**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 (ИКСС)**

**КАФЕДРА ЗАЩИЩЕННЫХ СИСТЕМ СВЯЗИ (ЗСС)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Курсовая работа по предмету:

«Разработка защищённых сетевых приложений»

на тему:

«Разработка файлового менеджера»

Выполнил студент группы ИБС-23:

Власов Григорий Михайлович

Проверил: Шариков Павел Иванович

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc153327995)

Глава 1 Описание работы программы [4](#_Toc153327995)-6

Глава 2 Описание использованных библиотек…………………………………..7-9

Глава 3 Пример работы программы..…………………………………………..10-32

Глава 4 Блок-Схема алгоритма………………………………………………....33-37

Заключение……………………………………………………………………...38-43

Список литературы....................................................................................................44

**Введение**

В современном информационном обществе, где цифровые технологии проникают во все сферы жизни, безопасность сетевых приложений становится крайне актуальным вопросом. Разработка защищённых сетевых приложений требует глубокого понимания принципов безопасности, эффективных методов обеспечения конфиденциальности и целостности данных, а также грамотного управления доступом к ресурсам. В данном контексте особенно важным становится создание сетевых приложений, которые, помимо высокой производительности и функциональности, обладают надёжной системой защиты.

Цель курсовой работы заключается в разработке файлового менеджера с использованием языка программирования Java. Файловые менеджеры представляют собой неотъемлемую часть операционных систем, предоставляя пользователям средства управления файлами и директориями. В данной работе акцент делается на функциональности и обеспечении эффективного и безопасного управления файловой структурой.

Задание №8 ставит перед разработчиком конкретную задачу, целью которой является реализация функционала копирования выбранного файла в другую директорию. Это требование предполагает техническую реализацию копирования файлов. Кроме того, предусматривается необходимость вывода занимаемого объема каждого файла в папке, что содействует более эффективному управлению ресурсами и оптимизации хранилища данных.

Работа призвана внести свой вклад в область разработки защищённых сетевых приложений и предоставить практический опыт в применении современных подходов к безопасной обработке файлов и данных в сетевых приложениях, реализованных на языке программирования Java.

**Глава 2 Описание работы программы**

Курсовая работа посвящена разработке защищённого сетевого файлового менеджера на языке программирования Java. Подробно рассмотрим каждый этап работы программы, начиная с открытия диалогового окна для выбора исходного файла.

**Шаг 1: Открытие диалогового окна для выбора исходного файла.**

В данном этапе применяется компонент `JFileChooser`, предоставляющий диалоговое окно для удобного выбора исходного файла. Пользователь взаимодействует с этим интерфейсом, выбирая файл, который будет подвергнут копированию. Это важное пользовательское взаимодействие предоставляет удобный способ определения исходных данных для последующих операций.

Открытие диалогового окна для выбора исходного файла в данном проекте реализовано с использованием компонента `JFileChooser`. Этот важный этап взаимодействия с пользователем придаёт программе дополнительный уровень удобства и интуитивной понятности, предоставляя удобный механизм для определения исходных данных перед выполнением операций.

Это визуальное взаимодействие не только обеспечивает легкий доступ к исходным данным, но и способствует созданию пользовательского опыта, который является интуитивно понятным и удовлетворительным. Пользователь может быстро и без труда выбрать файл, который необходим для копирования, что делает процесс управления данными более приятным.

Таким образом, использование `JFileChooser` для открытия диалогового окна выбора исходного файла не только улучшает функциональность программы, но и предоставляет пользователю эффективный и удобный инструмент для взаимодействия с приложением.

**Шаг 2: Вызов конструктора `FolderSize` с выбранным исходным файлом.**

После успешного выбора исходного файла пользователь инициирует вызов конструктора `FolderSize`. Этот конструктор является ключевым компонентом программы, предоставляющим пользователю детальную информацию о размере всех файлов в директории, откуда будет производиться копирование. Этот этап взаимодействия с пользователем предоставляет не только обзор объема данных, предназначенных для копирования, но и ценную информацию для оптимизации использования дискового пространства.

Конструктор `FolderSize` осуществляет сбор информации о размере файлов в выбранной директории. Данный функционал позволяет пользователю визуально оценить, сколько места на диске займут скопированные файлы. Это становится важным аспектом для оптимизации управления диском, поскольку пользователь получает предварительное представление о том, сколько свободного места будет использовано после копирования.

В конечном итоге, вызов конструктора `FolderSize` после выбора исходного файла не только предоставляет важную информацию о размере файлов в директории, но и дополняет пользовательский опыт дополнительными аспектами предварительного анализа перед выполнением операции копирования. Это делает программу более прозрачной и эффективной для пользователя, обеспечивая оптимальное использование ресурсов дискового пространства.

**Шаг 3: Открытие диалогового окна для выбора каталога назначения.**

После получения информации о размере исходной директории программа открывает новое диалоговое окно для выбора директории, в которую будет скопирован файл. Здесь пользователю предоставляется возможность выбрать целевую директорию, а интерфейс программы обеспечивает удобство выбора и понятный пользовательский опыт.

**Шаг 4: Копирование файла.**

Основной функциональный блок программы связан с копированием выбранного файла в указанную директорию. Для этого создается объект `File` для целевого файла в выбранной директории. Далее используются `FileInputStream` и `FileOutputStream` для чтения из и записи в файл соответственно. Создается буфер размером 1024 байта, благодаря чему файл читается блоками в буфер и записывается в новый файл, пока не завершится копирование. Затем закрываются потоки ввода и вывода. Пользователю выводится сообщение об успешном копировании или сообщение об ошибке, если операция не удалась.

**Шаг 5: Отображение информации о файле.**

После завершения операции копирования программа отображает информацию о скопированном файле. Здесь выводится размер файла в байтах, предоставляя пользователю дополнительные сведения о результатах выполненной операции.

**Описание использованных библиотек**

* **java.io:** библиотека java.io предоставляет обширный набор классов и интерфейсов для эффективной работы с потоками ввода и вывода, а также для манипуляции файлами в языке программирования Java. Она является частью стандартной библиотеки Java и широко используется в различных приложениях для обработки данных, сохранения и загрузки файлов, а также для взаимодействия с различными устройствами ввода/вывода.

Одним из ключевых компонентов библиотеки java.io являются классы `FileInputStream` и `FileOutputStream`. `FileInputStream` предоставляет возможность читать данные из файла, а `FileOutputStream` — записывать данные в файл. Эти классы позволяют эффективно осуществлять операции ввода и вывода с использованием файлового ресурса.

Применение `FileInputStream` позволяет создавать поток для чтения данных из файла. Последовательное чтение байтов из файла обеспечивает программе доступ к содержимому файла в удобном для обработки формате. С другой стороны, `FileOutputStream` предоставляет возможность записи данных в файл. Этот процесс осуществляется последовательной записью байтов в файл, что позволяет сохранять информацию для последующего использования.

Благодаря классам `FileInputStream` и `FileOutputStream`, библиотека java.io обеспечивает мощные инструменты для работы с файлами, что делает ее незаменимой при разработке приложений, где требуется обработка файлов и работа с потоками ввода/вывода.

* **javax.swing**: библиотека `javax.swing` предоставляет набор инструментов для создания графического пользовательского интерфейса (GUI) в языке программирования Java. Она является частью более широкого пакета Java GUI (графического интерфейса пользователя) и предоставляет компоненты, которые облегчают создание интерактивных и интуитивно понятных пользовательских интерфейсов. [[3](https://docs.oracle.com/javase/tutorial/uiswing/)]

Одним из ключевых элементов библиотеки `javax.swing`, использованном в описанной программе, является класс `JFileChooser`. Этот класс предоставляет диалоговое окно для выбора файлов и директорий, что делает взаимодействие с файловой системой более удобным и интуитивным для пользователя. `JFileChooser` поддерживает различные режимы, включая выбор файла, выбор директории и другие, что обеспечивает гибкость в использовании в различных сценариях приложения.

Для работы с `JFileChooser` разработчику необходимо создать экземпляр этого класса и использовать его методы для настройки и взаимодействия с диалоговым окном. После выбора файла или директории пользователем, информация о выбранном пути может быть использована в дальнейших операциях программы, таких как копирование файла.

Таким образом, библиотека `javax.swing` играет ключевую роль в создании удобного и интуитивно понятного пользовательского интерфейса, а `JFileChooser` предоставляет удобные средства для взаимодействия с файловой системой, улучшая опыт пользователя при работе с файлами и директориями в приложении. Эти компоненты совместно обеспечивают эффективное и удобное взаимодействие пользователя с приложением, что является важным аспектом в современной разработке программного обеспечения.

**org.apache.log4j:** библиотека "org.apache.log4j" (Log4j) представляет собой инструмент для регистрации событий (логирования) в приложениях на языке программирования Java. Она обеспечивает гибкое и эффективное ведение журнала, позволяя разработчикам настраивать уровни логирования, формат и расположение вывода логов. В программе используется данная библиотека, она позволяет следить за работой программы и записывает её в “protocol” (отчёт), в котором нам предоставляются сведения о пошаговой работе программы с соответствующими таймингами.

log4j.rootLogger=DEBUG, file

# Direct log messages to a log file

log4j.appender.file=org.apache.log4j.RollingFileAppender

log4j.appender.file.File=./Data/Library/protocol.log

log4j.appender.file.MaxFileSize=1MB

log4j.appender.file.MaxBackupIndex=1

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{ISO8601} %-5p %c{1}:%L - %m%n

Пояснение строк:

1. Описывает уровень сообщения DEBUG c символическим названием file, на которое будет выводиться сообщение.

