МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Ижевский государственный технический университет
имени М. Т. Калашникова»
Факультет «Информатика и вычислительная техника»
Кафедра «Программное обеспечение»

Отчет по лабораторной работе №1
«Изучение мандатной модели доступа»
по дисциплине «Защита информации»

Выполнил:
студент группы Б17-191-2 Иванов В.М.

Принял:
канд. техн. Наук Ардашев Д.В.
Кафедра «Защита информации в
компьютеризированных системах» - доцент
Кафедра «ПО» - доцент

Ижевск 2021

СОДЕРЖАНИЕ

ОСНОВНАЯ ЧАСТЬ

Цель работы

Получить практические навыки применения мандатной модели доступа.

Поставленная задача

Разработать консольное приложение (или приложение с графическим интерфейсом), которое будет обладать следующей функциональностью:

1. Хранит перечень пользователей (имя, пароль) — субъектов;
2. Хранит список файлов и каталогов (id, путь, имя, тип и id каталога (для файлов)) — объектов;
3. Хранит решётку уровней безопасности и матрицу доступа;
4. Выполняет авторизацию пользователя (по имени и паролю);
5. Реализует сеанс пользователя, в котором доступны команды: просмотр каталога; создание, удаление, копирование файла;
6. Контролирует возможность выполнения команд на основании данных решетки безопасности и матрицы доступа по правилам модели Белла-ЛаПадулы;
7. Реализует административный интерфейс для настройки системы (может быть реализован в форме конфигурационного файла).

Ход работы

Для решения задачи был использован язык программирования Java с модулем Swing для генерации формы. Организация хранения данных показана на диаграмме классов (рис. 1). Согласно этой схеме, заполненные матрица и решетка доступа хранятся в экземпляре класса «Checker», который используется для проверки прав на выполнение отдельной операции. Названия сущностей, в большинстве, совпадают с их назначением. «FileObjectFM» представляет объект типа файл, «Catalog» - директорию в которой хранятся файлы и т. п.

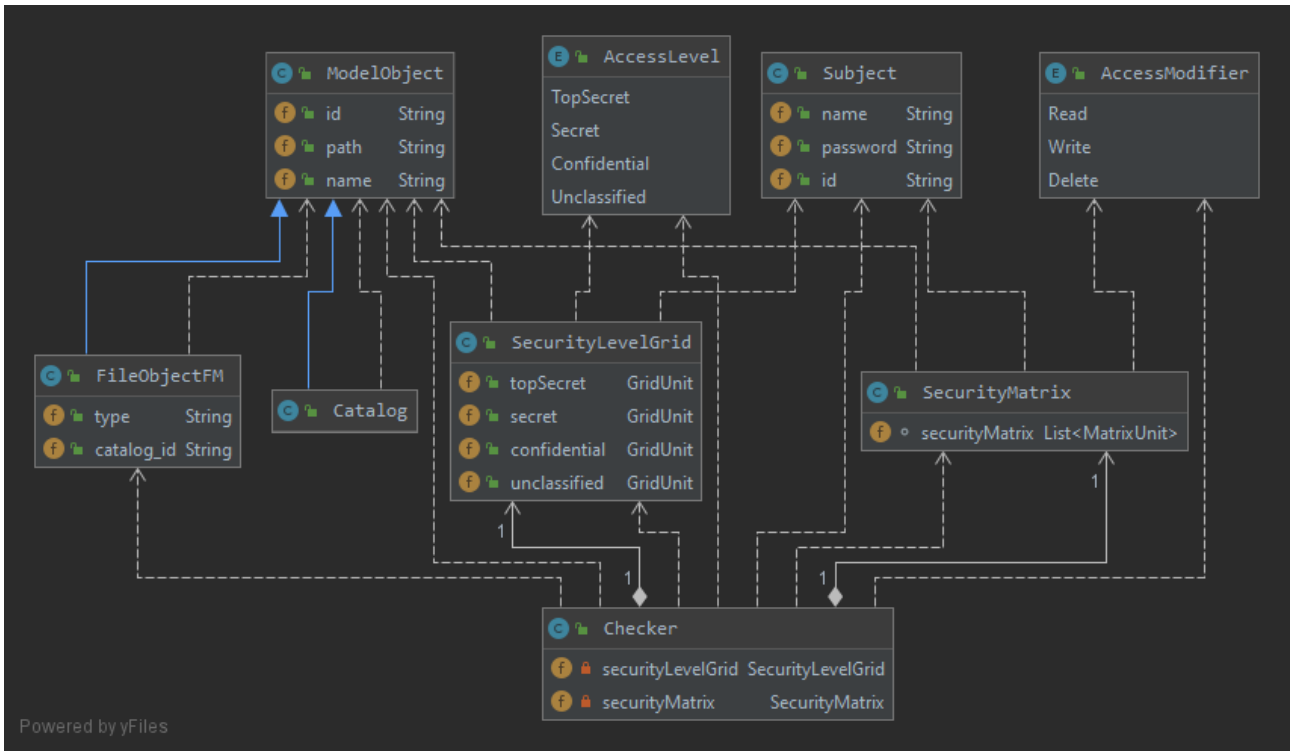

Рис.1. Диаграмма классов

В данной модели набор прав ограничен тремя: read (чтение), и write (запись), delete (удаление). При этом контроль доступа осуществляется в зависимости от уровней безопасности взаимодействующих сторон на основании двух простых правил.

