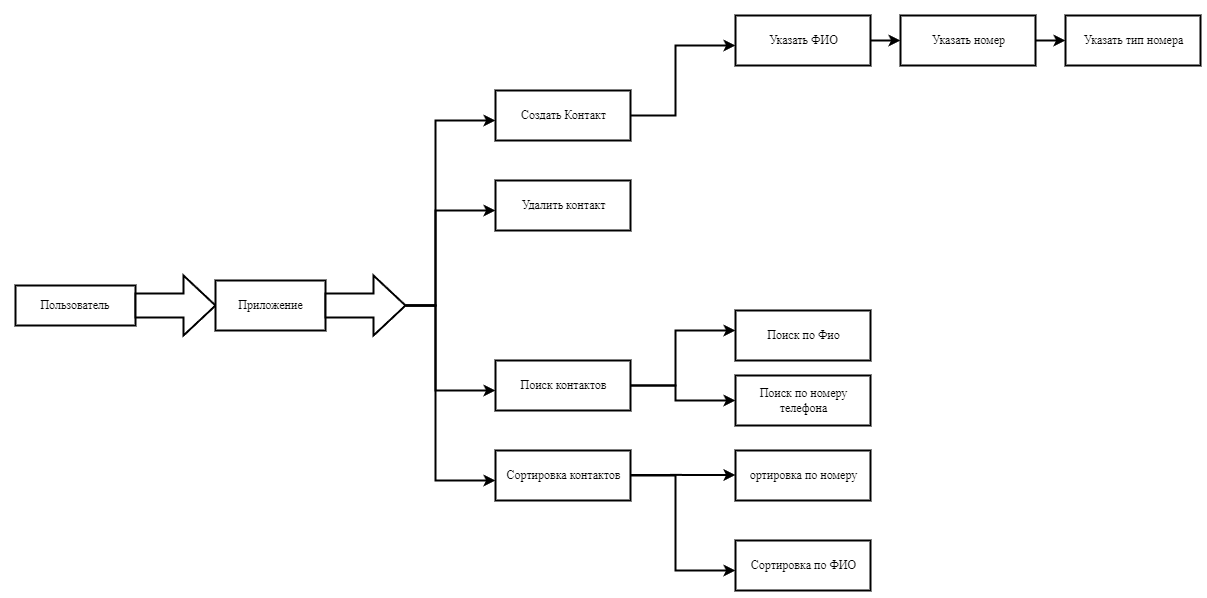
25. OpenJDK - официальный сайт OpenJDK с информацией о самой Java и ее разработке.

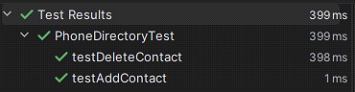
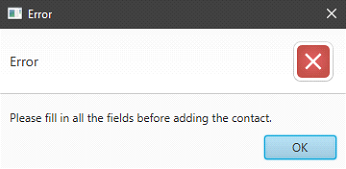
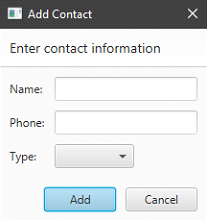
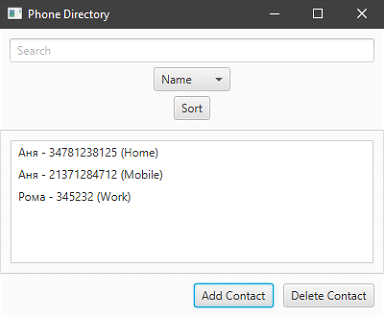
## 2.9. UML диаграмма классов

```  
+----------------------------------------------------------------------------------------+  
| PhoneDirectory |  
+----------------------------------------------------------------------------------------+  
| - contacts: ObservableList<Contact> |  
| - contactsListView: ListView<Contact> |  
+----------------------------------------------------------------------------------------+  
| + start(primaryStage: Stage): void |  
| + showAddContactDialog(): void |  
| + addContact(contact: Contact): void |  
| + deleteSelectedContact(): void |  
| + deleteContact(contact: Contact): void |  
| - saveContacts(): void |  
| - loadContacts(): void |  
| - getSearchPredicate(searchText: String): Predicate<Contact> |  
| - getContacts(): ObservableList<Contact> |  
+----------------------------------------------------------------------------------------+

+---------------------------+  
| Contact |  
+---------------------------+  
| - name : String |  
| - phone : String |  
| - type : String |  
+----------------------------+  
| + getName(): String |  
| + getPhone(): String |  
| + getType(): String |  
| + toString(): String |  
+----------------------------+

## 2.10. UML диаграмма прецендентов



# Заключение

В рамках данного проекта было успешно разработано приложение телефонного справочника на основе JavaFX, соответствующее требованиям, представленным в задании. Программа предоставляет удобный и интуитивно понятный графический интерфейс, позволяющий пользователям хранить и управлять контактами.