2. Указывает, что сообщение будет выводиться в файл.

3. Задаёт имя и путь файлу для записи.

4. Определяет максимальный размер файла.

5. Максимальное резервное копирование.

6. Описывает формат вывода текста.

7.Вывод данных.

**Глава 3 Пример работы программы**

**Пример 1**

В самом начале имеем папку, в которой находятся различные файлы с разными форматами. Как раз-таки из неё мы и будем выбирать нужный нам файл для копирования, чтобы в дальнейшем скопировать его в другую директорию.

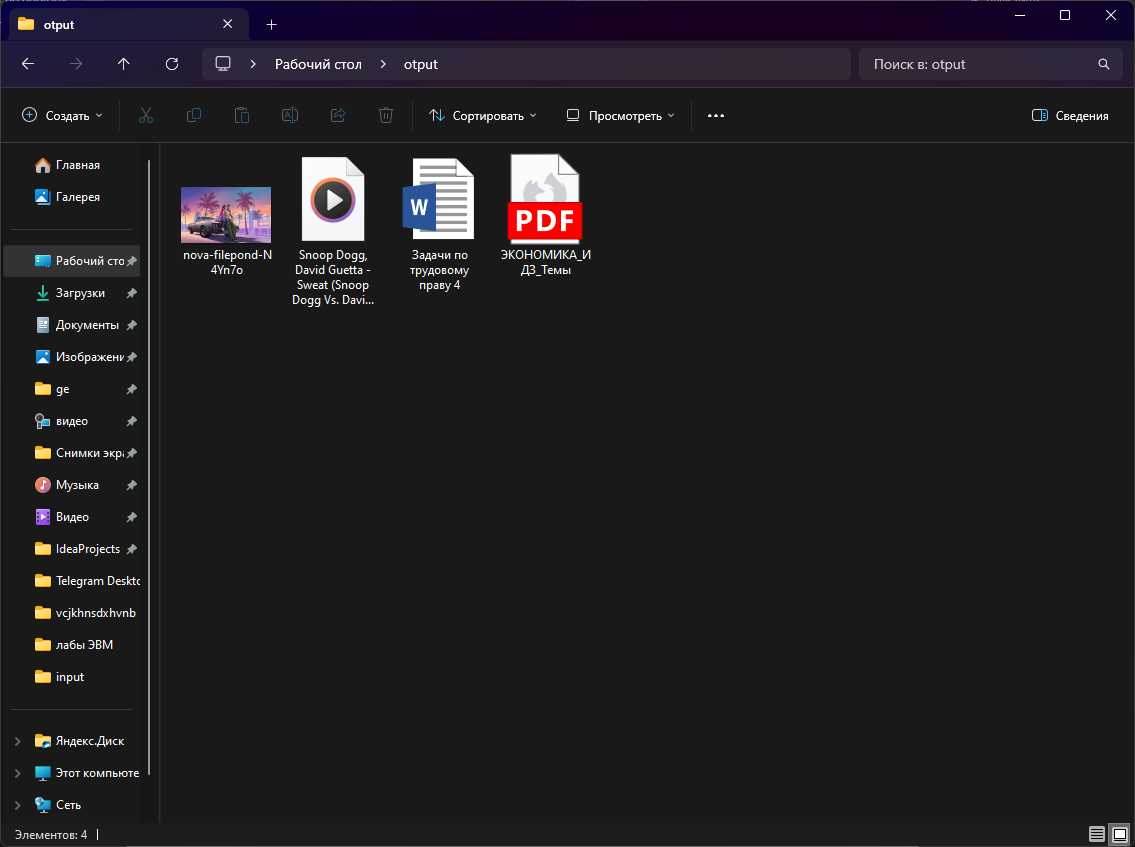


Рисунок 1 – исходная папка с файлами, расположенная по пути «C:\Users\tat-p\Desktop\otput».

Так как нам нужно скопировать файл и перенести его в другую директорию по условию задачи, выберем любую директорию, в данном случае она не содержит других каких-либо файлов. В неё мы и будем сохранять наш скопированный файл в процессе работы программы.

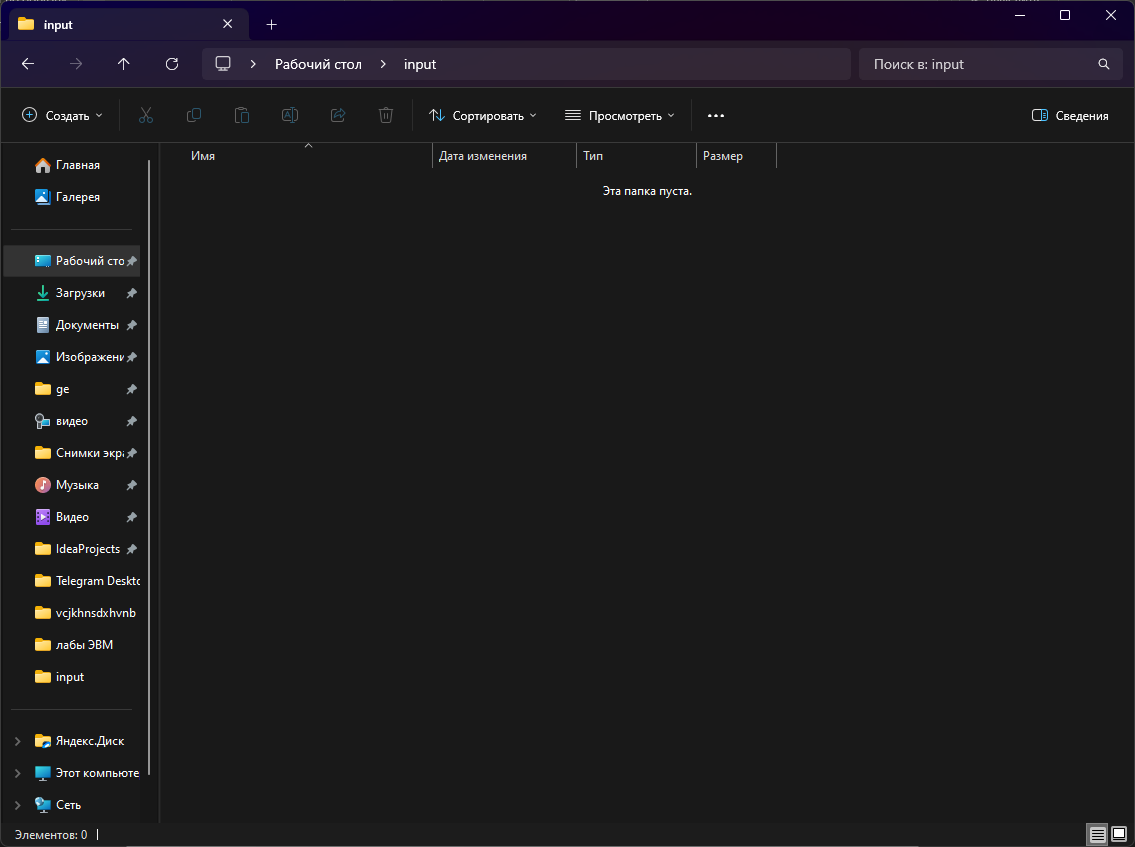


Рисунок 2 – исходная папка с файлами, расположенная по пути «C:\Users\tat-p\Desktop\input».

Как только программа инициируется, пользователь встречается с интуитивно-понятным диалоговым окном, реализованным при помощи библиотеки javax.swing.

Основной функционал начинается с выбора папки, из которой пользователь собирается скопировать файл. Для этого он может использовать интерфейс диалогового окна, который включает в себя стандартные элементы для навигации по файловой системе. Это позволяет легко перемещаться между папками, выбирать нужные директории и предоставлять точные пути к файлам.

После того как пользователь выбрал целевую папку, в которой находится файл для копирования, процесс копирования может начаться. Это действие может выполняться с использованием соответствующих функций библиотеки javax.swing, предоставляющих возможности для копирования файлов между различными директориями.

Важно отметить, что весь этот процесс взаимодействия осуществляется в рамках удобного и интуитивно-понятного пользовательского интерфейса, который обеспечивает легкость навигации, точность выбора файлов и надежность выполнения операций. Этот дизайн интерфейса направлен на обеспечение комфортного пользовательского опыта, где каждый шаг является четким и понятным взаимодействием между пользователем и программой.

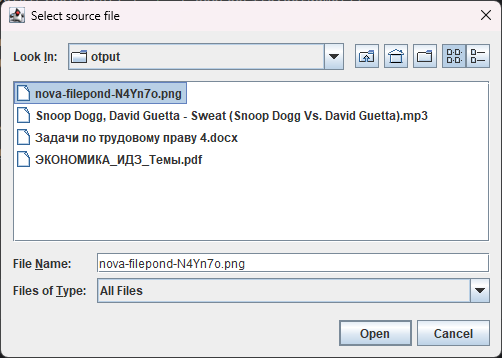


Рисунок 3 – диалоговое окно для выбора исходного файла.

После того, как пользователь определяет целевую директорию для сохранения файла, он взаимодействует с элементами интерфейса для подтверждения своего выбора. Это включает нажатие кнопки "Save".