1. Уполномоченное лицо (субъект) имеет право читать только те документы, уровень безопасности которых не превышает его собственный уровень безопасности. Данное правило обеспечивает защиту информации, обрабатываемой более доверенными (высокоуровневыми) лицами, от доступа со стороны менее доверенных (низкоуровневых).

2. Уполномоченное лицо (субъект) имеет право заносить информацию только в те документы, уровень безопасности которых не ниже его собственного уровня безопасности. Это правило предотвращает утечку информации (сознательную или несознательную) со стороны высокоуровневых участников процесса обработки информации к низкоуровневым.

Исходный код реализующий это правило:

```
List<AccessModifier> checkPrivilege(Subject s, ModelObject o) {
```

```
    List<AccessModifier> safeActions = new ArrayList<>();
    int cmp = securityLevelGrid.getSecurityLevel(s).compareTo(
securityLevelGrid.getSecurityLevel(o)
);
    if (cmp == 0) {
        safeActions.add(AccessModifier.Read);
        safeActions.add(AccessModifier.Write);
        safeActions.add(AccessModifier.Delete);
        return safeActions;
    }
    if (cmp <= 0) {
        safeActions.add(AccessModifier.Read);
        safeActions.add(AccessModifier.Delete);
    } else {
        safeActions.add(AccessModifier.Write);
    }
    return safeActions;
}
```

В переменную cmp заносится значение в соответсвтии с привелегиями доступа субъекта и объекта:

- 0 — если права доступа совпадают
- 1 — если права доступа у объекта выше чем у субъекта
- -1 — если права доступа у субъекта выше чем у объекта

В зависимости от результата, запрошенному действию задаются безопасные действия и если запрошенное действие входит в их перечень, то оно будет выполнено.

Класс “SecurityLevelGrid” служит для хранения и поиска прав доступа. Исходный код:

```
public class SecurityLevelGrid {
    public GridUnit topSecret = new GridUnit(AccessLevel.TopSecret);
    public GridUnit secret = new GridUnit(AccessLevel.Secret);
    public GridUnit confidential = new GridUnit(AccessLevel.Confidential);
    public GridUnit unclassified = new GridUnit(AccessLevel.Unclassified);

    public AccessLevel getSecurityLevel(ModelObject o) {
        if (topSecret.findObject(o)) return AccessLevel.TopSecret;
        if (secret.findObject(o)) return AccessLevel.Secret;
        if (confidential.findObject(o)) return AccessLevel.Confidential;
        if (unclassified.findObject(o)) return AccessLevel.Unclassified;
        return null;
```

```
    }

    public AccessLevel getSecurityLevel(Subject s) {
        if (topSecret.findSubject(s)) return AccessLevel.TopSecret;
        if (secret.findSubject(s)) return AccessLevel.Secret;
        if (confidential.findSubject(s)) return AccessLevel.Confidential;
        if (unclassified.findSubject(s)) return AccessLevel.Unclassified;
        return null;
    }
}
```

Класс «GridUnit» представляет ячейку в матрице доступа. Исходный код:

```
public class GridUnit {

    private AccessLevel accessLevel;
    private List<ModelObject> modelObjects;
    private List<Subject> subjects;

    public GridUnit(AccessLevel accessLevel) {
        this.accessLevel = accessLevel;
        this.modelObjects = new ArrayList<ModelObject>();
        this.subjects = new ArrayList<Subject>();
    }
    public GridUnit(AccessLevel accessLevel, List<ModelObject>
modelObjects, List<Subject> subjects) {
        this.accessLevel = accessLevel;
        this.modelObjects = modelObjects;
        this.subjects = subjects;
    }
    public boolean addSubject(Subject s) {
        return subjects.add(s);
    }
    public boolean addObject(ModelObject o) {
        return modelObjects.add(o);
    }
    public boolean findSubject(Subject s) {
        return subjects.contains(s);
    }
    public boolean findObject(ModelObject o) {
        return modelObjects.contains(o);
    }
}
```

Класс «Checker» хранит в себе матрицу (Security Matrix) и решетку доступа (SecurityLevelGrid). Метод “CheckAction” возвращает true в случае возможности доступа субъекта к объекту и false в противном случае, а также выводит окно с сообщением об ошибке.

Исходный код:

```
public class Checker {
    private SecurityLevelGrid securityLevelGrid;
    private SecurityMatrix securityMatrix;

    public void addNewFileToModel(FileObjectFM file, Subject s) {
        securityMatrix.addRecord(file, s, List.of(AccessModifier.Write,
AccessModifier.Read, AccessModifier.Delete));
        switch (securityLevelGrid.getSecurityLevel(s)) {
            case Secret:
                securityLevelGrid.secret.addObject(file);
                break;
            case TopSecret:
                securityLevelGrid.topSecret.addObject(file);
                break;
            case Confidential:
                securityLevelGrid.confidential.addObject(file);
                break;
            case Unclassified:
                securityLevelGrid.unclassified.addObject(file);
                break;
        }
    }

<...>

    public boolean checkAction(Subject s, ModelObject o, AccessModifier
requiredAction) {
        List<AccessModifier> safeActions = checkPrivilege(s, o);
        String message = "Нет права доступа на уровне решетки";
        for (AccessModifier am : safeActions) {
            if (am == requiredAction) {
                List<AccessModifier> availableActions =
securityMatrix.getAccessModifiers(s, o);
                if (availableActions.size() == 0){
                    message = "Нет права доступа на уровне матрицы";
                    JOptionPane.showMessageDialog(null, message);
                    return false;
                }
                for (AccessModifier ao : availableActions) {
                    if (ao == requiredAction) {
                        return true;
                    } else {
                        message = "Нет права доступа на уровне
матрицы";
                    }
                }
```

```
            }
        }
        JOptionPane.showMessageDialog(null, message);
        return false;
    }
}
```