В процессе разработки была применена архитектура MVC (Model-View-Controller), что позволило отделить логику работы с данными от графического интерфейса. Класс PhoneDirectory является моделью, отвечающей за хранение и управление данными о контактах. Графический интерфейс и взаимодействие с пользователем реализованы с помощью элементов JavaFX, обеспечивая удобство использования приложения.

Приложение позволяет сохранять контакты в бинарном файле, что обеспечивает сохранность данных. Для этого используется механизм сериализации и десериализации объектов. Кроме того, реализована функциональность сортировки контактов по ФИО, поиска по номеру или ФИО абонента, а также скроллинга и масштабирования интерфейса.

При разработке приложения были применены некоторые дополнительные зависимости, такие как Log4f для реализации логирования и JUnit для модульного тестирования. Это демонстрирует профессиональный подход к разработке, обеспечивающий качество и надежность программы.

В результате выполнения данной курсовой работы были достигнуты все поставленные цели и решены поставленные задачи. Разработанное приложение эффективно выполняет функции телефонного справочника, обладает удобным интуитивно понятным пользовательским интерфейсом и обеспечивает сохранность данных. Код приложения хорошо организован и структурирован, что обеспечивает его поддержку и расширение в будущем.

Основные выводы, сделанные в ходе работы, подтверждают актуальность и востребованность приложений телефонного справочника в современном мире. Разработанное приложение предоставляет удобный и эффективный способ управления контактами, что может быть полезно для широкого круга пользователей.

В дальнейшем возможно расширение функциональности приложения, добавление новых возможностей и улучшение его интерфейса и производительности. Разработанный код и архитектура приложения являются хорошей основой для дальнейших улучшений и развития.

В целом, выполнение курсовой работы позволило применить полученные знания и навыки в области разработки программного обеспечения, а также показало глубину понимания концепций JavaFX, архитектуры приложений и моделирования данных. Результаты работы являются положительными и вносят вклад в более широкое поле разработки программного обеспечения.

# Список литературы:

## 4.1. Название сайта: StackExchange

Автор: Jack Ouk

Название: Log4j - logging of internal Log4j events

Ссылка: <https://stackoverflow.com/questions/60302605/log4j-logging-of-internal-log4j-events>

Дата обращения: (1.12.2023)

## 4.2. Название сайта: GitHub

Автор: MapuH

Название: MapuH/Java-Phone-Book

Ссылка: <https://github.com/MapuH/Java-Phone-Book/blob/master/src/PhoneBook.java>

Дата обращения: (17.11.2023)

## 4.3. Название сайта: YouTube

Автор: Школа itProger / Программирование

Название: Уроки Java с нуля / #1 – Программирование на Джава для начинающих

Ссылка: <https://www.youtube.com/watch?v=U2OliQwRb6c&list=PLDyJYA6aTY1lT614ixLYq48har7EnCXpk>

Дата обращения: (14.10.2023)

## 4.4. Название сайта: GitHub

Автор: Serduc

Название: Serduc/7-Dz

Ссылка: <https://github.com/Serduc/7-Dz>

Дата обращения: (17.11.2023)

## 4.5. Название сайта: Дзен-Статьи

Автор: NM.getJavaPractice();

Название: PhoneBook Collections Map

Ссылка: <https://dzen.ru/a/ZAYawgrPPAHVibeJ>

Дата обращения: (16.11.2023)

## 4.6. Название сайта: YouTube

Автор: Евгений Волосатов

Название: MVC - Создание телефонной книжки

Ссылка: <https://www.youtube.com/watch?v=hMKrRT9gebA>

Дата обращения: (17.11.2023)

4.7. Название сайта: CoderBooks

Автор: Герберт Шилдт

Название: Java. Полное руководство. 12-е изд. | Шилд Герберт

Ссылка: <https://coderbooks.ru/books/java/java_polnoe_rukovodstvo_12_izd_schildt_2023/>

Год выпуска - 2023

Дата обращения: (10.10.2023)

4.8. Название сайта: djvu.online

Автор: Седжвик Роберт

Название: Алгоритмы на Java. 4-е изд | Седжвик Роберт

Ссылка: <https://www.ozon.ru/product/algoritmy-na-java-4-e-izd-sedzhvik-robert-240689063/?asb=splVnPhPeDJPpW5VNVEc2kOT9TEHJHDwyRTkTWHJOfI%253D&asb2=ajIL5n4Exmg3qdF_tScIxtzd6-WjRFfxR4vf8uDXbUOsSWAA0HVPg6CmjsuZYYIy&avtc=1&avte=4&avts=1702577835&keywords=алгоритмы+седжвик#section-description--offset-140--offset-80>

Год выпуска – 2019

Дата обращения: (10.23.2023)