В результате выполнения операции сохранения, выбранный файл сохраняется в указанной целевой директории, которая была предварительно определена пользователем. Это позволяет пользователю передвигать и организовывать файлы в другие директории на своем компьютере или в сети с помощью простого и удобного интерфейса программы.

Таким образом, после определения целевой директории, пользователь подтверждает свой выбор, нажимая кнопку "Save". Это запускает операцию сохранения, в результате которой выбранный файл копируется в новую директорию, исходя из предпочтений и указаний пользователя.

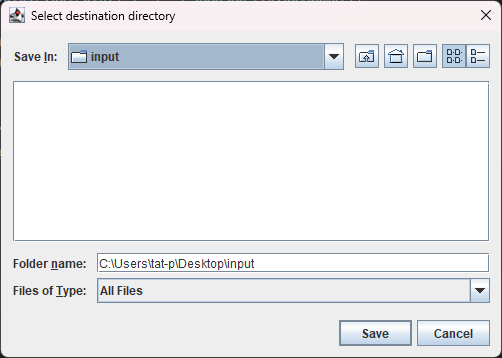


Рисунок 4 – диалоговое окно для выбора директории, в которую будет скопирован файл.

На первом этапе работы программы в терминале отображается текущий путь к директории, где выполняется действие. Это необходимо для обеспечения ясного контекста работы программы, предоставляя пользователю информацию о том, где именно происходят изменения.

Далее, терминал предоставляет полный список файлов, находящихся в указанной директории, сопровождая каждый файл информацией о его размере в байтах. Это предоставляет пользователю полное представление о распределении объема данных в данной директории, что может быть особенно полезным для оценки, какие файлы занимают больше пространства.

После завершения этого этапа, терминал информирует пользователя об успешном копировании файла. Это уведомление играет важную роль, поскольку подтверждает корректное выполнение программой поставленной задачи, что придает уверенность пользователю в надежности работы программы. Такой момент сигнализирует о завершении операции и успешной доставке файла в новое место.

Затем в терминале появляется выбор для пользователя между двумя вариантами: 1 – продолжить программу, 2 – завершить программу.

Таким образом, представленный в терминале отчет не только позволяет отследить каждый этап выполнения программы, но и предоставляет полезные данные о содержимом директории. Общая ясность и прозрачность в подобных операциях обогащают опыт пользователя, делая взаимодействие с программой более интуитивным и понятным.

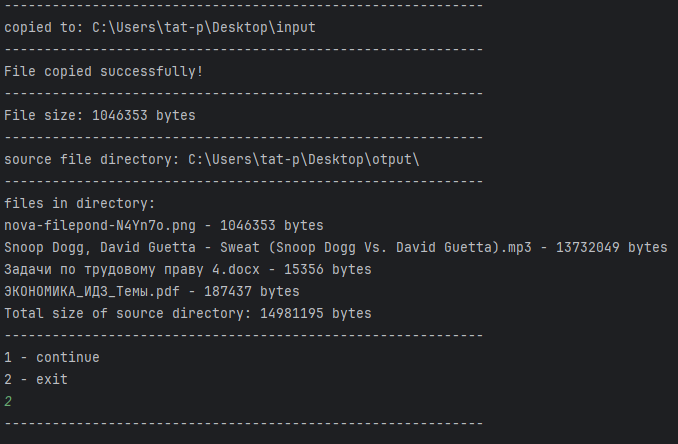


Рисунок 5 – сообщение в терминале.

При запуске программы, автоматически начинается процесс протоколирования, который фиксирует результаты работы программы и записывает их в лог-файл. Данный лог-файл находится в папке проекта и имеет название "protocol". Внутри лог-файла подробно описаны все основные действия и содержится информация о работе программы, такая как время выполнения, приоритет выполняемых действий и другие сопутствующие данные.

Эта информация позволяет вести подробный журнал работы программы, который помогает анализировать и отслеживать различные этапы выполнения программы, их порядок и результаты. Лог-файл демонстрирует последовательность действий программы и включает в себя важную информацию, такую как время начала и окончания каждой операции, приоритет и любую другую сопутствующую информацию, необходимую для точного воссоздания хода работы программы.

Для визуального представления данных, содержащихся в лог-файле, используется рисунок, который наглядно демонстрирует содержание лог-файла и связанные с ним данные.

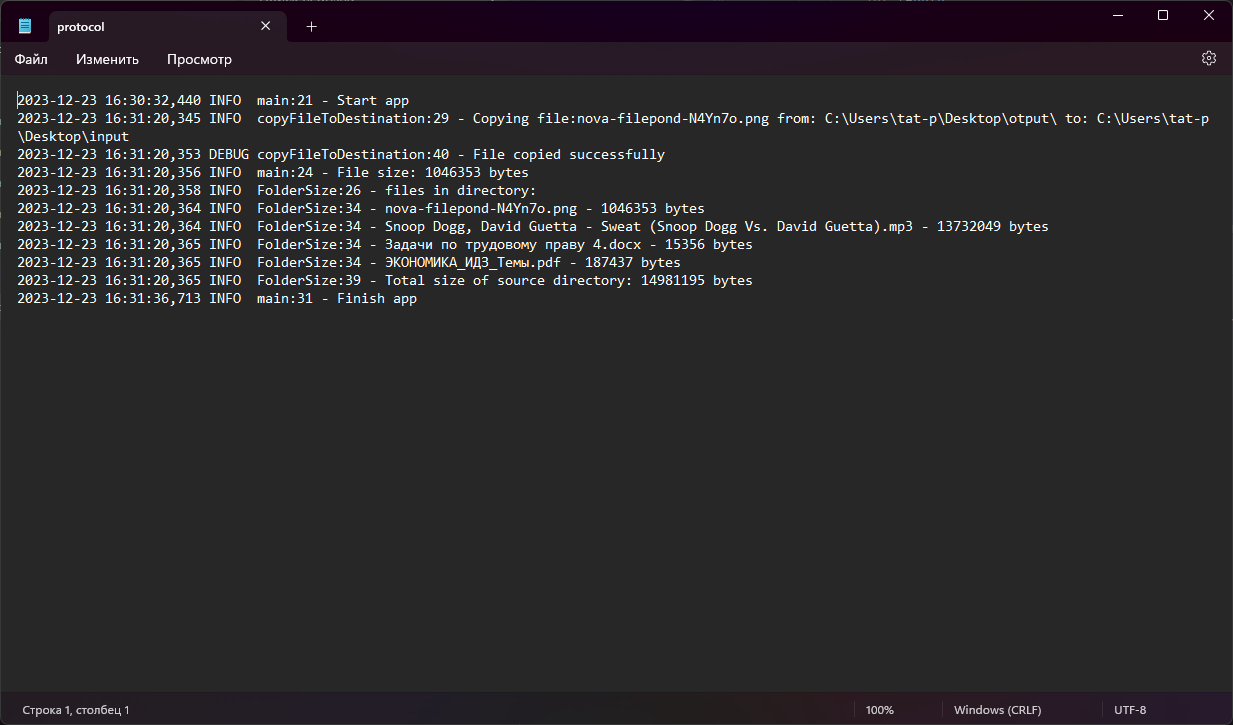


Рисунок 6 – Сообщение автоматического логирования в файле “protocol”

После завершения операции в терминале, где мы отслеживали текущий путь к директории, список файлов и уведомление о успешном копировании, решаем провести дополнительные проверки, чтобы гарантировать точность предоставленных данных.

Для этого переходим в свойства целевой папки, в которую мы копировали файл. В свойствах папки мы обращаем внимание на размер скопированного файла, проверяя его соответствие с тем, что было указано в терминале. Это предоставляет возможность удостовериться, что копирование прошло успешно и без изменений в содержимом.

В итоге, такие проверки обеспечивают более глубокое понимание происходящих в системе изменений и подтверждают успешное выполнение поставленных задач.

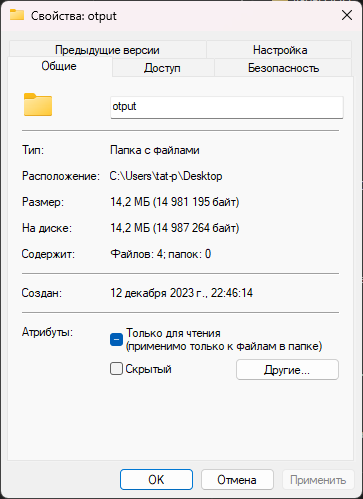


Рисунок 7 – Свойства папки output (можно удостоверится, что терминал вывел размер правильно).

После того, как нажали кнопку “Save” в диалоговом окне, в терминале выводится сообщение о том, что файл скопирован успешно, а также программа выводит размер этого файла.

Точно так же, как и с размером файлов, для проверки, правильно ли сработала программа, заходим в нашу директорию, в которую сохраняли наш выбранный файл и убеждаемся, что он скопировался, а значит программа сработала корректно.