Контрольные примеры

Определим решетку уровней безопасности и матрицу доступа для нашей системы:

Решетка уровней безопасности

Уровень безопасности	Субъекты	Объекты
Совершенно секретно	S_1, S_2	O_5
Секретно	S_3	O_1, O_2
Конфиденциально	S_4	O_4
Несекретно	S_5, S_6	O_3, O_6

Матрица доступа

	O_1	O_2	O_3	O_4	O_5	O_6
S_1	RWD	RD	RD	RD	RWD	RD
S_2	R	R	R	R	RW	R
S_3	RW	RW	R	R	W	R
S_4	W	W	R	D	W	R
S_5	W	W	RW	W	W	RW
S_6	W	W	RW	W	W	RW

Исходные данные, такие как описание объектов, субъектов, матрицы и решетки доступа хранятся в файле «matrix.csv», его содержание представлено в приложении 1.Сущности описываются в соответствии с полями классов (рис.

1), после запуска приложения этот файл обрабатывается, на основе чего создаются экземпляры объектов. Этот файл может быть использован для выставлении настроек перед запуском приложения.

На рисунке 2 показана форма приложения после его запуска. На нем видны основные функциональные блоки и элементы.

Логин осуществляется посредством ввода логина и пароля в поля с названиями «Имя» и «Пароль». Если данный пользователь найден, то кнопка становится зеленой, в противном случае красной (рис. 3, 4).

Далее пользователь может выбрать каталог для отображения файлов. Для этого он выбирает один из пунктов в селекте, под кнопкой войти. Результат зависит от прав доступа на чтение данного каталога. На рисунке 5 показан пример успешного отображения первого каталога.