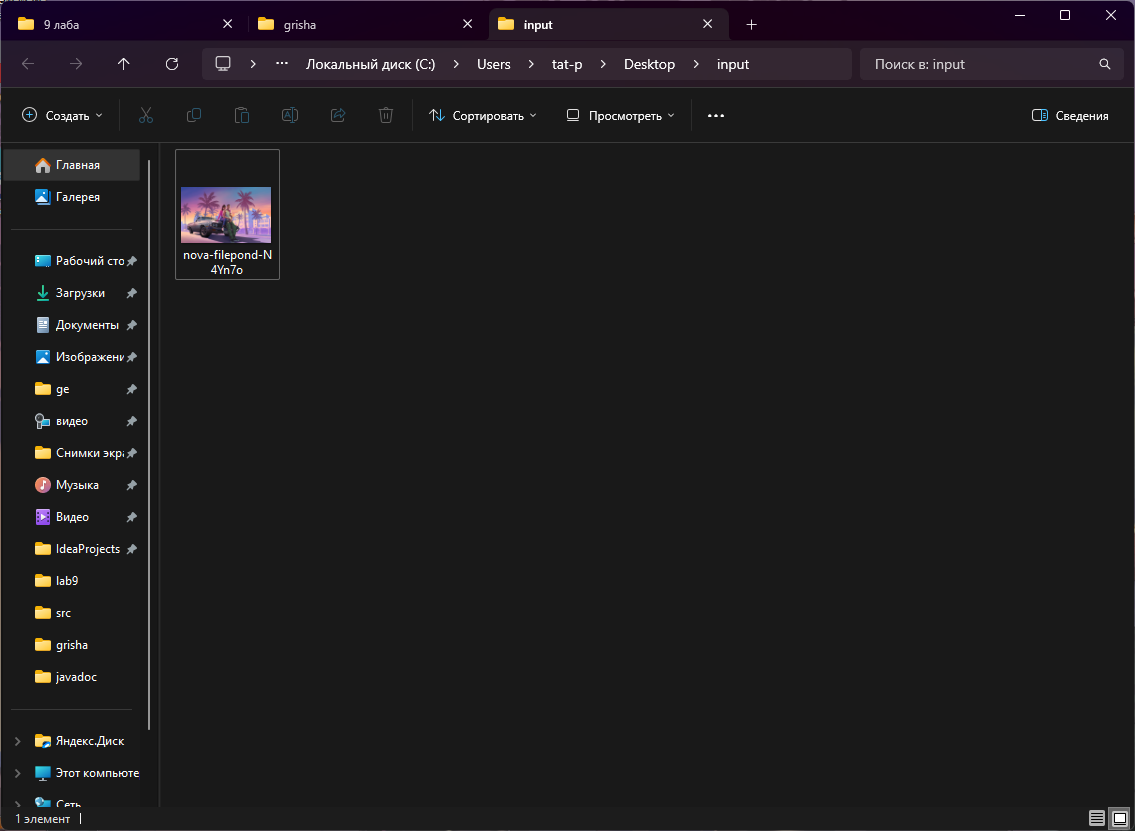


Рисунок 8 – папка input поле выполнения программы.

**Пример 2**

В данном примере попытаемся реализовать копирования нескольких файлов с разными форматами в другую директорию. Этот процесс является важным в контексте управления файловой системой, где корректная обработка подобных сценариев становится неотъемлемой частью эффективного управления данными.

В самом начале имеем папку, в которой находятся различные файлы с разными форматами. Выбираем любой, устроивший нас файл. Далее выбираем целевую директорию, в которую мы хотим скопировать несколько файлов. В результате успешного выполнения операции терминал выведет уведомление, подтверждающее, что копирование завершено успешно.

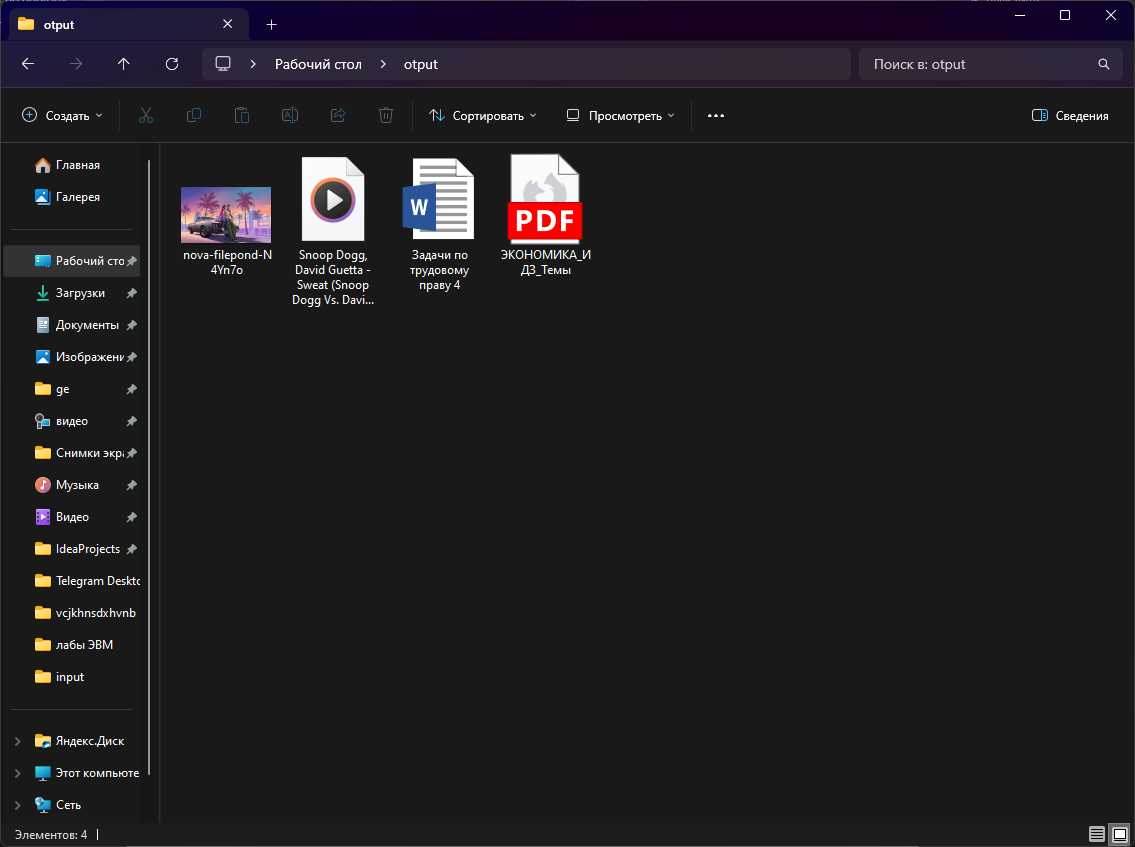


Рисунок 1 – исходная папка с файлами, расположенная по пути «C:\Users\tat-p\Desktop\output».

Для проверки возможностей файлового менеджера, мы хотим скопировать файлы и переместить их в другую директорию. Нашей целью является проверка работы файлового менеджера в циклическом режиме, то есть он не должен завершаться после первого копирования. Кроме того, мы хотим проверить его способность копировать файлы различных форматов.

Первым шагом в этом процессе является выбор подходящей директории для копирования файла. Это означает, что у пользователя есть возможность выбирать любую директорию на компьютере, куда мы будем копировать файлы. В нашем примере мы выбираем папку input в качестве директории, с которой мы будем работать и в которую мы будем копировать файлы.

Таким образом, мы начинаем с выбора папки, где находятся исходные файлы для копирования. Затем мы запускаем процесс копирования, после которого файлы копируются в указанную папку. Важно отметить, что после завершения первого копирования, файловый менеджер не завершает свою работу, а продолжает работать в циклическом режиме, ожидая новых инструкций и файлов для копирования. Это позволяет нам проверить его способность к работе в длительном режиме и выполнению повторных операций копирования.

Кроме того, важно проверить, способен ли файловый менеджер копировать файлы различных форматов, то есть файлы с разными расширениями и типами данных. Это позволяет убедиться в его гибкости и совместимости с различными форматами файлов.

В результате этих шагов, мы сможем проверить работу файлового менеджера на возможность циклического копирования файлов, а также его способность обрабатывать файлы разной структуры и формата. Такая проверка даст нам уверенность в надежности и функциональности данного инструмента.

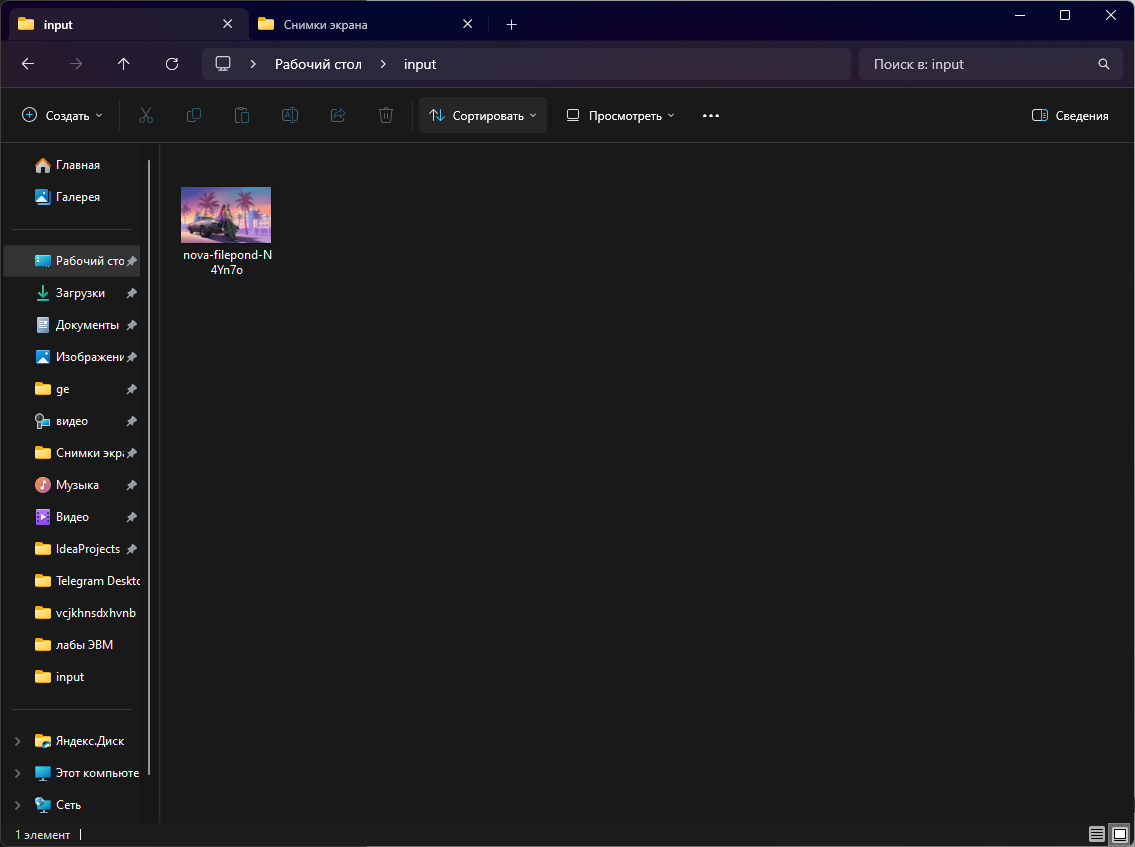


Рисунок 2 – исходная папка с файлами, расположенная по пути «C:\Users\tat-p\Desktop\input».

После запуска программы пользователю предоставляется возможность выбрать любой понравившийся файл для копирования в другую директорию.

После успешного выбора и копирования первого файла, пользователю предлагается ввести цифру "1" для продолжения работы программы. Это указание сигнализирует программе о том, что пользователь хочет продолжить выполнение операций с файлами.

Затем, пользователь может выбрать еще один файл для копирования в другую директорию. Важный момент здесь в том, что выбор файлов происходит поочередно с помощью последовательного выбора файла, после продолжения работы с программой (то есть после ввода цифры "1").

Таким образом, в данном сценарии после запуска программы пользователь выбирает файл для копирования, далее продолжает выполнение программы, вводя цифру "1" и затем выбирает еще один файл для копирования. Важно отметить, что выбор файлов осуществляется последовательно, после каждого ввода цифры "1". Этот подход позволяет пользователю выбирать и копировать файлы в другую директорию по своему выбору и в нужной последовательности, обеспечивая более гибкое и удобное использование программы.

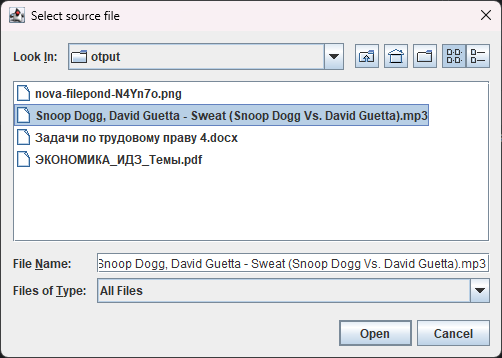
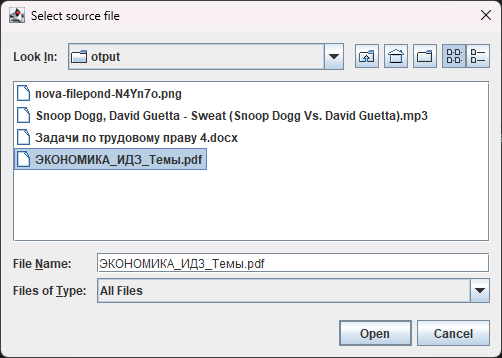


Рисунок 3 и 4 – диалоговые окна для выбора исходных файлов.

На начальном этапе, в терминале отображается текущий путь к директории, где проводится операция. Это необходимо для ясного понимания контекста работы программы.

Следующим шагом терминал предоставляет перечень всех файлов, расположенных в указанной директории. Каждый файл сопровождается информацией о его размере в байтах, что предоставляет полное представление о распределении объема данных в данной директории. Это может быть особенно полезным для пользователя, позволяя ему оценить, какие файлы занимают больше пространства.

После этого важный момент: терминал выводит уведомление о успешном копировании файла. Это подтверждение важно, поскольку оно свидетельствует о том, что программа выполнила свою задачу без сбоев. Этот шаг предоставляет уверенность пользователю в правильной работе программы, а также информирует о завершении операции.

Далее мы пишем цифру 1, чтобы продолжить работу с программой, для дальнейшего копирования ещё файлов в эту же папку. После открывается уже знакомый интерфейс, в котором мы снова выбираем файл для копирования в другую директорию. А после этого пишем 2, чтобы завершить работу программы.

Таким образом, представленный в терминале отчёт не только позволяет отследить каждый этап выполнения программы, но также предоставляет полезные сведения о содержимом директории. Общая ясность и прозрачность в подобных операциях улучшают опыт пользователя, делая взаимодействие с программой более интуитивным и понятным.

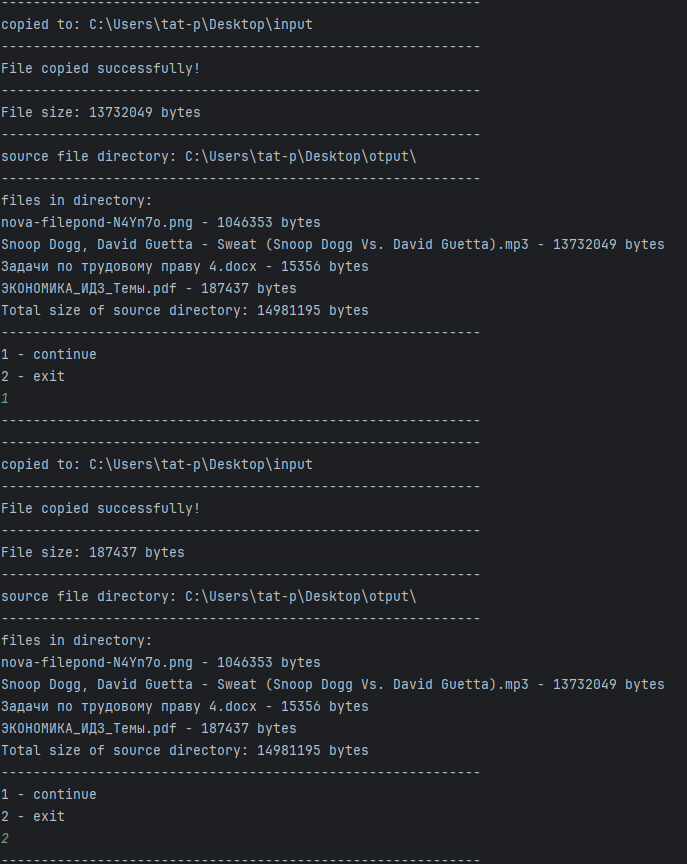


Рисунок 3 – результат работы программы в терминале.

На представленном скриншоте мы можем наблюдать процесс копирования файла в определенную директорию. В результате выполнения операций, программа скопировала несколько файлов в другую директорию. Этот шаг был предпринят с целью обеспечения более эффективного управления файлами.

Важно отметить, что данное действие включает в себя процесс обновления содержимого папки, где новый файл становится актуальной версией, заменяя предыдущий. Такой механизм замены предоставляет возможность удерживать в хранилище только последние версии файлов, что способствует оптимизации использования дискового пространства.

Эта стратегия также способствует обеспечению чистоты данных и избеганию возможных конфликтов, связанных с наличием одинаковых файлов в одной и той же директории. Таким образом, использование данного метода вносит значительный вклад в эффективное управление ресурсами и улучшение общей производительности системы хранения файлов.