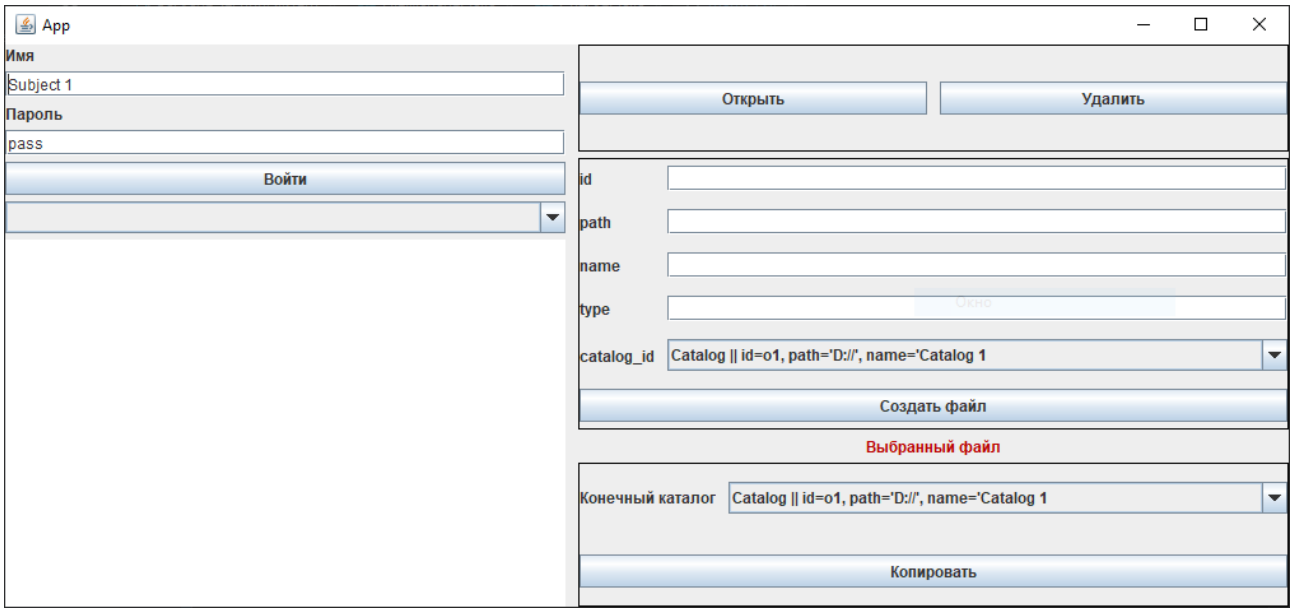

Рис. 2. Главная форма приложения

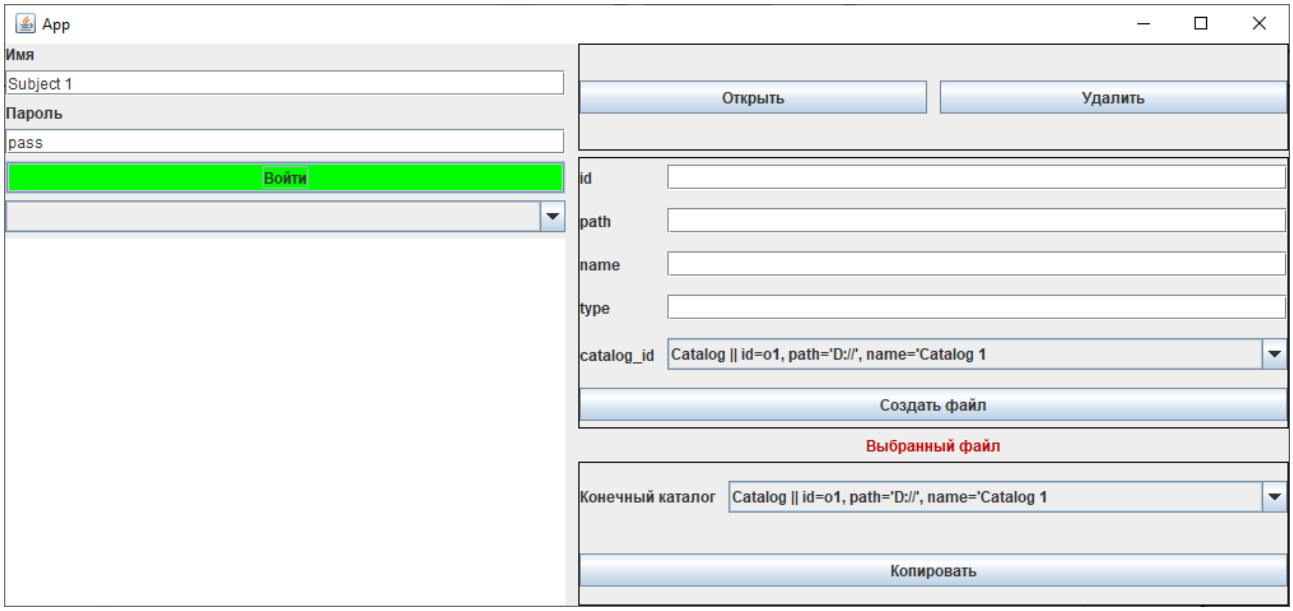

Рис.3. Успешный вход

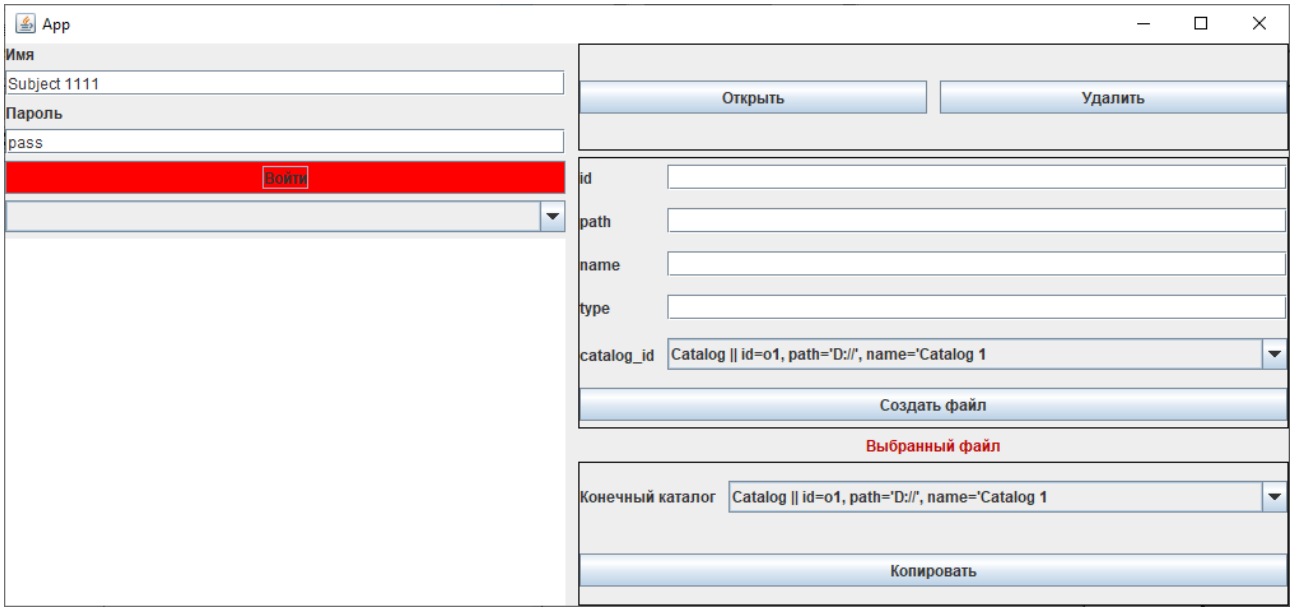

Рис.4.Ошибка введенных данных

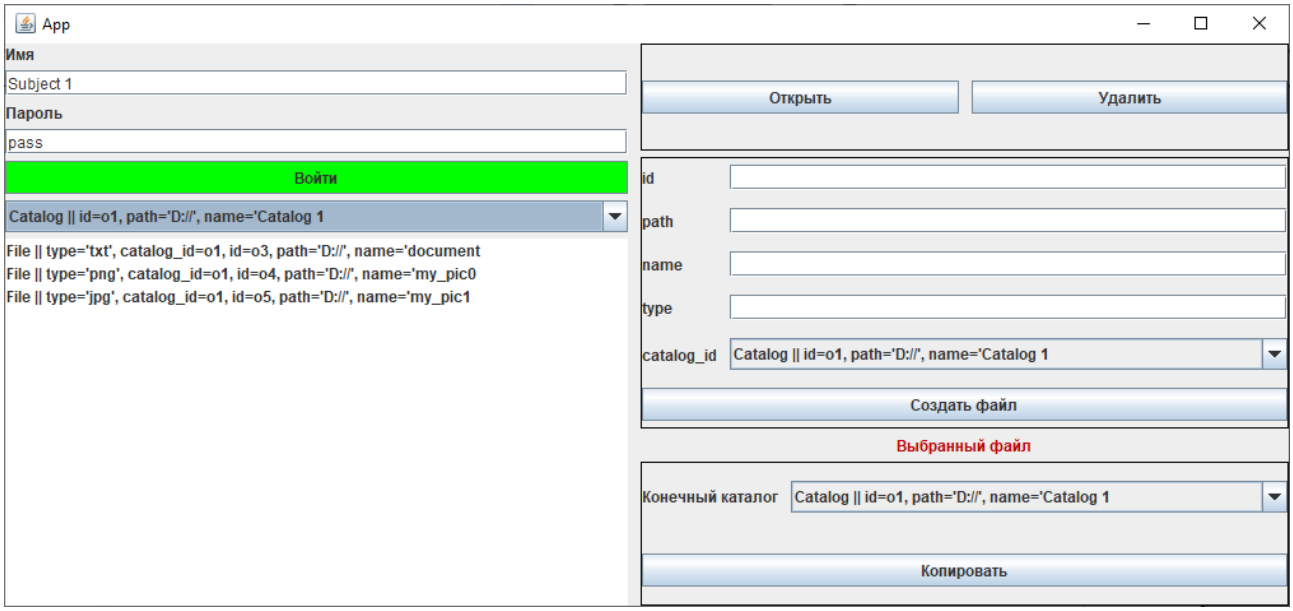

Рис. 5. Чтение файлов из каталога

Далее пользователь может открыть один из файлов, в результате он может получить один из трех исходов:

- Файл успешно открыт, его информация отображена (рис. 6)
- Отказ доступа на уровне решетки (рис. 7)
- Отказ доступа на уровне матрицы (рис. 8)