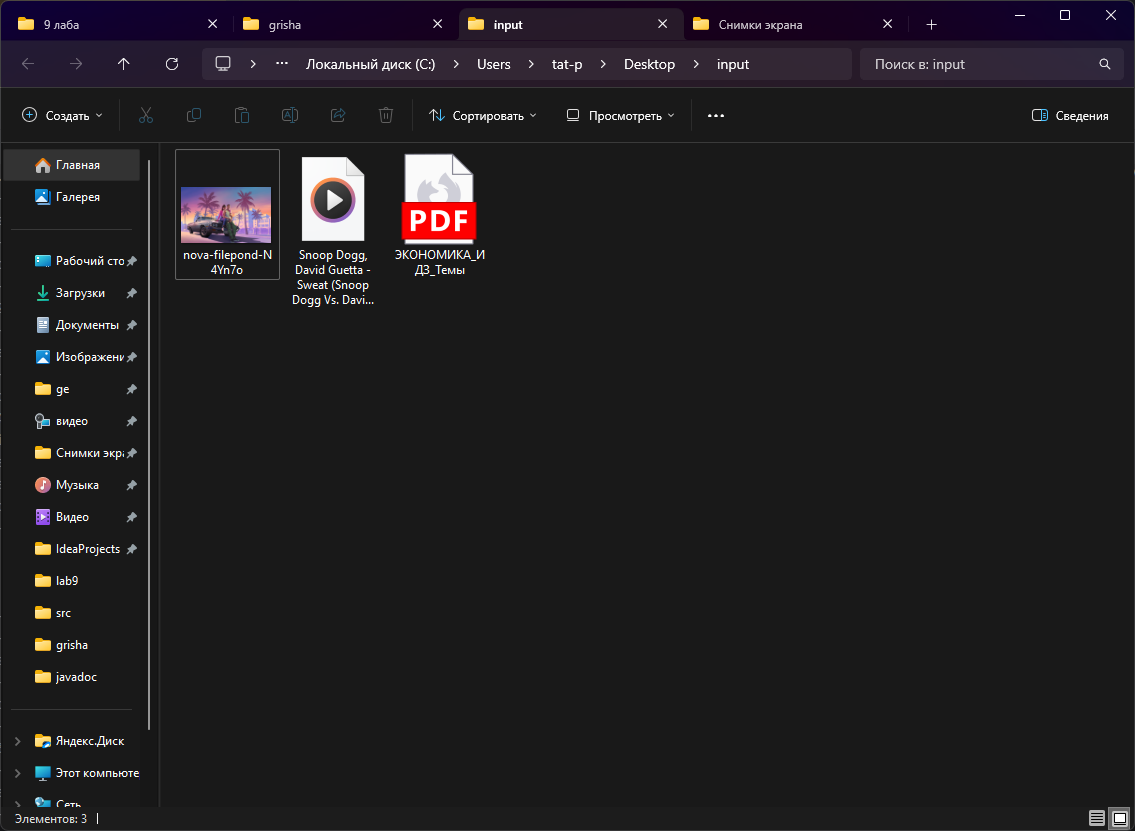


Рисунок 5 – результат работы программы в проводнике.

Так же можно зайти и посмотреть протокол логирования программы.

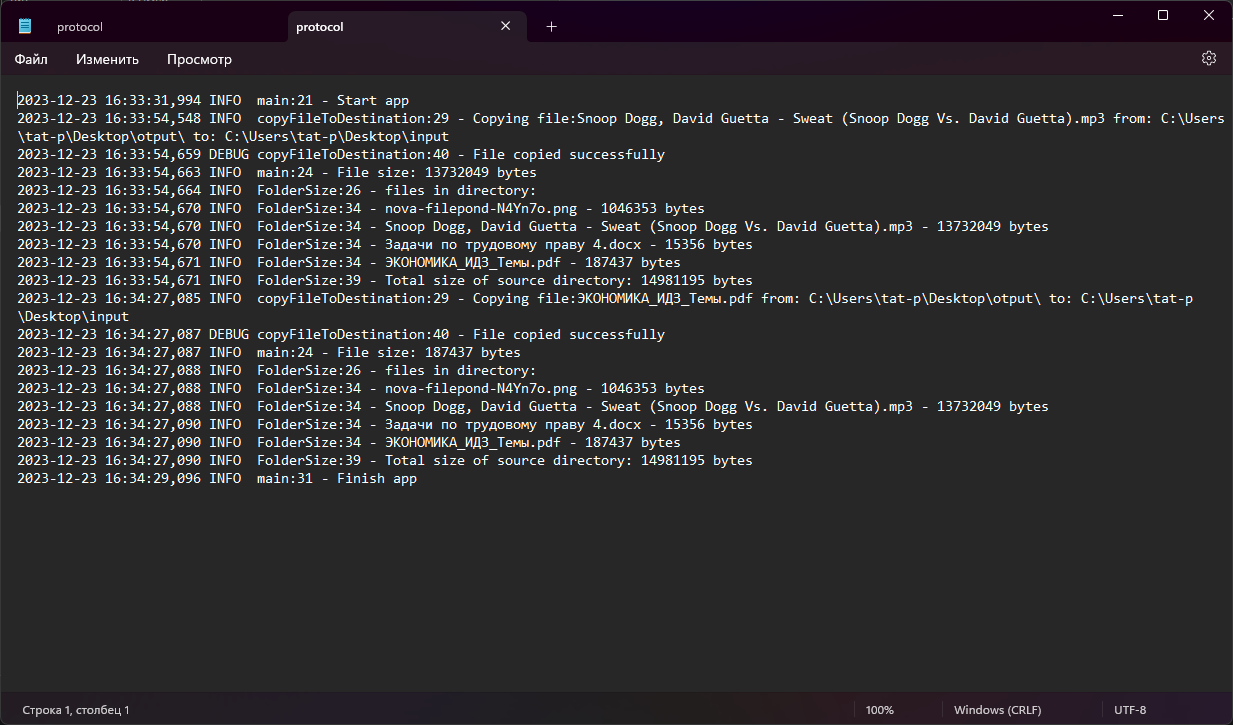


Рисунок 6 - Сообщение автоматического логирования в файле “protocol”.

**Пример 3**

Данный пример представляет случай, когда программа пытается скопировать файл, который не существует. Это может быть результатом ошибочного указания адреса или удаления файла. В такой ситуации программу могут возникнуть проблемы, в зависимости от того, как она обрабатывает ошибки.

Когда программа пытается скопировать несуществующий файл, она может сгенерировать исключение (exception), которое может быть поймано и обработано в программе. Обычно это осуществляется с использованием конструкции try-catch, в которой программа может предпринять определенные действия, такие как вывод сообщения об ошибке или запись информации в журнал (лог).

Если программа хорошо настроена для обработки ошибок и имеет соответствующее логирование, она может сообщить об ошибке при попытке скопировать несуществующий файл и записать эту информацию в журнал. Это позволяет программисту или администратору отслеживать и анализировать проблемы, возникающие в программе, и принимать соответствующие меры для их устранения.

Знание того, как программа будет вести себя в случае неудачного исхода, а также наличие проверки логирования позволяют управлять и анализировать ошибки и события в программе, повышая ее надежность и облегчая отладку и устранение проблем.

После запуска программы, пользователю предоставляется возможность выбрать директорию, из которой он хочет скопировать файл. Далее предоставляется возможность выбрать соответствующий файл или вписать название файла самостоятельно, что мы и сделаем. Однако, такой файл не существует в выбранной директории.

После того, как указан несуществующий файл, пользователь должен выбрать папку назначения, в которую он желает скопировать данный файл.

После выбора папки назначения, пользователю предоставляется возможность нажать на кнопку "Save". Это действие инициирует процесс копирования файла в выбранную папку.

Однако, поскольку указанный файл "11111.png" не существует, операция копирования не может быть выполнена успешно. В результате, должна возникнуть ошибка, которая сообщит пользователю о том, что указанный файл не найден.

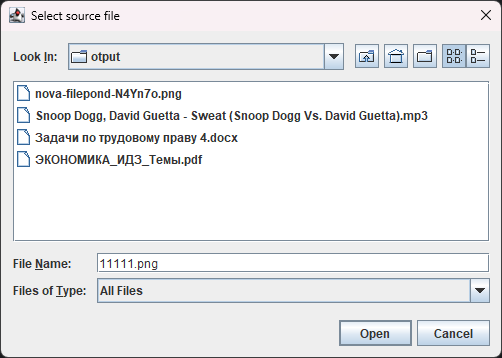


Рисунок 1 – выбор несуществующего файла.

После успешного выполнения операции копирования файла в другую директорию, появится диалоговое окно, которое содержит два варианта выбора: 1 - продолжить работу и 2 - завершить программу. Оно предоставляет пользователю возможность принять решение, как дальше вести работу с программой в связи с окончанием операции копирования.

Одновременно с появлением этого диалогового окна, пользователь увидит ошибку, указывающую на то, что файл, с которым проводилась операция копирования, не был найден.

Для получения более наглядного представления о происходящем можно обратиться к приложенному рисунку снизу, который демонстрирует соответствующую ситуацию с диалоговым окном и ошибкой.

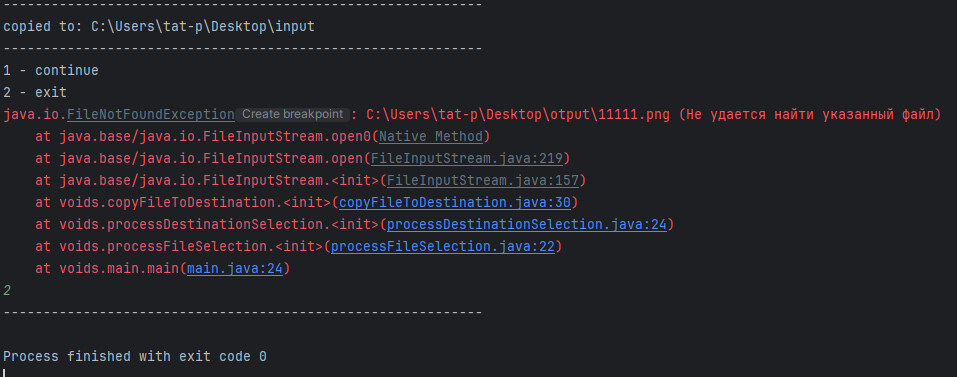


Рисунок 2 – вывод ошибки при копировании файла в терминале.

Когда мы войдем в протокол, откроем его и осуществим просмотр, мы увидим, что процесс регистрации событий и создания логов работает без каких-либо проблем или ошибок, соответствуя ожиданиям и ожидаемому результату. Мы сможем обнаружить, что информация о действиях, происходящих в процессе работы программы, успешно записывается и сохраняется в протоколе, и эти записи отражают актуальные события программы и соответствуют заданным форматам и требованиям.