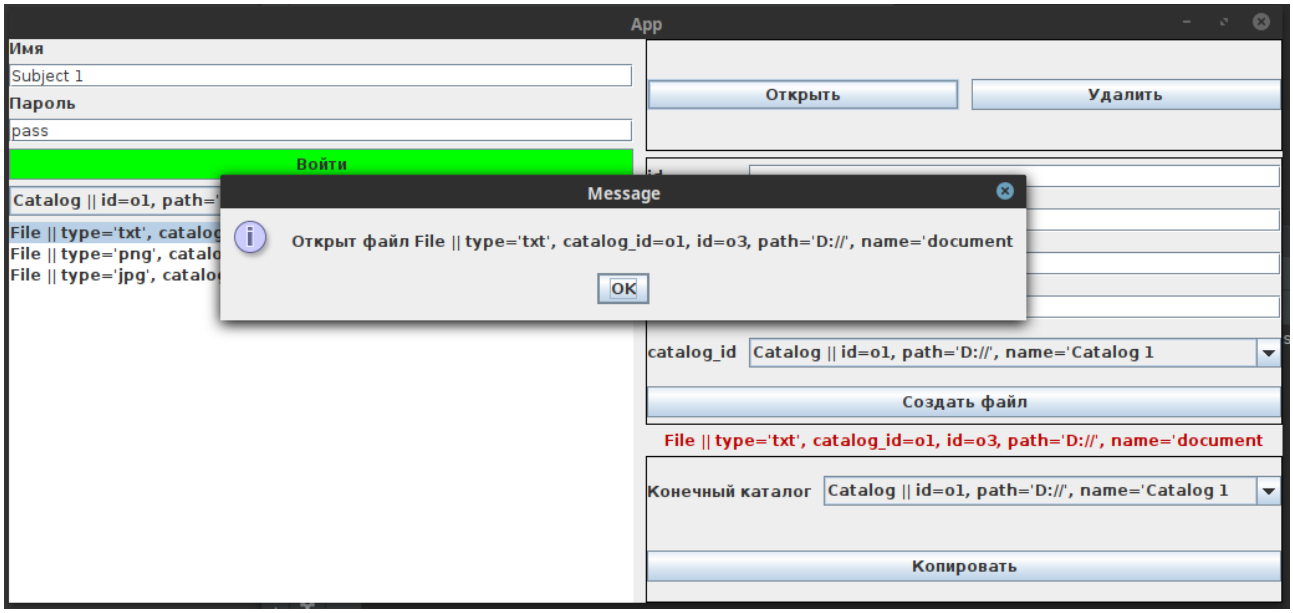

Рис.6. Файл успешно считан

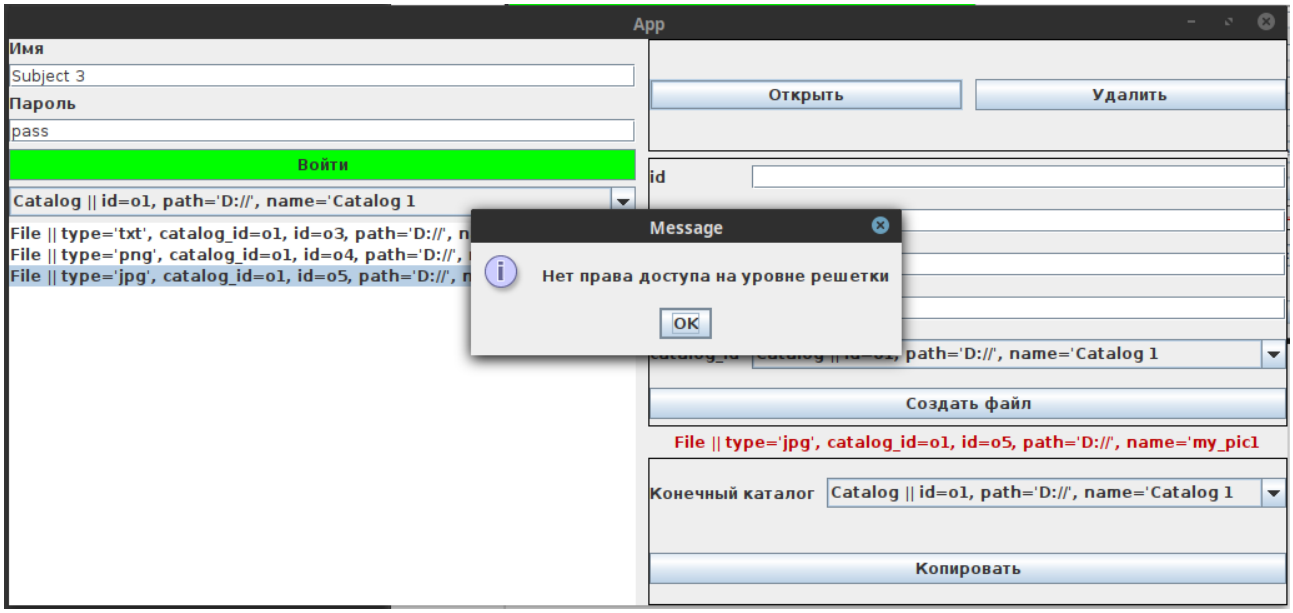

Рис.7. Ошибка чтения файла 1

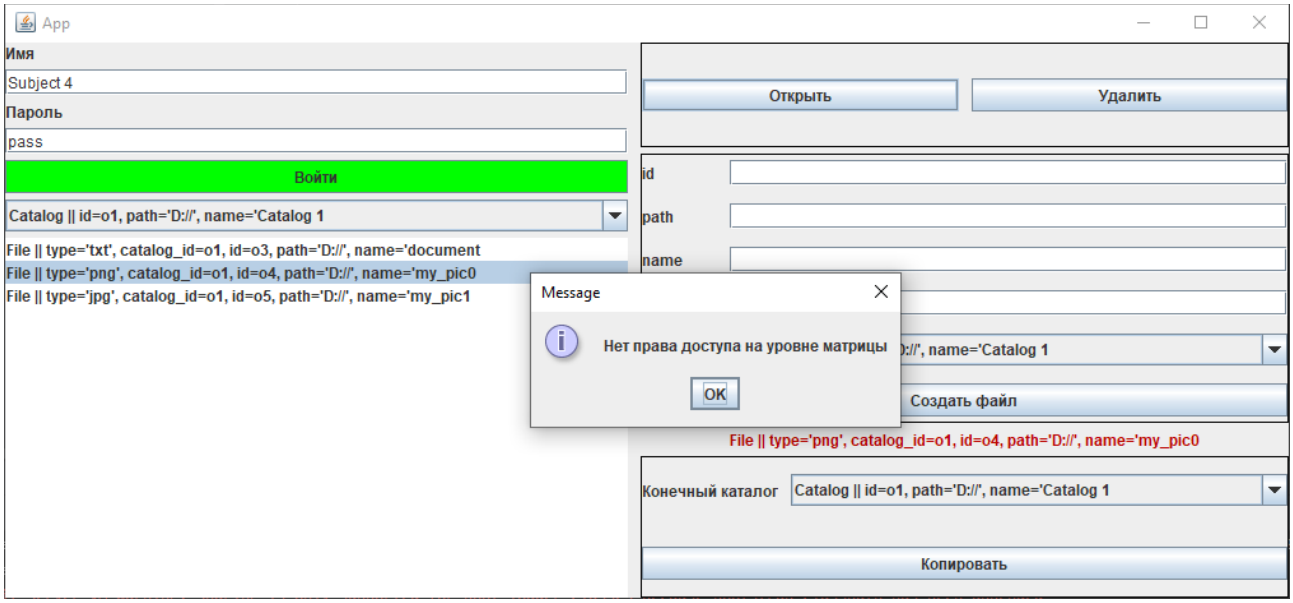

Рис.8. Ошибка чтения файла 2

Для создания файла пользователь заполняет поля формы с кнопкой «Создать файл», затем нажимает на нее. В результате новый файл появится в выбранном каталоге (рис. 9), либо будет выведена ошибка как при чтении файла (рис. 7, 8).

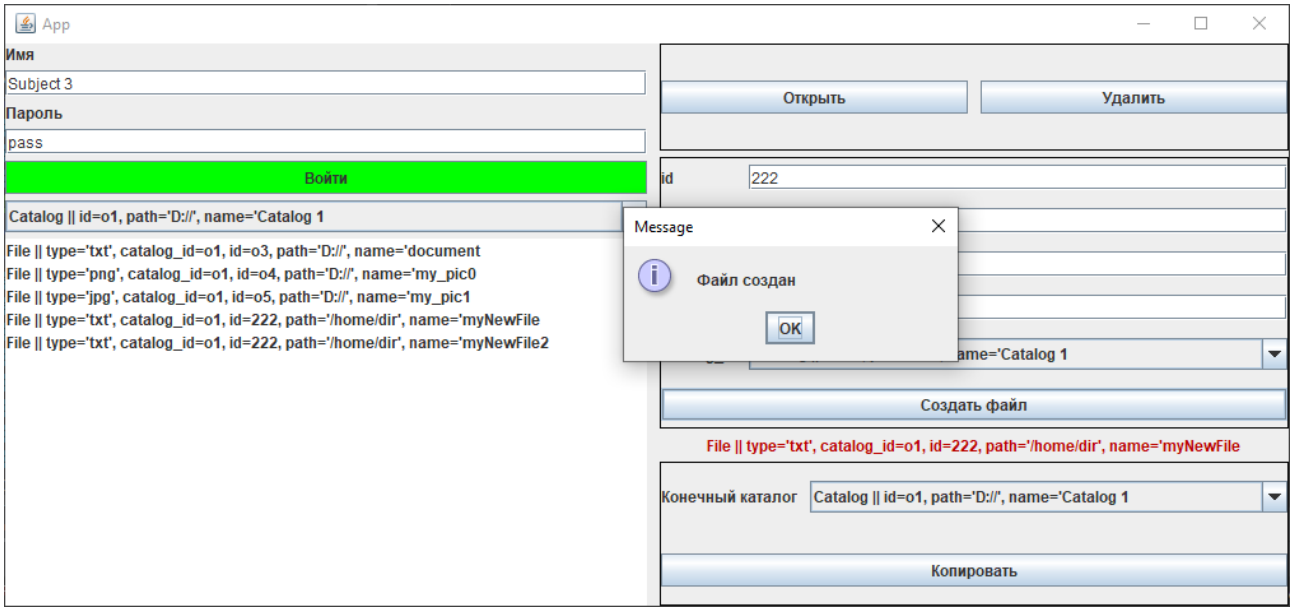

Рис. 9. Успешное создание файла

Для удаление файла пользователь выбирает его из списка щелчком мыши и нажимает кнопку «Удалить». В результате он получает сообщение о успехе или неудаче операции, в случае успеха файл будет удален из списка и условной файловой системы (Рис. 10).

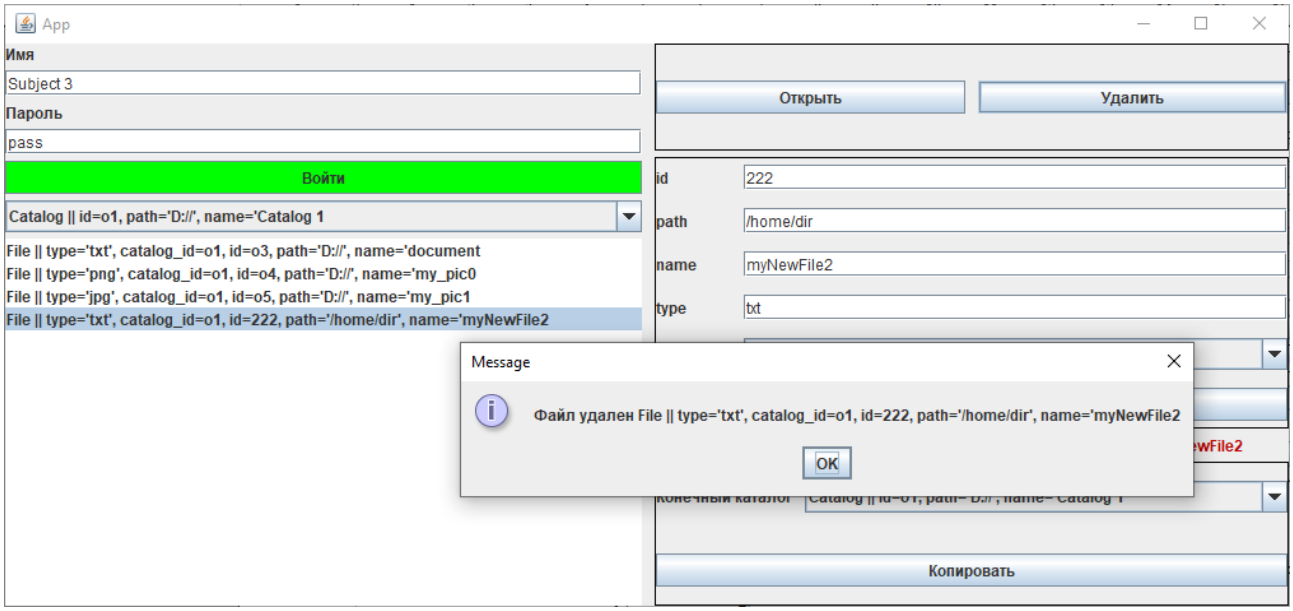

Рис. 10. Удаление файла

Для копирования файла пользователь выбирает нужный файл из списка, выбирает каталог, в который будет произведено копирование, нажимает кнопку «Копировать». Результат показан на рисунке 11.