Мы получим возможность убедиться в корректности работы механизма логирования, который служит ценным инструментом мониторинга и анализа, предоставляя нам доступ к качественной и точной информации о ходе выполнения программы и возможных проблемах или ошибочных событиях, которые могут возникнуть.

В целом, когда мы проведем проверку и изучим данные, содержащиеся в протоколе, мы сможем сделать вывод о том, что логирование работает с высокой эффективностью, надежностью и точностью, предоставляя нам полные и достоверные сведения о работе программы и ее состоянии.

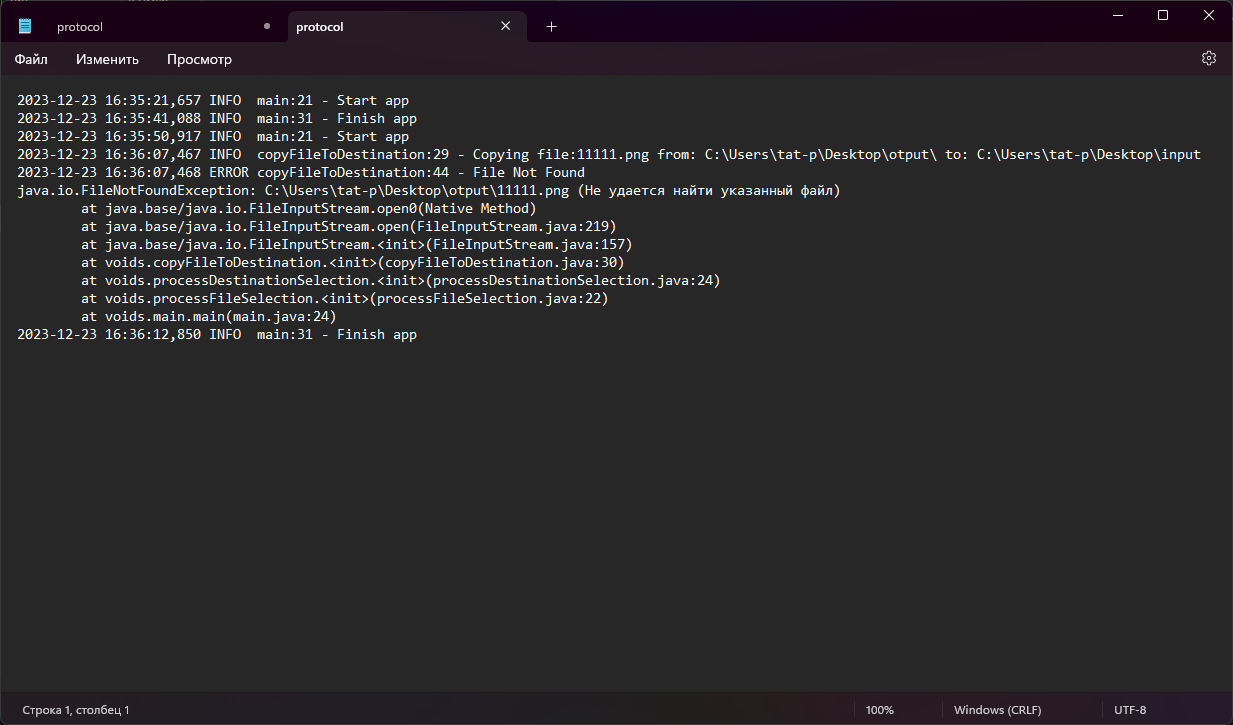
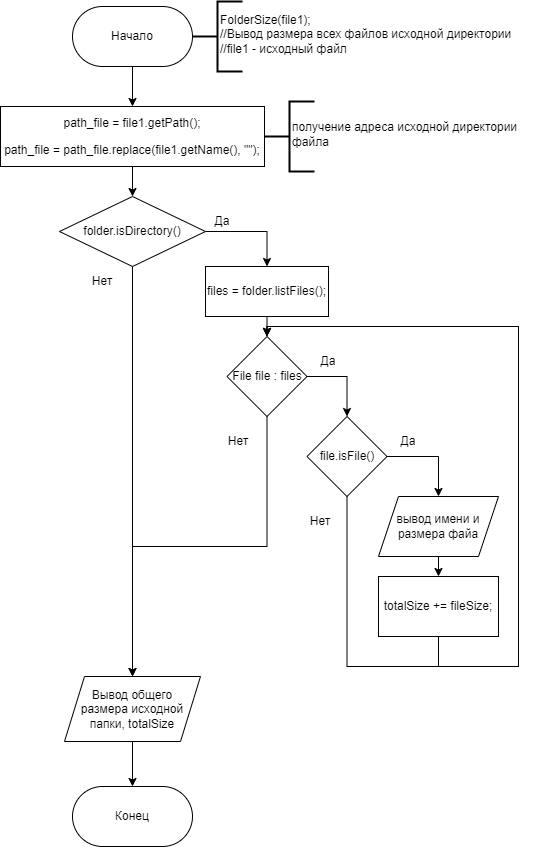
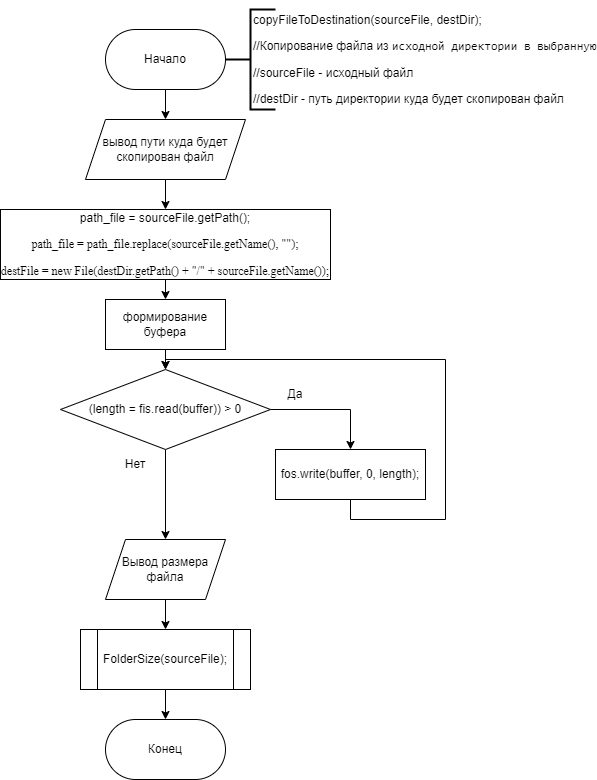
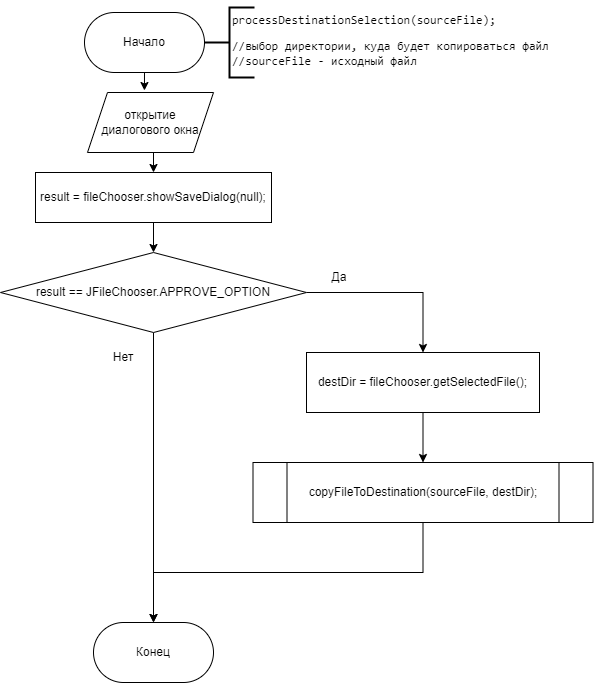
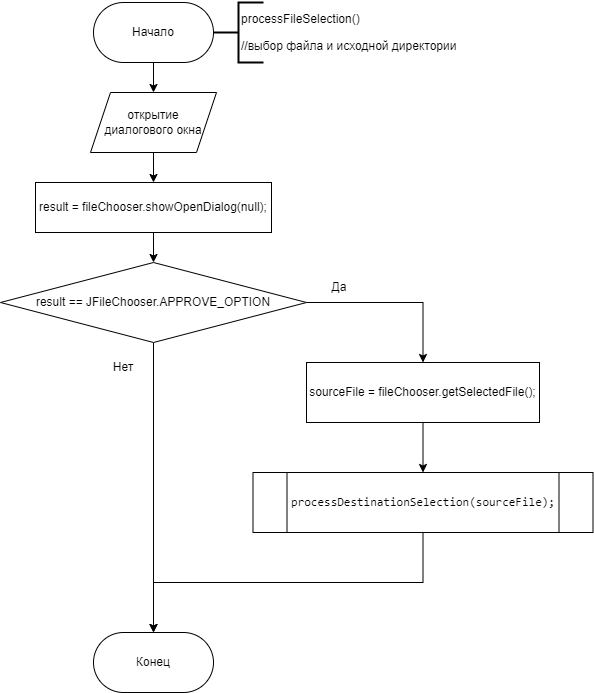
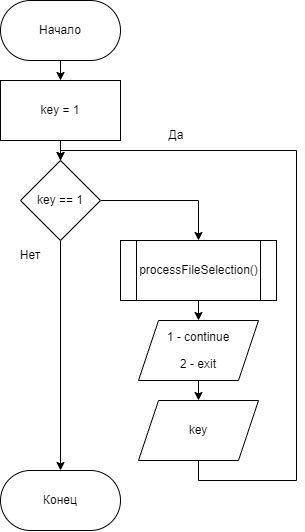


Рисунок 3 - Сообщение автоматического логирования в файле “protocol”.