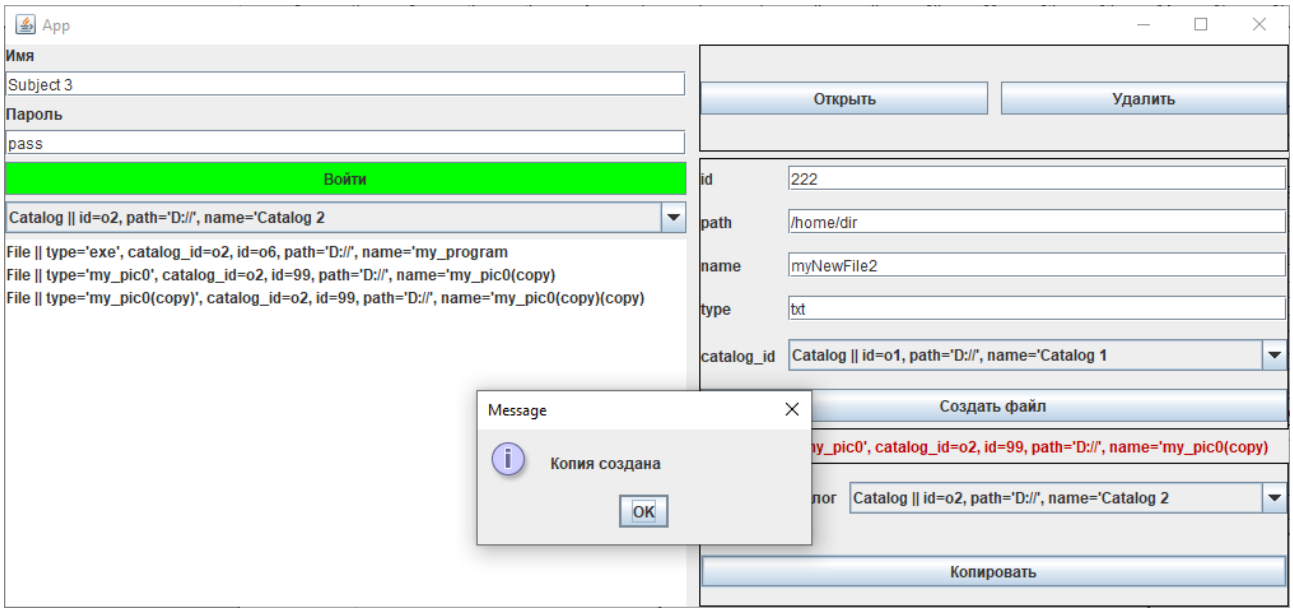

Рис. 11. Результат копирования

ЗАКЛЮЧЕНИЕ

В результате лабораторной работы я закрепил знания, касаемо мандатной системы распределения доступа, модели Белла – Лападуллы. Реализовал программу, которая отображает ее функционирование, предоставляет функции создания пользователя, выполнения основных операций с условной файловой системой. Полученная программа была проверена на конкретных примерах, результаты отображены в отчете. На основании чего можно говорить о корректности работы программы и о полноте выполнения задания.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://ru.wikipedia.org/wiki/Модель_Белла_—_Лападулы
2. https://ru.wikipedia.org/wiki/Мандатная_модель_—_Лападулы
3. Лекционные материалы

Приложение 1. Содержимое файла matrix.csv

```
subject,s1,Subject 1,pass
subject,s2,Subject 2,pass
subject,s3,Subject 3,pass
subject,s4,Subject 4,pass
subject,s5,Subject 5,pass
subject,s6,Subject 6,pass
catalog,o1,D://,Catalog 1
catalog,o2,D://,Catalog 2
file,o3,D://,txt,o1,document
file,o4,D://,png,o1,my_pic0
file,o5,D://,jpg,o1,my_pic1
file,o6,D://,exe,o2,my_program
grid,topSecret,s1
grid,topSecret,s2
grid,secret,s3
grid,confidential,s4
grid,unclassified,s5
grid,unclassified,s6
grid,secret,o1
grid,secret,o2
grid,unclassified,o3
grid,confidential,o4
grid,topSecret,o5
grid,unclassified,o6
matrix,o1,s1,r
matrix,o2,s1,r
matrix,o3,s1,r
matrix,o4,s1,rd
matrix,o5,s1,rwd
matrix,o6,s1,rd
matrix,o1,s2,r
matrix,o2,s2,r
matrix,o3,s2,r
matrix,o4,s2,r
matrix,o5,s2,rw
matrix,o6,s2,r
matrix,o1,s3,rw
matrix,o2,s3,rw
matrix,o3,s3,r
matrix,o4,s3,r
matrix,o5,s3,w
matrix,o6,s3,r
matrix,o1,s4,w
matrix,o2,s4,w
matrix,o3,s4,r
matrix,o4,s4,r
matrix,o5,s4,w
matrix,o6,s4,r
```

```
matrix,o1,s5,w
matrix,o2,s5,w
matrix,o3,s5,rw
matrix,o4,s5,w
matrix,o5,s5,w
matrix,o6,s5,rw
matrix,o1,s6,w
matrix,o2,s6,w
matrix,o3,s6,rw
matrix,o4,s6,r
matrix,o5,s6,w
matrix,o6,s6,rw
```