**Глава 4 Блок-схемы алгоритма**



**Заключение**

В ходе выполнения данной курсовой работы был предпринят всесторонний анализ и успешная реализация файлового менеджера с использованием языка программирования Java и интеграцией графического интерфейса, предоставляемого библиотекой Swing, а также знакомство и внедрение логирования программы при помощи подключенной сторонней библиотеки org.apache.log4j. Знакомство с логированием программы позволяет более эффективно управлять записью информации о событиях, происходящих в программе. Логирование используется для регистрации различных событий, таких как ошибки, предупреждения, информационные сообщения, а также отладочные сообщения. Этот проект охватывает в себе все этапы, начиная от выбора исходного файла и заканчивая отображением подробной информации о скопированном файле. Давайте подробнее рассмотрим ключевые этапы разработки и проанализируем детали, которые сделали проект столь успешным и полноценным.

**1. Использование библиотеки Swing:**

Swing, являясь частью Java API для создания графических пользовательских интерфейсов, предоставляет обширный набор компонентов, которые могут быть легко интегрированы в приложения. Эти компоненты включают в себя кнопки, текстовые поля, диалоговые окна и множество других элементов, создавая возможность для разработчиков создавать интуитивно понятные и привлекательные пользовательские интерфейсы.

Использование Swing позволяет нам добиться не только функциональности, но и эстетической привлекательности программы. Компоненты библиотеки Swing предоставляют не только основные элементы управления, но и многочисленные настройки для их стилизации, анимации и расположения. Такой дизайн интерфейса стремится не только обеспечить удобство использования, но и вызывать положительные эмоции у пользователей.

Привлекательный и интуитивно-понятный пользовательский интерфейс является ключевым фактором в общем восприятии программы. Визуальная составляющая, созданная с использованием Swing, может существенно повысить привлекательность программы и уровень комфорта для пользователя, делая его опыт более приятным и эффективным.

Таким образом, первый этап, связанный с использованием Swing, не только дает нам инструменты для построения функционального интерфейса, но и открывает двери для создания приятного и визуально привлекательного пользовательского восприятия программы.

**2. Обработка файлов:**

Процесс обработки файлов является краеугольным камнем функциональности программы, предоставляя пользователю инструменты для выбора исходного файла и указания целевой директории. В этом контексте используется мощный инструмент `JFileChooser`, который превращает этот процесс в интуитивно-понятное и удобное взаимодействие.

`JFileChooser` является ключевым компонентом библиотеки Swing, обеспечивающим графическое окно для выбора файлов и директорий. Его использование не только упрощает пользовательский интерфейс, но также обеспечивает высокую гибкость при работе с различными типами файлов на компьютере пользователя. Пользователю предоставляется возможность легко навигировать по файловой системе, выбирать нужные файлы и директории, всё это в интуитивно-понятном графическом окне.

Этот механизм выбора файлов делает программу максимально адаптивной, поскольку пользователь может взаимодействовать с его удобным и знакомым интерфейсом. Благодаря `JFileChooser` дополнительные возможности предоставляются в виде фильтрации файлов, просмотра дополнительной информации о файлах и многого другого. Это содействует универсальности приложения, делая его пригодным для разнообразных задач и потребностей пользователей.

Важно отметить, что использование `JFileChooser` не только обеспечивает удобство в выборе файлов, но и подчеркивает стремление программы к созданию интуитивно-понятного и приятного пользовательского интерфейса. Такой подход делает взаимодействие с программой более естественным и увлекательным, что важно для обеспечения положительного пользовательского опыта.

Таким образом, механизм обработки файлов в программе, основанный на использовании `JFileChooser`, предоставляет не только функциональность, но и высокий уровень удобства и гибкости для пользователей, делая приложение более привлекательным и универсальным.

**3. Копирование файла:**

Механизм копирования файла, реализованный в данной программе, представляет собой тщательно разработанную систему, использующую потоки ввода/вывода `FileInputStream` и `FileOutputStream`. Это стратегическое решение обеспечивает эффективное копирование даже для крупных файлов, разбивая их на блоки данных и оперируя этими блоками. Такой подход снижает риск переполнения памяти и в то же время оптимизирует использование ресурсов, делая процесс копирования эффективным и безопасным.

Использование потоков ввода/вывода в Java позволяет читать и записывать данные порциями, что особенно важно при работе с большими файлами. `FileInputStream` позволяет создать поток для чтения данных из исходного файла, а `FileOutputStream` создает поток для записи данных в новый файл. Это дает возможность программе читать и записывать данные блоками, избегая загрузки всего файла в память и эффективно управляя ресурсами системы.

Такая тщательная проработка механизма копирования не только обеспечивает безопасность и надежность операции, но и гарантирует оптимальную производительность в различных условиях. В частности, при работе с крупными файлами, эффективное использование потоков ввода/вывода минимизирует время выполнения операции копирования, что является важным критерием для пользователя.

Таким образом, механизм копирования файла в данной программе не только предоставляет функциональность, но и стремится к оптимальной производительности и эффективному использованию ресурсов системы, что делает его универсальным и надежным инструментом для обработки файлов.

**4. Обработка ошибок:**

Система обработки ошибок в данной программе представляет собой тщательно разработанный механизм, использующий механизм обработки исключений `IOException`. Этот подход гарантирует, что при возникновении проблем при чтении или записи файлов, программа обеспечит не только корректную реакцию на сбои, но и предоставит пользователям информативные сообщения об ошибках.

`IOException` является классом исключений, который охватывает различные ошибки ввода/вывода, такие как невозможность доступа к файлу, проблемы с чтением или записью данных и другие. Применение этого механизма позволяет программе предвидеть возможные проблемы и взаимодействовать с ними, вмешиваясь в поток выполнения программы и обеспечивая корректное завершение операции.

Детальная обработка ошибок включает в себя вывод информативных сообщений, которые не только уведомляют пользователя о возникшей проблеме, но и предоставляют понятные подсказки по ее устранению. Такой подход укрепляет устойчивость приложения, предоставляя пользователям возможность взаимодействовать с программой даже в случае непредвиденных ситуаций.

Важно отметить, что использование механизма обработки исключений в данной программе не только обеспечивает надежность в работе с файлами, но и подчеркивает фокус на качестве пользовательского опыта. Детальное и информативное сообщение об ошибке помогает пользователям легче понимать, что произошло, и принимать необходимые меры для решения проблемы.

Таким образом, механизм обработки ошибок в программе создает условия для высококачественного взаимодействия пользователя с приложением, гарантируя стабильность и надежность при работе с файлами.

**5. Информационный вывод:**

Дополнительной функциональностью программы стало отображение информации о скопированном файле, включая его размер. Этот аспект взаимодействия с пользователем предоставляет обратную связь о выполненной операции, что важно для создания полноценного и информативного пользовательского опыта.

Важно отметить, что вывод информации о размере скопированного файла дополняет операцию копирования, предоставляя пользователю дополнительный контекст о проделанной работе. Пользователь может легко оценить, насколько успешно и быстро прошла операция, а также контролировать использование дискового пространства. Это создает условия для более активного взаимодействия пользователя с программой и повышает удовлетворенность от использования.

В ходе выполнения работы мы успешно решили технические задачи, связанные с графическим интерфейсом, потоками ввода/вывода и обработкой ошибок. Разработанный код проекта представляет собой не только функциональное решение задачи файлового копирования, но и эффективное приложение, обогащенное графическим интерфейсом. Этот проект может служить не только отправной точкой для разработки более сложных файловых утилит, но и прекрасным материалом для изучения принципов построения графических приложений на платформе Java.

Проект не ограничивается простым выполнением задачи копирования файлов, но предоставляет также пример графического интерфейса, который может служить отправной точкой для разработки более сложных файловых утилит. Также, код проекта представляет ценный материал для изучения принципов построения графических приложений на платформе Java, что делает его полезным ресурсом для обучения.

В итоге данная курсовая работа успешно исследовала и реализовала файловый копировщик, предоставив глубокий анализ и подробный обзор различных аспектов его разработки. Полученный опыт и знания могут служить основой для дальнейших исследований в области разработки программного обеспечения и расширения функционала файловых утилит на языке Java.

**Список источников**

1. Объектно-ориентированное программирование на языке Java: Методические указания к лабораторным работам / Сост.: С. А. Беляев, М. Г. Павловский, Г. В. Разумовский; Под общ. ред. Г. В. Разумовского. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2013. 63 с. (дата обращения: 12.12.2023)
2. Java-online: сайт. – URL: <https://java-online.ru/libs-swing.xhtml> (дата обращения: 12.12.2023)
3. Oracle Java: сайт. – URL: <https://docs.oracle.com/javase/tutorial/uiswing/> (дата обращения: 12.12.2023)
4. Stackoverflow: сайт. – URL: <https://stackoverflow.com/questions/1146153/copying-files-from-one-directory-to-another-in-java> (дата обращения: 12.12.2023)